Recon Filtering Stateful Kernel Module Firewall

Etash Tyagi

February 2022

1 What Is It?

This is a Linux Kernel Module (LKM) stateful firewall, which can be loaded to the kernel to detect and block well known reconnaissance techniques, in order to reduce the information available to the scanner. Following are the scans it protects against in terms of Nmap:

1. **TCP_ACK:**
    a) *nmap -sA*
    b) *nmap -sW*
2. **TCP_XMAS:** *nmap -sX.*
3. **TCP_FIN:** *nmap -sF.*
4. **TCP_NULL:** *nmap -sN*
5. **TCP_RFC_O:**
    a) *nmap –scanflags PSH*
    b) *nmap –scanflags URG*
    c) *nmap –scanflags PSHURG*
    d) *nmap –scanflags PSHFIN*
    e) *nmap –scanflags URGFIN*
6. **TCP_SYN:**
    a) *nmap –sS*
    b) *nmap –sT* (log only on closed ports)

Note: This module was tested on Ubuntu 21.10 kernel 5.13.0-28-generic and ParrotOS kernel 5.14.0-9parrot1-amd64.

2 How To Make It Work?

2.0 Requirements

This module is dependent on 'conntrack' for connection tracking, which is useful for stateful firewalls. This can be installed using '**apt install conntrack**'. Also, iptables must have a connection tracking rule (Makefile line 11, removed when unloaded line 14).

2.1 Build The Module

The module can be simply be built using the provided 'Makefile' using: '**sudo make**'.

2.2 Run The Module

The module can be simply loaded using '**sudo make run**'. This starts the logging and blocking process.

2.3 Stop The Module

To unload the module, use: '**sudo make stop**'. This stops the logging and blocking process.

2.4 See Syslogs

To see syslogs (via dmesg), use: '**sudo make syslog [REGEX=...]**'. REGEX can be set to specify the searching criteria, currently all packets labeled as reconnaissance are shown.

PTO

3 Internal Workings

3.1 Scan Type Recognition

3.1.1 TCP_ACK

This is logged when TCP packet containing [ACK], is recieved but the connection is not ESTABLISED yet, i.e. is NEW (recon_ghost.c line 122).

3.1.2 TCP_XMAS

This is logged when illegal (see section 3.1.5) TCP packet containing only [FIN, PSH, URG] flags is received (recon_ghost.c line 84).

3.1.3 TCP_FIN

This is logged when illegal (sec 3.1.5) TCP packet containing only [FIN] is received (recon_ghost.c line 92).

3.1.4 TCP_NULL

This is logged when illegal (sec 3.1.5) TCP packet containing no flags is received (recon_ghost.c line 100).

3.1.5 TCP_RFC_O

This consists of other illegal packets according to 'TCP RFC'. According to this rule, every packet must have atleast one of: [ACK—RST—SYN], hence the packets not following this rule, and not caught by XMAS, FIN, or NULL are caught here (recon_ghost.c line 81 and 108).

3.1.6 TCP_SYN

This is detected either during the failure of 3-way handshake, when client sends [RST] instead of [ACK], or when port being scanned by -sS, -sT (or manual TCP) is not open (recon_ghost.c line 124 and 174).

3.2 Scan Blocking

3.2.1 TCP_ACK

We '**NF_DROP**' all the packets following criteria in 3.1.1 (recon_ghost.c line 142). This makes all ports appeared filtered.

3.2.2 TCP_XMAS

We '**NF_DROP**' even the packets whose port is closed (recon_ghost.c line 142). This makes all ports appear open||filtered.

3.2.3 TCP_FIN

Same as 3.1.2.

3.2.4 TCP_NULL

Same as 3.1.2.

3.2.5 TCP_RFC_O

This also makes all ports appear filtered using the same method in 3.1.2.

3.2.6 TCP_SYN

- **Open Ports:** Since all packets send SYN to establish connection first, we can not directly drop it, as it will make our exposed port unreachable. Hence here when the client sends RST in 3-way handshake, we just log it.
- **Closed Ports:** When the port is closed, server replies with RST to client's SYN request. Here using the output netfilter, we change this RST packet to a dummy [SYN, ACK] packet (recon_ghost.c line 173), signifying to client that the port is open (when it is actually not).

We then log and **'NF_ACCEPT'** the packet if it is this kind of scan. Using the above mentioned technique, all ports with *nmap -sS* are shown as open (even the closed ones, hence not providing any info to attacker). Even in case of *nmap -sT*, mapping closed port logs the entry.

PTO

4 Testing

Following things should work:

- When firewall is enabled result of open and closed port should be mainly different.
- When firewall is enabled aforementioned nmap scans should be recognized and blocked.
- The device on which firewall is running should not be hindered by the firewall.

For the sake of testing, we can enable port 80 on the machine having the firewall, and compare it's output with closed port 443. I have chosen these ports as we can '**curl**' to both of them if the port is open (http, https).

The test script is named test.sh and can be run using '**sudo ./test.sh [ip] [port1] [port2]**'

Now from the output of script when firewall is disabled (figure 1) we can see that the second parameter (result) of both packets is mostly different.

However when we enable the firewall in (figure 2), the results are the same (except the last, which is TCP full scan).

The syslogs are shown in figure 3 (first is because of curl and last because of tcp full scan (-sN)).

Figure 4 shows that the device on which firewall is running, is able to make http requests.

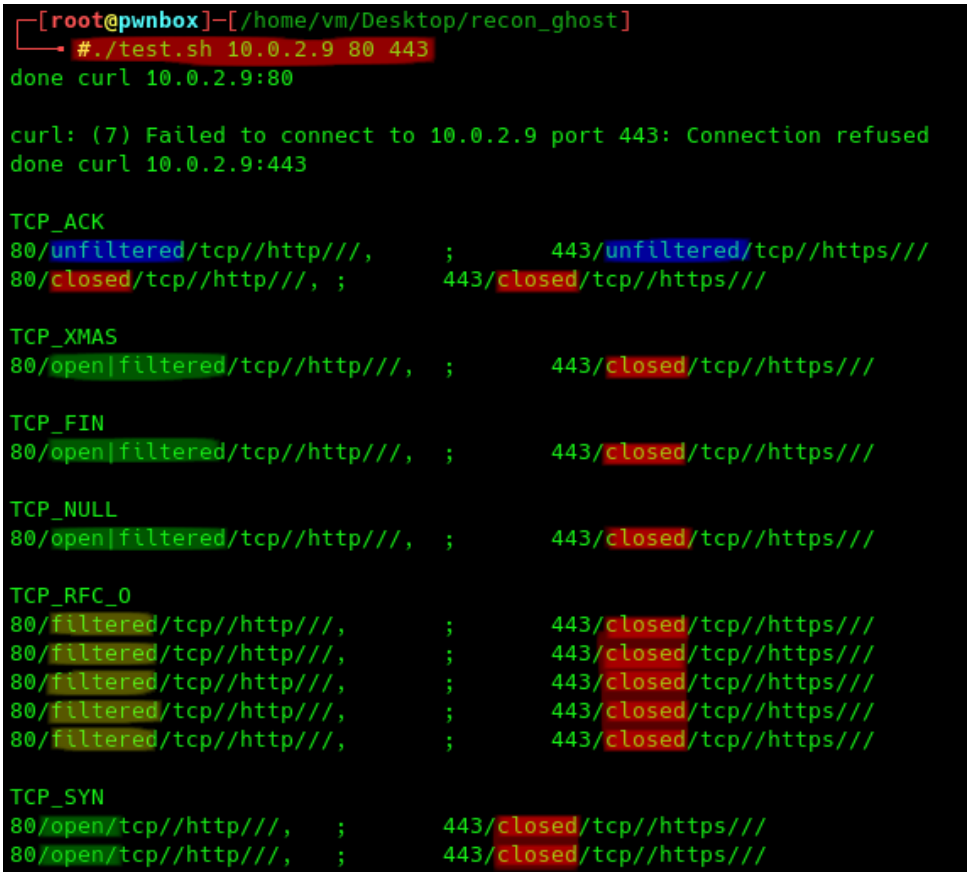

Figure 1: Test script when firewall disabled

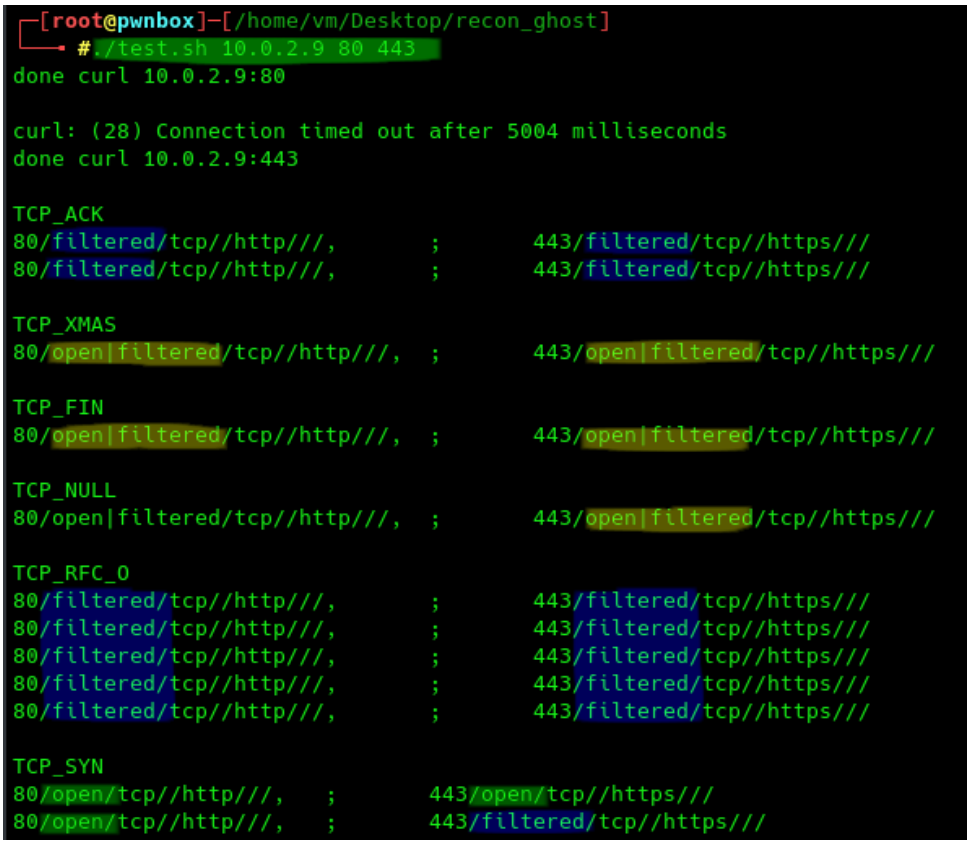

Figure 2: Test script when firewall enabled

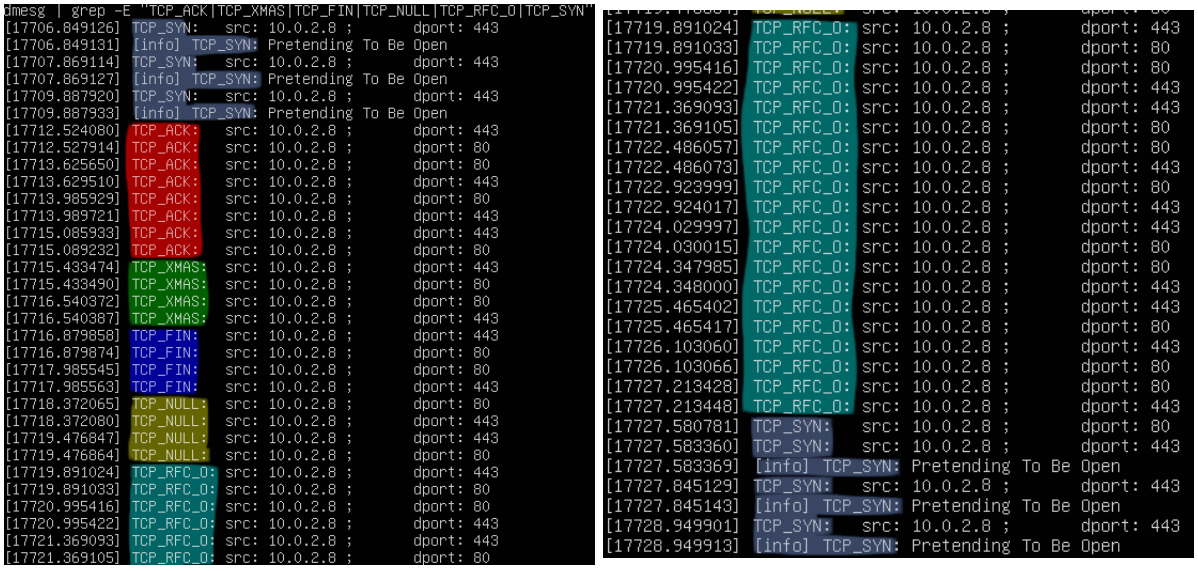

Figure 3: Syslogs

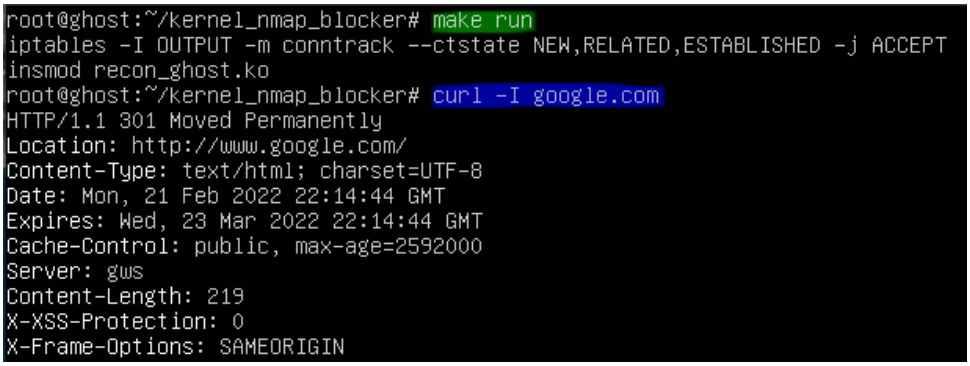

Figure 4: Curl on enabled firewall