Lab, Computer Security
Spring Term 2021

Lab 3

DA273B VT21

Niklas Lantau

Date: May 12, 2021

Faculity of Natural Science

Table of Contents

1. Configuring Iptables

The goal with this lab is to configure Iptables with a narrow and specific set of rules (step 4 in given instructions). Figure 1 shows the final configuration of Iptables.

The essential main difference between step 3 and step 4 in the instructions were:

- Step 3: Allow web traffic.
- Step 4: Allow only *outgoing* web traffic.

In practice, to satisfy step 3 of the given instructions the only needed change of the configuration in Figure 1 would be to add NEW to INPUT chain on source ports 80 and 443. The configuration of SSH was added due to the ease of live demonstration during the Zoom session. The configuration for SSH allows for computers residing on the same subnet to initiate an SSH session with the host, which will be honoured when restoring and changing the other rules of the Iptables configuration, thus not dropping the SSH session during the live demonstration.

Iptables is first configured to have its default chains DROP all traffic. The following policies are appended to allow the specific traffic:

- Web traffic
  - Allow ESTABLISHED and RELATED incoming TCP traffic from ports 80 and 443.
  - Allow NEW, ESTABLISHED and RELATED outgoing TCP traffic to ports 80 and 443.
- SSH
  - Allow NEW, ESTABLISH and RELATED incoming TCP traffic on destination port 22 from local subnet.
  - Allow ESTABLISHED outcoing traffic.
- UDP (for DNS)
  - Allow ESTABLISHED and RELATED incoming UDP traffic from port 53.
  - Allow NEW, ESTABLISHED and RELATED outgoing UDP traffic to port 53.
- Logging
  - LOG and DROP all NEW incoming on ports 80 and 443.

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     udp  --  wlp0s20f3 *     0.0.0.0/0            0.0.0.0/0           udp spt:53 ctstate RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  wlp0s20f3 *     0.0.0.0/0            0.0.0.0/0           multiport sports 80,443 ctstate RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  wlp0s20f3 *     192.168.86.0/24      0.0.0.0/0           tcp dpt:22 ctstate NEW,RELATED,ESTABLISHED
    0     0 logdrop    tcp  --  wlp0s20f3 *     0.0.0.0/0            0.0.0.0/0           multiport dports 80,443 ctstate NEW

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     udp  --  *      wlp0s20f3  0.0.0.0/0            0.0.0.0/0           udp dpt:53 ctstate NEW,RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      wlp0s20f3  0.0.0.0/0            0.0.0.0/0           multiport dports 80,443 ctstate NEW,RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      wlp0s20f3  0.0.0.0/0            0.0.0.0/0           tcp spt:22 ctstate ESTABLISHED

Chain logdrop (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 LOG        all  --  *      *       0.0.0.0/0            0.0.0.0/0           limit: avg 5/min burst 10 LOG flags 0 level 4
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0


 > Nameserver: 192.168.86.1
 > Nic       : wlp0s20f3
 > Logging   : NEW INPUT dports 80,443

------------

 > Logging New attempts on dports 80,443
 > Run "viewlog" to stream log
```

Figure 1. Iptables settings

2. Scripting

Five scripts were authored to automate serveral tasks, including:

- **firewall**:
  - Setting policies.
  - Saving the start date and time of the the firewall as a dummy-file, to be used in *savelog*.
  - Make modified configurations persistent by saving to */etc/iptables/iptables.rules*
- **restore_firewall**:
  - Flushing default chains (INPUT, FORWARD, OUTPUT).
  - Flushing non-default chains (logdrop)
  - Flushing *nat* and *mangle*.
  - Restoring */etc/iptables/iptables.rules*.
- **savelog**:
  - Locate the start date and time created by *firewall*.
  - Fetch and parse relevant logged data from *journalctl* to *latest.log* and the trimmed version *pretty_latest.log*.
  - Remove dummy-file created by *firewall*.
- **viewlog**:
  - Continuously print logged data parsed from *journalctl* to standard output.
- **curl_test**:
  - To be run on other UNIX computer on same subnet.
  - Locates local IPv4 for *archcrypt.lan* (hostname). Uses *nmap* for host detection if IPv4 for *archcrypt.lan* is not found.
  - Allows for the ease of testing to fetch a file (called *dummyfile*) from the computer running *firewall*.
  - Enables a sudo user to specify source port and destination port in a *cURL* command for testing.

3. Testing

- Testing Process - Host
  - Execute *./firewall -lv* (flags: logging and verbose output).
  - Execute *./viewlog*
- Testing Process - Client
  - Execute *./curl_test 80 443* (source and destination ports).

If the *client* uses source port 80 or source port 443, logging of the NEW INPUT attempts will occur on the *host machine*. This can be viewed live by running *viewlog* on the *host machine*. After *client* has tested various ports, *host* should execute *./savelog* to save the logged and parsed data to *latest.log* and *pretty_latest.log*.

```
File: pretty_latest.log

IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=60154 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=60192 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=80 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=80 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=80 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=80 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=80 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=80 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=80
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=443
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=443
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=443
IN=wlp0s20f3 SRC=192.168.86.169 DST=192.168.86.149 PROTO=TCP SPT=443 DPT=443
```

Figure 2. Example log *pretty_latest.log*. Source and destination ports highlightet in yellow.

The added chain *logdrop* will only log a burst of maximum 10 attempts, then it will a most log 5 attempts each minute. This is to ensure that malicious attempts will not create a huge log file on the host. Once logged, the packages are dropped. Figure 2 demonstrates the saved data from the parsed output of the *journalctl* log. A second file, *latest.log* will be available as well, containing more data (although some what more trimmed than the orginal data stored in *journalctl*.

As shown in Figure 3, the Iptables policies clearly blocks and logs the attempts of the clients request from port 80 and 443. The usage of logging and SSH made it possible to demonstrate a working firewall and thus verifying the used policies. By using *cURL*, both source and destination port were possible to be specified.

The black backgrounded terminals in Figure 3 and Figure 4 are active SSH-sessions, logged into the host of the running Iptables configurations. The white backgrounded terminals in Figure 3 and Figure 4 are the client. The host is running Arch Linux and the client is running MacOS.

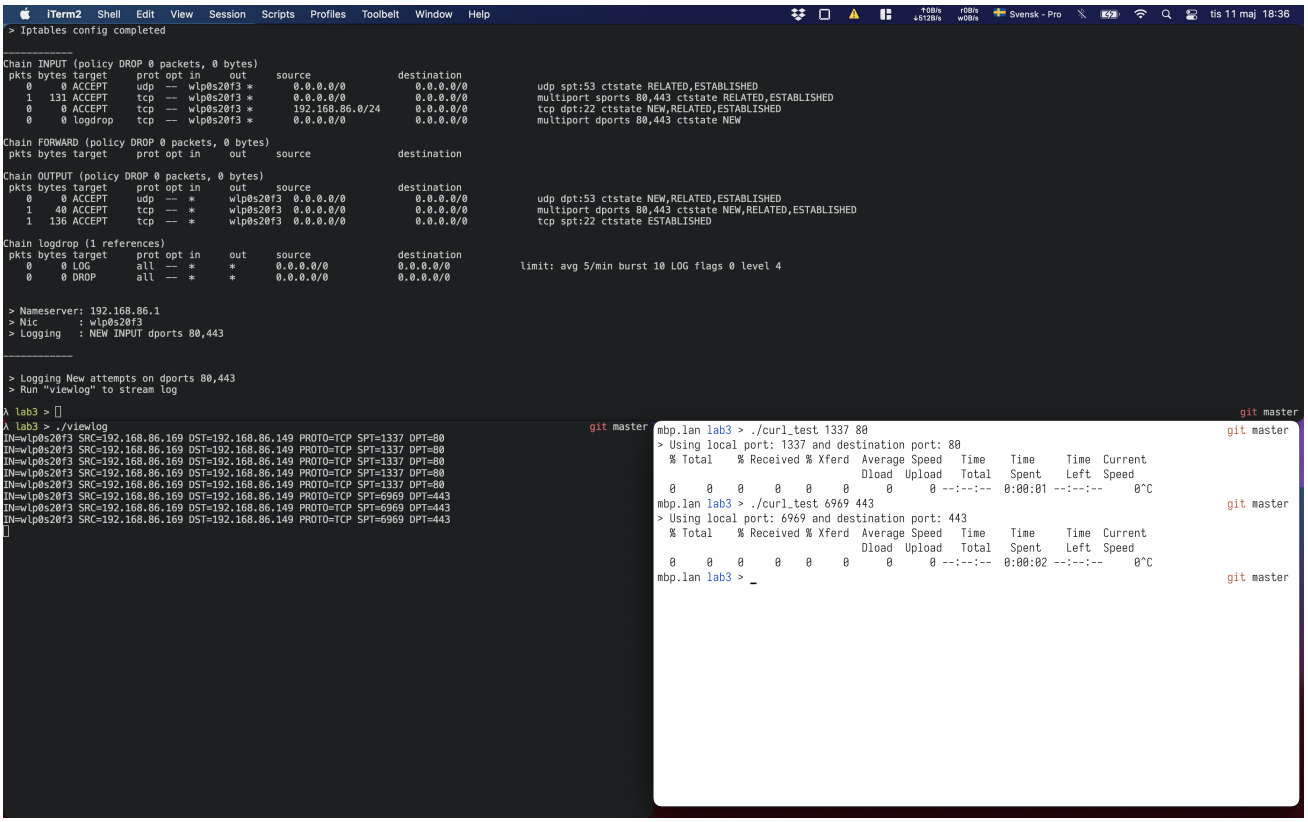

Figure 3. Back terminals: SSH-session. White Terminal: MacOS. Firewall active blocking curl-request and logging attempts.

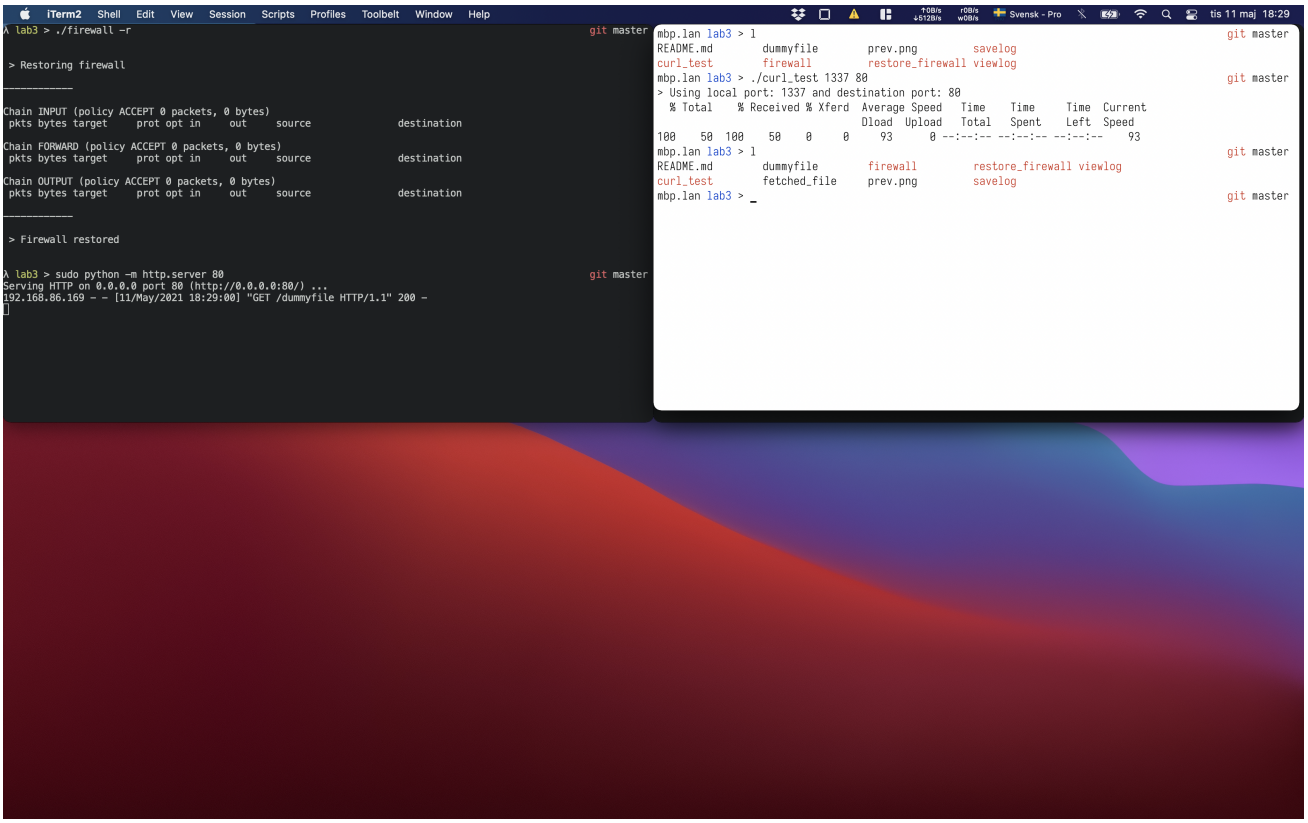

Figure 4. Black terminal: SSH-session. White Terminal: MacOS. Firewall turned off, allowing curl-request.