Securing the Virtual Machines in a Cloud Environment

Keerthi Bala Sundram
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Dallas, Texas 75080
Email: kxb152730@utdallas.edu

Aastha Dixit
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Dallas, Texas 75080
Email: axd155030@utdallas.edu

I. Introduction

The project aims to address one of the security issues that is faced in today's Cloud Computing environment. While researching on several issues for Cloud Computing, we came across four major areas of security concentration that led to this project. These are the four concerns that motivated us to develop a proof of concept for addressing these security issues.

A. *Who maintains the data ownership and control ownership?*

When a customer approaches the CSP for a Cloud Service, an argument is raised as to who would be the actual owner of the data and it's control. There is no proper mechanism to do so. If if the CSP can come to an agreement on the privacy of the data through a Service Level Agreement(SLA), many large scale and medium scale enterprises might not be willing to go for such a solution. Hence there arises the first source of security breach.

B. *Who maintains the audit records of the data?*

There are several logs and records being generated for each and every transaction in Cloud. These logs are the primary source of information about any security breach that could happen in the cloud. But the real question is who maintains this records of data, and who is the auditor of these records. There is no problem if only the customer has full access to these logs and their ownership. On the other hand, if the CSP has the full ownership and access, then what is the guarantee that the CSP might not handle it in an unethical way. This is the next security motivation for this project.

C. *What is the mechanism in the delivery of this audit record to the customer?*

This issue is analogous to the previous security concern. Let us assume that the CSP maintains the full authorization of the audit records, and as per the SLA, the customer can view the audit records at any time they want to. This may sound convincing but the real question is about the mechanism the CSP would follow to deliver this record to the customer.

D. *As a real owner of the data, does the CSP allow customers to secure and manage access from end-users (Customer's client)?*

The next issue is pertaining to the clients of the customers directly. Though the CSP's are the undisputed owners of the data, we are not sure if they allow the customers via an interface to let their end users to access the audit system to manage and secure their information in the cloud. This security issue would percolate from Customer to their end users, in turn to the customers of the end users and so on. This issues needs to be addressed to.

All these issues boil down to what we call as an "Insider Attack". One of the most popular security threats for many decades by now, insider attacks are gaining more vigor due to the new characteristics of cloud computing. When an insider attack is being performed at cloud service providers premises, the volume and velocity of its effect is multiple times than a traditional environment. This project would demonstrate this by detecting such attack.

In this project, we propose a method that can be amplified and used in the next generation Intrusion Detection systems in a Cloud environment. The method keeps monitoring some of the critical logs and then points out in case of a breach. The system employs a supervised machine learning technique using the LibSVM package. We have used our own method to extract the features of the logs and to employ these features in the validation.

II. Approach and Implementation

The complete experimentation was performed on the setup which had Ubuntu 14.04 64-bit LTS edition as native operating system. After the successful deployment of Cloud (IaaS) infrastructure using AWS, few VMs with varying requirements were created. One of the Cloud VM was installed with Ubuntu 14.04 32-bit LTS edition (henceforth referred as Instance-M). Instance-M was later configured with a model Java application deployed with few MySQL database tables. This cloud VM Instance-M has been used for all the further experimental purpose of this study.

A. Core Process

The most fundamental part of any computing system is its instant monitoring process in the form of raw reports, which are apparently known as system logs. System logs provides major source of information on status of systems, products, or even whys and wherefores of problems that occurred, and much more. Log files are produced in every field of computing; the features of these log files, particularly structure, context and language are different for different systems. Considering log files as the chief source for activities in the system/instance, the system's basic target was set to monitor the log files. The logical code written studies the log files at regular intervals. Principally, in Ubuntu Operating System, Zeitgeist is a service which logs the user's activities and events (files opened, websites visited, conversations hold with other people, etc.), and makes the relevant information available to other applications. It is the default logger of the Ubuntu Operating System. Using the Zeitgeist service the log files were tracked down and were logically used for the implementation. The following are the list of system generated log files that were parsed for this investigation:

- syslog - /var/log/syslog
- daemon.log - /var/log/daemon.log
- auth.log - /var/log/auth.log
- recently-used.xbel - /.local/share/recently- used.xbel.

As this experimentation was targeted to be a prototype proof-of-concept (POC), only the above four logs were subjected for implementation on Instance-M. The syslog keeps account of the system/kernel level activities and records them. Similarly, daemon.log defines the services such as USB Devices, Bluetooth Devices, MySQL Database, etc., running on user machine which is explicitly switched on or off by the user. When the user is authenticated in to Instance-M, such as logging in for first time or logging in after the screen is locked, that activity gets recorded in auth.log file. From this the authentication of the logged user can be identified. The recently-used.xbel keeps track of all the recently accessed/modified/viewed files and also maintains a record of all the activities pertaining to it. The log files that were considered are differed in format, structure and in data present inside it depending upon the configuration made at the system level respective kernel files. Rather than modifying the kernel file configurations (in each OS/platform), a uniform way to extract only the required fields were identified. This paved the way for the Audit Log Preprocessor module that basically preprocesses the log files for further processing. Utmost attention was given while choosing the required fields for extraction from these log files as the processed output would be surrendered to the SVM. For extraction of fields from the log files, a concept called as Feature Extraction which is the backbone of Artificial Intelligence practice was preferred. Feature extraction simplifies the amount of resources needed to represent a large set of data accurately. It was very important for the experimentation, as it gives a reduced set of features, which serves as an input to unsupervised technique. It is recognized that for each and every activity taking place in Instance-M, a log is generated, updated and maintained. The log file consists of many fields, which may or may not be relevant to identify an anomalous behavior while designing a security system. Fields such as type of access, time of access, user name, system name, message etc. are most crucial features which were chosen as candidates for extraction, and also which would help in successful identification of intrusion occurred in Instance-M.

The crucial features in the system logs were slightly different from the standard textual data. Although they are in natural language, they do not comply with grammar and differ in structure and context. And also since they are collected from various sources they are obvious to be heterogeneous in nature. Therefore, a technique which considers special vocabulary, heterogeneous and multi-source structures of log files was required for the implementation. Also the decisive challenge was to extract features from these logs and evaluate them for a patterned result.

Another challenge in machine learning theory is that the unsupervised algorithm takes numerical data for its functioning. Hence, for the design of the system, popular techniques such as Natural Language Processing (NLP) and Information Extraction (IE) theories were examined and avoided. The study was narrowed to find a suitable algorithm which overcame the following major issues.

- Parsing logs taken from different files.
- Conversion of extracted features to numerical vectors.

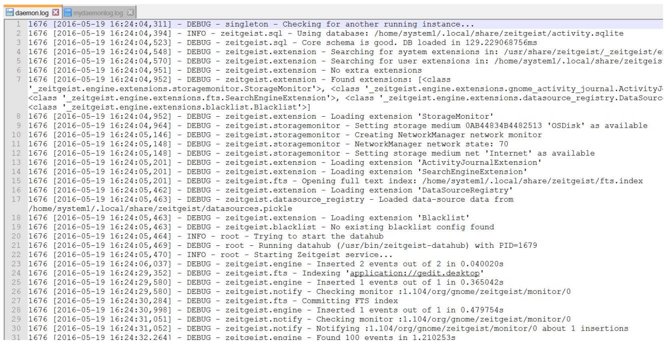
Basically, parsing is the process of analyzing a text, made of a sequence of tokens that are meaningful and can be processed. The Subsystem handles logs of different formats from various sources and unifies them to SVM. Further, processed these logs at logically defined intervals, and then the required data was generated in pre-defined structure. The extraction process for each log is detailed further.

B. Daemon Log Processing

The daemon log basically gives the relevant information when services such as USB Device, Bluetooth Device, MySQL Database, etc., was detected by the system. Important fields such as the file name extracted, application name accessed were analyzed via this log. Like syslog, the date format conversion is maintained in daemon log as well.

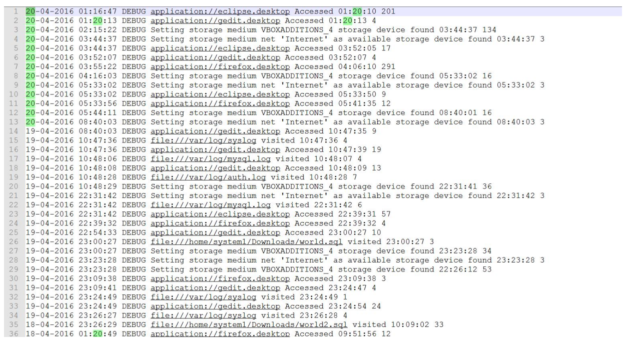
TABLE I
PROCESSING OF DAEMON.LOG FILE

Program	Daemon Log
Input	daemon.log
Process	Extracting fields. Calculating frequency. Converting Date.
Output	mydaemonlog.log
Note	Separate log files are created for each date and merged



```
1676 [2016-05-19 16:24:04,311] - DEBUG - singleton - Checking for another running instance...
1676 [2016-05-19 16:24:04,394] - INFO - zeitgeist.sql - Using database: /home/system1/.local/share/zeitgeist/activity.sqlite
1676 [2016-05-19 16:24:04,523] - DEBUG - zeitgeist.sql - Core schema is good. DB loaded in 129.229068756ms
1676 [2016-05-19 16:24:04,548] - DEBUG - zeitgeist.extension - Searching for system extensions in: /usr/share/zeitgeist/_zeitgeist/e
1676 [2016-05-19 16:24:04,570] - DEBUG - zeitgeist.extension - Searching for user extensions in: /home/system1/.local/share/zeitgeis
1676 [2016-05-19 16:24:04,951] - DEBUG - zeitgeist.extension - No extra extensions
1676 [2016-05-19 16:24:04,952] - DEBUG - zeitgeist.extension - Found extensions: [<class
'_zeitgeist.engine.extensions.storagemonitor.StorageMonitor'>, <class '_zeitgeist.engine.extensions.gnome_activity_journal.ActivityJ
<class '_zeitgeist.engine.extensions.fts.SearchEngineExtension'>, <class '_zeitgeist.engine.extensions.datasource_registry.DataSourc
<class '_zeitgeist.engine.extensions.blacklist.Blacklist'>]
1676 [2016-05-19 16:24:04,952] - DEBUG - zeitgeist.extension - Loading extension 'StorageMonitor'
1676 [2016-05-19 16:24:04,964] - DEBUG - zeitgeist.storagemonitor - Setting storage medium 0AB44834B4482513 'OSDisk' as available
1676 [2016-05-19 16:24:05,146] - DEBUG - zeitgeist.storagemonitor - Creating NetworkManager network monitor
1676 [2016-05-19 16:24:05,148] - DEBUG - zeitgeist.storagemonitor - NetworkManager network state: 70
1676 [2016-05-19 16:24:05,148] - DEBUG - zeitgeist.storagemonitor - Setting storage medium net 'Internet' as available
1676 [2016-05-19 16:24:05,201] - DEBUG - zeitgeist.extension - Loading extension 'ActivityJournalExtension'
1676 [2016-05-19 16:24:05,201] - DEBUG - zeitgeist.extension - Loading extension 'SearchEngineExtension'
1676 [2016-05-19 16:24:05,201] - DEBUG - zeitgeist.fts - Opening full text index: /home/system1/.local/share/zeitgeist/fts.index
1676 [2016-05-19 16:24:05,462] - DEBUG - zeitgeist.extension - Loading extension 'DataSourceRegistry'
1676 [2016-05-19 16:24:05,463] - DEBUG - zeitgeist.datasource_registry - Loaded data-source data from
/home/system1/.local/share/zeitgeist/datasources.pickle
1676 [2016-05-19 16:24:05,463] - DEBUG - zeitgeist.extension - Loading extension 'Blacklist'
1676 [2016-05-19 16:24:05,463] - DEBUG - zeitgeist.blacklist - No existing blacklist config found
1676 [2016-05-19 16:24:05,464] - INFO - root - Trying to start the datahub
1676 [2016-05-19 16:24:05,469] - DEBUG - root - Running datahub (/usr/bin/zeitgeist-datahub) with PID=1679
1676 [2016-05-19 16:24:05,470] - INFO - root - Starting Zeitgeist service...
1676 [2016-05-19 16:24:06,037] - DEBUG - zeitgeist.engine - Inserted 2 events out of 2 in 0.040020s
1676 [2016-05-19 16:24:29,352] - DEBUG - zeitgeist.fts - Indexing 'application://gedit.desktop'
1676 [2016-05-19 16:24:29,580] - DEBUG - zeitgeist.engine - Inserted 1 events out of 1 in 0.365042s
1676 [2016-05-19 16:24:29,580] - DEBUG - zeitgeist.notify - Checking monitor :1.104/org/gnome/zeitgeist/monitor/0
1676 [2016-05-19 16:24:30,284] - DEBUG - zeitgeist.fts - Committing FTS index
1676 [2016-05-19 16:24:30,998] - DEBUG - zeitgeist.engine - Inserted 1 events out of 1 in 0.479754s
1676 [2016-05-19 16:24:31,051] - DEBUG - zeitgeist.notify - Checking monitor :1.104/org/gnome/zeitgeist/monitor/0
1676 [2016-05-19 16:24:31,052] - DEBUG - zeitgeist.notify - Notifying :1.104/org/gnome/zeitgeist/monitor/0 about 1 insertions
1676 [2016-05-19 16:24:32,264] - DEBUG - zeitgeist.engine - Found 100 events in 1.210253s
```

The above is a snapshot of the daemon log file obtained. The daemon.log file has the logs of USB related activities. For instance, it can be realized that the first log in the snapshot corresponds to a log recorded when a file was opened from a USB drive directly, and that opens in LibreOffice Writer application which was set as the default application for opening documents in Ubuntu. Since this log file has numerous irrelevant messages, a separate daemon log parser was implemented specifically to capture the required fields from daemon.log. Below figure shows the output of the daemon.log parser with only required fields in the final message.



```
20-04-2016 01:16:47 DEBUG application://eclipse.desktop Accessed 01:20:10 201
20-04-2016 01:20:13 DEBUG application://gedit.desktop Accessed 01:20:13 4
20-04-2016 02:15:22 DEBUG Setting storage medium VBOXADDITIONS_4 storage device found 03:44:37 134
20-04-2016 03:44:37 DEBUG Setting storage medium net 'Internet' as available storage device found 03:44:37 3
20-04-2016 03:44:37 DEBUG application://eclipse.desktop Accessed 03:52:05 17
20-04-2016 03:52:07 DEBUG application://gedit.desktop Accessed 03:52:07 4
20-04-2016 03:55:22 DEBUG application://firefox.desktop Accessed 04:06:10 291
20-04-2016 04:16:03 DEBUG Setting storage medium VBOXADDITIONS_4 storage device found 05:33:02 16
20-04-2016 05:33:02 DEBUG Setting storage medium net 'Internet' as available storage device found 05:33:02 3
20-04-2016 05:33:02 DEBUG application://eclipse.desktop Accessed 05:33:50 9
20-04-2016 05:33:56 DEBUG application://firefox.desktop Accessed 05:41:35 12
20-04-2016 05:44:11 DEBUG Setting storage medium VBOXADDITIONS_4 storage device found 08:40:01 16
20-04-2016 08:40:03 DEBUG Setting storage medium net 'Internet' as available storage device found 08:40:03 3
19-04-2016 08:40:03 DEBUG application://gedit.desktop Accessed 10:47:35 9
19-04-2016 10:47:36 DEBUG file:///var/log/syslog visited 10:47:36 4
19-04-2016 10:47:36 DEBUG application://gedit.desktop Accessed 10:47:39 19
19-04-2016 10:48:06 DEBUG file:///var/log/mysql.log visited 10:48:07 4
19-04-2016 10:48:08 DEBUG application://gedit.desktop Accessed 10:48:09 13
19-04-2016 10:48:28 DEBUG file:///var/log/auth.log visited 10:48:28 7
19-04-2016 10:48:29 DEBUG Setting storage medium VBOXADDITIONS_4 storage device found 22:31:41 36
19-04-2016 22:31:42 DEBUG Setting storage medium net 'Internet' as available storage device found 22:31:42 3
19-04-2016 22:31:42 DEBUG file:///var/log/mysql.log visited 22:31:42 6
19-04-2016 22:31:42 DEBUG application://eclipse.desktop Accessed 22:39:31 57
19-04-2016 22:39:32 DEBUG application://firefox.desktop Accessed 22:39:32 4
19-04-2016 22:54:33 DEBUG application://gedit.desktop Accessed 23:00:27 10
19-04-2016 23:00:27 DEBUG file:///home/system1/Downloads/world.sql visited 23:00:27 3
19-04-2016 23:00:27 DEBUG Setting storage medium VBOXADDITIONS_4 storage device found 23:23:28 34
19-04-2016 23:23:28 DEBUG Setting storage medium net 'Internet' as available storage device found 23:23:28 3
19-04-2016 23:23:28 DEBUG Setting storage medium VBOXADDITIONS_4 storage device found 22:26:12 53
19-04-2016 23:09:38 DEBUG application://firefox.desktop Accessed 23:09:38 3
19-04-2016 23:09:41 DEBUG application://gedit.desktop Accessed 23:24:47 4
19-04-2016 23:24:49 DEBUG file:///var/log/syslog visited 23:24:49 1
19-04-2016 23:24:49 DEBUG application://gedit.desktop Accessed 23:24:54 24
19-04-2016 23:26:27 DEBUG file:///var/log/syslog visited 23:26:28 4
18-04-2016 23:26:29 DEBUG file:///home/system1/Downloads/world2.sql visited 10:09:02 33
18-04-2016 01:20:49 DEBUG application://firefox.desktop Accessed 09:51:56 12
```

C. *Auth Log Processing*

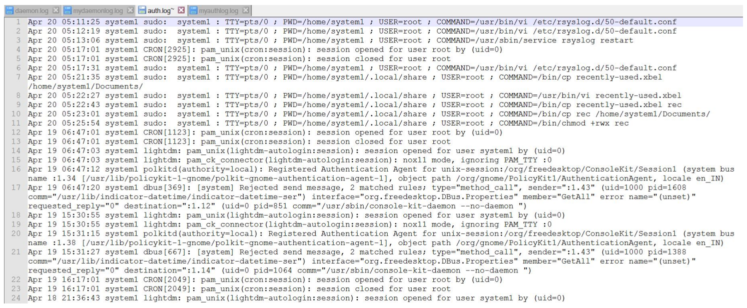
The table below summarizes the processing of the auth log.

TABLE II
PROCESSING OF AUTH.LOG FILE

Program	Auth Log
Input	auth.log
Process	Extracting fields. Calculating severity. Converting Date.
Final Fields	Date, Start and End time, Severity, Message, Frequency
Output	myauthlog.log
Note	Separate log files are created for each date and merged

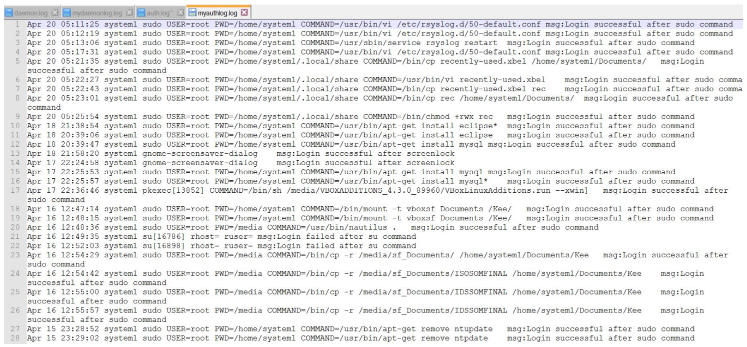
In Ubuntu OS, the Auth log tracks usage of authorization systems, the mechanisms for authorizing users, which prompt for user passwords, such as the Pluggable Authentication Module (PAM) system, the sudo command, remote logins to sshd, and so on. Basically, this log is useful for learning about user logins and usage of the sudo command is to view whether severity is present or not in the message. Similar to earlier parsing approaches, auth.log is also inspected, and the logs are parsed to convert into a human readable date format (dd-mm-yyyy).

Below figure is a snapshot of the auth.log file captured for preprocessing for extraction. The auth.log file has the logs of authorization related activities. For instance, it can be realized that the first log in the snapshot corresponds to a log recorded due to log in for a session by user1 in the system1. As it can be observed from the screenshot, since the log file has numerous irrelevant messages, a separate parser was implemented specifically to capture the required fields from auth.log.



```
Apr 20 05:11:25 system1 sudo:  system1 : TTY=pts/0 ; PWD=/home/system1 ; USER=root ; COMMAND=/usr/bin/vi /etc/rsyslog.d/50-default.conf
Apr 20 05:12:19 system1 sudo:  system1 : TTY=pts/0 ; PWD=/home/system1 ; USER=root ; COMMAND=/usr/bin/vi /etc/rsyslog.d/50-default.conf
Apr 20 05:13:06 system1 sudo:  system1 : TTY=pts/0 ; PWD=/home/system1 ; USER=root ; COMMAND=/usr/sbin/service rsyslog restart
Apr 20 05:17:01 system1 CRON[2925]: pam_unix(cron:session): session opened for user root by (uid=0)
Apr 20 05:17:01 system1 CRON[2925]: pam_unix(cron:session): session closed for user root
Apr 20 05:17:31 system1 sudo:  system1 : TTY=pts/0 ; PWD=/home/system1 ; USER=root ; COMMAND=/usr/bin/vi /etc/rsyslog.d/50-default.conf
Apr 20 05:21:35 system1 sudo:  system1 : TTY=pts/0 ; PWD=/home/system1/.local/share ; USER=root ; COMMAND=/bin/cp recently-used.xbel
/home/system1/Documents/
Apr 20 05:22:27 system1 sudo:  system1 : TTY=pts/0 ; PWD=/home/system1/.local/share ; USER=root ; COMMAND=/usr/bin/vi recently-used.xbel
Apr 20 05:22:43 system1 sudo:  system1 : TTY=pts/0 ; PWD=/home/system1/.local/share ; USER=root ; COMMAND=/bin/cp recently-used.xbel rec
Apr 20 05:23:01 system1 sudo:  system1 : TTY=pts/0 ; PWD=/home/system1/.local/share ; USER=root ; COMMAND=/bin/cp rec /home/system1/Documents/
Apr 20 05:25:54 system1 sudo:  system1 : TTY=pts/0 ; PWD=/home/system1/.local/share ; USER=root ; COMMAND=/bin/chmod +rwx rec
Apr 19 06:47:01 system1 CRON[1123]: pam_unix(cron:session): session opened for user root by (uid=0)
Apr 19 06:47:01 system1 CRON[1123]: pam_unix(cron:session): session closed for user root
Apr 19 06:47:03 system1 lightdm: pam_unix(lightdm-autologin:session): session opened for user system1 by (uid=0)
Apr 19 06:47:03 system1 lightdm: pam_ck_connector(lightdm-autologin:session): nox11 mode, ignoring PAM_TTY :0
Apr 19 06:47:12 system1 polkitd(authority=local): Registered Authentication Agent for unix-session:/org/freedesktop/ConsoleKit/Session1 (system bus
name :1.34 [/usr/lib/policykit-1-gnome/polkit-gnome-authentication-agent-1], object path /org/gnome/PolicyKit1/AuthenticationAgent, locale en_IN)
Apr 19 06:47:20 system1 dbus[369]: [system] Rejected send message, 2 matched rules; type="method_call", sender=":1.43" (uid=1000 pid=1608
comm="/usr/lib/indicator-datetime/indicator-datetime-ser") interface="org.freedesktop.DBus.Properties" member="GetAll" error name="(unset)"
requested_reply="0" destination=":1.12" (uid=0 pid=851 comm="/usr/sbin/console-kit-daemon --no-daemon ")
Apr 19 15:30:55 system1 lightdm: pam_unix(lightdm-autologin:session): session opened for user system1 by (uid=0)
Apr 19 15:30:55 system1 lightdm: pam_ck_connector(lightdm-autologin:session): nox11 mode, ignoring PAM_TTY :0
Apr 19 15:31:15 system1 polkitd(authority=local): Registered Authentication Agent for unix-session:/org/freedesktop/ConsoleKit/Session1 (system bus
name :1.38 [/usr/lib/policykit-1-gnome/polkit-gnome-authentication-agent-1], object path /org/gnome/PolicyKit1/AuthenticationAgent, locale en_IN)
Apr 19 15:31:27 system1 dbus[667]: [system] Rejected send message, 2 matched rules; type="method_call", sender=":1.43" (uid=1000 pid=1388
comm="/usr/lib/indicator-datetime/indicator-datetime-ser") interface="org.freedesktop.DBus.Properties" member="GetAll" error name="(unset)"
requested_reply="0" destination=":1.14" (uid=0 pid=1064 comm="/usr/sbin/console-kit-daemon --no-daemon ")
Apr 19 16:17:01 system1 CRON[2049]: pam_unix(cron:session): session opened for user root by (uid=0)
Apr 19 16:17:01 system1 CRON[2049]: pam_unix(cron:session): session closed for user root
Apr 18 21:36:43 system1 lightdm: pam_unix(lightdm-autologin:session): session opened for user system1 by (uid=0)
```

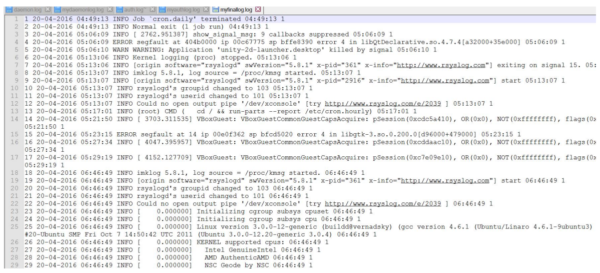
Below figure shows the output of the auth.log parser program with only required fields in the final message.



```
Apr 20 05:11:25 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/bin/vi /etc/rsyslog.d/50-default.conf msg:Login successful after sudo command
Apr 20 05:12:19 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/bin/vi /etc/rsyslog.d/50-default.conf msg:Login successful after sudo command
Apr 20 05:13:06 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/sbin/service rsyslog restart  msg:Login successful after sudo command
Apr 20 05:17:31 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/bin/vi /etc/rsyslog.d/50-default.conf msg:Login successful after sudo command
Apr 20 05:21:35 system1 sudo USER=root PWD=/home/system1/.local/share COMMAND=/bin/cp recently-used.xbel /home/system1/Documents/   msg:Login
successful after sudo command
Apr 20 05:22:27 system1 sudo USER=root PWD=/home/system1/.local/share COMMAND=/usr/bin/vi recently-used.xbel    msg:Login successful after sudo comma
Apr 20 05:22:43 system1 sudo USER=root PWD=/home/system1/.local/share COMMAND=/bin/cp recently-used.xbel rec    msg:Login successful after sudo comma
Apr 20 05:23:01 system1 sudo USER=root PWD=/home/system1/.local/share COMMAND=/bin/cp rec /home/system1/Documents/  msg:Login successful after sudo
command
Apr 20 05:25:54 system1 sudo USER=root PWD=/home/system1/.local/share COMMAND=/bin/chmod +rwx rec   msg:Login successful after sudo command
Apr 18 21:38:54 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/bin/apt-get install eclipse*  msg:Login successful after sudo command
Apr 18 20:39:06 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/bin/apt-get install eclipse   msg:Login successful after sudo command
Apr 18 20:39:47 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/bin/apt-get install mysql msg:Login successful after sudo command
Apr 18 21:58:20 system1 gnome-screensaver-dialog    msg:Login successful after screenlock
Apr 17 22:24:58 system1 gnome-screensaver-dialog    msg:Login successful after screenlock
Apr 17 22:25:53 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/bin/apt-get install mysql msg:Login successful after sudo command
Apr 17 22:25:57 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/bin/apt-get install mysql*    msg:Login successful after sudo command
Apr 17 22:36:46 system1 pkexec[13852] COMMAND=/bin/sh /media/VBOXADDITIONS_4.3.0_89960/VBoxLinuxAdditions.run --xwin]   msg:Login successful after
sudo command
Apr 16 12:47:14 system1 sudo USER=root PWD=/home/system1 COMMAND=/bin/mount -t vboxsf Documents /Kee/   msg:Login successful after sudo command
Apr 16 12:48:15 system1 sudo USER=root PWD=/home/system1 COMMAND=/bin/mount -t vboxsf Documents /Kee/   msg:Login successful after sudo command
Apr 16 12:48:36 system1 sudo USER=root PWD=/media COMMAND=/usr/bin/nautilus .   msg:Login successful after sudo command
Apr 16 12:49:35 system1 su[16786] rhost= ruser= msg:Login failed after su command
Apr 16 12:52:03 system1 su[16898] rhost= ruser= msg:Login failed after su command
Apr 16 12:54:29 system1 sudo USER=root PWD=/media COMMAND=/bin/cp -r /media/sf_Documents/ /home/system1/Documents/Kee   msg:Login successful after
sudo command
Apr 16 12:54:42 system1 sudo USER=root PWD=/media COMMAND=/bin/cp -r /media/sf_Documents/ISOSOMFINAL /home/system1/Documents/Kee    msg:Login
successful after sudo command
Apr 16 12:55:00 system1 sudo USER=root PWD=/media COMMAND=/bin/cp -r /media/sf_Documents/IDSSOMFINAL /home/system1/Documents/Kee    msg:Login
successful after sudo command
Apr 16 12:55:57 system1 sudo USER=root PWD=/media COMMAND=/bin/cp -r /media/sf_Documents/IDSSOMFINAL /home/system1/Documents/Kee    msg:Login
successful after sudo command
Apr 15 23:28:52 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/bin/apt-get remove ntupdate   msg:Login successful after sudo command
Apr 15 23:29:02 system1 sudo USER=root PWD=/home/system1 COMMAND=/usr/bin/apt-get remove ntpdate    msg:Login successful after sudo command
```

D. *Final processing*

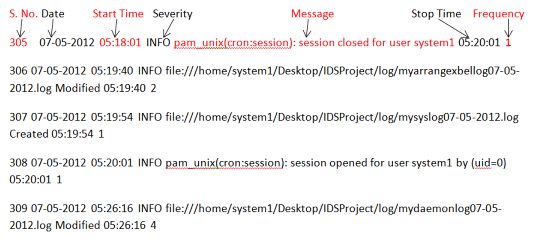
Once all the above preprocessing is completed,the program fuses all heterogeneous log files into a single unit and sorts it by date and time. Since preprocessing was carried out uniformly, the next part of implementation involved only the logic to merge the log files and sort them by date and time. Below figure is the final, fully preprocessed repository ready for the next processes of this system.



```
1 20-04-2016 04:49:13 INFO Job `cron.daily' terminated 04:49:13 1
2 20-04-2016 04:49:13 INFO Normal exit (1 job run) 04:49:13 1
3 20-04-2016 05:06:09 INFO [ 2762.951387] show_signal_msg: 9 callbacks suppressed 05:06:09 1
4 20-04-2016 05:06:09 ERROR segfault at 404b0000 ip 00c67775 sp bffe8390 error 4 in libQtDeclarative.so.4.7.4[a32000+35e000] 05:06:09 1
5 20-04-2016 05:06:10 WARN WARNING: Application 'unity-2d-launcher.desktop' killed by signal 05:06:10 1
6 20-04-2016 05:13:06 INFO Kernel logging (proc) stopped. 05:13:06 1
7 20-04-2016 05:13:06 INFO [origin software="rsyslogd" swVersion="5.8.1" x-pid="361" x-info="http://www.rsyslog.com"] exiting on signal 15. 05:
8 20-04-2016 05:13:07 INFO imklog 5.8.1, log source = /proc/kmsg started. 05:13:07 1
9 20-04-2016 05:13:07 INFO [origin software="rsyslogd" swVersion="5.8.1" x-pid="2916" x-info="http://www.rsyslog.com"] start 05:13:07 1
10 20-04-2016 05:13:07 INFO rsyslogd's groupid changed to 103 05:13:07 1
11 20-04-2016 05:13:07 INFO rsyslogd's userid changed to 101 05:13:07 1
12 20-04-2016 05:13:07 INFO Could no open output pipe '/dev/xconsole' [try http://www.rsyslog.com/e/2039 ] 05:13:07 1
13 20-04-2016 05:17:01 INFO (root) CMD (   cd / && run-parts --report /etc/cron.hourly) 05:17:01 1
14 20-04-2016 05:21:50 INFO [ 3703.311535] VBoxGuest: VBoxGuestCommonGuestCapsAcquire: pSession(0xcdc5a410), OR(0x0), NOT(0xffffffff), flags(0x
05:21:50 1
15 20-04-2016 05:23:15 ERROR segfault at 14 ip 00e0f362 sp bfcd5020 error 4 in libgtk-3.so.0.200.0[d96000+479000] 05:23:15 1
16 20-04-2016 05:27:34 INFO [ 4047.395957] VBoxGuest: VBoxGuestCommonGuestCapsAcquire: pSession(0xcddaac10), OR(0x0), NOT(0xffffffff), flags(0x
05:27:34 1
17 20-04-2016 05:29:19 INFO [ 4152.127709] VBoxGuest: VBoxGuestCommonGuestCapsAcquire: pSession(0xc7e09e10), OR(0x0), NOT(0xffffffff), flags(0x
05:29:19 1
18 20-04-2016 06:46:49 INFO imklog 5.8.1, log source = /proc/kmsg started. 06:46:49 1
19 20-04-2016 06:46:49 INFO [origin software="rsyslogd" swVersion="5.8.1" x-pid="361" x-info="http://www.rsyslog.com"] start 06:46:49 1
20 20-04-2016 06:46:49 INFO rsyslogd's groupid changed to 103 06:46:49 1
21 20-04-2016 06:46:49 INFO rsyslogd's userid changed to 101 06:46:49 1
22 20-04-2016 06:46:49 INFO Could no open output pipe '/dev/xconsole' [try http://www.rsyslog.com/e/2039 ] 06:46:49 1
23 20-04-2016 06:46:49 INFO [    0.000000] Initializing cgroup subsys cpuset 06:46:49 1
24 20-04-2016 06:46:49 INFO [    0.000000] Initializing cgroup subsys cpu 06:46:49 1
25 20-04-2016 06:46:49 INFO [    0.000000] Linux version 3.0.0-12-generic (buildd@vernadsky) (gcc version 4.6.1 (Ubuntu/Linaro 4.6.1-9ubuntu3)
#20-Ubuntu SMP Fri Oct 7 14:50:42 UTC 2011 (Ubuntu 3.0.0-12.20-generic 3.0.4) 06:46:49 1
26 20-04-2016 06:46:49 INFO [    0.000000] KERNEL supported cpus: 06:46:49 1
27 20-04-2016 06:46:49 INFO [    0.000000]   Intel GenuineIntel 06:46:49 1
28 20-04-2016 06:46:49 INFO [    0.000000]   AMD AuthenticAMD 06:46:49 1
29 20-04-2016 06:46:49 INFO [    0.000000]   NSC Geode by NSC 06:46:49 1
```

E. Numerical Conversion for Supervised Learning

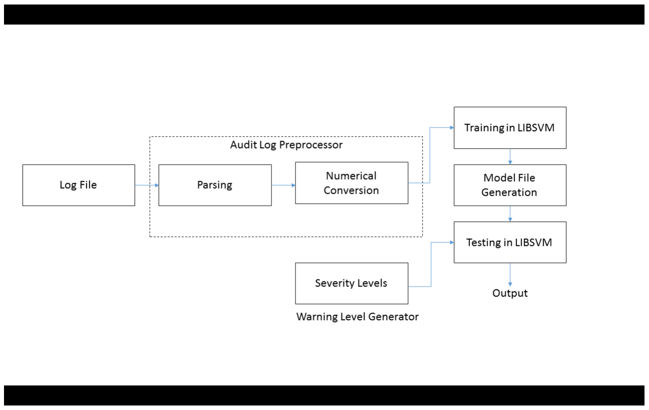
The final part of the system is the numerical conversion. The log files subjected for experimentation had various alphanumeric fields such as Serial No, Date, Start Time, Severity, Message, Stop Time, and Frequency. In machine learning theory, the process and functioning of unsupervised machine learning algorithms purely depends on numerical figures, i.e. Numerical Features. Hence the generated final.log file must be converted to a numerical form with a meaningful logic encoded by the conversion. For daemon.log, where file path is present, is encoded as a 64-bit character. ASCII value of these characters are computed and summed. This unique result served as a feature of the File Path. Base64 encoding scheme was the base for the logic used in this translation.


S. No.
Date
Start Time
Severity
Message
Stop Time
Frequency

```
305   07-05-2012 05:18:01 INFO pam_unix(cron:session): session closed for user system1 05:20:01 1

306 07-05-2012 05:19:40 INFO file:///home/system1/Desktop/IDSProject/log/myarrangexbellog07-05-
2012.log Modified 05:19:40 2

307 07-05-2012 05:19:54 INFO file:///home/system1/Desktop/IDSProject/log/mysyslog07-05-2012.log
Created 05:19:54 1

308 07-05-2012 05:20:01 INFO pam_unix(cron:session): session opened for user system1 by (uid=0)
05:20:01 1

309 07-05-2012 05:26:16 INFO file:///home/system1/Desktop/IDSProject/log/mydaemonlog07-05-
2012.log Modified 05:26:16 4
```

F. Supervised Learning

A summarized architecture of the supervised learning process is depicted in the figure below.



G. Selection criteria for SVM

Among supervised machine learning models, Support Vector Machine (SVM) is a contemporary and a powerful method for n-class classification. SVM is a widely used classifier in the both research and industry; but the best result from SVM requires a thorough knowledge of its working and the various ways in which accuracy can be increased. The advantages of SVM:

- Robust, works even when training examples contain errors
- Fast evaluation of the learned target function
- Prediction accuracy is high
- Able to generalize well to unseen examples
- There is no problem of local minima
- Based on sound mathematical foundation

The Disadvantages of SVM are:

- Long training time
- Difficult to understand the learned function
- Not easy to incorporate domain knowledge

SVM implementation comprises of,

- Training Phase
- Testing Phase

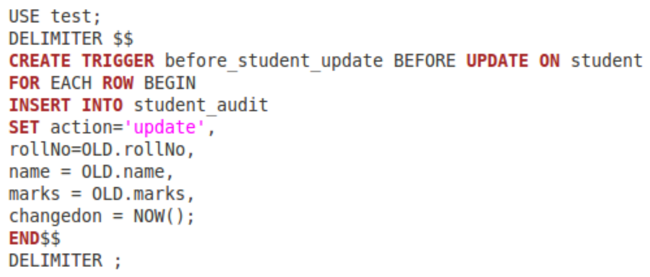
Training phase involves taking into account the known features and making the system to learn and make patterns out of it. In testing phase the data is kept aside from the training phase; but exclusively used to test the trained system. In the experimentation, SVM modeling is realized using LIBSVM tool, which is a popular library for Support Vector Machines.

H. Database Log Processing

While creating a prototype for database log monitoring, a real time scenario was imagined where the Cloud User (user of the cloud services) requires having security related to the database hosted in VM. Generally, most of the databases have their own events and activities being recorded as log files. For example, MySQL log installed in Ubuntu Operating System would be recorded under /var/log/mysql.log path. But, monitoring this file, preprocessing it and analyzing it with system again seems to be similar to how auth.log or other files were subjected. This mysql.log file generated may not be sufficient for designing a complete intrusion detection system as it records only the activities such as problems encountered in starting/stopping MySQL server (Error Log), changing the data (Binary Log), etc. Intrusions happening at database level, say which user has logged in to which sub-database, what the user did, etc. should also be recorded. One more level deeper, i.e. intrusions happening at database table level, even column or row level with respect to the corresponding insert/update/delete values intrusions can be recorded.

MySQL table level, row level and column level activities are monitored. This table level, row level and column level monitoring would play a vital role in real life applications. For example, consider a user manages some e-shopping store and wants to make sure that no unauthorized transaction information is altered in the database, which is hosted on a cloud. In this case, usage of triggers is one of the ideal options available. A trigger is a code section that gets executed whenever a table in a database is modified using the SQL Insert, Update or Delete statements. It is used generally as a validating tool or even as a security mechanism in database to monitor the changes made in a table. These can be employed on the tables, and whenever any modification viz. insertion, updating or deletion happens on the database, these triggers are generated and corresponding log is written in a custom SQL log file.

A database table level monitor in the form of trigger was implemented and a log gets recorded whenever a row level action is performed. A sample trigger is shown which was created for a student table with schema Roll Number, Name, Marks was present in the MySQL database. This trigger is automatically executed before an update operation is performed in the student table. For each entry, the old values of roll number, name and marks and the date of trigger execution is retrieved. These retrieved values are recorded in a custom log file by the system. The fields logged in the file are timestamp, module, and message.



```
USE test;
DELIMITER $$
CREATE TRIGGER before_student_update BEFORE UPDATE ON student
FOR EACH ROW BEGIN
INSERT INTO student_audit
SET action='update',
rollNo=OLD.rollNo,
name = OLD.name,
marks = OLD.marks,
changedon = NOW();
END$$
DELIMITER ;
```

Each entity in the training set for LIBSVM contains one target value" (i.e. the class labels) and several attributes" (i.e. the features or observed variables). The goal of SVM is to produce a model (based on the training data), which predicts the target values of the test data given only the test data attributes. The final output produced after numerical conversion is used as the training data for the SVM. When training the data set, a Model File is generated. This model file has all the patterns that are learnt by SVM in a native format and this file is used as a base input file during testing phase and in actual run. In eCloudIDS, once the system is deployed by the client at CSPs location, along with the configuration of SVM. The system would be populated based on the dry run of intrusion history of the client that include activities such as,

- Attacks in the history
- Hacking history
- Unauthorized list of users
- Unauthorized location access
- Unauthorized login and remote access, and so on.

On dry run with these intrusions, for each intrusion defined, different severity levels must be defined in Warning Level Generator Subsystem(WLG) in future and its corresponding action to be taken must be defined in the Alert System (in future). For example, unauthorized access of a file in root directory is an intrusion. On dry run, this would be identified as an intrusion pattern and this would be recorded in the Acute Audit Repository with a class label associated to it. For this intrusion, different security levels needs to be defined in Warning Level Generator and also the corresponding action to be taken must be configured in the Alert System. For example, in the above scenario, unauthorized access of a file in restricted directory by an admin user may have a low severity than unauthorized access of a file in restricted directory by a normal user may have a medium or a high severity. These security levels can be defined in WLG and corresponding actions to be taken are defined in Alert System. In the scenario taken, for unauthorized access with low priority send a mail to the admin, unauthorized access with medium severity would block the user temporarily and report it to admin as well as to the client. Hence by this dry run, the system would have a complete set of intrusion repository that the system would be able to classify and identify from its inception in CSPs location. In this implementation the following intrusions were considered and a dry run was performed to identify the patterns and severity levels were assigned to it in the WLG Subsystem.

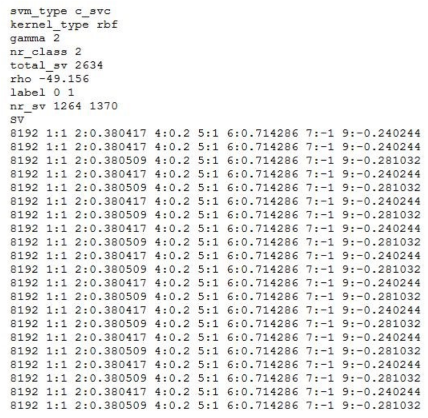
- Login successful after sudo command
- Login unsuccessful after sudo command
- Login failed after three attempts of sudo command
- Login successful after logout
- Changing the date and time by sudo command
- Changing the date and time by normal user
- Login by unknown user
- Login failed after screen lock
- Inserting a row in the table of MySQL database
- Deleting a row in the table of MySQL database
- Updating a row in the table of MySQL database
- Accessing /var/log/ directory by sudo command
- Accessing /var/log/ directory without sudo command

After a dry run in training the system, class labels were assigned for all the patterns generated. For all these patterns, for prototyping, severity levels are defined varying from 1 to 5, ascending starting with 1 as less severe. As a result of dry run, following would be accomplished

- Class label identification for the patterns
- Defined severity level for known intrusions
- SVM generated training Model File

The model file produced as a result of training phase in SVM model is used to enhance the learnability of SVM by defining the patterns and expected outputs corresponding to it. The SVM model was realized using the LIBSVM tool, a well-integrated and easy-to-use tool for Support Vector Classification.

LIBSVM supports multi-class classification and hence it was preferred for experimentation where multiple classes are required to be classified. Figure shows the generated model file after the training phase of LIBSVM in sX-Engine. In future, whenever the SVM encounters the exact pattern match or some similar pattern, then it has the ability to distinguish it into attack or normal data by looking into this file. Thus, this model file would be used in the testing phase of SVM as well in the deployment phase.



```
svm_type c_svc
kernel_type rbf
gamma 2
nr_class 2
total_sv 2634
rho -49.156
label 0 1
nr_sv 1264 1370
SV
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380509 4:0.2 5:1 6:0.714286 7:-1 9:-0.281032
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380509 4:0.2 5:1 6:0.714286 7:-1 9:-0.281032
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380509 4:0.2 5:1 6:0.714286 7:-1 9:-0.281032
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380509 4:0.2 5:1 6:0.714286 7:-1 9:-0.281032
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380509 4:0.2 5:1 6:0.714286 7:-1 9:-0.281032
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380509 4:0.2 5:1 6:0.714286 7:-1 9:-0.281032
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380509 4:0.2 5:1 6:0.714286 7:-1 9:-0.281032
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380509 4:0.2 5:1 6:0.714286 7:-1 9:-0.281032
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380509 4:0.2 5:1 6:0.714286 7:-1 9:-0.281032
8192 1:1 2:0.380417 4:0.2 5:1 6:0.714286 7:-1 9:-0.240244
8192 1:1 2:0.380509 4:0.2 5:1 6:0.714286 7:-1 9:-0.281032
```

The testing phase was important in the experimentation as the success factor of the training phase would be evaluated here. Some random logs that included anomalous patterns were primarily selected as testing data for the SVM. Any new entry in the log file is tested in order to determine whether it is an attack or not. This is done by the SVM, logically looking for a match if available with the SVM training data (i.e. Model File). Any log file subjected for testing using the SVM is firstly converted into a numerical form. This conversion is performed in a manner similar to the one already performed. This becomes the testing data to next phase for the SVM. In the testing phase, the LIBSVM uses the testing data obtained after numerical conversion and the model file generated during the training phase and classifies the pattern to a class label predefined.

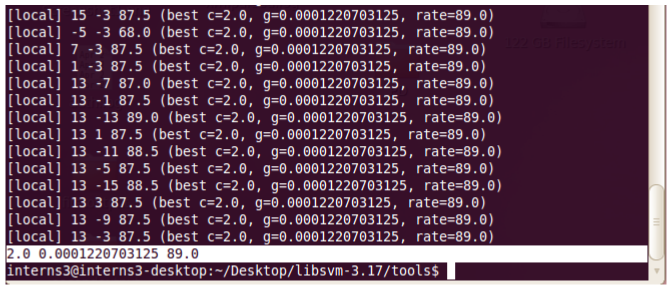
I. Analyzing the performance

As LIBSVM was used to realize the SVM model in experimentation, several parameters and configurations are available in LIBSVM to provide optimization of SVMs classifying capacity. Figure pictorially represents the LIBSVMs process flow. The Training File used can be optimized by passing it to the Scaler Process which is explained further. The Scaler Process must be followed in both Training Phase and Testing Phase. It cannot be applied to any one phase.

Scaling before applying SVM is very crucial to fine tune the data as subject. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Generally, scaling is for each feature in the range [-1, +1] or [0, 1]. The same scaling factor is used for training and testing data.

J. Cross Validation Accuracy

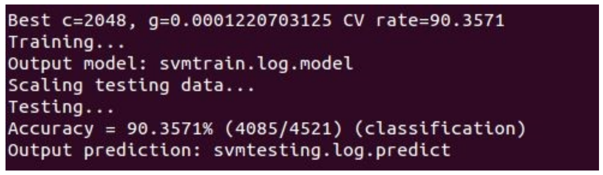
There are two parameters for an RBF kernel, they are C and gamma. It is not known beforehand which C and gamma are best for a given problem. Consequently some kind of model selection (parameter search) must be done. The goal is to identify good (C, gamma) so that the classifier can accurately predict unknown data (i.e. testing data). A common strategy is to separate the data set into two parts, of which one is considered unknown. The prediction accuracy obtained from the unknown" set more precisely reflects the performance on classifying an independent data set. An improved version of this procedure is known as cross-validation. In v-fold cross-validation, first divide the training set into v subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining v 1 subset. Thus, each entity of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified. The grid-search" was employed on C and gamma using cross-validation. Various pairs of (C, gamma) values are tried and the one with the best cross-validation accuracy was selected for optimizing the LIBSVM accuracy of SVM. Figure below depicts various C and gamma probabilities tried out in the experimentation. The Accuracy Rate is also obtained in the last column of each record.



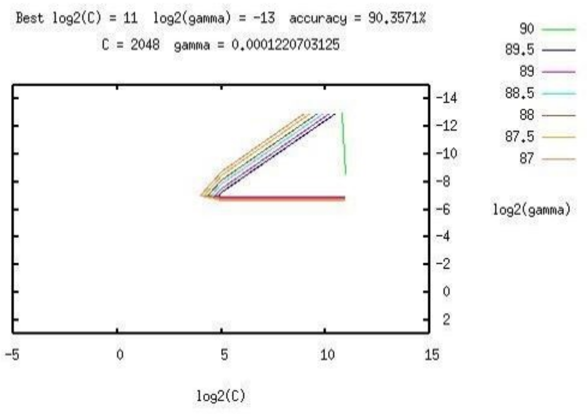
```
[local] 15 -3 87.5 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] -5 -3 68.0 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 7 -3 87.5 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 1 -3 87.5 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 13 -7 87.0 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 13 -1 87.5 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 13 -13 89.0 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 13 1 87.5 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 13 -11 88.5 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 13 -5 87.5 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 13 -15 88.5 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 13 3 87.5 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 13 -9 87.5 (best c=2.0, g=0.0001220703125, rate=89.0)
[local] 13 -3 87.5 (best c=2.0, g=0.0001220703125, rate=89.0)
2.0 0.0001220703125 89.0
interns3@interns3-desktop:~/Desktop/libsvm-3.17/tools$
```

The Cross Validation technique carried out during the training phase was meaningful. The Cross Validation was carried out directly in one of the Instance-M terminal using the LIBSVM package. Figure shows the Cross Validation repository where the final log file generated subjected to training and some randomly generated testing log file are placed. Figure shows the actual run performed using a python script named easy.py from the LIBSVM package.

This python script found in the LIBSVM would perform optimization of the training by suggesting the best C and gamma values. Figure shows the best values of C and gamma chosen by easy.py along with the CV rate and accuracy. From the log repository of 30 days collected from the three VMs, an accuracy of 90.3 percent was achieved during the Cross Validation. Figure below shows the graph plotted to reveal the Cross Validation Accuracy with c=2048 and gamma=0.0001220703125.



```
Best c=2048, g=0.0001220703125 CV rate=90.3571
Training...
Output model: svmtrain.log.model
Scaling testing data...
Testing...
Accuracy = 90.3571% (4085/4521) (classification)
Output prediction: svmtesting.log.predict
```



III. Challenges

The main challenges were in analyzing the information from the logs. A detailed study on these logs were made to find out which are vulnerable and which are authentic.

IV. Future Work and Thoughts

The work of database logs is still in progress. In future the system may be extended to almost all the logs to give a detailed analysis to the customer and end users about the cloud VM that is deployed.