JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 27, No. 2, pp. 511-517, Feb. 2026

Check for updates

차세대 리눅스 방화벽 기술로서 nftables의 구조 및 활용 분석

정 성 재[1] · 이 재 웅[2*]
[1](주)엔버 기업부설연구소 연구소장
[2]오산대학교 컴퓨터소프트웨어과 조교수

Analysis of Linux Firewall Trends and Structure Based on nftables

Sung-Jae Jung[1] · Jae-Ung Lee[2*]

[1]Director of Research Institute/CTO, Company-affiliated research institute, Enber Co., Ltd, Seoul 06103, Korea
[2]Professor, Department of Computer Software, Osan University, Gyeonggi 18119, Korea

[요 약]

본 논문은 리눅스 네트워크 보안의 핵심인 넷필터 프레임워크의 진화 과정을 고찰하고, 기존 iptables의 구조적 한계를 극복하기 위해 도입된 nftables의 설계 특징 및 활용 동향을 심층 분석하였다. nftables는 커널 내 가상 머신(VM) 기반의 바이트코드 실행 방식을 채택함으로써 기존의 하드코딩된 로직에서 벗어나 커널 코드를 경량화하였으며, 사용자 공간의 도구 업데이트만으로도 새로운 프로토콜에 신속히 대응할 수 있는 유연성을 확보하였다. 또한, 익명 세트와 맵을 활용한 비선형 검색 구조를 통해 대규모 규칙 환경에서도 패킷 처리 효율을 최적화하고, 규칙 업데이트 시 발생할 수 있는 보안 공백을 원자적 업데이트 방식을 통해 원천적으로 차단하였다. 분석 결과, nftables는 주요 리눅스 배포판에서 기본 방화벽으로 채택되어 그 기술적 타당성을 검증받고 있으며, 향후 복잡해지는 IT 인프라 환경에서 점진적인 기술 전환과 함께 리눅스 보안 체계의 표준으로 자리 잡을 것으로 전망된다.

[Abstract]

This paper examines the evolution of Linux firewall technology and provides an in-depth analysis of design features and utilization trends of nftables, which was developed to overcome structural limitations of the long-established iptables firewall. By adopting a virtual machine-based bytecode execution within the kernel, nftables has successfully streamlined kernel code and achieved significant flexibility, allowing rapid adaptation to new protocols through updates in userspace tools. Furthermore, nftables optimizes packet processing efficiency in large-scale ruleset environments using non-linear search structures including anonymous sets and maps, while eliminating potential security gaps during policy updates through atomic update mechanisms. Our analyses indicate that nftables is becoming increasingly validated as it becomes the default firewall in major Linux distributions. Given its structural efficiency and manageability, nftables is expected to see steady adoption and eventually establish itself as the standard for Linux security frameworks within increasingly complex IT infrastructures.

색인어 : 리눅스, 방화벽, 아이피테이블, 파이어월디, 엔에프테이블
Keyword : Firewall, Firewalld, Iptables, Linux, nftables

http://dx.doi.org/10.9728/dcs.2026.27.2.511

Received 20 January 2026; **Revised** 02 February 2026
Accepted 04 February 2026

***Corresponding Author; Jae-Ung Lee**

Tel: +82-31-370-2695
E-mail: leejaeung1990@osan.ac.kr

Ⅰ. 서 론

유닉스 운영체제를 기반으로 탄생한 리눅스는 공개 소프트웨어라는 특징과 비용 부담 없이 사용할 수 있다는 장점으로 인해 많은 사용자와 기업들의 지지를 받게 되면서 지속해서 발전했다. IT 인프라가 클라우드 컴퓨팅 및 빅데이터 기반 환경으로 바뀐 현재 시점에서도 리눅스 운영체제는 핵심적인 역할을 수행하고 있으며, 모바일 기기 및 가전제품 등을 비롯한 사물인터넷 분야에서도 중요한 위치에 있다. 특히 인공 지능(artificial intelligence)이 상용화로 가는 시작점에도 리눅스가 자리잡고 있다. 이렇듯 현재 IT 영역에서 리눅스를 제외하고서는 운영이 어려울 정도로 보편화되어 있고, 보안 분야에서도 이러한 현상은 동일하다. 특히 침입 차단 시스템이라고 부르는 방화벽과 침입 탐지 시스템의 대부분이 리눅스 기반으로 운영되고 있다. 최근 리눅스 기반 방화벽 프로그램으로 nftables가 새롭게 등장하였는데, 기존 iptables와 비교하여 구조적 최적화를 통한 자원 효율성과 정책 관리의 편의성을 개선하였다[1]. 본 논문에서는 리눅스 방화벽 역사에 대해 알아보고, 새로운 방화벽 프로그램인 nftables를 분석을 통해 리눅스 기반 방화벽의 변화와 동향에 대해 알아본다. 특히 본 연구는 nftables의 기술적 구조와 발전 흐름을 분석하는 데 중점을 두며, 정량적 성능 비교 실험을 목적으로 하지 않음을 밝힌다.

Ⅱ. 관련 연구

2-1 리눅스 방화벽의 역사

리눅스 운영체제는 운영체제의 핵심이라고 할 수 있는 커널(kernel)이 1991년 9월에 버전 0.01로 출시되면서 시작되었다. 리눅스 방화벽 프로그램도 리눅스 커널 버전의 발전과 함께 진행된다. 리눅스 커널이 처음 등장하고 3년간은 커널의 기본 기능 완성에 초점이 맞춰졌고, 1994년 3월에 커널 1.0 버전이 등장하면서 운영체제로서의 모습이 갖춰지게 되었다. 초기의 리눅스 운영체제는 유닉스 운영체제를 모방해서 만들었는데, 특히 프로그램 소스가 공개되어 있고 자유롭게 사용이 가능한 BSD(berkeley software distribution) 계열 유닉스에서 많이 차용하였다. 리눅스 커널 1.0 버전에서의 방화벽 프로그램은 유닉스에서 많이 사용하던 TCP Wrapper와 ipfw이라는 도구를 사용하였다[2]. TCP Wrapper는 IP 주소나 도메인을 기반으로 접근을 허가하는 /etc/hosts.allow 파일과 접근을 거부하는 /etc/hosts.deny 파일을 이용해서 제어하는데, 방화벽 전용 프로그램이기보다는 단순히 접근을 제어하는 응용 프로그램에 가까웠다. ipfw는 ipfirewall이라고도 부르는데, 커널 1.1부터 사용되었다. ipfw은 방화벽의 핵심 기능인 패킷 필터링(packet filtering) 및 트래픽 제어 기능이 내장된 프로그램으로 리눅스 운영체제에서 실제 구현된 첫 번째 방화벽이라고 할 수 있다. ipfw은 명령행에서 명령을 실행해서 설정할 수도 있었다.

표 1. ipfw 사용 예
Table 1. Examples of ipfw

```
#  vi /etc/firewall.conf

add 03000 deny icmp from any to any
add 04000 allow all from any to any

add 03100 allow ip from 192.168.0.1 to any 80 keep-state
add 03200 allow ip from 192.168.0.1 to any 443 keep-state

add 04000 deny all from any to any

# service ipfw start

# ipfw show

# ipfw list
```

표 2. ipfwadm 사용 예
Table 2. Examples of ipfwadm

```
# ipfwadm -I -f
# ipfwadm -F -a accept -b -P tcp -S 0.0.0.0/ 1024:65535
192.168.5.13 80
# ipfwadm -A -f
# ipfwadm -A out -i -S 192.168.5.0/24 -D 0.0.0.0/0
# ipfwadm -A in  -i -S 192.168.5.0/24 -D 0.0.0.0/0
# ipfwadm -F -a deny -o -P udp -s 0/0 -D 172.16.12.0/24
```

ipfw를 비롯한 초창기 유닉스 계열 방화벽 프로그램은 특정한 환경 설정 파일을 사용하여 설정하고, 방화벽의 구동 및 정지와 방화벽 설정 내역 확인 등만 명령어를 사용하는 추세였다. 표 1은 일반적인 ipfw를 사용한 방화벽 설정, 방화벽 가동, 방화벽 설정 확인하는 절차이다.

커널 2.0 버전부터 사용된 ipfwadm은 들어오고 나가는 패킷만 제어한 것이 아니라 들어온 패킷의 정보를 변환해서 다시 내보내는 IP 마스커레이딩(masquerading) 기법을 지원하면서 2세대 리눅스 방화벽이라고 불렸다. ipfw와 비교하면 IP 마스커레이딩 기법과 같은 진일보한 기술이 추가되었고, 명령행에서의 사용법이 더욱 체계화되었지만, 표 2과 같이 여전히 명령행에서 명령어의 옵션을 통해 제어하는 형식이었다. 커널 2.2 기반에서 더욱 구조적으로 체계화된 ipchains가 등장하게 되면서 대체되었다.

표 3. ipchains 사용 예
Table 3. Examples of ipchains

```
# ipchains -F
# ipchains -A input -s 192.168.56.120 -p tcp -j DENY
# ipchains -A input -s 192.168.56.140 -p icmp -j DENY
# ipchains -P forward DENY
# ipchains -A forward -s 192.168.0.0/24 -j MASQ
```

2-2 ipchains

커널 2.0 버전에 사용된 ipfwadm은 BSD 유닉스 운영체제의 ipfw을 구조화 작업을 통해 다시 작성해서 만들었는데, 커널 2.2 버전에 사용된 ipchains는 ipfwadm을 더욱 체계화하여 만들었다[3]. 특히 사슬(chain)이라는 개념을 사용했는데, 이것은 패킷이 오고 가는 일종의 통로를 영역화한 것이라

고 보면 된다. 사슬은 크게 input, output, forward로 구분하였는데, input은 들어오는 패킷을 처리하는 영역이고, output은 나가는 패킷을 처리하는 영역이다. 마지막으로 forward는 패킷이 들어온 후에 다른 시스템을 향하는 패킷을 처리하는 영역으로 흔히 IP 마스커레이딩이라고 부르는 주소 변환 작업을 진행한다. ipchains는 방화벽의 기본 기능에 충실한 프로그램으로 미리 정의된 사슬에 역할을 부여하여 사용자가 체계적으로 정책을 설정할 수 있도록 지원한다. 또한 사슬 단위의 정책 확인 및 설정이 가능하여 ipfwadm과 비교하면 사용하기가 좀 더 용이하다. 다만 사슬을 지정해서 설정한다는 것만 다를 뿐이고, 명령행 기반으로 사슬을 선언하여 지정하는 방식은 동일하다. 커널 2.4 기반으로 강력한 기능을 탑재한 iptables가 등장하면서 대체되었다.

2-3 iptables

2001년, 리눅스 커널 2.4 버전부터 사용되기 시작한 iptables는 ipchains를 확장해서 만들어졌다. ipchains는 패킷이 오가는 통로에 해당하는 사슬(chain)이라는 개념을 만들었다면 iptables는 사슬의 위 단계에 해당하는 테이블(table)이라는 개념을 만들어서 기능과 역할을 강화하였다[4]. iptables는 특정 역할을 수행하는 상위 개념의 테이블이 존재하고, 테이블 별로 각각의 기능을 수행할 수 있는 사슬을 정의하고 있다. iptables는 초기에 3개의 테이블로 설계되었으나 계속해서 기능을 확장하면서 filter, nat, mangle, raw, security와 같이 5개의 테이블로 구성되었다[5]. iptables의 기본 테이블에 해당하는 filter는 패킷 필터링을 담당하는 곳으로 ipchains 이전에는 filter 테이블만 존재했다고 이해하면 된다. nat 테이블은 IP 주소는 변환하는 역할을 수행하는 곳으로 ipchains의 forward 사슬을 확장해서 구성하였다. mangle은 패킷 데이터를 변경하는 특수 규칙을 적용하는 테이블로 성능 향상을 위한 TOS(type of service)를 설정하고, raw는 연결 추적을 위한 하위 시스템과 독립적으로 동작해야 하는 규칙을 설정하는 테이블이다. 마지막으로 security는 리눅스 보안 모듈인 SELinux에 의해 사용되는 MAC (mandatory access control) 네트워크 관련 규칙이 적용된다. 이 테이블은 SECMARK 및 CONNSECMARK에 의해 활성화된 규칙이 등록된다. 현재 security는 filter 테이블 다음에 호출되어 DAC(discretionary access control) 규칙이 MAC 규칙보다 먼저 적용된다. iptables는 기능적으로 완성된 방화벽 프로그램이라고 볼 수 있지만, 패킷 필터링을 직접 수행하는 것은 아니고 커널에 있는 넷필터(netfilter)라는 모듈이 수행한다. 넷필터는 리눅스가 제공하는 모든 종류의 패킷 필터링과 맹글링(mangle) 도구로 네트워크 스택으로 함수를 후킹(hooking)하는데 사용할 수 있는 커널 내부의 프레임워크이다.

iptables를 ipchains와 사용법 관점에서 비교하면 명령행에서 명령어를 사용해서 방화벽 정책을 설정하는 것은 동일하지만, 설정할 때 테이블명과 사슬명을 지정하여 더욱 구조화하여 체계적으로 설정할 수 있다. 또한 설정된 방화벽 정책을 파일로 저장하고, 다시 파일로 저장된 파일을 불러올 수 있게 iptables-save, iptables-restore와 같은 명령어도 제공한다. iptables는 ipchains와 비교해서 체계적인 구성, 대폭 좋아진 성능, 다양한 추가 명령어 제공 등으로 인해 20년이 넘는 세월 동안 리눅스 방화벽 프로그램의 강자로서 자리를 잡게 된다. 이러한 영향으로 리눅스를 사용하는 상용 방화벽 장비뿐만 아니라 가정집에서 사용되는 IP 공유기에서 사용되고 있다.

표 4. iptables의 테이블과 사슬
Table 4. Tables and chains in iptables

Chain	Table				
	filter	nat	mangle	raw	security
INPUT	○	○	○		○
FORWARD	○		○		○
OUTPUT	○	○	○	○	○
PREROUTING		○	○	○	
POSTROUTING		○	○		

표 5. iptables 사용 예
Table 5. Examples of iptables

```
# iptables -t filter -F
# iptables -t nat -F
# iptables -t filter -A INPUT -s 192.168.5.22 -j DROP
# iptables -t nat -A PREROUTING -p tcp -d 203.247.50.244
--dport 80 -j DNAT --to 192.168.12.22:80

# iptables-save > firewall.sh
# iptables-resotre < firewall.sh
```

표 6. firewall-cmd 사용 예
Table 6. Examples of firewall-cmd

```
# firewall-cmd --state
# firewall-cmd --zone=public --list-services
# firewall-cmd --permanent --zone=public
--add-port=8080-8081/tcp
# firewall-cmd --permanent --zone=public
--add-rich-rule="rule family="ipv4" ₩
  source address="192.168.0.4/24" service name="http"
accept“
# firewall-cmd --zone=public --add-service=https
# firewall-cmd --add-service=ssh --timeout 15m
```

2-4 firewalld

iptables는 리눅스 기반 방화벽으로써 필요한 기능을 모두 갖추고 있었고, 세세하고 체계적인 구성으로 세밀한 부분까지 정교하게 방어벽을 제어할 수 있다. 하지만 정교한 제어를 하려면 iptables의 체계화된 구조를 이해하고 복잡한 사용법을 숙지해야 한다. 이러한 점은 단순하게 특정한 네트워크 서비스나 포트를 여닫는 기능만 필요한 초보자의 진입을 어렵게 하는 문제점도 나타나게 되었다. 이에 초보자들도 iptables를 쉽게 설정할 수 있는 도구들이 등장하게 되었는데, 이러한 도구에는 lokkit, system-config-firewall, firewalld 등이 있다[6]. lokkit는 CLI(command line interface) 기반의 도구로써 네트워크 서비스만 여닫는 등의 단순한 기능이 위주이

고, system-config-firewall는 X 윈도 환경에서만 동작한다. 이러한 부분을 통합하여 만든 것이 firewalld이다[7]. firewalld는 D-Bus(desktop bus) 인터페이스가 있는 동적 관리 방화벽으로 관리 규칙이 변경될 때마다 방화벽 데몬을 재시작할 필요 없이 규칙을 작성, 변경, 삭제할 수 있다. iptables에서 방화벽 규칙을 설정하려면 사용하는 테이블과 사슬명을 지정해야 하지만, firewalld는 일반 사용자들이 좀 더 직관적으로 쉽게 이해할 수 있는 영역(zone)이라는 곳에 설정한다. D-Bus 인터페이스를 통해 특정 서비스나 응용 프로그램의 방화벽 설정을 조정할 수 있다. 또한 명령행에서 사용하는 firewall-cmd, X 윈도 환경에서 사용하는 firewall-config, 주요 설정을 손쉽게 확인할 수 있는 firwall-applet 등과 같은 다양한 도구도 제공한다. 마지막으로 방화벽 정책 적용을 런타임(runtime) 구성과 영구 구성 설정으로 분리할 수 있고, 서비스 시간을 지정하는 등 iptables에서 지원하지 않는 다양한 기능을 제공한다. 특히 본 논문에서 언급될 nftables로 바뀐 현재 시점에서도 firewalld는 여전히 사용할 수 있다.

Ⅲ. nftables의 구조 및 운용 방식

3-1 nftables의 설계 철학 및 통합 프레임워크의 구조

nftables는 기존 리눅스 커널의 네트워크 스택에서 파편화되어 관리되던 iptables, ip6tables, arptables, ebtables 등을 단일 프레임워크로 통합하고 대체하기 위해 설계되었다. nftables의 핵심적인 구조적 변화는 커널 내부에 네트워크 전용 가상 머신(Network-specific VM)을 도입했다는 점이다. 기존의 방식이 하드코딩된 프로토콜 처리 로직에 의존했다면, nftables는 명령행 도구인 nft를 통해 규칙을 바이트코드 형식으로 컴파일하여 Netlink API를 통해 커널 VM으로 전달하고 이를 실행하는 방식을 취한다. 이러한 설계는 커널 코드를 경량화하고 최신 네트워크 프로토콜에 대한 대응력을 높이는 데 기여한다.

3-2 계층적 논리 구조와 동적 객체 관리

nftables는 규칙을 체계적으로 관리하기 위해 테이블(table), 사슬(chain), 규칙(rule)으로 이어지는 계층 구조를 사용한다. 이전의 iptables가 사전에 정의된 고정 테이블과 사슬 구조를 강제하여 불필요한 연산을 발생시켰던 것과 달리, nftables는 사용자가 주소 가족(Address Family)에 따라 필요한 테이블을 자유롭게 정의할 수 있는 유연성을 제공한다.

사슬은 넷필터 후크(netfilter hook)에 직접 연결되어 패킷을 가로채는 기본 사슬(base chain)과, 특정 조건에 따라 호출되어 모듈식으로 작동하는 일반 사슬로 세분화된다. 이러한 구조는 특정 프로토콜이나 인터페이스에만 국한된 필터링 정책을 수립할 수 있게 하여 전체적인 시스템 자원 소모를 최적화하는 효과가 있다.

표 7. nftables의 주요 특징
Table 7. Main feature of nftables

feature	comment
Network-specific VM	the nft command line tool compiles the ruleset into the VM bytecode in netlink format, then it pushes this into the kernel via the nftables Netlink API. When retrieving the ruleset, the VM bytecode in netlink format is decompiled back to its original ruleset representation. So nft behaves both as compiler and decompiler.
High performance through maps and concatenations	Linear ruleset inspection doesn't scale up. Using maps and concatenations, you can structure your ruleset to reduce the number of rule inspections to find the final action on the packet to the bare minimum.
Smaller kernel codebase	The intelligence is placed in userspace nft command line tool, which is considerably more complex than iptables in terms of codebase, however, in the midrun, this will potentially allow us to deliver new features by upgrading the userspace command line tool, with no need of kernel upgrades.
Unified and consistent syntax	Unified and consistent syntax for every support protocol family, contrary to xtables utilities, that are well-known to be full of inconsistencies.

3-3 규칙 적용의 원자성 및 관리 효율성

실무 운영 측면에서 nftables의 가장 큰 기술적 진보는 규칙 적용의 원자성(Atomicity) 확보에 있다. 방화벽 규칙 세트를 업데이트할 때 전체 설정을 초기화하고 다시 로드하는 것이 아니라, 증분 업데이트(Incremental updates) 방식을 통해 변경된 부분만을 원자 단위로 적용한다. 이는 대규모 트래픽이 발생하는 서버 환경에서 규칙 변경 시 발생할 수 있는 보안 정책의 공백이나 일시적인 패킷 유실을 원천적으로 차단한다. 또한, 여러 개의 IP 주소나 포트를 세트(Set) 객체로 묶어 단일 규칙에서 처리함으로써 규칙의 개수를 획기적으로 줄이고 관리 가독성을 향상시킬 수 있다.

표 8. nftables 사용 예
Table 8. Examples of nftables

```
# nft list ruleset
# nft flush ruleset
# nft add rule inet firewall_table firewall_chain tcp dport 22 accept
```

3-4 관리 도구 인터페이스 및 대화형 모드 운영

nftables는 직관적인 명령행 인터페이스를 제공하며, 관리자는 nft 명령어를 통해 실시간으로 커널의 네트워크 보안 정책을 제어할 수 있다. 특히 nftables는 정책 수립의 편의성과 검증을 위해 대화형 모드(Interactive mode)를 지원한다. 대화형 모드 내에서 관리자는 실시간으로 규칙을 입력하고 시스템의 반응을 즉각적으로 확인할 수 있어, 복잡한 보안 스크립트를 작성하기 전 단계에서 정책의 유효성을 검토하는 데

매우 유용하게 활용된다.

```
[root@www posein]# nft -i
nft> add table inet firewall_table
nft> list tables
table inet firewall_table
nft>
```

그림 1. 대화형 모드 사용 예
Fig. 1. Examples of using interactive mode

Ⅳ. nftables와 iptables의 비교 분석

4-1 비교 분석의 기준 및 관점

본 장에서는 기존 리눅스 방화벽의 표준이었던 iptables와 차세대 기술인 nftables를 체계적으로 대조하기 위하여 구조적 설계, 관리 편의성, 확장성, 유지보수의 4가지 핵심 기준을 설정하여 분석한다. 이러한 분석 기준은 단순한 기능의 우열을 가리는 것을 넘어, 실제 운영 환경에서의 기술적 타당성과 전환 시 발생하는 실무적 이점을 검증하는 데 목적이 있다. 특히, 동향 분석 연구로서 각 기술의 아키텍처 차이가 보안 정책 수립의 정확성과 시스템 자원 효율성에 미치는 영향을 중점적으로 고찰한다.

4-2 관리 편의성 및 실무적 정책 수립 체계

관리 편의성 측면에서 nftables는 이전의 파편화된 명령어 체계를 직관적인 스크립트 형식으로 개선하여 운영 효율성을 높였다. iptables는 IPv4와 IPv6 등 주소 가족에 따라 별도의 도구(iptables, ip6tables)를 각각 실행해야 했으나, nftables는 단일 문법으로 통합 제어가 가능한 환경을 제공한다. 특히 nftables는 관리자의 선호도와 운영 환경에 따라 셸 스크립트 기반의 작성 방식과 명령어 직접 입력 방식을 모두 지원한다.

이러한 유연한 작성 방식은 수백 개의 보안 규칙을 관리해야 하는 엔터프라이즈 환경에서 정책의 가독성을 보장하고, 설정 오류로 인한 보안 사고를 예방하는 중요한 기제로 작용한다.

표 9. 셸 스크립트 형식의 작성 예
Table 9. Example of writing a shell script

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset

table inet example_table {
  chain example_chain {
    # Chain for incoming packets that drops all packets that
    # are not explicitly allowed by any rule in this chain
    type filter hook input priority 0; policy drop;

    # Accept connections to port 22 (ssh)
    tcp dport ssh accept
  }
}
```

표 10. 명령어 사용 형식의 작성 예
Table 10. Example of writing command format

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset

# Create a table
add table inet firewall_filter

# Create a chain for incoming packets drops all packets
add chain inet firewall_filter firewall_chain { type filter hook input priority 0 ; policy drop }

# Add a rule that accepts connection to port 22 (ssh) & ftp
add rule inet firewall_filter firewall_chain tcp dport ssh accept
add rule inet firewall_filter firewall_chain tcp dport ftp accept

# Create a chain for outgoing packets drop
add chain inet firewall_filter output { type filter hook output priority 0; }

# Add a rule that drop connection list
add rule inet firewall_fliter output ip daddr 203.242.226.20 drop
add rule inet firewall_fliter output ip daddr www.seowon.ac.kr drop
```

4-3 구조적 최적화 및 고급 데이터 객체 활용 분석

nftables의 가장 혁신적인 기능 중 하나는 익명 세트(Anonymous Sets)와 맵(Maps)을 활용한 규칙 최적화이다. iptables에서는 동일한 동작을 수행하더라도 대상 주소가 다를 경우 주소당 하나의 규칙을 개별적으로 생성해야 했으며, 이는 규칙의 선형적 증가와 성능 저하로 이어졌다. 반면 nftables는 다수의 객체를 하나의 세트로 묶어 단일 규칙 내에서 처리함으로써 연산 부하를 획기적으로 줄인다.

표 11. 익명 세트 사용 예
Table 11. Using anonymous sets in nftables

```
# nft add rule inet example_table example_chain tcp dport
{ 22, 80, 443 } accept
```

```
[root@www posein]# nft list table inet info_table
table inet info_table {
        chain tcp_packets {
                counter packets 7 bytes 625
        }

        chain udp_packets {
                counter packets 3 bytes 304
        }

        chain incoming_traffic {
                type filter hook input priority filter; policy accept;
                ip protocol vmap { tcp : jump tcp_packets, udp : jump udp_packet
s }
        }
}
[root@www posein]#
```

그림 2. 익명 맵 사용 예
Fig. 2. Examples of using anonymous map

4-4 실무적 활용 기능: NAT 및 트래픽 제어 최적화

네트워크 주소 변환(NAT) 및 트래픽 모니터링 또한 nftables에서 더욱 세밀하게 제어된다. nftables는 별도의 전용 테이블을 강제하지 않고도 유연하게 NAT 정책을 수립할 수 있어 관리적 유연성이 높다.

특히, 실시간 트래픽 제한과 분석을 위한 meter와 counter

매개변수는 유지보수 및 보안 관제 측면에서 매우 강력한 기능을 제공한다. 특정 서비스(예: SSH)에 대한 무차별 대입 공격을 방어하거나 실시간 트래픽 통계를 확보하는 데 유용하다.

표 12. NAT 사용 예
Table 12. Using NAT in nftables

```
# nft add table ip nat
# nft add chain ip nat prerouting '{ type nat hook preroute priority -100 ; }'
# nft add ip nat prerouting tcp dport 4513 redirect to :22
```

```
[root@www posein]# nft list meter ip firewall_filter firewall_meter
table ip firewall_filter {
        meter firewall_meter {
                type ipv4_addr
                size 65535
                flags dynamic
                elements = { 192.168.56.124 ct count over 2  }
        }
}
[root@www posein]#
```

그림 3. ssh 접속의 meter 확인 예
Fig. 3. Examples of checking the meter of ssh connection

표 13. counter 매개 변수를 사용한 규칙 생성 예
Table 13. Examples of rule creation with counter parameter

```
# nft add rule ip firewall_filter firewall_chain tcp dport 22 counter accept
```

```
[root@www posein]# nft list ruleset
table ip firewall_filter {
        chain firewall_chain {
                type filter hook input priority filter; policy accept;
                tcp dport 22 counter packets 49 bytes 6677 accept
        }
}
[root@www posein]#
```

그림 4. counter 매개 변수 확인 예
Fig. 4. Checking examples of counter parameter

4-5 편리해진 호스트의 접속하는 수 제한

종합적으로 볼 때, nftables는 구조적 설계의 유연성과 관리 편의성 면에서 이전 세대 기술을 크게 앞서고 있다. 심사위원의 지적대로 실제 운영 환경에서의 마이그레이션 비용이 고려되어야 하나, nftables가 제공하는 호환성 인터페이스(iptables-nft)는 기존 규칙을 유지하며 점진적 기술 전환을 가능케 한다. 또한, 무중단 규칙 갱신을 보장하는 원자적 업데이트와 세트 기반의 최적화는 고가용성과 고성능이 동시에 요구되는 현대적 IT 인프라 환경에서 nftables가 차세대 표준으로서의 당위성을 가지는 핵심 근거가 된다.

V. 결 론

본 논문은 리눅스 네트워크 보안의 중추인 넷필터 프레임워크의 기술적 진화 과정을 고찰하고, 기존 iptables의 설계적 한계를 극복하기 위해 도입된 nftables의 구조적 특성과 실무적 활용 방안을 심층 분석하였다. 분석 결과, nftables는 가상 머신(VM) 기반의 엔진과 유연한 객체 관리 체계를 채택함으로써 규칙 관리의 효율성을 최적화하고 커널 내 패킷 처리 구조를 혁신하였음을 확인하였다.

본 연구의 학술적 및 실무적 기여는 다음과 같이 요약할 수 있다. 첫째, 리눅스 방화벽 기술의 세대별 설계 철학을 체계적으로 분류하여 차세대 보안 프레임워크의 도입 당위성을 기술적으로 규명하였다. 둘째, nftables의 핵심인 원자적 업데이트와 비선형 검색 구조를 분석하여 대규모 네트워크 환경에서의 자원 효율성 개선 근거를 제시하였다. 마지막으로, 기존 시스템으로부터 nftables 체제로의 전환 시 발생하는 기술적 쟁점과 실무적 고려사항을 정리함으로써 보안 관리자를 위한 실무 가이드라인을 제공하였다.

현재 주요 리눅스 배포판에서 nftables 채택이 가속화됨에 따라, 보안 인프라 전반에서 점진적인 패러다임 전환이 이루어지고 있다. nftables는 기술적 유연성과 확장성을 바탕으로 차세대 리눅스 보안 체계의 핵심 표준으로 정착해가는 과정에 있다고 판단된다.

다만, 본 연구는 구조적 메커니즘 규명과 체계적 비교에 집중하였기에 실제 운영 환경에서의 수치화된 정량적 지표를 도출하는 데는 한계가 있었다. 이를 보완하기 위해 향후에는 가상화 및 컨테이너 기반의 클라우드 네이티브(Cloud Native) 환경으로 연구 범위를 확장하여, 복잡한 네트워크 토폴로지에서의 동적 보안 정책 제어 및 자동화 효율성을 검증하고자 한다. 또한, 다양한 인프라 환경에서 수집된 실측 데이터를 바탕으로 정량적 성능 벤치마크 연구를 병행함으로써, nftables가 차세대 리눅스 보안 표준으로서 갖는 기술적 완성도를 더욱 객관적으로 입증할 계획이다.

감사의 글

본 연구는 2025년 오산대학교 경기도 지역혁신중심 대학 지원체계(RISE) 구축·지원 사업 연구지원비에 의하여 수행된 연구임.

참고문헌

[1] Netfilter.org. The Nftables Project [Internet]. Available: https://www.netfilter.org/projects/nftables/index.html/.

[2] Y.-M. Bae and S.-J. Jung, "A Study on the Linux Firewall," *Journal of Security Engineering*, Vol. 8, No. 5, pp. 599-610, 2011.

[3] R. Russell. Linux IP Firewalling Chains [Internet]. Available: https://people.netfilter.org/rusty/ipchains/.

[4] O. Andreasson. Iptables Tutorial [Internet]. Available: https://www.frozentux.net/iptables-tutorial/iptables-tutorial.html/.

[5] S. J. Jung, Y. M. Bae, J. S. Park, B. W. Shin, Y. J. Jeon, and K. Y. Lee, *Understanding the Linux Core with Rocky Linux 8*, Seoul: Booksholic Publishing, 2022.

[6] K. Sung, "Analysis of Linux Firewall Trends and Its Management," *Journal of the Korea Convergence Society*,

Vol. 10, No. 2, pp. 43-48, 2019.
[7] Firewalld. Firewald's Official Project Website [Internet]. Available: https://firewalld.org/.

정성재(Sung-Jae Jung)

2003년 : 한남대학교 대학원 (공학석사)
2011년 : 한남대학교 대학원 (공학박사 -클라우드컴퓨팅)
2005년~2010년: 한남대학교 국제IT교육센터 전임강사
2011년~2015년: (주)스컴씨엔에스 기업부설연구소장
2015년~현 재: (주)엔버 기업부설연구소장/CTO
※관심분야 : 리눅스, 정보보안, 인공지능, 전자상거래, 클라우드 컴퓨팅 등

이재웅(Jae-Ung Lee)

2018년 : 한남대학교 대학원 (공학석사)
2021년 : 한남대학교 대학원 (공학박사 -수료)
2023년~현 재: 오산대학교 컴퓨터소포트웨어과 조교수
※관심분야 : 정보보안, 시스템보안, 디지털포렌식, 디지털사이니지, 머신러닝 등