**Ministry of Higher Education and Scientific Research**

**University of Babylon: College of Science for Girls**

**Computer Department: Fourth stage**

# Building a System for Monitoring Sensitive Files

**A research submitted to the Council of the College of Science for Girls, which is part of the requirements for obtaining a Bachelor's degree in Computer Science**

**By:- Zahraa Falah Khadir**

**Supervised by :- Dr. Farah Al-Shareefi**

**م 2024-2023**

**بسم الله الرحمن الرحيم**

**(( منْ عَمِلَ صَالِحًا مِنْ ذَكَرٍ أَوْ أُنْثَى وَهُوَ مُؤْمِنٌ فَلَنُحْيِيَنَّهُ حَيَاةً طَيِّبَةً ﴿٩٧﴾))**

**صدق الله العلي العظيم**

**[النحل:٩٧]**

# الشكر والتقدير

**الشكر والثناء لله تعالى على اتمام هذا البحث وعلى اتمام الدراسة وأرجو أن تنال رضاه فالحمد لله على هذه النعم ،**

**ومن ثم أتقدم بالشكر والتقدير الى :**

**أساتذتي الكرام في قسم الحاسبات وبالاخص الدكتورة "فرح علي حسن الشريفي" التي تفضلت بالاشراف على هذا البحث ، شكر كبير لها على نصحه والأرشاد ومساعدتها لي ..**

**الى أحباب قلبي أمي وأبي وإخوتي وأخواتي ...**

**الى من كانوا خير صحبة ورفقة لي خلال مسيرتي الدراسية في الجامعة....**

**الى كل من قدم لي الدعم والتوجيه كل الشكر والأمتنان لكم على كل شيء.**

# الاهداء

**قال تعالى: (قل أعملوا فسيرى الله عملكم ورسوله والمؤمنون )**

**إلهي لا يطيب الليل إلا بشكرك ولا يطيب النهار إلا بطاعتك . ولا تطيب اللحظات إلا بذكرك .. ولا تطيب الآخرة إلا بعفوك .. ولا تطيب الجنة إلا برؤيتك ...**

**الله جل جلاله**

**إلى من بلغ الرسالة وأدى الأمانة ... ونصح الأمة .. إلى نبي الرحمة ونور العالمين سيدنا محمد صلى الله عليه وآله وسلم**

**إلى من كلله الله بالهيبة والوقار .. إلى من علمني العطاء بدون انتظار .. إلى من أحمل أسمه بكل افتخار ..ستبقى كلماتك نجوم أهتدي بها اليوم وفي الغد وإلى الأبد**

**«والدي العزيز»**

**إلى ملاكي في الحياة .. إلى معنى الحب وإلى معنى الحنان والتفاني .. إلى بسمة الحياة وسر الوجود إلى من كان دعائها سر نجاحي وحنانها بلسم جراحي إلى أغلى الحبايب**

**«أمي الحبيبة»**

**إلى الذين حملوا أقدس رسالة في الحياة إلى الذين مهدوا لنا منارة العلم والعلماء إلى الصرح الشامخ طريق العلم والمعرفة .. أساتذتنا الأفاضل.**

# CONTENTS

# Abstract

File protection has always attracted considerable interest in both industry and academia, since almost all the data and information are stored in files in modern computer systems. Many of the security accidents are relevant to unauthorized file accesses. To protect sensitive files, an effective approach named as File Integrity Monitoring (FIM) is usually used to detect aggressive behaviors by verifying all the actions on these sensitive files. FIM is a widely used security control mechanism to examine the integrity of sensitive files, databases, and folders for signs of tampering or corruption, which may be an indication of a cyberattack. The checksum, hash, and attribute values (including file size, version, and modification date) of the files under observation are measured by FIM and compared to their initial values. FIM can be broadly classified into two types: online and offline monitoring schemes. When a scheme is offline, it indicates that the FIM tools will periodically check the integrity of the sensitive files based on user settings, rather than monitoring the files at real-time. This project primarily focuses on an off-line file integrity monitoring scheme that verifies the file size attribute. Python technology is utilized to investigate whether this technique can result in a delay in the detection of the alteration.

# Chapter One:
# General Introduction

# 1.1 Introduction

Ensuring integrity of sensitive files is imperative nowadays. The term 'integrity' means guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity [1]. The vast majority of attacks work through unapproved or unauthorized access to sensitive files to take secret data like secret keys, passwords, credit card numbers, and so on [2]. After that, attackers generally conceal their traces by subverting critical files like system logs. File system integrity monitoring (FIM) is a well-known way to deal with ensuring integrity of sensitive and critical files [3]. FIM is one of the security components that can be implemented in host environment. As a part of host based intrusion detection (HIDS) components, FIM should play a big role in detecting any malicious modification either from authorized or unauthorized users on their contents, access control, privileges, group and other properties. The main goal of related integrity checking or monitoring tools is to notify system administrators if any changed, deleted or added files detected [4]. File integrity checkers or monitors measure the current checksum, hash values, attribute values (attributes such as file size, version, modified by, creation date, modified date) of the monitored files with their original value. In general, FIM can be divided into two categories, off-line

and on-line monitoring scheme [5]. **Offline** scheme means that the FIM tools will **monitor** the **files** at scheduled times to check the **integrity** of the files, i.e., it monitors the integrity of the related files from time to time according to user setting. While on-line pattern monitors the FIM-related files in real time as each file is modified.

In this project the focus is specifically on off-line File Integrity Monitoring scheme which checks the file size attribute. The inspection of whether this scheme can cause delay in detection of the modification is performed with the help of Python technology.

## 1.2 Research Outline

- **Chapter one :-** General Introduction
- **Chapter two:-** Theoretical Background
- **Chapter three:-** The method of the applied FIM Scheme.
- **Chapter four :-** Results and Discussion
- **Chapter five :-** Conclusion and future work

# Chapter two
# Theoretical Background

## 2.1 Introduction

One of the fundamental goals of computer security is to ensure the integrity of sensitive data [1]. A large number of safe accidents result from modifying critical files in the system. The attackers intrude into the computer system, and hide their traces by tampering critical files, such as executable files and system logs. File Integrity Monitoring (FIM) is one of the most popular approaches to observe hostile behaviors, such as modifying system log, appending backdoors, inserting Trojan horses [2, 3].

In order to avoid serious damage to the sensitive files, it is necessary to observe malicious behaviors in them. This chapter introduces the background to this project and establishes the context underlying the research.

## 2.2 The CIA Triad

The CIA triad is composed of [11]:

1) **Confidentiality:** This refers to the need to ensure that sensitive information is safe from unauthorized access attempts. To this end, measures are established to differentiate the level of access that certain people can have to the information, depending on the type or level of confidentiality of the data.

2) **Integrity:** This involves maintaining the consistency, accuracy and reliability of the data during its life cycle, preventing unauthorized persons from modifying the data in transit or at rest. In the field of integrity control, the concepts of *digital signature* and non-repudiation measures, *cryptographic checksums* or File Integrity System (*FIM*) *systems* which we will discuss later, usually appear in the field of integrity control.

3) **Availability:** This refers to the ability to keep information consistently and easily accessible to authorized persons. Authentication mechanisms, storage, infrastructure and all the supports related to the information must work properly to make it available safe from crashes, failures, etc. As far as availability is concerned, we usually talk about preventive measures such as *redundancy, failover, system monitoring, backups and continuity plans*.

Figure 2.1 CIA Triad [11]

## 2.3 What is File Integrity Monitoring (FIM)?

**File integrity monitoring (FIM)** [2], sometimes referred to as file integrity management, is a security process that monitors and analyzes the integrity of critical assets, including file systems, directories, databases, network devices, the operating system (OS), OS components and software applications for signs of tampering or corruption, which may be an indication of a **cyberattack.**

Typically, FIM involves examining files to see if and when they change, how they change, who changed them, and what can be done to restore those files if those modifications are unauthorized. Companies can leverage the control to supervise static files for suspicious modifications such as adjustments to their IP stack and email client configuration. As such, FIM is useful for detecting malware as well as achieving compliance with regulations like the **Payment Card Industry Data Security Standard (PCI DSS)** [3, 4].

**File integrity monitoring** was invented in part by **Tripwire founder Gene Kim**. From there, it went on to become the security control around which many organizations now build their cybersecurity programs. The specific term "file integrity monitoring" itself was widely popularized by the **PCI standard** [5].

## 2.4 Why is File Integrity Important?

Concerns related to cyberattacks and data breaches have grown among enterprises worldwide. Threat actors are continually looking to access valuable and sensitive business data such as system access details, confidential client information, employees' financial data, and credit card information. Cyberattackers often start by targeting organizations' critical systems to gain easy access to personal data. File Integrity is essential to protect sensitive host files, folders, and other vital assets within an organization. It helps identify whether a file has been tampered with by unauthorized users. Here are some of the reasons why file integrity is essential [6, 7]:

1) **Protection from unauthorized access:** continuously monitoring file integrity acts as an additional security measure to detect unwanted intrusion across system files. It prevents illicit activities from causing significant disruption within an organization. By using FIM solutions to check file integrity, organizations can gain valuable insights into cyberthreats and the files targeted by threat actors. It can also help

companies prevent advanced persistent threats (APT) that are hard to detect and use highly sophisticated techniques to access a network.

2) **Visibility into file changes:** it's not uncommon for an employee within an organization to accidentally change or delete a sensitive file. Validating file integrity lets IT admins gain visibility into the file change, the source and time of the change, and the reason behind the change. It helps them track whether the changes were made by an unintentional authorized party or a threat actor.
3) **Better control:** checking file integrity is a great way to control critical system data. IT admins can manage file access privileges and permissions for different groups in an organization. It further helps ensure employees only have access to the data they need.
4) **Improved system health:** file integrity is a good indication of whether a file got corrupted by unauthorized access after its creation. Monitoring file integrity helps quickly track intrusion, alert the cybersecurity team, and troubleshoot faster. It helps mitigate future unwanted access and encourages organizations to adopt better security measures, ensuring better system health. IT admins can also analyze patches and updates for specific files via advanced **file integrity monitoring tools.**

## 2.5 What Files Should be Monitored?

The key file types that attackers will look to modify, or even delete, are [8, 9]:

**1. Operating system directories and files.**

It is important to assure that th base operating system is functioning as expected, so monitoring the system binaries and libraries should be the first step.

On Windows, the core OS binaries and key configuration files are typically located under [8, 9]:

C:WindowsSystem32 directory

On Linux, the critical directories to monitor include:

/bin

/usr/bin

/sbin

/usr/sbin

**2. Applications directories and files.**

The system is the foundation on which the application sits, however, it is the applications that the employees, partners, and customers interact with, and that store and manage the personal data. Thus, application binaries should be monitored accordingly [8, 9].

On Windows systems, most applications (by default) store their binaries and configuration files under:

C:Program Files

C:Program Files (x86)

On Linux systems typically install applications into:

/usr/bin

/usr/sbin

/opt

Depending on the type of server and applications being run, additional files and/or directories may also need to be monitored. For example, if the server is a web server, the directory where the web site files reside

should be monitored as well. This will vary by organization based on web server used and configuration of the web server.

**3. Configuration files.**

Modifying system and application binaries can be challenging, since they are often locked when the system starts up or when the services/daemons are running. That said, configuration files define how the system and applications on the system function and are typically read only when the system service or application starts up.

Configuration settings can be stored in many ways. On Windows platforms, the Windows Registry is typically used for configuration purposes. Text-based configuration files can be found across Windows, Unix/Linux, and OS X.

Attackers may target any of these configuration locations for a planned attack, or an administrator may inadvertently misconfigure a system, causing that system to be exposed and putting the data on that system and the rest of the infrastructure at risk [8, 9].

**4. Log files.**

Log files contain the transaction and activity history for the core operating system, its subsystems, and applications that reside on the system. They are often the first place an attacker will look to hide their tracks.

While actively written log files will continually change, only the system or application should be writing to them. To ensure that log files are not tampered with, you should establish an active log management collection method to pull (or push) the logs from the system to a separate log management solution for centralized monitoring and tamper-proof storage.

Archived log files are static in nature, so you can also monitor for any changes or deletions of those files [8, 9].

**5. Digital keys and credentials.**

Even with the availability of directory systems and hardware security modules, many systems and applications store their keys and credentials for authentication and encryption on a system. Monitoring those credential/key stores is also important to ensure your system is protected.

For example, Unix systems store their password file under /etc, and Windows under C:\Windows\System32\config. Windows Credentials is also there. You may be using other popular authentication applications such as Secure Shell (SSH) application [8, 9].

**6. Content files.**

Your corporate and your customer data is the lifeblood of your organizations. Data leakage remains one of the top security concerns of most organizations.

Even content as simple as your website is mission-critical. The effects on your brand and reputation can be significant should an attacker deface your public presence. Monitoring content files for unauthorized changes within the web server is critical to ensure the integrity and confidentiality of that data [8, 9].

## 2.6 How file integrity monitoring works

Below are the six main steps usually are taken to set up a successful file integrity monitoring process [10]:

**Defining the file integrity policy:** Before the FIM tool can be deployed, the organization must first specify what assets will be monitored and the types of changes that it wants to detect.

**Establishing a baseline:** FIM tools create an initial baseline, which will serve as a reference point for comparison for all future activity. This will contain all file attributes, including file size, content, access settings, privileges, credentials and configuration values. This project focuses on the file size attribute.

**Applying the cryptographic hash signature:** As part of developing the baseline, the IT team will also apply a cryptographic hash signature, which can't be altered, to each file. Any changes made to the files, whether authorized or not, will also change the hash value of the file, making it easy to detect file updates and alterations. This method has not been applied in this project, as the values of the files' size are compared.

**Monitoring, analyzing and verifying file integrity:** The FIM tool compares the hash values on the files to quickly and clearly detect anomalous changes. As part of this process, the IT team can also exempt certain changes from monitoring to avoid triggering alerts for planned changes or updates.

**Issuing an alert:** In the event an unexpected or unauthorized modification has been made, the FIM tool will also alert the information security (InfoSec) team of the need for further investigation and possible remediation.

**Reporting and regulatory compliance:** In some cases, the organization must report a breach to the relevant authorities, as dictated by law. The FIM tool helps the Regulatory Affairs team compile such information for reporting purposes.

# Chapter three
# The Designed System

## 3.1 Introduction

This chapter will briefly describe our designed method.

## 3.2 The Designed System

The purpose of this project is to implement a system that monitors a selected sensitive file and gives a notification if any modification ate this file is identified. Figure 3.1 illustrates this system schematically. It consists of the following stages:

1) **Selection**: at this stage a desired sensitive is selected to be monitored by system.
2) **Measurement**: the size attribute is used for monitoring changes. Typically, the sized of the selected file is measured and saved to be compared afterwards.
3) **Establishing the Monitoring Time**: **monitoring** for **file changes** (**files** that were either **altered** or **modified**) can be performed within **a time interval**. For example, every 3 seconds, or 2 minutes a file is checked. Accordingly, a user must set this interval to be observed periodically.
4) **Monitoring:** With a saved size value, a system can proceed to monitor a chosen file for changes. Essentially, every a period time a system compares the previous size value to the current one. If they are the same then the file has not changed; otherwise it can issue an indication for a malicious modification.
5) **Reporting a Notification**: In the malicious modification event, a system generates a report for the user in order to show them the time at which the file is maliciously altered. In fact, the generated report is a detailed one, as it contains information about every time the file checked by showing the results with green color if the file is intact, while a red color is used at changes detection case.

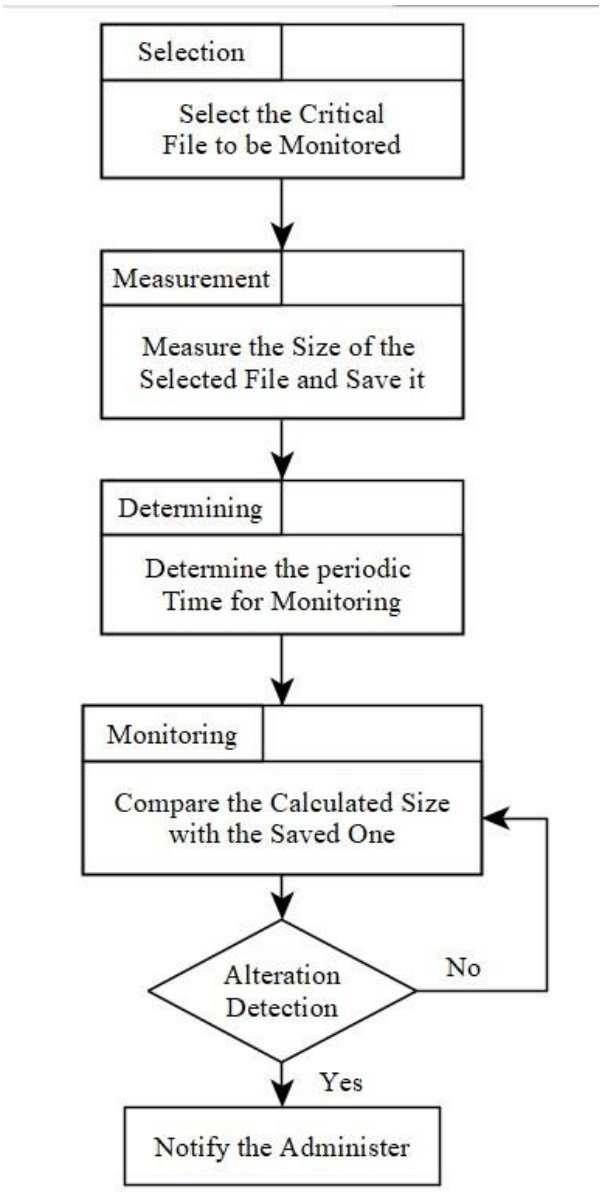


**Figure 3.1 The Designed System Diagram**

# Chapter four
# Results and Discussion

## 1.1 Introduction

This chapter will present the main result obtained from implementing our system for monitoring sensitive files.

## 4.2 Analysing the Designed System

Our system is analysed by using a laptop equipped with an Intel Core i7-4700MQ processor running at 2.4GHz, with 12GB of available RAM. The used programming language is Python.

After implementing our system, an interactive window, that is shown in

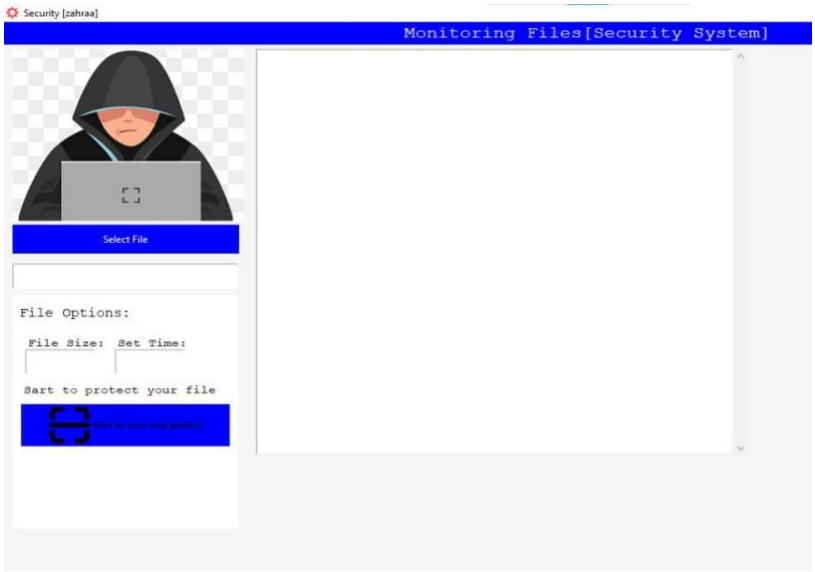


Figure 4.1, is launched.

**Figure 4.1 Initial Launched Interactive Window for the Designed System**

When a file that must be monitored is entered and the period time is determined, the system yields a report which is shown in Figure 4.2.

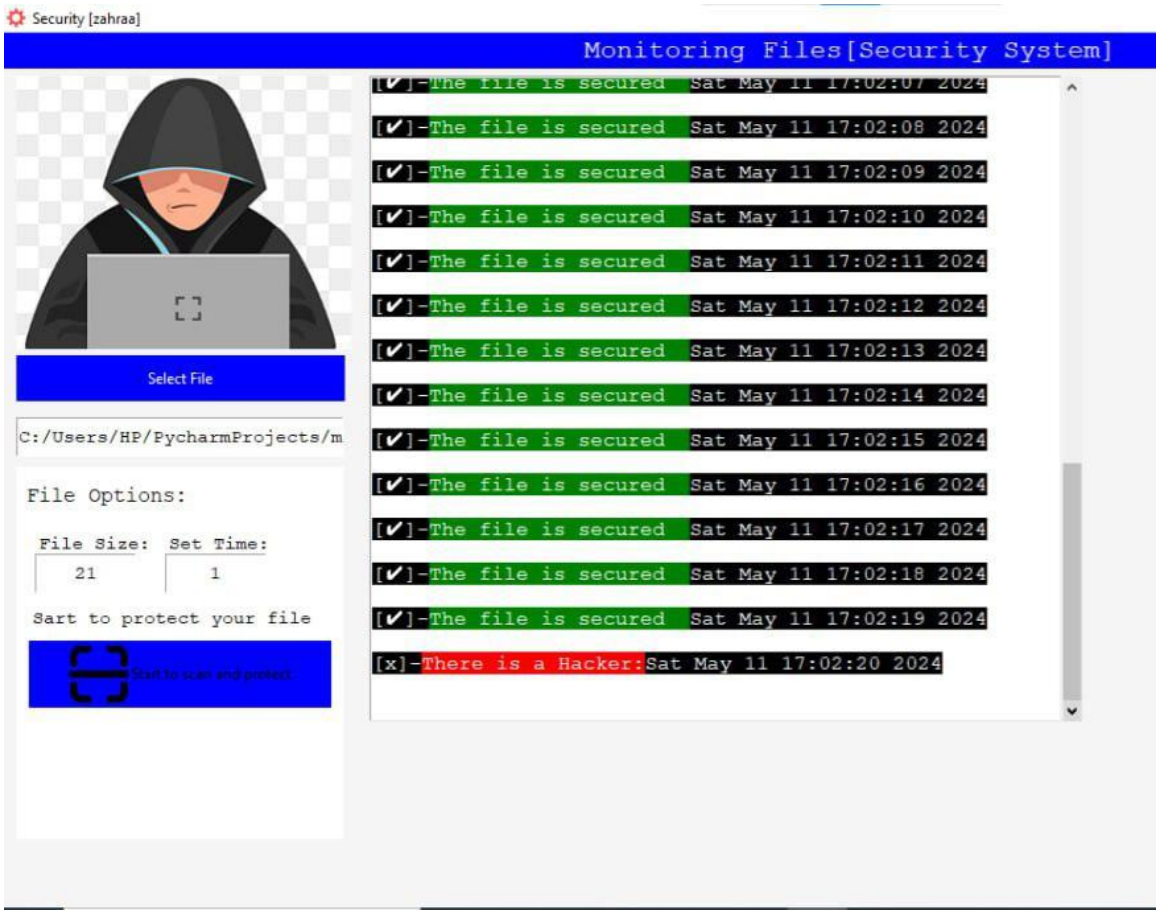


**Figure 4.2 An Interactive Window That Displays the Monitoring Report**

In order to evaluate our system, we calculate the time difference between the actual start time of modification and the changes detection time. This gives insight about the detection delay. We generate five random files using Python technology. All the evaluation results are shown in Table 4.1. From

this table, we can see the size of the observed file negatively affects the detection time.

| File Size | Monitoring Interval Time | Modification Time | Detection Time | Delay |
|---|---|---|---|---|
| 5 KB | 3 | 02:29:48 | 02:29:48 | 0 seconds |
| 10 KB | 3 | 02:26:08 | 02:26:10 | 2 seconds |
| 50 KB | 3 | 02:32:39 | 02:32:41 | 2 seconds |
| 100 KB | 3 | 02:35:40 | 02:35:43 | 3 seconds |
| 1 MB | 3 | 02:41:30 | 02:41:31 | 0 seconds |

**Table 4.1 Calculated Changes Detection Delay**

# Chapter five

# Conclusion and future work

## 5.1 Conclusion

A FIM system that is based on off-line monitoring scheme is presented. It focuses on the file size attribute to identify any malicious changes. It can be deduced that the performance of the periodic monitoring for the file size is degraded as the file size is increased. However, the file size attribute must not be ignored and it must be supported by another comparison technique.

## 5.2 Future Work

As a future step, we intend to carry out the following:

1) Instead of using only one attribute for monitoring, a set of attributes, like: file's version, file extension, etc. can be employed for this purpose
2) A hybrid FIM method can be developed that merges between the off-line and on-line schemes.
3) Performing the comparison checks by using hash values together with the attributes comparison.

# References

[1] Abdullah, Z. H., Udzir, N. I., Mahmod, R., & Samsudin, K. (2011). File integrity monitor scheduling based on file security level classification. In *Software Engineering and Computer Systems: Second International Conference, ICSECS 2011, Kuantan, Pahang, Malaysia, Proceedings, Part II 2* (pp. 177-189). Springer Berlin Heidelberg, , *June 27-29, 2011.*

[2] Peddoju, S. K., Upadhyay, H., & Lagos, L. File integrity monitoring tools: Issues, challenges, and solutions. *Concurrency and Computation: Practice and Experience*, *32*(22), e5825, 2020.

[3] Parida, P., Konhar, S., Mishra, B., & Jena, D. Design and implementation of an efficient tool to verify integrity of files uploaded to cloud storage. In *2017 7th International Conference on Communication Systems and Network Technologies (CSNT)* (pp. 62-66). IEEE, 2017.

[4] Zhan, D., Ye, L., Fang, B., Du, X., & Su, S. Cfwatcher: A novel target-based real-time approach to monitor critical files using vmi. In *2016 IEEE International Conference on Communications (ICC)* (pp. 1-6). IEEE, 2016.

[5] Gupta, S., Sardana, A., & Kumar, P. A light weight centralized file monitoring approach for securing files in cloud environment. In *2012 International Conference for Internet Technology and Secured Transactions* (pp. 382-387). IEEE, 2012.

[6] Quynh, N. A., & Takefuji, Y. A real-time integrity monitor for xen virtual machine. In *International conference on Networking and Services (ICNS'06)* (pp. 90-90). IEEE, 2006.

[7] Herrero Pérez de Albeniz, A. File integrity monitoring on Linux systems, 2021.

[8] Zlatkovski, D., Mileva, A., Bogatinova, K., & Ampov, I. A new real-time file integrity monitoring system for windows-based environments, 2018.

[9] Pinheiro, A., Canedo, E. D., De Sousa, R. T., & Albuquerque, R. D. O. Monitoring file integrity using blockchain and smart contracts. *IEEE access*, *8*, 198548-198579., 2020.

[10] Al-Muntaser, B., Mohamed, M. A., & Tuama, A. Y. Real-Time Intrusion Detection of Insider Threats in Industrial Control System Workstations Through File Integrity Monitoring. *International Journal of Advanced Computer Science and Applications*, *14*(6), 2023.

[11] Samonas, S., & Coss, D. The CIA strikes back: Redefining confidentiality, integrity and availability in security. *Journal of Information System Security*, *10*(3), 2014.

# *!Finished, praise be to Godty*