

Centralized Information Log Management Server

JARAMOGI OGINGA ODINGA UNIVERSITY OF SCIENCE
AND TECHNOLOGY

IMPLEMENTATION OF CENTRALIZED INFORMATION SYSTEMS LOGS SERVER

KAMITA PAUL KINUTHIA

ODHIAMBO RODGERS OKOTH

BACHELOR OF COMPUTER SECURITY AND FORENSICS

A PROJECT SUBMITTED TO THE SCHOOL OF INFORMATICS AND INNOVATIVE
STUDIES AT JARAMOGI OGINGA ODINGA UNIVERSITY OF SCIENCE &
TECHNOLOGY

DECLARATION

The content of this document is the original work based on our own research and to the best of our knowledge it has not been presented elsewhere for academic purposes.

KINUTHIA PAUL KAMITA

I132/0863/2013

Signed.....

Date:

ODHAMBO RODGERS OKOTH

I132/3146/2013

Signed.....

Date:

This project is submitted as part of the Examiners Board requirement for the award of the degree of Bachelor of Science in Computer Security and Forensics from Jaramogi Oginga Odinga University of Science and Technology.

Project supervisor:

MR. PAUL ABUONJI

Signed

Date:

ABSTRACT

Log management and analysis is a vital part of organization's network management and system information that refers to different security events, which occur within the system. Logs are used for different security events which occur within the system such as recording user activities, track authentication attempts and other security events. Due to increasing number of threats against networks and systems, the number of security logs increases. However, many organizations that work in a distributed environment face the following problems: log generation and storage, log protection and log analysis. Moreover, ensuring that security, system and network administrators analyze log data in an effective way is another issue. In this project we implement a centralized information systems log server. This allows real time detection and alert of all log transactions from different client machines to one centralized server.

The overall objective of this project is to implement a centralized information system log server from which keystrokes directory changes and windows open will reveal activities and program run by the user and directory changes on the system user machine. The methodology chosen for this project is agile methodology, scrum software development model.

The project was implemented focusing on development of sprints which involved of new release functionality, requirements, quality, cost and competition, results of the implementation were positive describing the success of our project, the system therefore achieved its goals meeting our objectives, however we recommend future work on our project to be done to ensuring it runs in a variety of operating systems.

ACKNOWLEDGMENT

We thank God Almighty for giving us the strength, good health and the opportunity to undertake this project

We are pleased to acknowledge Mr. Paul Abuonji for his insight, invaluable guidance and the many discussion sessions which gave us the enthusiasm to compose the project report as well as design and build this system during the course of this project work, without his guidance this project would have been an uphill task.

We are also grateful to other members of the School of Informatics and Innovative Studies team who co-operated with us regarding our project.

Finally we would like to express our gratitude towards our parents for their kind cooperation and encouragement which helped us in completion of this project.

LIST OF TABLES

Table 4.1: 38
Table 4.2: 39

LIST OF FIGURES

Figure 1: Project workflow	4
Figure 4.1:Graphical model of CILMS.....	17
Figure 4.2:Activity Diagram for CILMS.	18
Figure 4.3:Dataflow Diagram for CILMS.	19
Figure 4.4:UML for CILMS.	21
Figure 4.5:Dataflow Diagram for Blacklist	22
Figure 4.6:UML for blacklist	23
Figure 4.7: Blacklist Sourcecode	23
Figure 4.8: Test result for blacklist sprint	24
Figure 4.9:UML class for Onkeyboard Event	25
Figure 4.10: Onkeyboard Event Source code	25
Figure 4.11:Test results for keystrokes record.....	26
Figure 4.12:UML for windows open.	26
Figure 4.13.:Windows open development	28
Figure 4.14:Windows Open Testing and Debugging	29
Figure 4.15:Activity Diagram for Directory changes.....	29
Figure 4.16:UML class diagram for directory changes	30
Figure 4.17: Directory changes development	31
Figure 4.18:Debugging and testing of directory changes module	32
Figure 4.19:UML class diagram for screenshot.....	33
Figure 4.20:Screenshot module development.....	34
Figure 4.21:Testing and debugging of screenshot capture sprint.	35
Figure 4.22: UML class for database notation.....	35
Figure 4.23:Mysql query for creating CILMS_log table.....	36
Figure 4.24:Mysql query for creating CILMS user table	37
Figure 4.25:Database table screenshot from phpmysql.....	37
Figure 4.26:CILMS log table	41
Figure 4.27: UML case diagram for administration dashboard.	Error! Bookmark not defined. 2
Figure 4.24:Development of administration dashboard	43
Figure 4.25: Testing and debugging of administration dashboard	43

LIST OF ACRONYMS

CILMS	Centralized Information Log Information Management System
US	United States
IT	Information Technology
PERSEREC	Personnel Security Research Center
IR	Information Retrieval
UML	Unified Modelling Language
IP	Internet Protocol
VB	Visual Basic
SI	System Integration
UI	Unit Integration

Contents

DECLARATION i

ABSTRACT..... ii

ACKNOWLEDGMENT iii

LIST OF TABLES iv

LIST OF FIGURES 1

LIST OF ACRONYMS 3

CHAPTER ONE 7

1.0. Introduction..... 7

 1.1. Background of the study..... 7

 1.2. Problem Statement..... 8

 1.3. Objectives of the project..... 8

 1.3.1. Overall objective of the project..... 8

 1.3.2. Specific Objectives 8

 1.4. Research questions..... 9

 1.5. Justification of the study..... 9

 1.6. Limitations of the study..... 9

 1.7. Expected output 10

 1.8. Filter Rules 10

CHAPTER TWO 12

2.0. Literature Review 12

 2.1. Existing Systems..... 14

 1. ObserveIT 14

 2. WinWhatWhere Investigator..... 14

 3. WiIRE system..... 15

CHAPTER THREE 16

3.0. Methodology 16

 Agile methodology, scrum software development model..... 16

3.1. Justification to using Agile methodology 16

 3.2.1. Advantages of Agile 17

 3.2.2. Phases of Agile methodology 18

3.3. System Development life cycle activities 18

 3.3.1. Requirement Elicitation 18

3.3.2.	Analysis	18
3.3.3.	System Design	19
3.3.4.	Object Design	19
3.3.5.	Implementation	19
3.3.6.	Testing	20
CHAPTER FOUR		21
4.0. IMPLEMENTATION		21
4.1. Initial Planning & system architecture		21
4.2. Analysis and Requirements		21
4.2.1. Functional Requirements		21
Keystroke listener		21
Blacklist		21
Windows open detector		21
4.2.2. Nonfunctional requirements		22
4.2.3. Software requirement specification		22
4.3. System Architectural Design		23
4.3.1. Structured classes		23
4.3.2: Activity Diagram		23
4.3.3: UML Data flow diagram		25
4.3.4: Use Case Diagram		25
4.4. Implementation		26
4.4.1. Overall strategy for implementation		26
4.4.2. Programming language		26
4.5. Product backlog		27
4.5.1. Sprint 1: Words Blacklist		27
4.5.1.4. Development words blacklist		28
4.5.2 Sprint 2: Keystroke Recorder		30
4.5.2.3. Development		30
4.5.3 Sprint 3: Windows open recorder		32
4.5.3.3. Development		32
4.5.4. Sprint 4: Directory changes watchdog		33
4.5.4.4. Development		35
4.5.5.0. Sprint 5: Screenshot Capture		36

4.5.6. Sprint 6: system log database.....	37
4.5.6.3. Development	39
4.5.7. Sprint: Administrator Dashboard	42
4.5.7.3. Development	42
4.6. Testing.....	44
4.6.1. Unit Testing	45
4.6.2. Integration testing.....	46
4.6.3. System testing.....	47
4.6.4. Security Testing.....	47
CHAPTER FIVE	48
DISCUSSIONS, CONCLUSIONS AND RECOMMENDATIONS	48
5.1. Introduction.....	48
5.2. Discussion.....	48
5.3. Conclusion	49
5.4. Problems encountered during the project and how we overcame them	49
5.5. Future Work.....	50
APPENDICES.....	51
Appendix (I).....	51
Appendix (II).....	52
Appendix (III).....	53
The Scrum Team.....	53
APPENDIX (IV).....	54
APPENDIX (V)	55
CLIENT USER MANUAL	55
APPENDIX (VI).....	58
CILMS-SERVER APPLICATION.....	58
APPENDIX (VI).....	60
PROGRAM CODES.....	60
REFERENCES.....	65

CHAPTER ONE

1.0. Introduction

Log management is a collective process consists of policies used for analyzing, evaluating, monitoring and disposing the large volume of data. Timelogs provides very useful information for security management. Logs have evolved to contain information related to many different events occurring within networks or in systems. Computer security logs are audit logs that track user authentication attempts and on the other hand security device logs are the logs that record possible attacks. Log data contains a wide variety of information about the events and after analysis it becomes easier to investigate and to set the account on any individual. Log data is to be preserve because it is an important source for digital forensics investigation too.

Log management is essential for security, accountability and for various information security compliances.

Our system will be designed to work on the target computer's software. Key loggers are used in IT organizations to troubleshoot technical problems with computers and business networks. Families and business people use key loggers legally to monitor network usage without their users' direct knowledge. We intend to develop a system that listens to keys struck by the user, and analyze them with a dictionary containing a list of know attack key words.

1.1. Background of the study

An early keylogger was written by Kivolowitz and posted to the Usenet news group net.unix-wizards,net.sources on.

The posting seems to be a motivating factor in restricting access to `/dev/kmem` on [Unix](#) systems. The user-mode program operated by locating and dumping character lists (clists) as they were assembled in the Unix kernel.

In the 1970s, spies installed keystroke loggers in the US Embassy and Consulate buildings in Moscow and St Petersburg.

They installed the bugs in [Selectric](#) II and Selectric III electric typewriters (Maneki 2012.)

Soviet embassies used manual typewriters, rather than electric typewriters, for classified information—apparently because they are immune to such bugs. As of 2013, Russian special services still use typewriters. (Agence France-Presse, Associated Press. "Wanted: 20 electric typewriters for Russia to avoid leaks". inquirer.net Jump up Anna Arutunyan. "Russian security agency to buy typewriters to avoid surveillance")

A hook is a point in the system message-handling mechanism where an application can install a subroutine to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure.

1.2. Problem Statement

With the growing scale and complexity of the IT systems, system control has become more difficult for a system administrator to have an insight in every piece of details the system users are involved in; hence it would be handy if one could send all relevant log entries to a central server for easy monitoring and control.

A centralized information Systems log server uses keystrokes generated from user's keyboard, analyses, listens to them in order to determine user's activity on the client machine, directory changes like write, delete and create can be monitored to determine illegal activities on the client machine. Information from keystrokes, directory changes and windows opened, reveal activities program run by used and directory changes on the system user machine

1.3. Objectives of the project

1.3.1. Overall objective of the project

The main objective of this project is to implement a centralized information system log server from which keystrokes directory changes and windows open will reveal activities and program run by the user and directory changes on the system user machine.

1.3.2. Specific Objectives

2. To implement a centralized Information System log server where keystrokes being generated from user's keyboard can be analyzed or listened to determine user activity on the client machine

3. To implement a centralized Information System in which directory changes such as file changes like write, delete and create will be detected and monitored to determine illegal activities on the client machine.
4. To implement a centralized Information System in which windows open by users will reveal activities and programs being run by the user

1.4. Research questions

2. How can Key loggers, windows open, directory changes be used to detect and prevent an attack?
3. Does the existing system solve the problem using keystrokes effectively?
4. What are the benefits of using a centralized Information Log Server?

1.5. Justification of the study

Centralized Information system log server eases the network of every network administrator. It is a real log server tool used to centralized, secure stored log of platform that run on Information system log server also incorporates a host of powerful features including filtering based on message content, as well as analysis capabilities.

Keeping storage requirements static minimizes support staff cost as having the staff manually archive files is inefficient, error prone and waste of resources.

Avoids logistical problems searching a single transaction by logging into each server and searching through each

The whole system can be viewed as a combined structure with important process. Centralized log application also presents the real-time alerting, filtering and management. This feature provides a more user friendly interface to centralized log and understands the situation of the centralized information system log server.

1.6. Limitations of the study

In an incident where a user changes the host computer system clock, the user ends up interrupting the log server's timestamp

In case hacker finds the encryption keys of the administrators he or she can view all the logs

Centralized information system log server has a disadvantage in case of IP address spoofing.

Project Work flow

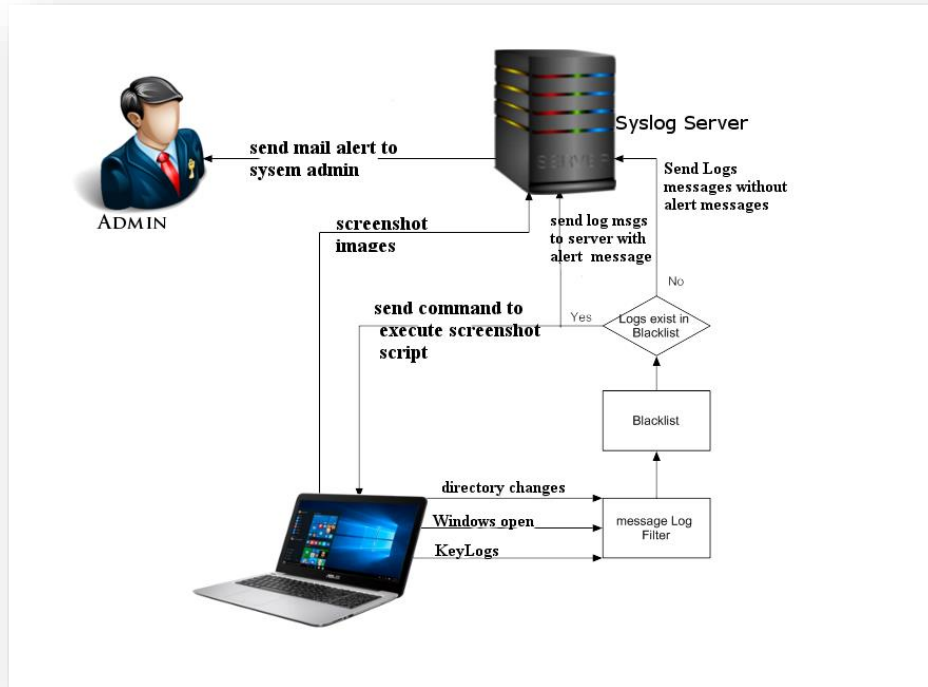


Figure 1.1: Project workflow

1.7. Expected output

At the final project stage of this project, a centralized information system log server is expected to be part of the information system, an alarm for an alert is very important to administrator to compare messages with alarm definition i.e. the blacklisted operations from the user’s machine, an alarm will be generated to the administrator if the operation matches the alarm definition.

1.8. Filter Rules

Filter rules are set of filters gathered in a filter list, each rule in the list is sequentially processed, and a rule contains conditions and actions

Possible conditions are:

- i. Keystrokes generated from the user’s keyboard are analyzed to determine the users activity

Centralized Information Log Management Server

- ii. Directory changes on client machine this include write, create and delete actions on a given directory.
- iii. Windows Open in that when a user opens a specific window open the logs take the name of the window opened for filtering.

CHAPTER TWO

2.0. Literature Review

This chapter presents the general layout of how centralized Information Systems Logs Server work. This chapter is the result of the literature study, and represents the work that others have done. It was important to gather this information because it enabled us to make choices for the prototype, and it would be difficult to create metrics without the knowledge of how such systems work

This section will discuss the fact finding techniques that have been adopted to gather relevant information pertaining this project, the significance and contribution conducted in related area are also conducted

The department of a trusted staff member with access to sensitive information can become a data breach if the staff member retains access to the data subsequent to termination of the trust relationship. In distributed systems, this can also occur with a breakdown in a web trust.

Previous studies have found that many of the information technology insiders who perpetrated malicious acts had been problem employees elsewhere. Others were inappropriately assigned to positions for which they were unqualified or were in other ways incompatible. Adequate recruitment and pre-employment screening could have prevented the resulting losses. In other cases, the manner in which the organization intervened with the at-risk employee actually escalated rather than mitigated the risk. These and other findings indicate that a number of basic organizational processes associated with employee hiring, placement, employee monitoring and management have direct implications for organizational security. The self-audit questions, standing alone, provide an evaluation tool for assessing the current level of vulnerability of any organization to damaging insider behavior and a means for developing an insider risk mitigation plan. (Shaw, 2009)

According to the 2007 Computer Security Institute survey, there was a 17% increase that year in reports of insider abuse, with 59% of respondents reporting insider problems (Computer Security Institute, 2007). While arrests for espionage have decreased in recent years, the theft of classified and sensitive information and technology by foreign adversaries continues to be a serious

problem. In the private sector, increasing percentages of corporate value (now as much as 75%) are directly linked to such intangible assets as intellectual property, indicating that these organizations are increasingly vulnerable to malevolent insider behavior. (Shaw, 2009)

According to The Defense Personnel Security Research Center (PERSEREC) earlier studies of espionage and computer abuse in the corporate and government sector have focused on fairly narrow behavioral, motivational, and technical case chronologies while not examining organizational or situational factors that can contribute to or mitigate insider risk. (Shaw, 2009)

In the book “Analysis, design and implementation of keyloggers and anti-keyloggers Canbek has proposed a new pattern of virtual keyboard . The solution in this paper emphasizes on login credential protection from screen capture software by using the concept of reordering of the keys. This is providing solution to only screen capturing software. But this captures the screen only when an event occurs. While in case of screen recording software there is no need of event occurrence. (Canbek, 2005)

Keystrokes can easily be guessed by analyzing the recorded video. The author in the book “Secure Authentication using Dynamic Virtual Keyboard Layout” (Agarwal & Mehra 2011), has proposed a solution to the screen capturing key logger by using a color coding mechanism and dynamic keyboard layout. The major flaw of this solution is that an attacker can identify the keys clicked on virtual keyboard. This can be done by analyzing the pattern of screen shot captured from the first appearance of the keyboard when no color coding mechanism is induced in it.

The problem faced today by system administrators are the correlation of Firewall logs, intrusion detection system event logs, operating system event logs, mail system logs, database logs, web server logs, antivirus logs, router/switch logs etc. (McIntyre, 2003) states that potentially, all these logs can identify a threat, and may receive hundreds or thousands of entries a day. The examination of these logs is often performed by staff short of time and knowledge, and for a company; resources may be a limitation. Despite the resource limitations we want log file correlation because it improves intrusion detection Abod, C. *etal* (2003)D. (Dillis, 2003) claims that some of the problems with existing tools today are that they are expensive, and that the most popular free tool is plagued with performance problems. Another problem is the complexity of the tools.

Shaw and Fischer (2004) found that security and personnel policies were lacking in eight out of the 10 cases of the insider attacks they reviewed. Missing policies and practices that could have deterred or prevented the attack led to earlier detection of risk, helped manage the at-risk employee, or reduced the odds of the attack. (Shaw & Fischer, 2004).

McIntyre (2003) examines some of the existing products. The products support a variety of systems and devices, so the first thing to do if one wishes to implement such systems is to find the one that best fit the needs of the environment.

Mcintyre, (2003) also provides some pricing information.

"The goal of security information management systems is to reduce the total cost of ownership of security devices by reducing the time security professionals spend on threat analysis and incident management, but when these products can cost anywhere from \$45,000 to \$100,000 and even up to \$300,000 or more depending on the number of devices supported, it may be hard for smaller companies to justify their purchase."

2.1. Existing Systems

1. ObserveIT

Privileged Session Recording makes it easy to monitor privileged users to detect leaks of sensitive and regulated information, because every privileged user action is monitored and analyzed in real-time. ObserveIT provides screen-recording technology to capture all user behavior, regardless of the environment. Beyond providing video playback of all user activity, ObserveIT turns these video recordings into metadata that can be easily searched, analyzed, audited and acted upon.

2. WinWhatWhere Investigator

Hancock-Beaulieu et al. (1990) supplemented their transaction logging with an application that included online questionnaires. Choo et al. (1998) had to develop their own logging software. Kelly (2004) used WinWhatWhere Investigator, which is a spy software package, used to covertly “monitor” a person's computer activities. Spy software has inherent disadvantages for use in user studies and evaluation, including granularity of data capture and privacy concerns.

3. WiIRE system

Toms, Freund, and Li (2004) develop the WiIRE system for conducting large scale evaluations. This system facilitates the evaluation of dispersed study participants; however, it is a server-side application focusing on the participant-interactions with Web server. As such, the entire “study” must occur within the WiIRE framework. This system facilitates the evaluation of dispersed study participants; however, it is a server-side application focusing on the participant-interactions with Web server. As such, the entire “study” must occur within the WiIRE framework. There are commercial applications for general-purpose (i.e., not specifically IR) user studies.

An example is Morae offered by TechSmith. Morae provides extremely detailed tracking of user actions, including video capture over a network. However, Morae is not specifically tailored for IR studies and captures so much information at such a fine granularity that it significantly complicates the data analysis process. To assist in addressing this need, a software application was developed for use in conjunction with transaction log and other types of IR studies. The application is coded in a standard programming language (Visual Basic 6). It is easy to install and collects a wide range of user–systems interactions. The application logs much of the user interactions identified by prior research (Kelly & Teevan, 2003; Oard & Kim, 2001), along with the content of the interaction (i.e., URL, document, results listing, etc.). These implicit feedback actions and documents are referred to as action–object pairs (Jansen, 2003). We have validated the application in a series of user studies (Jansen & Kroner, 2003; Jansen & McNeese, 2006) and have found the application to be extremely resilient, with near 100% operational effectiveness.

The existing models does not provide full-fledged solution to key logging and screen recording software. They provide security to some extent to key logging and screen capturing software. So there needs to be a solution for the same. Based on past studies of insider behavior, we have defined several areas of effective management intervention to mitigate the probability of damaging behaviors. These include policies and practices, recruitment, pre-employment screening, training and education, continuing evaluation and policy implementation, and employee intervention. In addition, we discuss features of organizational context that would magnify insider risk.

CHAPTER THREE

3.0. Methodology

In this chapter we discuss our project management model i.e. the model we employed for the design, planning, implementation and achievement of our project objectives

Agile methodology, scrum software development model

Agile is a system development methodology to build a system incrementally using short iterations, Agile adopts a process of frequent feedback where a workable product is delivered. The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework Centralized Information Log Management Server was implemented using agile, scrum software development model.

This methodology was chosen because it was suitable for this project. According to Linda Rising and Norman S. JanoffIt in the book “The Scrum Software Development Process for Small Teams” has the following features:

- Provides a working model earlier than traditional Software Development Models
- Frequent Requirement Changes is less impactful
- Self- Organization and employee empowerment
- Open and frequent Communication
- Opportunities for clear definitions
- Promotes Creativity, Collaboration, and Collaboration
- Improved processes
- Capitalizes upon feedback
- Iterative and Incremental working models
- Backlogs work items for better realization of work schedules and required work

3.1. Justification to using Agile methodology

We chose to use agile methodology because in our project several changes are anticipated during implementation of our project thus flexibility is much required agile methodology gives rooms for this flexibility as compared to the traditional methods of project management. We are able to make small objective changes without huge amendments to our budget and scheduling

We have been able to break down our project into prioritized requirements and delivering them individually within an iterative cycle, iteration in our case means the routine of developing small

sections i.e. dealing with smaller sections at a time. Being the development team we reviewed and assessed each iteration, we used the insights gained from each assessment to determine our next step in development. We were able to preschedule regular meetings to review the work completed on the previous iteration and plan on working on an upcoming iteration. We were able to set detailed goals in each iteration they included expected changes, time estimates, priorities and budgets.

Using agile method was essential as it bases on giving high priority the user participation from the very beginning of the development cycle, the object is to keep the client which happen to the system users in our cases, the objective is to keep the client involved in every step of our project so that they can have a project they are satisfied with at the end, the method saves time and money as the each level is tested and approved before moving to the next, if there exists any defects or challenges, then changes can be made during production cycles to fix the issue as compared to other methods which would not find defects s early because they don't test as often, this defects can find their way into final product thus increase in overhead prices and client dissatisfaction. Small incremental releases made visible to the project team through the project's development help to identify the issues early this ensures management of risk.

3.2.1. Advantages of Agile

- ✓ Monitoring - Pay regular attention to technical excellence and good design to enhance agility.
- ✓ Customer satisfaction
- ✓ Allows for changes to be made.
- ✓ Promotes collaboration
- ✓ Measure the Progress as per the Working Software.
- ✓ Maintain Constant Pace
- ✓ Deliver a working software frequently, ranging from a few weeks to a few months, considering shorter time-scale.
- ✓ Provides motivation between individual team members.
- ✓ Allows face-to-face Conversation

3.2.2. Phases of Agile methodology

- ✓ Agile follows the phases of software development life cycle which include;
- ✓ Project planning: initiate, ensure feasibility, plan schedule, and obtain approval.
- ✓ Analysis: understand computer lab needs and processing requirements
- ✓ Design: define solution system based on requirements and analysis decisions
- ✓ Implementation: construction, testing, user training, and installation of new system
- ✓ Support: keep system running and improve

3.3. System Development life cycle activities

In this section an overview of the technical activities associated with object oriented software engineering. Development activities deal with the complexity by constructing and validating models of the application domain or the system. The development activities include:

- ✓ Requirements Elicitation
- ✓ Analysis
- ✓ System Design
- ✓ Object Design
- ✓ Implementation
- ✓ Testing

3.3.1. Requirement Elicitation

During requirement elicitation, the client and developers define the purpose of the system; the result of this activity is a description of the system in terms of actors and use cases. Actors represent the external entities that interact with the system. Actors include roles such as end users, other computers the system needs to deal with and the environment. Use cases are general sequences of events that describe all possible actions between an actor and the system for a given piece of functionality

3.3.2. Analysis

During Analysis, developers aim to produce a model of the system that is correct, complete, consistent and unambiguous. Developers transform the use cases produced during requirements elicitation into an object model that completely describes the system. During this activity, developers discover ambiguities and inconsistencies in the use case model that they resolve with the client. The result of analysis is a system model annotated

with attributes, operations and associations. The system model can be described in terms of its dynamic interoperation

3.3.3. System Design

During the system design, developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams. Developers also select strategies for building the system such as the hardware or software platform on which the system will run, the persistent data management strategy, the global control flow, the access control policy and the handling of the boundary conditions. The result of system design is a clear description of each strategy, subsystem decomposition and a deployment diagram representing the hardware or software mapping of the system. Whereas both analysis and system design produce models of the system under construction, only analysis deals with entities that the client can understand. System design deals with a much more refined model that includes many entities that are beyond the comprehension and interest of the clients

3.3.4. Object Design

During the object design, developers define solution domain objects to bridge the gap between the analysis model and the hardware or software platform defined during system design. This includes precisely describing object and subsystem interfaces, selecting off-the-shelf components, restructuring the object model to attain design goals such as extensibility or understandability, and optimizing the object model for performance. The result of the object design activity is a detailed object model annotated with constraints and precise descriptions for each element

3.3.5. Implementation

During Implementation, developers translate the solution domain model into source code. This includes implementing the attributes and methods of each object and integrating all the objects such that they function as a single system. The implementation activity spans the gap between the detailed object design model and a complete set of source code files that can be compiled.

3.3.6. Testing

During this stage, the developers find the differences between the system and its models by executing the system with sample input data sets. During unit testing, developers compare the object design model with each object and subsystem. During integration testing, combinations of subsystems are integrated together and compared with the system design model. During system testing, typical and exception cases are run through the system and compared with the requirements model. The goal of testing is to discover as many faults as possible such that they can be repaired before delivery of the system. The planning of test phases occurs in parallel to the other development activities: system tests are planned during requirements elicitation and analysis, integration tests are planned during system design and unit tests are planned during object design

CHAPTER FOUR

4.0. IMPLEMENTATION

4.1. Initial Planning & system architecture

4.2. Analysis and Requirements

4.2.1. Functional Requirements

Centralized log Management system will include the following features

Keystroke listener

This feature will be used to listen to keyboard events of user.

Blacklist

Blacklisted words, alert message, application open and priority of risk will be contained on a file. Blacklist feature will be used to check whether logs contain any blacklisted word.

Windows open detector

This feature detects windows open by user and record it.

Notify user if suspicious activity is detected

It creates security awareness to the user by creating a popup notification containing security awareness message.

Directory changes detector

This features act as a watchdog for specified file directory. It will detect any activity done in that specific directory. For example; write, create or delete.

Send log

This feature will send log files to the centralized information system database.

Login

Administrators should input his/her login credentials before accessing logs dashboard.

View Logs Alert

Administrator showed be able to view logs alert

View session

The system should display number of sessions available to the user

View Risky users

The system should display number of sessions available to the user

4.2.2. Nonfunctional requirements

- ✓ Secure Connection and Encrypted Data Transfer
- ✓ Prevention of simultaneous Logins
- ✓ Application Loading - 7 second load time
- ✓ System Lock Out - locks the system after 5 failed attempts to authenticate
- ✓ System Time Out - logs the user out after 10 minutes' inactivity
- ✓ Keystroke logs are sent after the user key in 25 characters
- ✓ Windows pop notification last for 20 seconds

4.2.3. Software requirement specification

Windows Xp and above versions

Python

Wamp server

php 5.5.x

jQUERY 1.9.x

wampserver v2.0 +

Web Browsers.

Database: MySQL 5.4.3+

Documentation Tool: Microsoft Office

4.3. System Architectural Design

4.3.1. Structured classes

Figure 4.1. presents a graphical model of a centralized Information log management system's structure. It includes classes onKeyboardEvent, DirectoryChanges, windowsOpen, Blacklist, Screenshot, Notification and logDatabase. It shows that class Centralized Information log management system has a **one-to-one relationship** with class logDatabase—one CILMS object alert logs are stored in one logDatabase object.

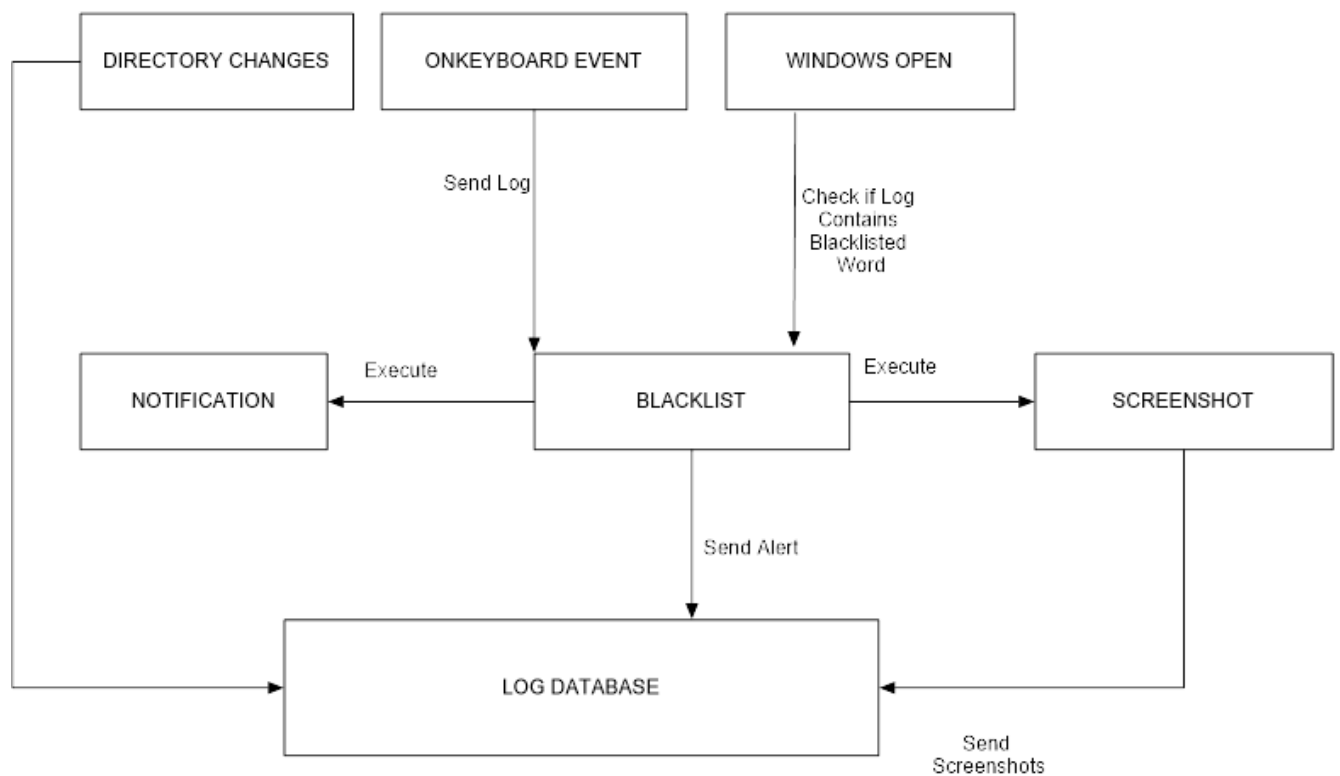


Figure 4.1: Shows graphical model of a Centralized Information Log Management System (CILMS).

4.3.2: Activity Diagram

Models an object's workflow (sequence of event) during program execution. Activity diagram models the activity the object will perform and in what order

Figure 4.2. shows an activity diagram for logs analysis notification and alert system. The system monitors keystrokes, windows open, directory changes: create security awareness by a popping up a notification that contains security alert, then sends alert to the administrator.

Centralized Information Log Management Server

The system receives logs from the user. The blacklist checks whether the received log contains words that are blacklisted by the administrator, if any blacklisted word exist the notification receives a request to send a security awareness alert to the user. Screenshots are then taken from the user's desktop. Then all logs are send to the remote database and system administrator is sent an alert.

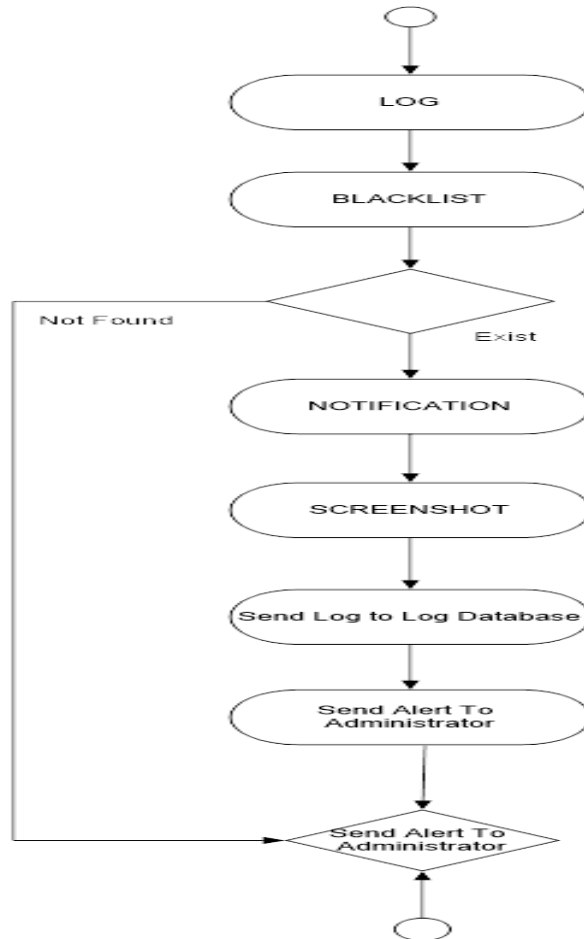


Figure 4.2: Activity Diagram for Centralized Information Log Management System (CILMS)

4.3.3: UML Data flow diagram

Figure 4.4.3: Shows Data Flow Diagram for Centralized Information Log Management System (CILMS).

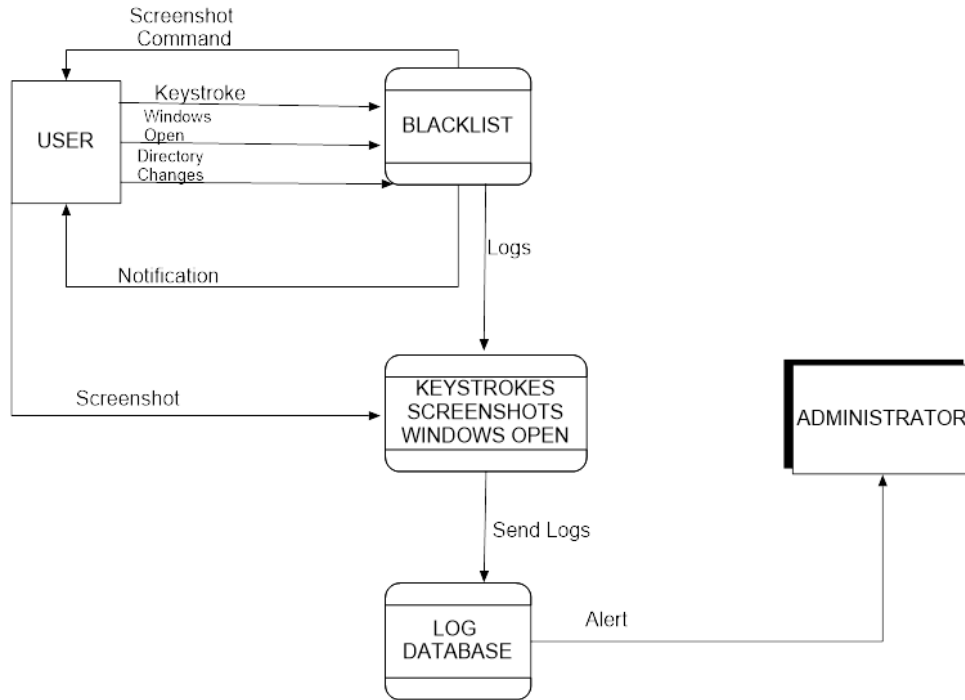


Figure 4.3. Data Flow Diagram for Centralized Information Log Management System (CILMS)

4.3.4: Use Case Diagram

Figure 4.4. Shows UML Use Case diagram for CILMS. The stick figure represents an actor, which defines the roles that an external entity—such as a person or another system—plays when interacting with the system. For our CILMS the actor is a User who can view an account balance, withdraw cash and deposit funds from

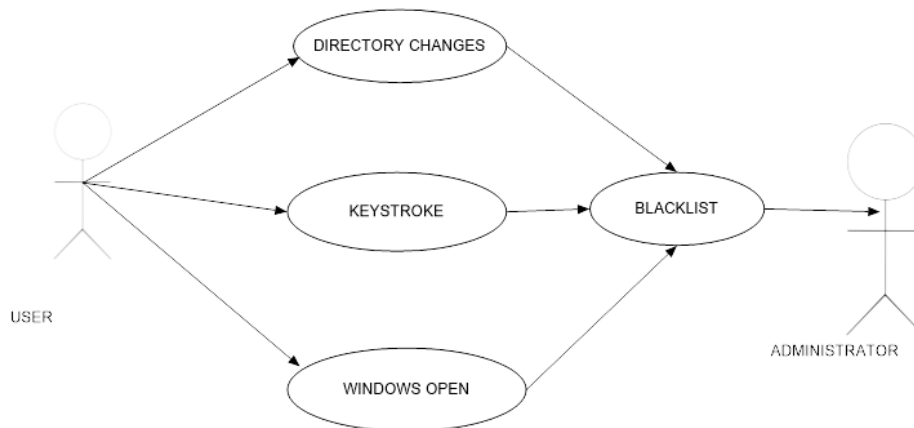


Figure 4.4| UML Use Case for Centralized Information Log Management System (CILMS)

4.4. Implementation

4.4.1. Overall strategy for implementation

Implementation of Centralized Information System Log Server (CILMS) will be focuses on development sprint. Development Sprints will involve development of new release functionality, with constant respect to the variables of time, requirements, quality, cost, and competition.

Interaction with these variables will defines the end of this phase. There are multiple, iterative development sprints, or cycles, that are used to evolve the system.

A Sprint is a set of development activities conducted over a pre-defined period, usually one to four weeks.

4.4.2. Programming language

Agent application software was developed using *Python programming language*. Python is a general-purpose language, which means it can be used to build just about anything, which will be made easy with the right tools/libraries.

Professionally, Python is great for backend web development, data analysis, artificial intelligence, and scientific computing.

Easy to Understand

Being a very high level language, Python reads like English, which takes a lot of syntax-learning stress off coding beginners. Python handles a lot of complexity for you, so it is very beginner-friendly in that it allows beginners to focus on learning programming concepts and not have to worry about too much details.

The Python documentation has a HOWTO section specifically for Python advocacy

Very Flexible

As a dynamically typed language, Python is really flexible. This means there are no hard rules on how to build features, and you'll have more flexibility solving problems using different methods (though the Python philosophy encourages using the obvious way to solve things). Furthermore, Python is also more forgiving of errors, so you'll still be able to compile and run your program until you hit the problematic part.

It Works Online Too

Web development is still a booming economic prospect for programmers. With Python's vast capabilities, you too can have a piece of the action. Django — the popular open source web application framework written in Python — is the foundation of such sites as Pinterest, *The New York Times*, *The Guardian*, Bit Bucket, and Instagram

4.5. Product backlog

1. Word Blacklist
2. Keystroke listener
3. Windows open recorder
4. Directory changes watchdog
5. Screenshot capture
6. Login page
7. Administrator dashboard
8. Notification/advice module

4.5.1. Sprint 1: Words Blacklist

4.5.1.1. Sprint planning

Word blacklist sprint backlog will have the following functionality:

- Load blacklisted word list
- Initiate an array of the blacklisted list
- Write blacklisted words to a file
- Set alert information (Alert message to be sent to a centralization server)

Blacklisted words, alert message, application open and priority of risk will be contained on a file. Blacklist feature will be used to check whether logs contain any blacklisted word.

4.5.1.2. Word Blacklist Design

Data Flow Diagram

Figure 4.6.1.1 Shows Data Flow Diagram for words blacklist.

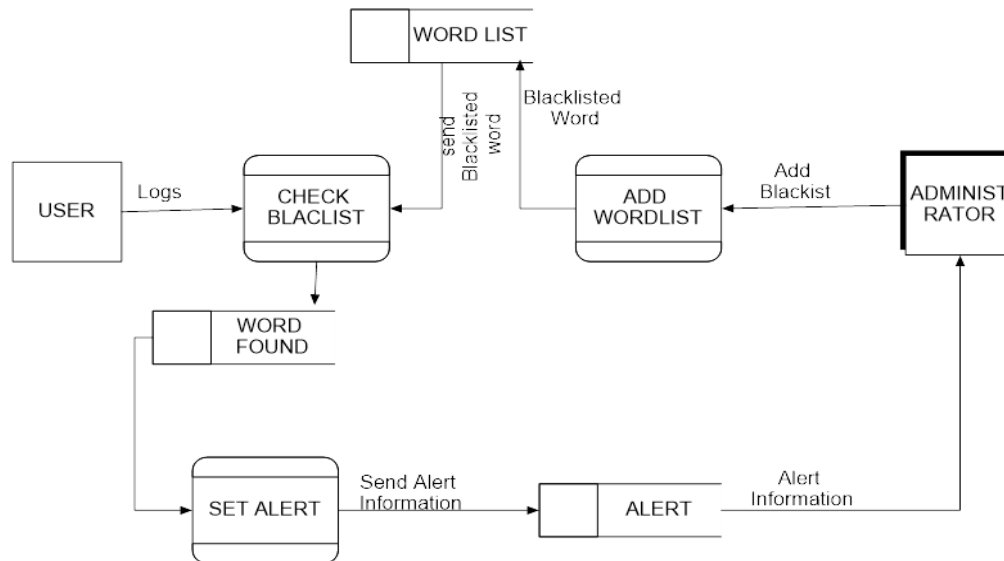


Figure 4.5 (CILMS) Data Flow Diagram for words blacklist

User logs will be checked against the words listed in the blacklist. If any blacklisted word is found in the user log, an alert message will be generated from the wordlist. The alert message will include;

1. blacklisted word that was in the log
2. Alert risk information
3. Priority of the risk
4. Application name

4.5.1.3. UML Class Diagram

Figure 4.6. Shows UML Class Diagram for words blacklist. It includes classes *SaveDictionary()*, *loadDictionary()*, *_set_Blacklist()*, *check_Blacklisted()* and *_setAlert()*.

Load blacklisted word list

Initiate an array of the blacklisted list

Write blacklisted words to a file

- Set alert information (Alert message to be sent to a centralization server)

SaveDictionary() class write an array, containing blacklisted word information, on blacklist file. Class *loadDictionary()* Load blacklisted word list. *._set_Blacklist()* class

Initiate an array of the blacklisted list. *check_Blacklisted()* class will check whether logs sent from the user contain blacklisted word and return false if not. If a blacklisted word is detected *_setAlert()* class is called to set alert information.



Figure 4.6 (CILMS) UML Class Diagram for words blacklist

4.5.1.4. Development (words blacklist)

The screenshot below shows source code of Blacklisted class.

```

4 class Blacklisted:
5     def __init__(self, log):
6         self.log = log
7         self.blacklist_file = 'blacklist.csv'
8         self.DICTIONARY = {}
9         self.info = {}
10        self.BLACKLIST = []
11        self.str_found = ''
12        self.found_a_string = False
13        self._LoadDictionary('blacklist.txt')
14        self._set_blacklist()
15    def _set_blacklist(self):
16        for key in self.DICTIONARY:
17            word = self.DICTIONARY[key]['word']
18            self.BLACKLIST.append(word)
19    def _setAlert(self,word):
20        for key in self.DICTIONARY:
21            if word in self.DICTIONARY[key]['word']:
22                self.info = {'word': word,'priority':self.DICTIONARY[key]['priority'],
23                    'Alert':self.DICTIONARY[key]['Alert'],
24                    'Application':self.DICTIONARY[key]['Application']}
25                return self.info
26                #print key
27    def check_blacklisted(self):
28
29    def SaveDictionary(self,dictionary, File):
30        with open(File, 'wb') as myFile:
31            pickle.dump(dictionary,myFile)
32            myFile.close()
33    def _LoadDictionary(self,File):
34        with open(File,"rb") as myFile:
35            dict = pickle.load(myFile)
36            myFile.close()
37            self.DICTIONARY = dict
38            return dict

```

Figure 4.7: (CILMS) Screenshot showing source code of Blacklisted class.

4.5.1.5. Testing word blacklist sprint

Successful test result for a word blacklist sprint as illustrated in a screenshot in Figure 4.8. Screenshot below shows a successful test result for word blacklist sprint. The word blacklist module was developed and tested. The module worked as expected and was able to detect blacklisted words

```

C:\Python27\Auto_logger\blacklisted>python Blacklisted.py
saving dictionary list... /n
starting time = 3.94821209164e-06
Dictionary saved successfully
Time take to save Dictionary file 0.00073373573511
Word blacklisted saved successfully
Word blacklisted loaded successfully
['Manage Accounts', 'Create New Account', 'Change Your Password', 'Change an Acc
out', 'Install', 'Change Account Type', 'AutoPlay', 'Tor', 'User Account Contro
l', 'Hack', 'bitcoin', 'Cain & Abel', 'Delete Multiple items', 'Copying', 'Folde
r Access Denied', 'Microsoft Windows']
-----end word listed -----
Check if Tor is blacklisted
Blacklisted word found =>Tor
Setting Alert message....
Alert message array =>
{'priority': 1, 'Application': 'torbrowser', 'word': 'Tor', 'Alert': 'Anonymous
Browsing'}

```

Figure 4.8: (CILMS) Screenshot showing a successful test result for word blacklist sprint.

Blacklisted words are written to the blacklist file successfully.

Load blacklisted word list from the blacklist file working perfectly

From the test above, we have set “Tor” as blacklisted word, then run the check_blacklisted() function using “Tor”. Tor is detected and alert message is set successfully

Set alert function working perfectly

4.5.2 Sprint 2: Keystroke Recorder

4.5.2.1 Sprint planning

Functionality of Keystroke Recorder

- Listen to user keystrokes
- Record keystrokes

4.5.2.2. Keystroke Recorder Design

UML Class Diagram

Figure 4.6.2.1 Shows UML Class Diagram for words blacklist. It includes classes *HookManager()*, *hookKeyboard()* and *save_log()*.

HookManager() registers and manage callbacks for low level mouse and keyboard events. hookKeyboard() begin watching for keyboard events. Save_logs() class save keystrokes from users keyboard.

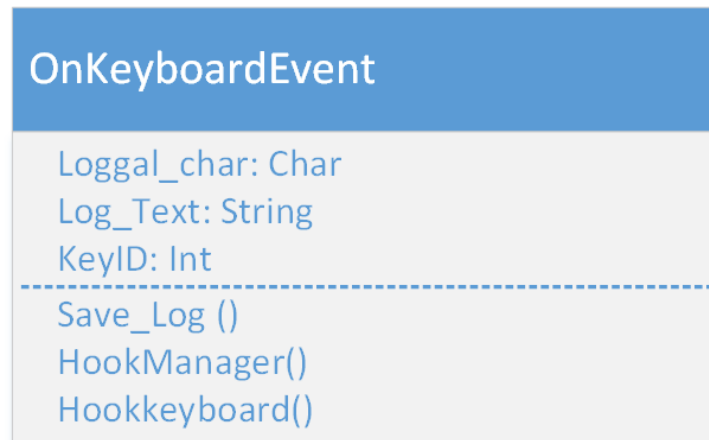


Figure 4.9: (CILMS) Uml class diagram for onkeyboard event

4.5.2.3. Development (Keystroke Recorder)

The keystroke recorder module was developed and tested. The module worked as expected and the section of code for this module is a shown in figure...

Figure 4.10 show source code of OnKeyboardEvent() function that record keystrokes.

```
def OnKeyboardEvent(event):
    global LOGGED_CHARS, LOG_NEWACTIVE, LOG_ACTIVE, LOG_TEXT, LOG_THREAD_ss, SEND_LOGS, START_TIME

    #check for new window activation
    wg = win32gui
    LOG_NEWACTIVE = wg.GetWindowText(wg.GetForegroundWindow())
    print LOG_NEWACTIVE
    #print LOG_NEWACTIVE
    if LOG_NEWACTIVE != LOG_ACTIVE:
        if event.KeyID == 13:
            text = '\r\n'
        elif event.KeyID == 165:
            text = '[Alt]'
        elif event.KeyID == 8:
            text = '[Back]'
        elif event.KeyID == 16:
            text = ''
        elif event.KeyID == 37:
            text = '[LARROW]'
        elif event.KeyID == 39:
            text = '[RARROW]'
        elif event.KeyID == 38:
            text = '[UARROW]'
        elif event.KeyID == 40:
            text = '[DARROW]'
        elif event.KeyID == 91 or event.KeyID == 92:
            text = '[WINKEY]'
        else:
            if chr(event.Ascii) in string.printable:
                text = chr(event.Ascii)
            else:
                text = ''
```

Figure 4.10 (CILMS) source code of OnKeyboardEvent() function that record keystrokes.

4.5.2.4. Testing and Debugging Keystroke Recorder splint

We tested the module by adding a code that print the first characters typed. The screenshot below shows a notepad that we used to type words that should be printed in the command prompt. The test produced positive results.

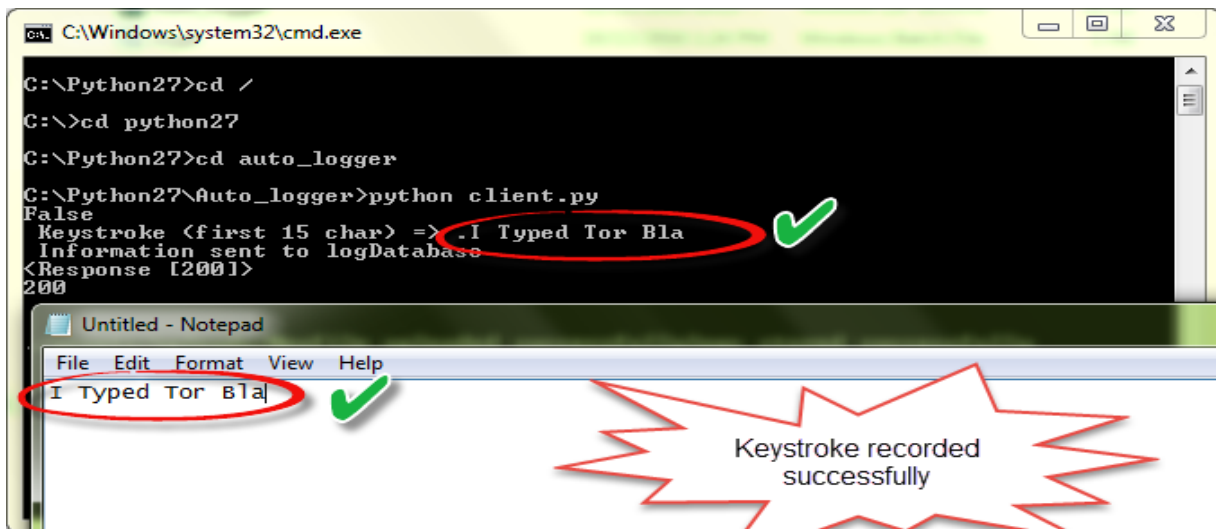


Figure 4.11: CILMS Keystrokes recorded successfully

4.5.2.5. Sprint Review

onKeyboardEvent() function can be used to capture keystrokes which will later be reviewed to monitor malicious activity

4.5.3 Sprint 3: Windows open recorder

4.5.3.1. Sprint planning

Windows open recorder should get foreground window, get window title and then record window title for analyses

4.5.3.2. Windows open recorder Design

UML Class Diagram

Figure 4.12 Shows UML Class Diagram for words blacklist. It includes classes `getWindowText()`

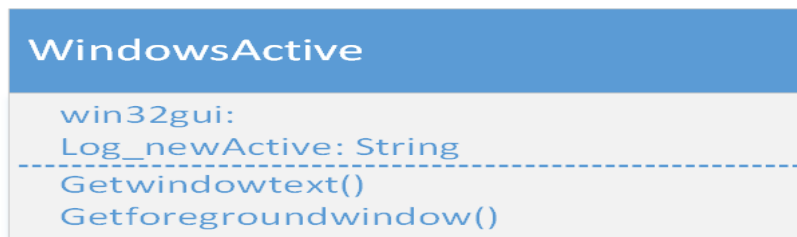


Figure 4.12: CILMS UML class diagram for windows open recorder

4.5.3.3. Development (Windows open recorder)

Windows open recorder was developed and the section of code for its development is shown in Figure 4.13 below

```
import win32gui
import time
class WindowsOpenRecorder():
    def __init__(self):
        self.wg = win32gui
    def printActiveWindow(self):
        LOG_NEWACTIVE = self.wg.GetWindowText(self.wg.GetForegroundWindow())
        print 'Active windodw => '+str(LOG_NEWACTIVE)

winactive = WindowsOpenRecorder()
while True:
    winactive.printActiveWindow()
    time.sleep(3)|
```

Figure 4.13. (CILMS) Development (windows open recorder)

4.5.3.4: Testing and Debugging Windows open recorder sprint

After developing windowsOpenRecorder class that print out titles of foreground window, we tested it by opening different windows the test results are shown in the screenshot in Figure 4.14 below.

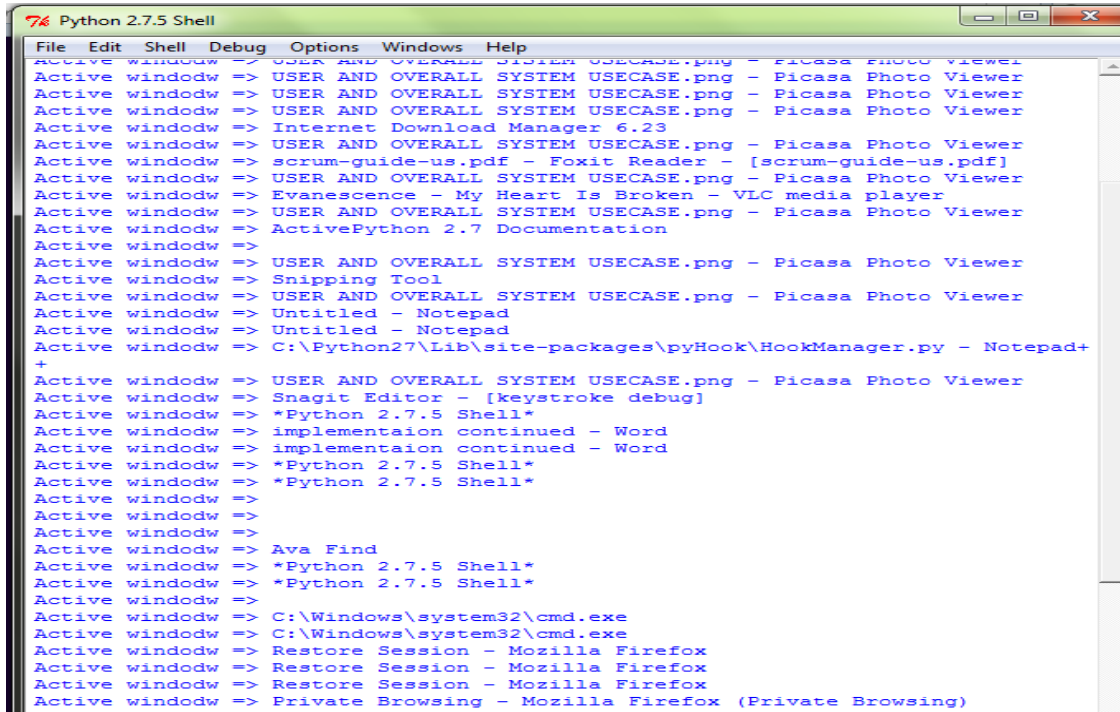


Figure 4.14 (CILMS) Testing and Debugging Windows open recorder print **Sprint Review**

Title of the windows open can be used to track what user's activities.

4.5.4. Sprint 4: Directory changes watchdog

4.5.4.1. Sprint planning

Directory changes watchdog uses watchdog python module to monitor directories specified as command-line arguments and logs events generated:

4.5.4.2. Directory changes watchdog Design

Activity Diagram

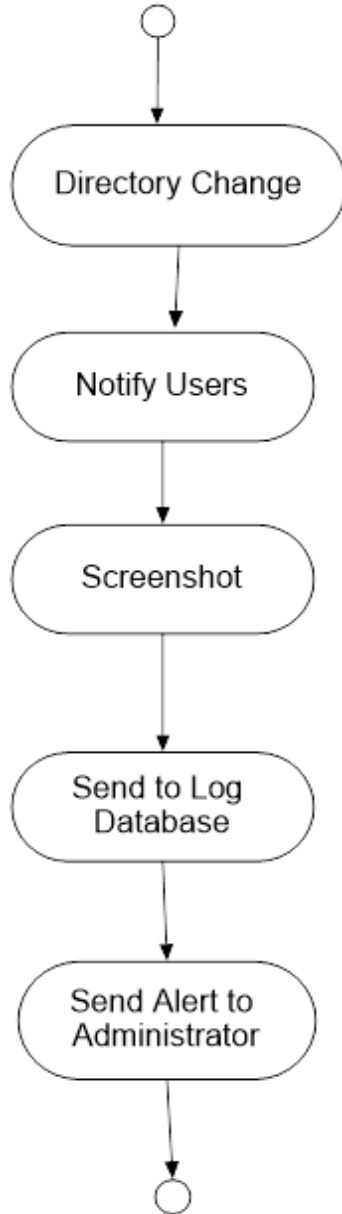


Figure 4.15 CILMS Activity Diagram for Directory changes watchdog

4.5.4.3. UML Class Diagram

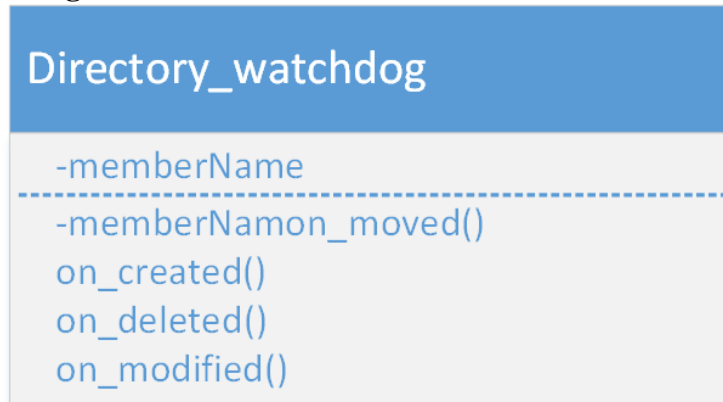


Figure 4.16 (CILMS) UML class Diagram for directory changes watchdog

4.5.4.4. Development (Directory changes watchdog)

Directory changes modules was developed and tested its source code is shown in the Figure 4.17 below

```

1 |import sys
2 |import time
3 |import logging
4 |from watchdog.observers import Observer
5 |from watchdog.events import LoggingEventHandler
6 |from watchdog.events import FileSystemEventHandler
7
8 |class MyHandler(FileSystemEventHandler):
9 |    def on_moved(self, event):
10 |        print event.dest_path
11 |        print event.event_type
12 |        print event.src_path
13 |        what = 'directory' if event.is_directory else 'file'
14 |        print "Moved %s: from %s to %s", what, event.src_path, event.dest_path
15
16 |    def on_created(self, event):
17 |        print 'Event type = '+event.event_type
18 |        what = 'directory' if event.is_directory else 'file'
19 |        print("Created %s: %s", what, event.src_path)
20
21 |    def on_deleted(self, event):
22 |        print 'Event type = '+event.event_type
23 |        what = 'directory' if event.is_directory else 'file'
24 |        print "Deleted %s: %s", what, event.src_path
25
26 |    def on_modified(self, event):
27 |        print 'Event type = '+event.event_type
28 |        what = 'directory' if event.is_directory else 'file'
29 |        print "Modified %s: %s", what, event.src_path
30
31
32 |if __name__ == "__main__":
33 |    event_handler = MyHandler()
34 |    observer = Observer()
35 |    observer.schedule(event_handler, path='.', recursive=False)
    
```

Figure 4.17 :(CILMS) Development of Directory changes watchdog

4.5.4.5. Testing and Debugging Directory changes watchdog splint

Directory changes watch dog was developed and after its development it was tested and debugged and the results were positive as shown in Figure 4.18 below

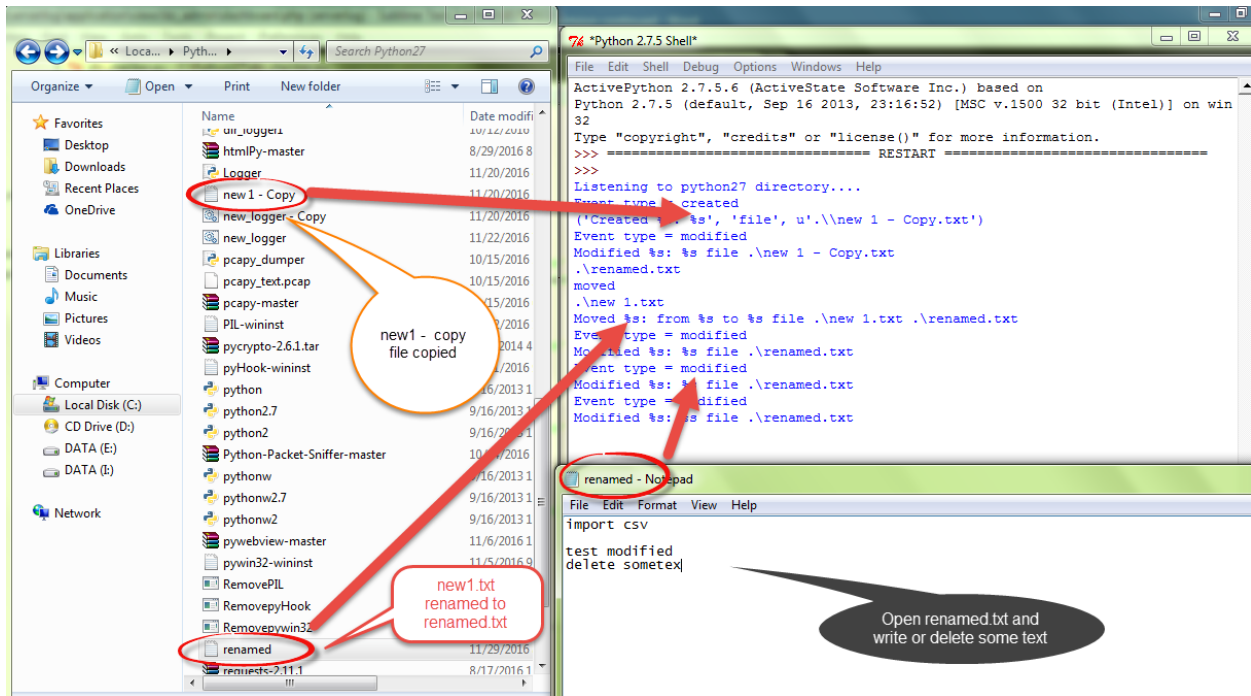


Figure 4.18. (CILMS) Testing and Debugging Directory changes watchdog sprint

4.5.5.0. Sprint 5: Screenshot Capture

4.5.5.1. Sprint 5 planning

Screenshot Capture Design

4.5.5.2. UML Class Diagram



Figure 4.19 (CILMS) UML class diagram for screenshot

4.5.5.3. Development (Screenshot Capture)

Screenshot capture module was developed and the source code is shown in the figure 4.20 below

```
1 #####
2 ## Author: Kamita Paul Kinuthia pkinuthia10@gmail.com
3 #####
4 import os
5 import Image, ImageGrab
6
7 def Screenshot(info):
8     global LOG_NEWACTIVE, USERNAME, HOST, LOGGED_CHARS,SESSION_ID
9     img = ImageGrab.grab()
10    saveas = os.path.join('C:\\Users\\Public\\publics\\'+time.strftime('%Y_%m_%d_%H_%M_%S')+'.png')
11    img.save(saveas)
12    addFile = str(saveas)
13
14
```

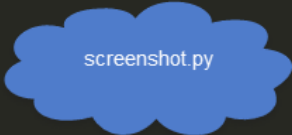


Figure 4.20: (CILMS) screenshot capture of development

4.5.5.4. Testing and Debugging Screenshot Capture sprint

After development of the screenshot capture model, the module was tested and debugged successfully as shown in Figure 4.21 below.

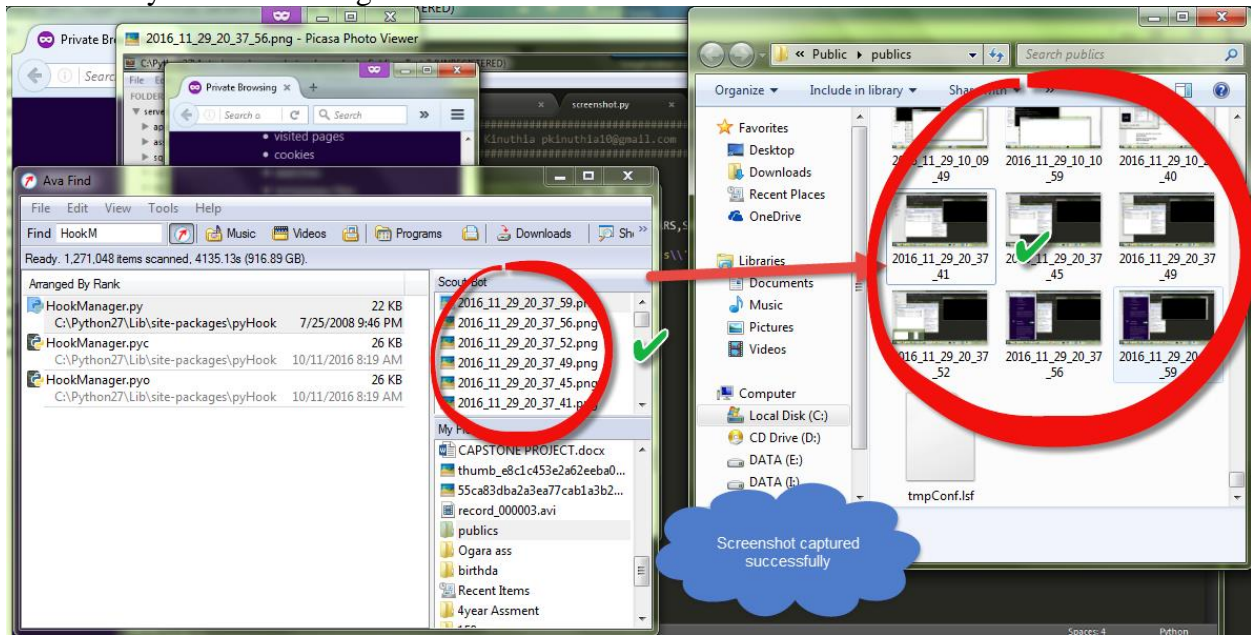


Figure 4.21 (CILMS) Testing and debugging screenshot capture sprint

4.5.6. Sprint 6: system log database

4.5.6.1. Sprint 6 planning

System log database consist of the following tables;

- CILMS_user table (stores user logins and logins activities) and
- CILMS_logs table (store logs sent by CILMS agent applications)

4.5.6.2. Administrator Dashboard Design

UML database Notation



Figure 4.22: UML Database Notation

4.5.6.3. Development (Administrator Dashboard)

Figure 4.23 below shows mysql query for creating CILMS_logs table.

```
-- -----  
-- author Paul Kinuthia : pkinuthia10@gmail.com|  
--  
-- Table structure for table `CILMS_logs`  
--  
CREATE TABLE IF NOT EXISTS `CILMS_logs` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `SESSION_ID` varchar(200) NOT NULL,  
  `logs` longtext NOT NULL,  
  `log_type` varchar(70) NOT NULL DEFAULT 'text',  
  `word` varchar(70) NOT NULL,  
  `alert` text NOT NULL,  
  `application` varchar(120) NOT NULL,  
  `priority` int(11) NOT NULL,  
  `windows` text NOT NULL,  
  `username` varchar(120) NOT NULL,  
  `host` varchar(120) NOT NULL,  
  `host_id` int(11) NOT NULL,  
  `is_read` smallint(6) NOT NULL DEFAULT '0',  
  `guid` varchar(150) NOT NULL,  
  `datetime` datetime NOT NULL,  
  `timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=568 ;  
--
```

Figure 4.23. (CILMS) mysql query for creating CILMS_logs table

Mysql query for creating CILMS_users table was created as illustrated in Figure4.24 below shows.

```
-- -----  
-- author Paul Kinuthia : pkinuthia10@gmail.com  
--  
-- Table structure for table `CILMS_users`  
--  
  
CREATE TABLE IF NOT EXISTS `CILMS_users` (  
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,  
  `email` varchar(100) NOT NULL,  
  `pass` varchar(64) NOT NULL,  
  `username` varchar(100) DEFAULT NULL,  
  `banned` tinyint(1) DEFAULT '0',  
  `last_login` datetime DEFAULT NULL,  
  `last_activity` datetime DEFAULT NULL,  
  `date_created` datetime DEFAULT NULL,  
  `forgot_exp` text,  
  `remember_time` datetime DEFAULT NULL,  
  `remember_exp` text,  
  `verification_code` text,  
  `totp_secret` varchar(16) DEFAULT NULL,  
  `ip_address` text,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=3 ;
```

Figure 4.24. (CILMS) mysql query for creating CILMS_users table

4.5.6.4. Testing and Debugging Administrator Dashboard Sprint

Testing and debugging of the administrator dashboard sprint was done and the positive results that came out on successful completion has been illustrated in Figure 4.25 showing database tables screenshots captured from phpMySql dashboard

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(11)			No	None	AUTO_INCREMENT	Change Drop Primary Unique Index More
2	SESSION_ID	varchar(200)	utf8_general_ci		No	None		Change Drop Primary Unique Index More
3	logs	longtext	utf8_general_ci		No	None		Change Drop Primary Unique Index More
4	log_type	varchar(70)	utf8_general_ci		No	text		Change Drop Primary Unique Index More
5	word	varchar(70)	utf8_general_ci		No	None		Change Drop Primary Unique Index More
6	alert	text	utf8_general_ci		No	None		Change Drop Primary Unique Index More
7	application	varchar(120)	utf8_general_ci		No	None		Change Drop Primary Unique Index More
8	priority	int(11)			No	None		Change Drop Primary Unique Index More
9	windows	text	utf8_general_ci		No	None		Change Drop Primary Unique Index More
10	username	varchar(120)	utf8_general_ci		No	None		Change Drop Primary Unique Index More
11	host	varchar(120)	utf8_general_ci		No	None		Change Drop Primary Unique Index More
12	host_id	int(11)			No	None		Change Drop Primary Unique Index More
13	is_read	smallint(6)			No	0		Change Drop Primary Unique Index More
14	guid	varchar(150)	utf8_general_ci		No	None		Change Drop Primary Unique Index More
15	datetime	datetime			No	None		Change Drop Primary Unique Index More
16	timestamp	timestamp			No	CURRENT_TIMESTAMP		Change Drop Primary Unique Index More

Figure 4.25: (CILMS) database tables screenshot from phpMySql

CILMS logs table shown in Figure 4.26

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(11)		UNSIGNED	No	None	AUTO_INCREMENT	Change Drop Primary Unique Index Spatial More
2	email	varchar(100)	utf8_general_ci		No	None		Change Drop Primary Unique Index Spatial More
3	pass	varchar(64)	utf8_general_ci		No	None		Change Drop Primary Unique Index Spatial More
4	username	varchar(100)	utf8_general_ci		Yes	NULL		Change Drop Primary Unique Index Spatial More
5	banned	tinyint(1)			Yes	0		Change Drop Primary Unique Index Spatial More
6	last_login	datetime			Yes	NULL		Change Drop Primary Unique Index Spatial More
7	last_activity	datetime			Yes	NULL		Change Drop Primary Unique Index Spatial More
8	date_created	datetime			Yes	NULL		Change Drop Primary Unique Index Spatial More
9	forgot_exp	text	utf8_general_ci		Yes	NULL		Change Drop Primary Unique Index Spatial More
10	remember_time	datetime			Yes	NULL		Change Drop Primary Unique Index Spatial More
11	remember_exp	text	utf8_general_ci		Yes	NULL		Change Drop Primary Unique Index Spatial More
12	verification_code	text	utf8_general_ci		Yes	NULL		Change Drop Primary Unique Index Spatial More
13	totp_secret	varchar(16)	utf8_general_ci		Yes	NULL		Change Drop Primary Unique Index Spatial More
14	ip_address	text	utf8_general_ci		Yes	NULL		Change Drop Primary Unique Index Spatial More

Figure 4.26 CILMS logs table

4.5.7. Sprint: Administrator Dashboard

4.5.7.1. Sprint planning

Administrator should first login to access the system dashboard. Among the notification that the administrator should get include

- Risky users
- Number of sessions
- Risky applications

On clicking to each user session, administrator should be able to view user sessions, keystrokes, screenshots captures, application name, alert message and date and time stamp of the log activity.

4.5.7.2. Administrator Dashboard Design

Use Case Diagram

Uml diagram below shows a use case diagram of administrator dashboard.

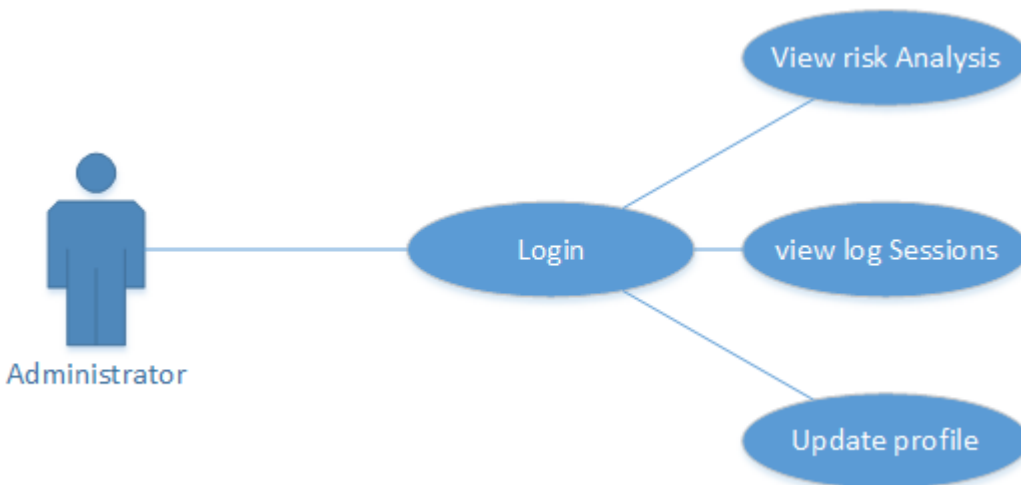


Figure 4.27: UML for administrator dashboard

4.5.7.3. Development (Administrator Dashboard)

The Administrator dashboard module was developed and tested. The module worked as expected and the section of screenshots for this module is shown in figure

Centralized Information Log Management Server

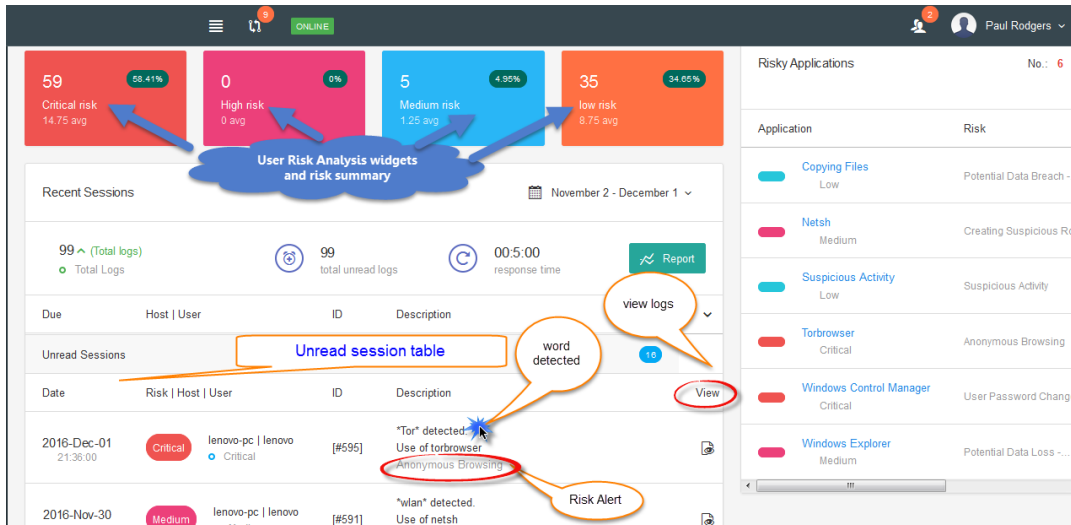


Figure 4.28: Development of administration dashboard

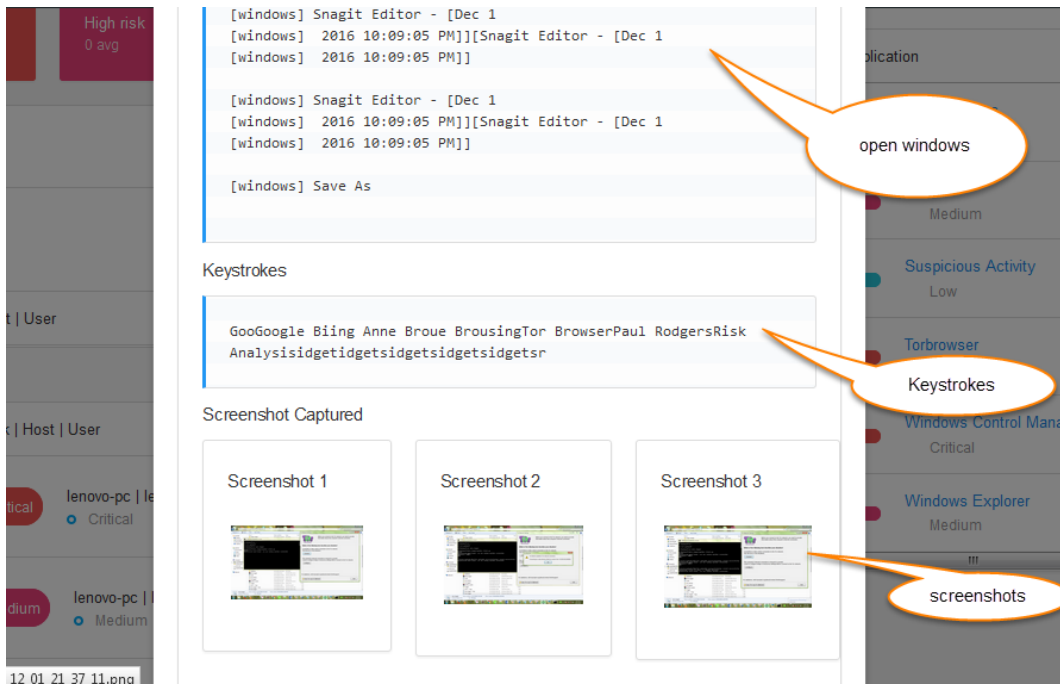


Figure 4.29: Administration dashboard testing and debugging

4.6. Testing

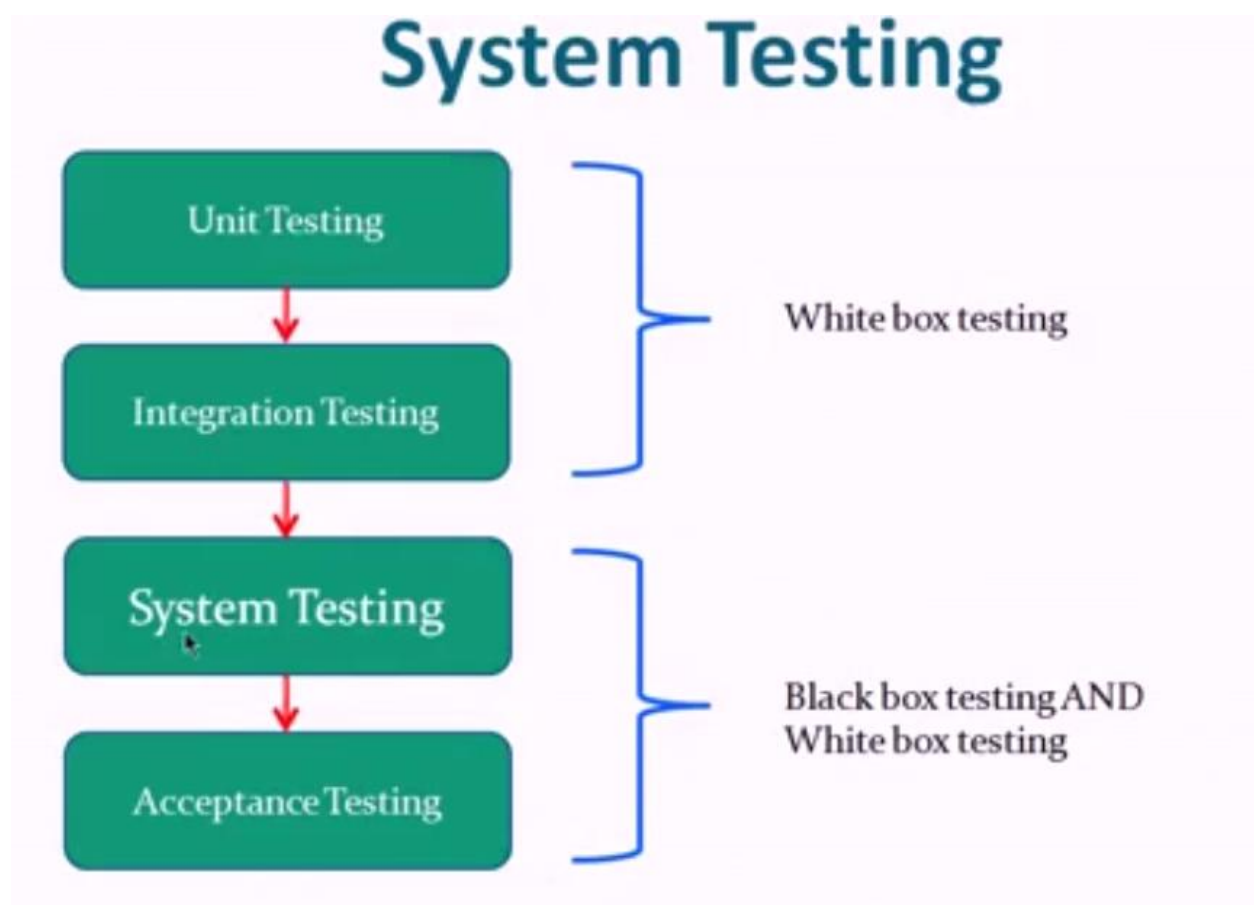
Testing and Validation

Primary purpose of testing is to detect software failures, so that defects may be discovered and corrected. It can also be stated as the process of validating and verifying that a software program or application or product:

- Meets the business and technical requirements
- Works as expected
- Can be implemented with the same characteristic

We will determine if the system complies with the requirements and performs functions for which it is intended and meets objectives.

Testing strategy employed during the testing include both black box and white box testing as shown in the figure below:



Unit testing involves testing the unit or element of the software. Since we used scrum model of agile methodology, **unit Testing** was done in each and every sprint. Interfaces and system functions were tested for proper information flow. Local data was examined to ensure that integrity was maintained.

4.6.1. Unit Testing

UNIT TESTING

Table 1 Unit test				
Purpose/ Objective: To perform Unit testing of CILMS				
UI NO.	Test Case	Expected Results	Observed Results	Status
1.	Keystroke Recording	It will record stoked words in command prompt.	The words typed by the keys were successfully recorded in the Command prompt	PASS
2.	Word Blacklist	It will detect certain blacklisted words upon typing	Words blacklisted were detected upon being typed	PASS
3.	Windows Open Recorder	It will detect title of the foreground window that has been opened	Opened windows were detected and their titles printed	PASS
4.	Directory Changes Watchdog	It will detect any modification in the files	Modification done on existing files were detected	PASS
5.	Screenshot Capture	It will capture screenshots	Screenshot captured successfully	PASS
6.	Administrator Dashboard	It will show risk summary, log sessions, risky applications and log analysis	Risk summary, log sessions, risky applications and log analysis showed by the administrator dashboard.	PASS
7.	Notification	It will notify users on prohibited activities on their system.	User notification successful a pop up notification appears	PASS

Table 4.1: Unit testing of CILMS

In this section we will document **integration testing**, **system testing** and **acceptance testing**.

4.6.2. Integration testing

Integration testing involved combining the different tested units and see if their integration points are working correctly

Integration Testing

Observed Results	Table 1 Integration			Status
Script sent to log database successfully	Purpose/ Objective: To perform integrated testing of CILMS			PASS
Alert sent to log database	SI NO.	Test Cases	Expected Results	PASS
Log sent to blacklist successfully	1.	Directory Changes	It will send a script to Log database upon change in directory	PASS
User received a notification	2.	Blacklist	It will send an alert to Log Database	PASS
Screenshot taken and sent to log database	3.	Windows Open	It send title of the foreground window to blacklist to check if log contains blacklisted word	PASS
Keystrokes were sent to blacklist successfully	4.	Notification	If blacklisted word is detected a user receives a notification	PASS
Log database received alert from blacklist, directory changes and taken screenshots	5.	Screenshot	If blacklisted word is detected a screenshot of opened window is taken	PASS
	6.	Onkeyboard Event	Keystrokes will be sent to blacklist	
	7.	Log Database	It will receive alert from blacklist, directory changes and taken screenshots	

Table 4.2: Integration Testing of CILMS

4.6.3. System testing

System Testing is done after integration testing is completed. It involved testing functional and nonfictional requirements.

Sanity Testing

We did a sanity test in our system to prove that our system was functioning according to the specifications, upon doing the sanity test we got positive results on the functioning of our system, functionalities such as connectivity to the log database, notification alert, alert on directory changes, all these proved to be functioning well according to our sanity test. The system can be started; administrator is able to log in such functionalities proved to be working

4.6.4. Security Testing

We used Advanced Encryption Standard, Symmetric or secret-key ciphers use the same key for encrypting and decrypting, so both the sender and the receiver must know and use the same secret key. All key lengths are deemed sufficient to protect classified information up to the "Secret" level with "Top Secret" information requiring either 192- or 256-bit key lengths

1. Confidentiality

Measures undertaken to ensure confidentiality are designed to prevent sensitive information from reaching the wrong people, while making sure that the right people can in fact get it: Access must be restricted to those authorized to view the data in question

the feature and function work completely. target 100% of required function work. priority must have

2. Integrity

Data should be secure, user name password sensitive information, session info encryption ,priority must have. Integrity involves maintaining the consistency, accuracy, and trustworthiness of data over its entire life cycle. Data must not be changed in transit, and steps must be taken to ensure that data cannot be altered by unauthorized people (for example, in a breach of confidentiality). These measures include file permissions and user access controls

3. Availability

Should be able t work in different platforms (Mozilla, internet explorer) It's also important to keep current with all necessary system upgrades. Providing adequate communication bandwidth and preventing the occurrence of bottlenecks are equally important. Redundancy, failover, RAID even high-availability clusters can mitigate serious consequences when hardware issues do occur. Fast and adaptive disaster recovery is essential for the worst case scenarios; that capacity is reliant on the existence of a comprehensive disaster recovery plan (DRP). Ensure high percentage uptime of the system always

CHAPTER FIVE

DISCUSSIONS, CONCLUSIONS AND RECOMMENDATIONS

5.1. Introduction

The section contains a brief overview of how the project performed in relation to objectives, how it performed in real life and closing statements on the project

5.2. Discussion

The main objective of this project is to implement a centralized information system log server from which keystrokes directory changes and windows open will reveal activities and programs being run by the user and also detect directory changes on the system user machine. This has been achieved using keystrokes listener to listen to the keys being stroked by the user and comparing them with the blacklisted words in the dictionary resulting to an alert to the system administrator if any prohibited activity is detected. The system has a feature that is able to alert the administrator once a prohibited window is opened by sending a screenshot of the window to the administrator, changing directories without the knowledge of the administrator is dealt with in this system as for all the directory changes the administrator must be alerted. Apart from monitoring Centralized Information Logs Server provides security awareness to system users by giving real time notification once a user is performing a prohibited action, this acts like a warning to the user.

The project has so far exceeded the objectives outlined in the proposal document with all primary objectives completed. In total four out of four objectives and the overall objectives was met. The performance of both the server and the client has been fully tested and are production quality and capable of being scaled with growth in its use, utilizing log management.

Even though log management provides centralized monitoring, it has been a problem to monitor individual activities from their client machines remotely, with the growing scale and complexity of the IT systems, system control has become more difficult for a system administrator to have an insight in every piece of details the system users are involved in, hence it would be handy if a powerful framework could be deployed where one could centrally monitor individual systems for specific blacklisted and prohibited activities.

Centralized Information Logs Server is one efficient tool that administrators could use to solve this problem as it listens to keystrokes and filters them, monitor directory changes and windows open and alerts the administrator once a blacklisted and prohibited activity has been detected Centralized Information System Logs Server is a Log Management system, log management is essential for security, accountability and for various information security compliances. This system has been written using python programming language. Python is a general-purpose language, which means it can be used to build just about anything, which will be made easy with the right tools/libraries. Professionally, Python is great for backend web development, data analysis, artificial intelligence, and scientific computing.

5.3. Conclusion

In conclusion the system developed has successfully completed its goal, as can be determined by the success of the discussion section, where the system the performance of both the server and the client has been fully tested and are production quality and capable of being scaled with growth in its use, utilizing log management. The project has enabled us to apply the knowledge we gained from project management and human computer interaction. One of the biggest skills we learnt from the project is time management, including planning the importance of planning during the early stages of development and amount of time that must be allotted to perform refactoring when knowledge improves at later date. Teamwork was also a concept we applied for the successful completion of this project

We can see that Centralized Information System Log Server is a powerful IT resource and system management tool. Its elastic property i.e. can be used to monitor a wide range of system use and also detect wide range of illegal activities within the system from keystrokes to windows open to directory changes and notifies user on security awareness. The most important benefit of using CILMS is the that system administrators can use it to detect insiders attack and create security awareness to the users who may be involved in executing malicious activity. Centralized log application also presents the real-time alerting, filtering and management. This feature provides a more user friendly interface to centralized log and understands the situation of the centralized information system log server it incorporates a host of powerful features including filtering based on message content, as well as analysis capabilities. Apart from monitoring CILMS can keep storage requirements static thus minimizing support staff cost as having the staff manually archive files is inefficient, error prone and waste of resources this will help avoid logistical problems searching a single transaction by logging into each server and searching through each.

5.4. Problems encountered during the project and how we overcame them

Short deadlines

The odds of successfully completing our project under the provided short time were generally not practical, to curb this problem we chose to use SCRUM methodology to do our project using backlogs for better realization of work schedules and required work, we also had to cope with school homework and had to save time for the project work because we knew it was an important unit.

Exhaustion

Working on the short deadlines made it hard for us and so exhaustion and fatigue came in, we handled this by working as a team and supported each other on the course of the project

Insufficient resources

We lacked sufficient resources to run the better part of the project efficiently however we managed to handle that by defining our needs properly and assigning prioritized resources throughout the duration of our project

Financial Constraints

It proved costly for instance we had to print document per chapter whenever we visited our supervisor this proved costly, we overcame this thanks to our supervisor who advised us to start submitting soft copies for assessment.

5.5. Future Work

In future we plan to develop a system that runs on different operating systems and different platforms

In future we plan to enable system restart when user kills it on the task manager

APPENDICES

Appendix (I)

Project Charter

Project charter contains a summary of the whole project plan, overview information, definition of the project and guide for the project activities.

Project Title: Centralized Information System Logs server (CIMSLS)				
Project Start Date: 18 th October, 2016 Projected Finish Date: December 2 nd , 2016				
Budget Information: The team estimates Ksh.600, 000 for this project. The majority of costs for this project will be internal labor. An initial estimate provides a total of 30 hours per week.				
Project Manager: Paul Kinuthia 0719739180. Pkinuthia10@gmail.com.				
Project Objectives: Develop a centralized information system log server from which keystrokes, directory changes and windows open will reveal activities and program run by the user and directory changes on the system user machine				
Approach:				
<ul style="list-style-type: none"> • Carry out a survey on how insiders attack are carried in order to determine which logs to monitor in order to decide whether to alert the administrator or not • Determine critical features of the new system. • Review the current alternatives in the market and decide a way forward for the system. • Do a comprehensive literature review? • Design all the system modules and draw models or prototypes of the same. • Develop the system (coding) based on proposed design. • Determine a way to measure the value of the system in terms alerting the user on a potential security breach by insider and creating security awareness to users. 				
Roles and Responsibilities:				
<i>Name</i>	<i>Role</i>	<i>Responsibility</i>	<i>Position</i>	<i>Contact Information</i>
Paul Abounji	Project Steward	Guiding and stewarding.	Lecturer	
Paul Kinuthia	Project Manager	Head of programming.	Student, B.Sc. Computer security and Forensics.	
Rodgers Okoth	Team Member	Head of Research and analysis.	Student, B.Sc. Computer security and Forensics	

Centralized Information Log Management Server

		Head of documentation		
Comments: (Handwritten or typed comments from above stakeholders, if applicable) “We will support this project as time allows.” All Members “We need to be extremely careful testing this new system, especially the security in giving it to the public and clients” Rodgers Okoth.				

Appendix (II)

SCOPE STATEMENT

Project Title: Centralized Information Log Management System (CILMS) Date: October 20 th , 2016 Prepared by: Rodgers Okoth, Head of Research and analysis, okoth364@gmail.com
Project Justification: Centralized Information system log server eases the network of every network administrator. It is a real log server tool used to centralize, secure stored log of platform that run on Information system log server also incorporates a host of powerful features including filtering based on message content, as well as analysis capabilities. Detection of malicious activity by system users will help elimination insider attack. Sending malicious logs to administrator will enhance accountability. The system will also create security awareness hence warning users against malicious activities.
Product Characteristics and Requirements: <ol style="list-style-type: none">1. Security2. Agent software (records user activity: keystroke, windows open and directory changes)3. Words blacklist4. Administrator accounts.5. Special accounts for other stakeholders.

Appendix (III)

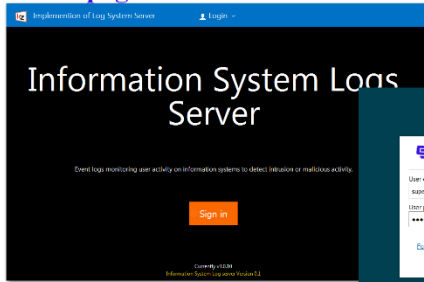
The Scrum Team

Name	Title	Role
Paul Abuonji	The Product Owner (supervisor)	<ul style="list-style-type: none"> • Maximizing the value of the product and the work of the Development Team • Clearly expressing Product Backlog items; • Ordering the items in the Product Backlog to best achieve goals and missions; • Optimizing the value of the work the Development Team performs; • Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next; and, • Ensuring the Development Team understands items in the Product Backlog to the level needed.
Kamita Paul Kinuthia	Scrum Master	leads the Scrum meetings, identifies the initial backlog to be completed in the sprint, and empirically measures progress toward the goal of delivering this incremental set of product functionality. The Scrum master ensures that everyone makes progress, records the decisions made at the meeting and tracks action items, and keeps the Scrum meetings short and focused.
Rodgers Okoth	Development Team	Designer /chief researcher
Paul Kinuthia		<i>Chief Programmer</i>

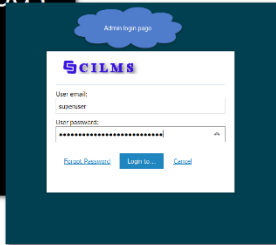
APPENDIX (IV)

CIMLS-SERVER APPLICATION SCREENSHOTS

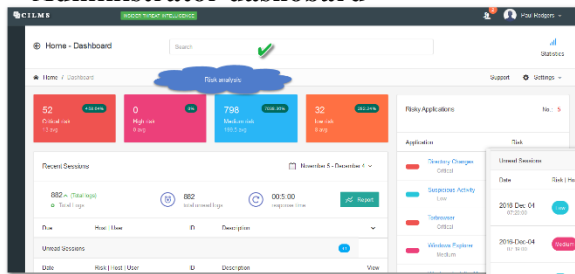
Home page



Login page



Administrator dashboard

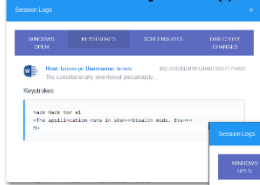


Logs table

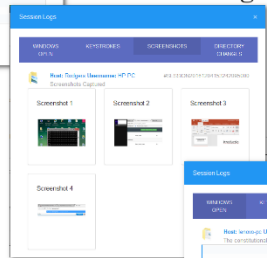
Date	Task	Host	User	ID	Description	View
2018-Dec-01 07:20:01	Setup	192.168.1.100	user	10000	"log" detected: Use of Remote Activity Surveillance Activity	View
2018-Dec-04 10:30:01	Admin	192.168.1.100	user	10001	"log" detected: Use of Windows Installer Manager Surveillance Activity: software installation	View
2018-Dec-04 10:30:01	Setup	192.168.1.100	user	10002	"log" detected: Use of Remote Activity Surveillance Activity	View
2018-Dec-01 07:20:01	Admin	192.168.1.100	user	10003	"log" detected: Use of Remote Activity Surveillance Activity	View
2018-Dec-04 10:30:01	Admin	192.168.1.100	user	10004	"log" detected: Use of Remote Activity Surveillance Activity	View

Click on view icon to see a pop window that shows you logs for specific session logs

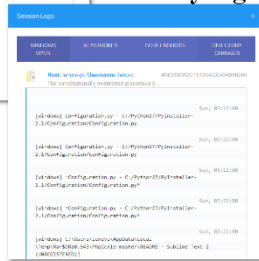
Windows open logs



Screenshots logs



Directory logs

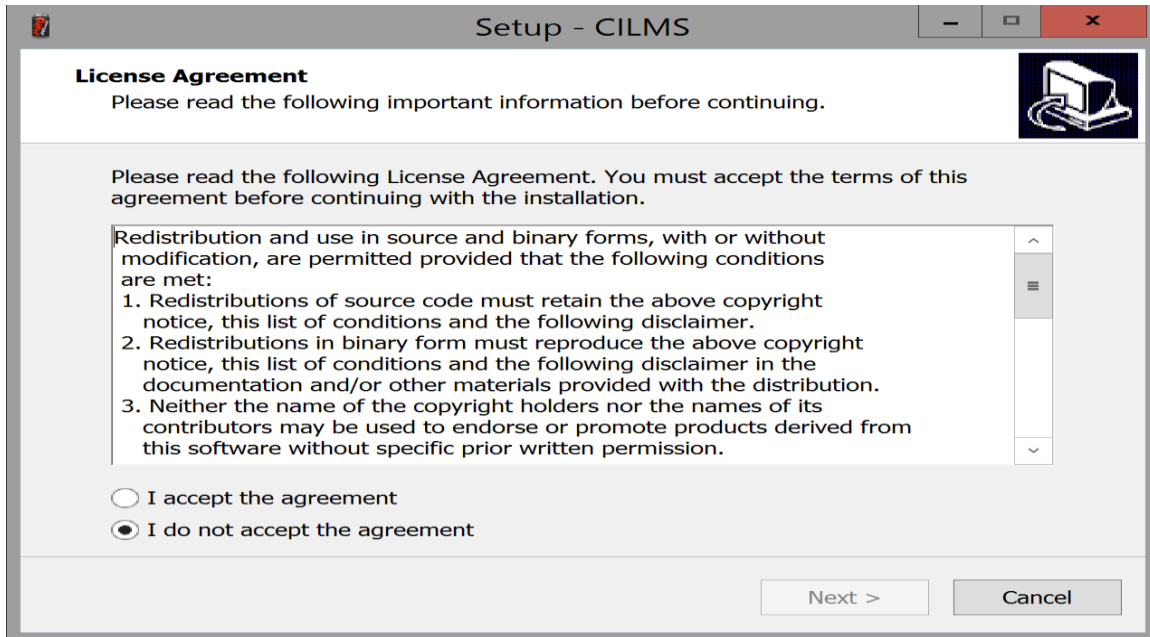


APPENDIX (V)

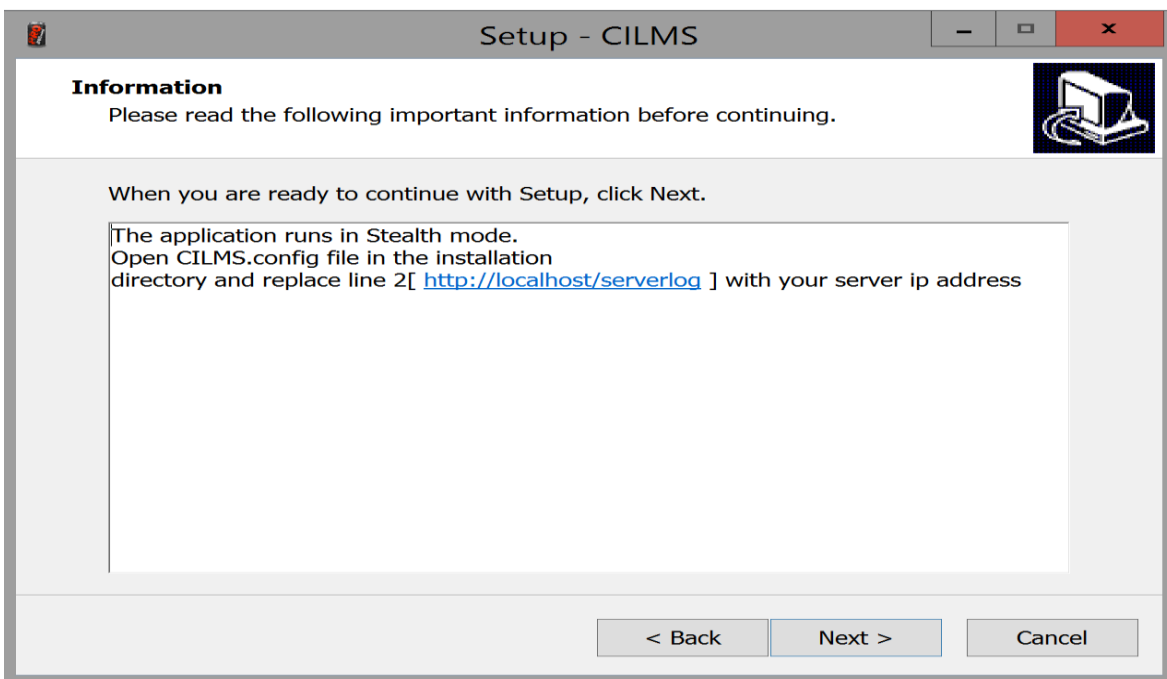
CLIENT USER MANUAL

Installation

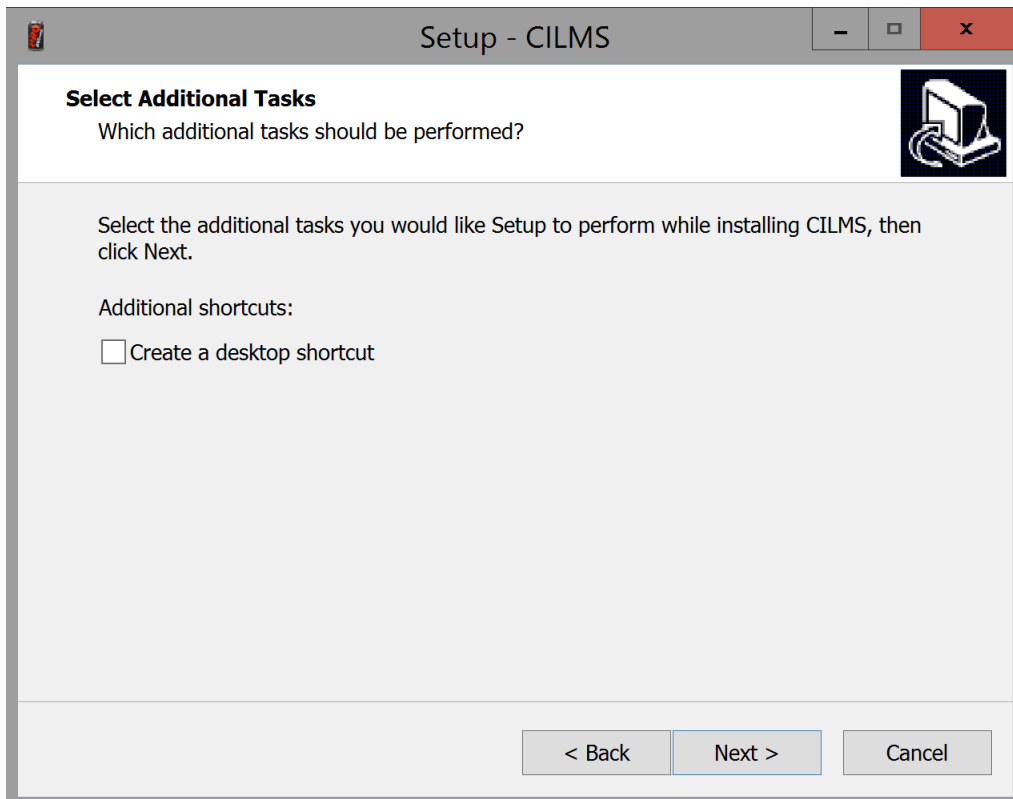
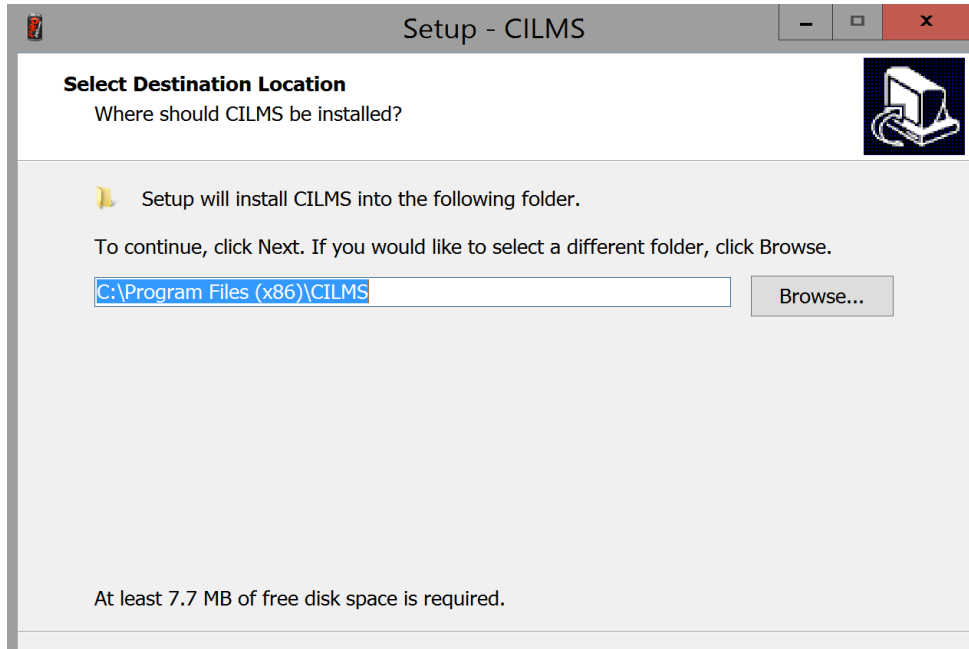
Read and understand the license agreement, then accept to activate the 'NEXT' button. Click on NEXT to proceed



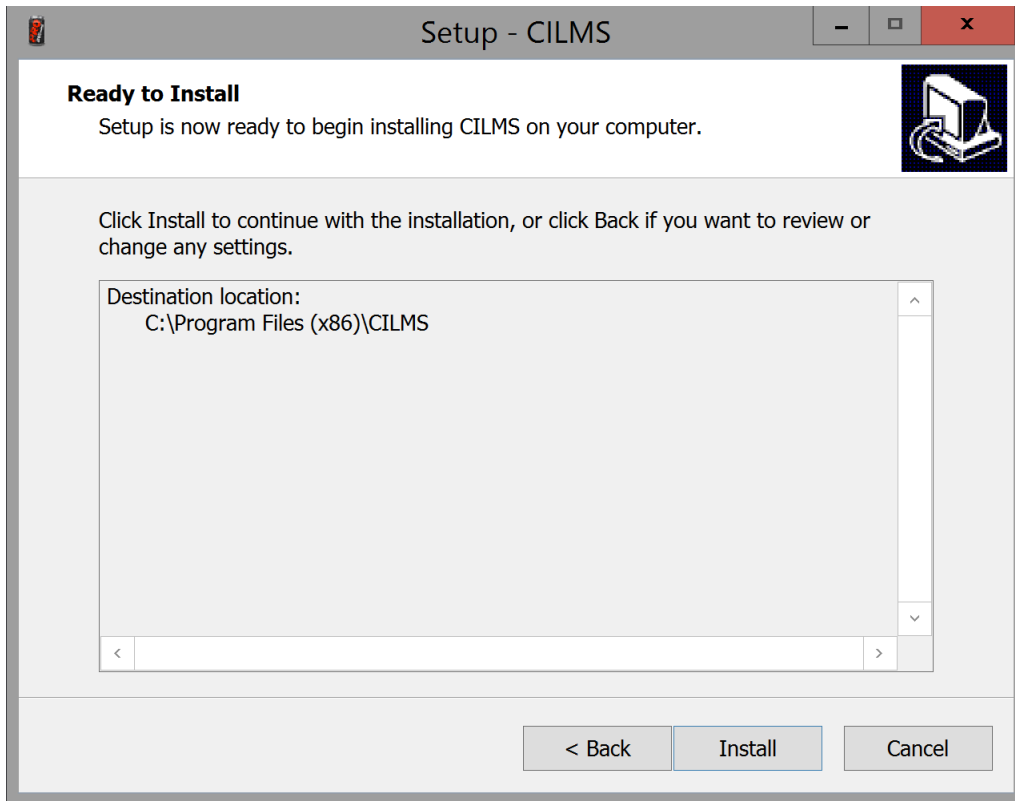
Read the given information then click on NEXT to proceed

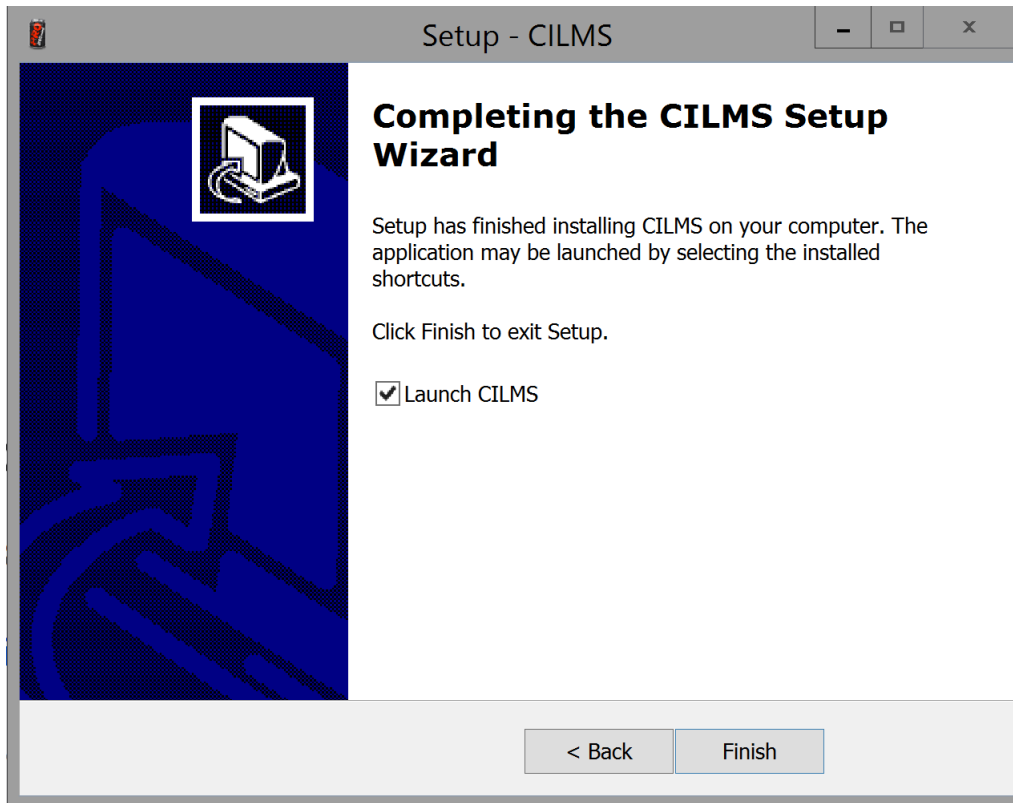


Centralized Information Log Management Server



Centralized Information Log Management Server





APPENDIX (VI)

CILMS-SERVER APPLICATION

Administrator user manual

Prerequisite Requirements

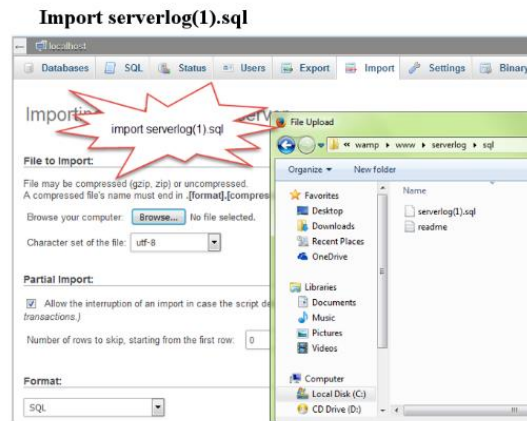
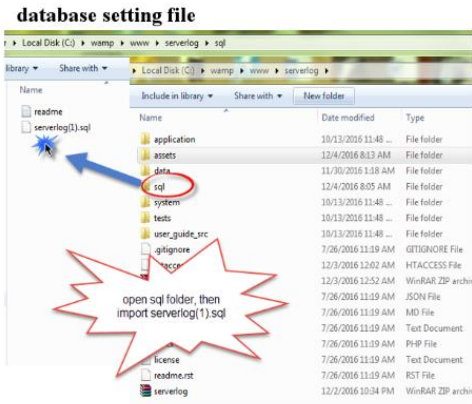
Database	Scripting language	Server
MySQL 5.6+	PHP 5.4+	Apache 2.4.+

Note:Tested on WAMP sever

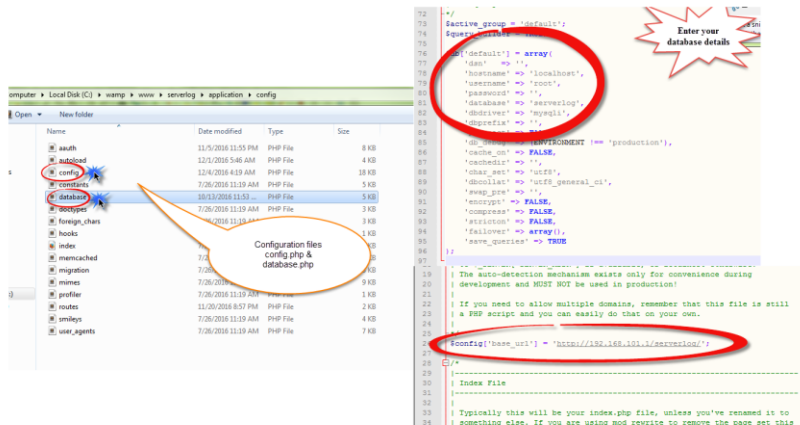
Server application

Open CILMS-server folder, then open sql folder to access the database sql file. Import the sql file to your database

Centralized Information Log Management Server



Copy CILMS-server folder to your public html folder (www folder for wamp server). Then open application/config folder and modify config.php (replace base_url with you're the url to your CILMS-server. Configure database.php with your database credentials



Administrator dashboard

Click login button, then login using **superuser** as username and **12345** as your password. You can now see risk analysis on the dashboard.

APPENDIX (VI)

PROGRAM CODES

Blacklisted word Source code

```
4 class Blacklisted:
5     def __init__(self, log):
6         self.log = log
7         self.blacklist_file = 'blacklist.csv'
8         self.DICTIONARY = {}
9         self.info = {}
10        self.BLACKLIST = []
11        self.str_found = ''
12        self.found_a_string = False
13        self._LoadDictionary('blacklist.txt')
14        self._set_blacklist()
15    def _set_blacklist(self):
16        for key in self.DICTIONARY:
17            word = self.DICTIONARY[key]['word']
18            self.BLACKLIST.append(word)
19    def _setAlert(self,word):
20        for key in self.DICTIONARY:
21            if word in self.DICTIONARY[key]['word']:
22                self.info = {'word': word,'priority':self.DICTIONARY[key]['priority'],
23                    'Alert':self.DICTIONARY[key]['Alert'],
24                    'Application':self.DICTIONARY[key]['Application']}
25                return self.info
26                #print key
27    def check_blacklisted(self):
28    def SaveDictionary(self,dictionary, File):
29        with open(File, 'wb') as myFile:
30            pickle.dump(dictionary,myFile)
31            myFile.close()
32    def _LoadDictionary(self,File):
33        with open(File,"rb") as myFile:
34            dict = pickle.load(myFile)
35            myFile.close()
36            self.DICTIONARY = dict
37            return dict
```

Onkeyboard Event source code

```
def OnKeyboardEvent(event):
    global LOGGED_CHARS, LOG_NEWACTIVE, LOG_ACTIVE, LOG_TEXT, LOG_THREAD_ss, SEND_LOGS, START_TIME

    #check for new window activation
    wg = win32gui
    LOG_NEWACTIVE = wg.GetWindowText(wg.GetForegroundWindow())
    print LOG_NEWACTIVE
    #print LOG_NEWACTIVE
    if LOG_NEWACTIVE != LOG_ACTIVE:
        if event.KeyID == 13:
            text = '\r\n'
        elif event.KeyID == 165:
            text = '[Alt]'
        elif event.KeyID == 8:
            text = '[Back]'
        elif event.KeyID == 16:
            text = ''
        elif event.KeyID == 37:
            text = '[LARROW]'
        elif event.KeyID == 39:
            text = '[RARROW]'
        elif event.KeyID == 38:
            text = '[UARROW]'
        elif event.KeyID == 40:
            text = '[DARROW]'
        elif event.KeyID == 91 or event.KeyID == 92:
            text = '[WINKEY]'
        else:
            if chr(event.Ascii) in string.printable:
                text = chr(event.Ascii)
            else:
                text = ''
```

Windows Open Source Code

```
import win32gui
import time
class WindowsOpenRecorder():
    def __init__(self):
        self.wg = win32gui
    def printActiveWindow(self):
        LOG_NEWACTIVE = self.wg.GetWindowText(self.wg.GetForegroundWindow())
        print 'Active window => '+str(LOG_NEWACTIVE)

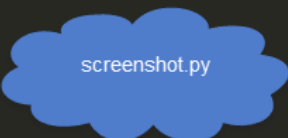
winactive = WindowsOpenRecorder()
while True:
    winactive.printActiveWindow()
    time.sleep(3)
```

Code for development of directory changes

```
1 import sys
2 import time
3 import logging
4 from watchdog.observers import Observer
5 from watchdog.events import LoggingEventHandler
6 from watchdog.events import FileSystemEventHandler
7
8 class MyHandler(FileSystemEventHandler):
9     def on_moved(self, event):
10         print event.dest_path
11         print event.event_type
12         print event.src_path
13         what = 'directory' if event.is_directory else 'file'
14         print "Moved %s: from %s to %s", what, event.src_path, event.dest_path
15
16     def on_created(self, event):
17         print 'Event type = '+event.event_type
18         what = 'directory' if event.is_directory else 'file'
19         print("Created %s: %s", what, event.src_path)
20
21     def on_deleted(self, event):
22         print 'Event type = '+event.event_type
23         what = 'directory' if event.is_directory else 'file'
24         print "Deleted %s: %s", what, event.src_path
25
26     def on_modified(self, event):
27         print 'Event type = '+event.event_type
28         what = 'directory' if event.is_directory else 'file'
29         print "Modified %s: %s", what, event.src_path
30
31
32 if __name__ == "__main__":
33     event_handler = MyHandler()
34     observer = Observer()
35     observer.schedule(event_handler, path='.', recursive=False)
```

Code for development of screenshot capture

```
1 #####
2 ## Author: Kamita Paul Kinuthia pkinuthia10@gmail.com
3 #####
4 import os
5 import Image, ImageGrab
6
7 def Screenshot(info):
8     global LOG_NEWACTIVE, USERNAME, HOST, LOGGED_CHARS,SESSION_ID
9     img = ImageGrab.grab()
10    saveas = os.path.join('C:\\Users\\Public\\publics\\'+time.strftime('%Y_%m_%d_%H_%M_%S')+'.png')
11    img.save(saveas)
12    addFile = str(saveas)
13
14
```



Source code for creating CILMS table

```
-- -----
-- author Paul Kinuthia : pkinuthia10@gmail.com
--
-- Table structure for table `CILMS_logs`
--
CREATE TABLE IF NOT EXISTS `CILMS_logs` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `SESSION_ID` varchar(200) NOT NULL,
  `logs` longtext NOT NULL,
  `log_type` varchar(70) NOT NULL DEFAULT 'text',
  `word` varchar(70) NOT NULL,
  `alert` text NOT NULL,
  `application` varchar(120) NOT NULL,
  `priority` int(11) NOT NULL,
  `windows` text NOT NULL,
  `username` varchar(120) NOT NULL,
  `host` varchar(120) NOT NULL,
  `host_id` int(11) NOT NULL,
  `is_read` smallint(6) NOT NULL DEFAULT '0',
  `guid` varchar(150) NOT NULL,
  `datetime` datetime NOT NULL,
  `timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=568 ;
--
```

mysql query for creating CILMS_users table

```
-- -----  
-- author Paul Kinuthia : pkinuthia10@gmail.com  
--  
-- Table structure for table `CILMS_users`  
--  
  
CREATE TABLE IF NOT EXISTS `CILMS_users` (  
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,  
  `email` varchar(100) NOT NULL,  
  `pass` varchar(64) NOT NULL,  
  `username` varchar(100) DEFAULT NULL,  
  `banned` tinyint(1) DEFAULT '0',  
  `last_login` datetime DEFAULT NULL,  
  `last_activity` datetime DEFAULT NULL,  
  `date_created` datetime DEFAULT NULL,  
  `forgot_exp` text,  
  `remember_time` datetime DEFAULT NULL,  
  `remember_exp` text,  
  `verification_code` text,  
  `totp_secret` varchar(16) DEFAULT NULL,  
  `ip_address` text,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=3 ;
```

REFERENCES

1. A. Cockburn, *Agile Software Development*, Addison-Wesley, Reading, MA, 2001.
2. Abad, C., Taylor, J., Sengul, C., Yurcik, W., Zhou, Y., & Rowe, K. 2003. Log correlation for intrusion detection: A proof of concept. In ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference, 255, Washington, DC, USA. IEEE Computer Society. Electronic version found at <http://www.ncassr.org/projects/sift/papers/ACSAC03.PDF> (Visited November.2016).
3. Choo, C., Betlor, B., & Turnbull, D. (1998). A behavioral model of information seeking on the Web: Preliminary results of a study of how managers and IT specialists use the Web. Paper presented at the 61st Annual Meeting of the American Society for Information Science, Pittsburgh, PA.
4. Bergadano, D. Cavagnino, P. Dal Checco, P. Andrea Nesta, M. Miragila, Secure logging for Irrefutable administration. 2005
5. G. Canbek, "Analysis, design and implementation of keyloggers and anti-keyloggers" Gazi University, Institute Of Science And Technology, M.Sc. thesis (in Turkish), Sept. 2005
6. Geoffrey Ingersoll. "Russia Turns To Typewriters To Protect Against Cyber Espionage". 2013.
7. J.F.Kelly, " An iterative design methodology for user-friendly natural language office information applications," *ACM Transactions on Information Systems* 1984
8. Ando, K Matsuura, A Baba An analysis of Ensuring order of log entries in distributed environment- Computer Security Symposium (CSS 2002), 2002
9. McIntyre, K.2003. Event correlation systems - the new threat frontline.
10. Monterey, CA: Defense Personnel Security Research Center. of the 19th Annual Computer Security Applications Conference, 255, Washington,

11. R Rinnan- 2005, Benefits of centralized log file correlation
12. Sharon . A. maneki "[Learning from the Enemy: The GUNMAN Project](#)"
12. Shaw, E.D. (2006). The role of behavioral research and profiling in malicious cyber insider investigations: Digital investigation. *The International Journal of Digital Forensics and Incident Response*, 3, 20-31. Exeter, UK: Elsevier.
13. Shaw, E.D., & Fischer, L.F. (2005). *Ten tales of betrayal: The threat to corporate infrastructures by information technology insiders: Analysis and observations*
- Shaw, E.D., Post, J.M., & Ruby, K. (1999). Profiling the dangerous IT professional. *Security Management*. 3(12).