DDoS 공격 상황에서 효율적인 iptables 정책 설계

변진영 · 한병우 · 이기영

인천대학교

Effective iptables Policy Design Against DDoS Attack

Jin-yeong Byeon, Byung-woo Han, Ki Young Lee

Dept. of Info & Telecom Engineering, University of Incheon

E-mail : skyrantt@yahoo.co.kr

요 약

본 논문에서는 DDoS(Distribute Denial of Service) 공격 상황에서의 효율적인 iptables 정책 설계 방법에 대해 제안한다. 현재 구축된 인터넷 상황에서는 누구나 손쉽게 DDoS Attack 환경을 구축하여 특정 사이트를 공격할 수 있다. 이를 극복하기 위해 DDoS 방어전용 장비 등을 사용 할 수 있지만 고가라는 단점 때문에 개인 혹은 소규모 서비스 시스템에서 사용하기에 어려운 점이 많다. 본 논문은 이러한 상황을 극복하기 위해 개인 및 소규모 시스템에 적용 가능한 효율적인 iptables 정책을 제안한다. 제안한 각 정책별 성능 평가는 웹 서비스 환경과 DDoS 유형별 공격 시나리오를 가정하여 시행하였고, 이는 실제적으로 효과가 있음을 확인하였다.

키워드

DDoS, iptables, iptables module, policy design

Ⅰ. 서 론

Netfilter Project에서 개발되어진 iptables 방화벽 시스템은 ipchain의 기능에서 버전업 된 새로운 형식의 네트워크 패킷 필터링 도구로써 리눅스 시스템과 연동하여 IP Packet에 대해 강력한 제어 기능을 갖춘 방화벽이다. 리눅스 커널은 Netfilter 라는 강력한 네트워킹 하부 시스템을 제공하고 있으며 이러한 Netfilter 하부 시스템은 NAT 및 IP 마스커레이딩 서비스를 비롯한 패킷 필터링을 제공한다.

기본적으로 iptables에는 NAT, Filter, Mangle, Raw 네 가지 테이블이 있다. 그중 패킷 필터링 규칙은 Filter 테이블에 적용된다. Filter 테이블에는 INPUT, OUTPUT, FORWARD 세 가지 체인이 있다. 패킷은 그림 1. 과 같이 이 세 가지 체인 중 하나를 반드시 통과하게 된다.[1~2]

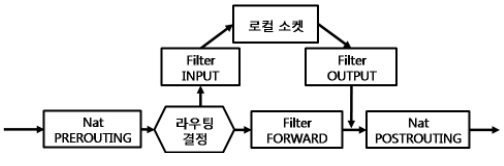

그림 1. iptables 패킷 흐름도

본 논문에서는 Filter 테이블에 규칙들을 설계함으로써 DDoS 공격에 효과적인 방어 법을 제시한다.

Ⅱ. DDoS 공격 유형

DDoS 공격이란 컴퓨터나 네트워크 자원을 고갈시켜 해당 컴퓨터와 네트워크가 정상적인 서비스를 제공할 수 없게 만드는 모든 공격 방법들을 지칭하며 주로 소프트웨어 / 프로토콜 구현상의

표 1. DDoS 공격 유형

공격 분류	공격 유형
Flooding	Syn Flooding Ack Flooding Rst Flooding Fin Flooding UDP flooding ICMP flooding TCP / UDP / ICMP flooding TCP / IP NULL 공격
Connection	Get Request Attack Http Connection Attack
Application	Cache Control Attack FTP, DNS, DHCP, SQL Attack 등

버그나 시스템의 설정 오류 등의 허점을 이용한다. DDoS 공격 증상에는 크게 시스템 / 네트워크 자원 고갈 현상과 네트워크 대역폭 (BW) 고갈 현상 그리고 시스템 상에서 동작하는 애플리케이션 등 프로그램 오류를 발생하는 현상이 있다. 시스템 / 네트워크 자원 고갈은 전송 받은 패킷을 처리 하는 과정에서 CPU, Memory, Disk 등의 시스템을 독점하여 과소비함으로써 서비스를 거부하게 하는 공격이고, 네트워크 대역폭 (BW) 고갈 현상은 네트워크 트래픽을 무한히 증가시켜 외부로부터 접속을 거부하게 하는 공격이다.

Ⅲ. iptables module

iptables는 다양한 모듈을 사용하여 효과적인 정책을 설계할 수 있다. DDoS 공격에 효과적으로 사용 가능한 모듈을 다음과 같이 간단히 설명한다.

1. recent module

최근에 사용된 주소를 기반으로 정책을 생성할 수 있는 기능을 제공한다.

2. connlimit module

Connection에 대한 제한을 IP 혹은 IP 대역별로 제한할 수 있는 기능을 제공한다. 일부 버전의 iptables에서는 기본으로 포함되어 있지 않기 때문에 추가 패치를 해 주어야 한다.

3. hashlimit module

접속하는 모든 패킷의 Source IP를 Hash 코드로 기억하여 리스트를 생성한 후 접속자들의 IP를 매칭하여 접속률을 제한한다.

4. string module

방화벽을 통과하는 패킷의 내용 중 문자열을 검색한다. 이 모듈은 Layer 7 수준에서 패킷에 대한 제어를 제공한다.

5. state module

들어오는 패킷의 상태를 추적하여 관리한다. 상태 NEW는 새롭게 연결을 시작하려 하거나 이전의 연결 추적 테이블에 보이지 않는 패킷을 뜻하고, ESTABLISHED는 클라이언트의 연결 시도 후 서버에서 응답하여 이미 연결되어 있는 상태를 의미한다. RELATED는 새롭게 연결을 시작하려고 하나 이미 연결 추적 테이블에 접속과 관련 있는 ESTABLISHED 테이블이 있는 경우를 말하고 INVALID는 연결 상태를 알 수 없거나 잘못된 헤더를 가지고 있는 경우를 뜻한다.

Ⅳ. DDoS 방어 정책 설계

본 논문에서 iptables 정책을 설계함에 있어서 각 공격유형별 체인을 생성하여 INPUT 체인에서 해당 체인으로 트래픽을 우회하였으며 공격별 성향을 분석하여 적합한 module을 사용해 해당 패킷을 차단하는 정책을 설계 하였다. 또한 가독성을 향상을 위해 정책은 iptables-save 형식으로 작성하였다.

1. Flooding Attack

1) 공격 방식

피해자에게 프로토콜의 허점을 노린 패킷을 반복 전송하여 시스템 / 네트워크 자원을 고갈 시키는 형태로 SYN Flooding이 대표적인 형태다.

2) iptables 방어 정책

```
*filter
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:flooding-attack - [0:0]
-A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
-A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
-A INPUT -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
-A INPUT -p tcp --tcp-flags ACK,FIN FIN -j DROP

-A INPUT -p tcp -m state --state NEW -j flooding-attack

-A flooding-attack -m hashlimit --hashlimit 12/s
--hashlimit-burst 24 --hashlimit-mode srcip
--hashlimit-name accept -j ACCEPT
-A flooding-attack -j DROP

COMMIT
```

그림 2. Flooding Attack 방어 정책

그림 2.의 정책은 flooding-attack을 생성하여 Flooding Attack 트래픽을 제어한다. 우선 INPUT 체인에 적용되는 내용을 살펴보면 4가지 어긋난 Flag 형태의 패킷을 우선적으로 차단한다. 그 다음 정책은 state module을 사용하여 새로 접속해 오는 패킷들을 flooding-attack 체인으로 우회 시킨다. 그리고 hashlimit module을 사용하여 초당 12개 패킷을 우선적으로 허용하지만 그 이상 접속하는 패킷은 다음 정책에서 차단한다. 이 정책은 DDoS공격이 어느 정도 일정한 근원지로부터 발생되는 경우에는 효과적으로 동작한다.

2. Connection Attack

1) 공격 방식

Connection 공격은 서버와 정상적인 접속을 이룬 후 의도적으로 접속을 끊지 않고 새로운 접속을 늘려나감으로써 서버의 세션 자원을 고갈 시키는 공격 방식이다. 이는 단지 접속만 해지 하지 않는 정상 접속과 똑같고 공격 시 많은 트래픽을 유발 하지도 않는다. 특히 slow attack 같은 경우는 천천히 공격을 진행하여 서버의 세션 자원을 고갈시키기 때문에 공격 여부를 쉽게 판단하기 힘들다.

2) iptables 방어 정책

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [15717:3668417]
:Connection - [0:0]

-A INPUT -p tcp -j Connection
-A Connection -m recent --rcheck --seconds 50 --name badguy --rsource -j DROP
-A Connection -m connlimit --connlimit-above 15 --connlimit-mask 32 -m recent --set --name badguy --rsource -j DROP
-A Conncetion -j ACCEPT

COMMIT
```

그림 3. Connection Attack 방어 정책

Connection 기반의 공격 방식은 서버와 클라이언트가 항상 연결을 성립하여야 한다. 이는 iptables의 Connlimit module의 IP당 세션 수를 제한함으로써 비교적 효과적으로 대처 할 수 있다. 이를 그림 3.에 나타내었다.

3. Application Attack

1) 공격 방식

애플리케이션 형태의 공격은 VoIP의 경우 SIP 단말의 등록을 위한 REGISTER 패킷을 과도하게 요청하는 레지스터 스톰 공격, FTP 공격, Email 스팸, DNS 공격, SQL 공격 등의 각종 프로토콜의 취약점을 활용한 다양한 형태의 애플리케이션 공격이 존재한다. Cache Control 공격은 보통 웹 서버의 메모리에 있는 데이터를 이용해서 서비스하나, 캐시를 이용하지 말고 직접 요청하는 리퀘스트가 오면 DB를 다시 쿼리하여 해당 데이터를 직접 읽게 된다. 이로 인해 CPU의 부하가 높아지게 하는 형태의 공격 방법으로 HTTP 1.1부터 추가된 기능이다.

2) iptables 방어 정책

```
*filter
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:CC-string - [0:0]

-A INPUT -p tcp --dport 80 -j CC-string
-A CC-string -m string --algo bm --to 512 --string "no-store" -j DROP

COMMIT
```

그림 4. Cache Control Attack 방어 정책

그림 4.의 정책은 Application Attack의 대표적인 공격인 Cache Control(CC) Attack에 대한 방어 정책이다. CC Attack의 판별 여부는 클라이언트로부터 전송되는 HTTP 요청 패킷에 Cache-Control : no-store, must-revaildate 이란 내용이 포함 되어 있는지를 확인하여 판단한다. 따라서 CC Attack을 방어하기 위하여 string module을 사용하여 해당 문자열을 검사한다. 이에 해당하는 패킷을 차단하여 방어한다.

Ⅴ. 정책 적용에 대한 실험 및 평가

1. 실험 환경

총 5대의 PC로 실험 환경을 구성하였고 Victim은 피해자 PC이고 Attacker는 공격 Tool을 사용하여 공격을 실행하는 PC이다. 각 성능은 다음 표 1, 2와 같다.

표 2. Victim PC 사양

성 능		O S	회 선
CPU	RAM		
Dual 2.8GHz	4 GB	LINUX (CENTOS)	100 Mbps

표 3. Attacker PC 사양

성 능		O S	회 선
CPU	RAM		
Dual 2.4GHz	2 GB	LINUX (CENTOS) Windows 7	100 Mbps

Victim PC에서는 PORT 80으로 Apache 2.0[3]을 사용해 웹서비스제공 상태임을 가정 하였고 상태정보를 모니터링 하기 위해 dstat[4]를 사용하였다. (그림 5.)

lis	act	syn	tim	clo
13	2	0	0	0
13	2	0	0	0
13	2	0	0	0
13	2	0	0	0
13	2	0	0	0
13	2	0	0	0
13	2	0	0	0

lis : listen
act : established
syn : syn_recv
tim : time_wait
clo : close

그림 5. Victim 정상 상태의 dstat 결과

2. Flooding Attack 실험

이번 실험을 위해 Flooding Attack 공격유형의 가장 대표적인 syn flooding 사용 하였다.

Attacker PC는 synk.c 프로그램을 사용하였고 단일 IP로 Spoofing하여 공격 시도 하였다. 공격 시 증상은 무방비 상태일 경우 수초내 syn_recv가 한계치까지 올라가는 것을 확인하였고 웹 서비스 또한 불능 상태가 되었다.

lis	act	syn	tim	clo
10	1	0	0	0
10	2	40	0	0
10	1	452	0	0
10	2	746	0	0
10	1	1	0	0
10	1	1	0	0
10	1	1	0	0

그림 6. 공격 성공 시 dstat 결과

lis	act	syn	tim	clo
10	1	0	0	0
10	1	26	0	0
10	1	37	0	0
10	1	49	0	0
10	1	60	0	0
10	1	70	0	0
10	1	82	0	0

그림 7. 정책 적용 시 dstat 결과

그림 6.에서 보이는 결과는 syn 자원이 0, 40, 452, 746, 1000 순으로 변화 된 결과이다. dstat 특성상 1000을 1로 나타낸다.

정책 적용 후 같은 공격을 다시 실행 하여 상태를 확인하였다. (그림 7.) 그 결과 syn_recv의

증가 속도가 현저히 감소한 결과를 볼 수 있었다. 방어 정책에서 초당 12개의 syn 패킷을 허용하였기 때문에 일정한 속도로 SYN_RECV가 증가하였고 이는 임계 치를 적절히 조절함으로써 자신의 서버에 맞는 환경을 구축 할 수 있다.

3. Connection Attack 실험

이번 실험을 위해 Connection Attack 공격유형 중 slow attack을 실현할 수 있는 slowloris 프로그램을 사용 하였다. 이는 perl script로 실행 시 일정 주기로 서버와 세션을 반복적으로 연결하여 서버의 세션 자원을 고갈 시키는 툴이다. 공격 시 서버는 수십 분내 세션 자원이 완전히 고갈되어 웹서비스 불능 상태가 되었다.

lis	act	syn	tim	clo
14	2	0	0	0
14	2	0	0	0
14	154	1	0	0
14	174	1	0	0
14	204	1	0	0
14	239	1	0	0
14	254	0	0	0
14	301	0	0	0
14	335	0	0	0
14	335	0	0	0
14	387	4	0	0
14	387	4	0	0

그림 8. 공격 성공 시 dstat 결과

lis	act	syn	tim	clo
16	2	0	0	0
16	2	0	0	0
16	14	3	0	0
16	14	3	0	0
16	14	3	0	0
16	14	3	0	0
16	14	3	0	0
16	14	3	0	0
16	14	3	0	0
16	14	3	0	0
16	14	3	0	0

그림 9. 정책 적용 시 dstat 결과

정책 적용 후 반복한 실험에서 그림 9.와 같은 결과를 얻었다. Connection attack과 같이 공격 근원지가 확실한 경우에는 iptables 정책이 상당히 효과적으로 작용함을 볼 수 있었다.

4. Application Attack 실험

Application attack 공격유형중 지난 2009년에 발생한 7.7대란 때 사용되었던 방법 중 하나인 Cache-Control Attack을 선정하여 실험 하였다. CC attack을 구현하기 위해 DoSHTTP[5]을 사용하였다.

lis	act	syn	tim	clo	usr	sys	idl	wai	hiq	sig	
17	2	0	3	0	5	4	89	0	0	2	
17	2	0	3	0	0	0	99	1	0	0	
17	105	1	1	2	15	9	59	0	1	17	
22	133	110	4	1	38	22	0	0	0	40	
17	154	147	4	1	29	6	0	0	0	65	
17	186	181	7	3	30	18	0	0	0	52	
16	191	74	10	5	28	15	0	0	0	57	
17	168	164	12	2	27	16	0	0	0	57	missed 2 ticks
17	16	59	12	14	10	1	0	0	0	89	missed 2 ticks
17	26	15	12	28	9	1	5	0	0	85	
17	2	173	12	28	7	1	19	0	0	73	
17	2	384	12	28	7	1	19	0	0	73	
17	2	384	12	26	15	4	6	0	0	75	

그림 10. 공격 성공 시 dstat 결과

그림 10.은 공격이 성공적으로 실행되었을 때 결과이다. Application Attack은 시스템 자원도 동시에 고갈시키기 때문에 이를 확인하기 위해 dstat --tcp --cpu 옵션을 사용하여 시스템 자원도 동시에 확인 하였다. usr 항목은 사용자 시스템 영역, sys는 커널 시스템 영역, idl은 사용 가능한 시스템 자원을 나타낸다. 위 결과를 살펴보면 공격을 받은 시점에서 idl 즉 사용가능한 시스템 리소스가 급격히 고갈하는 것을 볼 수 있다. 얼마 지나지 않아 apache 2.0 프로그램이 비정상 작동하여 syn 자원이 급격히 증가하는 것을 확인할 수 있다.

lis	act	syn	tim	clo	usr	sys	idl	wai	hiq	sig
17	2	0	0	0	5	7	87	0	0	1
17	2	0	0	0	2	0	98	0	0	0
17	2	0	0	0	3	1	94	0	1	1
17	2	1	0	0	7	7	85	0	0	1
17	2	1	0	0	3	0	96	0	0	1
17	2	1	0	0	3	1	96	0	0	0
17	2	1	0	0	4	6	89	0	0	1
17	3	0	0	0	3	0	96	0	0	1
17	3	0	0	0	3	1	96	0	0	0
17	3	0	0	0	4	6	89	0	0	1
17	3	0	0	0	2	1	97	0	0	0

그림 11. 정책 적용 시 dstat 결과

그림 11.은 정책을 적용 후 상태를 보여주는 것인데 공격 패킷을 차단하여 정상 접속만 증가한 것을 볼 수 있다.

VI. 결 론

DDoS 공격 툴이 점점 다양해지고 그 사용법도 쉬워짐에 따라 전문지식이 없는 사람도 쉽게 DDoS 상황을 구현할 수 있게 되었다. 이는 개인 및 기업의 서버운영에 있어서 상당한 위협임이 틀림없다.

본 논문에서는 Linux 기반에서 별다른 비용 없이 DDoS 상황에 효과적으로 적용할 수 있는 iptables 정책을 설계하고 테스트 하였다. 실험 결과 모든 실험에서 어느 정도 효과가 있는 것으로 나타났다. 특히 Connection attack 유형에는 iptables 정책이 상당히 효과적 이였다. 하지만 IP Spoofing 되어 들어오는 공격에는 효과를 기대할 수 없었다. IP Spoofing 되어 들어오는 공격을 막기 위해 첫 SYN 패킷을 차단 후 재전송 되는 SYN 패킷을 확인하여 실존하는 사용자임을 확인 후 서비스를 제공하려 했지만 recent module의 한계로 인하여 적은 규모의 공격에서는 효과가 있었지만 공격 규모가 커질 경우 정상적인 서비스가 불가능 하였다. 또한 UDP Flooding와 같이 네트워크 Bandwidth 자원을 노린 공격에는 iptables 정책을 적용 시킬 수 없었다.

이러한 문제점들을 해결하기 위한 DDoS 전문 하드웨어 장비를 활용한 방어법과 iptables module의 한계점 및 개선 방법에 대한 연구가 필요하다. 또한 iptables에 국한된 방법이 아니라 Kernel configure를 통한 서버 설정 최적화에 관한 연구도 필요할 것으로 보인다.

참고문헌

[1] Michael Rash, "Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort", No Starch Press, 2007
[2] 홍석범, "리눅스 서버 보안관리 실무", (주) 수퍼유저코리아, 2008
[3] http://www.apache.org/
[4] http://dag.wieers.com/home-made/dstat/
[5] http://www.socketsoft.net/