International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 26777-26788

http://www.ripublication.com

An Analytical Study on The Versatility of A Linux Based Firewall From A Security Perspective

Adwitiya Mukhopadhyay[#1],V.Srinidhi Skanda[#2],C.J.Vignesh[#3]
[#]Department of Computer Science, Amrita Vishwa Vidyapeetham, Mysore Campus, Mysore-570026, Karnataka, India .
[1]adwitiyamukhopadhyay@gmail.com, [2]skanda9051@gmail.com, [3]vigneshchinnapan@gmail.com

Abstract

Most corporations are seriously concerned about security of their networks. Hence in order to secure their network from outside intruders they install firewalls which will be frontier defense against attacks from outside networks. But present system will use firewalls that cost high and are not fully flexible (manageable with changing environment).This paper suggests implementation of firewalls which will be built upon open source Linux operating system. We leverage flexibility of netfilter framework towards various networking operations and make use of IP-tables that will allow system administrators to configure tables which contain a set of chains which are ordered and set of rules. We used ns3 simulator to showcase proposed system. By using ns3 code we designed and simulated different modules of system and we gave explanation for each module. Finally we concluded our work and gave small description about future work.

Keywords: network security, firewall, IP-tables, netfilter framework, chains, rules.

Introduction

With the Internet connections growing rapidly, network security has gained significant attention in both research and industrial communities. Due to the increasing threat of network attacks from Internet to the enterprise network, firewall has become important element in enterprise networks [1]. Firewalls have been the first line of defense for secure networks against attacks and unauthorized traffic by filtering out unwanted network traffics coming into or going from the secured network. Firewall controls and governs network access by allowing or denying the incoming or outgoing network traffics according to firewall policy rules written by system administrator. These rules are explicitly written and managed to filter out

any unwanted traffics coming into or going from the secure network [2]. However firewall used by corporations to secure their network are high cost and has less flexibility hence in this paper we propose configuring the firewall built on Linux operating system. Reason for this approach is that Linux is open source is built on UNIX operating system which is more secure and has multi-tasking ability. Another reason is netfilter, it is a framework located in Linux kernel will offers flexibility for various networking related operations to be implemented in form of customized handlers. netfilter offers various options for different functions like packet filtering, network address translation, and port translation. These functions provide the functionality required for directing packets through a network and provides ability to restrict packets entering into restricted network [3].

We use rule based firewall system called IP-tables which is a user-space tool that will parses command line and communicates a firewall policy to kernel. This tool will acts as a interface between user (here we are referring to system administrator who writes rule in command line) and kernel.

In this paper we are not doing comparison study since it is already proposed [4] [6] hence we study detailed information about structure and functionalities of our system. And finally we showcase our system by using ns3 simulator. We simulated different modules of our system and we gave detailed explanation for each module.

This paper is organized as follows. In section II we give description about existing system. In section III we give details about related works. In section IV we give introduction about netfilter and IP-tables. In section V we give examples of different firewall policies. In section VI we show our simulation details.

Exisiting System

Some corporations make use of firewall configured by windows or other internetworking operating system (IOS). These firewall systems are high cost and less flexible. Problem with the existing system [5] is when the network traffic is very high and networking environment changes frequently then the managing is very difficult hence we propose IP-tables firewall solution is integrated with the Linux operating system because of its features like robustness, reliability, flexibility and good scope for customization [6].

Below Table 1 displays difference between firewall configured by windows and other operating systems. This shows clearly advantage of using firewall configured by IP-tables.

Table 1: Comparison of Firewalls and IP-tables

Properties	**Windows Firewalls and Hardware Firewalls**	**IP-TABLES**
Cost efficiency	Available for Pricing or High Cost.	Free with all flavors of Linux.
Configuring	Works with IP and port blocking.	Works with each application.
Efficiency	Compromises are possible.	Highly Efficient.
Customization of Firewalls	Partial Endorsement.	Fully Customizable
LAN Security	Firewall filters malicious incoming traffic only after it hits the host Computer and the operating system.	The dedicated hardware firewalls you can get are actually just a small dedicated Linux box running IP tables, for the most part.
System administration intervention	Needs Administrator supervision to take care that nothing goes wrong.	Can be automated by postscripts and Custom Rules.
DMZ pinhole support	It can be created but more complex procedure and so much of surveillance is needed.	It provides DMZ pinhole creation with less surveillance

Many comparative studies are done that compares firewall based on windows and other internetworking operating system such as cisco IOS. Comparison is done based on the Table1.

Related Works

Performance of the system in different factors like cost, throughput, latency, security, configurability and user-friendliness. One such work is done by Su Wenhui and Xu Junjie "Performance Evaluation of Cisco ASA and Linux iptables Firewall Solution" [6] in this paper they done comparative study about Cisco ASA and Linux IP-tables firewall. Comparison is made based on the performance in factors like throughput, latency and concurrent Session. According to their result Linux IP-tables has stronger ability of handling burst, less expensive than equivalent Cisco ASA. Another paper "Windows vs. Linux: A Comparative Study" by Joe Cabrera [7] compares server configured by windows and Linux operating systems like previous study here also they are considering factors like cost, security, configurability and user-friendly. Result of comparison shows Linux operating system are secured, cost efficient, stable and allows maximum configurability features. These studies show that Linux

operating systems are more secured compare to windows and other internetworking operating systems. In this paper we gave detail description about structure and different modules of firewall configured using IP-tables and netfilter framework. Reason for this approach is studying internal structure and functionalities gives insight knowledge to system administrator about firewall system working.

Explanation Netfilter and Ip-Tables

1. Netfilter Explanation

Netfilter is a packet filtering framework inside Linux operating system of series Linux 2.4 and later kernel series. Above this framework a layer of different functionality of packet manipulation, IP-tables (a user-level tool) are constructed. Integrating this modules we construct a system that enables the Linux kernel to perform functionalities like firewalling, Network Address Translation [NAT], Port Address Translation [PAT], and many other useful functionalities that controls packet traffic incoming and outgoing through a network. Netfilter is implemented inside the Linux kernel. This framework allows callback functions to be attached different places in framework. Each of this callback functions invoked when packet traverse places where this functions are attached inside framework. These callback functions are implemented in kernel level, thus allowing IP-tables to inherit the flexibility of the Linux kernel module system [8].

2. Netfilter Architecture

Netfilter is series of hooks inside Linux kernel in each of this hook callback functions are registered with IPstack, a registered function is called for packets that traverses respective hook within IPstack. Multiple callbacks can be attached to each hook, and are called in order of precedence. Hooks are places in IPstack where packets are handed over to netfilter framework. Once a packet has been handed over to netfilter it contains functions that can permit packets to continue traversal, drop the packet if it not satisfies policy constraints, further manipulation of packets [8].

3. Netfilter Hooks Types

Netfilter framework contains hooks that are positioned in various points in protocol stack. In each of this hooks callback function are attached. When packet traverse each of this hook respective callback function are evoked.

1. NF_IP_PRE_ROUTING hook is called when incoming packets arrive at the system but before routing decisions are made. An important point to be noted here is that during NAT process packet address changed before routing decision implemented in this hook.
2. NF_IP_LOCAL_IN hook is called when incoming packet that has destination address of the node traverse. This hook generally called after routing decision is made.

3. NF_IP_LOCAL_OUT hook is traversed by the packet that is outgoing. This hook is placed before a routing decision has been made regarding an outgoing packet.
4. NF_IP_POST_ROUTING this hook is traversed by the packet that is outgoing. This hook is placed before a routing decision has been made regarding an outgoing packet [8].

4. Ip-Tables Explanation

IP-tables is logical set of layer implemented above netfilter framework layer. IP-Tables system consists of table. In its default setting, IP-tables has 3 types of tables in this paper we consider only filter table and NAT table. Chains of rules are associated within each of this table. In each of these chains IP-tables rules are processed in order.

4.1 Filter Table

Most commonly used table is filter table. In filter table firewalling rules are created. Filter table consists of three chains of rules this list of rules are called as firewall chains. This filter table is configured to a system that acts as firewall or packet filtering system. This firewall constitutes INPUT chains acts for the incoming packets that are destined to local server or host. FORWARD chains acts for the packet that are not destined for local server or system. OUTPUT chains are acts for packet that are destined out of firewall or packet filtering system that are coming from local server or host. Filter table will perform basic packet filtering functions like ACCEPT it will permit the packet traverse through it. DROP terminates packet and it will not send any ICMP message to the sender. REJECT terminates packet but unlike DROP it will send ICMP error message to sender. LOG causes kernel to log data about packets [4].

4.2 Nat Table

Like filter table this table also consists of 3 chains in its default settings. The first of these is the PREROUTING chain, which is called by the kernel before a routing decision is made about the packet. Second chain is POSTROUTING chain, which is called after routing decisions have been made about the packet. Finally there is the OUTPUT chain, which is processed on packets leaving the IP-tables machine [4]. Unlike filter table it includes additional functions like SNAT, DNAT and MASQUERADE. SNAT will manipulate packets source address. As you expects it happens in POSTROUTING. DNAT will manipulate packets destination address this will acts at PREROUTING. MASQUERADE maps the outgoing packet to the IP address of the interface on which it is leaving. It will acts only at POSTROUTING chain. Important point to be noted here is that MASQUERADE is used for dynamic NAT purpose.

Example For Different Firewall Policies

A. Input-Out Chains

Blocking of www.amazon.com accessed through port no 80
iptables -A OUTPUT -p tcp --dport 80 -d www.amazon.com -j DROP
Blocking of www.whatsapp.com accessed through port no 80
iptables -A OUTPUT -p tcp --dport 80 -d www.whatsapp.com -j DROP
Blocking of www.m.facebook.com accessed through port no 80
iptables -A OUTPUT -p tcp --dport 80 -d www.m.facebook.com.com - j DROP
Blocking of www.facebook.com accessed through port no 80
iptables -A OUTPUT -p tcp --dport 80 -d www.facebook.com -j DROP [].

B. Methods to track Attacker

In the mentioned method best thing is to drop the ICMP packets, by doing this we are not giving any clue to hacker whether the system is alive or not. Whereas if we do reject definitely hacker will come to know that ICMP packets are blocked and the system is live.

iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT.

Simulation

A. Simulating a netfilter:

The first three lines in Fig 1 shows the terminal output that specifies script is built successfully. The output of Fig1 gives information about hooks traversed by packet in client node. As soon as packet is created and sent out NF_INET_OUT is the first hook traversed by packet. NF_POST_ROUTING is the last hook traversed by the packet before leaving client node. Next line specifies that at time 2 second clients sent 512 bytes packet size to sever its IP address is 10.1.1.2 and its port number is 9. This attributes values and constraints can be set in script. Next 3 lines specifies hooks traversed by packets in node 2 i.e. gateway node. First packet traverses NF_INET_PRE_ROUTING hook before routing decision made. Next packets that are destined for other node traverse NF_INET_FORWARD hook. Then finally packet hits NF_INET_POST_ROUTING hooks. It leave nodes 1 i.e. gateway and travel towards node 2 server located outside public network. Next line specifies at time 2.00573 second server receives 512 bytes packet from client IP address 192.168.1.1. Next two lines describe packet hits NF_INET_LOCAL_OUT and NF_INET_POST_ROUTING hooks. Details of these hooks are already stated above. In next line server sends replay packet to client. Next lines describe same process as

the first few lines. Here node 1 acts as a gateway can be configured as firewall by attaching rules at particular hooks. For example we can write NAT rule at the NF_INET_PREROUTING hook before routing decision are made. Filtering rules configured in NF_INET_FORWARD hook.

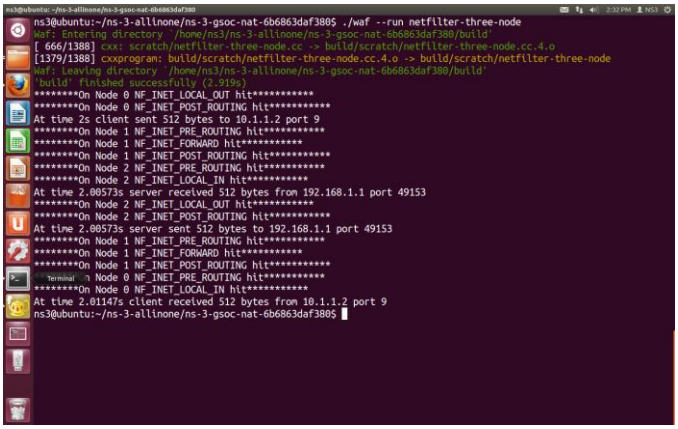

Figure 1:

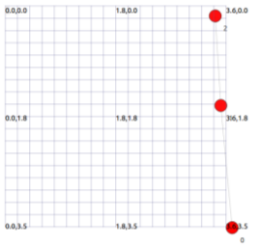

Figure 2(a):

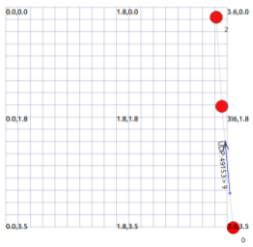

Figure 2(b):

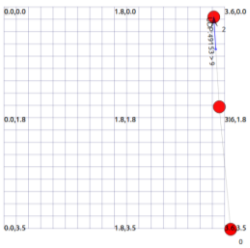

Figure 2(c):

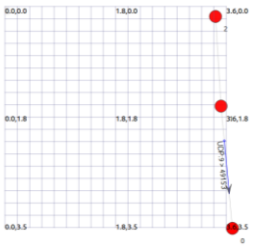

Figure 2(d):

Figure 2:

Fig1, 2(a)-2(d) shows the snapshots of working 3 nodes. Since we are not concerned about topology we took fewer nodes to simplify our explanation. Here we are assuming first node is client located inside a network, second node is gateway located between private network and Internet and third node is server located outside network.

B. Drop Trace

The first three lines in Fig 3 shows the terminal output that specifies script is built successfully. The following lines of Fig 3, 3.1 show the different reason for dropping the packets. Cases like DROP_TTL_EXPIRED, DROP_NO_ROUTE, DROP_ BAD_ CHECKSUM, DROP_INTERFACE_DOWN, DROP_ROUTE_ERROR, DROP_ FRAGMENT_TIMEOU, DROP_NF_DROP.

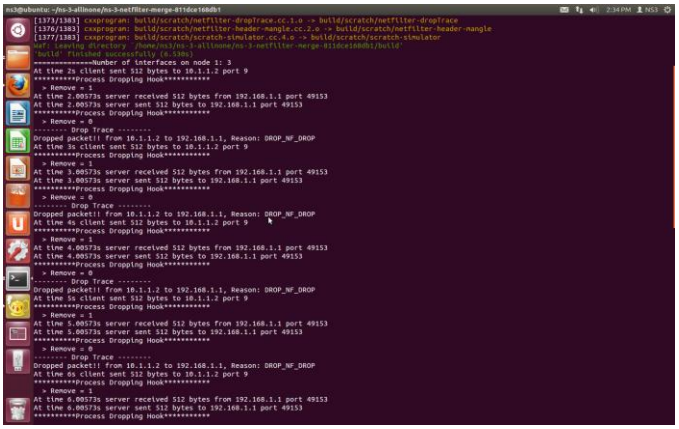

Figure 3:

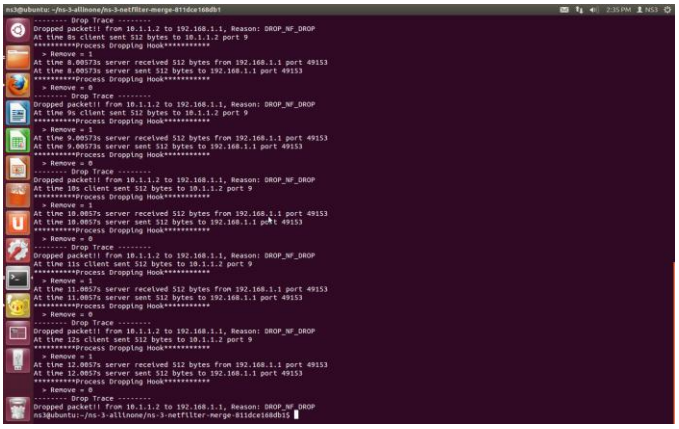

Figure 3.1:

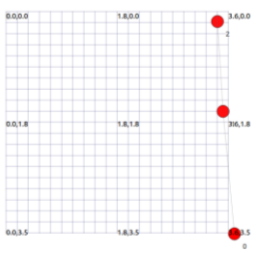

Figure 4(a):

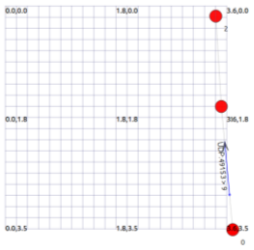

Figure 4(b):

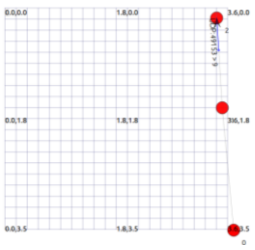

Figure 4(c):

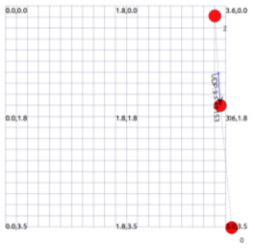

Figure 4(d):

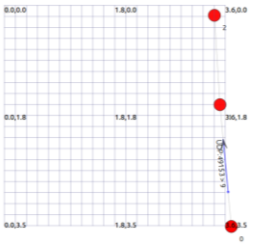

Figure 4(e):

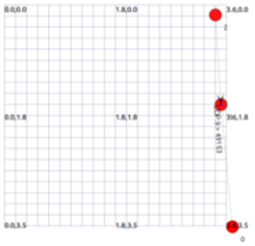

Figure 4(f):

Figure 4:

Fig 3, 3.1 and 4(a)-4(f) shows different cases where packets are dropped. Here also we are considering same 3 node network topology.

C. IPv4 Static Nat

The first three lines in Fig 5 shows the terminal output that specifies script is built successfully. Following lines of Fig 5 specifies the action of Static NAT. At time 2 seconds client which is in private network sends packet size of 512 bytes to server IP address 203.82.48.2 and port number is which is located in outside private network. At 2.00573 second server node receives packet sent from client. Here client address changed to 203.82.4.100 this is because we wrote rule to change public IP address into private address and port number. At time 2.00573 second server sent replay packet to client address 203.82.4.100 at port 8080 this public IP address is mapped to private IP address 192.168.1.1. Rule is implemented as callback chain at NF_INET_PREROUTING hook. Therefore whenever packet traverse at this hook callback chain is invoked to perform NAT functionalities. This hook situated before routing decision is made.

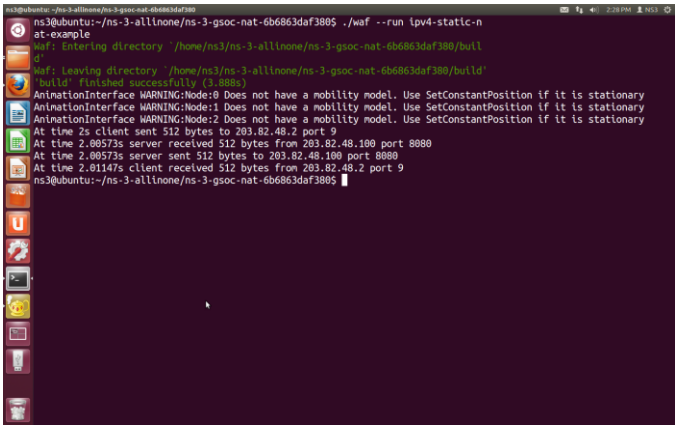

Figure 5:

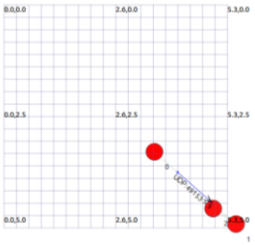

Figure 6(a):

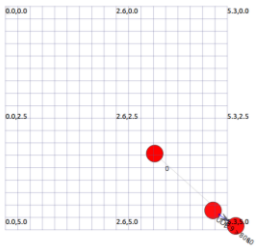

Figure 6(b):

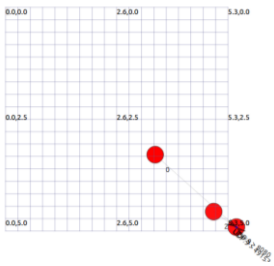

Figure 6(c):

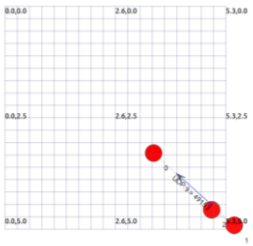

Figure 6(d):

Figure 6:

Fig 5 shows output for Static NAT. Fig 6(a)-6(d) are the snapshots of ipv4 Static NAT. Here also we consider 3 node network topology.

D. IPv4 Dynamic NAT

The first three lines in Fig 7 shows the terminal output that specifies script is built successfully. Following lines of Fig 7 specifies the action of Dynamic NAT. The following lines of Fig 7 specifies that client at 2 second sends packet size of 512 bytes to sever located outside network IP address 203.82.48.2 and port number is 9. Next few line specifies range port address that server handles. Here we took port

number range from 49153 to 49163. Dynamic NAT device dynamically allocates the port address within the mentioned range.

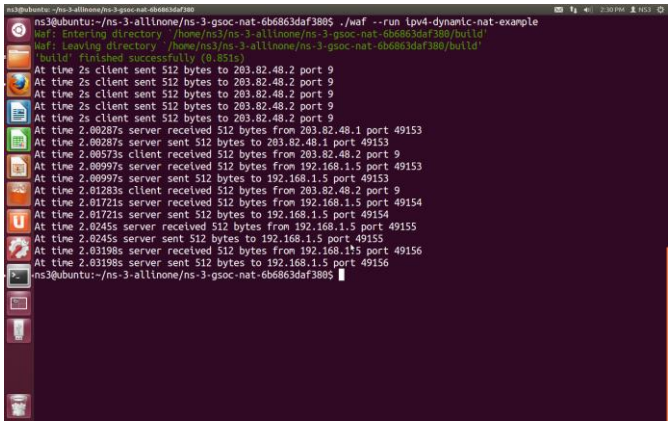

Figure 7:

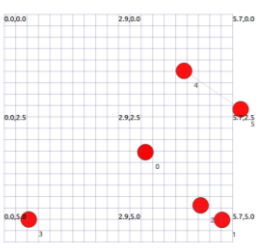

Figure 8(a):

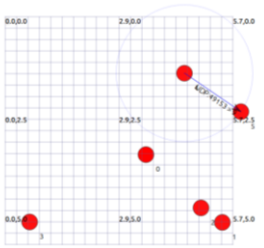

Figure 8(b):

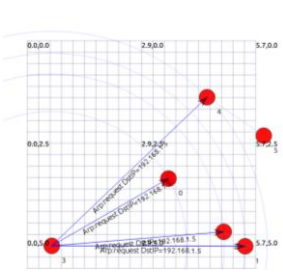

Figure 8 (c):

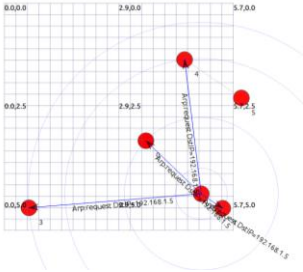

Figure 8(d):

Figure 8:

Fig 7 shows Dynamic NAT functionalities. Fig 8(a) - 8(d) show the snapshots of the IPV4 Dynamic NAT. Here we consider hybrid topology which includes CSMA nodes.

References

[1] Al-Shaer, Ehab S., and Hazem H. Hamed. "Firewall policy advisor for anomaly discovery and rule editing." In *Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on*, pp. 17-30. IEEE, 2003.

[2] Golnabi, Korosh, Richard K. Min, Latifur Khan, and Ehab Al-Shaer. "Analysis of firewall policy rules using data mining techniques." In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pp. 305-315. IEEE, 2006.

[3] http://en.wikipedia.org/wiki/Netfilter

[4] Petreley, Nicholas. "Security report: Windows vs linux." *The Register* 22 (2004): 1-24.

[5] Jaferian, Pooya. "Usability study of Windows Vista's firewall." *EECE512 course project at the University of British Columbia* (2008).

[6] Xu, Junjie, and Wenhui Su. "Performance Evaluations of Cisco ASA and Linux IPTables Firewall Solutions." (2013).

[7] Cabrera, Joe. "Windows vs. Linux: A Comparative Study." (2009).

[8] Jones, Alan. "Netfilter and IPTables–a structural examination." *SANS Institute Reading Room site* (2004).

[9] Kong, Jiejun, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang. "Providing robust and ubiquitous security support for mobile ad hoc networks." In *2012 20th IEEE International Conference on Network Protocols (ICNP)*, pp. 0251-0251. IEEE Computer Society, 2001.

[10] Perkins, Charles E., and Elizabeth M. Royer. "Ad-hoc on-demand distance vector routing." In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA'99. Second IEEE Workshop on*, pp. 90-100. IEEE, 1999.

[11] Sanzgiri, Kimaya, Daniel LaFlamme, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. "Authenticated routing for ad hoc networks." *Selected Areas in Communications, IEEE Journal on* 23, no. 3 (2005): 598-610.

[12] Al-Salihy, W., and M. Ganesan. "Security Enhancement for an Infrastructure Wireless Domain."

[13] Cryptography and Network Security by William Stallings 5th edition.

[14] Kiesel, Sebastian, and Jochen Kögel. "An operating system independent API for firewall control: design and implementation for Linux." *Other IFIP Publications* 1 (2011).

[15] Marmorstein, Robert M., and Phil Kearns. "An Open Source Solution for Testing NAT'd and Nested iptables Firewalls." In *LISA*, vol. 5, pp. 103-112. 2005.