AUJoGR

Air University Journal of Graduate Research.

Bi-Annual Reviewed Research Journal

Volume 1 Issue 2 2021

Air University, Islamabad

Aims and Scope

Air University Journal of Graduate Research (AUJoGR) is a peer-reviewed multi-disciplinary research journal primarily for graduate students, but open to all researchers.

AUJoGR is published biannually by the Faculty of Graduate Studies, Air University. The aims of AUJoGR are to provide a platform for research scholars to publish their research for the creation and dissemination of knowledge and for building bridges among universities and research institutes.

There are no publication charges for research published in AUJoGR.

Air University is a Federally chartered public sector university, established in 2002 has its Principal Seat at the foot of the picturesque green Margalla Hills towards the north of Islamabad. The University has a campus in Multan (Punjab) and at Kamra close to the Pakistan Aeronautical Complex where a state-of-the art *Aviation City* is being developed to become functional over the next five years.

Air University conducts over 20 postgraduate programs leading to the MS and PhD degrees in Management Sciences, Engineering, Basic and Applied Sciences, Computing, Social Sciences and Strategic Studies.

Engineering programs are conducted in Aerospace, Biomedical, Computer, Electrical, Mechanical and Mechatronics Engineering.

The University hosts the National Center for Cyber Security, the Center of Excellence for CPEC (China Pakistan Economic Cooperation) and publishes the Journal of Business and Economics, Corporum: Journal of Corpus Linguistics and Erevna Journal of Linguistics and Literature.

Guidelines for Authors

Submission to AUJoGR is online. The Word Template is given on the Journal website https://portals.au.edu.pk/aujogr and at the end of this Issue.

Indexing

ISSN No.

Copyright

Contacts

Dr Zafar ullah Koreshi, Prof.
Editor AUJoGR,
Dean Graduate Studies, Air University, Islamabad, Pakistan
Tel: +92 51 9153407 Email: zafar@mail.au.edu.pk

Dr Asghari Maqsood, S.I., Prof.
Editor AUJoGR,
Dept. of Physics, Air University, Islamabad, Pakistan
Tel: +92 51 9153407 Email: asghri.maqsood@mail.au.edu.pk

Mr Mudassar Rehan
PA to Dean Graduate Studies, Air University, Islamabad, Pakistan
Tel: +92 51 9153408 Email: mudassar.rehan@mail.au.edu.pk

Mr Ijlal Aamer
Officer Graduate Studies, Air University, Islamabad, Pakistan
Tel: +92 51 9153409 Email: ijlal.aamer@mail.au.edu.pk

Types of submissions accepted in AUJoGR

1. Review Paper
2. Scientific Paper
3. Short Paper
4. Technical Paper
5. Letter to the Editor

Frequency: Twice a year

Payment: Free of cost.

PROCEEDINGS

AUJoGR-21412

SALTED CRAM BASED AUTHENTICATION MECHANISM TO SECURE COMMUNICATION BETWEEN WIRELESS SDN PLANES

Muhammad Fawad
Department of Avionics Engineering
Air University
E-9 Islamabad
Muhammadfawad10@gmail.com

Dr. Ammar Masood
Department of Avionics Engineering
Air University
E-9 Islamabad
ammarmas@yahoo.com

Abstract
The global and cohesive vision of SDN embraces more robust and simplified security logic than current applications and making it easy to address challenging network safety issues. Security implementations rely on central network controllers and flow-based technology such as algorithms for intrusion sensing or malfunction detection for the implementation of state-based flow description policies. However, it is a challenge to develop safety applications on SDN as the safety of the SDN itself remains unsure. Security problems of computationally based networks were grouped into three categories: service denial, system confidence and device insecurity. Software Defined Networking (SDN) is a modern networking architecture that overcomes conventional networking access, management and security problems. The SDN controller deals with both computing and network complications instead of intelligent computers. In this research, we are introducing a new SDN security architecture which protects privacy using a cryptographic and multifactor hash protocol. The framework for stable and improved authentication of the challenge response is then built on the basis of the proposed main foundations of the security architecture. Moreover, through automatic validation of Internet security protocol, the protection characteristics of the proposed protocol shall be formally analyzed. Furthermore, the security characteristics of the proposed protocol are formally analyzed using automated validation of internet security protocol (AVISPA), followed by informal security analysis. Lastly, performance evaluation and comparative analysis of the scheme is carried out.

Keywords: SDN; Authentication; Security.

1. INTRODUCTION

The principle of the SDN solution is that dividing network feature power from network devices themselves (speed switches, routers, firewalls, etc.) would solve some inconveniences in connection with today's vertically integrated, closed, and proprietary networking facilities. Increased the need for this improvement in network specifications with the introduction of software based virtualization technologies and integration of voice, video, and IP networking information communications. Figure 1.1 displays a standard network in an SDN deployed data center environment [1].
Figure 1.1 shows how the information flux functions in the SDN model from infrastructure layer convergence to the device plane. The control plane is positioned directly in the middle of the software logic between data and the application plane. Control plane is also used as the means of communication between the layer and device layer of infrastructure. For designing and applying the Communication Protocol Logic, OpenFlow Protocol [2] is used.

Figure 1.1: SDN Architecture

The communication protocol analyzes the patterns of traffic produced by end-users and guides the flow of information within the SDN control layer. Detaching

network access from hardware equipment eliminates the need to configure each computer individually. With a core network strategy for SDN applications, the launch time is reduced and the revenues for the data centers and services providers are increased [3]. With power isolated from network hardware, managers will adapt the behavior of the computer by forcing system program modifications instead of fork-lift enhancements. this will enhance the revenue of the data center providers.
SDN offers an abstraction layer to isolate program administrators and managers from physical hardware management. SDN virtualizes a Network Operating System (NOS) by providing access to virtual disks and memory, which abstracts the network's physical topology [4] from applications.

1.1. **SDN Control Plane** The controllers track the environment and allow controllers to combine forwarding decisions with the management of the traffic in real time, in addition to the forwarding rules for vSwitches. The controllers are interacted by three contact interfaces (usually the southbound, northbound, and eastbound interfaces) [5].

1.2. **SDN Data Plane** The data plane for SDN architecture enables data transmission from sender to recipient (s). Data plane devices themselves do not produce or acquire data, but used for communication with the controller. Data plane need to use a southbound API to connect with controllers. Two forms come of the data plane devices: Open vSwitch, software based devices and hardware-based devices such as HP switching support for OpenFlow. As can be expected, software driven devices have a more comprehensive range of features.

1.3. **SDN Security Overview** Many implementations can be found in the enterprise cloud and data center network SDN. Not only for data storage and orchestration, but also for server security, SDN will offer advantages. Therefore, the safety of SDN itself is a very important research area. The SDN operating architecture can be broken down into application, control and data layers. Every layer contains a variety of attack vectors. Additionally, layer-to-layer communication channel, for example, an application control interface may be used to modify traffic and eavesdropping attacks. However, designing security applications for SDN is challenging, as the protection of the SDN itself remains unclear. For eg, it is optional to use a protected transport layer on the network Open Flow. As a consequence of the design of the communication protocol, safety concerns like DoS, fraudulent flow rules and regulatory changes can emerge. Application vulnerability could be possible in SDN controllers (Opendaylight, ONOS, FloodLight, etc.). Security threats are also seen between northbound and southbound APIs through communications routes.

1.3.1 Application plane attacks: In telemetry, orchestration and other SDN applications vulnerabilities may arise. All security problems that can occur in a standard web application are also extended to SDN through Cross-Site Request Forgery (CSRF), like Cross-Site Scripting (XSS). Misrepresented apps propagate attacks across the network [6].

1.3.2 Attacks on the Control Plane: This control unit contains at least one controller for managing several protocols such as OpenDaylight, POX, ONOS, and other applications and plugins. Invader can generate traffic using the IP address and send large amount of traffic to the controller.

1.3.3 Data Plane Attacks: The attacker can poison the network's global view by forging the Connection Layer Discovery Protocol (LLDP) packets. If an

intruder exploits the communication channel, the application running in the control and data planes exhibits delays. This will assist in assessing the controller's logic. Attackers may also use switches to their advantage. The data flow control switch frequently contains little memory and is vulnerable to flood attack [6].

1.3.4 Transmission/Communication channels:

On the communication channel between switches, controlled computers, and controllers, a Man-in-the-Middle (MITM) assault could be launched. Previous studies have shown that communication channels can be disrupted by passive attacks while the handshake takes place between hosts. [7]

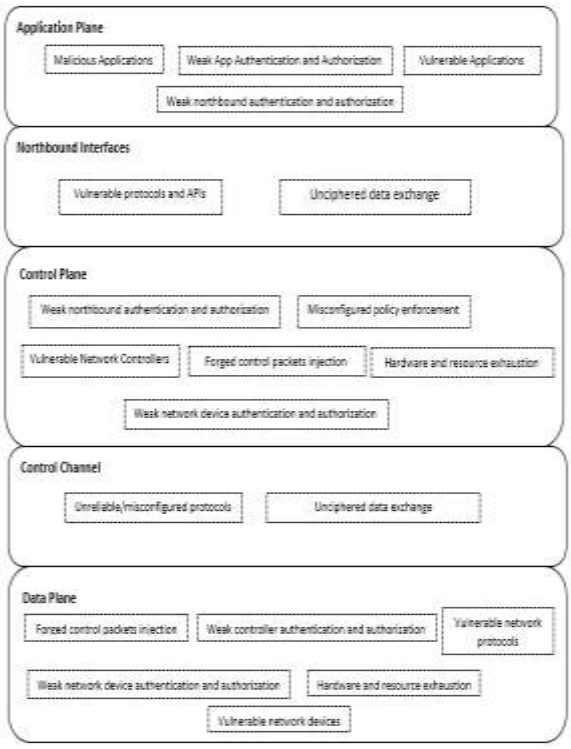

Figure 1.2: Security Threats overview in SDN [8]

A modern network horizon has opened up in the area of networking with software-defined networking by detaching control and data plane. This distinction helps network logic and decision making to be centralized in the Network Operating System (NOS) or control plane, simplifying network transmitting devices into network switches [9]. Whilst the implementation of NOS is a programming abstraction that is being used to manage the network data plane by the network application developers, the effects are severe at network security level [2]. For example, centralizing the controller adds a new weak authentication problem. Here we need a strong authentication mechanism with high performance in SDN planes. The objective of this thesis is to design a challenge response-based authentication scheme to avoid the access level attacks in SDN.

The SDN network has several threats, some of which are introduced by the inadequate procedures for authorization and authentication, others by the SDN architecture. It is necessary to address the issue of authentication mechanism between communication devices in order to establish a stable SDN networking environment. In the following paragraphs, there are several other authentication methods. We examined how useful CRAM is to compare their shortcomings.

Authentication protocols usually use cryptographic nonce to ensure that each challenge is distinct to protect against man in the middle and replay attack. If implementing a true nonce is impractical, a powerful cryptographically secure pseudorandom number of generator and cryptographic hash function may create challenges that do not come frequently. Often it is critical for time-based nonces not to be used because they can degrade servers with imprecise clocks in various time zones and servers. If the program is susceptible to a delayed usage, it may be useful to synchronize time-based nonces.

If the application is exposed to delayed message attack, it would be beneficial to use time-based nonces and synchronized clock. This attack happens when an intruder captures a transmission and blocks it from reaching the endpoint so that the caught transmission can be replayed after a pause of its choice. This can be done on wireless channels easily. Time-based nonce restrict the invader's ability to resend the message, because of its expiry time, having no effect on the application and thus diminish the attack.

Mutual authentication is conducted in both ways by means of a challenge-response handshake, ensuring the client knows the secret, and also assures that server knows the secret that guards against a bad server that impersonates actual server. Challenge–response authentication will overcome issues related to the

sharing of session keys for encryption. The challenge value and secret can be merged with a key derivation feature to produce an unpredictable session encryption key. This is especially successful against a man in the middle attack because the intruder cannot acquire session key from this challenge without understanding the secret.

To address the issue of device authentication, there must be a secure device authentication process that not only provides strong authentication but also provides privacy for credential management that is irreversible. Since most of the attack vectors occur at the time of the authentication phase. More importantly, authentication is the guardian of security tasks offering confidentiality, integrity, non-repudiation, and availability. Thus, Challenge Response Authentication Mechanism works better at this phase.

As discussed above the primary function is to authenticate the communicating devices. In this spirit, enhancement of verification procedure can be carried out that prevents and isolate adversary attack at the authentication phase. The details of our proposed CRAM technique are illustrated in figure 1.3. We used salted hash challenge response authentication mechanism that provides security against the above mentioned attacks that occur during the authentication process.

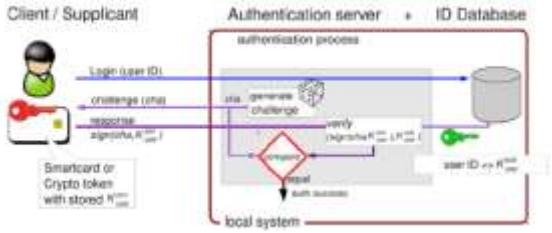

Figure 1.3: Proposed Solution

This paper is organized as follows: Section 2 briefly reviews the related work where two major previous works are studied including Secured Communication Channels in Software-Defined Networks. Section 3 highlights our proposed methodology, and implementation details. Also, we discussed the workflow of proposed scheme and simulation results. In section 4 security analysis of proposed scheme was carried out. Section 5 summarizes the thesis, with possible future study areas.

2. LITERATURE REVIEW

SDN not only beneficial for data storage and orchestration, but for server security also. Therefore, security of SDN itself is a very important research area. Security problems, such as distributed denial of services (DDoS) on SDN controllers, occurred by hierarchical nature of SDN. SDN consists of application, control, and data plane. In every plane, there are multiple attack vectors. In this section the literature review of above mentioned SDN security issues was carried out by reviewing the SDN based security challenges and their proposed solution in context of authentication schemes.

Author in [10] suggest a hierarchical authentication system for the IPv6 source address in SDN. It provides granular security assurance for IPv6-enabled devices. Work suggested in [11] offers an SDN-based 5G HetNet framework for security assurance handover verification. It simultaneously offers joint confirmation, key comprehension, and reduced verification expense among user equipment (UEs) and BSs on 5G HetNets. Author in [12] suggest trustworthy blockchain authentication (BTA) constructed on anonymous blockchain access (BAA).

Author in [13] suggest Kolmogorov-Smirnov (K-S) based authentication which is fast, efficient and increasingly suitable for layer characteristics having different structures of dissemination. A new SDN-based handover authentication was offered in [14]. The key circulation and board validation is requested for an authentication transfer module (AHM) in SDN controller.

A lightweight authentication scheme (HiAuth) is proposed in [15] to ensure the SDN controller disguises character credentials using powerful bitwise tasks. This scheme is called secret authentication. In [16] the hidden pattern (THP) is suggested which amalgamate secret diagrams and beforehand tests as an opportunity to concurrently prevent various kinds of confirmation attacks. Another MACsec introduced in [17] extends from one stage to the next the security

scope of MACsec. With no modification to existing MACsec standards, this will restrict cryptography procedures and switches.

The proposal was to use the cryptographically generated address (CGA) algorithm and a hah-generated address (HGA) algorithm to create an identity with lightweight two-way authentication (LTWA). [18]. The work in [19] offers a simple automation method that is used to validate communication via a hash, cryptographic hash and the REST API. Access is not allowed to unsuccessful applications.

The work in [20] suggests the presentation of remote signals chosen as authentication of knowledge from clients' regions. It provides the authentication of different properties.

3. CRAM ARCHITECTURAL DESIGN AND IMPLEMENTATION

CRAM is a Point-to-Point (PPP) authentication application used for verifying remote client identity. The Challenge Handshake Authentication Mechanism is an authentication scheme to protect the connectivity of the host against unauthorized entry. CRAM packets use an enquiry approach that guarantees that authenticators send a question packet twice until the answering packet responds. The performance packet will be returned if a relationship fails if the correct value for this message is calculated by hash validation by the authenticator.

3.1. **Proposed Authentication Scheme** To protect SDN, authentication mechanism was provided to secure contact between access points and applications to prevent malicious attacks by them. The authentication scheme for the device suggested as in Figure 3.1

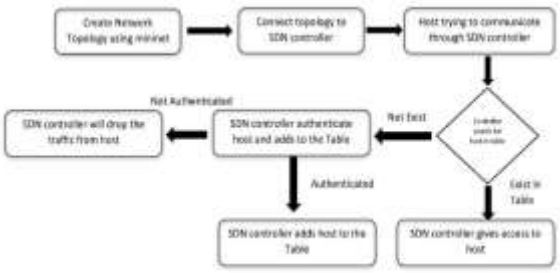

Figure 3. 1: Proposed Scheme Overview

This work offers a better authentication of the challenge-response. This authentication mechanism provides a process called the Hash Digest Multifactor Challenge/Response Chain. Factors such as Password, IPWC Index, Registration Date, Registry Index Date, Sequence Count and Session Key are used in the multifactor authentication process. The sequence counting method is a procedure in which the sequence counter begins its count from '1' between the communication devices settled on.

A predetermined common value between committed devices increases the sequence count value. The sequence count is updated to the history table in each stage of the transaction. In subsequent correspondence, the count value begins and the sequence counters are reset only until the full value is reached. The sequence counting system increases the authentication of the negotiating parties engaged.

The session key can only be created for the current session with TLS (Transport Level Security). Some aggressive and passive attacks may be prevented by that approach. The Hash Digest and XOR operations are used in this process. Adequate variables are XORed and turned into one-way hashing in the multifactor process (HF). The components of the transformed hash function are called Hash Digest (HD), an immutable message digest.

3.2 Work Flow Details of Proposed Authentication Scheme

This section includes design of proposed mechanism for authentication of access points and applications.

3.2.1 Access point and Application Registration: To store access points and application information, a hash table database is built. Access points have a single name or mac address for the controller's allocated data plane identification. In the hash table the plain identification and mac address is stored. The program is allocated the specific id and type before accessing network services. The SHA is like a signature to a document or records. The cryptographic SHA-256 hash function is used to encrypt database data. A near-unique, fixed-size 256-bit (32-byte) hash is provided by SHA-256 algorithm. It can't be decrypted back; hash is a single-way feature. That makes it suitable for password validation, and authentication as shown in Figure 3.2.

3.2.2 Credential gathering using Rest API: For authentication of the entry point, REST API receives passwords from the handler of access points. The outcome will be returned in the JSON format. The API output is then stored in the encrypted text file in the RSA and submitted for clarification to the authentication program. Application plan sends a request to authenticate the credentials before the network services are used. These passwords are used by authentication programs in the data database as seen in Figure 3.3.

3.2.3 Authentication of Access Points and Application After probing the database, it validates the data in the data base and authenticates access points and devices with the controller. If the log is not recognized or access point is incorrectly credited, it is impossible to communicate the authentication application and to disconnect access points with the controller.

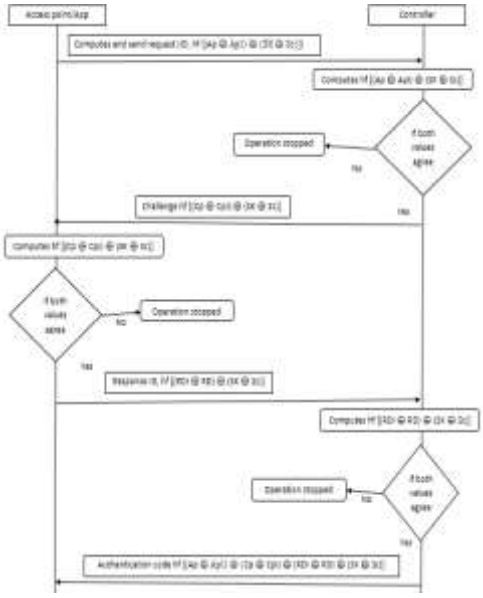

Figure 3. 2: Access Point Identification and Registration

Table 3.1: Keywords and abbreviations

Keywords	Abbreviations
Pre shared parameters before TLS	
Cp	Controller password
CpI	Controller password index
Ap	Access point password
ApI	Access point password index
RD	Registration date
RDI	Registration date index
Parameters shared after TLS	
SK	Session Key
Sc	Sequence Count

3.3 Implementation We use MATLAB to implement the proposed program. We first emulate the network topology in MATLAB using the C language to validate the authentication application. This topology is linked to the controller. After topology simulation, an authentication application is executed and results collected. The instruments for implementing the new scheme are being introduced.

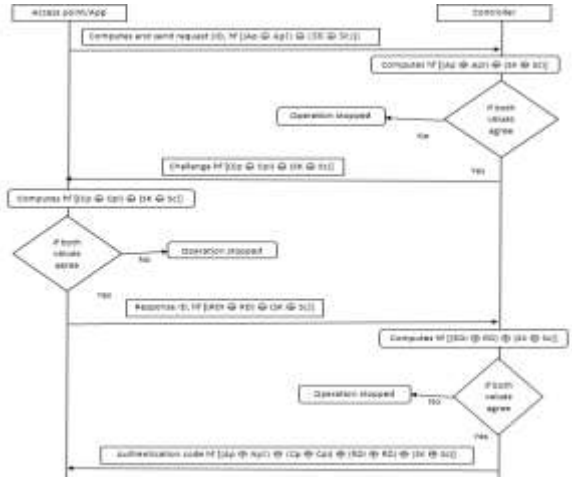

Figure 3.3: Authenticating Challenge

4. SECURITY ANALYSIS

Performance overhead and validation of proposed security protocol named CRAM is performed in that section.

4.1 Experimental Setup

This section examines and tests in depth the current authentication protocol specification outlined in Section 3. Second, to assess their performance the implementation has been checked with various appliances and platforms. The authentication protocol was tested using an automated internet security protocol validation tool (AVISPA).

4.2 Model Checking

This section models and checks the protocol for authentication. A sample tester must be used to verify the authentication protocol that guarantees that communications are genuine and confidential between the apps and the authenticating server. Thus, the AVISPA model is also used to track authenticity and confidentiality [30]. AVISPA is an automated model authentication testing device for wide-scale validation protocols. The AVISPA automation platform [30] offers four back-end authentication tools:

- OFMC
- CL-AtSe Classify the model (Constraint Logic-based Attack Searcher)
- SATMC (SAT-based Model-Checker)
- TA4SP (Tree-based model checker)

The protocol has been modeled on HLPSL to verify AVISPA for the CRAM. The High Level Protocol Specification Language (HLPSL) is used for the creation and description of safety protocols. HLPSL uses Alice and Bob Notation [31] to model security protocols. The authentication protocol using HLPSL is seen in Figure 4.1. Two protection goals occur in the AVISPA. In order to verify that the networks are authenticated, the following objectives have been set:

- User-side verification
- Server authentication via honesty of the challenge. In addition, the following objective is defined to decide if the authentication correspondence is kept secret.
- Secrecy of hashed message

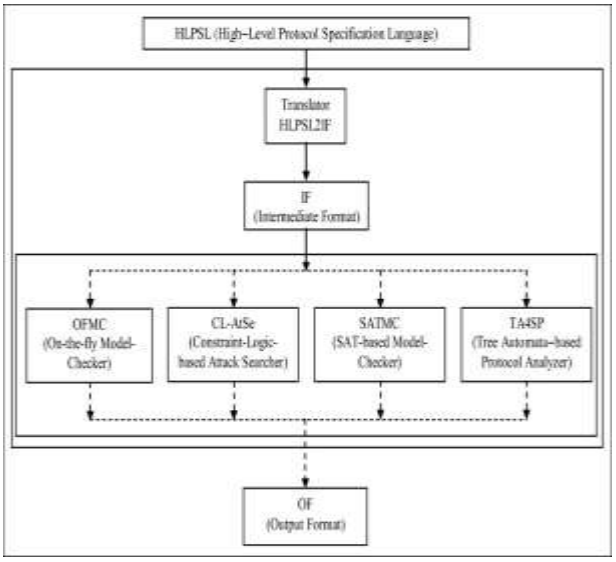

Figure 4.1: Authentication Protocol Modelled Using HLPSL

4.3 Results

Automatic validation of the specified protocol was carried out through the AVISPA model checker. To locate the limited number of sessions and falsifying protocols, CL-AtSe (Constraint Logic-based Attack Searcher) and SATMC (SAT-based Model-Checker) backend methods were used [47]. CL-AtSe completed the validation in six seconds. In contrast, it took 285.13 seconds for SATMC to verify. Therefore, on both sides, there have been no possible attempts on the protocol. In order to detect assumptions and replay attacks, backend OFMC test the Protocol.

4.4 Informal Security Analysis of the Proposed Mechanism

We concentrate on proving that our projected scheme can guarantee a range of essential safety parameters that are of considerable signification in SDN, such as authentication, confidentiality, stable key exchange, and privacy and replay attacks.

4.4.1 Password Guessing: An attacker cannot conjecture a password because the author has created two passwords, Controller Password and Access Point Password (AP), along with their respective CPI and API password indices, pre-shared between the controller and the access point. The attacker needs to conjecture these two passwords along with their index values while attempting to guess the answer. Service (ID, hf[(API:SC))].

4.4.2 Spoofing: Likewise, a deceiving access point cannot be an intruder as a master. For when you attempt a spoof controller, you have to reply by Challenge hf [(CPI umpCP) supplement (SK umpSC))] to the access point. The attacker needs to know all the mutual values, session key and sequence number that are established before computational challenges.

4.4.3 Replay Attacks: An attacker or eavesdropper can intercept any authentication steps such as requests, challenges, and responses between the controller and access point from previous observations during such attacks and overwrite them. If the Eve attempts to replay the request, the controller will detect it by contrasting the obtained request with the calculation application if it is using the phase UID, hf[(API TodayAP)], to the controller.

4.4.4 Modification Attack: This is a form of attack that an attacker can attempt to change the content of any authentication measures. This software avoids modifications because at each stage of the authentication it tests the component it gets from another hand, any improvements made in any step of authentication are quickly detected and further authentication steps are stopped automatically.

4.5 Performance Analysis

For performance monitoring, the program developed to authenticate multiple numbers of access points including 5, 10, 20, 50, and 100 records the time in milliseconds. The next step is to verify that the software itself authenticates. There is a notification and the app can be executed.

Table 4.1, 4.2 and 4.3 numerically represent the time taken to obtain the data while figure 4.2 ,4.3 and 4.4 graphically reflects the time consumed. As the program has an immediate response time, it enables a malicious system or application to quickly take effective action by an administrator.

It also lists illegal APs, including information such as ID names. The APs that have the wrong passwords (unauthenticated APs) will also be shown. APs are also mentioned.

Table 4.1: Number of Access points

Number of AP	**Time in ms**
5	23
10	35
15	40
20	50

Table 4.2: Number of SDN APP

Number of SDN APP	**Time in ms**
5	**15**
10	**21**
15	**27**
20	**37**

Table 4.3: Number of Hosts

Number of hosts	**Time in ms**
5	29
10	37
15	48
20	61

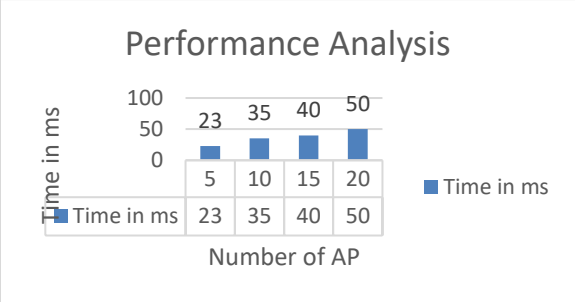

Figure 4.2: Performance Analysis for Access point authentication

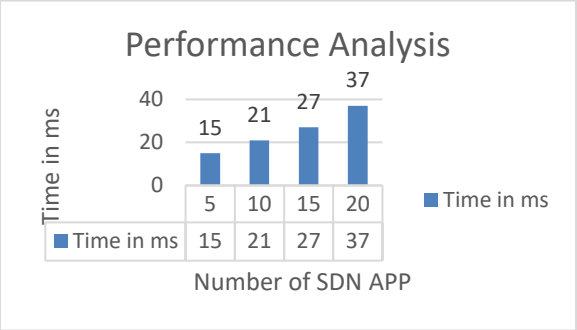

Figure 4.3 Performance Analysis for authentication of applications

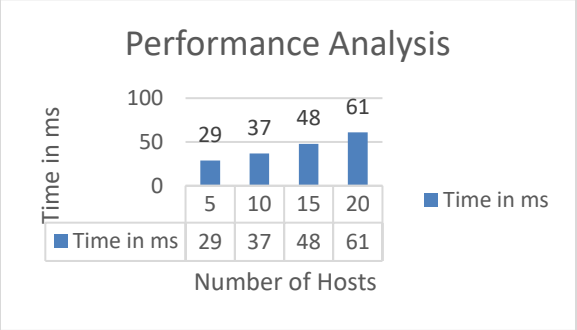

Figure 4.4 Performance Analysis for host authentication

5. CONCLUSION

The reliability of corporate network relies on successful access management systems and host authentication. As the business networks widely implement Software Defined Networking (SDN), the task of supplying SDN with security is becoming more significant.

The proposed process allows coordination between controller and APs to be secured by a mechanism to authenticate the solution to lightweight challenges. We plan on using our prototype also to validate its functionality and intensity in a true network environment against active attackers. We have rendered the informal and formal review of proposed scheme using AVISPA tool in order to demonstrate that our proposed scheme is stable.

Results also demonstrate that our proposed CRAM technique is more powerful than others because it provides less control overhead and facilitates the specification of flow access control policies in compliance with host access credentials.

We plan to incorporate CRAM as an authentication method and its default access control in Future Internet Testbed (FITS) in the future. In order to use signed certifications for access certificates we also plan to expand the CRAM for modern authentication mechanisms such as the EAP-TLS.

ACKNOWLEDGEMENTS

REFERENCES

[1] Ihsan H. Abdulqadder ; Deqing Zou ; Israa T. Aziz ; Bin Yuan “Enhanced Attack Aware Security Provisioning Scheme in SDN/NFV Enabled over 5G Network” 2018 27th International Conference on Computer Communication and Networks (ICCCN) Year: 2018 | Conference Paper | Publisher: IEEE.

[2] Beytullah Yigit, Gurkan Gur, Bernhard Tellenbach, Fatih Alagoz . “Secured Communication Channels in Software-Defined Networks” IEEE Communication Magazine | Oct, 2019 | Publisher: IEEE.

[3] Jin Won Kang, Sae Hyong Park, Jaeho You. “Mynah: Enabling Lightweight Data Plane Authentication for SDN Controllers”. 24th International Conference on Computer Communication and Networks (ICCCN) |Aug, 2015 | Publisher: IEEE.

[4] Ferrazani Mattos, D.M., Duarte, O.C.M.B. AuthFlow: authentication and access control mechanism for software defined networking. Ann. Telecommun. 71, 607–615 (2016) | Journal Paper | Publisher: Annals of Telecommunications

[5] Rajat Chaudhary ; Gagangeet Singh Aujla ; Sahil Garg ; Neeraj Kumar ; Joel J. P. C. Rodrigues “SDN-Enabled Multi-Attribute-Based Secure Communication for Smart Grid in IIoT Environment” IEEE Transactions on Industrial Informatics Year: 2018 | Volume: 14, Issue: 6 | Journal Article | Publisher: IEEE.

[6] Reem Melki ; Ali Hussein ; Ali Chehab "Enhancing Multipath TCP Security through Software Defined Networking" 2019 Sixth International Conference on Software Defined Systems (SDS) Year: 2019 | Conference Paper | Publisher: IEEE.

[7] Jingjing Xu ; Hanshu Hong ; Guofeng Lin ; Zhixin Sun "A New Inter-Domain Information Sharing Smart System Based on ABSES in SDN" IEEE Access Year: 2018 | Volume: 6 | Journal Article | Publisher: IEEE.

[8] Sugandhi Midha ; Khushboo Triptahi "Extended TLS security and Defensive Algorithm in OpenFlow SDN" 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence) Year: 2019 | Conference Paper | Publisher: IEEE.

[9] Yanbing Liu ; Yao Kuang ; Yunpeng Xiao ; Guangxia Xu "SDN-Based Data Transfer Security for Internet of Things" IEEE Internet of Things Journal Year: 2018 | Volume: 5, Issue: 1 | Journal Article | Publisher: IEEE.

[10] Xiao Liang ; Heyao Chen "A SDN-Based Hierarchical Authentication Mechanism for IPv6 Address" 2019 IEEE International Conference on Intelligence and Security Informatics (ISI) Year: 2019 | Conference Paper | Publisher: IEEE.

[11] Jin Cao ; Maode Ma ; Yulong Fu ; Hui Li ; Yinghui Zhang "CPPHA: Capability-based Privacy-Protection Handover Authentication Mechanism for SDN-based 5G HetNets" IEEE Transactions on Dependable and Secure Computing Year: 2019 | Early Access Article | Publisher: IEEE.

[12] Hui Yang ; Haowei Zheng ; Jie Zhang ; Yizhen Wu ; Young Lee ; Yuefeng Ji "Blockchain-based trusted authentication in cloud radio over fiber network for 5G" 2017 16th International Conference on Optical Communications and Networks (ICOCN) Year: 2017 | Conference Paper | Publisher: IEEE.

[13] Ting Ma ; Feng Hu ; Maode Ma "Fast and efficient physical layer authentication for 5G HetNet handover" 2017 27th International Telecommunication Networks and Applications Conference (ITNAC) Year: 2017 | Conference Paper | Publisher: IEEE.

[14] Cong Wang ; Yiying Zhang ; Xi Chen ; Kun Liang ; Zhiwei Wang "SDN-Based Handover Authentication Scheme for Mobile Edge Computing in Cyber-Physical Systems" IEEE Internet of Things Journal Year: 2019 | Volume: 6, Issue: 5 | Journal Article | Publisher: IEEE.

[15] Osamah Ibrahiem Abdullaziz ; Li-Chun Wang ; Yu-Jia Chen "HiAuth: Hidden Authentication for Protecting Software Defined Networks" IEEE Transactions on Network and Service Management Year: 2019 | Volume: 16, Issue: 2 | Journal Article | Publisher: IEEE.

[16] Liming Fang ; Yang Li ; Xinyu Yun ; Zhenyu Wen ; Shouling Ji ; Weizhi Meng ; Zehong Cao ; M.Tanveer "THP: A Novel Authentication Scheme to Prevent Multiple Attacks in SDN-based IoT Network" IEEE Internet of Things Journal Year: 2019 | Early Access Article | Publisher: IEEE.

[17] Ju-Ho Choi ; Sung-Gi Min ; Youn-Hee Han "MACsec Extension over Software-Defined Networks for in-Vehicle Secure Communication" 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN) Year: 2018 | Conference Paper | Publisher: IEEE.

[18] Yongli Tang ; Tao Liu ; Xu He ; Jinxia Yu ; Panke Qin "A Lightweight Two-Way Authentication Scheme Between Communication Nodes for Software Defined Optical Access Network" IEEE Access Year: 2019 | Volume: 7 | Journal Article | Publisher: IEEE.

[19] Yongli Tang ; Tao Liu ; Xu He ; Jinxia Yu ; Panke Qin "A Lightweight Two-Way Authentication Scheme Between Communication Nodes for Software Defined

Optical Access Network" IEEE Access Year: 2019 | Volume: 7 | Journal Article | Publisher: IEEE.

[20] Ke Ding ; Xiulei Wang ; Guomin Zhang ; Zhen Wang ; "A flow-based authentication handover mechanism for multi-domain SDN mobility environment" Ming Chen China Communications Year: 2017 | Volume: 14, Issue: 9 | Magazine Article | Publisher: IEEE

AUJoGR-21415

FACEBOOK MESSENGER APPLICATION FORENSICS ANALYSIS (ANDROID)

Muhammad Abdullah Siraj
Department of Avionics Engineering
Air University,
PAF Complex E-9 Islamabad
malik.abdullah185@gmail.com

Aqsa Naheed
Department of Avionics Engineering
Air University,
PAF Complex E-9 Islamabad
Aqsaa.naheed@gmail.com

Dr. Muhammad Qasim Saeed
Department of Avionics Engineering
Air University,
PAF Complex E-9 Islamabad
qasim51@yahoo.com

Abstract

Today in the world of technology Facebook is leading the market with a maximum number of user on its Messenger, Whatsapp, and Instagram. According to the latest studies, the monthly active user of Facebook Messenger is about 1300 Million as of October 2020. With this leading number of users, cybercrimes are also increasing, people use social media platforms to communicate and carry out criminal activities like planning for some illegal activity, abusive content sharing, etc. In this research, we try to find out the data present on a criminal phone after the arrest of the criminal. Many of them delete their data from the phone and it is hard to extract that kind of data, but we try to recover as much data that we extract from the suspect device and present it in the court of law using two forensics tools. After inspecting tests show that we cannot extract Facebook Messenger data with logical acquisition from the device. Physical acquisition is used for Facebook Messenger data extraction. The results are discussed below.

Keywords: Facebook Messenger; Magnet Axiom; MOBILedit; Rooted Device; Non-Rooted Device; Account Information; Conversation Details; Deleted Media.

1. INTRODUCTION

As of now, social media clients are getting speedier, one of them is Facebook Messenger, Facebook Messenger, a social media application, which positions moment, as it were to Whatsapp, is exceptionally prevalent. The increment within the number of Facebook Messenger clients certainly has an impact of great and awful, one of the awful impacts is that a few individuals who utilize Facebook Messenger commit advanced violations. If the smartphone is proving in criminal cases and Facebook Messenger is introduced on a smartphone, Figure 1-1 is the foremost downloaded social media application chart within the Android Play store application.
The fast development in utilization and application of Social Networking (SN) stages make them a potential target by cybercriminals to conduct noxious exercises such as personality burglary, robbery, illicit exchanging, sexual badgering, cyberstalking, and cyber-terrorism. Numerous SN stages are expanding their administrations to portable stages, making them an imperative source of proving in cyber examination cases. Hence, understanding the sorts of potential prove of users' SN exercises accessible on versatile gadgets is pivotal to legal examination and investigation. In this research, we look at SN applications: Facebook Messenger. We distinguish an assortment of artifacts (e.g. usernames, passwords, and login data, individual data, transferred posts, traded messages, and transferred comments from SN applications) that seem to facilitate a criminal examination. [1]

This paper is organized as follows: Section 2 of this paper will be describing the techniques of Mobile Forensics, and then in section 3, we will set up the working environment and analyze the digital evidence on Android-based Facebook Messenger in section 4, in section 5 discuss the proposed tool functionality and their limitation and we will conclude the research in this section.

2. TECHNIQUE

This chapter contains a detailed literature review regarding mobile forensics, its types, and Facebook messenger forensics. This literature review aims to gather as much information as possible about the topic that has been done before. It will be an overview of the forensics, and tools used for Mobile forensics which help us in gathering information after an illegal activity occurs. The aim is to compare the performance of forensics tools and the technique and get as much data from the mobile device.

a. Forensics

Mobile forensics is a subset of digital forensics, the retrieval of data from an electronic source. Specifically, mobile forensics deals with recovery evidence from mobile devices such as smartphones and tablets.

Mobile devices can provide all types of important data, ranging from call logs and text messages to web search history and location data that shows where the device owner might have been at a given time.

b. Working of a Mobile Forensics: Crimes do now not manifest in isolation from technological tendencies; consequently, mobile device forensics has to turn out to be a large part of virtual forensics. The cell forensics method pursuits to get better digital evidence or relevant facts from a mobile tool in a way with a purpose to maintain the proof in a forensically sound circumstance. To gain that, the mobile forensic system needs to set out unique guidelines with a purpose to capture, isolate, transport, keep for evaluation, and evidence virtual evidence properly originating from cell devices.

Figure 2-1 Working of a Mobile Forensics

3. Literature Review

In Forensic Benchmarking for Android Messenger Applications, the author describes forensics as, "Digital Forensics is recuperating, investigating and identifying the statistics to show the lifestyles of proof. Mobile forensics is a sub-branch of digital forensics. It isn't feasible to apply the equal technique entirely because of the distinctiveness of the investigations in every area. But, we can list the stairs as follows: a) identification and investigation, b) acquisition, c) evaluation." [2]

Innovation has reshaped the style within which we tend to accompany the globe and access information with the approach of cell phones. Fitly, the wants we've and also the answers for our necessities have likewise modified aboard the developing innovation. One of the foremost influenced matters from innovation is correspondence. With the various alternatives and skills, texting applications are begun to use for correspondence reason that is probably the best want of individual. We can send instant messages, video messages, voice chronicles, and provide areas utilizing these applications. considerably additional, we tend to presently do not need phone calls by GSM directors, and rather favor these applications for moment calls, even as conveyance non-public information to those applications, not only for individual each day life, likewise for business want. Then again, these applications bring probabilities with various benefits. One of them is security. We do not want these applications will store our data as its client. [3]

The zoom-in use and utilization of Social Networking (SN) stages make them a potential objective by digital hoodlums to lead noxious exercises like misrepresentation, theft, prohibit business, provocation, digital following, and digital fear-based oppressor act. A few tin-stages territory units stretching out their administrations to portable stages, making them a significant stock of verification in digital examination cases. In this manner, understanding the sorts of likely confirmation of clients' tin exercises possible on cell phones is essential to logical examination and investigation. During this research, we will in general look at Facebook, Twitter, Linked In and Google+, on mechanical man and iOS stages, to locate leftovers of clients' exercises that zone unit of expository interest. we will in general sight a scope of ancient rarities (for example usernames, passwords, login information, individual information, transferred posts, changed

messages, and transferred remarks from tin applications) that may encourage a criminal examination. [4]

4. Environment Setup

Following OS and tools are used for implementation and testing:

- Windows 10
- MOBILedit
- Magnet Axiom
- Samsung Galaxy S6 edge
- Samsung
- Redmi Note 9s

3.1. Windows 10

Window 10 is used as a host software on which we run all our forensics software.

3.2 MOBILedit

MOBILedit Forensic Express is a phone and cloud extractor, facts analyzer, and file generator all in one solution. A powerful 64-bit application the use of each the physical and logical data acquisition strategies, MOBILedit is extraordinary for its superior software analyzer, deleted information recuperation, live updates, and wide range of supported telephones which includes maximum characteristic telephones, first-rate-tuned reports, concurrent phone processing, and clean-to-use user interface. **[5] [6]**

Figure 3-2 MOBILedit

3.3 Magnet Axiom

It recovers digital evidence from most sources, including smartphones, cloud services, computers, IoT devices, and third-party images. Use powerful and intuitive Analytics tools to easily analyze all evidence data in one case file.

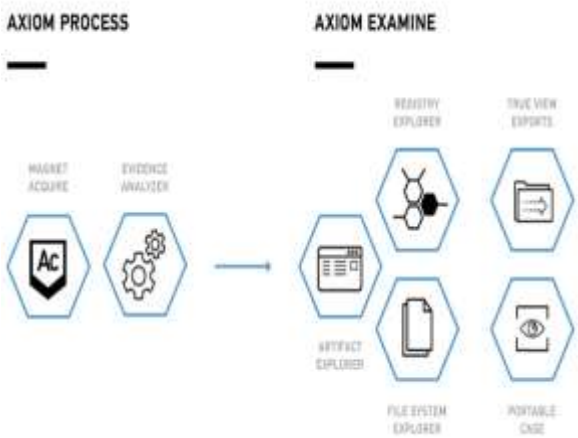

Figure 3-3 Magnet Axiom

5. Analysis

This chapter includes a detailed description of the analysis of the proposed tools, the workflow of the proposed tools is explained in section 4.1 followed by section 4.2 which will discuss the results obtained from these tools on a rooted and non-rooted android device.

4.1 Work Flow

NIST methodology for evidence is used to gather the evidence from an Android smartphone using MOBILedit and Magnet Axiom. Below is the NIST methodology for acquiring the evidence.

- Pick out, accumulate and defend facts related to a selected occasion
- Process the accrued data and extract applicable portions of information from it
- Analyze the extracted data to derive additional beneficial information
- Record the outcomes of the evaluation.

Figure 4-1 Steps of Forensic Methodology

First of all, we have to turn on the developer option in the smartphone, then we have to make changes in the developer mode like "Stay Awake" and the "USB Debugging" option should be on. Connect the device with the computer and start the procedure for the acquisition of the data from the device.

Firstly, we will examine the results of the non-rooted device on both tools.

4.2 MOBILedit (Non-Rooted)

MOBILedit extracts the data of all the applications on that device but our main concern is to analyze the data of Facebook Messenger. On a non-rooted device, MOBILedit only shows that messenger is installed on it, its version, application permission at the time of installation, last update of the application but it didn't show any messages, chat, or any other media file. Because the data about Facebook Messenger is placed in com.facebook.orca package but this folder in the smartphone didn't have permission to view, all of the files in this folder are hidden which can only be accessed by using the root level access of the smartphone.

Figure 4-2 Analysis of Messenger Application

4.3 Magnet Axiom (Non-Rooted)

AXIOM Process allows **you to add keyword lists, hash lists, whitelist non-relevant files, or build custom artifacts** to help customize your search and assist when handling large sets of data. Once you have added your evidence, selected your artifacts, and set your additional options, you can begin the processing.

Figure 4-3 Non-Rooted Redmi Phone Analysis

Figure 4-4 Com.facebook.orca Installed on Phone

4.4 MOBILedit (Rooted)

For the rooted device, we use Samsung Galaxy 6 edge and extract data from the smartphone. After testing we get data about Facebook Messenger from the package "com.facebook.orca" which include all the account detail, application detail, conversation, media files, deleted media files and messages. Below is the screenshot of the package extracted on a rooted device.

Figure 4-5 MOBILedit deleted Conversation User data

Figure 4-6 Media Files on Messenger

4.5 Magnet Axiom (Rooted)

For data extraction from a rooted device we use Samsung Galaxy Alpha and run magnet axiom on it, we get Messenger conversation as well as deleted and non-deleted media files.

Figure 4-7 Magnet Forensics Media Files

Figure 4-8 Magnet Forensics Messenger Messages

4.6 Android Database

We explore the database which we get through the physical acquisition of the android smartphone and we carry out different tests on the extracted database of the Facebook Messenger application and we get different tables having messages thread, users, groups, and different key tables which are used during the communication over Facebook Messenger.

Figure 4-9 Authentication Key for Facebook Messenger

The above image shows the user id, user name, Hashed Message authentication code (hmac), session key, access token, cookies string, and timestamp for the expiry of the access token. Below is the image for the hexadecimal value for the authentication.

Figure 4-20 Hex-editor screenshot for Authentication

The following image shows the details of the current user logged-in on that particular smartphone-like name, email id, date of birth, phone number if connected, current location prediction, and the gender of the user.

Figure 4-11 Logged-In User Information

The following image shows the keys used for authentication and encryption etc. This file contains the information of Master key, Identity Key, Public key, and Private Key. This also contains the information on the application installed on the phone.

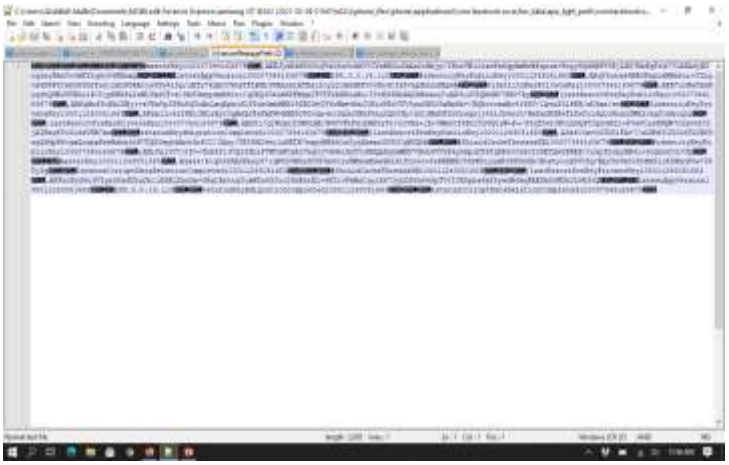

Figure 4-12 Secure Message Preferences

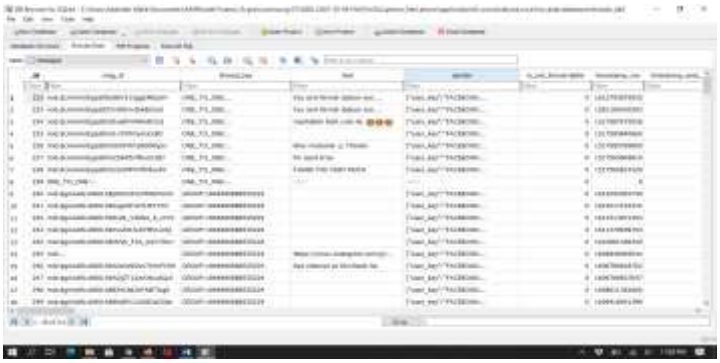

Figure 4-13 Messages Table

The above image shows the message table extracted from the acquisition it contains the messages which that user share with other users, its column shows message-id, thread key means is it a one to one message or shared in a group. The body of the message, sender information timestamp in 'ms'. If some chat is deleted it shows null in all the columns of that conversation. The deleted chat assigned a -1 message type which shows that this message is deleted. Images, videos, audios, and documents have different message type by which an examiner clearly distinguish between the messages from the database.

6. CONCLUSIONS

All the tool we tested, extracts data only from the rooted device for Facebook Messenger because its data file is hidden and it doesn't have privileges for the user to access them directly. For this purpose, we carry out many tests like we try app downgrade so that we may have some version that doesn't have such security privileges. One other thing we try to find an application for an android device that has root access and we extract data for Facebook Messenger but there is no such application that has this option. We carry out a one-time root methodology but all the devices they support are out of the market. There was a vulnerability of android which was currently patched that allow one-time root access it was a security patch of 2016 vulnerability on all the devices but now all the devices are upgraded to security patch 2020. This section sum up our research, as well as we, give some proposed solution we encounter during our research.

JTag is a tool that extracts root-level android data without rooting the device. It is a chip-off methodology for data extraction, we need to disassemble the android device and connect it to JTag and extract all the root level data from it.

XRY is a forensic tool, in research, the researcher performed explain that the use of the forensic tool XRY to perform the forensic analysis in the rooted smartphone and Non-Rooted smartphone is better. However, considering the funding or other restrictions, we can try to use the forensic tool, whose crack version is available because free tools are not up to mark to extract Facebook Messenger data. [5]

One of the solutions is we have to make an android .apk which has root-level access and it requests a folder to extract hidden data available in it which a user cannot access without root.

Although instant messaging software has the benefits of simplicity and immediacy, it is still unavoidable that cybercriminals will misuse it. Crimes are also exploited by vulnerabilities of apps, blogs, and online applications, using cloud resources to distribute ransomware, and further manipulating posts and connections from social media to bait users into fraud traps.

When investigating cybercrime on Facebook Messenger, we suggest that finding the culprit's account number, alias and network number should be the priority. The operating practices of criminal suspects on social networks can be used under account numbers and nicknames, such as posting photos, texting, calling, and time stamping.

The results from our research are that we can only extract data from Facebook Messenger in android only when the device is rooted. The amount of data extracted from this method is enough for evidence in the case of cybercrime.

References

[1] A. &. R. I. &. A. I. Yudhana, "Identification of Digital Evidence Facebook Messenger on Mobile Phone With National Institute of Standards Technology (NIST) Method," *Kursor*, vol. 09, no. 03, pp. 33-40, 2018.

[2] B. P. Lovanshi M., "Comparative Study of Digital Forensic Tools," *Data, Engineering and Applications*, pp. 195-204, 2019.

[3] S. D. A. K. A. J. L. a. K. V. Roberto Hoyle, "Was my message read? Privacy and Signaling on Facebook Messenger," *In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). Association for Computing Machinery, New York, NY, USA*, pp. 3838-3842, 2017.

[4] D. A. C. K.-K. M. Z. Yang TY, "Windows Instant Messaging App Forensics: Facebook and Skype as Case Studies.," *Pone*, vol. 11, no. 03, 2016.

[5] A. D. B. E.-S. &. K.-K. Farhan Norouzizadch Dezfouli, "Investigating Social Networking applications on smartphones detecting Facebook, Twitter, LinkedIn and Google+

artedacts on Android and iOS platform," *Australian Journal of Forensics Sciences*, 2016.

[6] "Mobiledit Forensic Express," Compelson Lab, 1991. [Online]. Available: https://www.mobiledit.com/forensic-express. [Accessed 2020].

AUJoGR-21416

ONTOLOGY BASED EMAIL FRAUD DETECTION USING HYBRID MACHINE LEARNING

Ahsan Ali
Department of Avionics & Aeronautics
Air University
Islamabad
ahsanalisam59@gmail.com

Dr. Mansoor Ahmed
Department of Avionics & Aeronautics
Air University
Islamabad
mansoorkhan75@gmail.com

Dr. Qasim Saeed
Department of Avionics & Aeronautics
Air University
Islamabad
qasim51@yahoo,com

Dr. Moneeb Gohar
Department of Computer Science
Bahria University
Islamabad
mgohar.buic@bahria.edu.pk

Dr. Adnan Fazil
Department of Avionics & Aeronautics
Air University
Islamabad
adnan.fazil@mail.au.edu.pk

Abstract
Emails are inevitable in this modern era. Spammers write junk/ unwanted email messages about target's interests primarily to earn money out of it and therefore, employ creative and developed methods. Some famous techniques are employed by fraudster to commit email fraud, leading to personal information and financial losses incurred/ faced by innocent targets. Therefore, identifying spam and email fraud is a widely researched topic. Resultantly, to overcome existing gaps and improve already developed systems, this research work presents a solution devised on the concept of ontology-based email fraud detection. Moreover, ontology-based email fraud detection is enhanced using hybrid machine learning, incorporating two class decision forest learning algorithm to detect spam/ fraudulent emails with more accuracy. The ontology based technique detect new types of fraudulent emails with 99.2% accuracy as compared to conventional techniques. Experiments performed using the proposed solution show promising results with greater accuracy.

Keywords: email; fraud detection; ml; decision forest.

1. INTRODUCTION

Email is an easy access platform for many groups, individuals, and organizations and it became an official mechanism for communication and knowledge sharing. It is widely used due to its reliability, convenient and ease of use. According to the definition of Wikipedia [1] spam is an irrelevant, undesired message that is sent intentionally to a large number of audience over the medium of internet for advertising, spreading malware, and phishing. Email spamming is referred to as distributing unwanted messages, mostly sent in bulk using email known as "spam" whereas those emails which are antithetical are known as "ham"/useful emails [2]. The word "Spam" came from canned precooked meat that was marketed in the year 1937 named as "Shoulder Pork HAM" which was taken by digital mail junks [3]. To overcome the issue of spam emails, spam filtering techniques are introduced. Many machine learning classifiers were used to classify ham and spam.

An Email that seems to be ham to one user may appear as spam to other user. It depends on user preference. For example, all emails related to cryptocurrency will be considered as ham by the user who are interested and at the same time it appear as spam to the user who are not interested. So, based on that ontology based spam filtering techniques were introduced. In this paper we compare old system with the new proposed model and test the accuracy on the selected data gathered over the internet through reliable source.

2. LITERATURE SURVEY

As we have worked on ontology-based email spam detection system using hybrid machine learning, it's relevant to review some historical work in the relevant field. As per a study in [4] there were over four billion email accounts were active in the year 2020; this calculation was projected to be 4.48 billion

approximately by the end of year 2024. This census clearly shows that half of the population in the entire world is using email. Electronic mail creates a lot of ease in our life for communication and interacting with a large audience with minimum effort and time, but on the other hand, it is prone to multiple threats. The most common and the popular threat is spam/ email fraud. Due to widespread use of email, 57.26% of accounts were used for spam in 2019 alone. A report released by Proof point [5] back in 2019 reported that advanced attackers don't exploit system flaws, they exploit human flaws. As a result, fraud has increased 86% in comparison to the previous year. There are 9.9 million accounts are identified as automated spam accounts [6]. Spam is a major concern in recent times especially in the COVID-19 which affects more than 80% of users. So, it is crucial to detect spam messages which may lead to privacy breach and fraud. FBI reported in 2019 that businesses around the globe bear a loss of around 12.5 USD billion due to spam and phishing [7]. The stats in Table 1-1 were reported by Statista for the year 2017 to 2020. They reported the top 3 scams in Australia (Investment scams Dating scams and Employment Scam), their frequency, and the total loss [8].

Year	Loss	Scams	Frequency
2017	**$90801407**	**Investment $31,326,476** **Dating $20,530,578** **Employment $5,270,948**	**Phishing 26,386** **Identity Theft 15,703** **False billing 13,455**
2018	**$107001471**	**Investment $38 846 635** **Dating $24 648 024** **False Billing $5512502**	**Phishing 24,291** **Threats to Life 19,455** **Identity Theft 12,800**
2019	**$142898217**	**Investment $61,813,801** **Dating $28,606,215** **False Billing $10,110,753**	**Phishing 25,170** **Threats to Life 13,375** **Identity Theft 11,373**
2020	**$52971358**	**Investment $20,650,486** **Dating $14,708,686** **False Billing $4,378,559**	**Phishing 10,689** **Threats to Life 4 255** **Identity Theft 4,237**

Table 1 Top 3 Scams Reported in Australia 2017-2020

The most recent spam traffic volumes for the last 180 days detected by AVTEST [9], an independent IT-Security institute, are depicted in Table 2-2. They reported top 10 countries which were massively hit by spammers and also indicate the origin of those spam emails.

Origin	Spam Traffic
Vereinigte Staaten von Amerika	43.3%
Russische Föderation	9.4%
Deutschland	7.0%
Frankreich	3.0%
China	2.8%
Bulgarien	2.1%
Ukraine	2.0%
Brasilien	1.9%
Spanien	1.6%
Großbritannien	1.5%

Table 2 Spam emails reported worldwide during COVID

During the research, we noticed that the content and operational mechanisms have changed over time and the techniques which are currently implemented may not be applicable in the coming years. This phenomenon is known as conceptual drift [10]. There are many spam detection mechanisms that have been presented so far and tested but the reported accuracy still pleads that more in-depth work in this domain is required to achieve desirable accuracy. Authors in [11] practice artificial neural networks for spam detection and achieve 86 percent accuracy. [12] Authors use the Naive Bayes approach with the incorporation of cost-sensitive and multi-objective genetic programming for feature extraction and they achieve only 79.3% accuracy. In [13] authors develop spam detection mechanisms using interval type 2 fuzzy sets and achieve an accuracy of 86.9% only. Authors [14] use a genetic algorithm-based hybrid, on the dataset used in the above-mentioned paper to hit the accuracy of 90%. In [15] authors achieve an accuracy of 91.22% on the testing set by using a negative selection based algorithm along with particle swarm optimization.

Developed Approaches	Accuracy (%)
Naïve Bayes (feature extraction)	79.3%
Artificial Neural Networks	86%
Interval type 2 fuzzy sets	86.9%
Genetic algorithm based approach	90%
Negative selection algorithm	91.22%

Table 3 Developed approaches and their accuracy

Yong Hu et.al [16] highlights several problems of available anti-spamming techniques. He presented a framework called intelligent Hybrid Spam filtering Framework that identifies spam messages by evaluating feature's in the header. Selected features are "originator field", "X-mailer", "destination field", "mail subject" and "sender server IP address". The reason for analyzing the email header is to improve currently implemented anti-spamming techniques. Christina et al. conducted a detailed study on modern spam filtering techniques including DAB (distributed adaptive blacklists), RBF (rule-based filtering), Bayesian classifier, KNN (K nearest neighbor), SVM (support vector machine), content-based spam filtering, artificial immune system from all of these most of the techniques are truly based on text categorization and they also suggest there is still a lot of scope for classifying text and multimedia messages with more advanced techniques which can claim 0% false positive and false negative ratio[17]. In [18] authors presented a method for spam classification using Bayesian approach. Their work considers only email body, which means content of email with domain featuring. Their work showed that accuracy can be achieved with more domain-specific knowledge by using automated domain knowledge discovery. In [19] researchers proposed a cost-sensitive model which reduces the error rate of miss classifying the ham and spam emails by using three-way filtering (Bayesian, thresholds, probability). The authors sparse the data into 2 sets first one is training and the second is testing with 80% to 20% ratio. The datasets were collected from UCI repository, Ling-spam Corporation, and PUI Corporation. After testing the accuracy was 89.88%, 93.94%, and 86.35%. In [20] a new case-based reasoning method is presented for spam detection. They use the instance selection approach for handling concept drift, which in the first step remove noisy cases and in the second step, redundant cases will be removed. In 2009 Wanli Ma et al. conducted research on email spam detection and created a detection engine with zero spam knowledge and a negative selection mechanism with antigen feedback. They use TREC07 dataset and Nilsimsa digest to represent emails in 256 bits. The results are encouraging and provide an opportunity to pursue this work by implementing different algorithms [21]. In [22] author's main idea is to classify an email by using parameters that are frequently used by spammers. Their model work on a predefined email list using that list model can allow or block senders from sending spam emails. They group all the important emails and block spam emails by using another group. Their machine learning model uses parameters (To, From, Message-ID, Cc/Bcc and body text) which were matched using keywords and punctuations. The system accuracy depends on the size of a dataset as the dataset grows, higher accuracy will be achieved. In 2013, Dhanalakshmi R and Chellapan C proposed a spam detection system that considers host lexical features, domain age, and page rank to classify URL as malicious or legitimate. They used a Bayesian classifier to reduce feature sets and use a phishtank dataset that has more specific parameters to improve system accuracy. Their work is limited to URL checking [23]. A considerable and well-researched paper has been published for detecting spam using content-based approach [24, 25]. Their work is limited to feature extracting but alone with this parameter we can't detect all types of spam mails because spammers are well aware of the advanced spam detecting systems so they well mixed the content and bypass the system. The behavior-based mechanism was developed by authors Hamid and Abawajy [26] To detect phishing/spam emails by using hybrid feature selection, they use four classifiers to mine the sender's behaviour to find whether a source is legitimate or not. So, their work is limited with checking unique sender and domain and they used data set of 3000 data points which were reported as valid or invalid sender. In [27] Almeida developed a procedure of aggrandizing short texts that are found in email spams. The author argued it is very hard to implement any sort of classification algorithm on a short text that contains abbreviations and idioms. So, the author proposed a solution based

on semantic dictionaries, which generate attributes that will be feedstuff in any classification algorithm but the researchers suggest improvements with thorough testing and measuring performance.

Developed Systems	Limitations
Intelligent Hybrid Spam Filtering Framework	Result rely on trusted servers address
Spam Filtering Techniques (DAB, RBF, KNN, SVM)	Unable to prevent spam from misusing network bandwidth
Method for automated construction of filters	Keywords matching using limited domain knowledge
Cost sensitive 3 way filtering model	Classification accuracy depends only on rough set model not on available datasets
Case-based Reasoning	Due to rebuild of case base and updating of cases periodically error rate is high
Negative selection method using antigen feedback	Results are based on assumptions due to nil spam email knowledge
Three-tier client server architecture	Accuracy depends on dataset volume and predefined email list
URL analyser (using lexical features)	Work is limited to URL checking
Feature extraction using language analysis	Work is limited to feature selection
Hybrid Feature Selection	Only check for unique sender and domain
Semantic dictionaries	Classification performance depend on limited set of attributes

Table 4 Developed systems and their limitations

3. PROPOSED MODEL

The gleaned paper is based on the intent of this research attempt. We read several research articles based on the listed index terms and thoroughly analyzed different presented methods to check whether they implemented machine learning. We also reviewed the degree of adaptation required to address which the solution may exhibit. In [28] the work was a significant impact using a conceptual approach by integrating ontology along with that system demands improvement by deeply understanding the model so this technique highlights the working of email fraud detection frameworks and their diversity in research directions.

3.1 Dataset Collection

Datasets are collection of instances that are very vital for any smart learning system. They share a common attribute for the training and evaluation; the more you feed, the better you get output from the system. As the dataset size grows, machine learning system performs faster, learns in a very short time and improves system accuracy. The datasets used in our system for training and evaluation were collected from different resources that are mentioned below.

3.2 UCI

This is a dataset repository for machine learning and intelligent systems A sample named as SMS spam collection data set were taken from this site the sample we used was created back in 2012 that is publically available for research work. The dataset characteristics are multivariate text and have domain theory. Attribute characteristics are real and the tasks performed on the dataset were classification and clustering. This sample contains 5574 instances with null missing value. The authors of this dataset are Tiago A. Almeida and MarÃa GÃ³mez Hidalgo from Department of Optenet Las Rozas, Madrid – Spain.

3.3 Grumble Text

This is a UK based forum on which smart phone users report spam messages and they carefully investigate the reported spam messages so that they can create a dataset of real spam messages and contribute in this domain. So, from those reported and classified messages we use a sample of 425 messages which was collected manually by grumble text few years back.

3.4 Caroline Tag's

A dataset named as Caroline tag's was taken from a PhD thesis which was created by Caroline Tagg. This dataset contains only 450 ham messages that were used for the thesis work. The author mentioned in her thesis that these messages came from anonymous public forum named as AOL which was discontinued

but the messages are still very helpful. This dataset contains real ham messages which were forwarded to the users at that time from different spammers.

3.5 Spam Corpus v.0.1 Big

A dataset that contain labelled messages which was collected for spam research. It contains 1319 messages in English, tagged as ham and spam. Out of total messages 1002 messages are ham & 322 are spam that was collected from Spam Corporation, dataset version 0.1. They incorporated the following researches [29] [30] [31] for the dataset creation. This corpus collection actually consists of two files one is SMS Spam Corpus v.0.1 Small that consist of 1002 ham messages and 82 spam messages whereas SMS Spam Corpus v.0.1 Big consist of 1002 ham messages and 322 spam messages. We use Corpus v.0.1 Big as it contain more spam messages.

3.6 SMS_SPAM

This dataset was taken from a project Machine Learning with R datasets that have 4879 messages labelled as spam messages. Zach Stednick created this dataset in 2014. These messages have real data which was collected by the author and the dataset is very helpful for the training of model with a large number of dataset.

Data sets	Total Instances	Classification (Spam/Ham)
UCI	5574	Spam (700) + Ham (4874)
Grumble text	425	Spam
Caroline's Tag	450	Ham
Spam Corpus v.0.1 Big	1324	Spam (322) + Ham (1002)
SMS_Spam	4879	Spam

Table 5 Datasets total instances and classification type

3.7 Pre-processing of Text

Pre-processing is performed on the text to make it meaningful. Following operations were executed before processed it to the training model:-

- Remove all the stop-words.
- Use regular expressions in the dataset to replace target strings.
- Lemmatization function convert multiple words to a single canonical form.
- Filter parts of speech.
- Case normalization from upper to lower.
- Remove certain classes such as repeated characters, numbers and special characters.
- Identify and remove emails and URLs.

After preprocessing, we selected columns in dataset for training and also exclude those which will not be part of the training model. After that, we processed metadata, which was not required most of the times.

3.8 Feature Hashing

Feature-hashing converts unique tokens into integers means pick lexions, roles and their frequency. FH is used for the implementation of ontological concept on which the whole paper is based. Ontology is a concept for generating lexions mean (special words and their role).

USERTEXT	SENTIMENT
I loved this novel	3
This novel was great	3
I hated this novel	1
I love novels	2

Table 6 Feature Hashing for Ontological mapping

FH creates a data dictionary of n-grams along with calculating bigrams frequency. N-gram feature helps to model the entire content and use unigrams and bigrams in n-gram feature sets.

TERM (bigrams)	Frequency
I loved	1
This novel	3
I hated	1
I love	1

Table 7 Generating role & features along with frequency

After creating data dictionary, FH converts dictionary terms into conceptual model using values of hash, and compute features used in each case. After hashing ontology model might appear like this:

Rating	Hashing feature 1	Hashing feature 2	Hashing feature 3
4	1	1	0
5	0	0	0

Table 7 Conceptual model based on hashing

3.9 Filter Based Feature Selection

A common problem faced during the implementation of machine learning is determining whether the features which were selected for prediction are relevant to our model or not. To overcome this issue and filter out irrelevant and redundant columns from our model, we used filter-based selection which inherits statistical measure called featuring scoring method that calculates the score for each feature column and returns the scores based on ranking. To improve the system accuracy and efficiency of the classification, we choose the best feature scoring method that best fits my data. In filter-based selection only the columns with the best scores use to build the predictive model and the columns with poor scores can be left out in the dataset and ignored while building a model. We use Chi-Squared method for filter-based selection.

3.10 Two Class Decision Forest

We used binary classification algorithm to compute results from a possible set of outcomes. For supervised learning we have datasets which we already discussed with 50% ham and 50% spam messages.

The reason for using Two Class Decision Forest algorithm is:-

- It can perform well on little data as well.
- It is very fast and supervised ensemble model.
- It take least time among other algorithms for training with highest accuracy.

Algorithms	Training Time (in seconds)
Two Class Decision Forest	14
Multi Class Decision Forest	35
One Class Support Vector	42

Table 8 Algorithms with training time

4. RESULTS

This chapter give the detailed overview of the statistical results. This also helps to find out whether our proposed system is rejected or accepted and also predicts which approach is better in terms of accuracy. All the results and the stats shown in the diagrams that follow are based on real data that is publicly available. The experiment was conducted using Azure machine learning studio which is ML based platform for developers to train and test their models. To evaluate our work, we use five types of the datasets which can contain spam and ham (50% ratio for both types) messages with classification.

4.1 Evaluation Measures

The results generated during this research work are based on the evaluation of Confusion Matrix. This means a “true positive” is a spam message that is truly detected and classified by the model and likewise “false positive” is a non-spam message that is wrongly classified as spam message by the model.

Actual	Prediction	
	Spam Emails (Positive)	Legitimate Emails (Negative)
Spam Emails (Positive)	True Positive	False Negative
Legitimate Emails (Negative)	False Positive	True Negative

Table 9 Confusion Matrix

Results in the diagram below show the quantitative statistics and also presents abutment of my work.

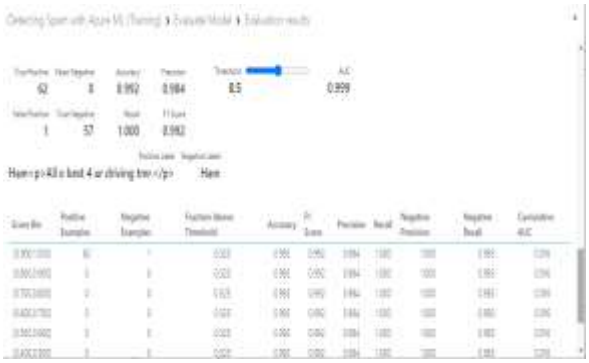

Figure 1 New Model Using ML Confusion Matrix Results

Figure 2 Old Model without ML Confusion Matrix Results

These stats clearly shows that our proposed ontology model based on machine learning(two class decision forest) achieved the highest value 0.992 out of 1 means 99.2 % accuracy, 100 % recall, 98.4 % precision, and 99.2 % F1 score. Whereas, the previous ontology based model without machine learning achieved 77.4% accuracy, 65.4% precision, 100% recall and 79.1% F1 score.

Model	Accuracy	Precision	Recall	F1 Score	AUC
OLD	77.4%	65.4%	100%	79.1%	92.9%
NEW	99.2%	98.4%	100%	99.2%	99.9%

Table 10 Comparison of models

Our proposed model outperforms by detecting true positives and true negatives with preeminent accuracy with only 0.8% false positive ratio. In comparison, the previous work based on ontology-based email fraud detection model have 23.6% false positive ratio. Results clearly establish that our model is a significant improvement, is viable and achieves better accuracy by integrating ML approach.Therefore, it is concluded that the ontological research work detects spam emails with more accuracy and the false positives and the negatives were mitigated using Hybrid Machine Learning approach in combination.

Approach	Accuracy	Testing Type	Total Instances
OLD	77.4 %	Batch	200
NEW	99.2 %	Batch	6541

Table 11 Batch Testing Results

5. CONCLUSIONS

This research work has been performed with the context of proposing an intelligent system that uses an ontological-based concept for email spam detection. Our proposed technique achieve higher accuracy of 99.2% as compared to some related techniques which was developed using ontological concept like ontology based spam filtering [32] and spongy technique [33]. They only achieve highest accuracy of 94.74% and 91% respectively. So, on these statistics our model is a contribution in this domain by outperforms the related works.

6. FUTURE WORK

Future work recommendation includes considering an Artificial Neural Network that works on unsupervised data using case-based reasoning. Case is a piece of knowledge representing an experience and also domain specific knowledge at operational level. In this concept, neural networks extract the symbolic rules and represent the rules as a case-based reasoning method which will understand the new problems using the previous knowledge. ANN system learns with minimum available data to cope up with new problems by modifying the training rules by itself with Artificial Intelligence. Case specification is defined as set of features used for the solution so to use only most relevant features knowledge acquisition method involves scenarios for the rule learning and apply best principles for the targeted case. As this approach use the best possible case for the solution but the search time is much more due to searching in the case base so to overcome the searching time case retrieval nets can be used which are memory structures that are flexible and efficient in retrieval of cases. Case based reasoning solves the problem by re-using or adapting the approaches that were used earlier to solve the

similar problem. Re-using can be done by diagnosis and implementing the solution with higher feature selection rate by updating the base score to use this case for the similar approaches. The past problem or experience encoded as case which contain feature characteristic of problem and their solution. Case based reasoning divides the process into sub-processes:-

- Retrieve related case from case base.
- Reuse the already used and better approach.
- Incremental changes done if the solution require revision.
- Retain the solution for future use.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Mansoor Ahmed, who continuously provided me with suggestions and motivated me in achieving my goal. It's all because of your support and coordination from the beginning till the end. It's all because of you. Thank you!

REFERENCES

[1] Youn, S., 2014. SPONGY (SPam ONtoloGY): Email classification using two-level dynamic ontology. The Scientific World Journal, 2014.

[2] O. Saad, A. Darwish, and R. Faraj, ''A survey of machine learning techniques for Spam filtering,'' Int. J. Comput. Sci. Netw. Secur., vol. 12, no. 2, p. 66, Feb. 2012

[3] M. K. Paswan, P. S. Bala, and G. Aghila, ''Spam filtering: Comparative analysis of filtering techniques,'' in Proc. Int. Conf. Adv. Eng., Sci. Manage. (ICAESM), Mar. 2012, pp. 170–176

[4] "Number of e-mail users worldwide from 2017 to 2024." [Online]. Available: https://www.statista.com/statistics/255080/number-of-e-mailusers-worldwide/

[5] Proofpoint. (2019, 0219-058). Winter 2019, Protecting People a Quarterly Analysis of Highly Targeted Cyber Attacks [Report]. Available: https://www.proofpoint.com/us/resources/threat-reports/quarterly-threatanalysis

[6] Y. Roth and D. Harvey. (2018, 26 June). How Twitter is fighting spam and malicious automation. Available: https://blog.twitter.com/en us/topics/company/2018/how-twitter-is-fightingspam-and-malicious-automation.html

[7] M.Guntrip, "https://www.proofpoint.com/us/corporateblog/post/fbi-reports-125-billion-global-financial-lossesdue-business-email-compromise."

[8] "Global spam volume as percentage of total e-mail traffic from January 2014 to September 2019, by month." https://www.statista.com/statistics/420391/spam-email-traffic-share/.

[9] https://www.av-test.org/en/statistics/spam/,

[10] K. Jackowski, B. Krawczyk, and M. Wozniak, "Application of adaptive ´ splitting and selection classifier to the spam filtering problem," Cybern. Syst. An Int. J.

[11] F. G. Levent Özgür, Tunga Güngör and F. Gürgen, "Spam Mail Detection Using Artificial Neural Network and Bayesian Filter," pp. 505–510, 2004.

[12] Y. Zhang, H. Li, M. Niranjan, and P. Rockett, "Applying Cost-Sensitive Multiobjective Genetic Programming to Feature Extraction for Spam Email Filtering," Springer Berlin Heidelberg, 2008, pp. 325–336.

[13] R. Ariaeinejad and A. Sadeghian, "Spam detection system: A new approach based on interval type-2 fuzzy sets," in 2011 24th Canadian Conference on Electrical and Computer Engineering(CCECE), 2011, pp. 000379–000384.

[14] F. Temitayo, O. Stephen, and A. Abimbola, "Hybrid GA-SVM for Efficient Feature Selection in E-mail Classification," ISSN, vol. 3, no. 3, pp. 2222–1719, 2012.

[15] I. Idris and A. Selamat, "Improved email spam detection model with negative selection algorithm and particle swarm optimization," Appl. Soft Comput., vol. 22, no. September 2014, pp. 11–27, Sep. 2014.

[16] Hu, Y., Guo, C., Ngai, E. W. T., Liu, M., & Chen, S. (2010). A scalable intelligent non-content-based spamfiltering framework. Expert Systems with Applications, 37(12), pp. 8557-8565

[17] Christina, V., S. Karpagavalli, and G. Suganya. "A study on email spam filtering techniques." *International Journal of Computer Applications* 12, no. 1 (2010): 0975-8887.

[18] Sahami, Mehran, Susan Dumais, David Heckerman, and Eric Horvitz. "A Bayesian approach to filtering junk e-mail." In Learning for Text Categorization: Papers from the 1998 workshop, vol. 62, pp. 98-105. 1998.

[19] B. Zhou, Y. Yao, and J. Luo, "Cost-sensitive threeway email spam filtering," J. Intell. Inf. Syst., vol. 42, no. 1, pp. 19–45, 2014

[20] S.J. Delany, P. Cunningham, A. Tsymbal, and L. And Coyle, "A casebased technique for tracking concept drift in spam filtering," Knowledge-Based Systems, p. 187-195, 2005

[21] Ma, Wanli, Dat Tran, and Dharmendra Sharma. "A novel spam email detection system based on negative selection." In *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, pp. 987-992. IEEE, 2009.

[22] Alurkar, Aakash Atul, Sourabh Bharat Ranade, Shreeya Vijay Joshi, Siddhesh Sanjay Ranade, Piyush A. Sonewar, Parikshit N. Mahalle, and Arvind V. Deshpande. "A proposed data science approach for email spam classification using machine learning techniques." In *2017 Internet of Things Business Models, Users, and Networks*, pp. 1-5. IEEE, 2017.

[23] Dhanalakshmi Ranganayakulu and Chellappan C., "Detecting malicious URLs in E-Mail - An implementation", AASRl Conference on intelligent Systems and Control, Vol. 4, 2013,pg. 125-131

[24] J. Martinez-Romo and L. Araujo, "Detecting malicious tweets in trending topics using a statistical analysis of language," Expert Systems with Applications, vol. 40, pp. 2992-3000, Jun 2013 2013.

[25] M. Alfifi and J. Caverlee, "Badly Evolved? Exploring Long-Surviving Suspicious Users on Twitter," in International Conference on Social Informatics, Cham, 2017, pp. 218-233.

[26] I. R. A. Hamid and J. Abawajy, ''Phishing Email feature selection approach,'' in Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun., Nov. 2011, pp. 916–921

[27] T. A. Almeida, T. P. Silva, I. Santos, and J. M. G. Hidalgo, ''Text normalization and semantic indexing to enhance instant messaging and SMS spam filtering,'' Knowl.-Based Syst., vol. 108, pp. 25–32, Sep. 2016

[28] Kerremans, K., Tang, Y., Temmerman, R. and Zhao, G., 2005, December. Towards ontology-based e-mail fraud detection. In *2005 portuguese conference on artificial intelligence* (pp. 106-111). IEEE.

[29] Gómez Hidalgo, J.M., Bringas, G.C., Sánz, E.P. and García, F.C., 2006, October. Content based SMS spam filtering. In *Proceedings of the 2006 ACM symposium on Document engineering* (pp. 107-114).

[30] Almeida, T., Hidalgo, J.M.G. and Silva, T.P., 2013. Towards sms spam filtering: Results under a new dataset. *International Journal of Information Security Science*, *2*(1), pp.1-18.

[31] Almeida, T., Hidalgo, J.M.G. and Silva, T.P., 2013. Towards sms spam filtering: Results under a new dataset. *International Journal of Information Security Science*, *2*(1), pp.1-18.

[32] Shajideen, N.M. and Bindu, V., 2018, July. Conventional and Ontology Based Spam Filtering. In 2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR) (pp. 1-3). IEEE.

AUJoGR-21417

COMPARITIVE ANALYSIS OF OPEN-SOURCE INTRUSION DETECTION SYSTEMS AND SUGGESTED ENHANCEMENT

Muhammad Zubaid Khalil
Department of IAA
Air University
Islamabad
zubaidkhalil@gmail.com

Dr Ammar Masood
Department of IAA
Air University
Islamabad
ammarmas@yahoo.com

Abstract

Internet technology has made significant changes in our lives. With the passage of time, internet has become more advanced with the inclusion of numerous efficient and user-friendly services along with corresponding increase in the underlying bandwidth. Demand for securing the ever-growing data network is becoming important to ensure that user security and privacy concerns are effectively mitigated. In this regard, Intrusion Detection System (IDS) is widely used for network monitoring and response to overcome the network security issues. IDS performance is highly impacted by factors such as the volume of network traffic, the number of traffic flows, the packet capturing and packet matching technique employed by IDS, and the network throughput in which the IDS is being deployed. In the given research, we are mainly concerned with open-source IDSs: Suricata, Snort and, Zeek along with their security testing and comparative performance analysis. We analyzed various factors including packet drop rate, usage of system resources, detection accuracy and packet processing which can limit the applicability of any IDS solution in an organization's network. Moreover, we also considered a wide-ranging performance and security analysis via different configurations, flows and, attacks to review the security and performance impact on the tested solutions. Through our research, we concluded that the reviewed IDSs are not optimally configured by default. Moreover, performance improvements can be achieved through better configurations. Thus, our work is expected to be very beneficial to IDS developers and network administrators as it will help them in selecting optimal configurations for enhanced security.

Keywords: IDPS; NSM; HIDS; NIDS; FPR; FNR.

1. INTRODUCTION

As technology evolves and enterprise networks expand, the visibility over the IT infrastructure and network decreases. Networks are more prone to threats that compromise the confidentiality, Integrity and Availability of data flowing through it. Adversaries and various threat actors tend to be more inclined towards networks that do not have appropriate counter measures in place. An insecure network most likely leads to the compromise of critical services and systems present on that network. In order to increase the overall visibility and security and to critically analyze what traffic flows in and out of the network, different techniques need to be incorporated in enterprises and campus networks.

The prevalent network security measures include the use of access control, firewalls, intrusion detection and prevention systems etc. [4]. The firewalls and access controls method are not sufficient to overcome security issues. In contrast, Intrusion Detection System (IDS) is widely used for network monitoring and response to effectively manage the network security issues. IDS is used to detect unauthorized activity in the network and to generate alerts to the network administrator. In this way, a network administrator has better visibility of traffic flow on the network which ultimately enables him to carry out timely detection of malicious traffic. The network administrator not only analyses the incoming and outgoing traffic in order to detect the unauthorized access to the network but also creates rules on different network intrusion devices for appropriate action to be taken if any abnormality is found in the traffic.

While there are number of commercial IDS systems such as McAfee NSP, Cisco Firepower NGIPS, Palo Alto Networks Threat Prevention etc., but the recurring cost of such systems severely limit their usage in organizations with limited resources. Thus, open-

source IDS appears to be a natural choice for normal users and even for Small and Medium Enterprises (SMEs). Some commonly used open-source network security monitoring tools include Snort, Suricata and Zeek [22].

Since network-based attacks have become more advanced and sophisticated, an effective IDS solution is indispensable for network monitoring and security. However, the performance of an IDS depends on a number of factors which if not configured correctly may lead to limited utilization of the system and in the worst case can even lead to denial of services by the system itself [3]. There are two facets of IDS effectiveness in a network, the performance evaluation in terms of various parameters such as bandwidth utilization, time utilization during detection, CPU and memory utilization and the security evaluation normally considered through a number of factors including False Positives, Accuracy and Detection [2, 8].

Prior work regarding open-source IDS evaluation has only considered performance issues with limited research on security features [3]. Moreover, none of the previous works have focused on Network Monitoring Mode which is supported by Zeek IDS. In contrast, our research is focused on both performance and security evaluation of current well-known open-source IDS solutions which include Snort, Zeek & Suricata. Our main contributions are: -

- Parametric performance evaluation of all 3 IDSs has been conducted in Network Security Monitoring mode which is also known as Sniffing mode.
- Performance Analysis in IDS-mode (10K rules for Snort & Suricata)
- Evaluation of security effectiveness of Snort and Suricata where attacks have been mapped under the NSS labs testing methodology

The rest of this article is organized as follows: Section 2 summarize the previous work on open-source IDS solutions regarding performance. In section 3, we present foundations and brief overview of three popular open-source IDSs. Section 4 describes our evaluation strategy, testing environment and use case scenarios. We discuss experimental results in section 5 along with a consolidated analysis. Section 6 concludes the article provides recommendations and highlights research directions for future work.

2. RELATED WORK

Since the development of the first Intrusion Detection System in 1984, IDSs have been used to identify any malicious activities in networks and the users connected to them. Although top IDSs such as Snort, Suricata, and Zeek help to effectively detect any potential threats on regular networks, there are limitations in the efficiency and effectiveness of these IDSs on currently employed networks. There are two primary factors on which IDS performance depends: the first factor is detection and minimization of risks that impact the performance of IDSs, while the other involves upgrading the overall performance of Intrusion Detection Systems [15, 2]. The performance of IDS can be affected by a myriad of different factors including the number of network flows, IDS configurations, flow duration etc. [15, 1, 11]. According to prior research, different OS and platforms yield different results on the performance of IDSs [15, 1]. Snort performance on different operating systems concluded that Snort IDS performed well in open-source Linux environment as compared to other operating systems [15]. The effectiveness of IDSs was greatly affected by varying duration of network flows and different kinds of network flows [11]. When dealing with large volumes of data, matching data packets with regular expression causes extreme stress to the system's resources which ultimately results in bottlenecking [16]. Hence, it reduces the performance of the IDS. Both Snort and Suricata use similar expression patterns to identify malicious attacks. Hence, both enhance the load on the system in such cases. Pattern-matching method employed by Snort consumes almost 70% of the processing time of the IDS [8]. After reviewing the studies mentioned above, it was observed that memory usage, CPU usage, and packet drop rates of IDSs could be influenced by different environments. By studying the existing research on IDS performance, we concluded that although much attention has been given to IDS Performance of Snort and Suricata on networks giving 1 Gb/s,2 Gb/s, and even 20 Gb/s throughput, there was no significant research done on any other IDS with different packet capturing mechanisms. Moreover, no significant research is done on security evaluation of open source IDSs. Through our work we highlighted the limitations of IDSs in networks, and proposed adjustments and suggestions on how to optimize the performance of IDSs in such networks.

Through our research we have analyzed key challenges that affect IDS performance in high-speed networks. We investigated these challenges on a network producing 1000Mb/s throughput. It was found that almost six cores will be needed to maintain optimal IDS performance results. If the IDS is not provided with sufficient resources, then this would prove detrimental to the system considering that many malicious threats may not be detected. For a network, even employing two common packet capturing techniques simultaneously will result in packet loss. It has been proven in previous studies that if even a very minute number of packets is missed by the packet capturing mechanisms, it will lead to catastrophic damage to the IDS's performance [21, 22].

3. FOUNDATIONS / BACKGROUND

3.1. Intrusion detection System (IDS)

Intrusion Detection System (IDS) is used for monitoring network for malicious activities or violation of security policies. A variety of open-source software like Snort, OSSEC, and Suricata are employed in several organization for network monitoring [22]. IDS is used to detect unauthorized / malicious activity in the network and generate alerts to the network administrator. IDSs can be categorized into two types: Host based IDS and Network based IDS [21].

3.2. Host Intrusion Detection System (HIDS)

HIDS is type of IDS which monitors the traffic on individual systems. It does not examine the whole network rather it checks for any malicious or unusual activity on the endpoint [18]. Management of HIDS is difficult as each host has to be managed individually. Major disadvantage of this IDS is excessive resource consumption during its processing.

3.3. Network Intrusion Detection System (NIDS)

NIDS is crucial for monitoring of the network. NIDSs are usually passive and can be deployed into existing networks with little disruption to normal network operations [23]. The advantage of NIDS is that it examines each and very packet which flows in or out of the network to find threats to the security of network. An IDS does not block or prevent attacks; they merely help to uncover them. Because of this, an IDS needs to be part of a comprehensive plan that includes other security measures and staff who know how to react appropriately. Some forms of attack are not easily discerned by NIDSs, specifically those involving fragmented packets.

NIDS complements HIDS to enforce detection both signature-based and anomaly-based and acts as a safeguard to monitor traffic going to and from an organization's network. Intrusion detection can be very expensive, so we selected the most significant open-source and free intrusion detection systems to protect the network.

3.4. Open-source intrusion detection tools

In this section, we discuss different open-source intrusion detection tools such as Snort, Suricata and Zeek. Aforementioned tools are widely used by organization to monitor and detect traffic that is either malicious or malformed [13]. We discuss these tools with respect to their operation and performance in order to get an idea on how these tools perform differently. Other architectural components of the IDS that includes packet capturing methods and intrusion detection mechanism will be discussed in the upcoming sections. The study carried out in the previous sections discussed on how IDS performance is affected by different packet capturing and intrusion detection mechanisms [3]. Our main objective is to understand the technologies being used in these IDS tool and test each of them using suitable configuration. In order to understand the IDS performance, it is very necessary to under the mechanism and operation involved in IDS operation.

3.5. Snort

Snort is an open-source Intrusion Detection System. Commonly, Snort is referred to as an Intrusion Prevention System (IPS) as well [24]. It is written in C programming language and was developed in 1998. Snort is a free network-based software which is primarily used as a packet sniffer to monitor systems and networks in real time. Snort is relatively simpler to use and this type of IDS can be operated in any of the operating systems available (Windows, Linux, and Mac etc.) due to its cross-platform support. Snort is highly acclaimed and easily customizable across different platforms. Snort allows network admins to add their own rules and signatures to the IDS and immediately add these new signatures to the intrusion detection process.

3.6. Suricata

Suricata is a free open-source network security program that can perform highly efficient functions such

as intrusion detection, intrusion prevention, and network security monitoring [23]. It is a very commonly used IDS whose functions are considerably similar to that of Snort. The current stable version of Suricata is 5.0 which is capable of allowing network admins to add their own protocols into the IDS system. Network admins can configure Suricata specifically according to their own requirements and needs. While the features of Suricata are very similar to Snort's functionalities, there are few key distinctions between the two systems. Suricata uses the multithreaded architecture, which enables it to operate more effectively in higher traffic volumes as compared to Snort. Suricata supports hashing and file extraction functionalities and allows for Lua scripting language (which helps the program to edit the output and design more complex detection logics).

3.7. Zeek (Bro)

Zeek, formerly known as Bro, is an open-source software framework that is used to analyze network traffic, detect any anomalies in the network, and detect malicious threats to a system (and the networks connected to the system). Zeek's functions have certain similarities with other popular IDSs such as Snort and Suricata. However, Zeek carries out other sophisticated functions more than simply detecting network intrusions.

Snort and Suricata are traditional IDS/IPS which do some deep packet inspection and then apply signatures on the traffic in order to detect attacks. Zeek does not claim to be an IDS; instead, it claims to be a network monitor and traffic analyzer. Zeek is therefore only used to capture the details of the traffic and forward these to some analysis system. Zeek allows network admins to perform functions such as incident response, data forensics, hashing, file extraction, automatic logging, along with other useful operations. Zeek allows its users to convert data related to network traffic into events then interpret those events. It provides a script interpreter which is a programming language that helps interpret the events (created from network traffic) and understand how these events affected security of the network. While it is running in the system, Zeek logs all activities in the network automatically. These logs include information relevant to network security such as: connection records, volume of packets sent/received, and other useful meta-data. One of the reasons why Zeek is more practical than other conventional IDSs is that the programming language integrated in it is readily customizable. In some cases, network activities that may be deemed harmless for one organization, can be considered malicious threats to another organization. For such cases, Zeek allows the network admin to create exceptions. Zeek can also be used to compute network statistics.

3.8. Design architecture of an IDS

In the previous section, we discussed Snort, Zeek and Suricata with their primary design goals. For instance, Snort is a basic signature- based IDS and can be used as a light IDS for solutions that do not require much customization. Zeek can be used as a powerful sniffer in a network which can capture the detail of the traffic and forward to analysis system Whereas, Suricata is designed for more flexible detection solutions to allow network administrators to customize the script-based detection. Having researched the top open-source Intrusion Detection Systems, we then moved on to understand how they operated. For that firstly we have to understand the architecture of IDS. The architecture is one among the foremost critical considerations in IDSs.

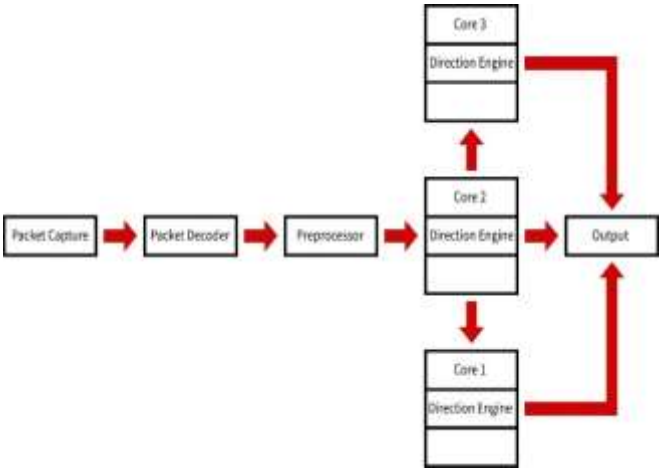

Fig. 1. IDS Components.

The process through which IDS such as Snort and Suricata (that use the multithreaded architecture) detect suspicious behavior in networks, is explained below.

- The IDS collect data packets using a packet collection method (libpcap is the default packet acquisition method). It then transfers these collected packets to the decoder layer. The pre-processor subsequently collects the decoded data from the decoder layer.
- The pre-processor then proceeds to break down the packets and reassemble them to record the sessions. If any

suspicious activity/data is identified, an alert is raised in the decoder layer before the packets reach the detection engine. This is where the main function of the IDS occurs.

- The function of the detection engine is to cross-check any suspicious behavior by employing the detection rules (either through default rules or the predefined rules set by the network admin specifically). In IDSs that use the multithreaded architecture, the network administrators can customize and select which threads to use and designates threads to CPU cores according to their own preferences. In most cases, each of the thread acts as a distinct detection engine which processes different levels of packets simultaneously. It ultimately increases the efficiency and rate of detection, although this may compromise distribution of system's resource allocation.
- If malicious behavior is identified in any of the detection cores, the network administrator is notified by an alert or the malicious packet is dropped altogether (while an alert is logged simultaneously for the outer module).

An effective architecture is one during which each machine, device, component, and process performs its role in an efficient and (often) coordinated manner, leading to efficient information processing and output. There are two types of core architectures

- Single Core Architecture
- Multi Core Architecture

3.8.1. Single Core Architecture

Single core architecture is responsible for deployment of IDS module i.e. packet capturing, processing etc. The single core module of IDS is mainly employed for efficient working. In Snort, it uses the Libpcap library for packet capturing module. This library is employed in Snort DAQ. Data acquisition (DAQ) is a library used for packet Input/Output. Snort DAQ is particularly used for our testing. The packets interact with NIC and then go to DAQ for further processing. The most important part known as packet capturing module needs optimization to overcome the security issues. We study the existing architectures for packet capturing in depth and try to optimize them using single NIC.

3.8.2. Multi Core Architecture

In multi core architecture, more than one core is responsible for deployment of IDS module i.e. pattern matching, detection method etc. This multi core architecture is highly efficient for the deployment of IDS because packet processing is carried out on single core and all other processes are on remaining core. It makes the system efficient to overcome the security issues.

3.8.3. Packet Capturing

IDSs capture packets from the Network Interface Controller (NIC) and transfer to the main detection engine. IDSs such as Snort, Suricata, and Zeek usually on external packet capturing mechanisms such as Libpcap which is the default packet capturing method used in all three IDSs. Libpcap is an open-source library that enables the network administrators to capture data packets from the NIC. The data packets are then transferred by the NIC driver to the protocol stack, where the OS's network protocol stack evaluates the data packets and then distributes the packets to their corresponding applications/programs. The whole packet capturing mechanism involves two key steps.

- Device initialization: The network administrator can call the pcaplookupdev() function (a function of the libpcap library) which lists down all the network devices (which are stored in the "pcapif" list). It is followed by calling the getifaddrs() which lists the IP address and relevant details of the listed network devices.
- Packet processing loop: IDS utilizes its pcapdispatch () method to declaim packets in the NIC. It then uses the pcapprocesspackets () function which crosschecks each data packet with the established network protocols. After being decoded, the processed packets are sent to the pre-processor.

3.8.4. Packet Decoding

The packets present on the NIC are transmitted to decoder for further processing i.e. for user space application. The decoders examine the attacks or malicious packet present in the traffic. It also generates alerts to the system if it finds malicious activity present in the network.

3.8.5. Packet Preprocessing

As indicated from the name, pre-processing module works in this part of IDS. Defragmentation process and anomaly detection method is applied in this phase. IP address checking and matching processes are also performed in this phase.

3.8.6. Packet detection

Traditionally packet detection is considered as the process through which an IDS inspects each individual

byte of every packet. However, in recent times, various methods of packet detection have been introduced. By reviewing the existing literature [7,17,20], it is analyzed that there are two main packet detection mechanisms which are most commonly used in IDSs nowadays. These two main packet detection mechanisms are the Aho-Corasick algorithm and the regular expression (RE) mechanism.

The Aho-Corasick algorithm was introduced in 1975 by Alfred V. Aho. It is a very basic packet matching technique [20]. The Aho-Corasick algorithm involves creating a "dictionary" of strings which are determined by the network administrator. The IDS compares these set strings with all the strings present in the packets of data that is being inspected. The Aho-Corasick algorithm comprises of three different functions including the failure function, the goto function, and the output function.

The regular expression mechanism is a signature matching technique that employs classes of characters, elements, unions, etc. to make the IDS more flexible. A state machine embodies a standard regular expression. Two types of state machines were introduced in previous studies, the Deterministic Finite Automaton and the Non-Deterministic Finite State Automaton [21]. A DFA outputs a single state whereas a NFA outputs a set of states. DFA is faster but it requires large amount of available memory, whereas the NFA is more compact but relatively slower than the DFA [7, 17]. In recent times researchers are struggling to figure out how to use the DFA more efficiently. In the work of Gong et al [17], a multi-dimensional Finite Automaton was introduced which reduced the construction time, the memory required, and the packet matching time.

4. EVALUATION STRATEGY

By conducting this research, it was found that there were some intricacies of IDSs that were yet to be fully understood. We conducted multiple experiments overall to prove the basis of our theoretical framework. In the first experiment, we tested the performance of Snort, Suricata and, Zeek under controlled environments to effectively monitor and compare the results. These solutions were operated in the Network Security Monitoring mode. We used Ostinato (a GUI based tool used to gauge network statistics) to create multiple UDP flows that were 1500 bytes in size individually. We then proceeded to measure the performance in NSM mode by steadily increasing the network throughput from 100Mb/s to 1000Mb/s. This experiment helped in inspecting the performance of each IDS and the impact on IDS performance in case of packet loss and CPU usages. We changed the Packet capturing mechanisms accordingly.

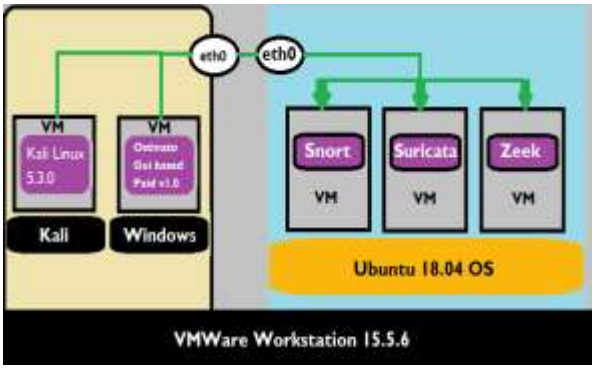

Fig. 2. Experimental test bed Environment.

In our 2nd experiment, we again tested the performance of Snort, Suricata and, Zeek under the same controlled environments to monitor and compare the subsequent results. We then proceeded to measure the performance in each IDS mode by steadily increasing the network throughput from 100Mb/s to 1000Mb/s. We used Ostinato (a GUI based tool used to gauge network statistics) to create multiple UDP flows, that were each 1500 bytes in size. This experiment helped us to inspect the performance of each IDS and the impact on IDS performance in terms of packet loss and CPU usages. We changed the Packet capturing mechanisms for Snort and Suricata IDSs and uses AF_Packet in replacement of default Libpcap and extract the results about the performances of both IDSs.

In our last experiment setup, we tested the security effectiveness of Snort and Suricata only. We did not test the Zeek because Zeek does not work in Intrusion Prevention System mode, and we also lack the mechanism of writing same rules for Zeek that have been embedded in Snort and Suricata. Security testing involved attacks as well as evasions. For the testing, we had two options under considerations: Pytbull and Kali Linux. Pytbull lacks the attack control and executes all the attacks at the same time. Whereas in the Kali Linux, we can launch an attack in specific order and take the readings accordingly. Hence, the Kali Linux's platform was preferred over Pytbull.

First Experiment (Parametric Performance in Monitoring Mode)

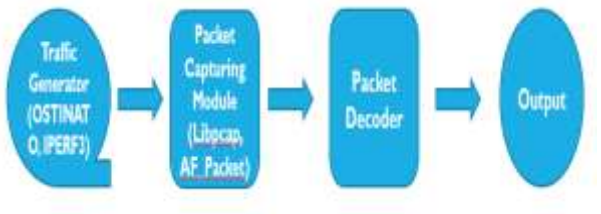

Fig. 3. Test Environment NSM Mode.

The objective of this experiment was to demonstrate the effect of increasing network throughput on IDS performance in NSM mode. We chose Snort, Suricata and Zeek so that we could present the performance evaluation (and comparison) between these three most popular IDSs and determine how they operate in increasing network throughputs. The experiment was set up in an ideal (controlled) environment as in this case there was no background traffic. Hence it enhanced the accuracy and reduced variability of the experiment's results. Snort, Suricata and Zeek were installed on a receiver server and Ostinato was used to generate the 1500 bytes (packet size) network flows. We selected a size of 1500 bytes per packet since as it is the Maximum Transmission Unit (MTU) that can be used in our test bed environment.

The experiment was initiated with the default IDS configuration in network security monitoring mode and protocols to monitor the CPU usage, memory usage, and network utilization. In the prior research, it was known that different packet capturing, and detection mechanisms cause varying impacts on the CPU usage, memory usage, and packet loss rate. For the first run of this experiment, we generated single UDP flow and measure the performance of the IDS via Libpcap in NSM mode. After examining the results, we tweaked the packet capturing and changed the Packet capturing mechanisms for Snort and Suricata. In this case, we used AF_Packet by replacing default Libpcap to extract the results and inspect the performance of Snort and Suricata. We repeatedly used AF_Packet until we achieved comparable performance results, thereby optimizing the IDS for maximum performance.

Second Experiment (Parametric Performance in Detection Mode)

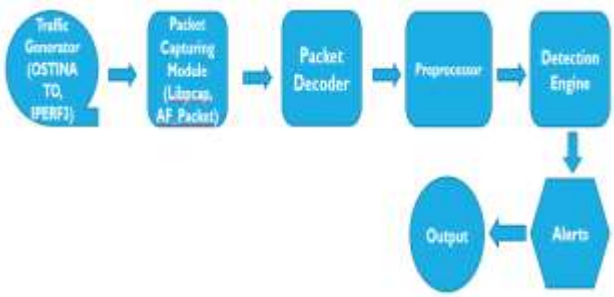

Fig. 4. Test Environment IDS Mode.

The objective of this experiment was to demonstrate the effect of increasing network throughput on IDS performance in IDS mode. The experiment was set up in an ideal (controlled) environment. Therefore, in this case, there was no background traffic. Hence it increased the accuracy and reduced variability of the experiment results. The experiment was initiated with the default IDS configuration in network security monitoring mode and protocols to monitor the CPU usage, memory usage, and network utilization. Snort, Suricata and Zeek were installed on a receiver server and Ostinato was used to generate the 1500 bytes (packet size) network flows. We generated UDP flows and measure the performance of the IDSs via Libpcap in IDS mode. After examining the results, we tweaked the packet capturing and changed the Packet capturing mechanisms for Snort and Suricata. For them, we now used AF_Packet in replacement of default Libpcap to extract the results and inspect the performance of both IDSs. It was repeatedly used until we achieved comparable performance results, thereby optimizing the IDS for maximum performance.

Third Experiment (Security Evaluation)

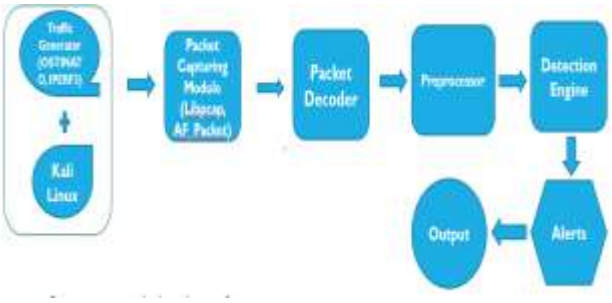

Fig. 5. IDS Security Testing Environment.

After employing the NSM and IDS mode, the last step was to check the security effectiveness of these IDS solutions. For that purpose, we simulated our attack environment inside Kali Linux. We launched several types of attacks from Kali Linux which include Brute Force (SSH), DOS, HTTP, Ping, ARP Poisoning, Scanning and, Brute Force (FTP). After launching these attacks, we monitored the response of our IDS solutions which included Snort and Suricata. Due to the non-availability of the Zeek rules we did not consider Zeek in the security analysis. The rule set was same in case of both Snort and Suricata which was the default rule set (10,000). We considered the false positive and false negative rates of our IDS solutions as they were the concerned features. NSS Labs also used these features during their evaluations, and we tried to map our testing scenarios to that of NSS Labs.

5. RESULTS

Following the conduction of experiments and recording the results, we proceeded to examine the data gathered in the process. In each of the experiments, we keenly monitored all the factors affecting IDS's performance.

Three main factors that directly impact IDS's performance are the CPU usage, the memory usage, and the packet drop rate. For each experiment we ran Snort, Suricata and Zeek separately; took the average of the results to obtain a more accurate representation of the data. Initially we used the default IDS configuration and a 1 GB/s network flow. After that we adjusted the packet capturing and packet detection techniques adaptively so that we could obtain maximum IDS efficiency.

For carrying out the procedure for data collection, we used the same data collection techniques that were used in existing research of Fitzsimmons [22], and Stammler et al [23]. Both Suricata and Snort display relevant network metadata to the user such as data regarding attacks, statistics of the analysed network traffic, and the packet drop rate of the IDS. For each experiment, we recorded the amount of data given as input to the IDS, as well as the output amount gathered in return. During each test, we keenly evaluated the effects of the trials on the CPU usage, memory usage, and packet drop rates.

5.1. IDS Performance

The primary purpose of the first experiment conducted was to evaluate IDS performance in monitoring mode. The experiment procedure was divided into two parts. In the first part, Snort, Suricata and Zeek were run in Monitoring mode. We generated traffic at 100 Mbps and 1 Gbps and recorded the performance parameters. The second part of our experimentation involved comparing the IDS performances of Suricata, Zeek and Snort with one another, under varying throughputs (but we retained the default IDS configurations).

In our second experiment, we adjusted the packet capturing mechanisms based on the packet drop rates that were being recorded. Since overloading the IDS results in high packet drop rate, this consequently decreases the effectiveness of the packet detection mechanism and thereby reduces the IDS's performance. In our experiment, once we recorded a high amount of packet loss in a trial, we stopped continuing the experiment with the current settings, since it would be a waste of time to continue when the IDS had already been proved ineffective.

In Fig 6 we have represented the comparison of the performances of all three IDSs which were recorded in the experiment. While processing the same flow, Snort CPU usage was 10% in 100Mb/s. It jumps to 38% when throughput increase to 1000Mb/s and memory usage was increased from 9.1% to 9.5%. The difference in the performances between Snort, Suricata and Zeek does not show as much improvement in terms of CPU usage as in case of packet loss. Zeek does not show any packet drop even when we increase throughput from 100Mb/s to 1000Mb/s. The packet drop rate reduction is the only monumental difference between all three IDSs including Snort, Suricata and Zeek.

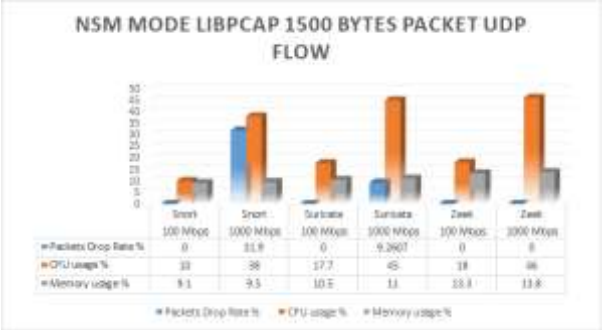

Fig. 6. NSM MODE libpcap 1500 Bytes packet UDP Flow.

To test the performance of IDSs in networks, we repeated the experiment for 1000Mb/s network flow. In this procedure, we noticed that high network flow overloaded the system and caused too much strain on the CPU and memory. The result was that many packets were dropped by the IDSs thereby reducing the detection rate significantly. To fix this problem, we re-adjusted our experiments variables by using 100Mb/s and 1000Mb/s network flows. Originally, we had planned to use both Libpcap and AF_PACKET for the experiments. However, after using both packet capturing mechanisms, we observed that when we ran Suricata with Libpcap on a 100Mb/s to 1 GB/s network flow in NSM mode, it resulted in a 0% packet drop in 100Mb/s throughput and we observed 9.2% packet drop in 1000Mb/s. As mentioned previously, we had decided to discontinue a specific variation of the experiment when we realized that IDS performance had be clearly compromised. Unlike Libpcap, when we ran use Suricata with AF_PACKET, no packets were dropped in monitoring mode till the network flow was increased.

Fig 7 presents the graphical representation of Snort's and Suricata's CPU usage, memory usage and packet drop rate when employed under increasing network throughputs. Since we had determined that Libpcap was not viable under higher network flows, we conducted the remaining experiment using AF_PAKET. We noticed that when dealing with 1 GB/s network flow, the average CPU usage of Suricata was 48% (whereas packet drop rate was 0%). When we employ the same experimental circumstances with Snort (using AF_PACKET and 1 GB/s network flow), the results showed up to 40% CPU consumption (and 21% packet drop rate).

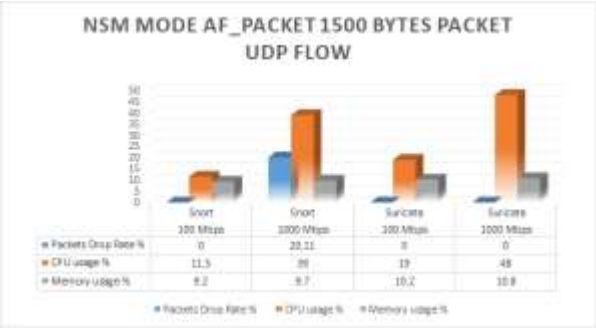

Fig. 7. NSM MODE AF_Packet 1500 Bytes packet UDP Flow.

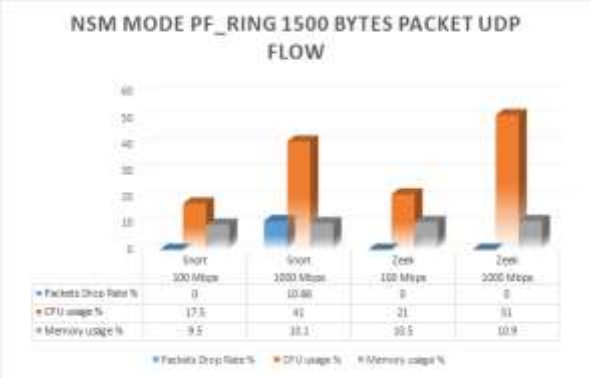

Fig. 8. NSM MODE PF_Ring 1500 Bytes packet UDP Flow.

Since Zeek is not compatible with AF-Packet socket, we conducted NSM mode testing for Zeek via PF_Ring. Fig 8 shows the graphical representation of Snort's and Zeek's CPU usage, memory usage and packet drop rate when employed under increasing network throughputs. We conducted our experiment using just PF_Ring. We noticed that when dealing with 1Gb/s network flow, the average CPU usage of Zeek was more than 50% (whereas packet drop rate was 0%). When we employ the same experimental circumstances on Snort (using PF_Ring and 1Gb/s network flow), the results showed up to 41% CPU consumption (and 10.1% packet drop rate).

By comprehending all these results, we summarized that as the network throughput increases, so does the strain on system resources.

In our next experiment, we ran Snort and Suricata in IDS mode and we adjusted the packet capturing mechanisms based on the packet drop ates that were being recorded. Ptacek [12] also proposed a similar correlation. In his research it was proposed that by overloading the IDS, attacks detection can be avoided. Since overloading the IDS results in high packet drop rate, this consequently decreases the effectiveness of the packet detection mechanism and reduces the IDS's performance. In our experiment, once we recorded a high amount of packet loss in a trial, we stopped continuing the experiment with the current settings as it would be a waste of time to continue when the IDS had already been proved ineffective.

In Fig 9, we represented the performance comparison of Snort and Suricata that were recorded in the experiment. While processing the same flow, Snort CPU usage was

29% in 100Mb/s. It jumps to 52% when throughput increase to 1000Mb/s and memory usage increased from 24.7% to 25%.

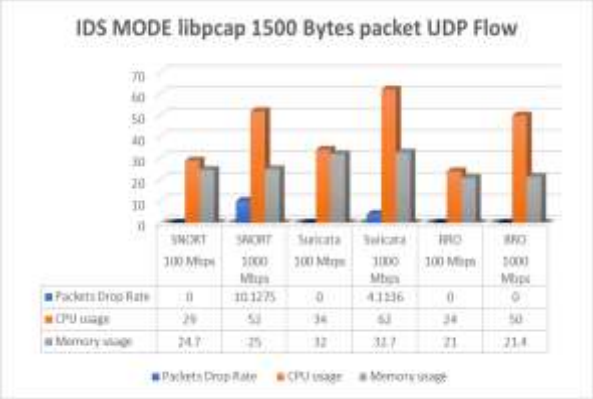

Fig. 9. IDS MODE Libpcap 1500 Bytes packet UDP Flow.

The differences in the performance between Snort and Suricata while using Libpcap does not show as much improvement in terms of CPU usage as in case of packet loss. Suricata and Zeek does not show any packet drop in 100Mb/s but Suricata shows 4.1% packet drop with 1000Mb/s throughput. The packet drop rate reduction is the only monumental difference between Snort and Suricata.

Fig 10 shows the graphical representation of IDS mode of Snort's and Suricata's CPU usage, memory usage and packet drop rate when employed under increasing network throughputs. We noticed that when dealing with 1 GB/s network flow, the average CPU usage of Suricata was more than 50% (whereas packet drop rate was 0%). When we employ the same experimental circumstances with Snort (using AF_PACKET and 1 GB/s network flow), the results showed up to 54% CPU consumption (and 0% packet drop rate). By analyzing the results of experiments, we summarized that as the network throughput increases, so does the strain on system resources, it ultimately results in higher CPU usage.

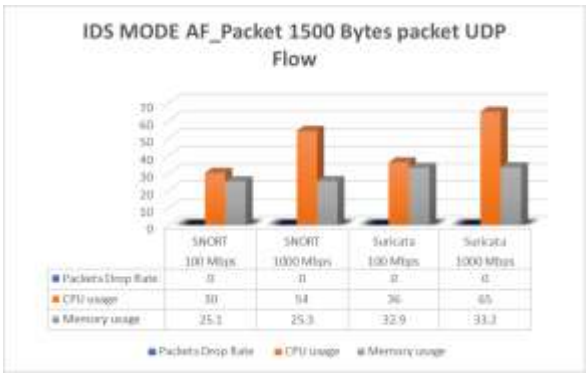

Fig. 10. IDS MODE AF_Packet 1500 Bytes packet UDP Flow.

5.2. Security Testing

After the NSM and IDS mode, the last step was to check the security effectiveness of these IDS solutions. For that purpose, we simulated our attack environment inside kali Linux. We launched several types of attacks from Kali Linux including Brute Force (SSH), DOS, HTTP, Ping, ARP Poisoning, Scan and Brute Force (FTP). After launching these attacks, we monitored the response of our IDS solutions which include Snort and Suricata. Due to the non-availability of the Zeek rules, we did not consider it in the security analysis. The rule set was same in case of both Snort and Suricata that was the default rule set (10,000). We considered the false positive and false negative rates of our IDS solutions as they were the concerned features. NSS Labs also used these features during their evaluations, and we tried to map our testing scenarios to that of NSS Labs. Following are the results of the tests conducted in tabular form:

- Brute Force (SSH): Brute Force is an attack launched to access credentials through multiple login attacks until the attack/attacker gets the authentication.
- DOS: Denial of Service is an attack launched to security perimeter solution to ensure the non-availability of internet or to access through the security perimeter.
- HTTP: HTTP is a type of DDOS (Distributed Denial of Service) attack. HTTP is a flood launched against web server or an application using GET and POST requests which are legitimate requests.
- Ping: It involves sending malicious pings to a computing system.
- ARP Poisoning: It include man in the middle attack having intent to receive network packets of targeted host.
- Scan: Network discovery.

- Brute Force (FTP): It is an attack launched to access credentials through multiple login attacks until the attack/attacker gets the authentication.

Attacks	Snort		Suricata	
	FPR (%)	FNR (%)	FPR (%)	FNR (%)
Brute Force (SSH)	7.94	0.22	6.34	0
DOS	2.85	0.98	2.71	0.3
HTTP	5.94	0.7	7.5	0.3
Ping	15.4	0.2	12.7	0
ARP Poisoning	6.2	0.61	8.3	0.91
Scan	1.75	1.89	1.41	2.5
Brute Force (FTP)	9.3	0.5	8.7	0

Table 1. Security Analysis.

SSH attack was achieved using Hydra tool on Kali Linux. Port 22 was left open by using Open SSH server in Snort and Suricata Virtual Machines. DOS attack was achieved using hping3 command using Kali Linux. HTTP attack was achieved using Kali Linux Metasploit. For the ping attack, which is also known as, ping of death, ping command was used by employing Kali Linux. For the ARP poisoning attack, DSniff package was installed on Kali Linux and then ARP spoof was used for the attack. NMAP command was used for scan attack using Kali Linux. Kali Linux Metasploit was used for the Brute Force (FTP) attack and port 21 was left open by installing vsftpd server in Snort and Suricata Virtual Machines.

For the purpose of attack, there were two choices to consider. One of these were Pytbull and another was Kali Linux. Kali Linux was preferred over Pytbull as it has more control using individual attacks that Pytbull does not offer.

For the evaluation purpose, the FPR and FNR were considered. Their description follows as.

- FPR (False Positive Rate): IDPS generates an alert in the absence of attack. The lower, the better.
- FNR (False Negative Rate): IDPS generates no alert in the presence of attack. The lower, the better.

Evasions

Bypassing security mechanism and devices in place to perform an attack or steal information without detection is known as evasion. This method is mostly used to nullify security appliances like IPS, IDS and network firewalls. A more detailed use of this method can be to crash or find a loophole in the network for a much more devastating attack.

After attacks, we stimulated evasion environment inside Kali Linux by using NMAP tool. Following are the evasions that were covered during the testing.

Evasion	Snort	Suricata
IP Packet Fragmentation	✓	✓
Stream Segmentation	✓	✓
RPC Fragmentation	✓	✓
FTP	✓	✓

Table 2. Evasion attacks

- IP Packet Fragmentation: Packet fragmentation is an old and common evasion technique that splits the packet header across many small packets. Internet Protocol fragments the bigger packet to smaller chunks so that the packets could pass through link which has smaller MTU (Maximum Transmission Unit).
- Stream Segmentation: It is the same evasion technique that we discuss above the only difference in Stream segmentation full Streams are divided into segments instead of packets.
- RPC Fragmentation: Remote Procedure Call is a protocol that can inter link the two systems without having the complete information of the network in the first place.
- FTP: There is a vulnerability in in FTP Protocol which is known as FTP Bounce Attack through which an attacker can leverage PORT Command to ask for ports indirectly by making use of the victim machine. Just like an open relay SMTP Server, the victim machine is used to relay request working as a proxy server. This is a technique which allows port scan host with discretion, and via this method access to specific ports is solicited by going around network access control list which cannot be access through a direct connection e.g. through NMAP port scanning. Keeping in mind this exploit, today's modern FTP Servers are designed to not allow PORT Commands by default that would allow connection to any host but only the host from where the request is originating, thus mitigating the FTP Bounce attacks.

First three evasions were achieved using NMAP command of Kali Linux. The FTP evasion was achieved by using Kali Linux Metasploit. Snort and Suricata were running in a Virtual Machine in VMware Workstation while Kali Linux was running as another Virtual Machine.

6. DISCUSSION

We initiated this research by identifying the shortcomings of the IDSs. The existing packet capturing mechanisms

cause great stress on system resources and their performance results deteriorate when dealing with high-speed networks. The packet capturing mechanisms that are currently employed in IDS solutions were also outdated as they report high packet drop rates when dealing with large volumes of network traffic. Evidently, we have seen that the performance of the IDS solutions is affected by the factors such as increasing network flows, inefficient packet capturing, and incompatibility with high network throughputs. To tackle the issues mentioned above, there are few solutions that includes using a load balance mechanism (to distribute network flows amongst multiple IDS instances) and using efficient regular expression algorithms in the packet matching process. For better multiple gigs packet-capturing module, Data Plane Development Kit (DPDK) can be considered. DPDK comes with the integration of a dedicated hardware solution that brings the compatibility issues in the long run.

From all the experiments, we have concluded that the performance of Suricata has greatly improved in its newer versions which now employs the multi-threaded as compared to Snort which uses single threading. We had deployed Snort's version 2.9.16 and Suricata's version 5.0.0. The newer version of Suricata has reported much better performance than its predecessors. Thus, we believe that it is possible for Snort as well to perform better if the multi-threaded architecture was adopted in the Snort IDS as well.

We have tested the performance of Suricata on 1000Mb/s network throughput and multi-threading has resulted in promising results. We have tested packet-capturing module Libpcap and AF_Packet for both the Snort and Suricata and compared them for 100Mb/s and 1000Mb/s. As mentioned earlier, it is evident that Suricata outperforms Snort in parametric performance testing. Besides the parametric performance testing, we carried out few security tests as well.

6.1. Suggestions / Future Work

Based on our parametric performance results, we prefer Suricata over Snort for the deployment of high traffic network. However, as far as the security is concerned, there is a need for more research before the deployment of any open-source solution. The rulesets need to be strengthened before the deployment of any open-source solution. Multiple attacks can be generated from different attack machines and rules can be embedded to the rule sets accordingly. The already existing rules can be studied and rewrite to be more effective.

There is a great variety of possible future work in the same domain. Same rule sets can be augmented to carry out work on Zeek which seems to be a research value. It will then enable to determine the security effectiveness can be evaluated between Snort, Suricata and Zeek. Parametric performances can be evaluated under extremely high traffic (above 10 gigs). The integration of DPDK can be valuable under extremely high traffic such as 40 gigs. The integration is dependent on the specific hardware, and it can produce some compatibility issues as well, but the details can be studied and highlighted in the literature. Snort beta version (3.0) can be used for the evaluation that has multi-threading capability and it can produce much better results. The current results have shown that the FPR and FNR values are quite ambiguous, and these can be improved by the integration of AI module to the existing solutions.

Through our research, we came across various areas of study related to IDS performance on which there is not any existing research. These areas are yet to be investigated and they can provide the basis for our future work. We can also evaluate the performance of IDS based on the effect of flow duration. To carry out research in this domain, multiple long-lived network flows can be configured, or alternatively multiple short-lived network flows can be configured in an experiment to observe the impact on IDS performance. Lastly, we can also investigate why IDS reports do not accurately reflect the network's packet drop statistics.

7. CONCLUSIONS

In this research work, we focused on determining the feasibility of employing popular open-source IDSs by analyzing their performance. The experiment results (detail in the 'Experiment results' chapter) proved that by adopting the multithreaded architecture, IDSs performance in terms of packet drop rate will be greatly improved. Additionally, it was also found out that both IDSs (Snort and Suricata) report better performance when they are utilized on network traffic under 1Gb/s. From our experiments, it can be concluded that the IDS performance degrades if the number of flows is increased.

We have tested the current popular IDS solutions such as Snort, Suricata, and Zeek. All of three solutions have been evaluated as Network Security Monitoring solutions and relative results have been obtained and reported in the experiment results portion of the thesis. Based on the Network Security Monitoring approach, we found out that Zeek is optimum choice. For the Intrusion Detection System mode, it was found out that Suricata is better than the Snort on the basis of packet loss. Moreover, Zeek does not have rule support. It was determined through this research that it is very difficult to convert Snort and Suricata rules for the Zeek. We then conducted security testing of Snort and Suricata. The results have been reported in the experiment result portion of the thesis. On the basis of the accumulative FPR and FNR test results performed in the security testing, Suricata can be concluded as better solution compared to Snort.

ACKNOWLEDGEMENTS

First and foremost, I must acknowledge my limitless thanks to ALLAH, the Ever-Magnificent; the Ever-Thankful, for His help and blessings. I am totally sure that this work would have never become truth, without His guidance.

I owe a deep debt of gratitude to my university for giving me an opportunity to complete this work.

I am grateful to some people, who worked hard with me from the beginning till the completion of the present research.

- I wish to express my sincere appreciation to my supervisor, AIR CDR Prof. Dr. Ammar Masood, who has the substance of a genius: he convincingly guided and encouraged me to be professional and do the right thing even when the road got tough. Without his persistent help, the goal of this project would not have been realized.
- I highly appreciate the efforts expended by Dr. Mansoor Ahmad and Dr. Nadeem Sial. I would like to take this opportunity to say warm thanks to all my beloved friends, who have been so supportive along the way of doing my thesis.

REFERENCES

[1] Salah K, Kahtani A. Performance evaluation comparison of Snort NIDS under Linux and windows server. J Netw Comput Appl 2010;33(1):6–15.

[2] Alhomoud A, Munir R, Disso JP, Awan I, Al-Dhelaan. Performance evaluation study of intrusion detection systems. Procedia Comput Sci 2011;5:173–80.

[3] Hu Qinwen, Asghar MR, Se-Young Yu. Analyzing performance issues of open-source intrusion detection systems in high-speed networks. In: Journal of Information Security and Applications. Elsevier; 2020 p. 1–6.

[4] Campbell S, Lee J. Intrusion detection at 100g. In: State of the Practice Reports. ACM; 2011. p. 14.

[5] Antonatos S, Anagnostakis KG, Markatos EP. Generating realistic workloads for network intrusion detection systems. ACM SIGSOFT Software Eng. Notes 2004; 29(1):207–15.

[6] Sidhu R, Prasanna VK. Fast regular expression matching using FPGAs. In: Field--Programmable Custom Computing Machines, 2001. FCCM'01. The 9th Annual IEEE Symposium on. IEEE; 2001. p. 227–38.

[7] Yang J , Jiang L , Bai X , Peng H , Dai Q . A high-performance Round-Robin regular expression matching architecture based on FPGA. In: 2018 IEEE symposium on computers and communications (ISCC). IEEE; 2018. p. 1–7.

[8] Clark CR, Schimmel DE. Efficient reconfigurable logic circuits for matching complex network intrusion detection patterns. In: International conference on field programmable logic and applications. Springer; 2003. p. 956–9.

[9] Purzynski M., Manev P. Suricata extreme performance tuning. 2016. https://suricon.net/wpcontent/uploads/2016/11/SuriCon2016 – MichalPurzynski_PeterManev.pdf.

[10] Johnson J. Reproducible performance testing of Suricata on a budget with Trex. 2018. https://suricon.net/wpcontent/uploads/2019/

01/SuriCon2018_Johnson.pdf.

[11] Schaelicke L, Freeland JC. Characterizing sources and remedies for packet loss in network intrusion detection systems. In: IEEE International. 2005 Proceed- ings of the IEEE workload characterization symposium, 2005. IEEE; 2005. p. 188–96.

[12] Ptacek TH, Newsham TN. Insertion, evasion, and denial of service: Eluding net- work intrusion detection. Tech. Rep. SECURE NETWORKS INC CALGARY AL- BERTA; 1998.

[13] Samoshkin A. Certified Snort integrator program. 2018. https://www.Snort.org/ integrators.

[14] McCanne S, Jacobson V. The BSD packet filter: a new architecture for user-level packet capture. USENIX winter, 46; 1993.

[15] Bezborodov S., et al. Intrusion detection systems and intrusion prevention sys- tem with Snort provided by security onion2016.

[16] MetaFlows. Open source IDS multiprocessing with PF_RING. 2016. https://www.metaflows.com/features/pf _ ring/.

[17] Gong Y, Liu Q, Shao X, Pan C, Jiao. A novel regular expression matching algorithm based on multi-dimensional finite automata. In: High performance switching and routing (HPSR), 2014 IEEE 15th international conference on. IEEE; 2014. p. 90–7.

[18] Becchi M, Wiseman C, Crowley P. Evaluating regular expression matching en- gines on network and general-purpose processors. In: Proceedings of the 5th ACM/IEEE symposium on architectures for networking and communications systems. ACM; 2009. p. 30–9.

[19] Matoušek D, Ko ˇrenek J, PušV. High-speed regular expression matching with pipelined automata. In: Field-programmable technology (FPT), 2016 Interna- tional Conference on. IEEE; 2016. p. 93–100.

[20] Aho AV, Corasick MJ. Efficient string matching: an aid to bibliographic search. Commun ACM 1975;18(6):333–40.

[21] Hopcroft JE, Motwani R, Ullman JD. Introduction to automata theory, languages, and computation. ACM Sigact News 2001;32(1):60–5.

[22] White JS, Fitzsimmons T, Matthews JN. Quantitative analysis of intrusion de- tection systems: Snort and Suricata. In: Cyber sensing 2013, 8757. International Society for Optics and Photonics; 2013. p. 875704.

[23] Stammler JH. Suricata performance white paper. 2011. https:// redmine.openinfosecfoundation.org/attachm ents/download/763/Suricata% 20performance%20writup%20final.pdf

[24] Day D, Burns B. A performance analysis of Snort and Suricata network intrusion detection and prevention engines. In: Fifth international conference on digital society, Gosier, Guadeloupe; 2011. p. 187–92.

[25] Leblond E. Why eBPF and XDP in Suricata matters. 2018. https://suricon.net/ wp- content/uploads/2019/01/SuriCon2018 _ Leblond.pdf.

[26] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, et al. Openflow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev 2008;38(2):69–74.