IV1013 VT17 – Introduction to Computer Security

Packet Filter Firewalls

Overview

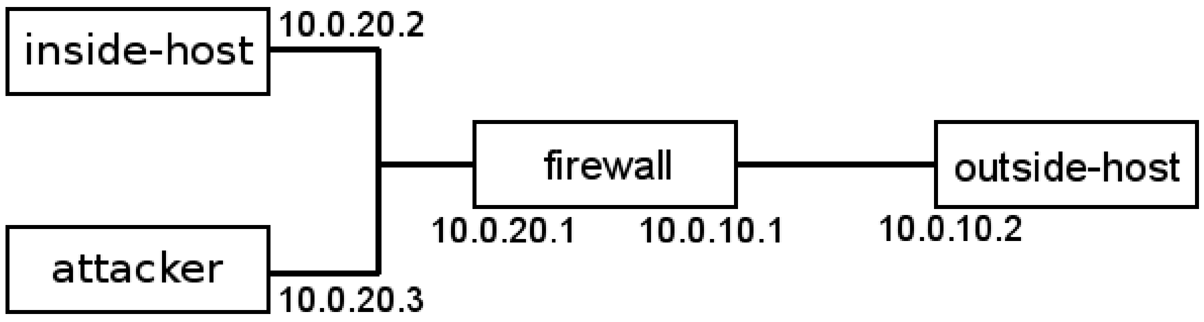
The learning objective of this lab is for students to gain the insights on how firewalls work by playing with firewall software and implement a simplified packet filtering firewall. Firewalls have several types, and the two most commonly known are the packet filter and application level gateway (application firewall). Packet filters act by inspecting the packets; if a packet matches the packet filter's set of rules, the packet filter will either drop the packet or forward it, depending on what the rules say. Packet filters are usually stateless; they filter each packet based only on the information contained in that packet, without paying attention to whether a packet is part of an existing stream of traffic. Packet filters often use a combination of the packet's source and destination address, its protocol, and, for TCP and UDP traffic, port numbers. In this lab, the focus is on packet filtering firewalls.

Lab environment

Network setup

To conduct this lab, you need four machines: a firewall, an inside host, an outside host, and an attacker. You will use an LXC container for each of these machines. Let the outside network be 10.0.10.0/24, and the inside network be 10.0.20.0/24.

Topology



Tools

wireshark - Sniffer and protocol analyzer
tcpdump - Command-line based sniffer
netwox - Tools to generate packets and spoof network traffic
netcat (nc) - Lots of different tools, can be used for a simple client/server.
iptables - tool to configure the tables provided by the Linux kernel firewall (Netfilter modules)
ufw - a firewall configuration tool for iptables (ufw = **u**ncomplicated **f**ire**w**all)

Tasks

Setting up the hosts

On the host VM, create a new LXC container for the attacker.

```
sudo lxc-clone -o lxc-template -n attacker -B overlayfs -s
```

Make sure you have an internet connection enabled on the host VM. Start the attacker:

```
sudo lxc-start –n attacker
```

Note: If the previous command does not work, use this instead:

```
sudo lxc-start --name attacker
```

At the login prompt, login to the attacker (same username and password as on the host VM), install the netwox tool, and then exit the container by shutting it down:

```
sudo apt-get update
sudo apt-get install netwox
sudo poweroff
```

On the host VM, change the attacker’s network link to the internal bridge.

edit `/var/lib/lxc/attacker/config` and change:

```
lxc.network.link = lxcbr0
```

to:

```
lxc.network.link = lxc-int-br
```

Start the LXC containers. Open one terminal window for each, and run these commands, one in each window:

```
sudo lxc-start -n attacker
sudo lxc-start -n inside-host
sudo lxc-start -n outside-host
sudo lxc-start -n firewall
```

Disable DHCP on the LXC containers. In each container, edit `/etc/network/interfaces` and change:

```
iface eth0 inet dhcp
```

to:

```
iface eth0 inet manual
```

Then restart the network:

```
sudo /etc/init.d/networking restart
```

Note: You might get a warning similar to "sudo: unable to resolve host attacker". You can ignore this warning or edit `/etc/hosts` and change:

```
127.0.0.1 localhost
```

to:

```
127.0.0.1 localhost, attacker
```

Similarly, you can edit all lxc containers.

Verify that you can run `ufw` on the firewall host. You might get an error message related to IPv6. The problem is that IPv6 requires a loadable kernel module, which is not loaded by default. There are two different ways to fix this.

1. On the firewall client, edit `/etc/default/ufw` and change `IPV6=yes` to IPV6=no.
2. Start `ufw` on the main VM. This will automatically load the missing kernel module. (Since LXC containers share the kernel with the main VM, this will also take care of the problem on the LXCs.) This is the preferred solution.

Task 0 – Configure Networking for LXC Containers

The first step is to set up networking on the containers.

On firewall:

```
sudo ip address add 10.0.10.1/24 dev eth0
sudo ip address add 10.0.20.1/24 dev eth1
```

On inside host:

```
sudo ip address add 10.0.20.2/24 dev eth0
sudo ip route add 10.0.10.0/24 via 10.0.20.1
```

On outside host:

```
sudo ip address add 10.0.10.2/24 dev eth0
sudo ip route add 10.0.20.0/24 via 10.0.10.1
```

On attacker:

```
sudo ip address add 10.0.20.3/24 dev eth0
sudo ip route add 10.0.10.0/24 via 10.0.20.1
```

Lastly, test connectivity by pinging each host from all others.

If you need to open an additional terminal for a container, use:

```
sudo lxc-attach -n <container_name>
```

Task 1 – Building a Firewall

In this task you will use `ufw to` build a simple firewall for your network.

1.1 Default Policy

It is good practice to take a conservative approach, and state that everything that is not explicitly allowed, is forbidden:

- Set the default policy for the `ufw` firewall to drop (deny) all packets.

1.2 Network Permissions

Now it is time to start allowing some carefully chosen traffic through the firewall.

- Create a rule that allows all traffic originating from the internal network arriving to the internal interface of the firewall host.
- Create a rule that allows all traffic originating from the firewall to reach the internal network.

Make sure you now can reach the internal network from the firewall, and the firewall from the internal network.

Question 1: What rules do you need in order to set up the default policy, and to allow communication with the firewall and the internal network?

Question 2: with `ufw` you can choose between “deny” and “reject” when you want to disable traffic. What is the difference between deny and reject? What are the advantages/disadvantages? Make an experiment where you set up the firewall to reject instead of deny, and explain what you see.

Report your `ufw` configuration for this part by using the command `ufw status verbose`.

1.3 Permitting a Service

SSH (Secure shell) is a common service for remote management of firewalls. Create rules which allow a host from the external network to connect to the firewall with SSH. SSH runs on port 22 and uses only TCP.
Make sure that the SSH daemon `sshd` is running on all hosts by executing `service ssh restart`.

Verify that

- you **can** connect to the firewall with `ssh` from the other hosts,
- you **cannot** connect directly from your external host to the internal host with `ssh`, and
- you **can** connect from your external host to the internal host if you first connect to the firewall with `ssh` and then connect from there with `ssh` to the internal host (note: think about which username you should use when logging in to the hosts with `ssh`).

Question 3: What are the advantages and disadvantages with opening up for `ssh` login from the outside?

Report your `ufw` configuration for this part by using the command `ufw status verbose`.

1.4 Stateful Filtering

In most cases we want to allow hosts on the internal network to connect to the external network, e. g., the Internet. However, we do not want hosts on the Internet to be able to connect to hosts on the internal network.

- Create a rule (or rules) to allow the inside host to establish connections to the outside. Keep blocking (dropping) all other connection attempts from the outside.

Verify that

- you **can** connect directly to the outside from the inside,
- you **cannot** connect directly from the outside to the inside.

Question 4: How can you verify that traffic from the outside going in is prevented, while traffic from the inside going out is allowed? Consider TCP, UDP as well as ICMP traffic. What tests do you perform to validate your configuration?

Report your `ufw` configuration for this part by using the command `ufw status verbose`.

1.5 Opening Ports

Sometimes, you want computers on the outside of the network to have access to a specific service on the inside. Your task now is therefore to add rules to your firewall, so that the outside host can reach a service at port 9000 on the inside host, for UDP as well as TCP. Use netcat (`nc`) for your service, both on the client and the server side.

Question 5: How can you verify that traffic to the service on the inside host at port 9000 is allowed, but no other services?

Report your `ufw` configuration for this part by using the command `ufw status verbose`.

1.6 Blocking Ports

Sometimes you do not want your internal users to be able to connect to the Internet on a specific port at all. One commonly blocked port is 135 which is used for Windows file sharing. Make sure your firewall blocks all traffic on port 135 (both TCP and UDP) from the computers on the internal network.

Note:
Actually there are more ports involved (135-139 on older Windows machines and 445 on more recent ones) but in this lab it is enough to block 135 as one example.

Question 6: How can you verify that the inside host cannot reach port 135 on the outside, but no other outgoing traffic is blocked?

Report your `ufw` configuration for this part by using the command `ufw status verbose`.

1.7 Verifying Your Setup

Verify your setup in a systematic manner, making sure all rules work correctly. To test a certain rule, you can listen to a port with `netcat` and try to connect to it using `netcat` or `telnet`:

On the receiving host run: `nc -l PORT`
On the sending host run: `telnet IPADDRESS PORT` or `nc IPADDRESS PORT`

After completing the tasks above, you should have the following rules active:

- Connections on **port 135** from the inside hosts are blocked.
- Connections on the **SSH port** are allowed to the firewall host from the outside.
- Connections on the **port 9000** are allowed to the internal hosts from the outside.
- Connections directly **to and from the firewall** are allowed from the internal networks.
- Connections **from the inside** are allowed out.
- Connections **from the outside** are blocked.

Task 2 – Defending Against SSH Brute-force Attacks

You have allowed connections to the SSH port (hence, its service) of the firewall from any host on the outside network. There is nothing that would avoid a brute-force attack or a Distributed Denial of Service (DDoS) attack because there are no limitations on the amount of times that one particular host can try a username and a password (besides the ones that the SSH server implementation might have internally).

Examine how you can use `ufw` to prevent excessive load on the firewall from SSH attacks. Verify that it works.

Question 7: How does `ufw` support protection against denial of service attacks? Explain how you configure your firewall.

Question 8: How can you verify that the protection works? Explain the experiments you perform.

Report your `ufw` configuration for this part by using the command `ufw status verbose.`

Task 3 – Ping and the Internet Control Message Protocol (ICMP)

The ping tool is usually used to test the reachability of a host that implements the Internet protocol (IP). It operates by sending packets using the Internet Control Message Protocol (ICMP) of the type ‘echo request’ to the host whose reachability is to be tested, and processing the response from that host (if any).

Ping can also be used in attacks, such as ping flooding, ping of death, and smurf attacks. Your overall task here is to design firewall rules that will allow inside-host to ping outside-host, but at the same time prevent an attacker from doing an attack where large amounts of ICMP echo requests or ICMP echo replies reach the inside-host.

There are no `ufw` commands to directly set up rules regarding ICMP echo requests and replies, so you may have to work with `iptables` to create such rule sets.

Blocking ICMP Echo Requests

`ping` the inside-host from the outside-host and describe what happens.
Set up a rule on your firewall to block all ICMP echo requests from the external network to your internal network. `ping` the inside-host again and describe what happens.

Rejecting ICMP Echo Requests

Replace the block rule above with a new rule which rejects all ICMP echo requests from the external network to your internal network.

Verify the following:

- That you can ping from the inside-host to the outside-host.
- That the outside-host cannot `ping` the inside-host.
- The firewall can `ping` both hosts.

Question 9: Can you `ping` from the outside-host to the inside interface of the firewall? Why/why not? Can this have any security implications?

Blocking ICMP Echo Replies

Let the attacker ping the outside-host while spoofing the source address by using the inside-host's address as source address. Use `netwox` if necessary.

Create a rule to prevent the ICMP Echo Replies from reaching the inside-host in the scenario above.

In your report, specify the rules you now have in your firewall. You can use the following commands to examine the rules:

- `sudo ufw status numbered`
- `sudo ufw status verbose`
- `sudo iptables --list --verbose`

Logging and Limits

One important task for a firewall is to log rejected or dropped packets, making it easier to trace attacks. A rule can be created with the jump target LOG to save information to the system log. `iptables` logs directly to the syslog system.

Create a rule that logs all rejected `ping` messages. Make sure the string "Ping rejected by <your name>:" is written in the log message. Check the system log so the traffic is saved. This can be done with:

- `tail -f /var/log/syslog`

The module `limit` can be used to limit how often a rule can be triggered. Use the `limit` module to make sure no more than 5 pings each minute are saved to the system log.

In your firewall there is now no possibility for the inside-host to ping the outside-host. Your final step in this task is now to configure the firewall so that you prevent the attacker from successfully creating large amounts of fake ICMP echo replies that reach your inside-host, while still making it possible to ping the outside-host from the inside-host.

The key here is to allow occasional ICMP replies from the outside to pass the firewall to the inside, but to make sure that ICMP replies arriving at a high rate cannot reach the inside. It is up to you to define high rate versus occasional ICMP replies.

Question 10: Explain how you can limit the amount of ICMP echo replies that reach the inside host. How can you verify that it works?

Your report must include a specification of the rule set created for this task. You should also provide packet traces (using e.g., wireshark) to demonstrate the behavior of your firewall.

Organization

This assignment is done individually.

Submission

Upload your solution as a single ZIP archive in moodle. The ZIP archive should contain the following:

- A detailed lab report that describes what you have done and what you have observed; you also need to provide explanation to the observations that are interesting or surprising.
- Specification of the rule sets you have created in the various tasks (`ufw status verbose`).
- A file called ”AUTHOR”, with full name and KTH email address of author.

Requirements and points

For this assignment, you can get a maximum of 100 points. The grading scale is as follows:

- 50 points: Task 0 and 1 correctly solved.

The remain two tasks (Task 2–3) require that you have solved Task 0 and 1, Otherwise they are independent, and you get points for them separately. The grading scale is as follows:

- Task 2: 20 points
- Task 3: 30 points

Note that there can be deduction in points depending on the quality of your report. If you aim for 100 points, for instance, but we assess that your solution only solves half of the challenges in a satisfactory manner, you might get 80 points.