An Improved Linux Firewall Using a Hybrid Frame of Netfilter

Nivedita
Department of Computer Science and Engineering
National Institute of Technical Teacher's Training and Research (NITTTR)
Chandigarh, India
nivedita.nahar19@gmail.com

Rakesh Kumar
Department of Computer Science and Engineering
National Institute of Technical Teacher's Training and Research (NITTTR)
Chandigarh, India
raakeshdhiman@gmail.com

***Abstract*—With the steady advancement of the network technology present day, network not only brings us a conducive and productive life, and is followed by a collection of network security threats. Due to awareness about the threats the need for security has never been more important that's why it has become extremely important to protect our web servers as well as our web assets. A firewall is main security component that allows and restrict access to specific network and ports. In this research main focus is on designing strong firewall filtering rules so that detection of malicious code will be achieved to the optimal level. The proposed framework is introduced to improve performance issues, code maintenance (i.e. code duplication), scalability, for improving performance of the network traffic etc. in the dataset. In this work, we examine the Linux Netfilter/iptable, nftable firewall technology. In this paper, a new hybrid approach is proposed where Geometric efficient matching algorithm and stateless firewall optimization algorithm is merged into the code of the Linux iptables and nftables open source firewall for securing Linux web server. "Geometric Efficient Matching algorithm" GEM- iptables & nftables execution manage to filter packets-per-second on a standard system. It is efficient and practical, algorithm for firewall packet matching. While there are a number of paths that can be followed to provide a best malware detection method for firewall security, this work will be beneficial for small enterprises in terms of money and time using Netfilter/nftables. This makes it easy and simple to configure the strong firewall to solve the security related problems & detect malware using strong firewall rules to achieve optimal level.**

Keywords—Linux firewall; metasploit; NFtables; Geometric efficient matching; Information security and protection

I. INTRODUCTION

The Internet has been used in various types of devices like laptop, desktop, smartphones, household equipment, sensors, Internet of Things etc. So it becomes the essential part of our daily life. As a connectivity of the network increases day by day, there are a lot of network security threats in our daily life. In recent years, all kinds of security attacks break out continuously such as in Russia there were over 100 user accounts were hacked, Dailymotion sites data were hacked 85.2 email addresses extracted, Myspace data hacked, a malicious attack on the Reserve Bank of India [1]. That's why it has become extremely important to protect our server or web assets. How should we proceed to establish a secure environment? There are many ways to ensure the security of your dedicated server, we will discuss firewalls their usage and configuration and which may be best for a different application.

LINUX FIREWALL

A firewall is an important tool that protects itself and other hosts on a network from an attacker on untrusted network[2]. A firewall can be very beneficial if it was used as a filter towards all of the Internet access to and from the system passes through it. A firewall system can also be configured to conceal multiple hosts behind a single Internet protocol address using a process known as Network Address Translation. The firewall protected server's schema as shown in figure 1.

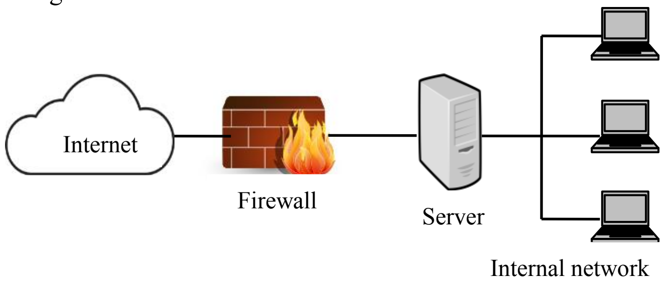

Figure1. Firewall protected server's schema

II. KEY TECHNOLOGY

A. NETFILTER

Netfilter is a packet filtering framework provided by the Linux kernel which allows different networking related operations to be implemented in the form of customized handlers[3][4]. It gives different functions and operations for network address translation, packet filtering, and port translation. It serves the functionality that is needed for guiding packets over a network likewise for serving a capacity to block the packets from reaching sensitive location with a computer network.

B. EXSITING FIREWALL ARCHITECTURE USING NETFILTER/IPTABLE

B. Wang et al. [3], proposed the architecture of the Netfilter/IPtable that is a subsystem of the Linux kernel, it's a new generation of Linux firewall for security. In this paper, author used Iptable rules for firewall configuration. The Netfilter follow the modular design and it has good expandability. Iptable is connected to the kernel mode of Netfilter architecture, as it is the essential tool of the module. Figure 2 shows Firewall Architecture Using Netfilter/IPtable.

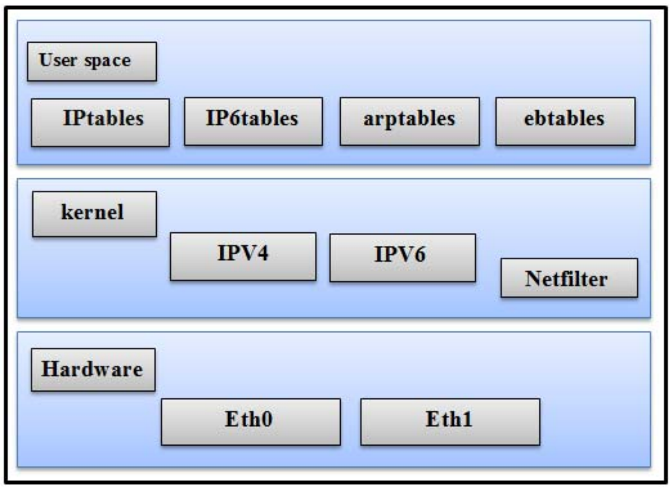

Figure 2. The Firewall Architecture Using Netfilter/IPtable.

There are three main interfaces of Netfilter/Iptable:

- User-space: Iptable is placed in first interface which is user-space. It provides an interface for setting up the firewall and firewall rules. The same rules apply to ip6tables, arptables, ebtables and so on.
- Kernel: Next interface introduces the Netfilter framework, Iptable is completely based on this interface. Netfilter implements a set of 'hooks' inside the Linux kernel that allows the kernel module to inspect packets in the protocol stack, such as IPv4. These sets of hooks allow kernel modules to interact with them. Iptables contain huge list of kernel modules which are used for firewall configuration.
- Hardware: In this interface network adapters are mentioned as eth0, eth1, and so on. Netfilter uses prerouting and postrouting to and from the network stack to check packets sent and received on each interface. So the packet checking is done to the kernel interface layer with the Netfilter. All the firewall rules and tools to manage the firewall are placed in the user-space.

III. RELATED WORK

All new discoveries don't show up out of nowhere; they basically build upon the result of previous research and findings. The literature review provides a direction to the research investigation and fit with what has been done before.

H. Mao et al. [2] explained firewall, their current state, classification of firewall, types of firewall with full information of firewall. This paper also gave a detailed view points in the future development trends and the drawbacks of firewall.

J. Phalguni et al. [4] worked on the network layer, using the port and IP protocol to control and protect the network from malicious attack. In this paper only network layer is discussed. This approach could not answer the attacks to the application layers, consisting of user recognized reorganization, business application, and others. It cannot bear the risks from application layers. Authors have used connection tracking technology that improves the efficiency of flow matching and results in reduced delay and increased network throughput.

S. Wang et al. [5] proposed an optimum design and implementation of Firewall architecture. In this research, they did a comparison on existing ASIC (Application Specific Integrated Circuit) architecture firewall hinge on State detection technology in dealing with the flaws of ASIC chip. They overcome the shortcoming which was present in ASIC firewall architecture. At the end, the technology provides optimum high performance, reliable, flexible and secure.

T. Isohara et al. [6] presented a Kernel base behavior analysis for android malware detection. There were two main functions as log collector and log analyzer which were used to detect malicious activities. System log data is analyzed by the Signature matching algorithm. Many applications were used to perform the detection. According to the results, they showed that their system can adequately catch malicious behavior of the anonymous application.

M. Weiquin et al. [7] presented a new class of attack called shadow attack to by-pass current behavior based malware detection by splitting one piece of malware into multiple "shadow process". Behavior based malware detection technique is followed. The Implementation of compiler level prototype tool to manifest its feasibility. They proposed technique that evaded certain malware detection effectively in the real world. Results showed that shadow process can automatically extract critical system call of malware.

R. Tyagi et al. [8] proposed unauthorized operating system detection scheme for the company. Euclidean distance estimation algorithm utilizing particular header fields then it resolved the OS of the machine producing packets. They correlated the fingerprinted OS results with the recorded OS in the company database of the OS. If a match fails to appear, an alert is developed indicating the presence of unauthorized or unregistered OS in the company. The alert utilized for human verification of a possible malware controlled virtual machine.

T. Zhang et al. [9] described a cloud-assisted vehicle malware defense framework. In this research, the author explained various threats which will happen to a vehicle, their challenges, existing solutions and their limitations were discussed. A cloud assisted vehicle technology deal with their challenges. Researcher explained the concern how to deal with the vehicles that facing malware attacks. There are various architecture was explained such as High-level security, cloud functional architecture, on-board malware defense functional architecture these architectures were implemented also.

A. Kumar et al. [10] Investigated the strengths of Netfilter/Iptable in beating the challenges of Intrusion Detection System (IDSs). The researcher trusts that the challenges of IDSs cloud be overcome by configuring Netfilter/Iptable to play out the elements of real-time prevention after the attacks have been recognized by the system. They put forwarded “a real time system that consists of an intrusion detection system based on self-organizing maps, for tracing down the malicious packets along with handling those packets through an intrusion protection system in the Linux environment”. Since most intrusion prevention frameworks depend on predefined databases of attacks marks, they are not ready to perform adequately when the attack mark is not in the predefined signature. In addition, they don't counteract inside attacks.

D. Rovniagin et al. [11] Proposed a classical algorithm called as “Geometric Efficient Matching.” The GEM is a logarithmic matching time performance. It tested on real traffic loads with large rules. Results showed that GEM needs 13 MB of space for 5,000 rules. They used space improving heuristics, further they able to reduce space of 2-3MB for 5,000 rules. They integrated algorithm into the code of the Linux iptables open source firewall, the execution handled to filter over 30,000 packets-per-second on a standard system, even with 10,000 rules. Therefore Geometric efficient matching algorithm is an efficient, & practicable, packet matching for firewall.

J. Alfaro et al. [12] Discussed firewall configuration, such as stateful and stateless firewall rules. Their results showed that the stateful firewall configuration tends to be more error-prone. In this research, the error occurred in stateful configuration rule. At the same time alternative can affect those holding both stateful and stateless firewall rules. They addressed an automatic solution to manage these problems. They identified flaws, error and then change in order to achieve a consistent configuration. In this research, they presented an automatic inspection tool which passed set up stateful firewall configuration rules & manages the discovered error in the rules. They proposed an algorithm that gave an optimal solution to handle error which based on an automatic theoretic approach. They proved the feasibility of this approach over a proof of concept prototype that depends on model-driven engineering. This approach was selected with the aim of separated the low level details of the problem from the enforcement of the algorithm solution.

K. Mindo et al. [13] Presented an analysis of network security and various firewall policies in a dynamic and heterogeneous network environment. For organization with geographically distributed offices and different network security infrastructure were most affected. In this research they looked into the issues of network security and firewall policies that was inevitable so as to cater high risk involved.

F. Shahzad et al. [14] Proposed a novel approach of genetic footprint drilling the information in the kernel process control blocks (PCB). This process can be used to detect malicious processes at run time. The experimental results showed that the depicted scheme accomplish a detection accuracy of 96% with 0% false alarm rate is less than 100 ms of the start of a malicious activity.

W. Kehe et al. [15] Introduced the security, defense technology stand on the Linux real-time monitoring to hinder malicious activities running from the operating system a kernel level. In this research, they described the principle theoretical model and implementation of the security defense technology. They used real-time monitoring to De poisoning machine and this technology gave effective results. It adequately removes virus like as machine dog, Trojan horse, etc.

IV. INTRA-STATE RULE MISCONFIGURATION

There are five fields present in the packet which is shown in table 1. The firewall packet matching issue finds the first rules that match a given packet on one or more fields from its header. The existing algorithms help to improve misconfigure firewall rules. These algorithms can be implemented in the stateful firewall rule cases as well and it allows us to transform rule sets with mixed policies. Note that the application of this extended solution does not change the configuration. These firewall configuration connects rules with relate to network layer information and reports those missing information with relate to stateful coverage, to detect the presence of intra-state misconfiguration rules. The pseudo code is compiled in first algorithm. It utilizes as input a stateful rule set R, in which each rule indicates an Action (e.g., ACCEPT or DENY) that will implement on set of condition attributes, like as

Table 1. Packet fields

Source address
Destination address
Source port
Destination port
Transition
Protocol.

Certainly, identifiers for the initial (Q_o), final (Q_n) and invalid states (Ø) Steps of the algorithm are:
(1) Make a set S that containing all the possible paths of valid transitions connecting the initial (Q_o) and the final states (Q_n) of automaton A (Line 6);
(2) Make a set T that containing all the transitions of automaton A to reach the invalid state (Ø) (Line 7);
(3) Then establish the report either S or T (Line 8 to 20), with respect to the Action (i.e., ACCEPT or DENY) and Transition attributes of all the rules in R;
(a) In case of a permission (i.e., rule in R whose Action attribute is set to ACCEPT) check one of the transition in a given path of s, then establish all the remainder transitions of such a path are also check by other permissions in R (Lines 11 and 15). If the established remainder fails, report those transition that are not check and the necessary rules to correct the failure (Line 16 and 18).
(b) In case of a prohibition (i.e., rule in R their Action attribute is set to DENY) checking, at least, one of the transition in T, establish all other transitions in T are also covered in R (Lines 13 and 15). If the established transition fails, then report those transitions that are not covered and the necessary rules to correct the failure (Lines 16 and 18).

Algorithm 1: [12] audit_ rule_ set (A, R, Qo, Q_n, and Ø).

```
1.  Input : A. protocol automaton
2.  Input: R, stateful rules set
3.  Input: $Q_0$. Initial state of automaton A
4.  Input : $Q_n$ final state of automata A
5.  Input : Ø invalid state of automaton A
6.  S $\leftarrow$ find all paths(A,$Q_0$,$Q_n$);
    /* S: Set of sets. S.t every element in S is a set of valid transitions connecting Q and q1 */
7.  T$\leftarrow$ transitions_to _state(A, Ø);
    /*T: Set of invalid transitions, s.t. every element in T is a transition of A to reach Ø */
8.  Repeat
9.  $r_i$ $\leftarrow$ next _unvisited _rule(R);
10. if ($r_i$ [Action] = ACCEPT) then
11. L $\leftarrow$ prune_ paths (S, $r_i$ [Transition]);
    /* L ⊂ S, s.t. every path in L is a set of valid transitions connecting $Q_0$ contains the state represented by $r_i$ [Transition] */
12. Else if ($r_i$ [Action] = DENY ) then
13. L$\leftarrow$ T;
14. end if
15. C$\leftarrow$ cover _with _rules (L, R, $r_i$);

16. M $\leftarrow$extract _missing _rules (C);
    /* M: set of mutually disjoint rules derived from R, s.t M∩R = Φ; let m be a rule in M, then m [Action] = $r_i$ [Action], Address and port attributes of m are consistent with rules in R, and m [Transition] is necessary to fully cover the set of transitions in L*/
17. if (M ≠ Φ) then
18. Report (intra state misconfiguration in R discovered via $r_i$ Update R based on rules in M);

19. end if
20. until no_ more_ rule_ to visit (R);

    /*no - more - rules - to – visit (R) holds true whenever all rules in R but the last are reported as visited (the last rule is not inspected given that it is just a mark to the default policy)*/
```

Algorithm 2: [12] For handle_ inter_ rule_ misconfiguration (L, F, A).
The evaluations use incremental configuration files (from 8 to 180 rules), based on iptables and the conntrack. match, representing [12]
(1) Open policies, containing only prohibitions to invalid TCP transitions
(2) Closed policies, containing only permissions to valid TCP transitions.
(a) Processing time needed to audit the files.
(b) Space necessary to store the associated structures in memory.

```
1.  Input: I, Set of stateless rules
2.  Input: P, Set of stateful rules
3.  Input: A, protocol automation
4.  /* A [$Q_2$] states for automation A*/
5.  For all q ∈ A [$Q_2$] do
6.  If rule exists (F, q) then
7.  /* Move following state*/
8.  Continue;
9.  Else if rule _exists (L, q) then
10. Report (‘stateles rule for q of protocol A’)
11. else
12. Report(‘missing rule for sale q of protocol A’)
13. End if
14. End for
```

Algorithm 3: [12] handle_ all_ protocols (L, F, Library)

```
1. Input: L, set of stateless rules
2. Input: F, set of stateful rules
3. Input: Library
4. For all A ∈ Library do
5. If rule_ exists (L,A [$q_0$]) then
6. Handle_ inter_ rule
7. End if
8. End for
```

The above mentioned algorithms have limitation that it does not warn about missing rules and code duplication.

V. PROPOSED METHOD

This proposed work will solve the problem of small scale and average enterprise/companies. Firewall filtering nftables rules designs a strong software firewall which will help average companies to protect their web servers from malicious content. The proposed technique is explained in the following steps as shown in figure 4.

1. Firstly, set up the network between client/server. After that configure the firewall using a hybrid approach which used the Netfiltering number of rules of Iptable, Nftables.
2. Then there will need to check vulnerable system with the help of tools. If any vulnerability found in any system in the network, then attacker easily injects malicious code.
3. Using the proposed algorithm, algorithm check for deep packet inspection and misconfiguration of firewall rules. Figure 4. Depicts the flow chart of the proposed method.
4. Determining the performance using packets-per-second, rules storage space, efficiency.
5. After the performance evaluation we will compare the results with existing parameters and final outcome will make strong firewall.

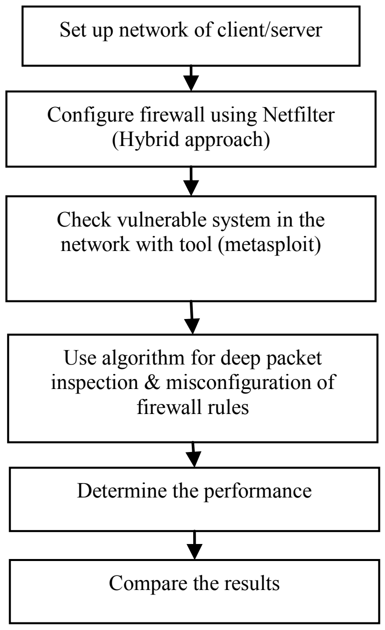
Set up network of client/server
Configure firewall using Netfilter (Hybrid approach)
Check vulnerable system in the network with tool (metasploit)
Use algorithm for deep packet inspection & misconfiguration of firewall rules
Determine the performance
Compare the results

Figure 4.Flow chart of the proposed method.

CONFIGURATION OF NFTABLE RULES

Nftable is a netfilter based process aims to replace the existing Ip, Ip6, arp, and etables framework. It provides a new packet filtering framework, a new user space utility nft and a compatibility layer for Ip and Ip6 tables. Whereas figure 5. Shows iptables and nftables families.

```
# /etc/nftables.conf   /*default file which already contain ipv4, ipv6 */
# nftables.service   /*it use to start and enable services*/
# nft list ruleset   /*to check ruleset*/
# /etc/modules-load.d/nftables.conf      /*it configure kernel module to load at*/
# lsmod | grep '^nf'   /*list of modules*/
```

There is no built in tables in nftables. The basic purpose of tables is to hold chains. There is great difference between iptable and nftable. There are no raw, filter, nat & mangle tables. In nftables you can specify some action in single rule only. while in case of iptable there are multiple action.

```
# nft add rule filter forward tcp dport 22 log drop
/*nftable cover this above action in single rule only*/
# Iptables A FORWARD -p tcp --dport 22 –j LOG
# Iptables –A FORWARD -p tcp –dport 22 –j DROP
```

There are most of the logic present in nftables is inside its userspace. It also provides a generic set and map infrastructure.

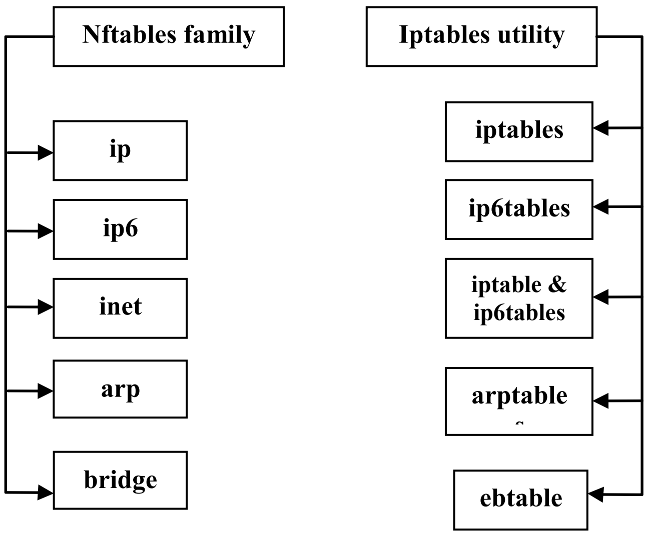

Figure 5. Shows iptables and nftables families

```
#libmnl /*It provides the interfaces between kernel and userspace via netlink*/
#libnftnl /*It provides the low level API to transform netlink message to objects*/
```

VI. CONCLUSION & FUTURE SCOPE

The present paper covers the up-to-date outline of existing algorithms reported in the literature for making effective software based Linux firewall using iptables. Using iptables there are many flaws exists such as code duplication, the problem in defining news rules, a single action can be performed using single rule, etc. whereas nftables covers all these above flaws existing in iptables. Basically, in this paper, a firewall is configured using which nftables helps to improve existing problems in Linux based firewall which achieves optimal level to detect malicious activities

In order to stop increasing malicious activities, packet filtering requires much better optimal solutions. Stateful firewall is the best solution to ensure such protection. The existence of errors, misconfiguration of firewall rules and code duplication is very likely to decrease the performance of firewall. In this paper, we proposed An Improved Approach of the Linux

Firewall Using a Hybrid Frame of Netfilter for Linux web server. While there are a number of paths that can be followed to provide a best malware detection method for firewall security. This work will be beneficial for small enterprise in terms of money and time using Netfilter/nftables makes it easy and simple to configure the strong firewall to solve the security related problems & detect malware using strong firewall rules to achieve optimal level. In future this presented work can be extended to an advanced level by detecting others layers of protocol. At present it only works on packet header but in future work can be done on other fields of packet.

REFERENCES

[1] "Cyber attacks hit banks: SBI blocks 6 lakh debit cards to ward off security threat," *http://indianexpress.com/article/business/banking-and-finance/cyber-attacks-hit-banks-sbi-blocks-6l-debit-cards-to-ward-off-security-threat-3092138/ [Accessed:4/27/2017].*

[2] H. Mao, "Current State and Future Development Trend of Firewall Technology," *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, vol. 2012, pp. 1–3, 2012.

[3] B. Wang, K. Lu, and P. Chang, "Design and Implementation of Linux Firewall Based on the Frame of Netfilter / IPtable," *The 11th International Conference on Computer Science & Education (ICCSE2016)*, pp. 949–953, 2016.

[4] J. P. Halguni and M. S. A. K. Rishna, "Design of a Firewall Based on Linux Netfilter using ARM9," *International Journal of Scientific Engineering and Technology Research*, vol. 4, no. 36, pp. 7744–7748, 2015.

[5] S. Wang and H. Chen, "Research on ASIC firewall based on state detection technology," *Applied Mechanics and Materials Trans Tech Publications,* vol. 650, pp. 3283–3286, 2014.

[6] T. Isohara, K. Takemori, and A. Kubota, "Kernel-based Behavior Analysis for Android Malware Detection," *Seventh International Conference on Computational Intelligence and Security*, pp. 1011–1015, 2011.

[7] W. Ma, P. Duan, S. Liu, G. Gu, and J. Liu, "Shadow Attacks : Automatically Evading System-Call-Behavior Based Malware Detection," *Journal in Computer Virology*, vol. 8, no. 1, pp. 1–13, 2012.

[8] O. S. Fingerprinting, "Packet Inspection for Unauthorized OS Detection in Enterprises," *IEEE Computer and Reliability Societies*, no. August, pp. 60–65, 2015.

[9] D. Dpul *et al.*, "The research and implementation of the Linux process real-time monitoring technology," *Information Sciences*, vol. 67, pp. 1046–1049, 2011.

[10] K. A, C. K, and A. K, "Framework for vulnerability reduction in real time intrusion detection and prevention systems using SOM based IDS with Netfilter-Iptables," *International Journal of Computer Science and Information Security*, 2010, pp. 229–233.

[11] D. Rovniagin and A. Wool, "The geometric efficient matching algorithm for firewalls," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 1, pp. 147–159, 2011.

[12] J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Boulahia, S. Martinez, and J. Cabot, "Management of stateful firewall misconfiguration," *Computers and Security*, vol. 39, pp. 64–85, 2013.

[13] K. Mindo, C. Sogomo, and N. M.Karie, "International Journal of Advanced Research in Computer Science and Software Engineering," *International Journal of Advance Research in Computer Science and Software Engineering*, vol. 3, no. 4, pp. 141–146, 2016.

[14] F. Shahzad, M. Shahzad, and M. Farooq, "In-execution dynamic malware analysis and detection by mining information in process control blocks of Linux OS," *Information Sciences*, vol. 231, pp. 45–63, 2013.

[15] W. Kehe, G. Yueguang, C. Wei, and Z. Tong, "The research and implementation of the Linux process real-time monitoring technology," *Fourth International Conference on Computational and Information Sciences*, pp. 1046–1049, 2012.