UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING & INFORMATICS

TOPIC: DISTRIBUTED SYSTEM SECURITY

TITLE: Addressing the challenges in aggregation and correlation of security event data from custom enterprise applications

Supervisor; Dr. Christopher Chepken

AREA OF STUDY: Distributed Systems

Project type: Application Development

Submitted by: Mark Meli Too – P53/73232/2014

A project report submitted to the school of computing and informatics in partial fulfilment of the requirement for the award pf masters of Science in Computer Science of the University of Nairobi

Declaration

This project is my original work and to the best of my knowledge, it has not been presented for a degree in any other university.

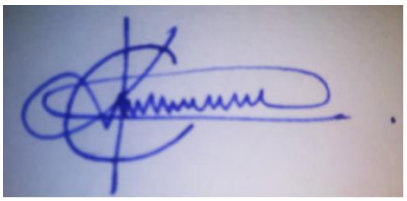
Mark Meli Too

Signed…….

Date……17/08/2021...……

This project has been submitted as partial fulfilment of requirements for the award of Masters of Science of the school of computing and informatics of the University of Nairobi, with my approval as the University Supervisor.

Dr. Christopher Chepken



Signed……...

Date……23/08/2021……..

Abstract

In Kenya most organizations are acquiring custom enterprise systems that help improve the efficiency of their business processes, these systems save their security logs in customized not standard format that cannot be extracted to upload to a SIEM.

This study is aimed at providing a solution that can assist IT security specialists to extract the security information events from the custom enterprise systems and automatically upload them to a SIEM.

This study outlines the development of a prototype application that extracts the security event information from the custom enterprise system aggregate them and apply some correlation rules then transmit the data to a SIEM. This is to ensure that IT security specialist have all the security information from the custom enterprise systems that will enable Security Operations center monitor and analyze activities

Acknowledgements

I would like to thank the almighty God for standing with me and giving me the strength to pursue my study.

To my parents and siblings, thank you for your prayers and encouragement during the entire time I was conducting the project.

To my supervisor, Dr. Christopher Chepken, thanks you for providing guidance and offering supportive advice as I was conducting the project

Table of Contents

List of figures

List of Tables

Abbreviations and Acronyms

IT Information Technology

SIEM Security Information and Events Management

SOC Security Operations Center

IT sec Information Technology Security

IDS Intrusion detection systems

DLP Data loss prevention system

SIM Security Information Management

SEM Security Events Management

VA Vulnerability Assessment

PCI Payment Card Industry standards

HIPAA Health Insurance Portability and Accountability Act

WMI Windows Management Instrumentation

IP Internet protocol

IETF Internet Engineering Task Force

1. INTRODUCTION

1.1 Background to the problem

Cyber security incidents can cause serious financial and reputation impact to an enterprise. In order to detect malicious activities, the SIEM (Security Information and Events Management) system are deployed in companies (Feng, Wu, & Liu, 2017). SIEM will aggregate and correlate event data from endpoint such as firewalls, IDS (Intrusion detection systems), DLP (Data loss prevention system), Windows/Unix security events and many other sources and send alerts in case of an event. These endpoints usually have data in a defined format depending on the type of endpoint, an example is the windows event manager which logs errors in a standard with event ID references that have been defined and made available to the public for reference purposes.

Endpoints that are usually neglected are the ones that have event data generated by custom enterprise application when in use. Developers usually create tables inside the system to log data such as successful or failed login attempts, changes done on a particular set of records, historical changes in user's profile, deletion of records among other things. This information is part of security information that is useful when investigating incidents that occurred inside the system.

The main challenge is that event data residing in these systems are not in a standardized format, Since there`s no standard way of creating these logs, developers are free to log these entries in any way they see fit. This makes it challenging to collect the data and save it in a SIEM. In case there`s an incident, the system administrator has to go through the data in the tables or a user interface designed by the developer to access that information. The purpose of this research is to develop a way to collect this data and standardize it in a way that can be fed to a SIEM.

1.2 Problem statement

When an organization purchases or develops an application, the application is deployed and used by the business for its intended business purpose. These applications already have an inbuilt mechanism which saves information related to security events inside its database.

Whenever a security incident related to that particular system occurs and event information is required for investigation, the IT security staff will request the system administrator to provide event information from the system that will be used for investigation. This is inefficient since the system administrator will spend time to extract and interpret the information then share with the information security officer for further investigation.

1.3 Research objective

The objective of this research is to create a prototype system that will collect and analyze custom event data from an enterprise application and feed the data to a SIEM to increase the efficiency in security incident management.

1.4 Significance of the research to key audiences

The aim of the research is to provide a prototype system that can be used by IT security specialist to collect custom security events data from custom enterprise applications that are already deployed in many organizations and feed them to existing SIEM systems.

1.5 Assumptions and limitations of the research

The risk to the project is the availability of sample data that will be used to generate the security events with running simulations

2. LITERATURE REVIEW

2.1 Introduction

This chapter we reviewed history and development of SIEM (Security Information and Events Management) systems, the operational principles and research on integrations done with Security Information and Event Management systems (SIEM). A Security Information and Event Management system is a system that provides two services in one, that is Security Information Management (SIM) and Security Events Management. SIM deals more with event log data management and SEM deals with event monitoring and incident management from systems, applications and network devices.

The term SIEM was coined by (Nicolett & Williams, 2005) in 2005, The practice back then was to do vulnerability assessments of the environment and apply security configuration on the systems to mitigate identified risks, (Nicolett & Williams, 2005) suggested that this process can be automated and also be used to collect real-time and historic events data from a wide set of heterogeneous sources. This could be used to filter incident information into data that can be acted on for the purposes of incident response and forensic analysis.

The first generations of SIEM systems started combining security event management with security information management for the first time (Gailey, 2020), they were used to achieve and maintain compliance such as Payment Card Industry (PCI) standards, the Health Insurance Portability and Accountability Act (HIPAA), Focus was mostly on what is happening within the organizations network. The current generation has evolved rapidly due the constantly changing threats and regulatory framework. (Pathak, Goldstein, & Horaist, 2020), this meant that it had to have the capability to aggregate data, apply some correlation rules and models so as to be able to identify threats that can be sent to IT security for action, it also had to have the capability to store this data for future use (such as forensic analysis and compliance reporting).

2.2 SIEM integration

The first step in SIEM system operational work flow is to collect security events logs data, a typical workflow is shown below,

Fig 1: SIEM operational workflow

SIEM systems works by deploying multiple collection agents in a hierarchical manner to gather security-related events from end-user devices, servers and network equipment, as well as specialized security equipment, such as firewalls, antivirus or intrusion prevention systems (Rosencrance, 2021).

Below is a representation from (Pathi, 2015) representing the various sources of data that a SIEM gets the event data. All these systems have to be integrated to a SIEM in order to share

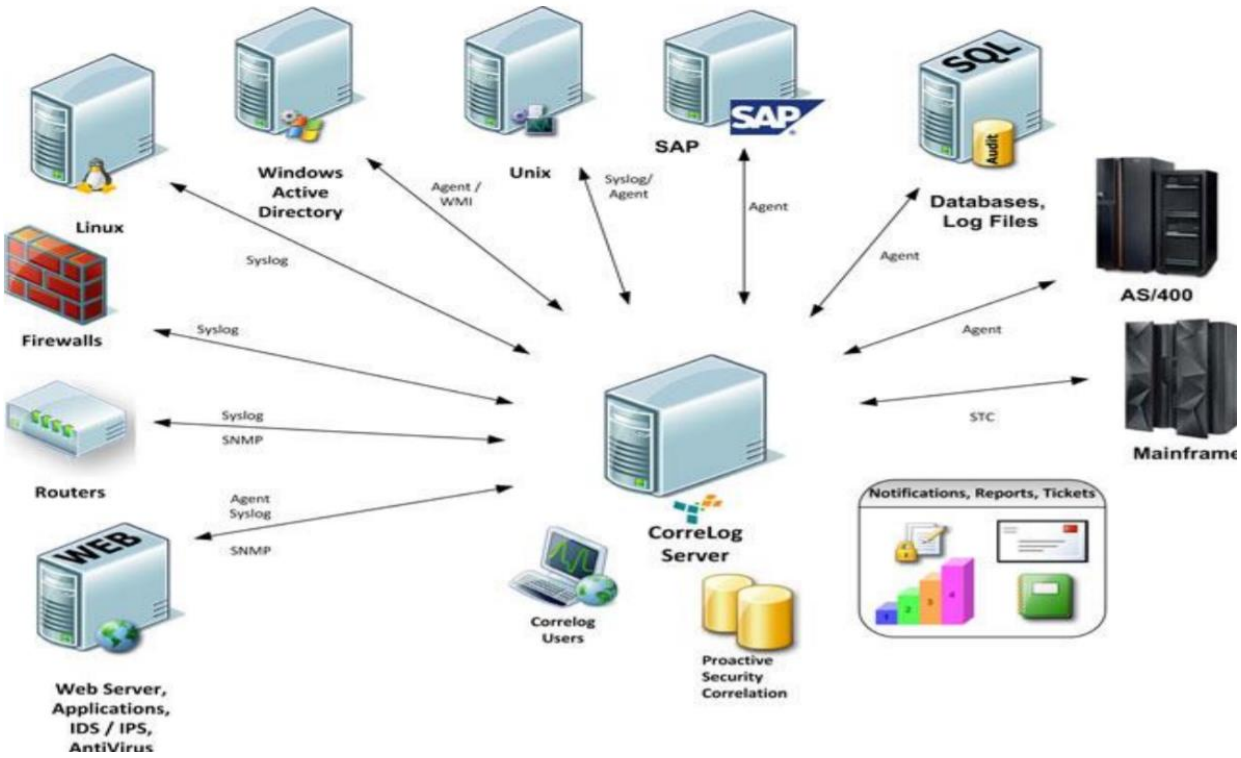


Fig 2: SIEM end points

security events data, they use a standardized message format 1.e Syslog, WMI, SNMP or SIEM agents installed in endpoints.

From the information above A SIEM system first needs to integrate with the system that generate the event data first so that they can collect data to start processing as shown in the work flow process.

2.3 Literature Summary

For SIEM to be effective it should be able to integrate to other systems such as custom enterprise applications and collect data for analysis and this is where the challenges emerges. Most SIEM are designed to integrate well with established application, data monitoring and identity

infrastructure (Kavanagh & Nicolett, 2014). What happens to the custom application being used within enterprises that were developed by small companies or developed by in-house teams?

The other major hurdle that came out across while researching on integration of SIEM with custom application is that there are very few scholarly articles on this topic. SIEM development has grown and improved during the recent years but the gap that still exists is collection of event data from the many types of Enterprise application. The more established companies such as Microsoft, oracle have created standardized event log library that have been made public making it easy for SIEM developers to create connectors compatible to them.

To close this gap a prototype system was designed and developed to collect security events information from an enterprise system process the information and feed the information to a SIEM.

2.4 Prototype Design

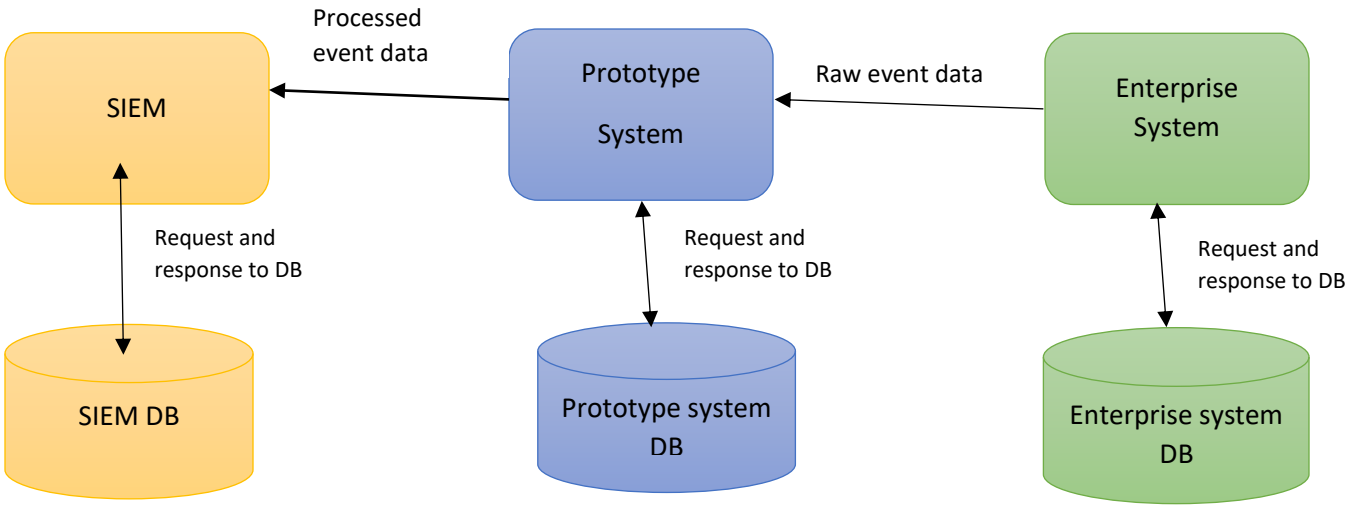


Fig 3: Prototype design

The IT security specialist who is the user who will run the application, the user will have an option to extract information from the enterprise system and load it to the prototype. Once the data is loaded into the prototype system the user will have an option the process the data and send it to the available SIEM system.

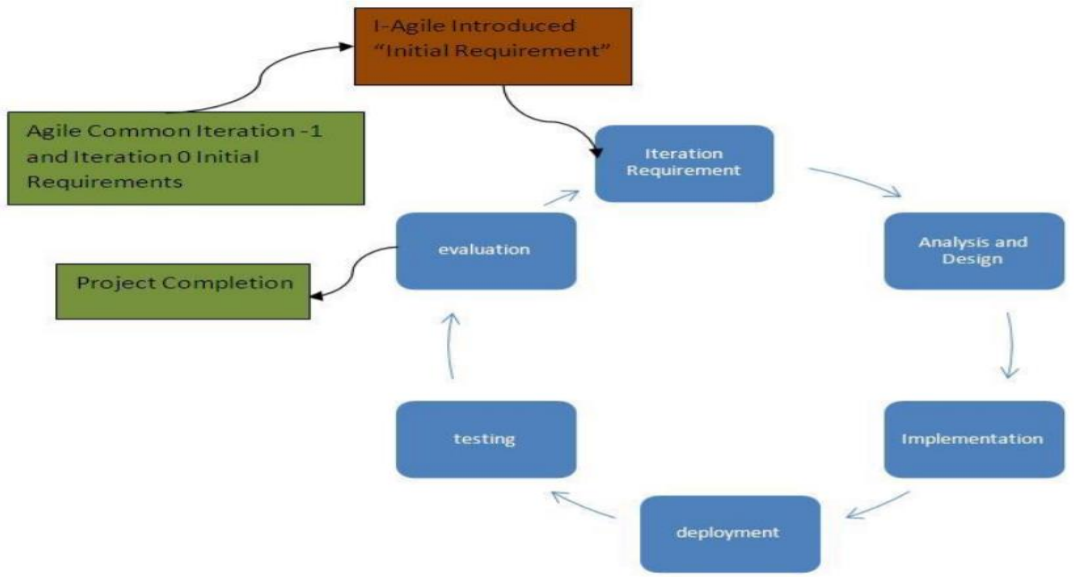
3. METHODOLOGY

3.1 Prototype Application development

3.1.1 Introduction

This chapter outline the methodology used for the development of the prototype system.

The software development methodology that was used is agile software development. It described the various phases involved in the development which includes an iterative requirement phase, analysis and design phase, Implementation phase, deployment phase, testing phase and Evaluation phase. These phase



:

Fig 4: Agile development methodology adapted from (Rekaby & Soliman, 2012)

This was the detailed design phases that involved how the features and interfaces were developed, tested and deployed to ensure the prototype systems performs the functions that were expected. The phases involved;

3.1.2 Agile Methodology

The prototype application had several components that were development and tested individually, this made it appropriate to use and agile methodology with the following phases;

3.1.2.1 Iteration requirement Phase

During this phase the rundown on how the application was to work in order to achieve the required objective were explored. The functional requirements were;

- The system was supposed to have a user interface that a user will use to interact with the system to view the extract data, initiate extraction, processing and get feedback to indicate the success or failure of an operation
- The application was supposed to be able to connect the enterprise system so that it can be able to get the data it required to perform its operations
- The system was supposed to be able to aggregate event data from several modules with the system
- The system was supposed to be able to generate standard Syslog messages and send the message to a syslog server
- The system was supposed be able to apply a correlation rule on the data aggregated from enterprise system and send it to a syslog server
- Systems that were required to demonstrate the functionality of the prototype were also identified.

The functional and components requirements provided the foundational principles that were used in the analysis and design phase of the development

3.1.2.2 Analysis and design

3.1.2.2.1 Requirements Analysis

The primary requirement of the system was to be able to collect and aggregated security event data from and enterprise application and forward to a SIEM. We broke the requirement to four parts that we used to breakdown the requirement details, they are Raw event data collected from source, the processes that were applied on the data, the message format that was used send the event information to a SIEM and the correlation rules that were applied on the processed data before it was forwarded to the SIEM

Raw Event data

The raw data from the enterprise application used were user login event data and user activity data. User login event data contains details related to login sessions such as the login date time, login user details and the IP address where the users has logged in to the system from. Below is a user login activity log screenshot from the enterprise system

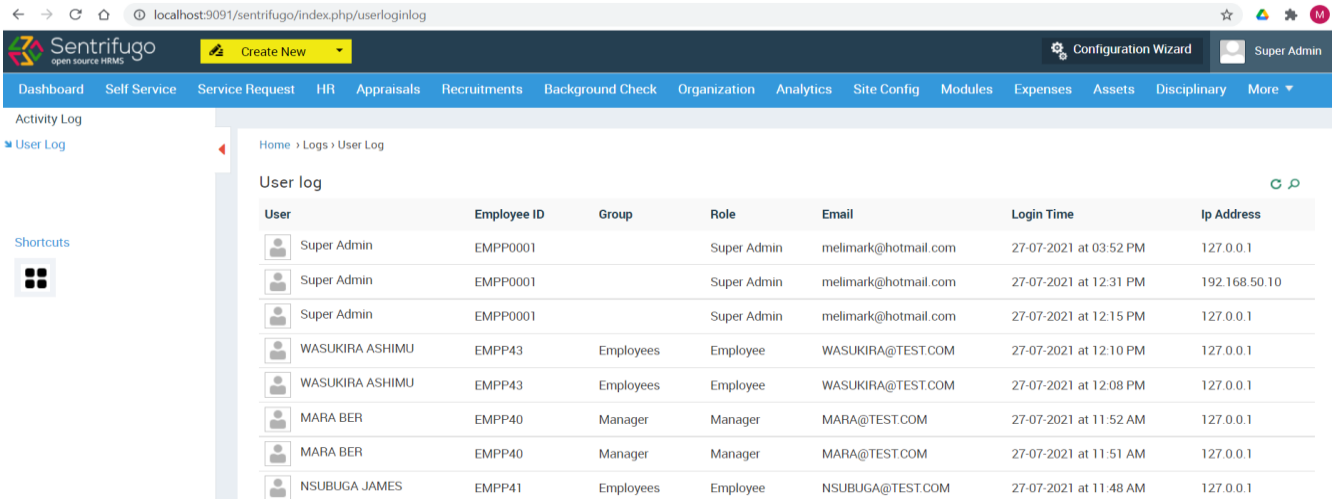


Fig 5: Sentrifugo Users log

The user activity data contained detailed changes of what the user performed in the enterprise application, these includes what action was performed (either a record was added, modified,, or deleted), who performed the action, when the action was performed, and what record was modified.

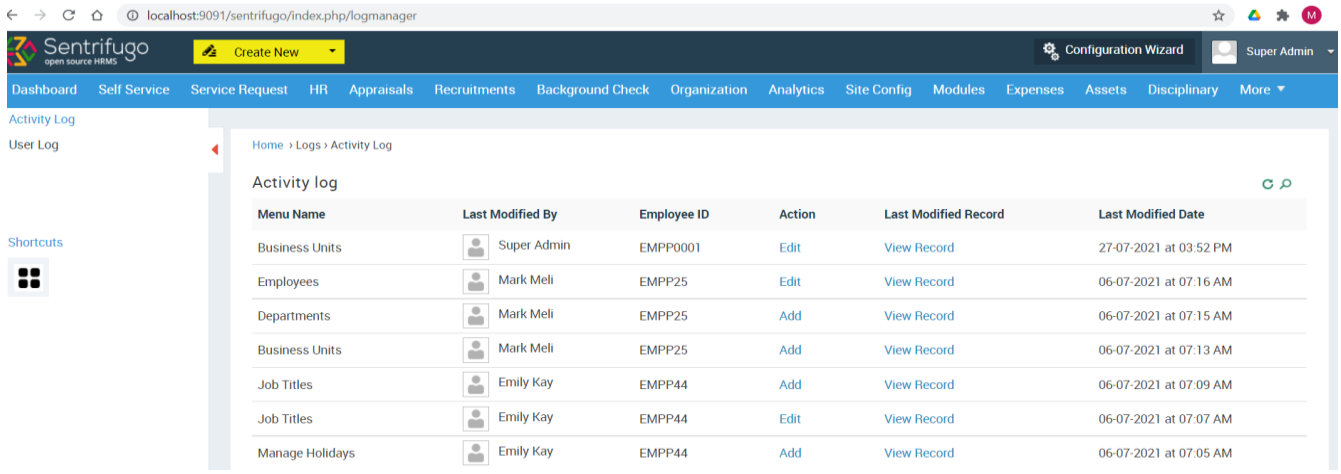


Fig 6: Sentrifugo Activity log

The user activity log the system generated was collected directly from tables containing the data, an example is a table called main_businessunits. This table contained the details of

business units configured in the system, there are columns defined that were used to record details of who created the first record, when they created the record, who modified the record, when it was modified. Below is an example of the description of the table.

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	int unsigned	NO	PRI	NULL	auto_increme
unitname	varchar(255)	NO		NULL	
unitcode	varchar(50)	YES		NULL	
description	text	YES		NULL	
startdate	date	YES		NULL	
country	int unsigned	YES		NULL	
state	int unsigned	YES		NULL	
city	int unsigned	YES		NULL	
address1	text	YES		NULL	
address2	text	YES		NULL	
address3	text	YES		NULL	
timezone	int	YES		NULL	
unithead	varchar(255)	YES		NULL	
service_desk_flag	tinyint unsig...	YES		1	
createdby	int unsigned	YES		NULL	
modifiedby	int unsigned	YES		NULL	
createddate	datetime	YES		NULL	
modifieddate	datetime	YES		NULL	
isactive	tinyint(1)	YES		1	

Fig 7: Business Units table

The fields called createdby, modifiedby, createddate, and modifieddate were used in all tables defined in the system, and they are used to track changes done to records in the tables. The user activity log menu uses these fields to display the changes done on the system menu. The main issue was that the system didn`t have the capability to send the information contained in these columns to a SIEM. Most custom enterprise systems log changes in this way.

Data processing

The processing that the solution was required to perform on the data was to collect the user login logs and user activity logs, aggregate the event data from multiple tables into on table, apply three correlation rules on the data and send the data to a SIEM.

The first step was to collect the data to establish the relationship between the tables that contained the event data and the table containing the user details, the first table that was used was the main_usersloginlog. The description of the table is shown below;

Main_userloginlog
id
userid
emprole
group_id
employeeId
emailaddress
userfullname
logindatetime
empipaddress
profileimg

Table 1: Sentrifugo User login log table

The main_businessunits table will be used to demonstrate the process. Below is the relationship between the main_businessunits table that the main users table that contains all the details about the users

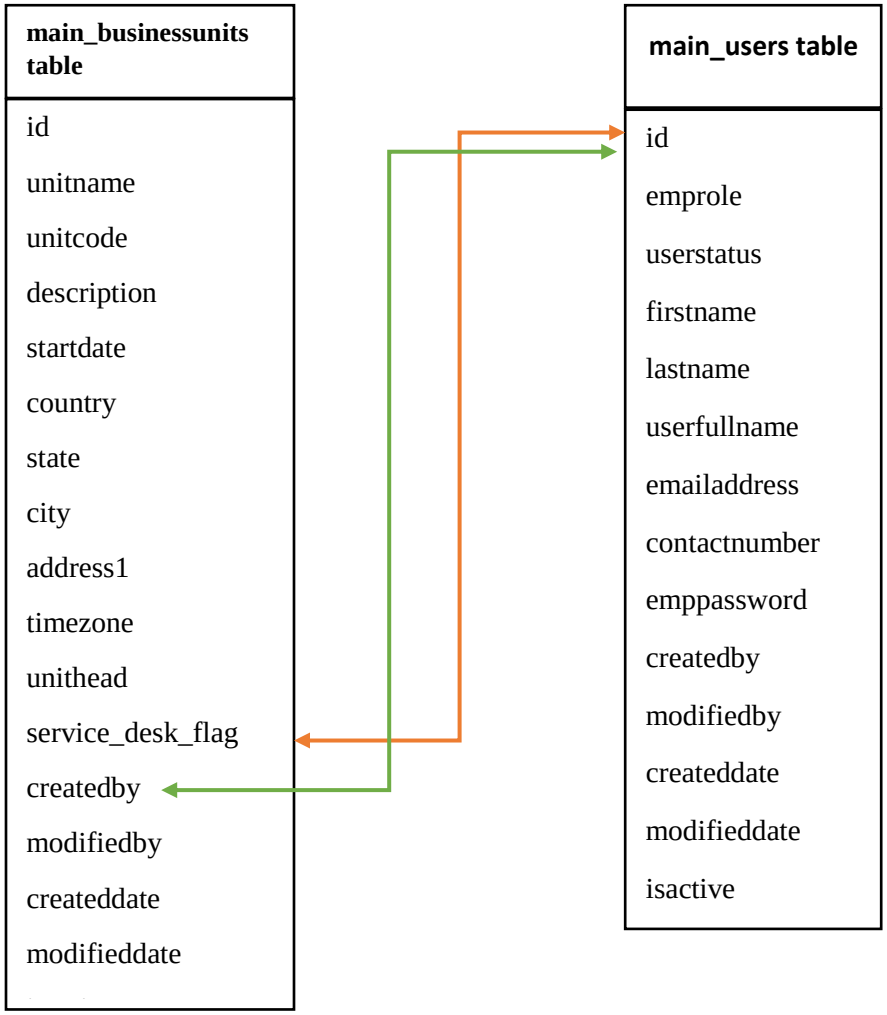


Table 2: Relationship between Sentrifugo Main_Businessunits table and Main_users table

In this scenario the createdby and modifiedby fields were foreign keys in main_businessunits table that correspond to the id field which is the primary key in main_users table. A view was created to extract the record details and the details of the user who performed the action. This was replicated on all the tables whose user activity details were being collected from.

The event data collected from multiples tables as described in the previous paragraph were aggregated together into one table that was used to apply the correlation rules and generate the security activity event messages that was to be sent to the SIEM.

Syslog format

The message format chosen to deliver the security event data was the Syslog format as defined by Internet Engineering Task Force (IETF) in (Gerhards, The Syslog Protocol, 2021). Syslog stands for system logging protocol that is used by systems to communicate messages with the logging server.

The protocol was initially used to monitor network devices and send notification messages to log collectors or syslog servers where SIEMs can get the information from. Currently the protocol has attained wide support on many operating systems including Linux, UNIX, Apples MacOS and windows, the solution took advantage of this support and extended the functionality to send security event data generated by custom enterprise systems.

The syslog message components that were used are;

Timestamp	Facility Code and Severity	Device ID	Message Text

Table 3: Syslog Message Format

- Timestamp

  This is the date and time the event occurred
- Device ID

  The device id field was used to identify the source of the message, it can be the device hostname, IP address, text string. IP address was to be used as the device ID
- Facility Code and Severity

  Syslog protocol has defines facility codes and severities as shown below;

As standard some facilities have already been assigned to specific operating system daemons and process so we facility code 16 was chosen since it1s in the range of facility codes that have not been explicitly assigned to any system. The severity level to be used was to be determined by the result after the correlation rules were applied

Numeric Code	Facility
0	kernel messages
1	user-level messages
2	mail system
3	system daemons
4	security/authorization messages
5	messages generated internally by syslogd
6	line printer subsystem
7	network news subsystem
8	UUCP subsystem
9	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon (note 2)
16	local use 0 (local0)
17	local use 1 (local1)
18	local use 2 (local2)
19	local use 3 (local3)
20	local use 4 (local4)
21	local use 5 (local5)
22	local use 6 (local6)
23	local use 7 (local7)

Table 4: Syslog facilities

Severity Code	Severities
0	Emergency: system is unusable
1	Alert: action must be taken immediately
2	Critical: critical conditions
3	Error: error conditions
4	Warning: warning conditions
5	Notice: normal but significant condition
6	Informational: informational messages
7	Debug: debug-level messages

Table 5: Syslog Severities

- Message text

The message text contained the description of the operations that was performed, the timestamp, user details, the record details, the host details and the severity of the event

Correlation rules

There are three Correlation rules, two with two second level rules that were applied on the aggregated data collected from the enterprise system. The result from the rules were used to create syslog messages with a severity that matches the risk exhibited by the event. There rules were;

1. **Login from outside the corporate network rule**

This rule filters all successful login by users who are from outside the corporate network. The logic behind was the system was not supposed to be accessed from outside the corporate network, incase this policy was violated the system was to generate an alert and send the information to the SIEM. The system generated a Syslog message with an alert severity level that was forwarded to a Syslog server, the system provided and option to either forward the message or not after which it proceeds to executing the second rule which is a sub rule of the first as described below.

1.1. Extract the all user activities of the user who logged in from and external network sub rule

The second sub rule filtered records related to the first event. It picked all user activity events related to the login event of a user who logged in from an external network. It generated Syslog messages with and alert severity and forwarded the messages to a Syslog server. As shown by the flowchart below;

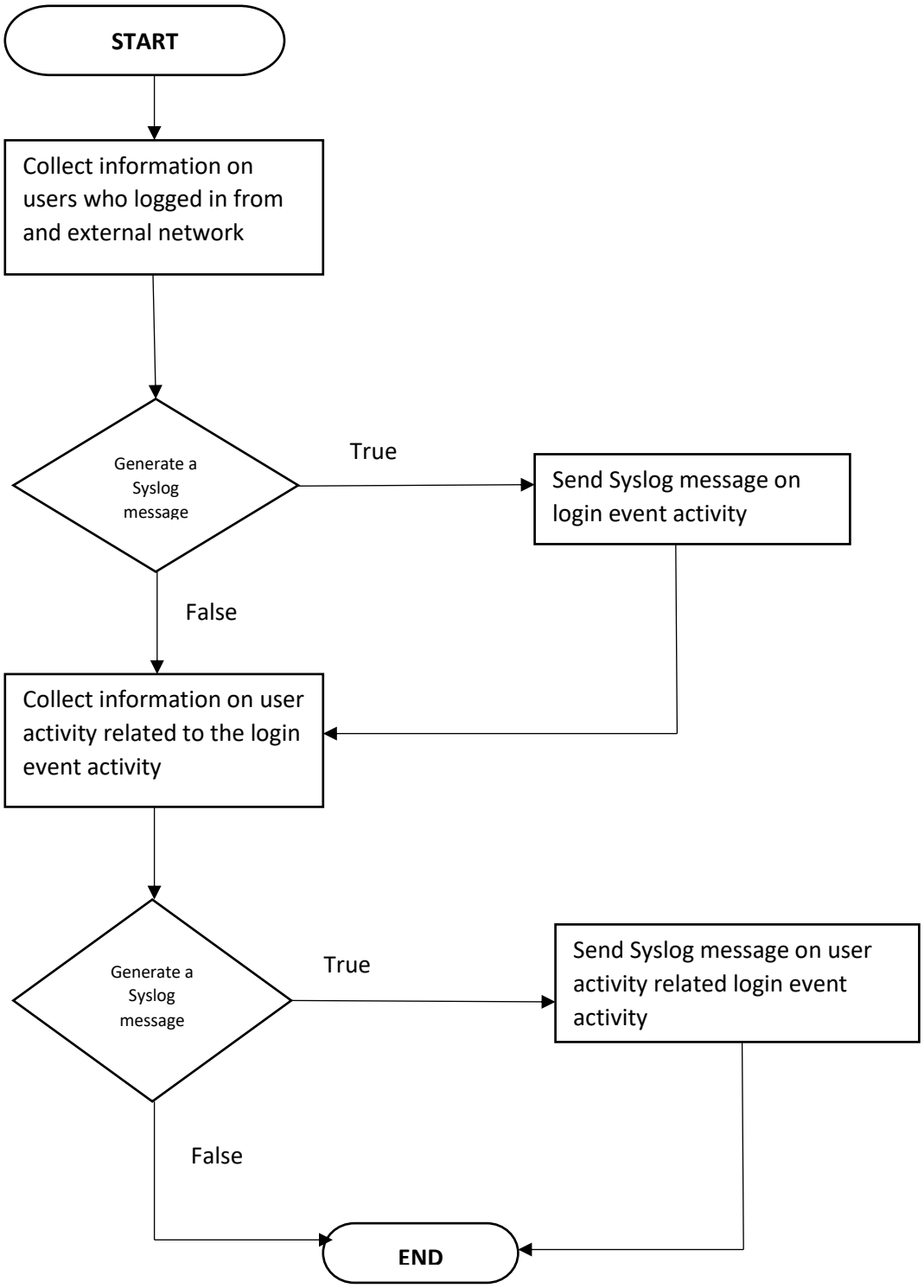


Fig 8: Login from outside the external corporate network rule flowchart

2. Deleted record rule

This rule collected activities of all delete actions on the system. The reason behind the rule was to send an alert any time any record is deleted. The system collected the timestamp, user identification, record identification of the delete action and generate a Syslog message with and

alert severity and forward it to the Syslog server. This rule was also succeeded with a the sub rule below;

2.1.Extract the delete records sub rule

The second sub rule collected event data related to the previous events. It collected the details of the actual record that was deleted and generated a Syslog log message with a warning severity message that was forwarded to a Syslog server as shown in the flow chart below;

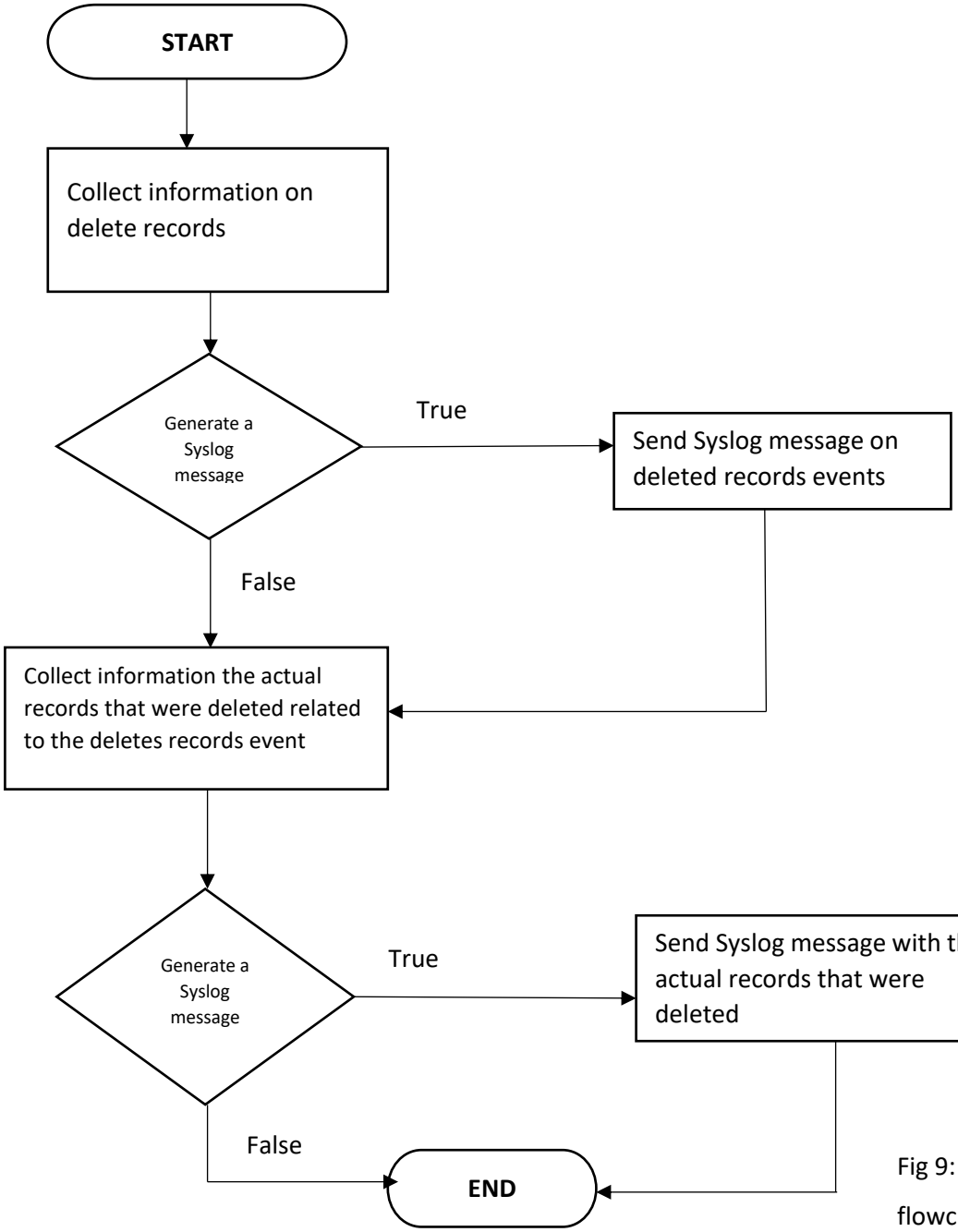


Fig 9: Deleted record rule flowchart

3.1. Collect user login event and activity data rule

This rule collected all user login activity and user activities, then generated syslog messages with an information severity level and provided an option to forward the messages to a Syslog server as shown in the flowchart below;

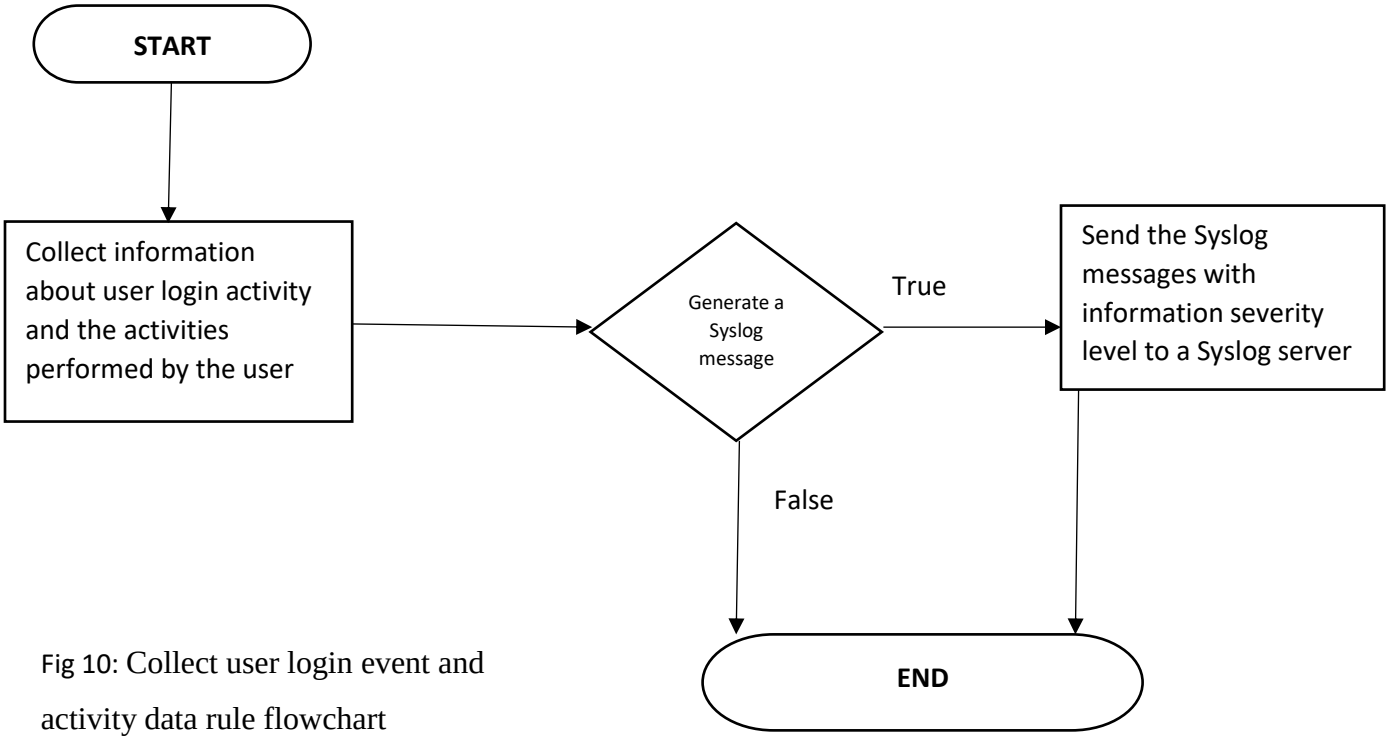


Fig 10: Collect user login event and activity data rule flowchart

3.1.2.2.2 Solution design

Solution architecture

The data collected from the iterative requirements phase was used to develop the architecture of the solution. The architecture designed consisted of three systems;

1. An open source enterprise system
2. An open source security and events management system
3. The prototype system the collected data from the enterprise system and fed it to a SIEM

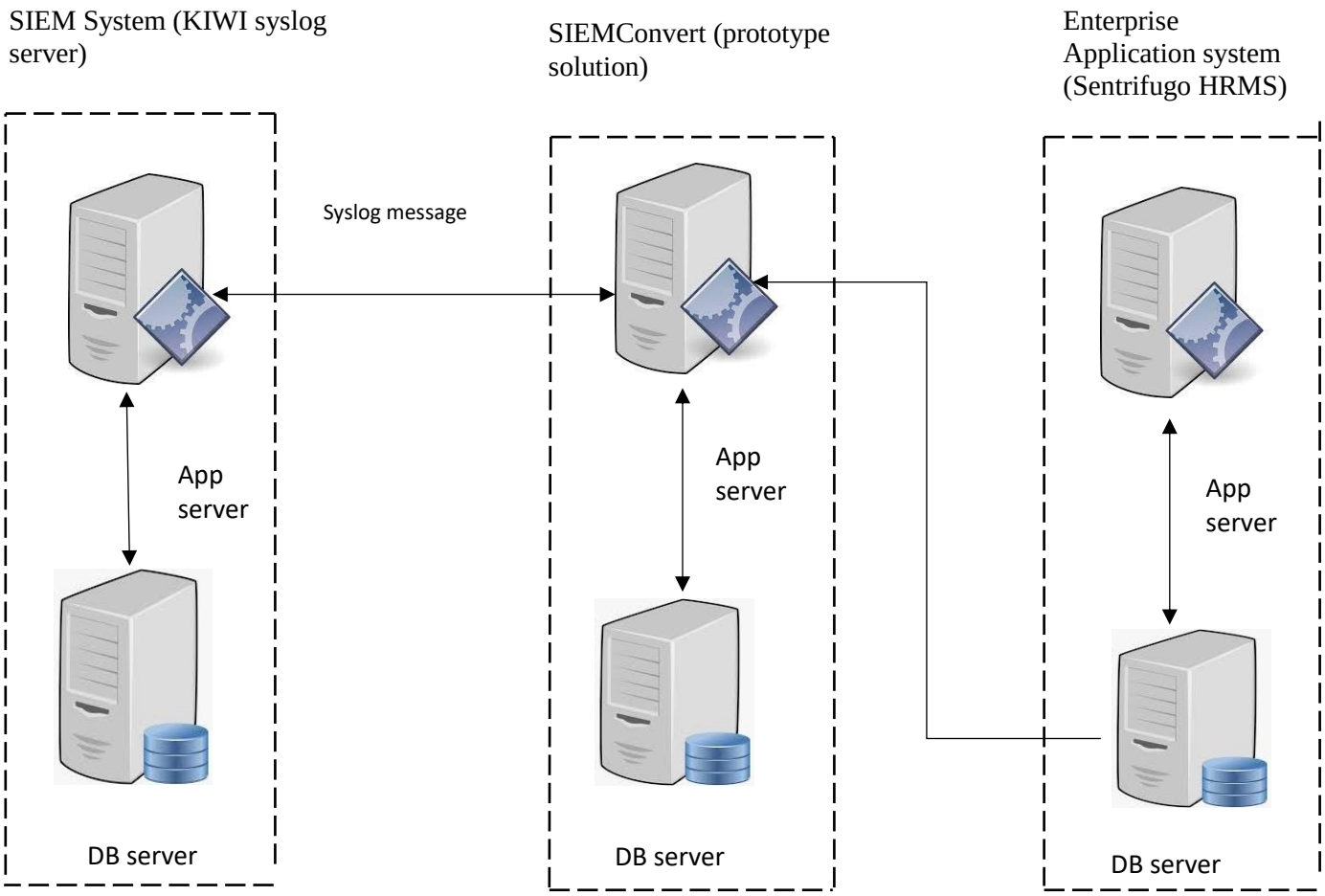


Fig 11. The solution architecture

Open Source enterprise system

The open source system that was used is called Sentrifugo HRMS. Sentrifugo is a free HRMS that is licensed under the GNU General public license V3.0. This copy left license provide

permissions for free commercial use, modification, distribution, patent user and private use (GitHub, 2021). The system was developed using PHP and uses a database deployed on a MySQL database.

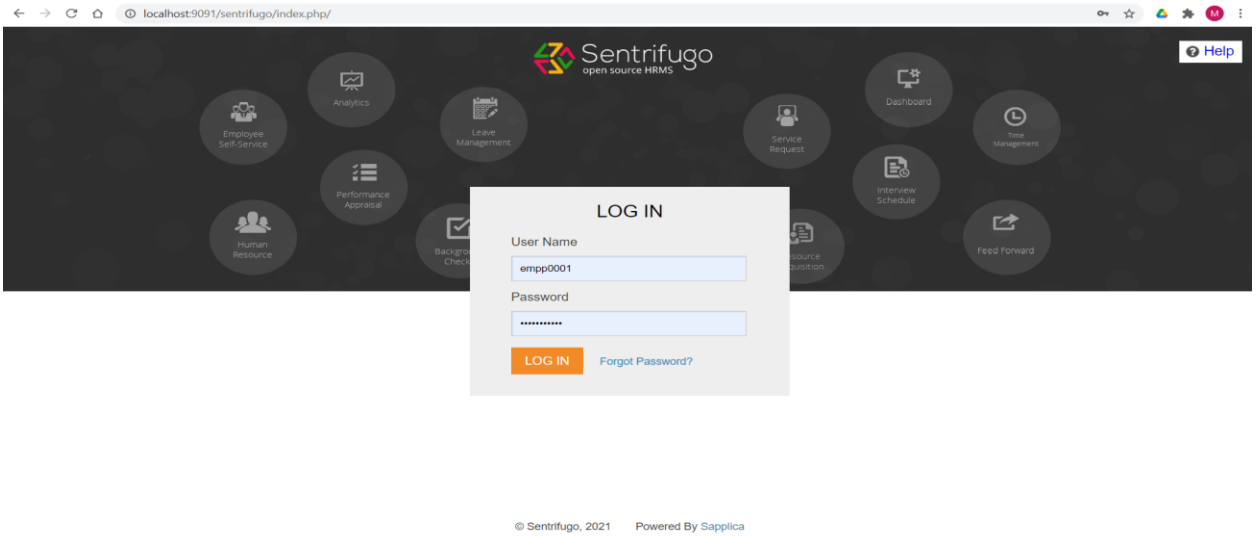


Fig 12: Sentrifugo login page screenshot

SIEM Syslog Server

The Syslog server that was selected in designing the solution architecture was a free Syslog server called KIWI syslog server provided by solar winds (Solarwinds, 2021). The syslog server

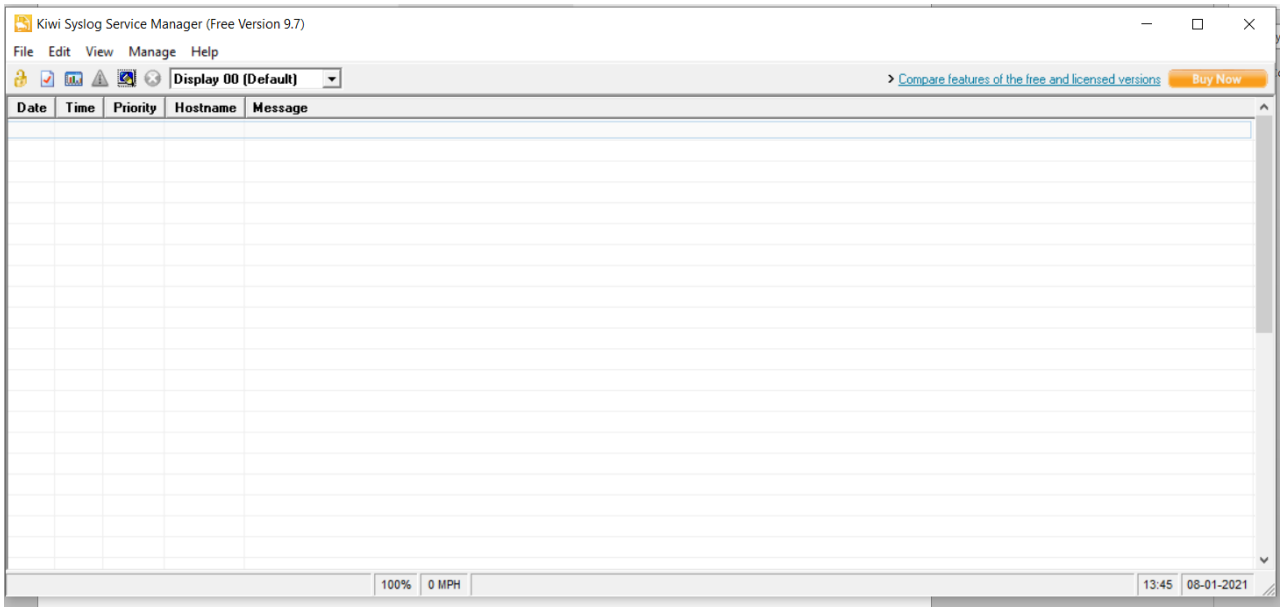


Fig 13: KIWI Syslog Server

has the capability to centrally manage syslog server messages, receive real-time alerts based on syslog messages, automatically respond to syslog messages, store and archive logs for regulatory

compliance, Schedule the generation of syslog reports via email View syslog data anywhere with secure web access.

The prototype system (SIEMConvert)

The third system is the prototype system called SIEMConvert that was developed to be able to process the event data from the enterprise system (Sentrifugo) and send the processed data to the Syslog server (KIWI Syslog server).

Interfaces

There are two interfaces between the three systems. The interface between the Prototype System (SIEM Convert) and the Enterprise system was using a direct to sentrifugo database. SIEM Convert had access and was able to read data from sentrifugo database. The second interface was between SEIMConvert and the Syslog server. The interface involved exchange is syslog message defined by Syslog protocol (Gerhards, Internet Engineering Task Force, 2018).

User Interface Design

The user interface design based on the functions that the IT sec specialist need to trigger in order the prototype to perform the required function. The user journey show below show the functions available to the IT sec specialist.

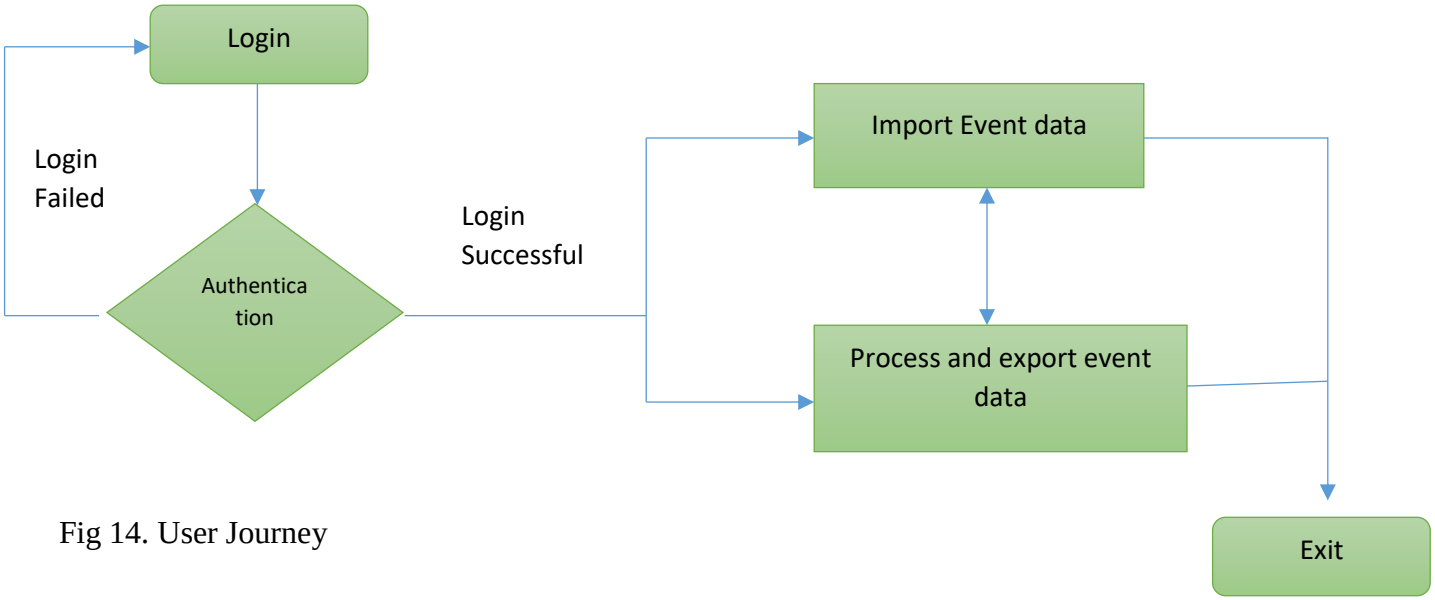


Fig 14. User Journey

The prototype system has two user interfaces;

1. The login Screen

   The Login screen has fields to key in credentials to provide access to the system as shown below;

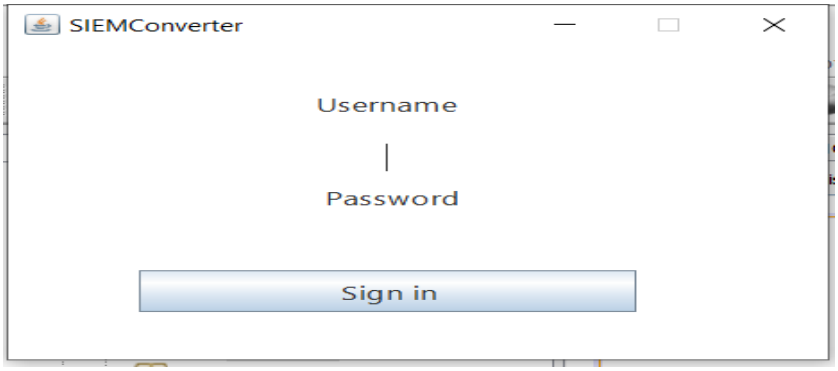


Fig 15. Login Page

2. The main screen

   The main page has two tabs, one tab has menus to import data into the system and the second tab has options to process and export data as shown below;

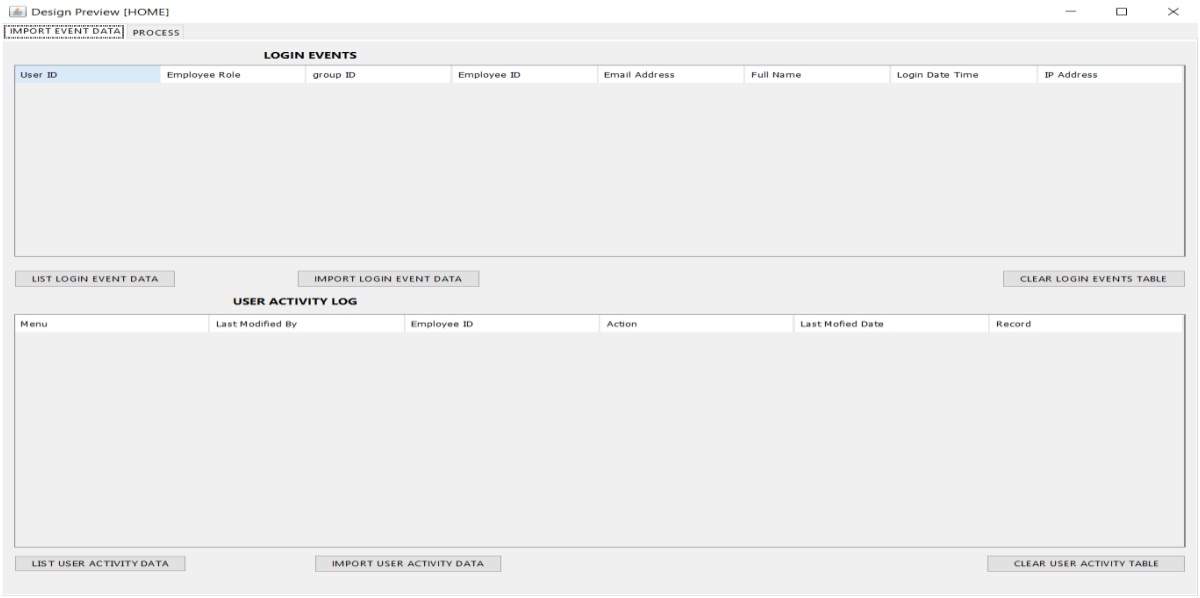
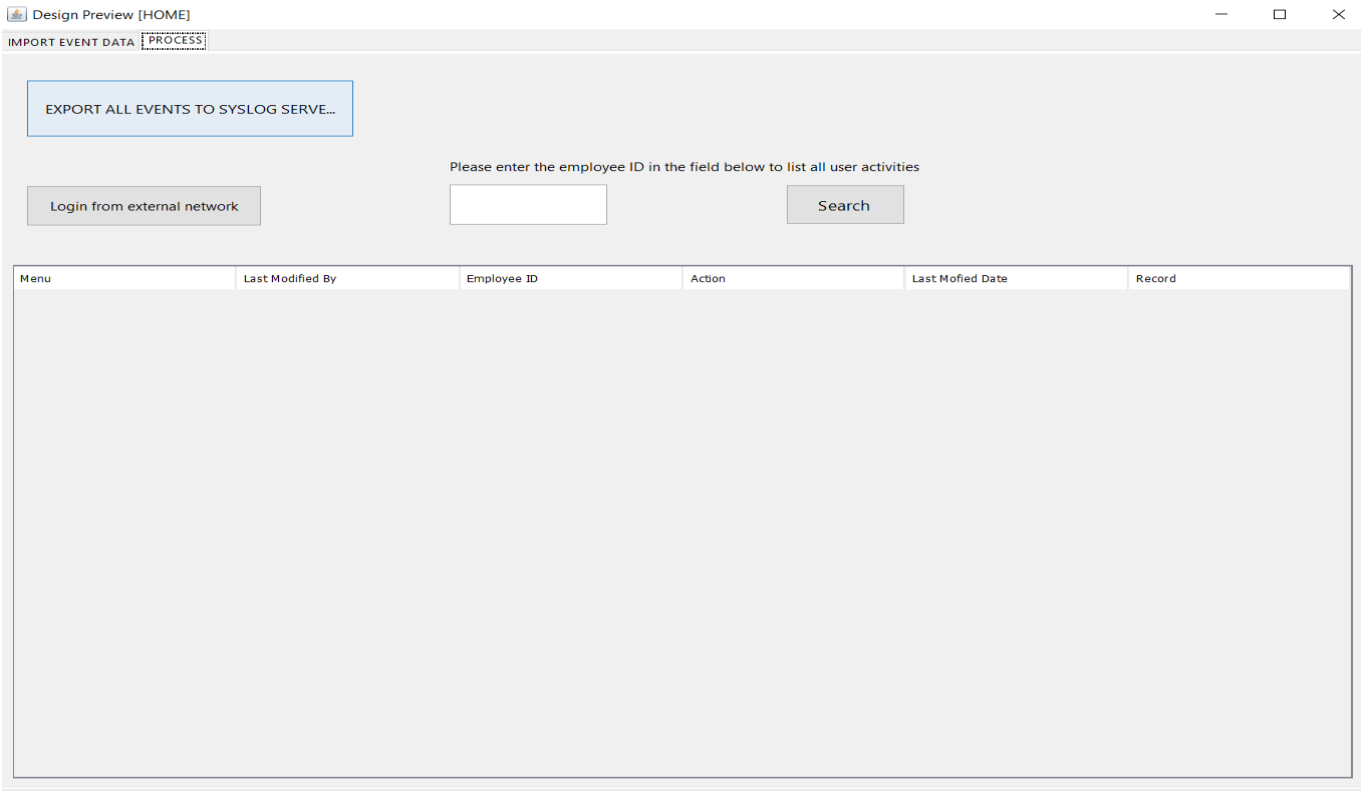


Fig 16. Import Tab

Fig 17. Export tab



Data Flow diagram

The data flow in the architecture was from the enterprise system to the prototype solution which then flowed to the Syslog server as shown below in details

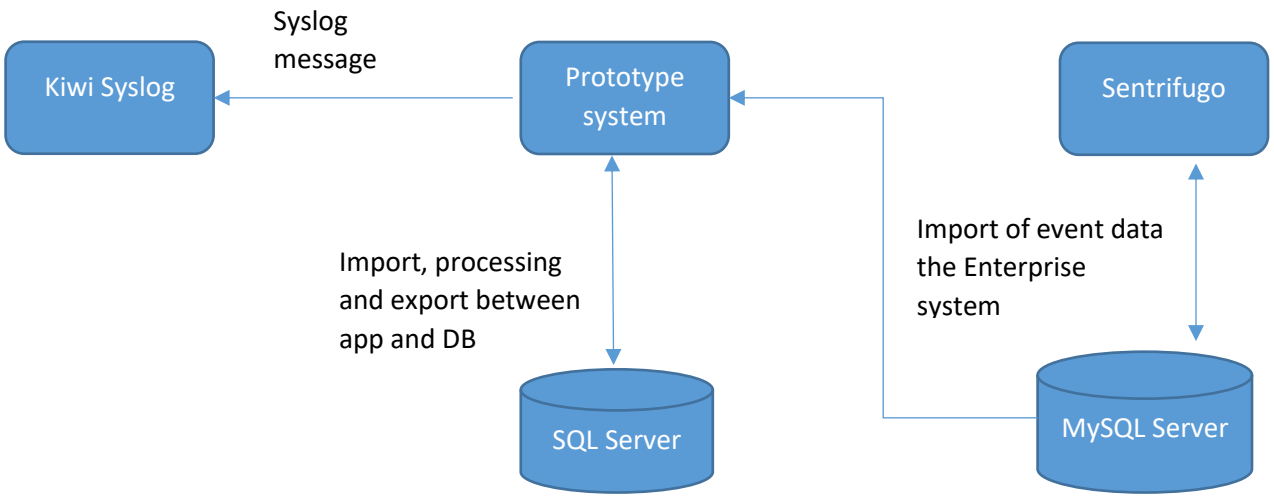


Fig 18. Dataflow diagram

Data design

This is how the data was being held in the database, the system was developed using MS SQL server database to hold the data during importation, processing and exportation of security event data.

Table Structure:

Table 6. Users Table definition

COLUMN_NAME	DATA_TYPE
USER_ID	int identity
USER_NAME	varchar
FULL_NAMES	varchar
USER_PASSWORD	varchar
D_UPDATE	datetime

COLUMN_NAME	DATA_TYPE
Userid	numeric
emprole	varchar
group_id	numeric
employeeId	varchar
emailaddress	varchar
userfullnames	varchar
logindatetime	datetime
empipaddress	varchar
status	varchar

Table 7. UserLoginLog Table

COLUMN_NAME	TYPE_NAME
ID	numeric() identity
menu	varchar
last_modified_by	varchar
last_modified_date	datetime
employeeid	varchar
action	varchar
record	varchar

Table 8. UserActivitylog table definition

3.1.2.3 Implementation

This phase involved the development of the prototype application. The system was developed by following the documentation developed in the analysis and design phase of the methodology. The prototype system was developed using java programing language in NetBeans integrated development environment and Microsoft SQL server for database,

3.1.2.4 Deployment

The system solution architecture was deployed a local PC each with their own configuration. Sentrifugo enterprise system was deployed on an apache server running in the local computer connected to an instance of MySQL server running on the local computer. The service was set on localhost on port 9091.

KIWI Syslog server was also deployed on the local computer running on localhost port 514.

The prototype system was also deployed on the local computer with access to the enterprise system Database through port 3306 and broadcasting message to syslog server on port 514.

3.1.2.5 Testing

The testing process involved the following level of testing;

1. Unit testing
2. Integration testing
3. System testing
4. Acceptance testing

The order of testing was unit testing, integration testing, system testing, the acceptance testing,

- Unit testing; this was the first tests done to ensure that prototype systems units work well. The first unit was the application login to ensure the user were able to login to the application. The second unit tests were the functions required to import and process data from the enterprise system, The third unit test was the functions implemented to create the send the syslog messages
- Integrations testing; Integration testing was done to test the two integration points that were connected to the prototype solution. The first one was between the prototype solution and enterprise solution to ensure that the prototype system was able to collect

valid data, the second part was testing the transmission of syslog messages between the prototype system and the syslog server.

- Systems testing was done to ensure that the system was fully functional. This included checking that all the functions and integrations done are all working.
- The final acceptance test was done confirm that the system met the requirements defined in the requirement phase.

3.1.2.6 Evaluation

An evaluation was done at the end of each sprint to determine if the goal and objective of the sprint were met. This assisted in deciding the next step of the development process.

3.2 Results

The system was able to successfully generate Syslog messages from the relevant events and send them to a Syslog server. Below are screenshots of a record delete event from the enterprise system, the prototype system (SIEM Convert) and the Syslog server.

The screenshot below shows a screenshot of the enterprise system activity log showing that activities that users have performed on the system. Please pay attention to the second record regarding a business unit delete event. We will track this event from the enterprise system all the way to the Syslog server to demonstrate how it worked.

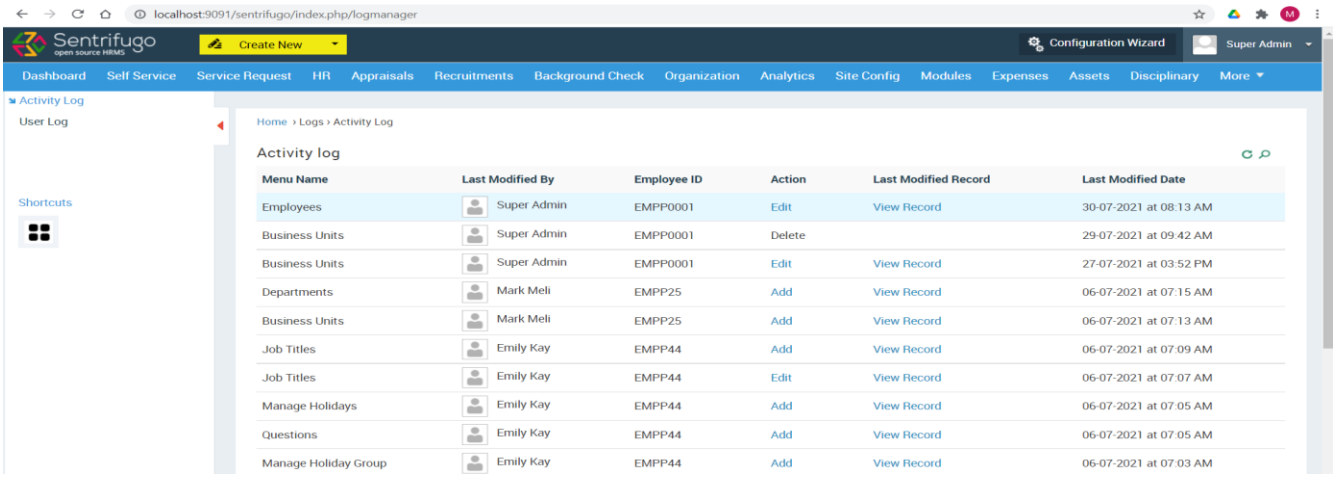


Fig 19: Sentrifugo delete event activity

Below is a screenshot showing the same delete event processed by the prototype system and sent to the Syslog server

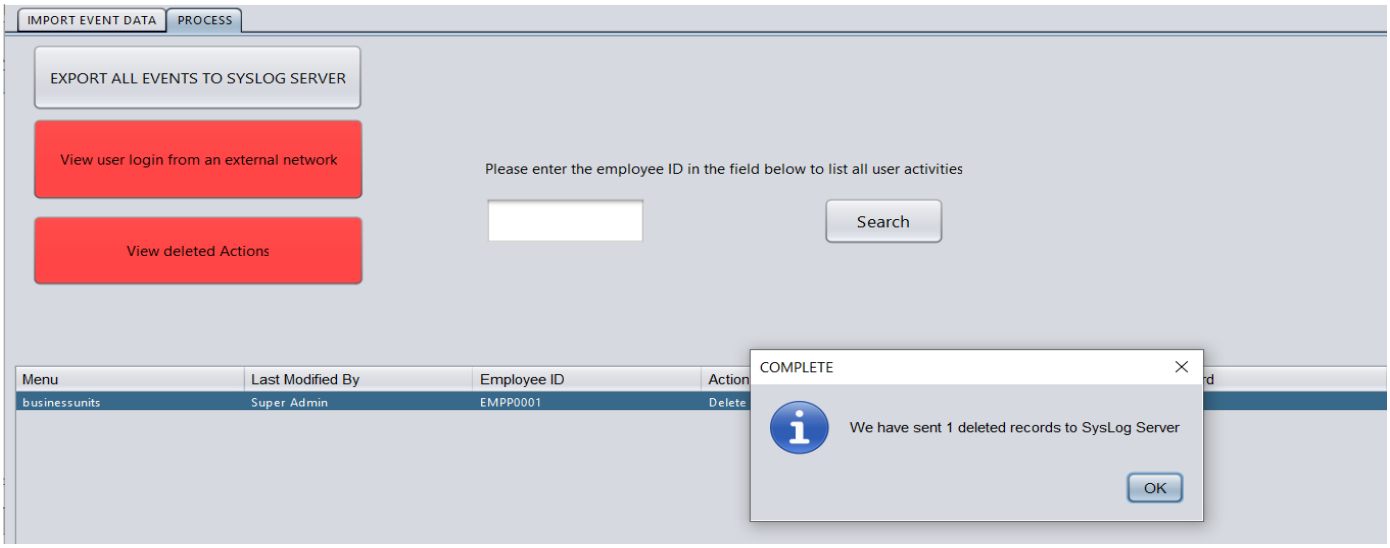


Figure 20: SIEM Convert (Prototype system)

Below is a screenshot of from the Syslog server with the Syslog message received from the prototype system

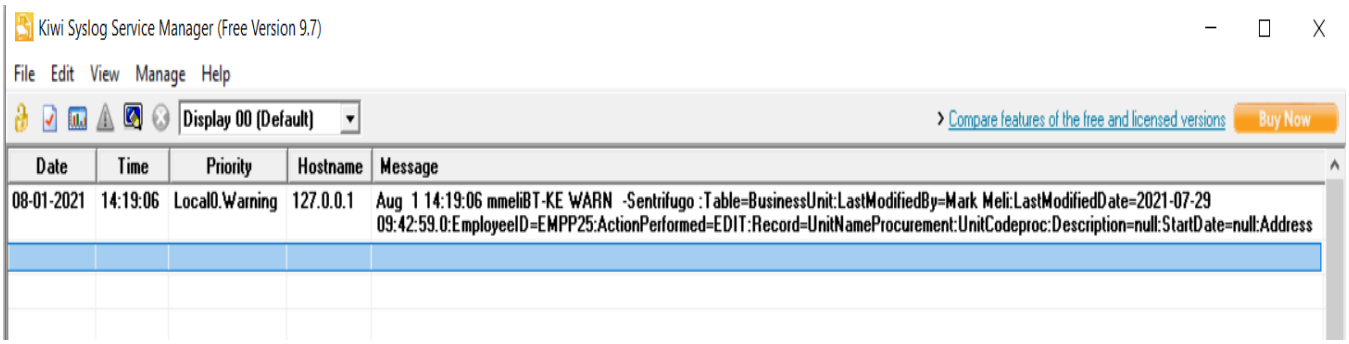


Figure 21: KIWI Syslog Server Message

3.3 Sources of data and relevance of data to the problem

The data used during development and testing of the system was generated by creating dummy actions and configuration in the enterprise system to simulate normal user activities. The simulations created security event information that similar to when the system is being used

4. CONCLUSION AND RECOMMENDATIONS

4.1 Conclusion

From this project, it can be concluded that security information generated by custom applications can be easily be availed and accessed by IT security specialists. The prototype was able to bridge the gap by providing the security information to the SIEM which helps in improving the security posture of an organization

4.2 Key achievements

The study was able to achieve the laid out objectives. The overall objective of this research was to create a prototype system that collects and analyzes custom event data from an enterprise application and feed the data to a SIEM to increase the efficiency in security incident management. This was achieved through development of a java application that was able to collect the event information from and enterprise system, process the data and send it to a SIEM using Syslog protocol.

4.3 Limitation of the study

During the study there were some challenges that were encountered. These challenges are;

It was challenging getting scholarly work done on integrations done between SIEM system and custom applications. This is largely because there`s no unified standard defined to collect security event information that developers can use when designing their system. Right now they have the freedom do as they please. This makes it a challenge when security specialist need to investigate and incident related to the system leaving them at the mercy of the developer.

The other challenge is SIEM systems have rapidly evolved in the last few years with vendors trying to outdo each other. Getting a system that works is a challenge due to their different implementation architecture and proprietary technology that are not interoperable. The future looks great since there are institutions such as open cybersecurity alliance (OCA) who are trying to simplify integrations between cybersecurity products by providing an ecosystem where system can inter operate without the need for customization (open, 2021).

REFERENCES

Feng, C., Wu, S., & Liu, N. (2017). A user-centric machine learning framework for cyber security operations center. *2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China*, 173-175.

Gailey, S. (2020, January 19). *A Brief History of SIEM.* Retrieved from Cybersecurity Magazine: https://cybersecurity-magazine.com/a-brief-history-of-siem/

Gerhards, R. (2018, December 20). *Internet Engineering Task Force.* Retrieved from The syslog protocol: https://datatracker.ietf.org/doc/rfc5424

Gerhards, R. (2021, June 25). *The Syslog Protocol*. Retrieved from Internet Engineering Task Force: https://datatracker.ietf.org/doc/html/rfc5424

GitHub. (2021, 06 10). *github.com.* Retrieved from Sentrifugo: https://github.com/sapplica/sentrifugo/blob/master/LICENSE.txt

Kavanagh, K. M., & Nicolett, M. (2014). Magic Quadrant for Security Information and. *Gartner Research Note G00212454*.

Nicolett, M., & Williams, A. T. (2005). *Improve IT Security With Vulnerability Management.* Gartner.

open, O. (2021, 06 26). *Open Cyber Security allaince*. Retrieved from https://opencybersecurityalliance.org/

Pathak, P., Goldstein, J., & Horaist, L. (2020, February 24). The Past, Present and Future of Security Information and Event Management (SIEM).

Pathi, N. K. (2015, Dec 14). *SIEM Architecture.* Retrieved from https://www.slideshare.net/pathinishanth/siem-architecture

Rekaby, A., & Soliman, M. (2012). *Towards Intermediate-Agile Model Based On Agile.* Cairo: Helwan University, Cairo, Egypt.

Roschke, S., Cheng, F., & Meinel, C. (2011). A New Alrt Correlation Algorithm Based on Attack Graph. *4th Internation Confrence on computational Intelligence in Security for Information Systems* . Terremolinos, Spain: Hasso Plattner Institure.

Rosencrance, L. (2021, 4 5). *security information and event management (SIEM).* Retrieved from TechTarget: https://searchsecurity.techtarget.com/definition/security-information-and-event-management-SIEM

Solarwinds. (2021, 05 25). *Solarwinds.* Retrieved from KIWI syslog server: https://www.solarwinds.com/kiwi-syslog-server

TOUHID. (2019, May 27). *Types of computer security*. Retrieved from CYBER THREAT PORTAL: https://cyberthreatportal.com/types-of-computer-security/

APPENDICES

APPENDIX A: DEPLOYMENT STEPS

Step 1: Download and install Sentrifugo .

The system is available on https://sourceforge.net/projects/sentrifugo/

Step 2: Download KIWI Syslog server

It application is available on https://www.kiwisyslog.com/kiwi-syslog-server

Step 3: Install the SQL server database engine

The database engine is available on https://www.microsoft.com/en-us/sql-server/sql-server-downloads

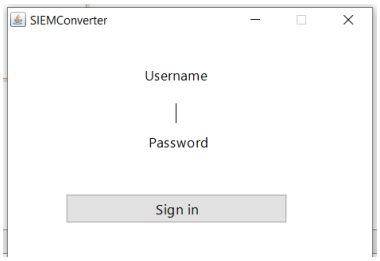
Step 4: Install SIEM Convert (Prototype system)

The system can be installed or run an eclipse project available below

APPENDIX B: USER MANUAL

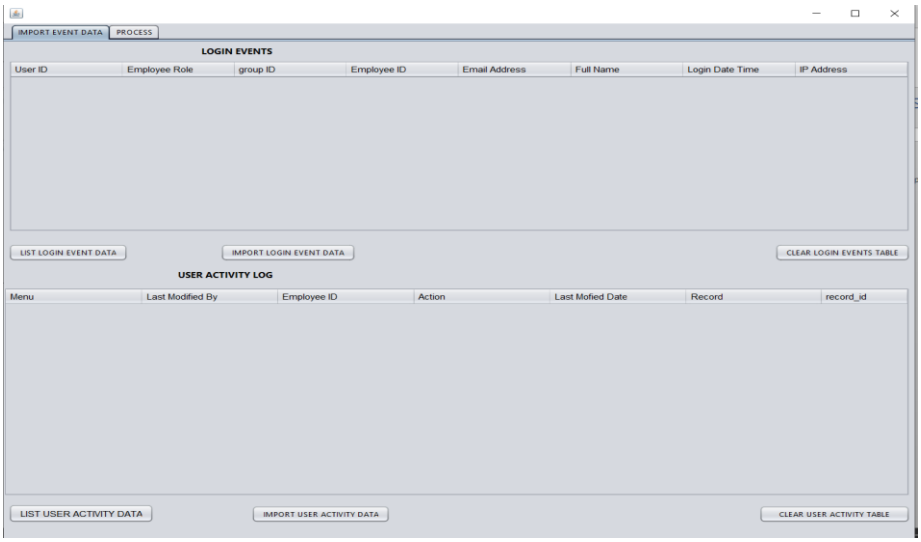
Step I. Login to the application

Default Username= Meli Password=meli



Step 2. Click on Import Event Data tab

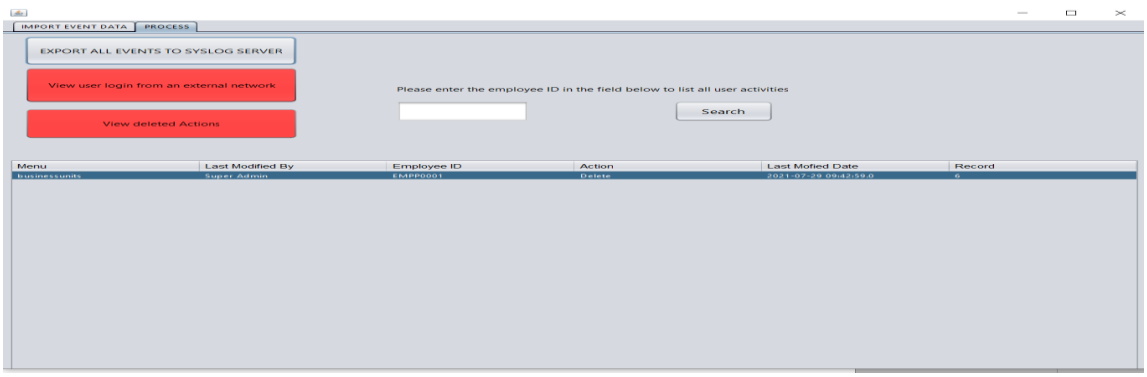
This tab has options to import data from the enterprise system and processing the data



Step 3: Click on Process tab

The process tab has options to export all events to Syslog server and two buttons to apply the two correlation rules. There`s also an option to search all events executed by a particular user.

After Clicking on one of the buttons in red to run the correlation rule, you have to click on the record in the table to effect the sub rules



APPENDIX B: SAMPLE CODE

Main Application processing code

```
package siemconverter;


import java.awt.Point;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

import siemconverter.message.IMessageTransmitter;

import siemconverter.message.kiwi.KiwiMessageTransmitterImpl;

import siemconverter.message.kiwi.KiwiMessageTransmitterImpl_1;


/**

 *

 * @author mark.meli

 */

public class HOME extends javax.swing.JFrame {

     DBconnection consql = null;

     ResultSet rs=null;

     DBconnectionMysql conMysql=null;

   private int checkact=0;

   /**

    * Creates new form HOME

    */

   public HOME() {


       initComponents();

       conMysql = new DBconnectionMysql();

       consql = new DBconnection();

   }


   /**

    * This method is called from within the constructor to initialize the form.

    * WARNING: Do NOT modify this code. The content of this method is always
```

```
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    process_tab = new javax.swing.JTabbedPane();
    jPanel1 = new javax.swing.JPanel();
    ImportSecEventbtn = new javax.swing.JButton();
    list_event_btn = new javax.swing.JButton();
    SecEventScrollPane = new javax.swing.JScrollPane();
    SecEventTbl = new javax.swing.JTable();
    jLabel1 = new javax.swing.JLabel();
    UserActivityScrollPane = new javax.swing.JScrollPane();
    UserActivityTbl = new javax.swing.JTable();
    jLabel2 = new javax.swing.JLabel();
    ImportUserActivityEventbtn = new javax.swing.JButton();
    ListUserActivityBtn = new javax.swing.JButton();
    ClearloginEventtbkbtn = new javax.swing.JButton();
    clearactivitytblbtn = new javax.swing.JButton();
    jPanel2 = new javax.swing.JPanel();
    exportSysLogbtn = new javax.swing.JButton();
    ProcessUserActivityScrollPane = new javax.swing.JScrollPane();
    ProcessUserActivityTbl1 = new javax.swing.JTable();
    Externalloginbtn = new javax.swing.JButton();
    Searchempactvtytxtbtn = new javax.swing.JButton();
    Searchempactvtytxtbox = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();
    Deletedbtn = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    process_tab.setFont(new java.awt.Font("Segoe UI", 0, 10)); // NOI18N

    ImportSecEventbtn.setFont(new java.awt.Font("Segoe UI", 0, 10)); // NOI18N
    ImportSecEventbtn.setText("IMPORT LOGIN EVENT DATA");
    ImportSecEventbtn.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        ImportSecEventbtnActionPerformed(evt);
    }
});

list_event_btn.setFont(new java.awt.Font("Segoe UI", 0, 10)); // NOI18N
list_event_btn.setText("LIST LOGIN EVENT DATA");
list_event_btn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        list_event_btnActionPerformed(evt);
    }
});

SecEventTbl.setFont(new java.awt.Font("Segoe UI", 0, 10)); // NOI18N
SecEventTbl.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {
        "User ID", "Employee Role", "group ID", "Employee ID", "Email Address", "Full Name", "Login Date Time", "IP Address"
    }
) {
    boolean[] canEdit = new boolean [] {
        false, false, false, false, false, false, false, false
    };

    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return canEdit [columnIndex];
    }
});
SecEventScrollPane.setViewportView(SecEventTbl);

jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel1.setText("LOGIN EVENTS");

UserActivityTbl.setFont(new java.awt.Font("Segoe UI", 0, 10)); // NOI18N
UserActivityTbl.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
```

```
    },
    new String [] {
        "Menu", "Last Modified By", "Employee ID", "Action", "Last Mofied Date", "Record", "record_id"
    }
) {
    boolean[] canEdit = new boolean [] {
        false, false, false, false, false, false, false
    };

    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return canEdit [columnIndex];
    }
});
UserActivityScrollPane.setViewportView(UserActivityTbl);
if (UserActivityTbl.getColumnModel().getColumnCount() > 0) {
    UserActivityTbl.getColumnModel().getColumn(6).setPreferredWidth(2);
}

jLabel2.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel2.setText("USER ACTIVITY LOG");

ImportUserActivityEventbtn.setFont(new java.awt.Font("Segoe UI", 0, 10)); // NOI18N
ImportUserActivityEventbtn.setText("IMPORT USER ACTIVITY DATA");
ImportUserActivityEventbtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ImportUserActivityEventbtnActionPerformed(evt);
    }
});

ListUserActivityBtn.setText("LIST USER ACTIVITY DATA");
ListUserActivityBtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ListUserActivityBtnActionPerformed(evt);
    }
});

ClearloginEventtbkbtn.setFont(new java.awt.Font("Segoe UI", 0, 10)); // NOI18N
ClearloginEventtbkbtn.setText("CLEAR LOGIN EVENTS TABLE");
```

```
        ClearloginEventtbkbtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ClearloginEventtbkbtnActionPerformed(evt);
            }
        });

        clearactivitytblbtn.setFont(new java.awt.Font("Segoe UI", 0, 10)); // NOI18N
        clearactivitytblbtn.setText("CLEAR USER ACTIVITY TABLE");
        clearactivitytblbtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                clearactivitytblbtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGroup()
                                .addContainerGap()
                                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                    .addGroup(jPanel1Layout.createSequentialGroup()
                                        .addComponent(list_event_btn)
                                        .addGap(111, 111, 111)
                                        .addComponent(ImportSecEventbtn)
                                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(ClearloginEventtbkbtn))
                                    .addGroup(jPanel1Layout.createSequentialGroup()
                                        .addComponent(ListUserActivityBtn)
                                        .addGap(117, 117, 117)
                                        .addComponent(ImportUserActivityEventbtn)
                                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 442, Short.MAX_VALUE)
                                        .addComponent(clearactivitytblbtn, javax.swing.GroupLayout.PREFERRED_SIZE, 181,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                        .addComponent(SecEventScrollPane)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(237, 237, 237)
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 131,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(209, 209, 209)
                        .addComponent(jLabel2)))
                .addGap(0, 0, Short.MAX_VALUE))
            .addComponent(UserActivityScrollPane, javax.swing.GroupLayout.Alignment.TRAILING))
        .addContainerGap())
    );
    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jLabel1)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(SecEventScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 259,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(list_event_btn)
                .addComponent(ImportSecEventbtn)
                .addComponent(ClearloginEventtbkbtn))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jLabel2)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(UserActivityScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 315,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(ImportUserActivityEventbtn)
                .addComponent(ListUserActivityBtn)
                .addComponent(clearactivitytblbtn))
            .addContainerGap(30, Short.MAX_VALUE))
    );

    process_tab.addTab("IMPORT EVENT DATA", jPanel1);
```

```
exportSysLogbtn.setFont(new java.awt.Font("Segoe UI", 0, 13)); // NOI18N
exportSysLogbtn.setText("EXPORT ALL EVENTS TO SYSLOG SERVER");
exportSysLogbtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        exportSysLogbtnActionPerformed(evt);
    }
});

ProcessUserActivityTbl1.setFont(new java.awt.Font("Segoe UI", 0, 10)); // NOI18N
ProcessUserActivityTbl1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {
        "Menu", "Last Modified By", "Employee ID", "Action", "Last Mofied Date", "Record"
    }
) {
    boolean[] canEdit = new boolean [] {
        false, false, false, false, false, false
    };

    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return canEdit [columnIndex];
    }
});
ProcessUserActivityTbl1.addFocusListener(new java.awt.event.FocusAdapter() {
    public void focusGained(java.awt.event.FocusEvent evt) {
        ProcessUserActivityTbl1FocusGained(evt);
    }
});
ProcessUserActivityTbl1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        ProcessUserActivityTbl1MouseClicked(evt);
    }
});
ProcessUserActivityScrollPane.setViewportView(ProcessUserActivityTbl1);
```

```
Externalloginbtn.setBackground(new java.awt.Color(255, 51, 51));
Externalloginbtn.setFont(new java.awt.Font("Segoe UI", 0, 12)); // NOI18N
Externalloginbtn.setText("View user login from an external network");
Externalloginbtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ExternalloginbtnActionPerformed(evt);
    }
});

Searchempactvtytxtbtn.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
Searchempactvtytxtbtn.setText("Search");
Searchempactvtytxtbtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        SearchempactvtytxtbtnActionPerformed(evt);
    }
});

Searchempactvtytxtbox.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
Searchempactvtytxtbox.setHorizontalAlignment(javax.swing.JTextField.CENTER);

jLabel3.setFont(new java.awt.Font("Segoe UI", 0, 12)); // NOI18N
jLabel3.setText("Please enter the employee ID in the field below to list all user activities");

Deletedbtn.setBackground(new java.awt.Color(255, 51, 51));
Deletedbtn.setFont(new java.awt.Font("Segoe UI", 0, 12)); // NOI18N
Deletedbtn.setText("View deleted Actions");
Deletedbtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        DeletedbtnActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel2Layout.createSequentialGroup()
        .addGap(21, 21, 21)
```

```
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel2Layout.createSequentialGroup()
                    .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(Externalloginbtn, javax.swing.GroupLayout.DEFAULT_SIZE, 261, Short.MAX_VALUE)
                        .addComponent(Deletedbtn, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE,
0, Short.MAX_VALUE))
                    .addGap(94, 94, 94)
                    .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 383, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(jPanel2Layout.createSequentialGroup()
                            .addComponent(Searchempactvtytxtbox, javax.swing.GroupLayout.PREFERRED_SIZE, 126,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(139, 139, 139)
                            .addComponent(Searchempactvtytxtbtn, javax.swing.GroupLayout.PREFERRED_SIZE, 95,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addComponent(exportSysLogbtn))
            .addContainerGap(329, Short.MAX_VALUE))
        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(ProcessUserActivityScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 1066, Short.MAX_VALUE)
                .addContainerGap()))
    );
    jPanel2Layout.setVerticalGroup(
        jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel2Layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(exportSysLogbtn, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(Externalloginbtn, javax.swing.GroupLayout.PREFERRED_SIZE, 72,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(Deletedbtn, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(jPanel2Layout.createSequentialGroup()
                    .addGap(101, 101, 101)
                    .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
                    .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(Searchempactvtytxtbox, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(Searchempactvtytxtbtn, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addContainerGap(528, Short.MAX_VALUE))
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
                    .addContainerGap(269, Short.MAX_VALUE)
                    .addComponent(ProcessUserActivityScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 467,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap()))
        );

        process_tab.addTab("PROCESS", jPanel2);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(process_tab)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(process_tab)
        );

        pack();
    }// </editor-fold>

    private void list_event_btnActionPerformed(java.awt.event.ActionEvent evt) {

        for( int i = SecEventTbl.getModel().getRowCount() - 1; i >= 0; i-- )
        {
            ((DefaultTableModel)SecEventTbl.getModel()).removeRow(i);
        }
        int SecEventCount=0;
    String query="SELECT id,userid,emprole,group_id,employeeId,emailaddress,userfullname,logindatetime+interval 3 HOUR as
logindatetime,empipaddress FROM SENTRIFUGO.main_userloginlog order by logindatetime desc;";
    //String Header = "id\tUserid\temprole\tgroup_id\temployeeId\temailaddress\tuserfullname\tlogindatetime\tempipaddress\n=====";
```

```
   //SecEventTbl.append(Header+"\n");
   rs=conMysql.retrieve(query);
    try
      {
      while (rs.next())
      {
        // String id=rs.getString("id");
         String userid=rs.getString("userid");
         String Emprole=rs.getString("emprole");
         String Groupid=rs.getString("group_id");
         String employeeid=rs.getString("employeeId");
         String Emailaddress=rs.getString("emailaddress");
         String userfullname=rs.getString("userfullname");
         String logindatetime=rs.getString("logindatetime");
         String Empipaddress=rs.getString("empipaddress");



         ((DefaultTableModel)SecEventTbl.getModel()).addRow(new
Object[]{userid,Emprole,Groupid,employeeid,Emailaddress,userfullname,logindatetime,Empipaddress});
         SecEventCount++;
      }
      JOptionPane.showMessageDialog(null,"We have found "+SecEventCount+" Events","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);
      }
      catch(SQLException w)
      {
         JOptionPane.showMessageDialog(null,w.getMessage()," NO RECORD FOUND ",JOptionPane.ERROR_MESSAGE);
      }
   }


   private void ImportSecEventbtnActionPerformed(java.awt.event.ActionEvent evt) {


      int SecEventCount=0;
     for( int i = SecEventTbl.getModel().getRowCount() - 1; i >= 0; i-- )
      {


         String userid =SecEventTbl.getModel().getValueAt(i,0).toString();
         String emprole =SecEventTbl.getModel().getValueAt(i,1).toString();
         String groupid =SecEventTbl.getModel().getValueAt(i,2).toString();
```

```
        String employeeid =SecEventTbl.getModel().getValueAt(i,3).toString();

        String emailaddress =SecEventTbl.getModel().getValueAt(i,4).toString();

        String userfullname = SecEventTbl.getModel().getValueAt(i,5).toString();

        String logindatetime =SecEventTbl.getModel().getValueAt(i,6).toString();

        String employeeipaddress =SecEventTbl.getModel().getValueAt(i,7).toString();



        String SecUserLoginLog = "INSERT INTO userloginlog
(Userid,emprole,group_id,employeeId,emailaddress,userfullnames,logindatetime,empipaddress,status) VALUES
('"+userid+"','"+emprole+"','"+groupid+"','"+employeeid+"','"+emailaddress+"','"+userfullname+"','"+logindatetime+"','"+employeeipaddress+"','
Login Successfull')";

        consql.update(SecUserLoginLog);

        SecEventCount++;


      }

     JOptionPane.showMessageDialog(null,"You have imported "+SecEventCount+" security Events","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);


  }


  private void exportSysLogbtnActionPerformed(java.awt.event.ActionEvent evt) {


  int SecEventCount=0;

  String querylogin= "SELECT * FROM [SIEM_CONVERTER].[dbo].[userloginlog] ORDER BY logindatetime DESC";

  String queryactivity="SELECT * FROM [SIEM_CONVERTER].[DBO].[useractivitylog] ORDER BY last_modified_date DESC";

  rs=consql.retrieve(querylogin);


  try
    {
    while (rs.next())
    {
      // String id=rs.getString("id");
        String userid=rs.getString("userid");
        String Emprole=rs.getString("emprole");
        String Groupid=rs.getString("group_id");
        String employeeid=rs.getString("employeeId");
        String Emailaddress=rs.getString("emailaddress");
        String userfullname=rs.getString("userfullnames");
        String logindatetime=rs.getString("logindatetime");
        String Empipaddress=rs.getString("empipaddress");
```

```
            String status=rs.getString("status");


             final IMessageTransmitter kiwiMessageTransmitter = new KiwiMessageTransmitterImpl();

kiwiMessageTransmitter.send(status+":UserIPaddress="+Empipaddress+":LoginDateTime="+logindatetime+"EmployeeID="+employeeid+":Us
erID="+userid+":"+userfullname+":EmployeeRole="+Emprole+":UserGroupID="+Groupid+":EmailAddress="+Emailaddress);


            SecEventCount++;
        }
        JOptionPane.showMessageDialog(null,"We have sent "+SecEventCount+" Login Events to SysLog Server","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);
        }
        catch(SQLException w)
        {
            JOptionPane.showMessageDialog(null,w.getMessage()," NO RECORD FOUND ",JOptionPane.ERROR_MESSAGE);
        }


    rs=consql.retrieve(queryactivity);
    try
        {
        while (rs.next())
        {
            String menu=rs.getString("menu");
            String last_modified_by=rs.getString("last_modified_by");
            String last_modified_date=rs.getString("last_modified_date");
            String employeeid=rs.getString("employeeId");
            String action=rs.getString("action");
            String record=rs.getString("record");


             final IMessageTransmitter kiwiMessageTransmitter = new KiwiMessageTransmitterImpl();

kiwiMessageTransmitter.send("Table="+menu+":LastModifiedBy="+last_modified_by+":LastModifiedDate="+last_modified_date+":Employee
ID="+employeeid+":ActionPerformed="+action+":Record="+record);


            SecEventCount++;
        }
        JOptionPane.showMessageDialog(null,"We have sent "+SecEventCount+" User Activity Events to SysLog Server","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);
        }
        catch(SQLException w)
```

```
    {
        JOptionPane.showMessageDialog(null,w.getMessage()," NO RECORD FOUND ",JOptionPane.ERROR_MESSAGE);
    }
}

private void ImportUserActivityEventbtnActionPerformed(java.awt.event.ActionEvent evt) {

    int SecEventCount=0;
    for( int i = UserActivityTbl.getModel().getRowCount() - 1; i >= 0; i-- )
    {

        String menu =UserActivityTbl.getModel().getValueAt(i,0).toString();
        String last_modified_by =UserActivityTbl.getModel().getValueAt(i,1).toString();
        String last_modified_date =UserActivityTbl.getModel().getValueAt(i,4).toString();
        String employeeid =UserActivityTbl.getModel().getValueAt(i,2).toString();
        String action =UserActivityTbl.getModel().getValueAt(i,3).toString();
        String record = UserActivityTbl.getModel().getValueAt(i,5).toString();
        String id = UserActivityTbl.getModel().getValueAt(i,6).toString();



        String SecUserLoginLog = "INSERT INTO [useractivitylog]
(menu,last_modified_by,last_modified_date,employeeid,action,record,recordid) VALUES
('"+menu+"','"+last_modified_by+"','"+last_modified_date+"','"+employeeid+"','"+action+"','"+record+"','"+id+"')";
        consql.update(SecUserLoginLog);
        SecEventCount++;

    }
    JOptionPane.showMessageDialog(null,"You have imported "+SecEventCount+" security Events","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);

}

private void ListUserActivityBtnActionPerformed(java.awt.event.ActionEvent evt) {

    for( int i = UserActivityTbl.getModel().getRowCount() - 1; i >= 0; i-- )
    {
        ((DefaultTableModel)UserActivityTbl.getModel()).removeRow(i);
    }
```

```
        int SecEventCount=0;

    String query="SELECT * FROM SIEMCONVERTBU UNION SELECT * FROM SIEMCONVERTDEPT union SELECT * FROM
SIEMCONVERTJT UNION SELECT * FROM SIEMCONVERTPS UNION SELECT * FROM SIEMCONVERTEREMPS ORDER BY
LASTMODIFIEDDATE DESC";

    rs=conMysql.retrieve(query);

      try

        {

        while (rs.next())

        {


            String menu=rs.getString(1);

            String modifiedby=rs.getString(2);

            String employeeid=rs.getString(3);

            String id=rs.getString(4);

            String action=rs.getString(5);

            String lastdatemodified=rs.getString(6);

            String record=rs.getString(7);







            ((DefaultTableModel)UserActivityTbl.getModel()).addRow(new
Object[]{menu,modifiedby,employeeid,action,lastdatemodified,record,id});

            SecEventCount++;

        }

        JOptionPane.showMessageDialog(null,"We have found "+SecEventCount+" Events","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);

        }

        catch(SQLException w)

        {

            JOptionPane.showMessageDialog(null,w.getMessage()," NO RECORD FOUND ",JOptionPane.ERROR_MESSAGE);

        }

    }


    private void ClearloginEventtbkbtnActionPerformed(java.awt.event.ActionEvent evt) {

      for( int i = SecEventTbl.getModel().getRowCount() - 1; i >= 0; i-- )

        {

            ((DefaultTableModel)SecEventTbl.getModel()).removeRow(i);

        }

    }
```

```
private void clearactivitytblbtnActionPerformed(java.awt.event.ActionEvent evt) {

    for( int i = UserActivityTbl.getModel().getRowCount() - 1; i >= 0; i-- )
    {
       ((DefaultTableModel)UserActivityTbl.getModel()).removeRow(i);
    }
}

private void ExternalloginbtnActionPerformed(java.awt.event.ActionEvent evt) {

  SimpleDateFormat formatter= new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
  Date date= new Date();

    for( int i = ProcessUserActivityTbl1.getModel().getRowCount() - 1; i >= 0; i-- )
    {
       ((DefaultTableModel)ProcessUserActivityTbl1.getModel()).removeRow(i);
    }

     for( int i = ProcessUserActivityTbl1.getModel().getRowCount() - 1; i >= 0; i-- )
    {
       ((DefaultTableModel)ProcessUserActivityTbl1.getModel()).removeRow(i);
    }
    int SecEventCount=0;
String query="SELECT * FROM SIEMWRONGSUBNET  ";
rs=conMysql.retrieve(query);
  try
    {
    while (rs.next())
    {

       String menu=rs.getString(1);
       String modifiedby=rs.getString(2);
       String employeeid=rs.getString(3);
       String action=rs.getString(4);
       Date lastdatemodified=rs.getTimestamp(5);
       String record=rs.getString(6);
```

```
        ((DefaultTableModel)ProcessUserActivityTbl1.getModel()).addRow(new
Object[]{menu,modifiedby,employeeid,action,lastdatemodified,record});

        SecEventCount++;

      }

      JOptionPane.showMessageDialog(null,"We have found "+SecEventCount+" Events","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);

      }

      catch(SQLException w)

      {

        JOptionPane.showMessageDialog(null,w.getMessage()," NO RECORD FOUND ",JOptionPane.ERROR_MESSAGE);

      }

    if(SecEventCount!=0)

      {

        int reply=JOptionPane.showConfirmDialog(null,"Do you want to sent this report to Syslog Server","",JOptionPane.YES_NO_OPTION);

        if (reply==JOptionPane.YES_OPTION)

        {


          for ( int i = ProcessUserActivityTbl1.getModel().getRowCount() - 1; i >= 0; i-- )

          {

            String menu =ProcessUserActivityTbl1.getModel().getValueAt(i,0).toString();

            String last_modified_by =ProcessUserActivityTbl1.getModel().getValueAt(i,1).toString();

            String last_modified_date =ProcessUserActivityTbl1.getModel().getValueAt(i,4).toString();

            String employeeid =ProcessUserActivityTbl1.getModel().getValueAt(i,2).toString();

            String action =ProcessUserActivityTbl1.getModel().getValueAt(i,3).toString();

            String record = ProcessUserActivityTbl1.getModel().getValueAt(i,5).toString();


        final IMessageTransmitter kiwiMessageTransmitter = new KiwiMessageTransmitterImpl_1();


kiwiMessageTransmitter.send("Table="+menu+":LastModifiedBy="+last_modified_by+":LastModifiedDate="+last_modified_date+":Employee
ID="+employeeid+":ActionPerformed="+action+":Record="+record);



      }

      JOptionPane.showMessageDialog(null,"You have imported "+SecEventCount+" security Events","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);



      }

      }

      checkact=1;

    }
```

```
   private void SearchempactvtytxtbtnActionPerformed(java.awt.event.ActionEvent evt) {


      String employee_id = SearchempactvtytxtboxgetText().trim();

      if (employee_id.isEmpty())

      {

         JOptionPane.showMessageDialog(null," PLEASE ENTER A EMPLOYEE ID","BLANK FIELDS" ,
JOptionPane.INFORMATION_MESSAGE);

      }

      else

      {

        for( int i = ProcessUserActivityTbl1.getModel().getRowCount() - 1; i >= 0; i-- )

            {

             ((DefaultTableModel)ProcessUserActivityTbl1.getModel()).removeRow(i);

             }

             int SecEventCount=0;

             String query="SELECT * FROM SIEMCONVERTBU WHERE EMPLOYEEID like '%"+employee_id+"%' UNION SELECT *
FROM SIEMCONVERTDEPT WHERE EMPLOYEEID like '%"+employee_id+"%' union SELECT * FROM SIEMCONVERTJT WHERE
EMPLOYEEID like '%"+employee_id+"%' UNION SELECT * FROM SIEMCONVERTPS WHERE EMPLOYEEID like
'%"+employee_id+"%' UNION SELECT * FROM SIEMCONVERTEREMPS WHERE EMPLOYEEID like '%"+employee_id+"%' order by
lastmodifieddate desc";

               rs=conMysql.retrieve(query);

                try

                 {

                   while (rs.next())

                       {


                           String menu=rs.getString(1);

                           String modifiedby=rs.getString(2);

                           String employeeid=rs.getString(3);

                           String action=rs.getString(4);

                           String lastdatemodified=rs.getString(5);

                           String record=rs.getString(6);




                           ((DefaultTableModel)ProcessUserActivityTbl1.getModel()).addRow(new
Object[]{menu,modifiedby,employeeid,action,lastdatemodified,record});

                           SecEventCount++;

                       }

                     JOptionPane.showMessageDialog(null,"We have found "+SecEventCount+" Events","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);

                  }
```

```
                catch(SQLException w)
                {
                    JOptionPane.showMessageDialog(null,w.getMessage()," NO RECORD FOUND ",JOptionPane.ERROR_MESSAGE);
                }
               if(SecEventCount!=0)
    {
       int reply=JOptionPane.showConfirmDialog(null,"Do you want to sent this report to Syslog Server","",JOptionPane.YES_NO_OPTION);
       if (reply==JOptionPane.YES_OPTION)
       {

          for ( int i = ProcessUserActivityTbl1.getModel().getRowCount() - 1; i >= 0; i-- )
          {
            String menu =ProcessUserActivityTbl1.getModel().getValueAt(i,0).toString();
            String last_modified_by =ProcessUserActivityTbl1.getModel().getValueAt(i,1).toString();
            String last_modified_date =ProcessUserActivityTbl1.getModel().getValueAt(i,4).toString();
            String employeeid =ProcessUserActivityTbl1.getModel().getValueAt(i,2).toString();
            String action =ProcessUserActivityTbl1.getModel().getValueAt(i,3).toString();
            String record = ProcessUserActivityTbl1.getModel().getValueAt(i,5).toString();

       final IMessageTransmitter kiwiMessageTransmitter = new KiwiMessageTransmitterImpl();

kiwiMessageTransmitter.send("Table="+menu+":LastModifiedBy="+last_modified_by+":LastModifiedDate="+last_modified_date+":Employee
ID="+employeeid+":ActionPerformed="+action+":Record="+record);


    }
     JOptionPane.showMessageDialog(null,"You have imported "+SecEventCount+" security Events","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);


       }
    }
              }


  }


  private void ProcessUserActivityTbl1MouseClicked(java.awt.event.MouseEvent evt) {
    if (checkact==1)
    {

    if (ProcessUserActivityTbl1.getSelectedRow() > -1) {
```

```
        String EMPid=((ProcessUserActivityTbl1.getValueAt(ProcessUserActivityTbl1.getSelectedRow(), 2).toString()));

        String EventDate=((ProcessUserActivityTbl1.getValueAt(ProcessUserActivityTbl1.getSelectedRow(), 4).toString()));


       int SecEventCount=0;

   String queryactivity="SELECT * FROM [SIEM_CONVERTER].[DBO].[useractivitylog] where [employeeid]='"+EMPid+"' AND
[last_modified_date] > '"+EventDate+"'";


   rs=consql.retrieve(queryactivity);

   try

      {

      while (rs.next())

      {

         String menu=rs.getString("menu");

         String last_modified_by=rs.getString("last_modified_by");

         String last_modified_date=rs.getString("last_modified_date");

         String employee_id=rs.getString("employeeId");

         String action=rs.getString("action");

         String record=rs.getString("record");


          final IMessageTransmitter kiwiMessageTransmitter = new KiwiMessageTransmitterImpl_1();

kiwiMessageTransmitter.send("Table="+menu+":LastModifiedBy="+last_modified_by+":LastModifiedDate="+last_modified_date+":Employee
ID="+employee_id+":ActionPerformed="+action+":Record="+record);


         SecEventCount++;

      }

      JOptionPane.showMessageDialog(null,"We have sent "+SecEventCount+" Deleted Activity Events to SysLog Server for Employee ID
"+EMPid+" ","COMPLETE" , JOptionPane.INFORMATION_MESSAGE);

      }

      catch(SQLException w)

      {

         JOptionPane.showMessageDialog(null,w.getMessage()," NO RECORD FOUND ",JOptionPane.ERROR_MESSAGE);

      }


      } }

      else if (checkact==2)

      {

      if (ProcessUserActivityTbl1.getSelectedRow() > -1) {

         String recid=((ProcessUserActivityTbl1.getValueAt(ProcessUserActivityTbl1.getSelectedRow(), 5).toString()));

         String EventDate=((ProcessUserActivityTbl1.getValueAt(ProcessUserActivityTbl1.getSelectedRow(), 4).toString()));
```

```
        int SecEventCount=0;

    String queryactivity="SELECT * FROM [SIEM_CONVERTER].[DBO].[useractivitylog] where [recordid]='"+recid+"' AND
[last_modified_date] >= '"+EventDate+"'";


    rs=consql.retrieve(queryactivity);

    try

        {

        while (rs.next())

        {

            String menu=rs.getString("menu");

            String last_modified_by=rs.getString("last_modified_by");

            String last_modified_date=rs.getString("last_modified_date");

            String employee_id=rs.getString("employeeId");

            String action=rs.getString("action");

            String record=rs.getString("record");


             final IMessageTransmitter kiwiMessageTransmitter = new KiwiMessageTransmitterImpl_1();

kiwiMessageTransmitter.send("Table="+menu+":LastModifiedBy="+last_modified_by+":LastModifiedDate="+last_modified_date+":Employee
ID="+employee_id+":ActionPerformed="+action+":Record="+record);


            SecEventCount++;

        }

        JOptionPane.showMessageDialog(null,"We have sent "+SecEventCount+" deleted records to SysLog Server ","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);

        }

        catch(SQLException w)

        {

            JOptionPane.showMessageDialog(null,w.getMessage()," NO RECORD FOUND ",JOptionPane.ERROR_MESSAGE);

        }


        }

        }

    }


    private void ProcessUserActivityTbl1FocusGained(java.awt.event.FocusEvent evt) {


    }
```

```
    private void DeletedbtnActionPerformed(java.awt.event.ActionEvent evt) {

        SimpleDateFormat formatter= new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");

      Date date= new Date();


        for( int i = ProcessUserActivityTbl1.getModel().getRowCount() - 1; i >= 0; i-- )

        {

            ((DefaultTableModel)ProcessUserActivityTbl1.getModel()).removeRow(i);

        }


         for( int i = ProcessUserActivityTbl1.getModel().getRowCount() - 1; i >= 0; i-- )

        {

            ((DefaultTableModel)ProcessUserActivityTbl1.getModel()).removeRow(i);

        }

        int SecEventCount=0;

    String query="SELECT * FROM SIEMDELETE  ";

    rs=conMysql.retrieve(query);

      try

        {

        while (rs.next())

        {


            String menu=rs.getString(1);

            String modifiedby=rs.getString(2);

            String employeeid=rs.getString(3);

            String action=rs.getString(4);

            Date lastdatemodified=rs.getTimestamp(5);

            String record=rs.getString(6);




            ((DefaultTableModel)ProcessUserActivityTbl1.getModel()).addRow(new
Object[]{menu,modifiedby,employeeid,action,lastdatemodified,record});

            SecEventCount++;

        }

        JOptionPane.showMessageDialog(null,"We have found "+SecEventCount+" Events","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);

        }

        catch(SQLException w)

        {

            JOptionPane.showMessageDialog(null,w.getMessage()," NO RECORD FOUND ",JOptionPane.ERROR_MESSAGE);
```

```
    }
   if(SecEventCount!=0)
    {
       int reply=JOptionPane.showConfirmDialog(null,"Do you want to sent this report to Syslog Server","",JOptionPane.YES_NO_OPTION);
       if (reply==JOptionPane.YES_OPTION)
       {

          for ( int i = ProcessUserActivityTbl1.getModel().getRowCount() - 1; i >= 0; i-- )
          {
            String menu =ProcessUserActivityTbl1.getModel().getValueAt(i,0).toString();
            String last_modified_by =ProcessUserActivityTbl1.getModel().getValueAt(i,1).toString();
            String last_modified_date =ProcessUserActivityTbl1.getModel().getValueAt(i,4).toString();
            String employeeid =ProcessUserActivityTbl1.getModel().getValueAt(i,2).toString();
            String action =ProcessUserActivityTbl1.getModel().getValueAt(i,3).toString();
            String record = ProcessUserActivityTbl1.getModel().getValueAt(i,5).toString();

       final IMessageTransmitter kiwiMessageTransmitter = new KiwiMessageTransmitterImpl_1();

kiwiMessageTransmitter.send("Table="+menu+":LastModifiedBy="+last_modified_by+":LastModifiedDate="+last_modified_date+":Employee
ID="+employeeid+":ActionPerformed="+action+":Record="+record);


       }
      JOptionPane.showMessageDialog(null,"You have imported "+SecEventCount+" security Events","COMPLETE" ,
JOptionPane.INFORMATION_MESSAGE);


       }
       }
       checkact=2;
   }


   /**
    * @param args the command line arguments
    */
   public static void main(String args[]) {
       /* Set the Nimbus look and feel */
       //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
       /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
        * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
        */
```

```
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(HOME.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(HOME.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(HOME.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(HOME.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new HOME().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton ClearloginEventtbkbtn;
private javax.swing.JButton Deletedbtn;
private javax.swing.JButton Externalloginbtn;
private javax.swing.JButton ImportSecEventbtn;
private javax.swing.JButton ImportUserActivityEventbtn;
private javax.swing.JButton ListUserActivityBtn;
private javax.swing.JScrollPane ProcessUserActivityScrollPane;
private javax.swing.JTable ProcessUserActivityTbl1;
private javax.swing.JTextField Searchempactvtytxtbox;
private javax.swing.JButton Searchempactvtytxtbtn;
private javax.swing.JScrollPane SecEventScrollPane;
```

```
    private javax.swing.JTable SecEventTbl;
    private javax.swing.JScrollPane UserActivityScrollPane;
    private javax.swing.JTable UserActivityTbl;
    private javax.swing.JButton clearactivitytblbtn;
    private javax.swing.JButton exportSysLogbtn;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JButton list_event_btn;
    private javax.swing.JTabbedPane process_tab;
    // End of variables declaration
}
package siemconverter.message;

public interface IMessageTransmitter {

        void send( final String message );
}

package siemconverter.message.kiwi;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import siemconverter.message.IMessageTransmitter;

public class KiwiMessageTransmitterImpl implements IMessageTransmitter {

        private static final Logger     LOGGER  = LoggerFactory.getLogger( KiwiMessageTransmitterImpl.class );

        @Override
        public void send( final String message ) {
                LOGGER.info( "Sentrifugo :{} ", message );

        }
}
```

```
package siemconverter.message.kiwi;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import siemconverter.message.IMessageTransmitter;

public class KiwiMessageTransmitterImpl_1 implements IMessageTransmitter {

	private static final Logger	LOGGER  = LoggerFactory.getLogger( KiwiMessageTransmitterImpl_1.class );

	@Override
	public void send( final String message ) {
		LOGGER.warn("Sentrifugo :{} ", message );

	}
}
<?xml version="1.0" encoding="UTF-8"?>
<configuration>

	<appender name="KIWI" class="ch.qos.logback.classic.net.SyslogAppender">
		<syslogHost>localhost</syslogHost>
	<!--<remoteHost>192.168.50.10</remoteHost>-->
	<port>514</port>
	<facility>local0</facility>
		<suffixPattern>%-5level -%msg%n</suffixPattern>
	<!--<suffixPattern>%thread: %-5level %logger{36} - %msg%n</suffixPattern> -->
	</appender>
	<logger name="siemconverter.message.kiwi" level="INFO">
		<appender-ref ref="KIWI" />
	</logger>
</configuration>
```

Database connectivity class

```
package siemconverter;

import java.sql.*;
import javax.swing.JOptionPane;
```

```
public class DBconnection  {

    Connection connect=null;
    Statement statement=null;
    ResultSet result =null;
    public DBconnection()
    {
       try
       {
       Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver"); //load the driver
       String url ="jdbc:sqlserver://localhost:1433;DatabaseName=SIEM_CONVERTER";
       String username = "siem";
       String password = "maneno70";
       connect = DriverManager.getConnection(url, username, password);//establishes a connection
       statement = connect.createStatement();
       }
       catch(ClassNotFoundException e)
       {
          JOptionPane.showMessageDialog(null,e.getMessage());
       }
       catch(SQLException e)
       {
          JOptionPane.showMessageDialog(null, e.getMessage());
       }
    }
       //method to handle inserting data and updating data in the database
      public void update(String query)
       {
          try
          {
               statement.executeUpdate(query);
          }
          catch(SQLException e)
          {
             JOptionPane.showMessageDialog(null, e.getMessage());
          }
       }
       //method to handle retrieving data from the database
```

```
    public ResultSet retrieve(String query)
    {
       try
       {
       result = statement.executeQuery(query);
       }
       catch(SQLException e)
       {
          JOptionPane.showMessageDialog(null,e.getMessage());
       }
       return result;
    }
    //garbage collector

}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package siemconverter;

import java.sql.*;
import javax.swing.JOptionPane;
/**
 *
 * @author mark.meli
 */
public class DBconnectionMysql {

   Connection connect=null;
   Statement statement=null;
   ResultSet result =null;
   public DBconnectionMysql()
   {
      try
      {
      Class.forName("com.mysql.jdbc.Driver"); //load the driver
```

```
  String url ="jdbc:mysql://localhost:3306/sentrifugo";
  String username = "root";
  String password = "kenya123*";
  connect = DriverManager.getConnection(url, username, password);//establishes a connection
  statement = connect.createStatement();
  }
  catch(ClassNotFoundException e)
  {
     JOptionPane.showMessageDialog(null,e.getMessage());
  }
  catch(SQLException e)
  {
     JOptionPane.showMessageDialog(null, e.getMessage());
  }
}
  //method to handle inserting data and updating data in the database
  public void update(String query)
  {
     try
     {
          statement.executeUpdate(query);
     }
     catch(SQLException e)
     {
        JOptionPane.showMessageDialog(null, e.getMessage());
     }
  }
  //method to handle retrieving data from the database
  public ResultSet retrieve(String query)
  {
     try
     {
     result = statement.executeQuery(query);
     }
     catch(SQLException e)
     {
        JOptionPane.showMessageDialog(null,e.getMessage());
     }
```

```
        return result;
    }
    //garbage collector

}
```

Database scripts

```
USE [master]
GO
/****** Object:  Database [SIEM_CONVERTER]    Script Date: 01/08/2021 16:47:51 ******/
CREATE DATABASE [SIEM_CONVERTER]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'SIEM_CONVERTER', FILENAME = N'E:\DB_DATA\Data\SIEM_CONVERTER.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )
 LOG ON
( NAME = N'SIEM_CONVERTER_log', FILENAME = N'E:\DB_DATA\Data\SIEM_CONVERTER_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )
 WITH CATALOG_COLLATION = DATABASE_DEFAULT
GO
ALTER DATABASE [SIEM_CONVERTER] SET COMPATIBILITY_LEVEL = 150
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [SIEM_CONVERTER].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [SIEM_CONVERTER] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET ANSI_NULLS OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET ANSI_PADDING OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET ARITHABORT OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET AUTO_CLOSE OFF
```

```
GO
ALTER DATABASE [SIEM_CONVERTER] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [SIEM_CONVERTER] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [SIEM_CONVERTER] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET  DISABLE_BROKER
GO
ALTER DATABASE [SIEM_CONVERTER] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [SIEM_CONVERTER] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET RECOVERY FULL
GO
ALTER DATABASE [SIEM_CONVERTER] SET  MULTI_USER
```

```
GO
ALTER DATABASE [SIEM_CONVERTER] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [SIEM_CONVERTER] SET DB_CHAINING OFF
GO
ALTER DATABASE [SIEM_CONVERTER] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [SIEM_CONVERTER] SET TARGET_RECOVERY_TIME = 60 SECONDS
GO
ALTER DATABASE [SIEM_CONVERTER] SET DELAYED_DURABILITY = DISABLED
GO
ALTER DATABASE [SIEM_CONVERTER] SET ACCELERATED_DATABASE_RECOVERY = OFF
GO
EXEC sys.sp_db_vardecimal_storage_format N'SIEM_CONVERTER', N'ON'
GO
ALTER DATABASE [SIEM_CONVERTER] SET QUERY_STORE = OFF
GO
USE [SIEM_CONVERTER]
GO
/****** Object:  User [siem]    Script Date: 01/08/2021 16:47:51 ******/
CREATE USER [siem] FOR LOGIN [siem] WITH DEFAULT_SCHEMA=[dbo]
GO
ALTER ROLE [db_owner] ADD MEMBER [siem]
GO
/****** Object:  Table [dbo].[useractivitylog]    Script Date: 01/08/2021 16:47:51 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[useractivitylog](
        [ID] [numeric](18, 0) IDENTITY(1,1) NOT NULL,
        [menu] [varchar](50) NULL,
        [last_modified_by] [varchar](50) NULL,
        [last_modified_date] [datetime] NULL,
        [employeeid] [varchar](50) NULL,
        [action] [varchar](50) NULL,
```

```
	[record] [varchar](500) NULL,
	[recordid] [numeric](18, 0) NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[userloginlog]    Script Date: 01/08/2021 16:47:51 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[userloginlog](
	[Userid] [numeric](18, 0) NULL,
	[emprole] [varchar](50) NULL,
	[group_id] [numeric](18, 0) NULL,
	[employeeId] [varchar](50) NULL,
	[emailaddress] [varchar](50) NULL,
	[userfullnames] [varchar](50) NULL,
	[logindatetime] [datetime] NULL,
	[empipaddress] [varchar](50) NULL,
	[status] [varchar](50) NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[USERS]    Script Date: 01/08/2021 16:47:51 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[USERS](
	[USER_ID] [int] IDENTITY(1,1) NOT NULL,
	[USER_NAME] [varchar](10) NOT NULL,
	[FULL_NAMES] [varchar](50) NULL,
	[USER_PASSWORD] [varchar](20) NULL,
	[D_UPDATE] [datetime] NULL,
PRIMARY KEY CLUSTERED
(
	[USER_ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
```

```
) ON [PRIMARY]
GO
USE [master]
GO
ALTER DATABASE [SIEM_CONVERTER] SET  READ_WRITE
GO
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `siemconvertbu` AS
    SELECT
        'BusinessUnit' AS `menu`,
        `mu`.`userfullname` AS `lastmodifiedby`,
        `mu`.`employeeId` AS `employeeid`,
        `mbu`.`id` AS `id`,
        IF((`mbu`.`createddate` = `mbu`.`modifieddate`),
            'ADD',
            'EDIT') AS `action`,
        (`mbu`.`modifieddate` + INTERVAL 3 HOUR) AS `lastmodifieddate`,
        CONCAT('UnitName',
                COALESCE(`mbu`.`unitname`, ''),
                ':UnitCode',
                COALESCE(`mbu`.`unitcode`, 'null'),
                ':Description=',
                COALESCE(`mbu`.`description`, 'null'),
                ':StartDate=',
                CONVERT( COALESCE(`mbu`.`startdate`, 'null') USING UTF8),
                ':Address=',
                COALESCE(`mbu`.`address1`, 'null')) AS `record`
    FROM
        (`main_businessunits` `mbu`
        LEFT JOIN `main_users` `mu` ON ((`mbu`.`createdby` = `mu`.`id`)))
    WHERE
        (`mbu`.`id` IS NOT NULL)
CREATE
```

```
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `siemconvertdept` AS
    SELECT
        'Departments' AS `menu`,
        `mu`.`userfullname` AS `lastmodifiedby`,
        `mu`.`employeeId` AS `employeeid`,
        `mdpt`.`id` AS `id`,
        IF((`mdpt`.`createddate` = `mdpt`.`modifieddate`),
            'ADD',
            'EDIT') AS `action`,
        (`mdpt`.`modifieddate` + INTERVAL 3 HOUR) AS `lastmodifieddate`,
        CONCAT('DepartmentName=',
                COALESCE(`mdpt`.`deptname`, 'null'),
                ':DeptCode=',
                COALESCE(`mdpt`.`deptcode`, 'null'),
                ':StartDate',
                CONVERT( COALESCE(`mdpt`.`startdate`, 'null') USING UTF8),
                ':Address=',
                COALESCE(`mdpt`.`address1`, 'null'),
                ': DeptHead=',
                CONVERT( COALESCE((SELECT
                            `hmu`.`userfullname`
                        FROM
                            `main_users` `hmu`
                        WHERE
                            (`hmu`.`id` = `mdpt`.`depthead`)),
                    'null') USING UTF8)) AS `record`
    FROM
        (`main_departments` `mdpt`
        LEFT JOIN `main_users` `mu` ON ((`mdpt`.`modifiedby` = `mu`.`id`)))
    WHERE
        (`mdpt`.`id` IS NOT NULL)
CREATE
    ALGORITHM = UNDEFINED
```

```
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `siemconverteremps` AS
    SELECT
        'Employee Details' AS `menu`,
        `mu`.`userfullname` AS `lastmodifiedby`,
        `mu`.`employeeId` AS `employeeid`,
        `emps`.`id` AS `id`,
        IF((`emps`.`createddate` = `emps`.`modifieddate`),
            'ADD',
            'EDIT') AS `action`,
        (`emps`.`modifieddate` + INTERVAL 3 HOUR) AS `lastmodifieddate`,
        CONCAT('EmployeeID=',
                COALESCE(`emps`.`employeeId`, 'null'),
                ':JoinDate=',
                CONVERT( COALESCE(`emps`.`date_of_joining`, 'null') USING UTF8),
                ':DateOfLeaving=',
                CONVERT( COALESCE(`emps`.`date_of_leaving`, 'null') USING UTF8),
                ':EmployeeName=',
                COALESCE(`emps`.`userfullname`, 'null'),
                'EmailAddress=',
                COALESCE(`emps`.`emailaddress`, 'null'),
                ':Contacts=',
                COALESCE(`emps`.`office_number`, 'null'),
                ',',
                COALESCE(`emps`.`contactnumber`, 'null'),
                ',',
                COALESCE(`emps`.`extension_number`, 'null'),
                ':BusinessUnitName',
                COALESCE(`emps`.`businessunit_name`, 'null'),
                'Department=',
                COALESCE(`emps`.`department_name`, 'null'),
                '',
                ':EmployementStatus=',
                COALESCE(`emps`.`emp_status_name`, 'null'),
                ':PositionName=',
```

```
            COALESCE(`emps`.`position_name`, 'null'),
            'ReportingManager=',
            COALESCE(`emps`.`reporting_manager_name`, 'null')) AS `record`
    FROM
        (`main_employees_summary` `emps`
        LEFT JOIN `main_users` `mu` ON ((`emps`.`modifiedby` = `mu`.`id`)))
    WHERE
        (`emps`.`id` IS NOT NULL)
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `siemconvertjt` AS
    SELECT
        'Job title' AS `menu`,
        `mu`.`userfullname` AS `lastmodifiedby`,
        `mu`.`employeeId` AS `employeeid`,
        `mjt`.`id` AS `id`,
        IF((`mjt`.`createddate` = `mjt`.`modifieddate`),
            'ADD',
            'EDIT') AS `action`,
        (`mjt`.`modifieddate` + INTERVAL 3 HOUR) AS `lastmodifieddate`,
        CONCAT(COALESCE(`mjt`.`jobtitlecode`, ''),
            ':JobTitle=',
            COALESCE(`mjt`.`jobtitlename`, 'null'),
            ':JobDescription=',
            COALESCE(`mjt`.`jobdescription`, 'null'),
            ':JobPayGrade=',
            COALESCE(`mjt`.`jobpaygradecode`, 'null'),
            ':JobPayFrequency=',
            COALESCE(`mjt`.`jobpayfrequency`, 'null')) AS `record`
    FROM
        (`main_jobtitles` `mjt`
        LEFT JOIN `main_users` `mu` ON ((`mjt`.`modifiedby` = `mu`.`id`)))
    WHERE
        (`mjt`.`id` IS NOT NULL)
```