

**HO CHI MINH CITY NATIONAL UNIVERSITY**

UNIVERSITY OF INFORMATION TECHNOLOGY



**COURSE PROJECT**

**WIRELESS EMBEDDED NETWORK SYSTEMS**

**IMPLEMENTING NETWORK SECURITY WITH**

**NETFILTER/IPTABLES ON VIRTUAL MACHINES**

**GROUP 17**

Supervisor – PSG.TS Le Trung Quan

Students:

21522323 – Tran Thuy Hien Mai

21522659 – Nguyen Ngoc Thanh Thuy

**HO CHI MINH CITY, 2023**

**TABLE OF CONTENTS**

[**CHAPTER 1: INTRODUCTION** 6](#_heading=h.gjdgxs)

[**CHAPTER 2: LITERATURE REVIEW/RELATED WORK** 6](#_heading=h.30j0zll)

[2. 1: EVOLUTION OF FIREWALL TECHNOLOGIES 6](#_heading=h.1fob9te)

[2. 2: HISTORY OF NETFILTER/IPTABLES 6](#_heading=h.3znysh7)

[2. 3: NETFILTER AND ITS ROLE IN LINUX-BASED FIREWALLS 7](#_heading=h.2et92p0)

[2. 4: UNDERSTANDING IPTABLES AS A USER-SPACE UTILITY 7](#_heading=h.tyjcwt)

[2. 5: THE INTERPLAY OF FIREWALL, NETFILTER, AND IPTABLES 7](#_heading=h.3dy6vkm)

[2. 6: SIGNIFICANCE OF NETFILTER AND IPTABLES IN LINUX-BASED FIREWALLS 8](#_heading=h.1t3h5sf)

[**CHAPTER 3: METHODOLOGY** 9](#_heading=h.4d34og8)

[3. 1: NETFILTER 9](#_heading=h.2s8eyo1)

[3.1 1: Callback function 10](#_heading=h.17dp8vu)

[3.1 2: Netfilter hooks 10](#_heading=h.3rdcrjn)

[3.1 3: Netfilter action 12](#_heading=h.1ksv4uv)

[3. 2: IPTABLES 13](#_heading=h.2jxsxqh)

[3.2 1: Iptables tables 13](#_heading=h.z337ya)

[3.2 2: Iptables chains 14](#_heading=h.4i7ojhp)

[3.2 3: Iptables targets 15](#_heading=h.qsh70q)

[3.2 4: Iptables command structure 16](#_heading=h.1pxezwc)

[3.2 5: Iptables matches options 16](#_heading=h.49x2ik5)

[**CHAPTER 4: CODE ANALYSIS/DEBUGGING/TRACING/LOGGING, DEMO SCENARIOS, EXPLANATIONS ON RESULT** 17](#_heading=h.2p2csry)

[4. 1: ENVIRONMENT SETUP USING CONTAINER 17](#_heading=h.147n2zr)

[4. 2: INSTALL NETFILTER AND IPTABLES 20](#_heading=h.3o7alnk)

[4. 3: SCENARIO 1: PROTECT THE VM MACHINE 20](#_heading=h.23ckvvd)

[4.3 1: Code 21](#_heading=h.ihv636)

[4.3 2: Debugging\_Results 23](#_heading=h.32hioqz)

[4. 4: SCENARIO 2: PROTECT THE INTERNAL NETWORK 192.168.60.0/24 24](#_heading=h.1hmsyys)

[4.4 1: Set rule Iptables 26](#_heading=h.41mghml)

[4.4 2: Results 27](#_heading=h.2grqrue)

[**CHAPTER 5: CONCLUSION** 29](#_heading=h.3fwokq0)

**LIST OF ACRONYMS/ABBREVIATIONS**

| GPL | General Public License |
| --- | --- |
| Ipfw | Internet Protocol Firewall |
| BSD | Berkeley Software Distribution |
| IP | Internet Protocol |
| NAT | Network Address Translation |
| UFW | Uncomplicated Firewall |
| FOSS | Free and Open Source |
| LKM | Loadable Kernel Module |
| VM | Virtual Machine |
| CLI | Command-line Interface |
| NF | NetFilter |
| ICMP | Internet Control Message Protocol |
| TCP | Transmission Control Protocol |
| GPL | General Public License |

**LIST OF FIGURES**

Figure 1 Netfilter hooks in Ipv4

Figure 2 Iptables tables

Figure 3 Network packet traversal through iptables

**LIST OF TABLES**

Table 1 Netfilter hooks

Table 2 Return codes of Netfilter callback functions

Table 3 Iptables tables

Table 4 Iptables chains

Table 5 Different types of targets in iptables

**CHAPTER 1: INTRODUCTION**

In modern networking, the imperative need to safeguard digital environments against unauthorized access and malicious threats has propelled the evolution of firewalls, fundamental guardians of network security. A firewall is a combination of hardware and software that monitors and controls network traffic based on determined rules, allowing some packets to pass and blocking others. It establishes a barrier between a trusted network and an unauthorized network.

At the heart of Linux-based firewall solutions lies Netfilter, a component within the kernel, offering a strong framework for packet filtering, address translation, and manipulation. Complementing this framework is iptables, a flexible user-space utility to configure rules, control traffic flow, and fortify network defenses. Together, they empower administrators to establish security architectures, enforce policies, and traffic control over the traversal of data packets within Linux-based networks. Understanding the coordination between firewalls, Netfilter, and iptables is crucial in discovering network security.

**CHAPTER 2: LITERATURE REVIEW/RELATED WORK**

2. 1: EVOLUTION OF FIREWALL TECHNOLOGIES

The initial firewalls, categorized as 'network layer' firewalls, had their origins in the 1980s. They were developed by Cisco Systems and Digital Equipment Corporation. These firewalls’ main functionality is just filtering packets based on basic criteria such as source, destination, and connection type. Subsequently, a shift occurred in the early 1990s with the advent of 'application layer' firewalls. As we entered the early 21st century, a prevalent trend emerged-firewalls evolved into hybrids, merging the strengths of both network layer and application layer, aiming to combine speed and depth in protecting networks.

2. 2: HISTORY OF NETFILTER/IPTABLES

- 1998: Rusty Russell started the Netfilter/Iptables project. He was also the original author of the project's precursor named "ipchains."

- 1999: As the project progressed, Russell established the Netfilter Core Team. The software they developed, Netfilter, adopted the General Public License (GPL). In March 2000, it was integrated into version 2.4.x of the Linux kernel. August 2003: Harald Welte became the president of the Netfilter Core Team.

- April 2004: Due to GPL violations by a router manufacturer (Sitecom Germany), a German court issued a historic injunction against Sitecom Germany. The company had refused to comply with GPL terms, leading to a disruption in the project.

- September 2007: Patrick McHardy, who had led the development in the preceding years, was elected as the new leader of the Netfilter Core Team.

- Before iptables: Prior to iptables, the main software packages for creating Linux firewalls were ipchains in Linux kernel 2.2.x and ipfwadm in Linux kernel 2.0.x. Both were based on BSD's ipfw. Both ipchains and ipfwadm manipulated network code to handle data packets, as Linux lacked a unified packet control framework until the advent of Netfilter.

2. 3: NETFILTER AND ITS ROLE IN LINUX-BASED FIREWALLS

- Kernel-Level Framework: Netfilter is an essential framework for packet processing within the Linux kernel. Netfilter provides features such as packet filtering, network address translation NAT, and port translation.

- Hooks in the Kernel: Netfilter provides hooks at different points in the network stack, allowing it to inspect and manipulate packets as they traverse through the network subsystem.

2. 4: UNDERSTANDING IPTABLES AS A USER-SPACE UTILITY

- "Iptables" is a user-space utility in Linux that allows administrators to configure and interact with the Netfilter firewall infrastructure. It provides a command-line interface for administrators to define rules that govern how packets are treated.

- Configuration of Netfilter Rules: Iptables allows the configuration of rules in the Netfilter tables (filter, NAT, and mangle) and chains. These rules specify the conditions under which packets should be accepted, rejected or modified.

2. 5: THE INTERPLAY OF FIREWALL, NETFILTER, AND IPTABLES

- The interplay of Firewall, Netfilter, and iptables is a fundamental aspect of network security on Linux systems.

- Interplay and Workflow:

* Packet Arrival: When a packet arrives at the Linux kernel, it goes through the Netfilter hooks. Netfilter triggers these hooks at various points in the packet processing pipeline, such as PREROUTING, INPUT, FORWARD, OUTPUT, and POSTROUTING.
* Netfilter Processing: Netfilter examines the packet based on the rules configured using iptables. The packet is matched against the rules in the appropriate Netfilter table and chain.
* Iptables Configuration: Iptables, as a user-space utility, allows administrators to define rules and policies. These rules include conditions (matches) and actions (targets).
* Action on Matches: If a packet matches a rule, the specified action (target) is applied. Common actions include ACCEPT, DROP, REJECT, and forwarding to another rule or chain.
* Firewall Enforcement: The configured rules act as the policies of the firewall, determining which packets are allowed and which are denied. The firewall, based on Netfilter rules set by iptables, acts as a security gateway, controlling traffic flow according to the defined policies.

2. 6: SIGNIFICANCE OF NETFILTER AND IPTABLES IN LINUX-BASED FIREWALLS

Netfilter and iptables play a crucial role in the implementation of firewalls on Linux-based systems. Their significance lies in providing a robust and flexible framework for controlling network traffic, enforcing security policies, and protecting systems from unauthorized access:

* Packet Filtering and Security:

+ Granular Control: Netfilter, combined with iptables, allows administrators to define rules for packet filtering based on various criteria such as source and destination IP addresses, ports, and protocols.

+ Security Policies: Iptables rules enable the creation of security policies, determining which packets are permitted and which are denied. This granular control is essential for enforcing security best practices.

* Network Address Translation (NAT): IP Masquerading: Netfilter's NAT functionality, configurable through iptables, enables IP masquerading. This is particularly useful for home or small office networks, allowing multiple internal devices to share a single public IP address.
* Connection Tracking: Stateful Inspection: Netfilter maintains a connection tracking table, allowing for stateful inspection of network connections. This enhances security by tracking the state of established connections and ensuring that only legitimate traffic is allowed.
* Dynamic Rule Configuration: Real-time Adjustments: Iptables enables administrators to dynamically adjust firewall rules without requiring a system restart. This flexibility is crucial for adapting to changing network conditions and responding to security incidents in real-time.
* Integration with Users-pace Tools: User-Friendly Interface: Iptables provides a user-friendly command-line interface for managing firewall rules. Additionally, user-space tools like firewall and UFW offer higher-level abstractions, making it easier for administrators to configure and manage firewalls.
* Logging and Monitoring: Audit Trails: Iptables supports logging capabilities, allowing administrators to create audit trails for network activity. This is valuable for monitoring and investigating security incidents by reviewing logged information.
* Core Component of Linux Kernel: Inherent to Linux Systems: Netfilter is an integral part of the Linux kernel, ensuring that firewall functionality is seamlessly integrated into Linux-based operating systems. This native integration provides efficiency and reliability.
* Compatibility and Extensibility: Integration with Applications: Netfilter and iptables are designed to be compatible with various applications and protocols. They can be extended with additional modules to support specific functionalities and meet the requirements of diverse network environments.
* Community Support and Documentation: Widespread Adoption: Due to their widespread adoption, Netfilter and iptables benefit from a large community of users and developers. This leads to extensive documentation, community support, and a wealth of knowledge available for administrators.

**CHAPTER 3: METHODOLOGY**

NetFilter stands as a FOSS project, offering packet filtering software for Linux systems since kernel version 2.4. Its key functionalities include stateless and stateful packet filtering for both IPv4 and IPv6, NAT, Connection track, and additional packet manipulation tools.

The software platform utilized for implementing, configuring, and testing the operation of Netfilter/Iptables is a virtual machine running Ubuntu version 20.04 on VMWare. This platform facilitates a comprehensive understanding and analysis of these networking tools.

To implement a simple packet filtering firewall, the process involves writing a LKM using the Netfilter framework. This LKM enables interception and processing of network packets within the Linux kernel, allowing for the establishment of packet filtering rules.

3. 1: NETFILTER

A custom LKM is developed using programming languages like C and Netfilter macros. Primarily, NetFilter provides a set of hooks that allow kernel modules to register callback functions with the network stack.

### 3.1 1: Callback function

- A callback function is a user-defined function that is passed as an argument to another function and is executed after a particular operation or event.

- A callback function is invoked when each packet passes through the specific hook that was initially registered within the network stack. The functions of Netfilter provide guidance for packets traversing a network and prevent unwanted packets within a computer network.

3.1 2: Netfilter hooks

- A hook refers to a series of techniques which is used to modify or enhance the behavior of an operating system, applications, or other software components by intercepting and responding to functions such as calls, messages, or events between software components.

- Netfilter hooks act as entry points within the kernel's networking processing flow. They enable developers to attach their code to certain stages of packet processing, such as when a packet enters the system (ingress) or leaves the system (egress), allowing them to inspect, modify, or control the packet flow based on specific criteria.

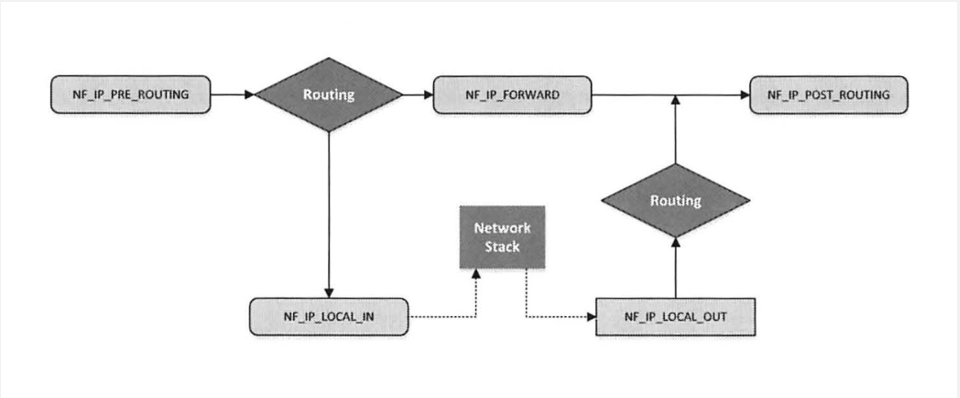
- Developers often use these hooks to implement firewall rules, perform packet filtering, NAT, or other packet-related operations by registering their functions or modules to execute at these specific interception points within the networking stack.

- Netfilter provides five IPv4 hooks which are essentially predefined points in the Linux kernel's networking stack where custom code or modules can be attached to intercept and manipulate network packets. When a packet traverses the networking stack, it encounters these hooks, which trigger the execution of registered functions or modules.

- Each following hook represent a different stage in the packet processing path:

| **Hook** | **Trigger** |
| --- | --- |
| NF\_INET\_PRE\_ROUTING | Triggers when an incoming packet arrives at the network interface and before the kernel performs any routing. It's the earliest point at which packets can be intercepted and manipulated. Actions at this stage can include packet filtering, altering packet headers, or performing NAT. |
| NF\_INET\_LOCAL\_IN | Triggers when the incoming packet goes through routing. This hook decides whether the packet is for other machines or for the host itself. Firewalls often apply rules at this stage to determine whether to accept or drop packets destined for the local system. |
| NF\_INET\_FORWARD | Triggers when the packet is determined to be forwarded to another host or through the local system. Firewall rules related to packet forwarding between different networks are applied here. |
| NF\_INET\_LOCAL\_OUT | Triggers when a packet is destined to leave this host. It allows manipulation of outgoing packets, such as modifying headers or applying filtering rules before the packets are sent out. |
| NF\_INET\_POST\_ROUTING | Triggers when outgoing or forwarded traffic after routing has taken place and just before handing off to the network interface for transmission. At this final point, actions often involve NAT of the source IP addresses of outgoing packets. |

*Table 1: Netfilter hooks*

**

*Figure 1:* *Netfilter hooks in Ipv4*

3.1 3: Netfilter actions

- After the return functions have completed their processing of packets, they must provide decisions regarding whether to DROP or ACCEPT the packet. These decisions are predefined within Netfilter using the following return codes:

| **Return Code** | **Meaning** |
| --- | --- |
| NF\_DROP | This return code instructs the system to discard the packet. The packet processing stops immediately |
| NF\_ACCEPT | This return code allows a packet to continue through the networking stack. The packet proceeds to the next hook or continues its journey based on subsequent rules or configurations. |
| NF\_STOLEN | This return code temporarily removes the packet from regular processing but retains it within the kernel's resources. It's a way to signal that the packet has been taken over or claimed by an extension or module. |
| NF\_QUEUE | Packets are directed into a queue for further processing by user-space applications or firewall software. It allows additional inspection or handling of packets outside the kernel. |
| NF\_REPEAT | This return code prompts the kernel to reprocess the packet. It can trigger the packet to revisit the same hook or undergo a specific processing sequence again. |
| NF\_STOP | When a packet receives this code, it means that further processing or passing through additional hooks should be stopped. The packet is held at the current hook, bypassing any remaining stages in the networking stack. |

*Table 2 Return codes of Netfilter callback functions*

3. 2: IPTABLES

- We can build a simple firewall using netfilter. Furthermore, Linux already has a built-in firewall based on Netfilter.

- Iptables is a widely used firewall tool that interacts with the Netfilter packet handling of the Linux kernel. It utilizes the command-line interface (CLI) in Linux. Users interact with iptables by using specific command-line tools to configure firewall rules and manage packet filtering.

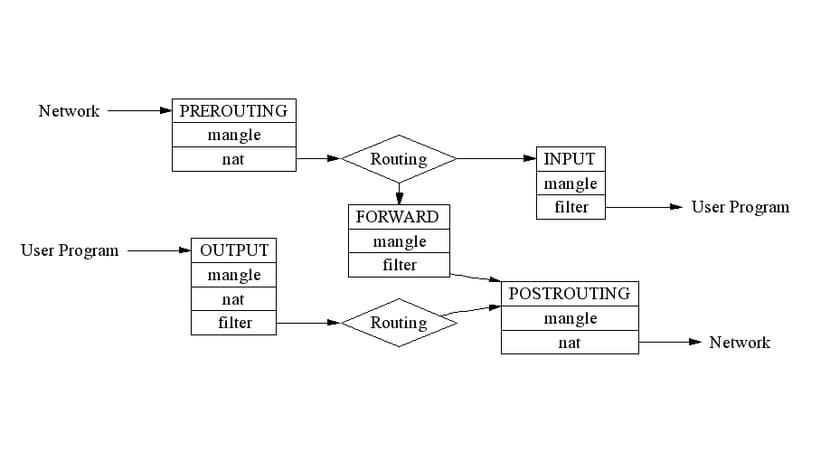
- Iptables operates by interfacing with hooks within the Linux kernel's network stack. Essentially, iptables is responsible for communicating with users and subsequently pushing user-defined rules to Netfilter for processing. Fundamentally, iptables is built of three core components: tables, chains, and targets.

3.2 1: Iptables tables

- The iptables firewall uses tables to organize its rules. These tables categorize the rules based on the type of decision they are meant to be used for. For instance, a rule will be added to the NAT table if it copes with NAT. If the rule is used to determine whether to permit the packet to go to its destination, it would most likely be added to the filter table. There are some tables to be mentioned here: Filter table, NAT table and Mangle table.

| Table | Functionality | Chains |
| --- | --- | --- |
| Filter | Packet filtering | INPUT: Manages incoming packets destined for the local system.  OUTPUT: Handles outgoing packets generated by the local system.  FORWARD: Controls packets passing through the system (neither for nor from the local machine). |
| NAT | Modifying source or destination network address | PREROUTING: Applies NAT before the routing decision.  POSTROUTING: Applies NAT after routing.  INPUT:  OUTPUT: Used for locally generated packets. |
| Mangle | Alters or modifies packet headers for specialized purposes | PREROUTING: Applies modifications before routing decisions.  INPUT:  FORWARD  OUTPUT  POSTROUTING: Allow modifications to be applied for packets before they leave the local system |

*Table 3: Iptables tables*



*Figure 2 Iptables tables*

- Each iptables table is organized with distinct “chains”. Adding chains to a table enables the processing of packets at different stages, determining when to handle packets.

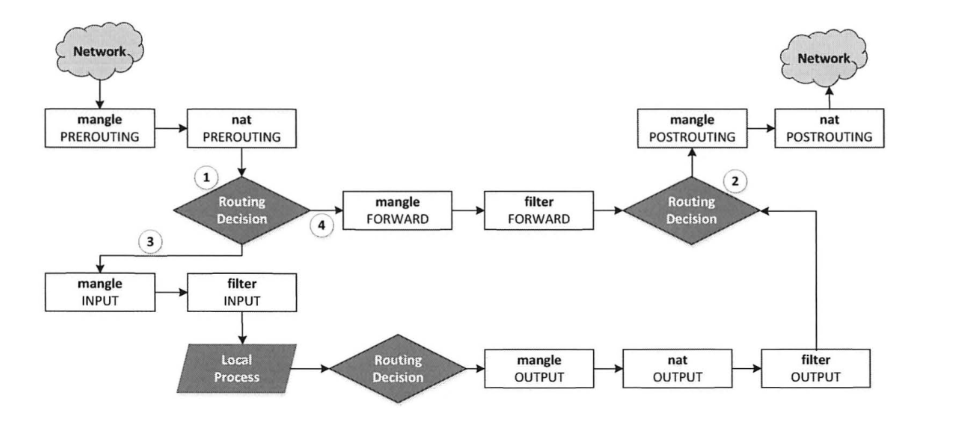
3.2 2: Iptables chains

- There are five chains in iptables: PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING. Each chain corresponds to specific Netfilter hooks they link to, determining when in the packet's journey through the networking stack the rules within the chain are applied.

| **Netfilter hook** | **Iptables chains name** | **Description** |
| --- | --- | --- |
| NF\_IP\_PRE\_ROUTING | PREROUTING | Triggers when a packet arrives at the network interface and before any routing decisions are made. |
| NF\_IP\_LOCAL\_IN | INPUT | Triggers when a packet’s destination IP address matches this machine. |
| NF\_IP\_FORWARD | NAT | Triggers when the packet is determined to be destined for another host and needs to be forwarded through the local system. |
| NF\_IP\_LOCAL\_OUT | OUTPUT | Triggers when a packet, originating from the machine, is leaving the machine. |
| NF\_IP\_POST\_ROUTING | POSTROUTING | Triggers when any packet (regardless of origin) is leaving the machine. |

*Table 4 Iptables chains*

- Presentation formula: [Iptables chain name column] chain is linked to [Netfilter hook column] hook



*Figure 3**: Network packet traversal through iptables*

3.2 3: Iptables targets

In iptables, targets are actions specified within rules that determine what happens to packets that match specific criteria.

| **Target** | **Meaning** |
| --- | --- |
| -j ACCEPT | Allows the packet to proceed, permits packets to continue their journey through the firewall |
| -j DROP | Discards the packet like it never exits without sending any notification or error message to the sender |
| -j REJECT | Discards the packet and sends an error message to the sender |
| -j LOG | Logs the packet information to the system log files |
| -j RETURN | Exit a specific chain and return to the calling chain for further processing |

*Table 5 Different types of targets in iptables*

3.2 4: Iptables command structure

sudo iptables <action with chains/rules> -t <table> -p <protocol> -i <input interface> -o <output interface> -s <IP source> -d <IP destination> --sport <source port number> --dport <destination port number.> -j <target>

3.2 5: Iptables matches options

-A <chain name>: Add a new rule at the end of a specific chain within iptables

- D <chain name>: Delete (remove) a rule from a specific chain in iptables.

-i <interface name>: Denotes the input interface receive packet

-o <interface name>: Specify the output interface that sends packets outward

-p <protocol>: Define the protocol that the rule will match

-j <target>: (jump) Determine the action/target to take if a packet matches the rule

-s <source IP address>: Specifies the "source" IP address, allowing rules to match packets originating from a particular source IP

-d <destination IP address>: Filter packets based on their destination IP address

--sport <source port>: Filter packets based on their destination IP address

--dport <destination port>: Filter packets based on their destination port number

-L: List all rules of all chains in iptables

**CHAPTER 4: CODE ANALYSIS/DEBUGGING/TRACING/LOGGING, DEMO SCENARIOS, EXPLANATIONS ON RESULT**

In examining two distinct firewall deployment scenarios within a Linux environment, the first scenario involves building a firewall as a loadable kernel module employing the Netfilter framework, which approaches directly into the Linux kernel. This firewall can enforce rules and policies to manage data flow effectively. On the other hand, the second scenario is about using iptables, a robust tool native to the Linux operating system, to create a straightforward firewall.

4. 1: ENVIRONMENT SETUP USING CONTAINER

**Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, hàng

Mô tả được tạo tự động**

Step 1: Write the docker-compose.yml file to configure containers

Ảnh có chứa văn bản, phần mềm, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, phần mềm, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

Step 2: Build and start the containers

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

4. 2: INSTALL NETFILTER AND IPTABLES







4. 3: SCENARIO 1: PROTECT THE VIRTUAL MACHINE

Write a Netfilter kernel module to prevent other machines from pinging the VM, and to telnet ( telnet’s default port 23) into the VM

- The result before applying the blocking module:

+) Host A - 10.9.0.5 pings to VM machine 10.9.0.1 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host A - 10.9.0.5 telnets to VM machine 10.9.0.1 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

4.3 1: Code

- Code in file netfilter\_kernel\_module.c

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

- Header Includes: Import necessary kernel and Netfilter-related header files.

- Callback Functions:

+) block\_ping: Checks incoming packets and drops ICMP (ping) packets destined for the IP address 10.9.0.1.

+) block\_telnet: Checks incoming TCP packets and drops packets destined for port 23 (Telnet port) targeting the IP address 10.9.0.1.

- Netfilter Hooks Registration:

+) setupFilter: Registers the block\_ping and block\_telnet functions as Netfilter hooks for the NF\_INET\_PRE\_ROUTING stage, which intercepts packets before routing. These hooks have the highest priority (NF\_IP\_PRI\_FIRST) in the Netfilter chain.

- Initialization and Exit Functions:

+) module\_init(setupFilter): Initializes the module by registering the Netfilter hooks.

+) module\_exit(removeFilter): Cleans up and unregisters the Netfilter hooks when the module is removed.

- Module Information:

+) MODULE\_LICENSE("GPL"): Specifies the license of the module as GPL, indicating it's open source.

- Makefile:

Ảnh có chứa văn bản, phần mềm, Trang web, Biểu tượng máy tính

Mô tả được tạo tự động

Build:

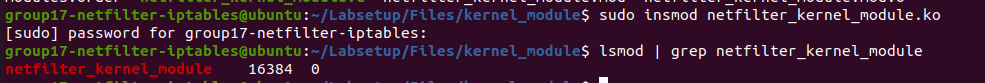
- Build netfilter\_kernel\_module.c file to get netfilter\_kernel\_module.ko file

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động



- Insert this module:



4.3 2: Debugging\_results



- The debug result after pinging from Host A 10.9.0.5 to VM 10.9.0.1: ping unsuccessfully

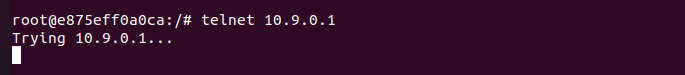
Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động

- The debug result after pinging from Host A 10.9.0.5 to VM 10.9.0.1: ping unsuccessfully



Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

4. 4: SCENARIO 2: PROTECT THE INTERNAL NETWORK 192.168.60.0/24

Implement iptables rules on the router to protect the internal network 192.168.60.0/24, following these restrictions:

(1) Outside hosts cannot ping internal hosts but can ping the router.

(2) Internal hosts can ping outside hosts.

(3) All other packets between the internal and external networks should be blocked

- The result before applying iptables rules

+) Host A - 10.9.0.5 pings to Router 10.9.0.11 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngs

+) Host A - 10.9.0.5 of network 10.9.0.0/24 pings to Host 1 - 192.168.60.5 of network 192.168.60.0/24 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host A - 10.9.0.5 of network 10.9.0.0/24 pings to Host 2 - 192.168.60.6 of network 192.168.60.0/24 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host A - 10.9.0.5 of network 10.9.0.0/24 pings to Host 1 - 192.168.60.7 of network 192.168.60.0/24 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host 1 - 192.168.60.5 of network 192.168.60.0/24 pings to Host A - 10.9.0.5 of network 10.9.0.0/24 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

+) Host A - 10.9.0.5 of network 10.9.0.0/24 telnets to Host 1 - 192.168.60.5 of network 192.168.60.0/24 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

+) Host 1 - 192.168.60.5 of network 192.168.60.0/24 telnets to Host A - 10.9.0.5 of network 10.9.0.0/24 successfully.

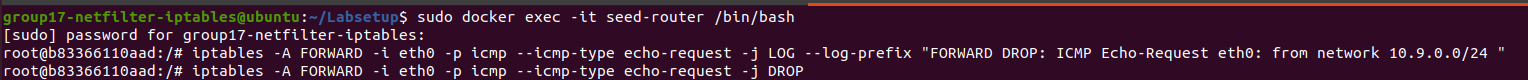
Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

4.4 1: Set iptables rule

- Set rules that outside hosts (from the network 10.9.0.0/24) cannot ping internal hosts (in the network 192.168.60.0/24) but can ping the router:

+) Log and drop ICMP echo-request packets coming through the eth0 interface.



- Set rules that internal hosts (in the network 192.168.60.0/24) can ping outside hosts (in the network 10.9.0.0/24)

+) Log and accept ICMP echo-request packets coming through the eth1 interface.



+) Log and accept ICMP echo-reply packets coming through the eth0 interface.



- Block all other packets between the internal and external networks which don’t match the rules

+) Set the default policy of the FORWARD chain to DROP and log dropped packets.



4.4 2: Results

- Check rules in the FORWARD tables

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

- The result after applying iptables rules

+) Host A - 10.9.0.5 pings to Router 10.9.0.11 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host A - 10.9.0.5 of network 10.9.0.0/24 cannot ping to Host 1 - 192.168.60.5 of network 192.168.60.0/24.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host A - 10.9.0.5 of network 10.9.0.0/24 cannot ping to Host 2 - 192.168.60.6 of network 192.168.60.0/24.

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động**

+) Host A - 10.9.0.5 of network 10.9.0.0/24 cannot ping to Host 3 - 192.168.60.7 of network 192.168.60.0/24.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host 1 - 192.168.60.5 of network 192.168.60.0/24 pings to Host A - 10.9.0.5 of network 10.9.0.0/24 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host 2 - 192.168.60.6 of network 192.168.60.0/24 pings to Host A - 10.9.0.5 of network 10.9.0.0/24 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host 3 - 192.168.60.7 of network 192.168.60.0/24 pings to Host A - 10.9.0.5 of network 10.9.0.0/24 successfully.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host A - 10.9.0.5 of network 10.9.0.0/24 cannot telnet to Host 1 - 192.168.60.5 of network 192.168.60.0/24.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

+) Host 1 - 192.168.60.5 of network 192.168.60.0/24 can telnet to Host A - 10.9.0.5 of network 10.9.0.0/24.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

**CHAPTER 5: CONCLUSION**

In this report, we conducted a detailed study on the implementation of Netfilter and Iptables in a Virtual Machine, two powerful network tools on the Linux operating system. We focused on introducing, presenting specific theoretical concepts, how they operate, and how to configure and apply them in managing network traffic and protecting the system.

We examined how Netfilter works at the kernel level and how iptables provides a user-friendly interface for interaction at the user level. Through demonstrations and specific tutorials along with the achieved results, we described how to build rules on the router and utilize iptables' features to control and secure the network system.

An important point to note is the flexibility and robustness of Netfilter/iptables, allowing network administrators to customize and control traffic in various ways. This not only helps improve system performance but also enhances safety and security.

Additionally, we discussed challenges that may arise when using Netfilter/iptables, including rule management, monitoring, and removal. However, with deep knowledge and practical experience, these challenges can be effectively overcome. There are, of course, other aspects that we haven't detailed in this report.

In conclusion, we believe that this report provides a more comprehensive view of Netfilter/iptables and equips readers with the necessary knowledge to deploy and manage them in real-world system environments. Thank you for your interest, and we hope that it proves valuable to those interested in this field.

**LIST OF REFERENCES**

<https://luanvan.co/luan-van/de-tai-tim-hieu-ve-tuong-lua-va-netfilter-61839/>

<https://tonylixu.medium.com/linux-networking-netfilter-5b378db9e33b>

<https://www.digitalocean.com/community/tutorials/a-deep-dive-into-iptables-and-netfilter-architecture>

<https://www.britannica.com/technology/firewall>

<https://medium.com/@boutnaru/the-linux-security-journey-netfilter-90c6cf12ca40>

<https://hmn.wiki/vi/Netfilter>

<https://seedsecuritylabs.org/Labs_20.04/Networking/Firewall/>

“Computer and Internet Security - A Hands-on Approach” Book of Professor Wenliang Du

**APPENDIXES**