**보고서**

**Computer Network System**

**-Network Security Tool-**

지상준

**1. 목적**

☞ DDOS 공격 프로그램과 이를 감시할 DDOS 감시 프로그램을 작성합니다.

☞ DDOS 공격 프로그램은 pthread library를 이용하여 여러 좀비 역할을 하는 스레드를 생성해두고, 좀비 스레드들에게 사용자의 공격 명령을 전달하기 위해서 UDS(Unix Domain Socket)를 사용합니다. 좀비들은 명령에 따라 TCP-SYN, TCP-FIN, UDP 공격하는 스레드를 만들거나, 이 공격 스레드를 죽이거나 합니다.

(UDS 관련 참고) <http://forum.falinux.com/zbxe/index.php?document_srl=406064&mid=network_programming>

☞ DDOS 감시 프로그램은 사용자로부터 패킷의 전송 속도를 입력 받습니다. raw socket으로 매번 패킷을 잡고 관련된 곳으로부터 패킷들의 전송 속도를 측정할 수 있도록 linked-list 형태로 저장합니다. 패킷 전송 속도가 입력 받은 속도보다 빠른 패킷들에 대해 감시하고 출력합니다.

**2. 기능**

☞ DDOS 공격 프로그램의 메인 함수는 생성된 좀비들을 사용자가 제어할 수 있도록 명령을 계속적으로 받아 들입니다.

제어 명령의 예로,

**start TCP-SYN 111.222.333.444:80 5**

첫번째 필드는 start 또는 stop을 이용하여 공격을 시작하라 또는 멈춰라는 기본적인 내용입니다.

두번째 필드는 공격의 타입을 지정하는 것이며, TCP-SYN, TCP-FIN, UDP를 사용한 공격 타입을 지시하는 내용입니다.

세번째 필드는 공격 당할 IP주소와 포트번호입니다. 이때 구분자로 콜론(:)을 이용합니다.

네번째 필드는 공격에 가담하거나 해당 공격을 멈출 좀비들의 수입니다.

예로 보인 명령어를 모두 멈추고 싶다면,

**stop TCP-SYN 111.222.333.444:80 5**

첫번째 필드를 stop으로 변경하면 됩니다. 여기서 공격에 가담하는 좀비들을 줄이고 싶다면

네번째 필드에 수를 4이하로 준다면 해당 주소에 공격 가담하는 좀비 수가 줄어듭니다.

쓰레드 풀처럼 좀비들을 미리 만들어 놓았습니다. 각 좀비들마다 각자 다른 공격지로 공격이 가능합니다.

☞ DDOS 감시 프로그램은 사용자로부터 경계값이 되는 속도를 pps나 bps 단위로 받아들입니다. 사용자로부터 명령을 받고 난 뒤 패킷 캡쳐를 시작하여 관련된 곳의 패킷을 묶어서 하나의 노드를 만들어서 linked-list에 저장하며, 패킷이 도착할 때마다 속도를 측정하여 경계값을 넘어서면 관련된 곳의 정보인 ip 주소, port 번호, 사용하는 프로토콜 정보를 뿌려줍니다.

**3. 구현(소스코드의 주석 내용도 참고 가능)**

☞ DDOS 공격 프로그램 :

(메인 함수 역할) : 우선 좀비들을 정해진 수 만큼 미리 만들어 두며, 이들과 연락할 소켓을 전역변수에다가 등록합니다. 이때 통신을 위한 파일이 필요하며, /tmp/test\_server 파일이 있어야 합니다.

#define FILE\_PATH "/tmp/test\_server"

이제 계속해서 사용자의 명령을 받아들여 좀비들에게 해당 소켓으로 명령을 전달합니다.

(좀비 역할) : 메인 함수에게 명령어를 받기를 기다리며 read()함수로 계속 Blocking 상태에 빠져있습니다. 명령이 들어오면 이를 읽어 들여 해독합니다. 공격 쓰레드를 만들거나 공격 쓰레드를 죽입니다. 그리고 다시 read() 함수로 명령이 들어오길 바랍니다.(결과적으론 좀비가 만들어낸 공격 쓰레드가 일을 하는 것)

☞ DDOS 감시 프로그램 :

(패킷 캡쳐를 위해서 libpcap library가 필요하며, 또 해쉬함수 MD5()를 이용하기 위해서 openssl library도 필요합니다.)

(메인 함수 역할) : 사용자에게서 경계값이 되는 속도를 입력 받아 전역변수에 표시해 둡니다. 패킷 캡쳐를 위한 함수를 셋팅하고, 캡쳐링 된 패킷을 처리하기 위한 콜백 함수 포인터를 pcap\_loop() 3번째 인자로 전달합니다.

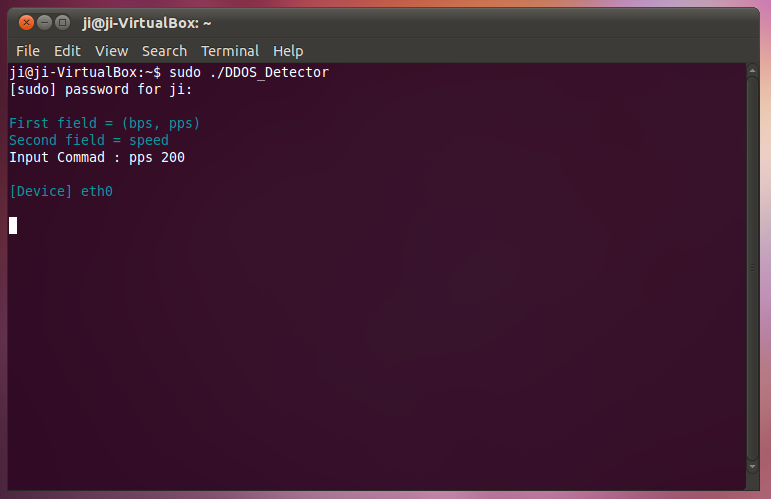
(캡쳐링 된 패킷을 처리하는 콜백 함수 역할) : 매번 캡쳐링 된 패킷의 양은 많으므로 양을 줄이기 위해서 HASH\_INFO 구조체에 해당 IP주소, 포트번호, 프로토콜이 같은 패킷들을 모아 하나의 노드를 만들도록 데이터를 정리합니다.(각 노드를 구별하는 기본정보는 MD5() 함수로 해쉬하여 결과값인 다이제스트를 이용합니다)

HASH\_INFO 구조체에서는 속도를 잴 수 있도록 노드가 만들어진 시간을 기록하는 변수가 있으며, 그 시간으로부터 누적되어온 패킷의 총 갯수, 총 몇 바이트가 들어왔는지를 기록하는 변수가 있습니다.

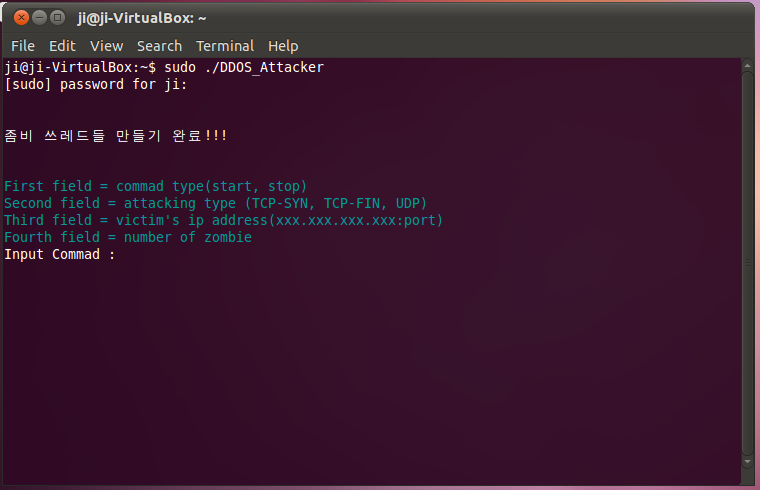
HASH\_INFO 구조체로 만들어진 하나의 노드를 linked-list 형태로 저장해두고 있다가, 패킷이 캡쳐되어 콜백 함수가 불리면 linked-list의 모든 노드를 검색하며 같은 기본정보의 노드가 있는지 다이제스트끼리 확인하며 노드가 없다면 노드를 만들거나, 노드가 있다면 노드에 기록을 누적하고 속도를 측정하여 사용자가 지정한 경계속도가 넘는지 아닌지 판단하고 출력합니다.

**5. 실행 예시**

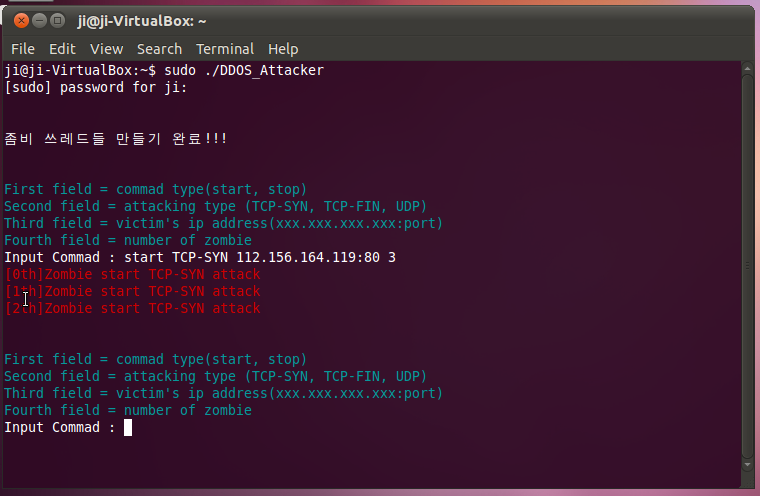
(DDOS\_Detector 실행 초기 화면)



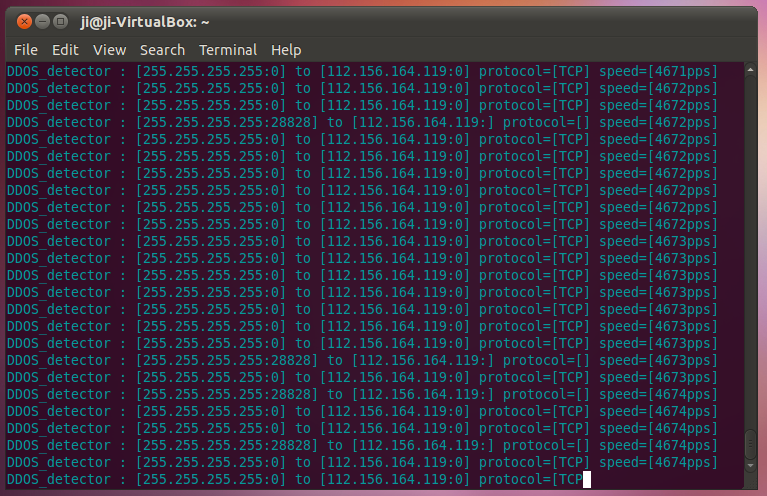
(DDOS\_Attacker 실행 초기 화면)



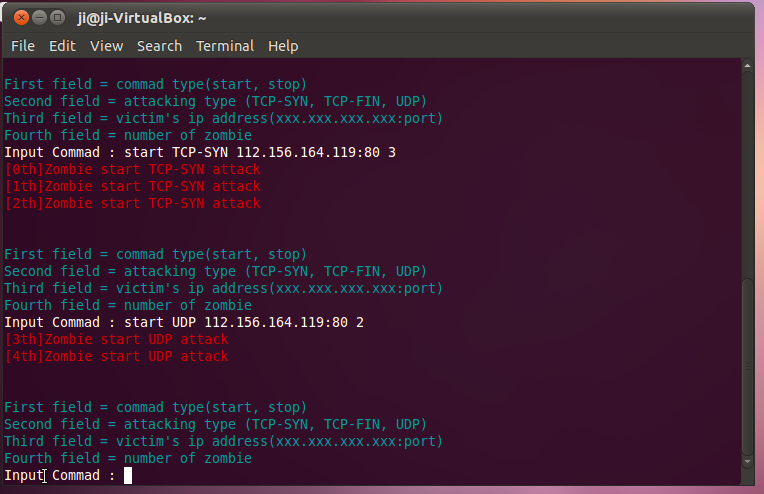
(DDOS\_Attacker 3마리 좀비들이 TCP-SYN 공격을 가하는 명령어를 내림)



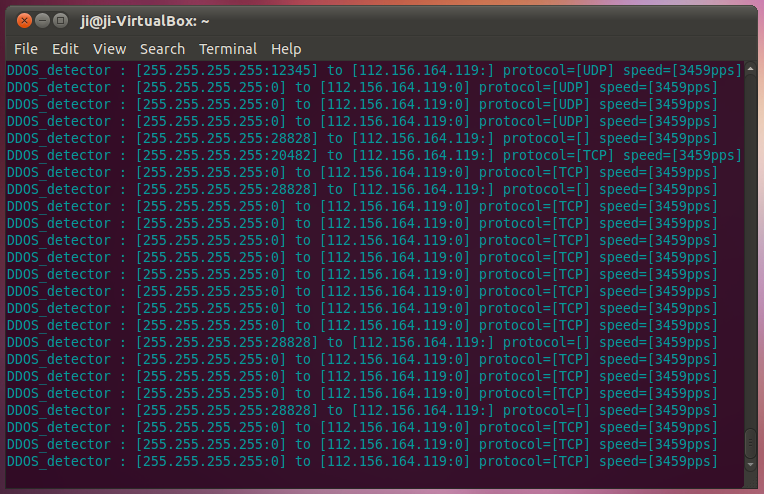
(DDOS\_Detector에서 잡히는 모습)



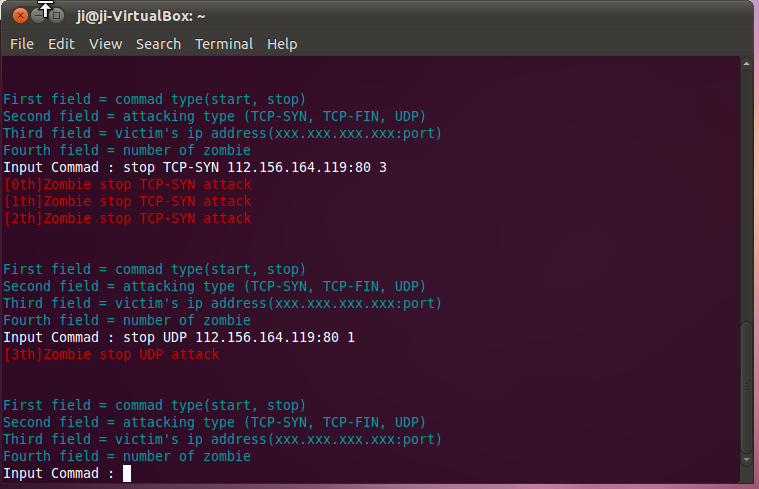
(2마리 좀비들이 UDP 공격을 가하는 명령어를 내림)



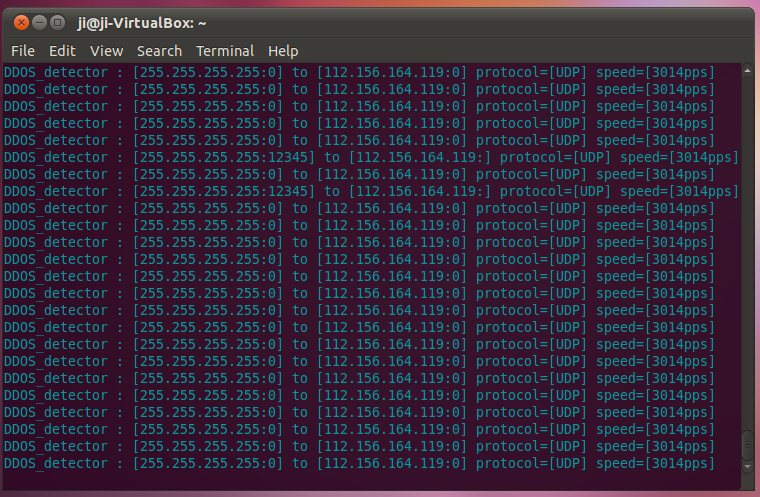
(DDOS\_Detector에서 잡히는 모습)



(해당 공격을 가하는 좀비들을 멈추게 하는 명령어를 내림. 아직 UDP 공격하는 좀비 하나 있음)



(DDOS\_Detector에서 잡히는 모습)



(모든 공격 종료)

