澳門理工學院
Instituto Politécnico de Macau
Macao Polytechnic Institute

School of Applied Sciences (B.Sc. In Computing)

COMP 225: Network and System Administration Notes #9: SSH and Simple Firewall

K. L. Eddie Law, PhD
Associate Professor
Computing, MPI

Academic Year 2nd Semester, 2019-2020

On Security and Protection

- Network security is extremely important with today’s Internet
- For protecting communications, the popular remote login application program is Secure Shell (SSH), and SSH uses TCP as the underlying transport protocol
- For protecting incoming and outgoing traffic, basic firewall-like protection mechanisms are available in Linux
- One of them is the Netfilter, a packet handling engine, and its command line tool, the

```
$ sudo iptables …
```

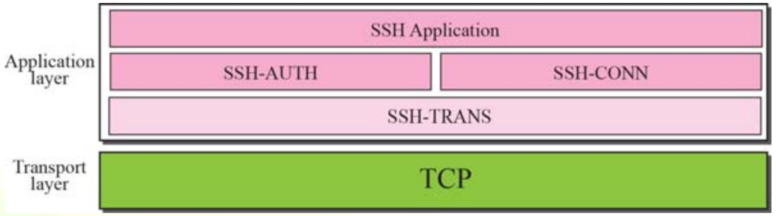
Communication Protection – SSH

- SSH allows logging in a remote computer (or a local) computer

```
$ ssh [-l username] [computer_name]
```

- In fact, can use SSH to log in localhost instead of using command `su`
- SSH is a secure replacement for the legacy text-based “`telnet`”
- SSH requires that an SSH daemon, the server, be running on the remote host. You will also need the password of the user you wish to log in as

Component of SSH



SSH Man in the Middle Warning

- The first time we SSH into a host, we likely see a message similar to the one below

```
The authenticity of host 'localhost (::1)' can't be established.
RSA key fingerprint is 20:d6:36:a1:e7:2f:98:97:58:f5:00:a8:85:3e:9d:58.
Are you sure you want to continue connecting (yes/no)?
```

- SSH uses public key cryptology to add security to the process
- This message is shown because this is the first time seeing this incoming host's public key
- Answering "yes" causes SSH to import this host's public key into the logging in user's `~/.ssh/known_hosts` file

Public Key Infrastructure for SSH

- SSH allows authentication using digital signing, a secure method of proving ones identity
- `$ ssh-keygen`
  - Creates public/private key pairs and stores them in a user's `.ssh` directory
  - On running the `ssh-keygen` command, always prompt for a passphrase
  - The passphrase is **NOT** a password to login to a server; it is a password that is used to encrypt your private key

Public Key Infrastructure for SSH (cont'd)

- `$ ssh-copy-id -i [identity_file] [remote_system]`
  - It copies the public key into the `authorized_keys` file on remote systems, enables you to login those system using public keys encryption rather than your system password
  - E.g., `$ ssh-copy-id -i ~/.ssh/id_rsa.pub testSSH@localhost`

The User's ~/.ssh Directory

- The .ssh directory holds important files for SSH operations
  - **`id_rsa`**: user's private key if rsa is used, keep this key secret!
  - **`id_rsa.pub`**: user's public key if rsa is used; copy this file to `authorized_keys` on machines like to log into in future
  - **`id_dsa`**: user's private key if dsa is used, keep this key secret!
  - **`id_dsa.pub`**: user's public key if dsa is used; copy this file to `authorized_keys` on machines like to log into in future
  - **`known_hosts`**: the hosts and host keys of computers that this user has used SSH to connect to
  - **`authorized_keys`**: grants user's access to log into this account with digital signature authentication; for each public key listed in this file, the associated private key can be used to login to this account

Personal Hygiene: Protection of Private Keys

- **IMPORTANT**: Do NOT allows anyone to access your private keys
- An attacker, gains your private key, can use it to log into other machines without a password, if your associated public key is in the `authorized_keys` file on any other machines
- Also possible for someone to log into your account on this machine if they can insert their own public key into your `authorized_keys` file
- Some administrators put their public keys in the `authorized_keys` file on remote servers. This allows them to use SSH to launch commands on remote computers without a password (via cron scripts etc.)

Secure File Transfers

- OpenSSH provides a number of ways to create encrypted remote logins and file transfer connections between clients and servers
- The OpenSSH Secure Copy (`scp`) and Secure FTP (`sftp`) programmes are the secure replacements for traditional text-based FTP

Installing SSH Daemon

- For Ubuntu server, SSH server usually is installed, if not, then run

```
$ sudo apt install openssh-server
```

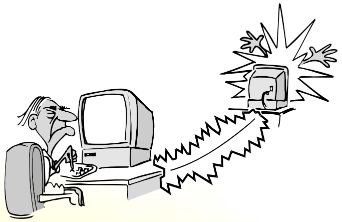
- Then start the SSH server daemon, and enable it next upon rebooting

```
$ sudo systemctl start sshd
$ sudo systemctl enable sshd
```

- The SSH server and client configuration files can be found at /etc/ssh/sshd_config and /etc/ssh/ssh_config, respectively
- Any changes made in these files, should “restart” the daemon again

```
$ sudo systemctl restart sshd
$ sudo systemctl status sshd
```

Simple Defense – Netfilter and iptables



Firewalls Built inside Linux Kernels

- Linux 2.0.x – ipfwadm
- Linux 2.2.x – ipchains
- Since Linux 2.4.x till today – Netfilter and iptables
- Any new systems coming in?
  - nftables using nft commands was in since 2014 for Linux 3.13, but failed so far
  - Yes, a newer one, the bpfilter from the BSD operating systems, but not yet ready in Linux

Netfilter/iptables?

- Netfilter and iptables are building blocks of a framework inside the Linux kernel
- This framework enables packet filtering, network address [and port] translation (NA[P]T) and other packet mangling

What is iptables?

- Stateful packet inspection
  - The firewall keeps track of each connection passing through it, an important feature in the support of VoIP
- Filtering packets based on a MAC interface, IPv4, IPv6
  - mportant in WLAN's and similar environments
- Filtering packets based the values of the flags in the TCP header
  - Helpful in preventing attacks using malformed packets and in restricting access
- Network address translation and Port translating NAT/NAPT
  - Building DMZ and more flexible NAT environment to increase security

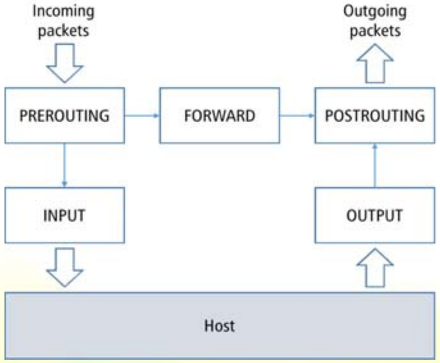
What is iptables? (cont'd)

- Source and stateful routing and failover functions
  - Route traffic more efficient and faster than regular IP routers
- System logging of network activities
  - Provides the option of adjusting the level of detail of the reporting
- A rate limiting feature
  - Helps to block some types of denial of service (DoS) attacks
- Packet manipulation (mangling) like altering the ToS/DSCP/ECN bits of the IP header
  - Mark and classify packets dependent on rules, the first step in QoS

Quality of Service (QoS) not covered or tested in this course

Designs of iptables

- iptables structures packet examinations through tables
- 3 widely used tables: `filter`, `nat`, `mangle` tables
- For all possible traffic flows, 5 chain designs are shown for packet examining or content editing, e.g., INPUT chain, etc.
- Each table has its own set of chains



Filter, NAT and Mangle Tables

- Table `filter`, for packet filtering, set firewall policy rules in chains
  - **Input** chain: filters packets destined for the firewall
  - **Forward** chain: filters transit packets to/from locations protected by firewall
  - **Output** chain: filters packets originating from the firewall
- Table `nat`, for network address translation, interested in 2 chains
  - Remember to set the system to permit IP forwarding for NAT to work (uncomment the `net.ipv4.ip_forwarding=1` in `/etc/sysctl.conf`)
  - **Pre-routing**: NAT packets when destination address need changes
  - **Post-routing**: NAT packets when source address need changes
- Table `mangle`
  - Manipulate QoS bits in TCP header through the input and output chains, if needed; usually not used by home users

Checking out the iptables

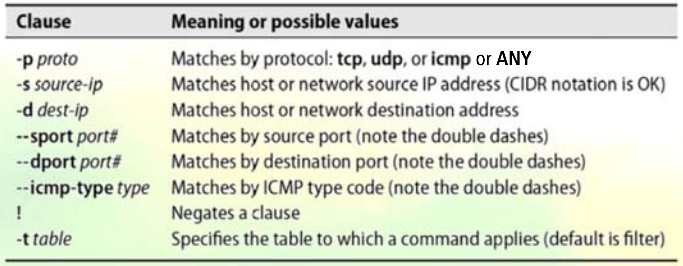
- Check if iptables is installed and running

```
$ sudo iptables –L -v
```

  - This lists all chains in all tables… could be many…
- FYI, there are many options for iptables, we are going to only discuss some features of iptables (already quite a lot!!)
- All following iptables rule setting, we use `$ sudo iptables` …
  - Fyi, the rules input in terminal will not be persistent upon rebooting
- But we can write scripts for iptables to run at machine boot-up

Switches/Options for iptables

- -F flush; deletes all the rules in the selected ***Table***
- -A [chain name] append to the end of the named chain
- -j [target] jump out to a targeted decision, see in two slides
- -P [chain name] default policy for a chain, if needed
- -D [chain] [rule #] delete a rule with the order number indicated



Clause	Meaning or possible values
-p *proto*	Matches by protocol: **tcp**, **udp**, or **icmp** or **ANY**
-s *source-ip*	Matches host or network source IP address (CIDR notation is OK)
-d *dest-ip*	Matches host or network destination address
--sport *port#*	Matches by source port (note the double dashes)
--dport *port#*	Matches by destination port (note the double dashes)
--icmp-type *type*	Matches by ICMP type code (note the double dashes)
!	Negates a clause
-t *table*	Specifies the table to which a command applies (default is filter)

Protocol Switches

- If “-p” is used, we can mark TCP, UDP, ICMP

Protocol Switch	Description
-p tcp --sport [source port #]	TCP with source port #, range of ports “`starting_port:ending_port`”
-p tcp --dport [destination port #]	TCP with destination port #, range of ports permitted
-p tcp --syn	New TCP connection request with SYN bit set; “`! --syn`” SYN bit not set
-p udp --sport [source port #]	UDP with source port #, range of ports permitted
-p udp --dport [destination port #]	UDP with destination port #, range of ports permitted
--icmp-type [type]	Most common types are echo-request or echo-reply, i.e., `ping` commands

Jump to a Target/Decision

- After matching all conditions in an iptables rule statement, then we can make a decision using the switch “-j”
- A decision target queue must be appended after the “-j” switch
- Commonly used targets are
  - `ACCEPT`
  - `DROP`
  - `REJECT`
  - `LOG`
  - `MASQUERADE`

The Targets (1)

- ACCEPT
  - Leaving iptables, the packet is passed to application or the OS for further processing
- DROP
  - Packet is dropped quietly without any further processing
- REJECT
  - Packet is dropped, but an ICMP message is returned to packet sender
  - ”`--reject-with [qualifier]`” can be added, where ”qualifier” is an ICMP message
- LOG
  - Packet information is sent to syslog daemon for logging, and packet is then checked by next iptables’ rule
  - ”`--log-prefix ’reason’`” can be added
  - If doing LOG and DROP, then two rules are needed, cannot be integrated into one rule

The Targets (2)

- MASQUERADE
  - The regular NAT (Network Address Translation), the source address is changed to the outgoing IP address of the firewall
  - Port can be changed explicitly through ”`{--to-ports [port]{-[port]}}`”, or automatically
- SNAT
  - Source NAT – the source address is modified
  - Source address is user-defined, ”`--to [IPaddress]{:[port]}`” or a range for selection ”`--to [IPaddress{-[IPaddress]}]{:[port]{-[port]}}`”
- DNAT
  - Destination NAT – the destination address is changed
  - ”`--to [IPaddress]`”

Using the Protocol Switch

- The "-p" permits us to match specific protocol
- E.g., eth0 is facing the Internet, eth1 is facing an internal machine
  - Accept incoming HTTP traffic

```
$ sudo iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
```

  - Accept all new TCP connection request from internal machine

```
$ sudo iptables -A INPUT -i eth1 -p tcp --syn -j ACCEPT
```

  - Accept an UDP datagram from source 10.0.0.1 coming in for destined port 53

```
$ sudo iptables -A INPUT -s 10.0.0.1 -p udp --dport 53 -j ACCEPT
```

  - TCP traffic from anywhere going to 192.168.1.1

```
$ sudo iptables -A INPUT -s 0/0 -i eth0 -d 192.168.1.1  -p tcp -
j ACCEPT
```

- Some popular TCP port numbers: SSH (22), HTTP (80), HTTPS (443)

Any IP addresses

More Sophisticated Setting, "-m" Switch

- Matching rule with "-m"
- TCP is stateful, for "`-p tcp -m state --state [States]`", we should supply the states of a connection that the iptables shall check
  - Permitted states in TCP: **NEW**, **ESTABLISHED**, **RELATED**, **INVALID**
- For rate control with "`-m limit`", which specifies the maximum average number of matches per second in the forms of /second, /minute, /hour, or /day. E.g., 3/s is an abbreviations for 3/second

```
$ sudo iptables -A INPUT -p icmp --icmp-type echo-request
-m limit --limit 1/s -j ACCEPT
```

  - Only accept one ping request message per second

Setup of NAT

- Make sure the "`ip_forward`" is set in the system
  - Check if content of file /proc/sys/net/ipv4/ip_forward is 1
  - If not, `$ sudo echo 1 > /proc/sys/net/ipv4/ip_forward`
- Since using "systemd" in the latest Debian/Ubuntu versions for starting up
  - May have to add a file in /etc/systemd/network for the effect of IPv4 forwarding
  - Read http://manpages.ubuntu.com/manpages/disco/man5/systemd.network.5.html

Setup of NAT (cont'd)

Setting up the NAT for all traffic leaving interface eth0

- Suppose `eth0` facing the Internet, `eth1` facing computer inside organization
- An example, the iptables commands

```
$ sudo iptables -t nat -A POSTROUTING -o eth0 -j
MASQUERADE
$ sudo iptables -A FORWARD -i eth0 -o eth1 –m state
--state RELATED,ESTABLISED -j ACCEPT
$ sudo iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

Permits all traffic from internal computing devices

Permits traffic coming in eth0 and going out at eth1 only if the connection was established or related
⇒ this implies the connection was started by the computer

More Examples

- Allow HTTP traffic for the Apache2 web server over port 80

```
$ sudo iptables -A INPUT -p tcp --dport 80 -i eth0 -j ACCEPT
```

- Allow FTP traffic for VSFTPD daemon over port 21 to service FTP requests

```
$ sudo iptables -A INPUT -p tcp --dport 21 -i eth0 -j ACCEPT
```

- Allow SSH traffic for Secure Shell connections over port 22 to service SSH requests

```
$ sudo iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
```

- After applying the rules for the incoming traffic accepted in the **INPUT** chain, then applying a final “catch-all” rule to block those failed to meet any previous rules:

```
$ sudo iptables -A INPUT -p tcp -i eth0 -j DROP
```

- **Catch-all rules** MUST be applied the LAST

Summary

- Some basic security-related commands or tools are introduced
  - SSH – secured communications
  - iptables – for setting simple defense firewall
- If using scripts for running iptables while starting up
  - In general, we should clean up all those tables before adding any rules! For example, the general starting commands are

```
iptables -F
iptables -P INPUT DROP
iptables -P FORWARD DROP
```

  - Actively running iptables script can be saved using the command “**`iptables-save`**”