How it works...

In the `intruder_detect.sh` script, we use the `auth.log` file as input. We can either provide a logfile as input to the script by using a command-line argument to the script or, by default, it reads the `/var/log/auth.log` file. We need to log details about login attempts for valid usernames only. When a login attempt for an invalid user occurs, a log similar to `Failed password for invalid user bob from 203.83.248.32 port 7016 ssh2` is logged to `auth.log`. Hence, we need to exclude all lines in the logfile having the word `invalid`. The `grep` command with the invert option (`-v`) is used to remove all logs corresponding to invalid users. The next step is to find out the list of users for which login attempts occurred and failed. The SSH will log lines similar to `sshd[21197]: Failed password for bob1 from 203.83.248.32 port 50035 ssh2` for a failed password. Hence, we should find all the lines with words `Failed password`.

Next, all the unique IP addresses are to be found out for extracting all the log lines corresponding to each IP address. The list of IP addresses is extracted by using a regular expression for the IP address and the `egrep` command. A `for` loop is used to iterate through the IP address, and the corresponding log lines are found using `grep` and are written to a temporary file. The sixth word from the last word in the log line is the username (for example, bob1). The `awk` command is used to extract the sixth word from the last word. `NF` returns the column number of the last word. Therefore, `NF-5` gives the column number of the sixth word from the last word. We use `sort` and `uniq` commands to produce a list of users without duplication.

Now, we should collect the failed login log lines containing the name of each user. A `for` loop is used for reading the lines corresponding to each user and the lines are written to a temporary file. The first 16 characters in each of the log lines is the timestamp. The `cut` command is used to extract the timestamp. Once we have all the timestamps for failed login attempts for a user, we should check the difference in time between the first attempt and the last attempt. The first log line corresponds to the first attempt and the last log line corresponds to the last attempt. We have used `head -1` to extract the first line and `tail -1` to extract the last line. Now, we have a timestamp for first (`tstart`) and last attempt (`tends`) in string format. Using the `date` command, we can convert the date in string representation to total seconds in Unix Epoch time (the *Getting, setting dates, and delays* recipe of *Chapter 1, Shell Something Out*, explains Epoch time).

The variable's start and end has the time in seconds corresponding to the start and end timestamps in the date string. Now, take the difference between them and check whether it exceeds two minutes (120 seconds). Thus, the particular user is termed as an intruder and the corresponding entry with details are to be produced as a log. IP addresses can be extracted from the log by using a regular expression for the IP address and the `egrep` command. The number of attempts is the number of log lines for the user. The number of lines can be found out by using the `wc` command. The hostname mapping can be extracted from the output of the host command by running with the IP address as the argument. The time range can be printed using the timestamp we extracted. Finally, the temporary files used in the script are removed.

The previous script is aimed only at illustrating a model for scanning the log and producing a report from it. It has tried to make the script smaller and simpler to leave out the complexity. Hence, it has few bugs. You can improve the script by using a better logic.

Remote disk usage health monitor

A network consists of several machines with different users and requires centralized monitoring of disk usage of remote machines. The system administrator of the network needs to log the disk usage of all the machines in the network every day. Each log line should contain details like the date, IP address of the machine, device, capacity of device, used space, free space, percentage usage, and health status. If the disk usage of any of the partitions in any remote machine exceeds 80 percent, the health status should be set as `ALERT`, else it should be set as `SAFE`. This recipe will illustrate how to write a monitoring script that can collect details from remote machines in a network.

Getting ready

We need to collect the disk usage statistics from each machine on the network, individually, and write a logfile in the central machine. A script that collects the details and writes the log can be scheduled to run every day at a particular time. SSH can be used to log in to remote systems to collect disk usage data.

How to do it...

First, we have to set up a common user account on all the remote machines in the network. It is for the disklog program to log in to the system. We should configure auto-login with SSH for that particular user (the *Password less auto-login with SSH* recipe of *Chapter 7*, *The Old-boy Network*, explains configuration of auto-login). We assume that there is a user test in all remote machines configured with auto-login. Let's go through the shell script:

```
#!/bin/bash
#Filename: disklog.sh
#Description: Monitor disk usage health for remote systems


logfile="diskusage.log"

if [[ -n $1 ]]
then
  logfile=$1
fi

if [ ! -e $logfile ]
```