Network Security

Assignment 4, Thursday, September 24, 2015, version 1.1

Handing in your answers: Submission via Blackboard (http://blackboard.ru.nl)

Deadline: Friday, October 2, 23:59:59 (midnight)

In this exercise you will be using the following tools:

- iptables: http://netfilter.org/projects/iptables/index.html
- sshuttle: https://github.com/apenwarr/sshuttle

Do not compile these programs from source. iptables is likely installed by default. Use your Linux distribution's package manager to install the package sshuttle (e.g. Ubuntu / Debian: `apt-get install sshuttle`, Arch: `pacman -S sshuttle`, Fedora: `yum install sshuttle`, Gentoo: `emerge sshuttle`).

This assignment consists of some theoretical questions and two practical exercises. Please turn in all your work in plain text files (program source files are also plain text). If you prefer a document with formatting for whatever reason, like including images, use the PDF format to turn in your work (most editors allow you to export to PDF). Note that it's okay to include images separately and then refer to them from within the text files.

Many commands in this exercise need to be run with root rights. This is denoted by a prefix `#`. When a command should be run without root rights, it will be prefixed with `$`. Do not include the prefix when typing the command.

1. In this exercise you will use iptables to create two firewall configurations: one for a client machine, one for a masquerading server. You are encouraged to test your configuration on your own (virtual) machine.

   You can use the commands `iptables-save` and `iptables-restore` to save and restore iptables rules to and from a file, respectively. Usage: `# iptables-save > filename` stores the firewall configuration in the file `filename`. `# iptables-restore < filename` restores the firewall configuration from `filename`. It might be a good idea to run `iptables-save` on the firewall configuration you have before starting this exercise.

   If you get in unrecoverable trouble, you can completely reset the firewall configuration by running the following commands:

   ```
   # iptables -F
   # iptables -X
   # iptables -t nat -F
   # iptables -t nat -X
   # iptables -t mangle -F
   # iptables -t mangle -X
   # iptables -P INPUT ACCEPT
   # iptables -P FORWARD ACCEPT
   # iptables -P OUTPUT ACCEPT
   ```

   This will flush (`-F`) all built-in chains and delete (`-X`) all user-defined chains in the standard tables, and set the default policy (`-P`) to accept.

   Create a folder called `exercise1` to hold the answers to this exercise.

   Note that under some Linux distributions (most notably Ubuntu), you may have to add a rule allowing traffic from localhost to localhost in order to allow some local processes to communicate.

   (a) Use the iptables manpage (`man iptables`), the netfilter documentation on http://www.netfilter.org/documentation/index.html#documentation-howto (especially the Packet Filtering HOWTO), the lecture slides, and any sources you want to build a client firewall that does the following:

   - Allow all outgoing traffic.

- Deny all incoming traffic, except
  - traffic that belongs to an established connection, and
  - incoming SSH traffic (filter on transport protocol and port).
- Allow all ICMP traffic, except ICMP redirects.[1]

If you use tutorials or examples, please make sure you understand what the rules do.

Write your firewall configuration, preferably dumped by `# iptables-save`, to `exercise1a.fw`

(b) If you felt exercise 1a was easy, try your hand at this one. Otherwise, skip to exercise 2 and come back if you have time left over.

Do the same for a masquerading server / router with two network cards: `eth0` with IP address 198.51.100.42 and `eth1` with address 10.0.0.1/24. `eth1` is the internal card, the internal network should be obvious from its address. `eth0` is the external card, which is the link to the Internet. The firewall should do the following:

- Masquerade traffic coming from the local network going to the Internet.
- Allow outgoing traffic to the Internet that's forwarded from the local network.
- Block all outgoing traffic from the machine itself to the Internet, except for ICMP and traffic that belongs to an established connection.
- Allow all incoming ICMP traffic.
- Allow all outgoing traffic from the machine itself to the local network.
- Block all incoming traffic from the Internet, except traffic that belongs to an established connection.
- Accept incoming TCP connections on port 80 (let's say that's a webinterface) and port 22 from the Internet.
- Forward TCP and UDP traffic on port 2222 and 8080 to some other host in the local network. Feel free to pick that host yourself.

Since you likely cannot test this, simply give it your best shot. This is the type of configuration you'd use if you use a Linux machine as your home router.

Write your firewall configuration, preferably dumped by `# iptables-save`, to `exercise1b.fw`

2. In this exercise you will use sshuttle to set up a secure tunnel to the lilo login server of the Faculty of Science, and then inspect and analyze the resulting iptables rules. For this, you will need to use your Faculty of Science account. If you do not have one, and do not have access to an alternative SSH server on which sshuttle works, send me an e-mail as soon as possible.

   Use the manpage of sshuttle (`man sshuttle`) to figure out the command to route everything through the VPN. The remote host to use is ¡username¿@lilo.science.ru.nl, where ¡username¿ is your Science login name. Write the command you use to `exercise2`. Remember to run sshuttle as root.

   Now, view the resulting iptables configuration using either `# iptables -t <table> -L` for each table listed in the manual page, or use `# iptables-save`. Write the rules to `exercise2`, and explain what they do and why they route everything through the VPN. Try e.g. looking for the port number you see in a listing of listening ports.

   Feel free to play with other configurations (e.g. routing only certain networks through the VPN, or using exceptions) and explain what the different firewall rules for these configurations do as well.

[1] There are other ways to handle ICMP redirects securely, however. E.g. on Linux, there is the /proc/sys/net/ipv4/conf/*/secure_redirects directive, which tells the kernel to ignore all ICMP redirects on that interface that don't redirect to an already known gateway.

3. In a later lecture you will be told something about OpenVPN. It is another form of VPN software than sshuttle. For now, the main difference you need to understand is that OpenVPN provides a virtual ethernet interface to route traffic through, in contrast to sshuttle which redirects traffic using iptables. The result is that the routing table contains rules to route normally, as well as rules to route traffic over the VPN.

   Create a folder called `exercise3` to hold the answers for this exercise.

   My IP address is 145.116.128.31/22. When I'm not connected to my VPN, my routing table looks like this:

```
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags  Iface
0.0.0.0         145.116.128.1   0.0.0.0         UG     wlp3s0
145.116.128.0   0.0.0.0         255.255.252.0   U      wlp3s0

$ ip r show
default via 145.116.128.1 dev wlp3s0
145.116.128.0/22 dev wlp3s0  proto kernel  scope link  src 145.116.128.31
```

   Let's say that my VPN runs on a machine with IP address 198.51.100.42. When I connect to my VPN, a new interface (`tap0`) is created, and the routing table is changed (I have slightly altered the output for clarity):

```
$ route -n
Kernel IP routing table
    Destination     Gateway         Genmask         Flags  Iface
 1. 0.0.0.0         10.50.9.1       128.0.0.0       UG     tap0
 2. 128.0.0.0       10.50.9.1       128.0.0.0       UG     tap0
 3. 10.50.9.0       0.0.0.0         255.255.255.0   U      tap0

 4. 10.0.0.0        145.116.128.1   255.0.0.0       UG     wlp3s0
 5. 172.16.0.0      145.116.128.1   255.240.0.0     UG     wlp3s0
 6. 192.168.0.0     145.116.128.1   255.255.0.0     UG     wlp3s0

 7. 0.0.0.0         145.116.128.1   0.0.0.0         UG     wlp3s0
 8. 131.174.117.20  145.116.128.1   255.255.255.255 UGH    wlp3s0
 9. 145.116.128.0   0.0.0.0         255.255.252.0   U      wlp3s0
10. 198.51.100.42   145.116.128.1   255.255.255.255 UGH    wlp3s0


$ ip r show
 1. 0.0.0.0/1 via 10.50.9.1 dev tap0
 2. 128.0.0.0/1 via 10.50.9.1 dev tap0
 3. 10.50.9.0/24 dev tap0  proto kernel  scope link  src 10.50.9.60

 4. 10.0.0.0/8 via 145.116.128.1 dev wlp3s0
 5. 172.16.0.0/12 via 145.116.128.1 dev wlp3s0
 6. 192.168.0.0/16 via 145.116.128.1 dev wlp3s0

 7. default via 145.116.128.1 dev wlp3s0
 8. 131.174.117.20 via 145.116.128.1 dev wlp3s0
 9. 145.116.128.0/22 dev wlp3s0  proto kernel  scope link  src 145.116.128.31
10. 198.51.100.42 via 145.116.128.1 dev wlp3s0
```

Other relevant information is in the DHCP leases I got:

```
$ dhcpcd --dumplease wlp3s0
dhcp_server_identifier=131.174.117.20
domain_name_servers=131.174.117.20
ip_address=145.116.128.31
network_number=145.116.128.0
routers=145.116.128.1
subnet_cidr=22
subnet_mask=255.255.252.0

$ dhcpcd --dumplease tap0
dhcp_server_identifier=10.50.9.1
ip_address=10.50.9.60
network_number=10.50.9.0
subnet_cidr=24
subnet_mask=255.255.255.0
```

For all the following questions, keep in mind that a more specific route (i.e. one that applies to a smaller network, a smaller number of hosts) overrides more generic routes. So a route with a netmask of 255.255.255.0 (a /24) overrides any route with a netmask of 255.0.0.0 (a /8) that covers the same hosts.

Also keep in mind that the VPN server is a machine with IP address 198.51.100.42. Internally the VPN uses the network 10.50.9.0/24, as can be seen in the dhcp lease for `tap0`. Finally, the dhcp protocol requires periodic communication with the dhcp server to keep the address lease active.

In these questions, when asked "where traffic goes", please answer with the Gateway IP address and the interface.

(a) Look at routes 1, 2, and 7. Where does traffic not matched by any of the other routes go, and why? Write your answer to `exercise3a`.

(b) Route 9 is one of the two original routes, also present when the VPN is not active. Similarly, routes 4–6 are always added by my VPN setup script. Explain what these routes accomplish. What traffic do they match, where does that traffic go, and why?

Note the IP ranges used, and try to imagine the usage scenario for a VPN. Write your answers to `exercise3b`.

(c) Explain why route 3 is necessary, in light of what routes 4–6 accomplish. Look at the dhcp lease for the `tap0` interface. Write your answer to `exercise3c`.

(d) Look at route 8 and the dhcp lease for the `wlp3s0` interface. Why is route 8 necessary? What happens if it is not present? Write your answer to `exercise3d`.

(e) Route 10 is the most important route present. Explain what it does, and what would happen if it was *not* present. Also try to explain what happens when I connect e.g. to an SSH server running on the same machine as the VPN server. Does that traffic get tunneled or not? Explain why. Write your answers to `exercise3e`.

4. Take a look at RFC 5508, "NAT Behavioral Requirements for ICMP" (http://tools.ietf.org/rfc/rfc5508.txt). Read sections 2 (especially the part on "ICMP Message Classification"), 3, 4, and 10 (looking at 9 for the requirements).

   Create a folder called `exercise4` for the answers.

   - Why should a NAT drop inbound ICMP Error messages which do not belong to an existing NAT session? What should it do with inbound ICMP Error messages which do belong to an existing NAT session, and why? Write your answers to `exercise4a`.

   - Why should a NAT drop outbound ICMP Error messages which do not belong to an existing NAT session? What should it do with outbound ICMP Error messages which do belong to an existing NAT session, and why? Write your answers to `exercise4b`.

   - Explain in your own words how NAT for ICMP works for the three different kinds of ICMP packets. Highlight some (two or three) security concerns, explain them, and explain how they are mitigated. Write your answers to `exercise4c`.

5. Place the files and directories `exercise1`, `exercise2`, `exercise3`, and `exercise4`, and all their contents in a folder called `netsec-assignment4-STUDENTNUMBER1-STUDENTNUMBER2`. Replace `STUDENTNUMBER1` and `STUDENTNUMBER2` by your respective student numbers, and accomodate for extra / fewer student numbers. Make a `tar.gz` archive of the whole `netsec-assignment4-STUDENTNUMBER1-STUDENTNUMBER2` directory and submit this archive in Blackboard.