COMP 8006 - Assignment #3

Mario Enriquez

British Columbia Institute of Technology

COMP 8005, COMP 6D

Aman Abdullah

2016-03-03

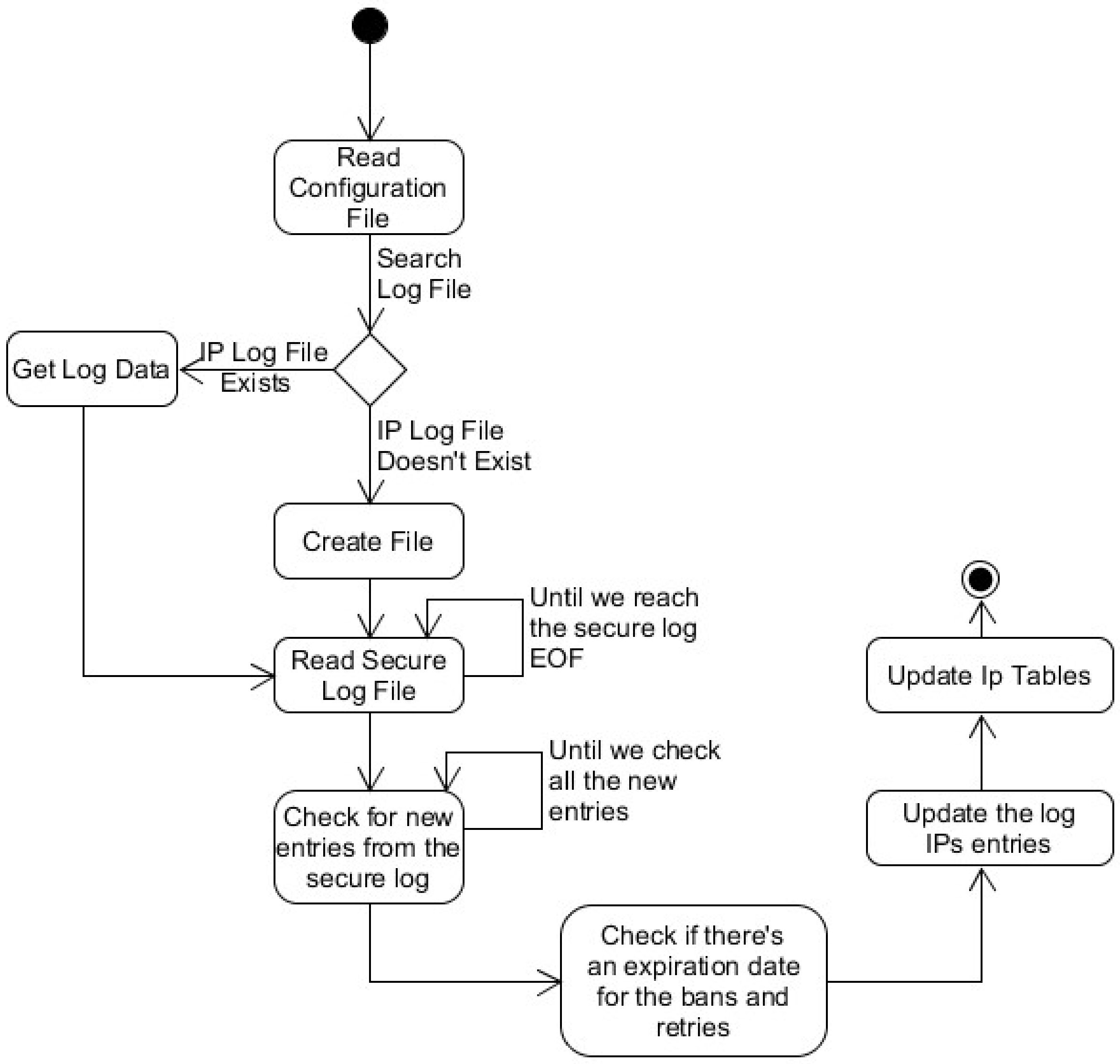
Contents

Introduction

In this assignment, I designed a simple monitor for the log file “secure” located in the /var/log folder. The program reads the file, checks for specific terms within the logs entries and checks if there are failed attempts to log in. After the user tries an specific amount of times he is blocked by using the netfilter command iptables.

Design Work

Configuration File


Read Configuration File
Search Log File
IP Log File Exists
Get Log Data
IP Log File Doesn't Exist
Create File
Until we reach the secure log EOF
Read Secure Log File
Until we check all the new entries
Check for new entries from the secure log
Check if there's an expiration date for the bans and retries
Update the log IPs entries
Update Ip Tables

How-to install

Copy the config file, the shellscript file and the 3 java classes file to an specific folder. It doesn't matter which folder but the route is necessary.

Modify the contents of the config file:
IMPORTANT: The format must be "Variable space = space value"

Verify the path of the secure log
SEC_PATH = /var/log/secure

Verify the path of where the log for blocking IPs is going to be
LOG_PATH = /root/Documents/C8006_Assignment_3/Log

Set the maximum number of attempts
MAX_ATT = 3

Set the maximum number of failed connections permitted (not implemented)
MULT_ATT = 10

Set the time to lift the ban
BAN_TIME = 480000

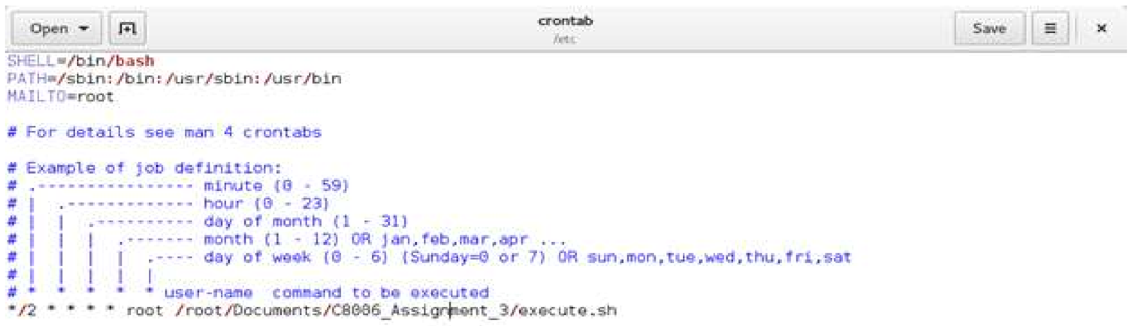
Set the time to remove the retries
COOLDOWN = 240000

Set the terms to search when reading the secure log
PROTOCOL = sshd
SEARCHTERM = password

Set the port to block
PORT_ = 22

Set the path to the java class
JAVACLASS_PATH = /root/Documents/C8006_Assignment_3/
JAVAFILE_NAME = logfile

After finishing the configuration, add the path to the shellscript to the crontab.



```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
*/2 * * * * root /root/Documents/C8006_Assignment_3/execute.sh
```

Figure 1Crontab File Configuration

Then use in the terminal:

```
systemctl stop crond
systemctl start crond
```

Test Cases

Test	Description	Tool	Expected Results	Actual Results
1	Test that the task set in crontab works	Terminal	We see a log indicating that cron executed the task	Success. Cron executed the task every 2 minutes.
2	Test when a client fails to connect to the server less than the maximum retries	Client Computer 192.168.0.23/ Server Log	The client is not blocked, but is registered in the log	Pass, The server has no problem handling this load
3	Test when a client exceeds the maximum number of tries	Client Computer 192.168.0.21/ Server Log	The server has no problem handling this load	Pass, The server has no problem handling this load
3.1	Check IP tables results	Terminal	Ip tables added the Ip	Pass, the Ip was added
4	Reset the number of tries after a set amount of time	Server log	See the tries column drop to 0 after a 240000 miliseconds (4 min)	Pass, the column was updated and the user could retry again
5	Reset the ban after a certain amount of time	Client Computer 192.168.0.21/ Server log	Ip removed from iptables and ban set to no and retries to 0.	Pass, user can retry again and registry is removed

6	Test that the logs are updated properly	Server log	Check that the attempts are updated correctly after being resetted	Pass, both logs have the correct amount set in the logfile
7	Make multiple attempts to log in, more than the permitted	Client	User should not be able to make several attempts	Failure, difficult to correct due to nature of cron

For the results, see Annex.

Observations:

I don't feel that cron is really well suited for the task of executing the task of monitoring a file that handles a lot of password requests as the minimum time allowed is once per minute. I could very well exceed the number of password attempts for ssh in a minute if the maximum number of attempted logins is low. Obviously augmenting is a potential security risk and a program that would read the file every time a change is made would be suited better for the task.
Cron works okay in these circumstances when the number of computers attempting to guess the password is small.

Pseudo code:

```
logfile.java


import libraries

class logfile{

  create object LogData {

    create object variables ipAddr, int tries, isBanned and lastTry

    constructor LogData(variables ipAddr, int tries, isBanned and lastTry){

      set the object variables

    }

  }

  create object NewData {
```

```
  create object variables ipAddr, status and date



  constructor NewData(variables ipAddr, status and date){

    set the object variables

  }

}

Method getMonth(Month as text){

  variable monthNo;

  switch(month){

    Assign to monthNo 01,02,03….12 depending on the letters of Month

  }

  return monthNo;

}

Main function (receives args){

  try{

    Declare string variables i,j,index,operation_time,attempts,multi_attempts, secure_path,
log_path,config_path,logDate,lastCheck,aux,month,split,splitaux,banTime,cooldownTime;

    Declare long variables ban,cooldown,k;

    Declare ArrayLists search_terms, log_entries, sec_entries

    Declare DateFormat dateFormat

    Declare Date variables date,curDate;

    Declare File variables f,sec,config;

    Declare FileReader variables fr;
Declare FileWriter variables fw;
Declare BufferedReader variables br;
Declare BufferedWriter variables bw;
Get the path of the configuration file
get current date in curDate
get config file in config
if config file exists{
  create new FileReader with config file
  create new BufferedReader with filereader
  set variables according to their type and name
  close BufferedReader
  close FileReader
}
Get log file in f
If log exists{
  create new FileReader with log file
  create new BufferedReader with filereader
 get log entries
  close BufferedReader
  close FileReader
} if it doesn’t exist {
  Create the file
}
```

```
Get secure file in sec
create new FileReader with log file
create new BufferedReader with filereader
while reading the secure log{
  if any of the keywords is found {
    add to the New Data Object
}
For all the entries in the secure file{
  Set index as not found
  For all the entries in the log file{
    If an entry of the secure file is found, get the index
  }
  If the secure log marks the attempt as failed{
    If entry exists in the log{
      If entry is new{
        Set the latest entry date
        Add to the number of tries
      }
    } else {
      Add new entry in the log with one entry attempt
    }
  } else {
    If entry exists in the log{
      If entry is new{
        Set the latest entry date
        Reset Counter
      }
    }
  }
}
close BufferedReader
close FileReader
Create FileWriter
Create BufferedWriter
For  the entries in the log file object{
 If unbanning is allowed and is banned{
    Add unban time to the last attempt
    If current date is bigger than the last attempt {
      Reset ban and user tries
    }
  }
  If cooldown is allowed and is not banned{
    Add cooldown time to the last attempt
    If current date is bigger than the last attempt {
      Reset user tries
    }
```

```
        }
        If the number of tries exceed user value{
          Ban the user
        } if not {
          Don't ban the user
        }
        Write entry into the log
      }
      Close Buffered Writer
      Close File Writer
  } catch Exception {
    Print exception
  }
  }
}
```