**SECURITY INFORMATION AND EVENT MANAGMENT SYSTEM**

**BY**

**JOHN VAKUTE**

**NSC/PT2/CMP/0011/17/18**

**PROJECT 1**

**SUBMITTED TO**

**NASARAWA STATE UNIVERSITY,**

**FACULTY OF NATURAL AND APPLIED SCIENCE**

**MARCH, 2024**

**APPROVAL PAGE**

This research project has been read and approved

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Project supervisor) Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date

(Project Co-ordinator)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Head of Department) Date

**DECLARATION**

I hereby declare that this project has been writing by me and it is a report of my research work.

.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

JOHN VAKUTE

**DEDICATION**

This researched project is dedicated to Almighty God.

**ACKNOWLEDGEMENT**

I wish to express my profound gratitude to God. For preserving my life and for the wisdom and knowledge to accomplished this program.

I also appreciate my supervisor who despite his tight schedule guided tremendously throughout the period of this research work.

My appreciation also goes to my project coordinator and other lecturers who impacted knowledge to me. May God bless you all.

Finally, I appreciate my entire family for all their effort to catapult me to the land of success, may God grant all your hart desires in Jesus Name Amen.

**TABLE OF CONTENT**

Title Page………………………………………………………………….……i

Approval Page…………………………………………………….……………ii

Declaration……………………………………………………….………….…iii

Dedication………………………………………………………………………iv

Acknowledgement……………………………………………………………...v

Table of content…………………………………………………………………vi

Abstract…………………………………………………................................vii

**CHAPTER ONE: INTRODUCTION**

* 1. Background of the study………….………………………………………1
  2. Statement of the problem…….….……………………………………...3
  3. Aims of objective of the study……….………………………………….4
  4. Scope of the study ……………………………………………………….4
  5. Significant of the study…………………………………………………..5
  6. Research methodology………..............................................................6
  7. Organisation of the study………………………………………………...6
  8. Definition of Term………………………………………………………..7

**CHAPTER TWO: REVIEW LITERATURE**

* 1. Core Concept Information Security………………………………….…...8
  2. Existing Literature and Software on Security information managment.....9
  3. Comparison of Different Security Software…………………………..…14
  4. Conclusion……………………………..………………………………….14

**CHAPTER THREE: SYSTEM ANAYSIS AND DESIGN**

* 1. Introduction………………………………………………………………..….16
  2. Analysis of the existing system……………………………………………….16
  3. Analysis of the proposed system……………………………………………..16
  4. Proposed Methodology……………………………………………………….17
  5. System Design……………………………………….……………..….......18

CHAPTER FOUR: SYSTEM IMPLEMENTATION

4.1 Introduction

4.2 Choice of Development Tools

4.3 System Requirements

4.3.1 Software Requirements

4.3.2 Hardware Requirements

4.4 System Interface

4.5 System Testing.

4.5.1 Verifying the functionality of the Device identifier Module

4.5.2 Verifying the functionality of the Device Recognizer Module.

CHAPTER FIVER: SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

5.2 Conclusion.

5.3 Recommendation.

5.3.1 Recommendation for Device identifier Module

5.3.2 Recommendation for the log Normalizer Module

5.3.3 Recommendation for the log Consolidation Module

5.4 Reference

5.5 Appendix

References…………………………………………………………………....35

**ABSTRACT**

The Security Information and Event Management (SIEM) enhances the security management of an organization by storing and analyzing logs coming from different network devices and giving possible recommendations that can be warnings, notices or alarms. Companies are beginning to invest in SIEM to protect their data and to help network or system administrators monitor the state of their workplace. A lot of SIEM products focus on security tools and lack log consolidation and incident management solutions. In this study we propose an Adaptable Software-based Log Consolidation and Incident Management (AdLCIM), a type of SIEM that works on a typical Local Area Network (LAN) where various network devices report status to the system. The system is capable of collecting different logs coming from different, identified network devices. It is also capable of standardizing logs into its format, consolidates and correlates patterns through its inventories. All resolvable attack logs are event sniped, while non-resolvable logs are flagged as alerts. The system is capable of handling different scenarios with different devices, and tests result confirmed successful log analysis. The system, moreover, is capable in running for long durations of time to see if the system is capable of analyzing all the logs coming from different, identified network devices. Overall, the performance of the system came up with the correct and accurate results in verifying log analysis from different network devices having different scenarios

**CHAPTER FOUR**

**SYSTEM IMPLEMENTATION**

**4.1 INTRODUCTION**

Before a new system is implementation, it may be necessary to conduct through testing of data to ensure that the system works properly. At the implementation stage the conceptual/ design of the proposed system will be turned into a working system. The system would be thoroughly tested and designed to be explicate and less tedious so that any other users to the system can perform the specified operations for which the application was designed.

**4.2 Choice of Development Tools**

The programming language are developed with the primary objective of facilitating large number of people to use computer without the need to know in details the internal structure of the computer system. It is very good to choose a programming language that one can handle, create, manipulate, maintain, extract and summarizes the data records and files with provision for data manipulations, reliability, accurate integrity and security.

Programming language that was be used for this research project are Go 75.4%, TypeScript 11.0%, Shell 6.3%, HTML 5.8%, Ruby 0.9%, JavaScript 0.3%, Other 0.3%. These languages were chosen because of accessibility and availability of the language in the computing environment and which the system was developed. Also they has a powerful graphical user interface, which is user friendly and can effectively handle the system under study.

**4.3 System Requirements**

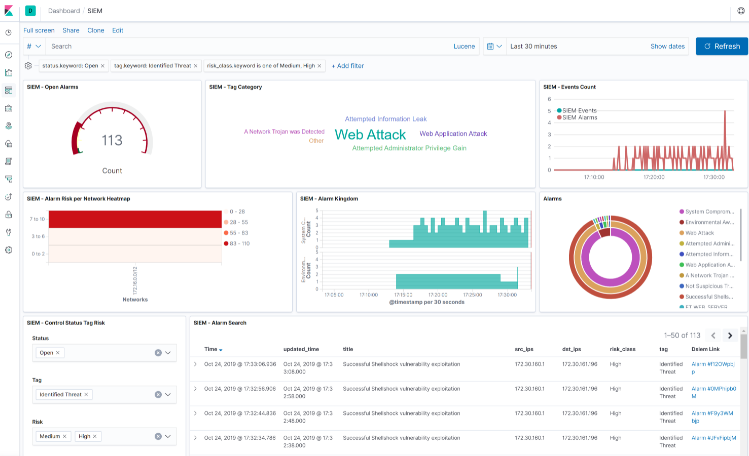
**4.3.1 Software Requirements**

* OSSIM: This software is a shareware SIEM version that is available to everybody. It is used to have a comparison of SIEM to the SIEM study that is implemented.
* WinSysLog: This is sample log consolidation software that can be used as a basis in creating software for study.
* Wireshark: It is a network protocol analyzer used to check packets passing through a network.

**4.3.2 Hardware Requirements**

* Mediation server: A computer which is used as the main server of the system where it analyzes and groups logs and gives recommendations.
* Network devices: Routers, switches or hubs that can be used to elicit logs for the software implemented.
* Workstation: Computers are used by users that are connected to network devices.

4.4 System Interface



System homepage

**4.5 System Testing**

Tests were done to verify the functionality of each module. Unit tests were done separately on each module.

**4.5.1 Verifying the Functionality of the Device Identifier Module**

This experiment tests if the network administrator can add, modify or delete network devices in the identified\_device table of the system’s database to be able to accept logs from it. If the network administrator chooses to add a network device to the database, the network device should be added successfully to the database. If the network administrator chooses to modify an existing identified network device, that network device should be successfully updated. Lastly, if the network administrator chooses to delete an existing identified device, that network device should be removed from the database.

**Table 6-1. Experiments on Adding Device Information**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Sequence** | **Navigation Path** | **Test Description** | **Data Input** | **Input Type** | **Expected Result** | **Added to database?** | **Successful Attempts / Total Attempts** |
| Add device with one field having a blank input | User is directed to the Device Manager tab. | User tries to add a device by supplying a hostname, device type, device specification but leaving the device details field blank. | User inputs DLSU-PC as hostname, COMPUTER as device type, WINDOWS as device specification and left the device details blank. | Invalid | User is prompted to fill all fields. | NO | 0 / 20 (0%) |
| Add device with complete informatio n | User is directed to the Device Manager tab. | User tries to add a device by supplying a hostname, device type, device specification and device details. | User inputs DLSU-PC as hostname, COMPUTER as device type, WINDOWS as device specification and XP as device details. | Valid | The device is added together with its other information to the identified devices inventory. | YES | 20 / 20 (100%) |

As seen in Table 6-1, two scenarios are done to see the functionality of the

module when it comes to adding devices. When the user left some of the fields empty, the system is able to prompt the user to inform him or her that some fields are left blank and that the he or she must fill up all the fields. On the other hand, when the user filled up all the required fields, the device, together with its information, is added successfully to the identified devices inventory.

**4.5.2 Verifying the Functionality of the Device Recognizer Module**

This experiment tests if the network administrator can successfully identify a desired network device. For example, a Windows computer, which is not found in the identified\_device table, tries to send a Syslog message to the system. The system should be able to detect that the Windows computer is not found in the identified\_device table and the Windows computer is considered as an unidentified device. Its Hostname/ IP address should be put in the unidentified\_device table. Furthermore, this experiment also tests whether the unidentified device can be identified manually by supplying the device type, device specification and device details. When the device is identified, it is checked whether its logs are received and recognized successfully.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Sequence** | **Navigation Path** | **Test Description** | **Data Input** | **Input Type** | **Expected Result** | **Logs Received?** | **Successful Attempts / Total Attempts** |
| An unidentified device tries to send logs | User is directed to the Device Recognizer tab. | An unidentified device tries to send Syslog messages to the server. | DLSU-PC, an unidentified device, tries to send Syslog messages to server. | Invalid | The server does not receive logs from the unidentified device. | NO | 0 / 20 (0%) |
| An identified device tries to send logs | User is directed to the Device Recognizer tab. | An identified device tries to send Syslog messages to the server. | DLSU-PC is now identified and tries to send Syslog messages to server. | Valid | The server receives logs from the identified device. | YES | 20 / 20 (100%) |

In Table 6-4, it shows that a device is configured to send its Syslog messages to the server. However in the first test, the device is unidentified. The hostname is sent to the unidentified\_device table. In the second test, the previously unidentified device is already identified by providing its corresponding Device Type, Device Specification and Device Details. The device is already included in the identified\_device table and removed from the unidentified\_device table. When the device tries to send logs to the system again, the logs are then accepted because the device can be seen in the identified\_device table.

**CHAPTER FIVE**

**SUMMARY, CONCLUSION AND RECOMMENDATION**

**5.1 SUMMARY**

We have been to design and develop a security information and event management system. The system is able to perform its purpose of collecting logs from different network devices, analyze and consolidate it and determining normal, alert and attack logs. With this, it is essential to point out how the system is able to perform its purpose by proving that the objectives are met. The SIEM is able to accept logs, as seen in Table 6-6, from different devices that are connected to a network. These devices include Defender Intrusion Detection System (IDS), Snort IDS, client Windows (XP and >7 models) and Linux machines, Apple Time Capsule router and file server, Cisco and D-Link switch, Cisco router, Cisco PIX firewall and NMAP Device Discoverer. As long as the devices are identified, the system is able to collect logs in networks of different locations. The system is tested in different laboratories both in manually set up network in the Computer Technology Laboratory (CT-LAB) and a live normal network environment such as CNIS. The system also is able to accept logs from devices whether a hub (3-Com) or router (Linksys or Cisco), or switch (D-Link or Cisco) is used. The system can also accept logs in different subnets of the same connection.

Both in a manually set up network and in a live normal network environment, the SIEM was tasked to receive logs for a whole day. It was able to accept logs from different network devices no matter how vast it was and it remained stable. The importance of both the log normalizer and log consolidator was seen. The devices reported on a standardized and simpler format, which makes the network monitoring and administration easier. The essence of log consolidation was also seen because there were recurring logs that were generated a lot of times and good thing all of those were consolidated into a single log.

All the consolidated logs are stored and archived in a table called alpha\_logs. The only necessity in the said module is to query necessary data in the alpha\_log table and show it in a Graphical User Interface (GUI). The recommendation part, however, exists for all devices that are specified in its appropriate normalizers. All alert logs must have a recommendation so that the network administrator can be able to resolve issues easier. The recommendation part of the module is also completed. During the comprehensive tests, each tester tries to invoke an alert so that it can check whether logs are classified as alert and see whether the recommendation is helpful enough. As a result, the alerts are logged in the alert manager and recommendations are seen. It is also summarized in the generated chart report classified as an alert log.

**5.2 CONCLUSION**

Different logs were successfully gathered and classified by the system. All crucial component-related logs are considered alerts and must be recorded together with a recommendation as seen in Table 6-18 and tested in Table 6-19. All logs classified by the IDS are considered priority “HIGH.” After correlation of the Attack and Device Inventories, the log may be classified “LOW” if the vulnerable operating system of the attack is not equal to the corresponding operating system of the device. All priority “HIGH” logs that have TCP as its protocol is event sniped, meaning a RST packet is sent to cut the connection. Other protocols aside from TCP are considered non-resolvable and are sent to the Alert Manager Module. Other tests suggest that some GUI backdoors are not consistently event sniped due to the fact it is the limitation of the IDS since events are not detected continuously.

During the comprehensive test of the system where it was run in a live normal network environment, different packet scenarios were tested using test scripts as seen and tested. Attack logs were invoked by launching attacks such as Tini, Netbus, Donald Dick and SYN Flood in different times and durations. Alert logs were also forced like shutting down interfaces in a switch and stopping crucial services in client computers. The effectiveness of log classification can be evidently seen in those scenarios. Of course, normal logs were also received and consolidated as well.

The system is adaptable to new devices invented or discovered just as long that it has a Syslog sending capability. It is just up to the network administrator to make device normalizers and classifying the hostname with the right device type and device specification. Overall, the system is able to perform its purpose in different scenarios and useful in network monitoring.

**5.3 RECOMMENDATION**

The system implemented has met its objectives per se. The system however during experimentation and implementation, has found other ways in making it more efficient and useful which is beyond its scope and limitations.

**5.3.1 Recommendation for the Device Identifier Module**

After the system implementation and during the experimentation, the Device Identifier Module has played a vital role in determining the devices that can send logs to the system. In this regard, it can be recommended that in the next time that the system is deployed, the network administration should probably mandate that all the identified devices use a hostname rather an IP Address. According to RFC 3164, the hostname of the device is given a higher priority of identification rather than the IP Address.

Although the IP Address is also used in replacement of the hostname, it was found out that in actual system deployment, most devices use a dynamic IP Address and therefore frequent update in the Device Identifier Module is needed. Unlike if the hostnames are used, whether or not the IP Addresses are changed, the identity of the device remains the same

**5.3.2 Recommendation for the Log Normalizer Module**

The Log Normalizer Module is implemented as planned. However the module can be improved by making a normalizer class that is inherited by the specific classes aside from the super normalizer class. In the current state of the module, all normalizers inherit some main functions from the super normalizer class and thus creating newly specified class for a particular device like a COMPUTER\_WINDOWS\_NORMALIZER or an IDS\_SNORT\_NORMALIZER. It is recommended to create a class that can be placed between the super normalizer class and the specific class. For example, instead of creating all the functions for the IDS\_SNORT\_NORMALIZER, some functions can be put in the IDS\_NORMALIZER. This IDS\_NORMALIZER has the normalization powers for an IDS and can be used by other normalizers even if it is not SNORT.

**5.3.3 Recommendation for the Log Consolidator Module**

The consolidation and correlation of the module are useful currently. But during the experimentation, there have been similar logs that are received by the system but not physically formed. There are patterns seen manually in logs generated by a certain device, but formed very differently, which make the consolidator hard to consolidate very well. Possible use of data mining might be recommended in future study for log consolidator improvements.

**5.3.4 Recommendation for the Incident Manager Module**

For incident management, only the TCP packets can be resolved when an attack occurs. It is therefore recommended to search for other ways on how to resolve attacks that are in another protocol like UDP or ICMP to make the system more useful

**REFERENCES**

Adriano, A., Sanchez, N., Trogo, J. & Villamor, R. (2014). Central Security Manager. Published in HNICEM, Undergraduate thesis, De La Salle University, Manila, Philippines.

AdventNet. (2007). *Analyzing Logs for Security Information Event Management.*AdventNet Inc

ArcSight. (2021). ArcSight. Retrieved May 2021 from https://www.microfocus.com/en-us/cyberres/secops

Bellare, M. & Yee, B. (2007). *Forward Integrity for Secure Audit Logs*. University of California, San Diego, USA.

Bulanhagui, P., Redulla, G. & Tan, L. (2015). *Online Network Monitoring System Using RMON Standards, NEMSYS Online,* Undergraduate Thesis, De La Salle University

Gomez, M. & Wong, S. (2017). Virtual Information Security Testing System, EPSILON, Undergraduate thesis, De la Salle University, Manila, Philippines

Gruschke, B. (2018). Integrated Event Management: Even Correlation using Dependency Graphs. Department of Computer Science, University of Munich, Germany

Hallawell, A. (2014). Gartner Global Security and Privacy Best Practices. Gartner Analyst Reports, Retrieved April 02, 2020, from www.csoonline.com/analyst/ report2332.html

Hellerstein, J. & Ma, S.(2016) Mining Event Data for Actionable Patterns. IBM T.J. Watson research Center Hatwthrone, New York, USA

Lara, C., Kebschull, U. & Schiefer, A. (2018). Distributed and Decentralized Event Management for Embedded Systems. Department of Computer Engineering, University of Leipzig, Leipzig, Germany.

Manticore Technology Corporation. (2018). Intellitactics Security Manager. Retrieved November 2008 from http://www.intellitactics.com.

MIT-Xperts. (2018). iSims SI (EIT/SDT) Management System. Retrieved November 2018 from http://www.mit-experts.com/products/isims/ isims.pdf.

Net-SNMP. (2017). SNMP. Retrieved November 2020 from http://www.net-snmp.org.

Nicolett, M., Proctor, P. & Williams, A. (2016). Magic Quadrant for Security Information and Event Management. Gartner RAS Core Research Note.

OSSIM Development Team. (2008). Open Source Information Management. Retrieved November 2008 from http://www.ossim.net.

RSA Security Incorporated. (2017). Design and Implementation for Security Information Management. The Security Division of EMC.

Schneir, B. (2015). Managed Security Monitoring: Network Security for the 21st Century. Retrieved March 10, 2021, from http://www.counterpane.com.

Security Information Management Resource Guide. (2008). *The Security Information Pocket Guide*. Retrieved November 2008 from http://www.simbuyer.com

Tenable Network Security. (2008). Nessus. Retrieved November 2008 from http://cgi.tenablesecurity.com

**Appendix**

**Source Code**

import { enableProdMode } from '@angular/core';

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';

import { environment } from './environments/environment';

if (environment.production) {

enableProdMode();

}

platformBrowserDynamic().bootstrapModule(AppModule)

.catch(err => console.log(err));export interface Alarm {

id: string;

title: string;

status: string;

kingdom: string;

category: string;

timestamp: string;

update\_time: string;

risk: number;

risk\_class: string;

tag: string;

src\_ips: string;

dst\_ips: string;

networks: string;

rules: [{

timeout: number;

protocol: string;

from: string;

to: string;

port\_from: string;

port\_to: string;

plugin\_id: number;

stage: number;

start\_time: number;

end\_time: number;

reliability: number;

plugin\_sid: number;

occurrence: number;

events\_count: number;

}];

}

export interface AlarmSource {

source: Alarm;

}