Real Time Activity Logger: a User Activity Detection System

Hitesh Kumar Sharma, Ishu Khanchi, Nihal Agarwal, Pawas Seth, Prashant Ahlawat

Abstract-With the size of the enterprise network is growing, more and more computer terminals, internal information leakage and security threats emerging security incidents and daily management to bring a lot of pressure. Business users need a tracking system to address these security issues.A technique and framework for checking and following the exercises of a client of a PC. Affiliations are set up between the conditions of certain PC framework parameters and explicit exercises. The present movement of the client is then definite by the framework dependent on the present condition of the PC framework. As the condition of the PC framework changes, changes in the client's action are checked and followed. The exercises are followed related to time in order to record the time spent on every movement.Through this project we are developing the understanding of the working and the basics of the Linux operating system. We are analysing the need of monitoring the system and solving the issues related to security, unauthorised access and other threats. Through this tool we are trying to address these issues.

***Key Words*- Tracing, Monitoring, Log files, Database, Server**

I. INTRODUCTION

These days security is to be given highest priority and that too without compromising user's privacy.But still, this is a matter of concern for a lot of users and administrators. Many software's/installers came into market but each one has its own issues and disadvantages. Many suspicious user activities can compromise security across the network and will lead to data breaches, so finding the root cause of the breach or any other problem regarding file system. Manual Analysis increases the time to solve the threats. Keeping this in mind the ultimate need of a tracer to track down the activities of any third party given access by the user/administrator, this project is a utility tool. This utility tool is made to monitor and trace the activities done on the user's system by any third party and keeping the logs of those activities in the log files. We have used the concept of different Linux commands, and have implemented their functionality in C programming language. We have integrated all the functionality to make the application user friendly and user and administrator can use for different purposes like knowing the uptime of the system or for security concerns and many more. This is also used for monitoring the activities of the user.

Revised Manuscript Received on October 15, 2019

Hitesh Kumar Sharma, University of Petroleum & Energy Studies, Dehardun, Uttarakhand, India
Ishu Khanchi, University of Petroleum & Energy Studies, Dehardun, Uttarakhand, India
Nihal Agarwal, University of Petroleum & Energy Studies, Dehardun, Uttarakhand, India
Pawas Seth, University of Petroleum & Energy Studies, Dehardun, Uttarakhand, India
Prashant Ahlawat, University of Petroleum & Energy Studies, Dehardun, Uttarakhand, India

Associations are established between the states of certain computer system parameters and specific activities.
Strategies and frameworks are accommodated catching utilization information from a client PC, preparing a subset of such information to shape yield, and offering access to perspectives of such yield, for example, to help an organization's administration in checking PC use in a workplace. A system and method for monitoring computer usage is disclosed. A computer operator specifies discrete moments of a computer's usage at which screen captures are executed and saved to a log. By constantly recording the logs, users will be able to determine/predict strategize security failovers and appropriately accommodate the precautionary measures. For e.g.: If any third person given access to the system, removes any files or make sudden changes without user's permission, user can track that from the records stored in log files and can take actions against him.Data recorded can be stored in files and uploaded on the server so as to save system's memory and also provide the administrator the access to its system from any place.

II. PROBLEM STATEMENT

The question why this tool is needed can be answered in terms of security.Whenever any third party will have access to administartor's system,he can keep a check on it so as to assure every possible security.This tool will be very beneficial as it traces user's activity,monitors user's usage ,maintains log files ,uses databases and stores the files on server so that administrator can access them from anywhere and this also leads to less memory usage as data will be stored on the server. Data breach can be avoided.Non-centralized and manual log analysis increases the time it takes to remediate threats. Finding the root cause of an incident, and tracing it to a suspected user, prevent downtime of the businesses.

III. LITERATURE REVIEW

Activity fetcher like any other activity tracer, is used for monitoring the activities of the user working on Ubuntu(LINUX OS). Created with a main purpose of security and data privacy,this tool will be useful for the users working with various businesses.
The main objectives are:

- Tracing down users' activities on the system.
- Monitoring activities done by the user on the network.
- Maintaining and accessing the log files that store the session time.

Similarly several other activities like web based monitoring, screen monitoring,kernel monitoring etc. can be included in this tool.This tool saves memory as data is stored in databases on the server. Data can be easily accessed by the administrator and can be monitored from wherever on the server.

Focusing on various objectives, this tool is necessary from various point of views. Many companies will be benefitted from this tool.

IV. IMPLEMENTATION

After legitimate usage of the accessible resources, we have dissected that what are the conceivable routes through which we can discover the information of our system. On account of which we have built up our very own approach to keep up every one of the records of the opened or running application and the introduced applications inside the system by utilizing Linux commands and implanting them using C.
A few commands are recorded underneath through which we can get extra data about the system:-

- uptime
- ctime
- ls

Etc.
We further continue where we had made connectivity of our code with MYSQL database by utilizing PHP. From that point onward, we are getting the information in regards to introduced and running applications and store the particular information inside the database as a table for quick recovery at the season of following the movement of the system. This code will keep running at the back-end after an explicit measure of time. The present time we have brought from the system itself and showed onto the system.

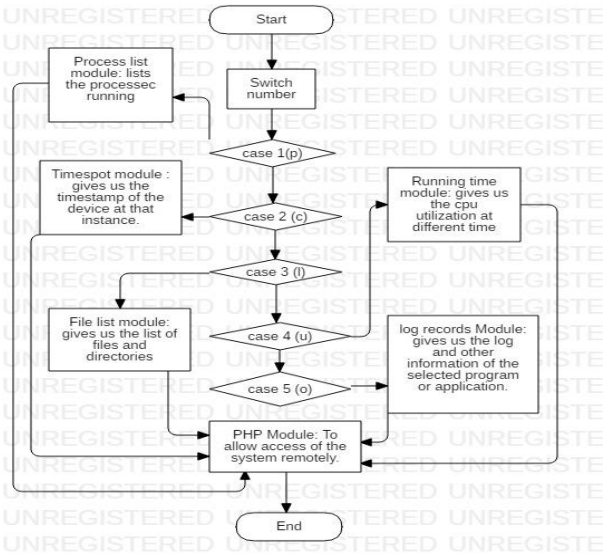

Fig.1: (DATA FLOW DIAGRAM)

```
if(decide == 's')
{
    printf("Give the complete Directory you want to save the process list log file in \n");
    scanf("%s", c);

    fp = fopen(c , "r");
    if(fp == NULL)
    {
    *comm = *(char*)'p';
    strcat(comm,(char*)'s aux > ');
    strcat(comm,c);
    system(comm);
    printf("File is created\n");
    }
    else
        printf("The file already exists \n");
}
```

```
else if(decide == 'd')
{
    printf("Choose to display for a specific  active process(p) or the whole file(f) of active processes(p/f)");
    scanf("%c", &decide);

    if(decide == 'f')
        {
        *comm = *(char*)'p';
        strcat(comm,(char*)'s aux');
        system(comm);
        }
    else if(decide == 'p')
        {
            printf("Enter the process name \n");
            scanf("%s" , proc);
            *comm = *(char*)'p'; //two processes come instead of one(redundancy)
            strcat(comm,(char*)'s aux | grep ");
            strcat(comm,proc);
            system(comm);
        }
}

}
```

```
else if(progchoice == 'c') { // 2nd choice

    char* my_strftime(const char *format, struct tm *tmp) {
        static char buf[BUF_SIZE];
        size_t s;

        if(tmp == NULL)
        return NULL;

        s = strftime(buf, BUF_SIZE, (format != NULL) ? format: "%a %b %z %l:%M:%S %p %Z %Y", tmp);
        /*%x = Preferred date representation
        %I = Hour as a decimal number (12-hour clock). %M = Minutes in decimal ranging from 00 to 59.
        %p = Either "AM" or "PM" according to the given time value, etc. %a = Abbreviated weekday name
        %A = Full weekday name %b = Abbreviated month name
        %B = Full month name %c%d = Date and time representation
        %d = Day of the month (01-31) %H = Hour in 24h format (00-23)
        %I = Hour in 12h format (01-12) %j = Day of the year (001-366)
        %m = Month as a decimal number (01-12) %M = Minute (00-59)*/
        return (s == 0) ? NULL : buf;
    }

    void err_exit(const char *format, ...) {
        va_list args;
        va_start(args, format);
        vfprintf(stderr, format, args);
        va_end(args);

        exit(EXIT_FAILURE);
    }

        /*if(setlocale(LC_ALL, "") == NULL) {
            err_exit("Error in setlocale %s\n", strerror(errno));
            }

        real_time = time(NULL);
    if((tmp = localtime(&real_time)) == NULL) {
        err_exit("Error in localtime, %s\n", strerror(errno));
        }*/
    real_time = time(NULL);
    tmp = localtime(&real_time);

    printf("ctime() is %s\nasctime() is %s\n", ctime(&real_time), asctime(tmp));
    printf("%s\n", my_strftime(NULL, tmp));
    exit(EXIT_SUCCESS);
    }
```

```
else if(progchoice == 'o') { // 5th choice

    FILE *infile;
    infile = fopen ("/var/log/apport.log", "r"); // Open log file for reading
    fseek(infile,0L,SEEK_END); // Find the file size
    int sz=ftell(infile);
    char* buffer =(char *) malloc(sz* sizeof(char)); // dynamically allocates storage in a buffer
    rewind(infile);
    if (infile == NULL)
    {
        fprintf(stderr, "%s\n\error opening file\n");
        exit (1);
    }
    fread(buffer,sz , 1, infile); // read file contents till end of file
    for(i=0;i<sz;i++)
    {
        printf ("%c",buffer[i]);
    }

    // close file
    fclose (infile);
}

return 0;
}
```

Fig. 2 Java Code Screenshot

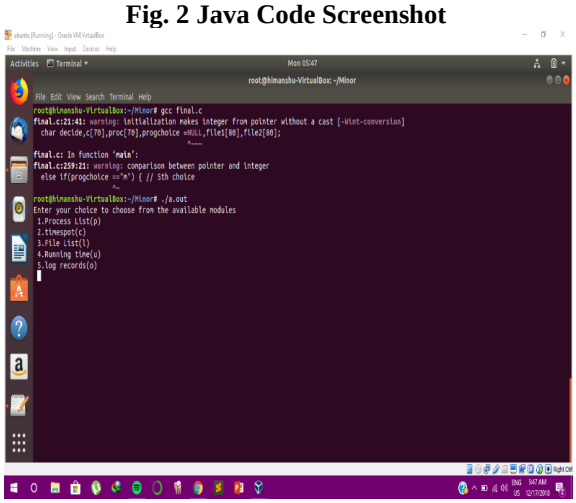

Fig. 3: User Interface with the various options for the user to select.

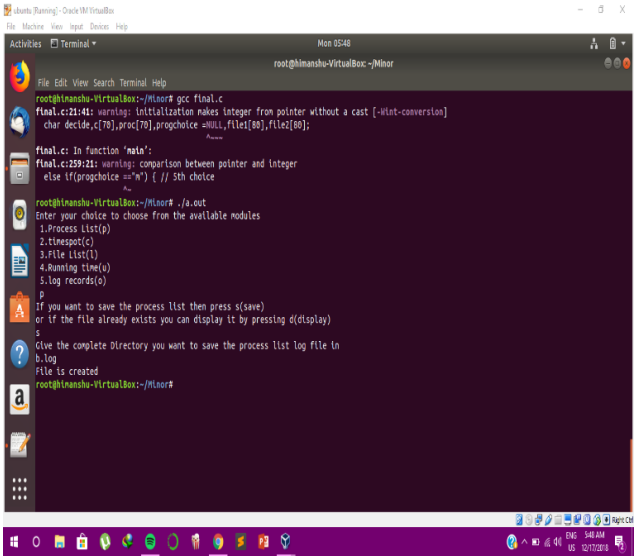

Fig. 4: Showing the list of the processes that are currently running.

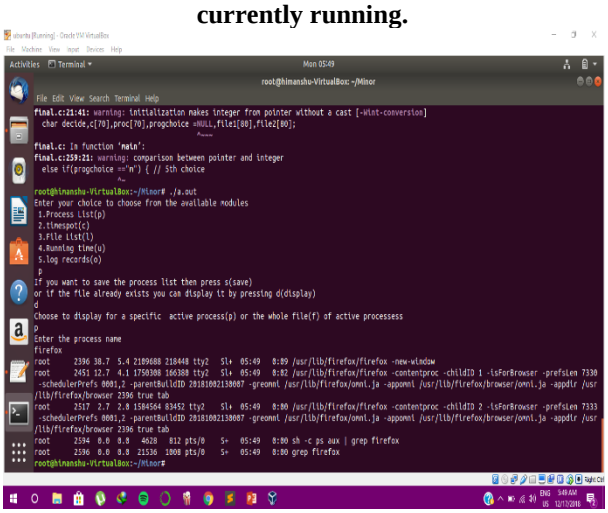

Fig. 5: Further detailed execution and output of the process list command.

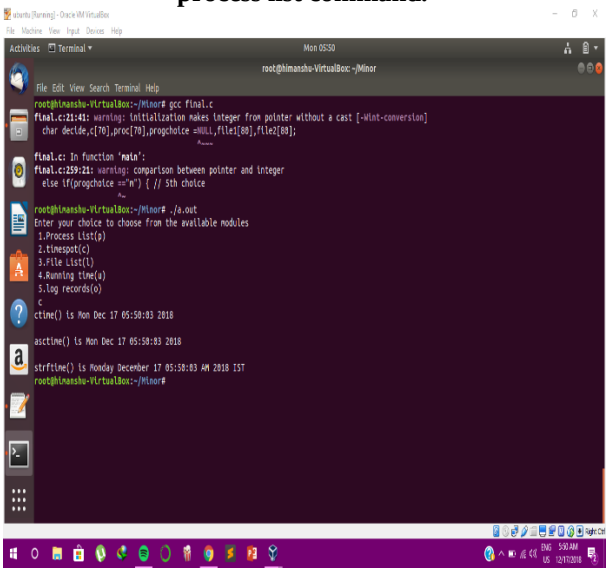

Fig. 6: User selecting the timespot command and getting the output.

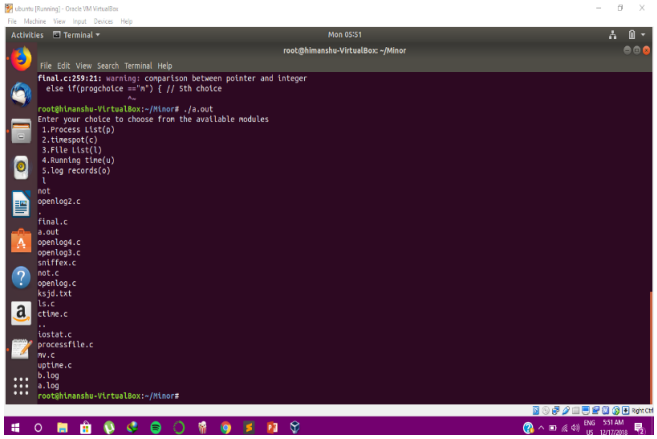

Fig. 7: User selecting the File list command and getting all the files in the directory as output.

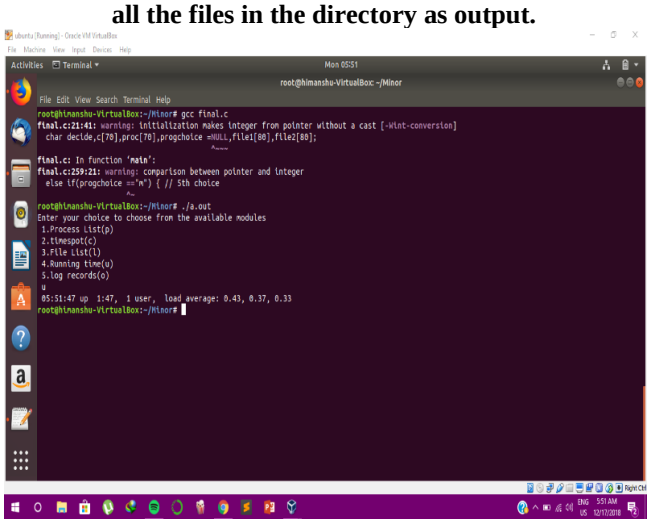

Fig. 8: Selection of the Running time command and getting the CPU utilization as output.

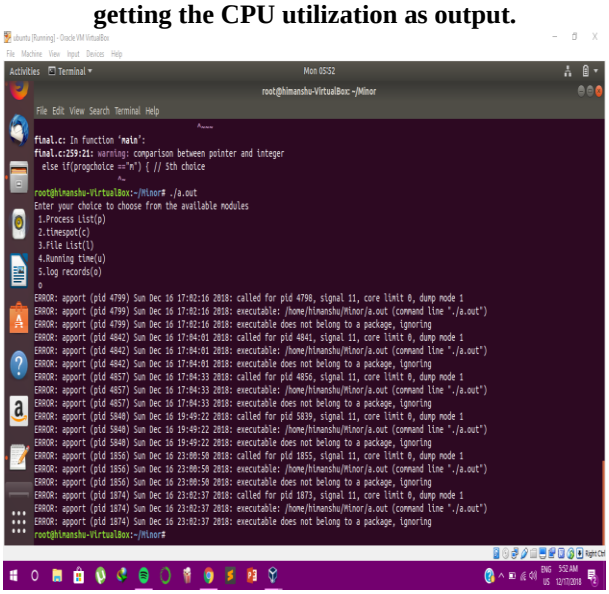

Fig. 9: Selecting the log records option and displaying the contents of the accessed log record.

Finally we have made a PHP page which works like a web portal through which we can access the database as per our motive of using the application.The databases can be uploaded on the server. Through this administrator can access databases from anywhere remotely so as to keep track of the user's activities on the system.As shown below:

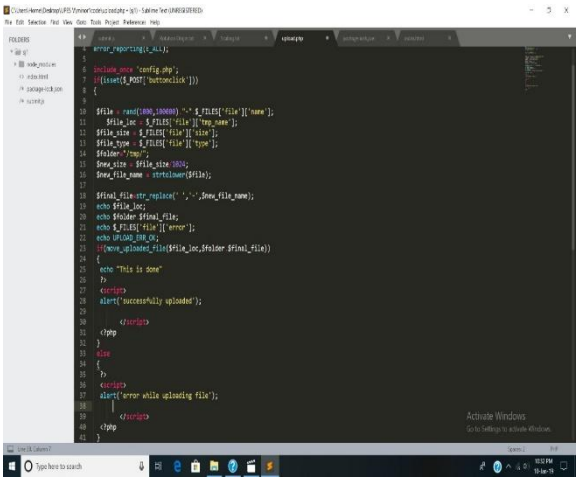

Fig. 10: PHP Code Screenshot

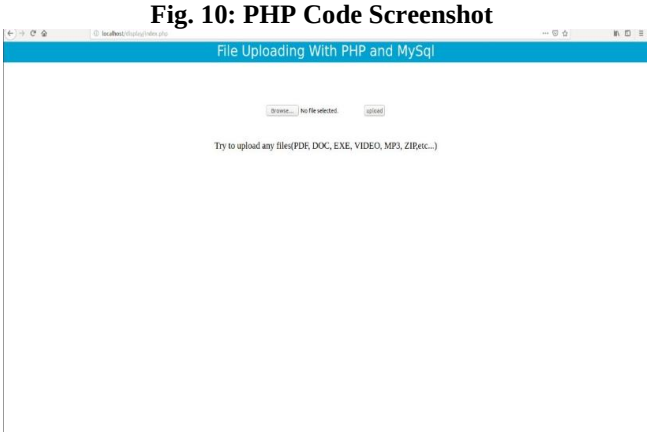

Fig.11: UI

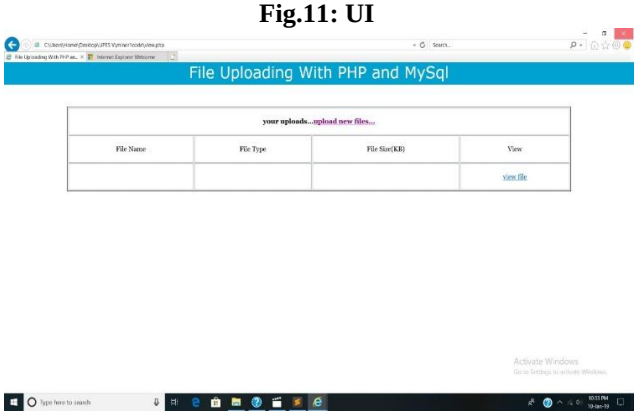

Fig. 12: UI

Fig. 13: UI

V. CONCLUSION

Our project is a model which address this issue in a comprehensive way. It would prove very beneficial in the real world as the project can work in dynamic conditions. It provides a very optimized solution to this problem very focused on the goal of monitoring the system and user activities and accessing data remotely from the server.

There are various techniques came to our knowledge regarding the monitoring of the system. We can use our approach in integration with various others approaches discussed. It will finally come to the users need. And according to the users need our developed approaches can be modified.

REFERENCES

1. HK Sharma, A Shastri, R Biswas, “A Framework for Automated Database TuningUsing Dynamic SGA Parameters and Basic Operating System Utilities”, Database System Journal, Dec 2013.
2. HK Sharma, A Shastri, R Biswas, “Auto-selection and management of dynamic SGA parameters in RDBMS”, Database System Journal, Dec 2015.
3. Ming Huang, ShujieGuo, Xu Liang, Xudong Song. Research on Real-time Performance Testing Methods for Robot Operating System, “JOURNAL OF SOFTWARE, VOL. 9, NO. 10, OCTOBER 2014”.
4. N. Djuric, J. Zhou, R. Morris, M. Grbvoic, V. Radosavljevic, and N. Bhamidipati. Hate Speech Detection with Comment Embeddings. In WWW, 2015.
5. I. Kayes, N. Kourtellis, D. Quercia, A. Iamnitchi, and F. Bonchi. The Social World of Content Abuser in Community Question Answering. In WWW, 2015.
6. C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang. Abusive Language Detection in Online User Content. In WWW, 2016.
7. K. Dinakar, R. Reichart, and H. Liebrman. Modelling the Detection of Textual Cyberbullying. The Social Mobile Web, 11, 2011.
8. C. Van Hee, E. Lefever, B. Verhoeven, J. Mennes, B. Desmet, G. De Pauw, W. Daelemans, and V. Hoste. Automatic Detection and Prevention of Cyberbullying. In Human and Social Analytics, 2015.
9. V. Nahar, S. Unakard, X. Li, and C. Pang. Sentiment Analysis for Effective Detection of Cyberbullying. In APWeb, 2012.
10. J. M. Xu, X. Zhu, and A. Bellmore. Fast Learning for Sentiment Analysis on Bullying. In WISDOM, 2012.
11. Wang, Xin. Predicting Polarities of Tweets by Composing Word Embeddings with Long Short-Term Memory. ACL, 2015.
12. Huang, Minlie, Y. Cao, and C. Dong. Modelling Rich Contexts for Sentiment Classification with LSTM. arXiv preprint arXiv: 1605.01478, 2016.