Does tcpdump bypass iptables?

Asked 7 years, 2 months ago
Modified 2 years, 1 month ago
Viewed 32k times

72

23

I mistakenly setup open resolver DNS server, which was soon used for a bunch of DDoS attacks originating somewhere from / to Russia. For that reason I completely blocked port 53 on both DNS servers for everyone except for trusted IP's. It does work, in that I am not able to connect to them anymore, but what seems weird to me is that when I run tcpdump on `eth1` (which is interface on server with public Internet) I see lots of incoming packets from attacker to port 53.

Is it normal that tcpdump displays these packets even if iptables drops them? Or did I configure iptables wrongly?

On other hand I don't see any outgoing packets from my server, which I did before, so I suppose that the firewall is kind of working. It just surprises me that the kernel doesn't drop packets entirely? Or is `tcpdump` hooked to the kernel in a way that it sees the packets even before they get to iptables?

networking | iptables | tcpdump

Share
Improve this question
Follow

edited Jun 23, 2020 at 22:25
auspicious99
117 1 10

asked Jun 8, 2015 at 15:05
Petr
2,053 5 25 36

1 Answer
Sorted by:

Highest score (default)

99

This is a nice question.

As a matter of fact, *tcpdump* is the first software found after the wire (and the NIC, if you will) on the way *IN*, and the last one on the way *OUT*.

Wire -> NIC -> tcpdump -> netfilter/iptables

iptables -> tcpdump -> NIC -> Wire

Thus it sees all packets reaching your interface, and all packets leaving your interface. Since packets to port 53 do not get a reply, as seen by tcpdump, you have successfully verified that your iptables rules have been correctly configured.

EDIT

Perhaps I should add a few details. *tcpdump* is based on *libpcap*, a library which creates a *packet socket*. When a regular packet is received in the network stack, the kernel **first** checks to see whether there is a packet socket interested in the newly arrived packet and, if there is one, it forwards the packet to that packet socket. If the option *ETH_P_ALL* is chosen, then **all** protocols go thru the packet socket.

libpcap implements one such packet socket with the option activated, keeps a copy for its own use, and duplicates the packet back onto the network stack, where it is processed by the kernel in the usual way, including passing it first to *netfilter*, the kernel-space counterpart of *iptables*. Same thing, in reverse order (*i.e.*, first netfilter then last the passage thru the packet socket), on the way out.

Is this prone to hacking? But of course. There are certainly proof-of-concept rootkits using *libpcap* to intercept communications destined to the rootkit *before* the firewall can lay its hand on them. But even this pales in comparison with the fact that a simple Google query unearths **working** code hiding traffic even from *libpcap*. Still, most professionals think the advantages vastly outweigh the disadvantages, in debugging network packet filters.

Share
Improve this answer
Follow

edited Aug 9, 2018 at 13:17

answered Jun 8, 2015 at 17:00
MariusMatutiae
45.8k 10 77 127

Is there a way to display it so that I can see which packets were allowed and which were dropped?
– Petr
Jun 9, 2015 at 11:42

!

@Petr you can log the packets which iptables dropped, thegeekstuff.com/2012/08/iptables-log-packets
– MariusMatutiae
Jun 9, 2015 at 12:07

So is there a way that I can drop packets even before tcpdump can see them?
– 4253wyerg4e
Feb 15 at 22:51

@4253wyerg4e And why do you want to drop them even before seeing them?
– MariusMatutiae
Feb 17 at 6:31