Firewall 발표

1 Firewall

firewall

- 미리 정의된 보안 규칙에 기반하여, 들어오고 나가는 네트워크 트래픽을 모니터링하고 제어하는 네트워크 보안 시스템
- 신뢰할 수 있는 내부 네트워크와 신뢰할 수 없는 외부 네트워크 간의 장벽을 구성

- 기능에 따라 다음과 같은 종류로 분류

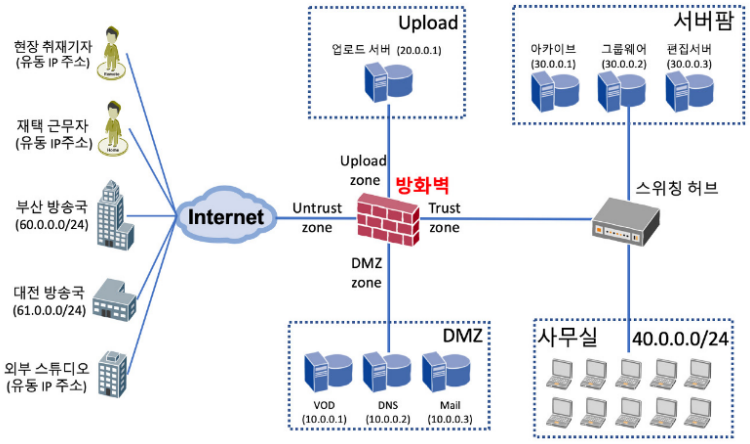
- 패킷필터
  - 패킷을 검사하여 미리 설정된 정책에 맞지 않을 경우 동과시키지 않도록 하는 형태의 방화벽을 패킷 필터
  - 패킷 자체를 본 것인지 TCP/UDP세션도 함께 관리하는 지에 따라 stateless, stateful firewall로 나뉨
  - stateless firewall - 내부적으로 상태를 관리할 필요가 없음
  - stateful firewall - 패킷이 속하는 세션을 관리하여 이 세션에 속하는 패킷들에 대해서 모두 동일한 처리를 하게 하는 방화벽

- 프록시
  - 세션에 포함되어 있는 정보의 유해성을 검사하기 위해서 방화벽에서 세션을 종료하고 새로운 세션을 형성하는 방식의 방화벽
  - 출발지~목적지 세션을 출발지에서 방화벽까지의 세션과 방화벽에서 목적지까지의 두 세션으로 만듦
  - 다른 세션으로 정보를 넘겨주기 전에 검사를 수행

- NAT
  - 내부 네트워크에서 사용하는 IP 주소와 외부에 드러나는 주소를 다르게 유지
  - 서로 다른 세션이 외부에서는 하나의 세션으로 보일 경우가 생김
  - 세션 충돌이 생겼을 경우 출발지 포트를 변경하여 충돌을 피하는데 이를 포트 주소 변환 (PAT)라고 함

blocked URL

- 대부분의 방화벽은 정책 기반의 방화벽이며 다양한 수준의 정책으로 네트워크 간의 트래픽을 제어



	요구 사항	보안정책 반영 내용
1	일반 시청자들이 인터넷을 통해 VOD서버에서 영상 검색과 시청이 가능해야 하며 서비스가 계속 증가하므로 서비스포트는 모두 열어주어야 한다.	Untrust --> DMZ ANY, 10.0.0.1/32, ANY, Permit
2	인터넷을 통해 VOD, Mail, 업로드 서버의 도메인주소로 접속이 가능해야 하고, 사내에서 인터넷사용시 DNS서버 사용에 문제가 없어야 한다.	Untrust --> DMZ ANY, 10.0.0.2/32, UDP53, Permit Trust --> DMZ 40.0.0.0/24, 10.0.0.2/32, UDP53, Permit
3	회사내부에서 업무용 메일을 받고(SMTP) 보낼 (POP3) 있어야 하고, 사내에서 뿐만 아니라 출장이나 재택근무중에도 메일을 사용할 수 있어야 한다.	Trust --> DMZ 40.0.0.0/24 10.0.0.3/32, TCP25(SMTP)/TCP110(POP3), Permit Untrust --> DMZ ANY, 10.0.0.3/32, TCP25(SMTP)/TCP(POP3), Permit
4	그룹웨어 서버는 보안을 위해 외부에서 인터넷을 통한 접속은 못하고 본사 사내와 지역 방송국에서만 접속이 되어야 한다.	Untrust --> Trust 60.0.0.0/24, 30.0.0.2/24, TCP80, Permit 61.0.0.2/24, 30.0.0.2/24, TCP80, Permit
5	현장에서 취재한 사진과 기사는 인터넷을 통해 업로드서버로 전송이 가능해야 하고, 서버에 전송된 사진과 기사는 사내에 있는 편집 서버에서 FTP로 사진과 기사를 가져갈 수 있어야 한다.	Untrust --> Upload ANY, 20.0.0.1/32, TCP80, Permit Trust --> Upload 30.0.0.3/32, 20.0.0.1/32, FTP, Permit
6	사무실에 있는 노트북에서 인터넷 사용이 가능해야 한다.	Trust --> Untrust 40.0.0.0/24, ANY, TCP80, Permit
7	코로나 사태 등으로 재택근무가 필요할 경우 인터넷을 통해 사내에 있는 그룹웨어에 접속이 가능해야 한다.	Untrust --> Trust ANY, 30.0.0.2/32, TCP80, Permit

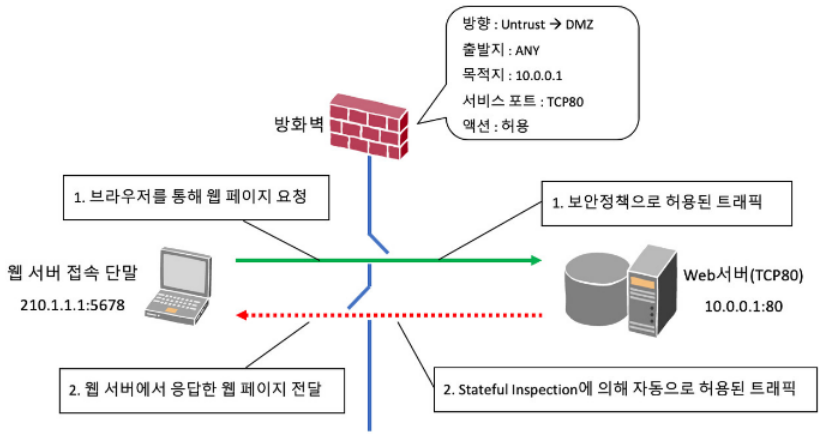
- 네트워크 구성도와 고객 요구 사항이 있을 때 방화벽에서 어떻게 보안 정책을 반영하는지

- 1번: 요구 사항 - 인터넷은 통해 시청자가 VOD 서버에 접근이 가능하게 하는 것
- 방화벽 - Untrust 구역에서 들어오는 출발지 IP 주소를 지정할 수 없음 → ANY도 하여 모든 IP가 접근 가능하게 설정
- 목적지인 DMZ는 모든 포트에 대해 접속을 허용하여 요청은 VOD 서버도 전달함

- 5번: 요구 사항 - 외부에서 인터넷은 통해 취재한 사진과 기사를 송고해야 함 → 기자의 스마트폰 등은 통해 인터넷 접속은 해야 함
- 이때 할당받는 공인 IP는 무작위도 할당 → 지정할 수 없으므도 ANY도 함
- 사내의 편집 서버에서 자료를 받아 가기 위해 FTP(파일 전송 프도도콜) 서비스를 통해 접근이 가능하게 설정 필요

stateful firewall

- 들어오는 패킷은 필터링 할 뿐만 아니라 클라이언트와 서버 간 통신 상태를 모니터링하여 연결 테이블은 만들고 관리하면서 좀 더 세밀한 트래픽 제어가 가능해짐



- 왼쪽 단말 장비에서 오든쪽 웹 서버도 접속 시도 가정
- 방화벽에는 인터넷에서 DMZ 구역으도 들어오는 패킷에 대해, 춛받지는 ANY, 목적지는 10.0.0.1, 서비스 포트는 TCP 80 포트에 대해 허용하는 보안 정책이 설정되어 있음
- 접속 단말이 서버도 웹 페이지를 요청 → 방화벽은 해당 요청에 대해 보안 정책을 확인하여 패킷을 웹 서버도 전달 → 서버는 요청에 대한 응답으도 접속 단말을 목적지도 하는 패킷을 생성하여 전송 → 응답 패킷은 방화벽에서 다시 확인되어 패킷을 허용핟 건지 차단핟 건지 결정함
- 위 방화벽 정책은 외부에서 DMZ 구역으도 갇 수 있는 보안 정책이 있지반, 방대 방향에 대한 보안 정책은 없음
- 방화벽은 기본적으도 보안 정책에 적용되지 않는 모든 패킷은 차단함
- 따라서 서버에서 단말도 가는 응답 패킷은 차단될 것으도 예상됨
- 하지만 stateful firewall 득성에 의해 서버에서 보낸 페이지는 방화벽을 정상적으도 동과함
- 방화벽이 관리하는 세션 테이븓을 참조하여 들어오는 응답 패킷의 춛받지, 목적지 IP, port와 일치하는 세션리스트가 있는지 확인되면, 해당 패킷은 응답 패킷으도 판단하여 보안 정책이 없더라도 해당 패킷을 클라이언트에게 전송
- 방화벽은 데이터를 요청하는 트래픽이 들어오면, 서버도 전달하면서 동시에 세션 테이븓은 만들어서 서버와 클라이언트 간의 동신 내역은 모니터딩하고 제어하는 용도도 사용
- 관리자가 서버의 응답 패킷은 예상해서 보안정책은 미리 만들어 둘 필요 없이 방화벽이 자동으도 응답 패킷에 대한 보안정책은 생성했다가 연결이 종료되면 삭제하는 것처덤 작동
- 미리 만들어 둔 보안정책은 동해 받생 가능한 잠재적인 보안 위협은 감소시킫 수 있는 장점

2 Netfilter, Iptables

netfilter

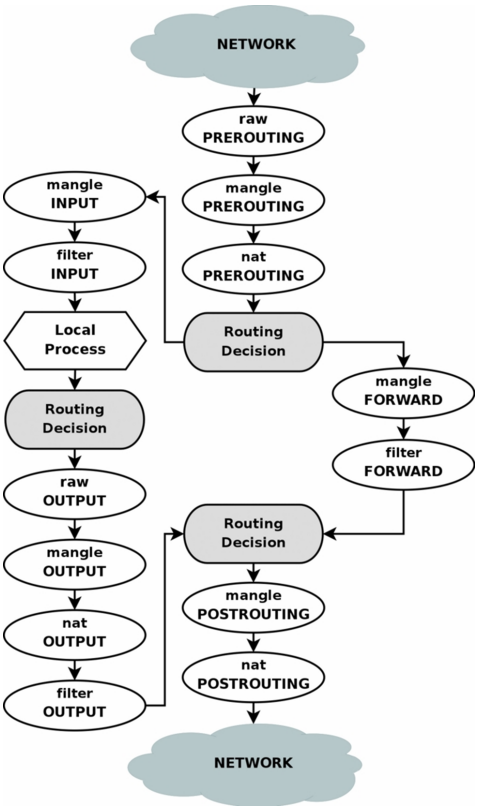
- Linux kernel 모듇도, 네트워크 패킷은 처리하기 위한 프레임워크
- filtering, NAT, PAT, packet mangling 기능은 제공
- iptables 커맨드에 의해 컨트롤됨
- iptables은 user-space에서 netfilter를 설정핟 수 있는 tool
- iptables은 rules를 구성하기 위해 tables를 사용
- 각각의 tables에서, rule들은 CHAIN으도 구성됨
- 각 CHAIN에서, packet은 상위 rule부터 순차적으도 적용 받음

Iptables

- filter - default table, 패킷의 ACCEPT, DROP 여부를 결정
- nat - 네트워크 IP 주소를 변환하기 위한 목적
- mangle - 패킷의 IP 헤더를 다양한 방식으도 수정핟 때 사용
- raw - 연결 트래킹에 대한 정보를 마킹 하기 위해 사용

	PREROUTING	INPUT	FORWARD	OUTPUT	POSTROUTING
(routing decision)				✓	
raw	✓			✓	
(connection tracking enabled)	✓			✓	
mangle	✓	✓	✓	✓	✓
nat (DNAT)	✓			✓	
(routing decision)	✓			✓	
filter		✓	✓	✓	
security		✓	✓	✓	
nat (SNAT)		✓			✓

- Incoming packets destined for the local system: PREROUTING → INPUT
- Incoming packets destined for another host: PREROUTING → FORWARD → POSTROUTING
- Locally generated packets: OUTPUT → POSTROUTING



Netfilter flow which can be configured by iptables

- routing decision 단계에서 패킷의 목적지가 local machine이 아니라면
- FORWARD chain으로 routing됨
- routing decision 단계에서 패킷의 목적지가 local machine이라면
- INPUT chain으로 routing됨

Iptables command

iptables command

Basic usage:

iptables [-t table_name] -COMMAND CHAIN_NAME matches -j TARGET

Table	Command	CHAIN	matches	Target/Jump
filter (default) nat mangle raw	-A (append) -I (insert) -D (delete) -R (replace) -F (flush) -Z (zero) -L (list) -S (show) -N -X	INPUT OUTPUT FORWARD PREROUTING POSTROUTING USER_DEFINED	-s source_ip -d dest_ip -p protocol --sport source_p --dport dest_p -i incoming_int -o outgoint_int -m mac -m time -m quota -m limit -m recent	ACCEPT DROP REJECT LOG SNAT DNAT MASQUERADE LIMIT RETURN TEE TOS TTL

실습

실습환경: Ubuntu 16.04 LTS

blocked URL

pc1

IP: 192.168.177.1

blocked URL

pc2

IP: 192.168.177.130

command

-L

```
[01/15/24]seed@VM:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[01/15/24]seed@VM:~$
```

-A

```
[01/15/24]seed@VM:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
[01/15/24]seed@VM:~$ sudo iptables -A INPUT -p tcp --dport 22 -i eth0 -j DROP
[01/15/24]seed@VM:~$ sudo iptables -A INPUT -p tcp --dport 25 -m iprange --src-range 10.0.0.10-10.0.0.18 -j DROP
[01/15/24]seed@VM:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https
DROP       tcp  --  anywhere             anywhere             tcp dpt:ssh
DROP       tcp  --  anywhere             anywhere             tcp dpt:smtp source IP range 10.0.0.10-10.0.0.18

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[01/15/24]seed@VM:~$
```

-I

```
[01/15/24]seed@VM:~$ sudo iptables -I INPUT -p tcp --dport 25 --syn -m connlimit --connlimit-above 5 -j REJECT --reject-with tcp-rst
[01/15/24]seed@VM:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
REJECT     tcp  --  anywhere             anywhere             tcp dpt:smtp flags:FIN,SYN,RST,ACK/SYN #conn src/32 > 5 reject-with tcp-reset
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https
DROP       tcp  --  anywhere             anywhere             tcp dpt:ssh
DROP       tcp  --  anywhere             anywhere             tcp dpt:smtp source IP range 10.0.0.10-10.0.0.18

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[01/15/24]seed@VM:~$
```

-D

```
[01/15/24]seed@VM:~$ sudo iptables -D INPUT 2
[01/15/24]seed@VM:~$ sudo iptable s-L
sudo: iptable: command not found
[01/15/24]seed@VM:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
REJECT     tcp  --  anywhere             anywhere             tcp dpt:smtp flags:FIN,SYN,RST,ACK/SYN #conn src/32 > 5 reject-with tcp-reset
DROP       tcp  --  anywhere             anywhere             tcp dpt:ssh
DROP       tcp  --  anywhere             anywhere             tcp dpt:smtp source IP range 10.0.0.10-10.0.0.18

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[01/15/24]seed@VM:~$
```

-F

```
[01/15/24]seed@VM:~$ sudo iptables -F
[01/15/24]seed@VM:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[01/15/24]seed@VM:~$
```

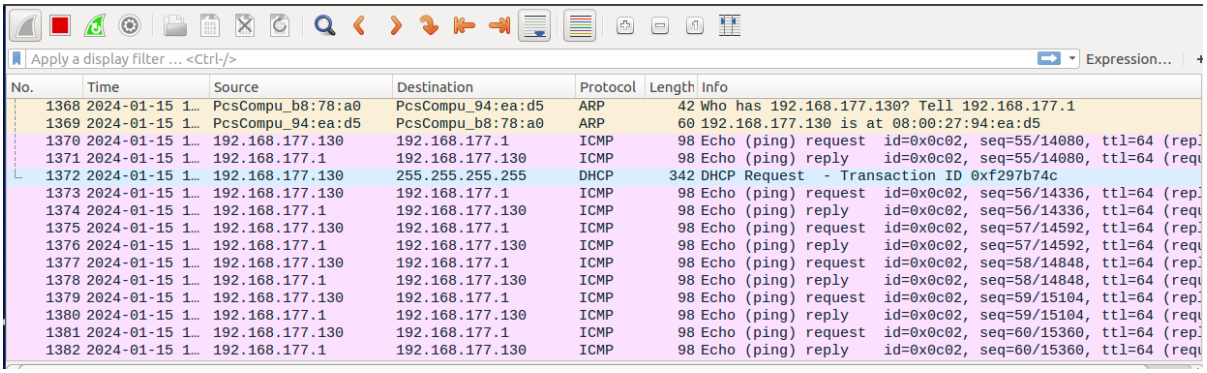
case1

- pc2 → pc1으로 ping 보냄
- pc1, pc2는 모든 패킷은 ACCEPT하도록 rule 설정

- iptables -P INPUT ACCEPT
- iptables -p OUTPUT ACCEPT
- iptables -p FORWARD ACCEPT

```
[01/15/24]seed@VM:~$ ping 192.168.177.1
PING 192.168.177.1 (192.168.177.1) 56(84) bytes of data.
64 bytes from 192.168.177.1: icmp_seq=1 ttl=64 time=1.02 ms
64 bytes from 192.168.177.1: icmp_seq=2 ttl=64 time=1.02 ms
64 bytes from 192.168.177.1: icmp_seq=3 ttl=64 time=1.14 ms
64 bytes from 192.168.177.1: icmp_seq=4 ttl=64 time=4.26 ms
64 bytes from 192.168.177.1: icmp_seq=5 ttl=64 time=1.07 ms
64 bytes from 192.168.177.1: icmp_seq=6 ttl=64 time=1.04 ms
64 bytes from 192.168.177.1: icmp_seq=7 ttl=64 time=0.960 ms
```

pc2→pc1으로 ping



pc1 wireshark

case2

- pc2 → pc1으로 ping 보냄
- pc2는 모든 패킷을 ACCEPT 하도록 rule 설정
- pc1을 pc2로부터 오는 패킷을 DROP 하도록 rule 설정
- DROP된 패킷을 확인하기 위한 LOGGING라는 CHAIN 정의
- 해당 로그를 /var/log/kern.log에서 확인

```
[01/15/24]seed@VM:~$ sudo iptables -A INPUT -s 192.168.177.130 -j LOG
[01/15/24]seed@VM:~$ sudo iptables -A INPUT -s 192.168.177.130 -j DROP
```

- pc1에서 pc2로부터 오는 패킷에 대해 log를 남기기 위해 첫번째 rule에서 target을 LOG로 설정
- pc2로부터 오는 패킷을 DROP 하기 위해 INPUT CHAIN에서 해당 source IP에 대해 target을 DROP으로 설정

```
[01/15/24]seed@VM:~$ sudo iptables -N LOGGING
[01/15/24]seed@VM:~$ sudo iptables -A INPUT -j LOGGING
[01/15/24]seed@VM:~$ sudo iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix "IPTables-Dropped:
" --log-level 4
[01/15/24]seed@VM:~$ sudo iptables -A LOGGING -j DROP
[01/15/24]seed@VM:~$
```

- 첫번째 커맨드를 통해 사용자 정의 LOGGING CHAIN 생성
- DROP 된 패킷을 LOGGING CHAIN으로 리다이렉트
- --limit 2/min 옵션을 통해 분당 최대 2번까지만 로그 기록 허용
- --log-prefix를 통해 "IPTabels-Dropped" 라는 문자열을 로그 메시지에 접두어로 추가

- --log-level 은 동해 로그 레벨 지정
- LOGGING CHAIN에 DROP 규칙 추가

```
Jan 15 14:12:43 VM kernel: [ 9039.300321] IN=enp0s3 OUT= MAC=08:00:27:b8:78:a0:08:00:27:94:ea:d5:08:00 SRC=192.
168.177.130 DST=192.168.177.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=16548 DF PROTO=ICMP TYPE=8 CODE=0 ID=3434 SEQ
=1
Jan 15 14:12:44 VM kernel: [ 9040.326981] IN=enp0s3 OUT= MAC=08:00:27:b8:78:a0:08:00:27:94:ea:d5:08:00 SRC=192.
168.177.130 DST=192.168.177.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=16649 DF PROTO=ICMP TYPE=8 CODE=0 ID=3434 SEQ
=2
Jan 15 14:13:52 VM kernel: [ 9108.107634] IPTables-Dropped: IN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:
08:00 SRC=127.0.0.1 DST=127.0.1.1 LEN=74 TOS=0x00 PREC=0x00 TTL=64 ID=31707 DF PROTO=UDP SPT=57170 DPT=53 LEN=5
4
Jan 15 14:13:57 VM kernel: [ 9113.118994] IPTables-Dropped: IN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:
08:00 SRC=127.0.0.1 DST=127.0.1.1 LEN=74 TOS=0x00 PREC=0x00 TTL=64 ID=31789 DF PROTO=UDP SPT=57170 DPT=53 LEN=5
4
Jan 15 14:14:02 VM kernel: [ 9118.122870] IPTables-Dropped: IN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:
08:00 SRC=127.0.0.1 DST=127.0.1.1 LEN=74 TOS=0x00 PREC=0x00 TTL=64 ID=32177 DF PROTO=UDP SPT=33873 DPT=53 LEN=5
```

- tail /var/log/kern.log를 동해 DROP된 패킷에 대해서 IPTables-Dropped라는 메시지와 함께 로그가 남은 것을 확인할 수 있음