

**FILE INTEGRITY MONITOR USING HASHING ALGORITHMS**

**By**

**KEN MATHEWS**

**TP054901**

**APU3F2211CS(CYB)**

A project submitted in partial fulfilment of the requirements of Asia Pacific University of Technology and Innovation for the degree of

**B.Sc. (Hons) in Computer Science (Cyber Security)**

Supervised by Dr. Mohammed Shabbir Hamza Abdulnabi

2nd Marker: Ms. Noris Ismail

26-JULY-2023

**Acknowledgement**

I would want to convey my sincerest thanks to all of those who contributed support and aid to accomplish this project. First and foremost, I want to express my gratitude to the project manager, Mr. Dhason Padmakumar, for giving us the tools, advice, and support we needed to complete the project. He was a great assistance to us all along the way. I also want to thank my project supervisor, Dr. Mohammed Shabbir Hamza Abdulnabi and project 2nd marker, Ms. Noris Ismail, for assisting and guiding me along the way while working on this project.

I am grateful to my friends for their advice, assistance, and encouragement during the project as well as while we were all working on our own projects.

I also want to thank everyone who participated in the survey providing me valuable data and ideas in order to create a foundation for this project.

**Table of Contents**

**1. Introduction to Study ………………………………………………………………….….5**

1.1 Background to the project …………………………………………………..….…5

1.2 Problem context ………………………………………………………………..…6

1.3 Rationale …………………………………………………………………….……6

1.4 Potential benefits …………………………………………………………….……7

1.4. 1 Tangible benefits ……………………………………………………….7

1.4. 2 Intangible benefits ……………………………………………………...7

1.5 Target users ……………………………………………………………………….8

1.6 Scope and objectives ……………………………………………...........................8

1.6.1 Aims …………………………………………………………………….8

1.6.2 Objectives ……………………………………………………………….8

1.6.3 Deliverables - Functionality of the proposed system …………………...9

1.6.4 Nature of Challenges ……………………………………………………9

1.7 Overview of this Investigation report ……………………………………………10

1.8 Project Plan ………………………………………………………………...........11

**2. Literature Review ………………………………………………………………………..12**

2.1 Introduction …………………………………………………………...…………12

2.2 Domain research …………………………………………………………………13

2.3 Similar Systems …………………………………………………………………21

2.4 Summary ………………………………………………………………………...24

**3. Technical Research ……………………………………………………………………...26**

3.1 Programming language chosen ………………………………………………….26

3.2 IDE (Interactive Development Environment) chosen ……………...……………27

3.3 libraries chosen / Tools chosen ………………………………………………….27

3.4 Operating System chosen ………………………………………………………..28

3.5 Summary ………………………..……………………….………………………28

**4. System Development Methodology ……………………………………………………..29**

**5. Research Methods ……………………………………………………………………….32**

5.1 Introduction ……………………………………………………………………...32

5.2 Design ……………………………………………………………………………32

5.3 Summary ………………………………………...………………………………36

**6. Requirements Validation ……………………………………………………………..…39**

6.1 Analysis of Data …………………………………………………………………39

6.2 Summary ………………………………………………………………………...46

**7. System Architecture ……………………………………………………………………..47**

7.1 Introduction ……………………………………………………………………...47

7.2 Abstract Architecture ……………………………………………………………49

7.2.1 System Design ……………………………………………………...….49

7.2.2 Interface Design …………………………………………………...…..53

**8. Project Plan ………………………………………………………………………….…..58**

8.1 Features …………………………………………………………..……………...58

8.2 Release Plan ……………………………………………………………………..59

8.2.1 Version 1.0 …………………………………………………………….59

8.2.2 Version 2.0 …………………………………………………………….60

8.3 Test Plan …………………………………………………………………………62

8.3.1 Unit Testing ……………………………………………………………62

8.3.2 User Acceptance Testing ………………………………………………63

**9. Implementation …………………………………………………………………………..64**

9.1 Screenshots ………………………………………………………………………64

9.2 Sample Codes ……………………………………………………………………67

**10. System Validation ……………………………………………………..………………..73**

10.1 Unit Testing …………………………………………………………………….73

10.2 User Acceptance Testing ……………………………………………………….78

10.3 Summary ……………………………………………………………………….80

**11. Conclusion and Reflections ………………………..…………………………………..81**

11.1 Critical Evaluation ……………………………………………………………..81

11.2 Conclusion ……………………………………………………………………..82

**References ……………………………………………………………………………….…84**

**Appendices …………………………………………………………………………………86**

**1 Introduction to the Study**

* 1. **Background of the Project**

As the world moves forward in this digital era, the security and integrity of digital assets and data are becoming of higher importance and value. Today, it is easy for anyone to learn how to hack and gain unauthorized access to any type of confidential data or assets, compromising the integrity of the files and making systems vulnerable enough to steal valuable data.

According to the CIA triad, Confidentiality, Integrity, and Availability of data is of prime importance in reference to cyber security.

**Confidentiality** - Limiting unauthorized parties' access to sensitive information

**Integrity** - Ensuring that data is accurate, reliable, and consistent over the course of its existence.

**Availability** - refers to ensuring that the IT infrastructure operates without hiccups, internal system disputes, or any other problems that can prevent access to crucial files.

A software program called a File Integrity Monitor (FIM) is intended to find and notify users of any illegal modifications to files and directories within a computer system. It operates by continuously checking the status of files and folders and comparing it to the known, anticipated condition.

File integrity monitoring became necessary as a result of the rise in cyberattacks, particularly those aimed at malware and data breaches. Attackers frequently try to change or update system files in order to implant malware, obtain unauthorized access, or hide their tracks. (Kedgley, n.d.)

To find file alterations, FIM systems employ a variety of methods, including as system log analysis, change monitoring, and file hashing. They can be set up to notify users when a file is edited, keep account of who made the changes, and, if necessary, roll back the modifications. FIM systems may be helpful for any business concerned with data security. They are frequently employed in sectors with high security needs, such as banking, healthcare, and government. (CrowdStrike, 2022)

By assisting organizations in identifying unauthorized changes to crucial files on their systems, File Integrity Monitor lowers the possibility that data will be lost or compromised, which would cost time and money in terms of lost productivity, lost revenue, reputational damage, and legal and compliance fines. In an IT context, FIM software will scan, examine, and report on any unforeseen modifications to crucial data. By doing this, file integrity monitoring adds a crucial layer of data, application, and file security and speeds up incident response. (Udzir, N.I., 2011)

* 1. **Problem Context**

Most often during cyber-attacks, it is important to know whether any files critical to the operating systems have been altered or tampered with. And with other attack prevention tools, it would be too late to recover from any damage done. Manually going through thousands of files to check for changes is an impossible task as sometimes the changes are too small and are overlooked. Therefore, having a FIM in place would make that process much easier and any changes would be alerted instantly. Such a system also meets compliance with legal requirements like GLBA, SOX, HIPAA, and PCI DSS necessitate the capacity to audit modifications and monitor and report specific sorts of activities.

Implementing FIM can also be challenging, as it requires configuring and managing a complex system of alerts and reports. FIM also generates a large amount of data, which can be difficult to analyze and prioritize. Therefore, organizations need to carefully consider their FIM strategy and tool selection, as well as invest in training and resources to effectively manage their FIM program. (Kedgley, n.d.)

* 1. **Rationale**

The rationale for using a file integrity monitor (FIM) is based on the need to ensure the integrity and security of critical data and systems within an organization.

**Protection against cyberattacks**: Preventing cyberattacks is one of the main benefits of using a FIM. Attackers may try to install malware, change, or remove important system files, or take sensitive data. These illegal modifications may be found in real-time with the use of a FIM, enabling enterprises to act fast and stop possible security breaches.

**Operational effectiveness**: By lowering the time and effort needed to monitor and maintain crucial files and systems, a FIM can assist increase operational effectiveness. Organizations may lower the risk of mistakes, increase system availability, and react swiftly to any events by automating the monitoring process.

**Forensic analysis**: In the case of a security issue, a FIM can offer insightful forensic information on any illegal modifications that were made to the system. This can aid investigators in locating the attack's origin and enabling them to take the necessary precautions to stop such assaults in the future. (Von Ogden, n.d.)

* 1. **Potential Benefits**
     1. Tangible Benefits

A file integrity monitor (FIM) is a software tool that helps to detect and report any changes made to files or directories on a computer system. There are several tangible benefits associated with using a file integrity monitor. First off, it may greatly improve security by spotting any unauthorized or unusual changes to crucial system files that can be caused by malware or other intrusions. Second, regulatory compliance with data protection and privacy laws like the PCI DSS and GDPR frequently calls for file integrity monitoring. Lastly, FIMs can enhance operational efficiency by speeding up procedures for compliance reporting and auditing and detecting and correcting changes to system files or settings. Fourthly, FIMs offer a record of modifications made to important files or settings, which may be helpful for forensic analysis, accountability, auditing, and compliance reporting. Lastly, file integrity monitoring may assist avoid unintentional or illegal modifications to system files or settings, which can cause system instability or downtime, and increase system stability. (Team. R, 2022)

* + 1. Intangible Benefits

In addition to the tangible benefits of file integrity monitoring, there are also several intangible benefits that can be gained by organizations. First off, using a FIM may enhance communication between IT teams, security employees, and compliance staff as well as boost trust in system security. Second, by giving enterprises real-time visibility into changes made to crucial files and configurations, FIMs may assist organizations in managing file integrity threats more effectively. Lastly, by showcasing a dedication to data security and privacy, a FIM implementation may improve an organization's standing and trustworthiness. Fourthly, FIMs can assist in lowering the expenses related to security lapses, compliance lapses, and system outages. Ultimately, file integrity monitoring is a crucial tool for enhancing security and lowering risk because of the intangible advantages it may have on an organization's operations and performance. (Team. R, 2022)

* 1. **Target Users**

File Integrity Monitor (FIM) is a valuable tool for various users responsible for maintaining the security and integrity of critical data and systems. Information Security Professionals can use FIM to detect and respond to any unauthorized changes made to the system and reduce the risk of security breaches. Compliance officers can ensure that the organization is complying with regulatory requirements such as PCI DSS, HIPAA, SOX, and others. System administrators can use FIM to monitor system files and directories for any changes that may impact system availability. IT Managers can use FIM to monitor the activity of system administrators and ensure that they are following established policies and procedures. DevOps teams can detect any unauthorized changes to software code or configuration files and ensure the integrity of the software development process. Overall, FIM is a valuable tool for any organization concerned with data security and regulatory compliance and can be used by various users, including Information Security Professionals, Compliance Officers, System Administrators, IT Managers, and DevOps teams. (Abdullah & Mahmod, 2011)

* 1. **Scope and Objectives**
     1. Aims

To build a system that continuously checks the integrity of any given files using Hashing Algorithms such as SHA-256 or SHA-512 in order to determine if the files have been corrupted or tampered with.

* + 1. Objectives
* To study how file integrity monitors are used in the modern world to prevent cyber-attacks or breaches
* To develop a system (FIM) that would prevent unauthorized access to any assets that would result in data thefts, tampering of critical files or DDOS attacks.
* To maintain the integrity of all files requires more than just policy, education, and access governance. Implementing a FIM is the next step to ensure data isn’t tampered with.
* To test the capabilities and functioning of the proposed system in a safe environment to make sure the expected results are yielded.
  + 1. Deliverables – Functionality of the proposed system

This system will be able to monitor given files in a server, folder, or database to detect any kind of changes or tampering in order to alert the user of threats or cyber-attacks on the system. The FIM will use a hashing algorithm such as SHA-256 or SHA-512 in order to determine if the files have been corrupted or tampered with. Through the use of the post-patch checksum, this will enable users to determine if installed versions of files have been updated to the most recent version as well.

A systems administrator can swiftly address this issue if a user makes improper modifications to sensitive files or launches a suspicious access attempt by revoking that user's access credentials. The following features will be able for the user while using this FIM system –

* Detect tampering of any sort for the files provided by constant monitoring.
* Receive notifications when breaches or data alterations occur.
* Manually check files with reference to the baseline of files provided.
  + 1. Nature of Challenges

It might be difficult to construct a file integrity monitor (FIM), as it necessitates a full understanding of the system architecture, file structure, and possible threats. The following are some of the main difficulties that might arise when creating a FIM:

**Complexity of systems**: Modern computer systems can be highly complex, with large numbers of files and configurations that are constantly changing. Building a FIM that can handle this complexity and scale with the system can be challenging.

**False positives**: FIMs can generate a large number of alerts and notifications, which can be overwhelming for administrators and security personnel. Reducing false positives and ensuring that only relevant alerts are generated is a critical challenge in building an effective FIM.

**Resource utilization**: A FIM that continuously monitors all files and directories on a system can consume significant system resources, which can impact system performance. Balancing the need for comprehensive monitoring with resource utilization is a key challenge.

**Configuration management**: Keeping track of system configurations and changes can be complex, particularly in large organizations with multiple systems and users. Ensuring that the FIM can accurately track changes and report on configuration compliance can be challenging.

**Integration with existing systems**: FIMs need to be integrated with existing security and compliance systems, which can be challenging if the systems use different protocols or data formats.

**Keeping up with evolving threats**: Cyber threats are constantly evolving, and FIMs need to be updated regularly to keep up with new threats and attack methods.

* 1. **Overview**

This Investigation Report about the file integrity monitor is divided into multiple sections. In the first section, the reader will be introduced to the project background, the aims and objectives of the project, potential benefits, challenges, target user and deliverables.

In the next section of the report, in-depth research about the subject is conducted as part of the literature review, where domain research is conducted, and similar systems are compared.

In the next section of the report, technical research is conducted on the subject discussing about the programming language chose, preferred IDE, libraries and tools that will be used and the choice of Operating System.

In the next section of the report, the system development methodology is discussed in order to figure out what development approach will be adopted in order to successfully complete the project.

In the next section of the report, the research methods for data gathering is discussed and why the choice is made. The section also demonstrates the questionnaire and provides reasoning for the choice of questions.

In the next section of the report, a requirement validation is conducted where we analyse the data gathered from the questionnaire provided to participants and discuss how it impacts the project.

The final section of the report talks about conclusion and reflections on working on the first phase of this project and provides a summary of everything that has been discussed in this report.

* 1. **Project Plan**

Chart, bubble chart

Description automatically generated

*Fig. 1 Gantt chart of project plan*

A graph of a project

Description automatically generated*Fig. 2 Gantt chart of project implementation*

1. **Literature Review**
   1. **Introduction**

Every component of an operating system, including instructions, device drivers, and other data, is preserved in files. In a modern operating system environment, there are a ton of files. The majority of the time, files become the main focus of attackers that want to hack operating systems. The assault can be carried out by editing or changing existing files, as well as by deleting, adding, and hiding relevant files. Attackers can target files in the operating system environment using a variety of methods, making file security an essential duty. To do it, FIM implementation and other relevant system security solutions are required. (Udzir, N.I., 2011)

A data-centric security plan for an organization must include file integrity monitoring (FIM). FIM is the process of evaluating every attempt to access or edit files and folders holding sensitive information and determining if the attempt was permitted, lawful, and compliant with corporate and legal regulations. You may reduce security breaches by proactively recognizing improper and unapproved modifications and access events. As a result, numerous compliance standards call for FIM. (Melnick, J. 2020)

According to one source, File integrity monitoring creates a "digital fingerprint" by looking at several elements of a file. Afterward, it contrasts this fingerprint with a well-known baseline fingerprint. Although native auditing tools are available, they all typically have drawbacks, including, but not limited to, decentralized storage of security logs from various domain controllers, a lack of information about previous settings in the log entry, and an inability to recover the object or configuration from the audit log A good FIM tool will monitor all components of your IT environment, including:

* Network devices and servers
* Workstations and remote devices
* Databases, directories, OS, and middleware
* Cloud-based services
* Hypervisor configuration, and Active Directory
  1. **Domain Research**

The aim of this domain research is to gain an expert point of view on a what is a File Integrity Monitor, and to develop a comprehensive understanding of the key concepts, trends, challenges, and best practices in that domain. There are several key areas to discuss when conducting domain research for a File Integrity Monitor such as –

* + 1. Types of FIM

It is important to understand the different types of FIMs available in the market and their features. For example, there are host-based FIMs that monitor files on a single system and network-based FIMs that monitor files across multiple systems. There are also agent based FIMs that require software to be installed on each system and agentless FIMs that use network protocols to monitor files.

File Integrity Monitors (FIMs) are tools that ensure the integrity of files on a system or network by monitoring any changes made to them. There are primarily two types of FIMs: host-based and network-based FIMs. (*SolarWinds*, n.d.)

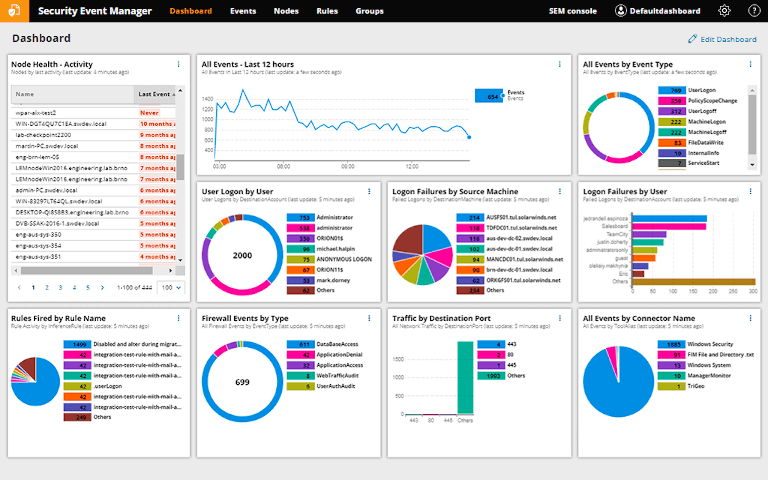
Host-based FIMs are installed on a single system and monitor files on that specific system. These FIMs require agents to be installed on the system where file monitoring needs to be performed. The agents monitor the files and report any changes in the system files to the central monitoring server. Host-based FIMs are suitable for small to medium-sized businesses and are known for their ease of deployment. (Singh & Singh, 2014)

On the other hand, network based FIMs monitor files across multiple systems on a network. These FIMs scan the network and identify the systems on it, and do not require agents to be installed on the monitored systems. Instead, they use network protocols to access and monitor files. This makes them more scalable and suitable for large organizations with a distributed IT infrastructure. (Singh & Singh, 2014)

Additionally, both host-based and network-based FIMs can be either agent-based or agentless. Agent-based FIMs require software to be installed on each system that needs to be monitored. The agent monitors the files and reports any changes to the central monitoring server. Agent-based FIMs are more effective in detecting file changes because they monitor files in real-time. However, they can impact system performance because of the additional processing required by the agent software.

In contrast, agentless FIMs do not require software to be installed on the monitored systems. Instead, they use network protocols such as SSH, SNMP, or WMI to access and monitor files. Agentless FIMs are less intrusive and do not affect system performance. However, they may be less effective in detecting file changes because they typically rely on periodic scans rather than real-time monitoring. (Singh & Singh, 2014)

Choosing the right type of FIM depends on the specific needs and requirements of the organization. The size and complexity of the IT infrastructure, the sensitivity of the data being monitored, and the level of intrusion allowed on the monitored systems are all factors that should be considered when selecting the appropriate type of FIM.



*Fig. 3 Realtime data from a FIM*

* + 1. Functionality and capabilities

It is important to explore the various functionalities and capabilities of FIMs. This includes features such as real-time monitoring, automated alerts, file tracking, version control, and reporting. A file integrity monitor (FIM) is a powerful tool that monitors the integrity of files on a system or network. Its main function is to ensure that files remain secure and have not been tampered with in any way.

Real-time monitoring is a key feature of FIMs. It allows administrators to monitor files in real-time, which provides immediate alerts to any changes made to files. This enables them to take corrective action before any damage is done. Configuration management is another important feature of FIMs. It helps ensure that files are correctly configured, reducing the risk of misconfigurations that could lead to security breaches. (Von Ogden, n.d.-b)

Compliance management is a critical feature of FIMs. It ensures that systems are compliant with industry and regulatory standards, which is essential for data security. Reporting is another important function of FIMs. They generate detailed reports on the changes made to files, including information on when changes were made, who made them, and what the changes were. These reports can be used for audit purposes and to track compliance.

Centralized management is also an important feature of FIMs. It provides a centralized management console where administrators can manage file monitoring across multiple systems or networks. This simplifies management and ensures consistency across the organization. FIMs can also integrate with Security Information and Event Management (SIEM) systems and Endpoint Detection and Response (EDR) systems, providing an additional layer of security and ensuring that changes made to files are detected and prevented. (Von Ogden, n.d.-b)

* + 1. Deployment models

File integrity monitors (FIMs) can be deployed in various ways, each with its own advantages and disadvantages. One common deployment model is the agent-based model, where an agent is installed on each system that needs to be monitored. The agent is responsible for monitoring changes to files on that system and reporting back to the centralized management console. This model provides high visibility into changes made on each individual system, but it can be complex to manage and can impact system performance.

Another deployment model is the agentless model, where there is no need to install an agent on each system. Instead, the FIM uses remote access protocols to monitor files on remote systems. This model provides simplified management and reduces the impact on system performance, but it may not provide the same level of visibility into changes made on each system.

A hybrid deployment model combines elements of both the agent-based and agentless models. The FIM may use agents to monitor changes on some systems and remote access protocols to monitor changes on other systems. This model provides a flexible approach that can be tailored to the needs of each individual system. (File Integrity Monitor (FIM), n.d.)

Cloud-based deployment models host the FIM in the cloud, and data is monitored from there. This model provides a high degree of flexibility and scalability, as the FIM can be accessed from anywhere with an internet connection. However, it may not be suitable for organizations that require a high level of control over their data.

On-premises deployment models install the FIM on-premises, and data is monitored locally. This model provides a high degree of control over data and may be more suitable for organizations with strict data security requirements. However, it may require significant resources to set up and manage.

Ultimately, the choice of deployment model for a FIM depends on the specific needs and requirements of each organization. Factors such as system complexity, data security requirements, and budget will all need to be considered when choosing the most appropriate deployment model. (File Integrity Monitor (FIM), n.d.)

* + 1. Integration with other security tools

File integrity monitors (FIMs) can integrate with various other security and IT systems to provide enhanced functionality and visibility into the security posture of an organization. One of the most common integrations for FIMs is with Security Information and Event Management (SIEM) systems. FIMs can send alerts to the SIEM when changes are detected, and the SIEM can correlate that information with other security events to provide a comprehensive view of the security landscape. This integration enables organizations to quickly identify and respond to potential security threats. (Integrating FIM and SIEM Solutions, 2021)

FIMs can also integrate with vulnerability scanning tools to identify potential security weaknesses in the system. By monitoring changes to critical files and comparing them to a known baseline, the FIM can detect unauthorized changes that may indicate a vulnerability has been introduced. This integration helps organizations to proactively address security weaknesses before they are exploited by attackers.

Configuration management tools can also be integrated with FIMs to ensure that changes made to the system are authorized and follow established policies. By monitoring changes to critical files, the FIM can alert administrators when unauthorized changes are made and ensure that any changes that are made follow established guidelines. This integration helps organizations to maintain consistent and secure configurations across their systems. (Integrating FIM and SIEM Solutions, 2021)

Compliance monitoring tools can also be integrated with FIMs to ensure that the system meets regulatory requirements. By monitoring changes to critical files and comparing them to a known baseline, the FIM can detect unauthorized changes that may indicate a violation of compliance requirements. This integration helps organizations to maintain compliance with industry regulations and avoid potential legal and financial penalties.

Finally, FIMs can integrate with cloud security platforms to provide a unified security monitoring solution across both on-premises and cloud-based systems. By monitoring changes to files in both environments, the FIM can provide a comprehensive view of the security landscape. This integration is particularly important for organizations that have adopted cloud-based systems and need to maintain consistent security monitoring across both on-premises and cloud environments. (Integrating FIM and SIEM Solutions, 2021)

 *Fig.4 Working of SIEM with FIM integration*

* + 1. Regulatory Compliance

File integrity monitoring (FIM) is an important security control for organizations that need to comply with various regulatory frameworks. FIM helps organizations to ensure that critical files and configurations remain unchanged and unauthorized changes are detected and investigated promptly. Several regulations mandate FIM as a requirement, including the Payment Card Industry Data Security Standard (PCI DSS), Sarbanes-Oxley Act (SOX), Health Insurance Portability and Accountability Act (HIPAA), and Federal Information Security Modernization Act (FISMA). (Von Ogden, n.d.-b)

PCI DSS is a security standard that applies to organizations that process credit card transactions. It requires FIM to be implemented for critical system files and logs to detect unauthorized access and changes to sensitive data. FIM can help organizations to ensure that cardholder data is protected and detect any unauthorized changes or access to that data.

SOX is a law that applies to publicly traded companies in the United States. It requires the implementation of internal controls to ensure the accuracy and integrity of financial reporting. FIM is an important control to ensure that financial data and system configurations remain unchanged, and unauthorized changes are detected and investigated promptly.

HIPAA is a regulation that applies to the healthcare industry in the United States. It requires the implementation of security controls to protect electronic protected health information (ePHI). FIM can help organizations to ensure that ePHI is protected and detect any unauthorized access or changes to that information. (Von Ogden, n.d.-b)

FISMA is a law that applies to federal agencies in the United States. It requires the implementation of security controls to protect federal information and systems. FIM can help federal agencies to ensure the integrity of critical system files and detect unauthorized changes or access to sensitive data. (Von Ogden, n.d.-b)

In addition to these regulations, there are other frameworks and standards that recommend the implementation of FIM as a security control, including the National Institute of Standards and Technology (NIST) Cybersecurity Framework and the Centre for Internet Security (CIS) Controls.

Overall, regulatory compliance is an important consideration for organizations implementing FIM. The choice of FIM solution should be carefully evaluated to ensure it meets the specific requirements of the regulatory framework that applies to the organization. Additionally, regular monitoring and reporting should be conducted to ensure ongoing compliance with the relevant regulations. (Von Ogden, n.d.-b)

* + 1. Industry trends

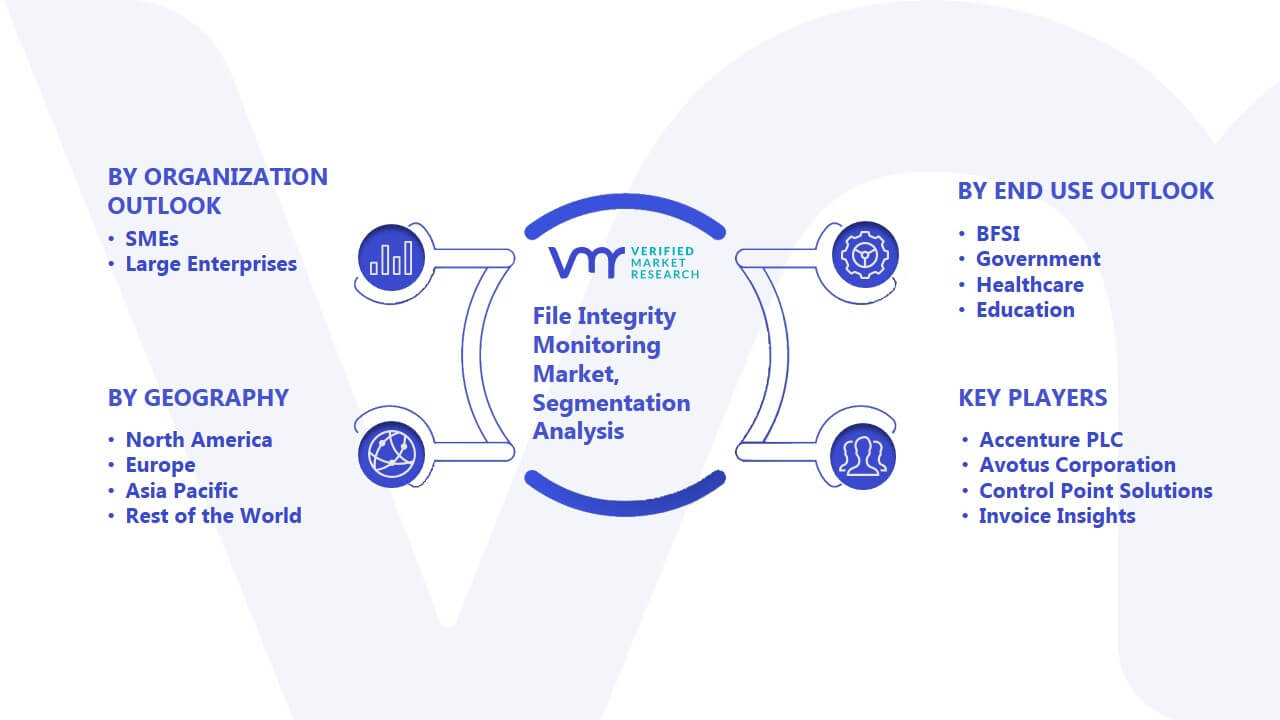
File integrity monitoring (FIM) is a crucial aspect of cybersecurity that helps organizations maintain the integrity of their files and systems. As the cybersecurity landscape continues to evolve, FIM solutions are also evolving to meet the changing needs of organizations. One of the current industry trends in FIM is the shift towards cloud-based solutions. Cloud-based FIM solutions offer greater scalability, flexibility, and cost-effectiveness than traditional on-premises solutions, making them a popular choice for many organizations. (Abdullah & Mahmod, 2011)

Another trend in FIM is the integration of automation and machine learning capabilities. FIM solutions that incorporate these capabilities can improve their ability to detect and respond to security threats, while also reducing the workload on security teams. This integration can also provide a more comprehensive view of an organization's security posture by integrating with other security tools such as SIEM, IDPS, and EDR systems. (FIM Market Analysis, n.d.)

Real-time monitoring is another trend in FIM that is becoming increasingly important for organizations. Real-time monitoring enables organizations to detect and respond to security incidents in real-time, rather than waiting until after the fact. This trend is driven by the need for organizations to detect and respond to security threats as quickly as possible. (FIM Market Analysis, n.d.)

Regulatory compliance is another important driver for FIM. As more regulations are introduced or updated, FIM solutions will need to evolve to meet these new requirements. Compliance with regulations such as HIPAA, PCI DSS, and GDPR are critical for many organizations and FIM solutions can help organizations achieve and maintain compliance. (FIM Market Analysis, n.d.)

In conclusion, FIM is a critical security control that is constantly evolving to meet the changing needs of organizations. Organizations that implement FIM should stay up to date on the current industry trends and ensure that their FIM solutions are able to meet their evolving security needs.



*Fig. 5 Industry trends of a FIM*

* 1. **Similar Systems**

Throughout security professionals have built and tested multiple systems focusing on the same issue. These systems may employ various functionality and security monitoring techniques, but they all aim to find and stop illegal changes to the data and systems that make up an organization's IT environment.

1. **Security Information and Event Management (SIEM) systems**

SIEM systems and file integrity monitors (FIMs) are two important security tools used in cybersecurity to protect against threats and attacks. They share some similarities, such as detection capabilities, real-time monitoring, centralized data storage, and alerting and reporting. Both tools can detect security events and incidents, monitor the system in real-time, collect and store data from various sources in a centralized location, and generate alerts and reports based on predefined rules and thresholds. However, SIEM is designed to monitor security events across the entire IT infrastructure, while FIM focuses specifically on monitoring changes to critical system files and directories. SIEM provides a broader view of security incidents by using a wider range of data sources, such as network traffic data and user behaviour data, while FIM is more specialized in its monitoring capabilities, limited mainly to file and directory monitoring. (File Integrity Monitoring and SIEM - Logsign, n.d.)

1. **Endpoint Detection and Response (EDR) systems**

Both EDR and FIM focus on endpoint security, which means protecting individual devices such as computers and mobile devices. EDR systems and FIMs can detect unauthorized changes or modifications to system files, which are often indications of malware or other types of attacks. Both tools can monitor devices in real-time, detect threats, and provide alerts or notifications to administrators. Additionally, both EDR and FIM are capable of providing detailed forensic data for investigations of security incidents. While EDR is a more comprehensive tool that includes additional features such as behavioural analysis and incident response capabilities, FIM is more specialized in monitoring file integrity. Overall, both tools can enhance endpoint security and provide organizations with valuable insights into potential threats and attacks. (AT&T Cybersecurity, 2023)

1. **Intrusion Detection Systems (IDS)**

Both IDS and FIM focus on identifying and detecting potential security breaches within a system. IDS systems monitor the network traffic and system logs in real-time to detect and identify suspicious activities or patterns that may indicate an attack. Similarly, FIMs monitor changes to critical system files and directories and alert administrators when unauthorized modifications occur, indicating potential malicious activity. Both tools provide early warning of potential threats and can help identify the scope of a security breach. While IDS is more comprehensive and can detect a broader range of threats, including both internal and external attacks, FIM is more specialized in monitoring file integrity. Overall, both tools can provide valuable insights into potential threats and help organizations enhance their security posture. (AT&T Cybersecurity, 2023)

1. **Security Configuration Management (SCM) systems**

SCM systems focus on managing and monitoring the security configurations of IT systems and networks, ensuring that they meet established security policies and standards. FIMs, on the other hand, focus on monitoring changes to critical system files and directories to detect any unauthorized modifications. Both tools help in identifying potential security risks by ensuring that system configurations and file integrity are maintained according to established policies and standards. Both tools provide real-time monitoring capabilities, generate alerts, and allow for the quick identification and remediation of security issues. Additionally, both tools maintain a centralized repository of data, allowing administrators to track changes and maintain an audit trail for compliance purposes. Overall, SCM and FIM can work together to provide a comprehensive approach to security management and monitoring, ensuring that organizations can identify and address potential security threats proactively. (AT&T Cybersecurity, 2023)

1. **Vulnerability Assessment and Management (VAM) systems**

VAM systems focus on identifying and managing vulnerabilities in a system or network, while FIMs focus on detecting unauthorized changes to critical system files and directories. Both tools provide real-time monitoring capabilities, generate alerts, and allow for the quick identification and remediation of security issues. Additionally, both tools maintain a centralized repository of data, allowing administrators to track changes and maintain an audit trail for compliance purposes. By identifying and addressing vulnerabilities and monitoring changes to critical files, organizations can take proactive measures to mitigate potential security risks. Overall, VAM and FIM can work together to provide a comprehensive approach to security management and monitoring, helping organizations to maintain the security and integrity of their systems and data. (AT&T Cybersecurity, 2023)

* 1. **Summary**

File integrity monitoring (FIM) is the process of evaluating every attempt to access or edit files and folders holding sensitive information and determining if the attempt was permitted, lawful, and compliant with corporate and legal regulations. It creates a "digital fingerprint" by looking at several elements of a file and comparing it with a well-known baseline fingerprint. Native auditing tools have drawbacks, such as decentralized storage of security logs, lack of information about previous settings, and inability to recover the object or configuration from the audit log. A good FIM tool will monitor all components of the IT environment.

There are primarily two types of FIMs: host-based and network-based FIMs. Host-based FIMs are installed on a single system and monitor files on that specific system. These FIMs scan the network and identify the systems on it, and do not require agents to be installed on the monitored systems. Agent-based FIMs are more effective in detecting file changes because they monitor files in real-time.

A file integrity monitor is a powerful tool that monitors the integrity of files on a system or network. Real-time monitoring is a key feature of FIMs. It allows administrators to monitor files in real-time, which provides immediate alerts to any changes made to files. Reporting is another important function of FIMs. They generate detailed reports on the changes made to files, including information on when changes were made, who made them, and what the changes were. Centralized management is also an important feature of FIMs. It provides a centralized management console where administrators can manage file monitoring across multiple systems or networks.

As the cybersecurity landscape continues to evolve, FIM solutions are also evolving to meet the changing needs of organizations. Cloud-based FIM solutions offer greater scalability, flexibility, and cost-effectiveness than traditional on-premises solutions, making them a popular choice for many organizations. Regulatory compliance is another important driver for FIM. As more regulations are introduced or updated, FIM solutions will need to evolve to meet these new requirements. Compliance with regulations such as HIPAA, PCI DSS, and GDPR are critical for many organizations and FIM solutions can help organizations achieve and maintain compliance.

The similar systems to a FIM include –

* Security Information and Event Management (SIEM) systems
* Endpoint Detection and Response (EDR) systems
* Intrusion Detection Systems (IDS)
* Security Configuration Management (SCM) systems
* Vulnerability Assessment and Management (VAM) systems

1. **Technical Research**
   1. **Programming Language**

The programming language used to build a file integrity monitor (FIM) can vary depending on the specific tool and platform being used. However, some commonly used programming languages for building FIMs include C, C++, Python, and Java. C and C++ are commonly used for building FIMs for Linux and Unix systems, while Python and Java are often used for developing FIMs for Windows and other platforms. The choice of programming language often depends on factors such as the performance requirements of the FIM, the complexity of the monitoring rules and algorithms, and the compatibility with the target platform.

**Building a FIM using Python**

Building a file integrity monitor (FIM) using Python can provide several advantages. Python is a high-level programming language that is easy to learn and use, which makes it ideal for developers of all skill levels. Its portability allows for the development of FIMs that can run on different operating systems. Python has a large community of developers who have created libraries and tools that simplify the FIM development process. Python's dynamic nature allows for rapid prototyping and testing of the FIM, which helps speed up the development process. Python can be easily integrated with other security tools, such as intrusion detection systems and security information and event management systems, making it a powerful language for building reliable and effective FIMs.

**Building a FIM using Windows PowerShell**

To build a file integrity monitor (FIM) using Windows PowerShell, the built-in features for monitoring file system events and calculating hash values can be utilized. The "FileSystemWatcher" class can be used to monitor file changes, and the "Get-FileHash" cmdlet can be used to calculate hash values. By comparing the hash value of a file to an expected value, PowerShell can generate alerts or notifications based on specific criteria set by the developer. Additionally, PowerShell can maintain logs of file system events and corresponding hash values to track changes and detect unauthorized modifications. Building a FIM with PowerShell is a straightforward process, and it can be a cost-effective solution since PowerShell is included with Windows and doesn't require additional software.

In this project however, Windows PowerShell will be the choice for a programming language as it is more convenient to use since it is an integrated language and use of additional tools or libraries are unnecessary and as mentioned above, PowerShell can generate alerts or notifications based on specific criteria set by the developer and maintain logs of file system events and corresponding hash values to track changes and detect unauthorized modifications. (Z, 2022)

* 1. **IDE**

Windows PowerShell does not have a specific Integrated Development Environment (IDE) dedicated to it, so Windows PowerShell Integrated Scripting Environment (ISE) will be the choice for an Interactive Development Environment (IDE) as it is a free and built-in tool in Windows that offers an environment for writing, testing, and debugging PowerShell scripts. It features a script editor, console, and an integrated debugger.

* 1. **Libraries and Tools**

Several in-built libraries will be used to build the File Integrity Monitor (FIM) –

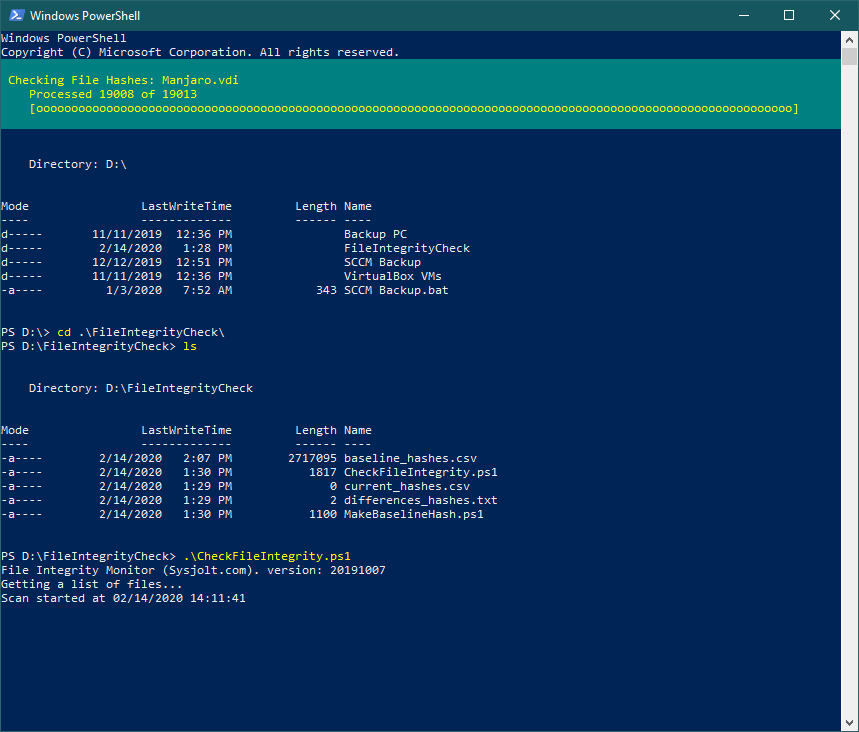
**Get-FileHash**: This is a built-in cmdlet in PowerShell that calculates the hash value of a file. It supports several hashing algorithms, including SHA1, SHA256, SHA384, SHA512, and MD5.

**FileSystemWatcher**: This is a built-in class in PowerShell that can be used to monitor file system events, such as file creation, modification, or deletion.

**Send-MailMessage**: This is a built-in cmdlet in PowerShell that can be used to send email notifications when a file integrity violation is detected.

**Windows Event Logs**: PowerShell can interact with Windows Event Logs to log events when file system changes are detected.

In addition to these built-in tools, developers can also use external libraries and modules in PowerShell to enhance their FIM solutions. For example, the "Pester" module can be used for unit testing, and the "PowerShell Logging" module can be used to log events to a central log management system. By leveraging these built-in libraries and tools, developers can build a powerful and effective FIM solution using hashing algorithms in Windows PowerShell. (Z, 2022)



*Fig. 6 FIM with Windows PowerShell*

* 1. **Operating System**

For the purposes of this project, Windows 11 will be the Operating System chosen as Windows PowerShell is integrated to it along with all the libraries and tools that will be used in the building of the File Integrity Monitor (FIM).

* 1. **Summary**

In order to successfully build the File Integrity Monitor, the programming language to be used will be Windows PowerShell which is an integrated language in windows operating systems. . The choice of programming language often depends on factors such as the performance requirements of the FIM, the complexity of the monitoring rules and algorithms, and the compatibility with the target platform. It is a more convenient way to build the FIM since it has various tools and libraries integrated within the IDE. In addition to having built in libraries, external modules could be added to improve the project or add more features.

1. **System Development Methodology**

There are several system development methodologies that can be used for building a file integrity monitor, depending on the specific requirements and preferences of the development team.

* Waterfall model: This is a linear and sequential approach that involves completing each phase of the development process before moving on to the next phase. This methodology may be suitable if the requirements are well-defined and there is little risk of changes during the development process.
* Spiral model: This is a risk-driven approach that involves building a prototype and then refining it through multiple iterations. This methodology may be suitable if there are significant risks associated with the development process, such as uncertainty around the requirements or the need to integrate with complex systems.

For the purposes of this project, the waterfall model will be the choice for a system development methodology. The Waterfall model emphasizes documentation, which can be beneficial for a file integrity monitor, making it easier to maintain and troubleshoot the system over time. Finally, the Waterfall model is less prone to scope creep than other development methodologies, reducing the risk of requirements changing during the development process. Overall, the Waterfall model can be advantageous for building a file integrity monitor when the project requirements and goals are well-defined and a linear and sequential approach is appropriate.

In this approach, the development process is divided into a sequence of distinct phases: requirements gathering, design, implementation, testing, and deployment. Each phase is completed before moving on to the next, with little opportunity to go back to earlier stages once they are complete.

The first phase is requirements gathering, where the development team works with stakeholders to define the goals and requirements for the file integrity monitor. This includes identifying the types of files that need to be monitored, the types of events that should trigger alerts, and any other specific requirements. Once the requirements are documented, the team moves on to the design phase.

In the design phase, the team creates a detailed plan for the file integrity monitor. This includes designing the architecture, choosing technologies, and mapping out how the system will work. The team then moves on to the implementation phase, where they build the system according to the design.

Once the implementation is complete, the team moves on to testing. In this phase, the file integrity monitor is thoroughly tested to ensure that it meets the requirements and works as expected. This includes testing for different types of events and edge cases to ensure that the system is robust and reliable.

Finally, in the deployment phase, the file integrity monitor is released to production. The system is installed, configured, and integrated with other tools and systems as necessary. Once the file integrity monitor is in use, ongoing maintenance and support are provided to ensure that it continues to function as intended.

Text

Description automatically generated

*Fig. 7 Flow chart for system development*

Diagram, logo, company name

Description automatically generated

*Fig. 8 Working of a FIM*

The FIM works by creating a baseline of the files on the system and then continuously monitoring them for any changes. When the FIM scans the files on the system, it calculates a unique hash value for each file using a hashing algorithm such as SHA-256 or MD5. A hash value is a digital fingerprint that represents the contents of a file. Even a small change to the contents of a file will result in a different hash value being generated.

The hash values of the files on the system are then stored in a database or file for later reference. This baseline is considered the trusted state of the system, and any changes made to files will be compared to this baseline.

As the system is used, the FIM continuously monitors the files and recalculates the hash values at regular intervals. If any changes are made to a file, the FIM will detect it by comparing the new hash value to the baseline hash value. If the hash values are different, the FIM will generate an alert indicating that a change has been made to the file. This allows administrators to quickly identify any unauthorized changes to files on the system, including modifications made by malware, hackers, or insider threats. By using hashing algorithms to detect changes, the FIM provides an effective means of maintaining the integrity and security of files on a system. (Von Ogden, n.d.-d)

1. **Research Methods**
   1. **Introduction**

The data gathering and analysis for this project will aim towards measuring participants knowledge about the File Integrity Monitor and related cyber security events in order to better understand if the survey can yield expected results about how advantageous the system will be for them and collect suggestions or opinions, they would have for the system in order to develop a FIM that better suits the daily needs of anyone even with basic knowledge about cyber security. The method used will be questionnaire using Google Forms as this helps to better analyse the collected data in order to improve and apply modifications to the system.

The deliverables of this system will mainly focus on being able to monitor given files in a server, folder, or database to detect any kind of changes or tampering in order to alert the user of threats or cyber-attacks on the system.

* 1. **Design**

In this section we will be discussing the questionnaire prepared for the survey with a short description of the main focus of each question and the expected results in order to progress with the build of the File Integrity Monitor.

Table

Description automatically generated with low confidence

The questionnaire requires the participant to choose their age group to understand the demographic of the target audience as it helps relate certain age groups with other questions that are part of this survey.

Graphical user interface, text, application, chat or text message

Description automatically generated

The gender is asked to specify in order to understand the demographic of participants who commonly use security tools or who are related to the cyber security field.

Graphical user interface, text, application

Description automatically generated

The current occupation of participants is necessary to understand and measure their knowledge on the subject and to analyse if they have ever come across a security event or happen to work with security tools

A picture containing chart

Description automatically generated

The field of study or work gives more insight into the background of the participants to better understand how cyber security is dealt with in different fields.

A picture containing graphical user interface

Description automatically generated

This is a basic question to understand the participants basic knowledge about security events or tools that are used to prevent breaches or attacks.

A screenshot of a computer

Description automatically generated with medium confidence

This question seeks to understand what participants think about security in an organizations and to see if they actually consider it important.

Graphical user interface, text

Description automatically generated

This question is asked to understand if participants have been victims of cyber attacks to gather more insight on their background of cyber security.

Graphical user interface, application

Description automatically generated

This question is asked to understand the preferences of participants in the event of a breach.

Graphical user interface, application

Description automatically generated

As the survey progresses, the questions become more specific towards cyber security and FIM and seeks to understand if participants think using a FIM is beneficial.

Text

Description automatically generated

This is a personal question to see if participants have prior knowledge or experience with a FIM.

Graphical user interface

Description automatically generated with medium confidence

This question leans towards understanding if participants are satisfied with security tools already available to them in order to understand if the addition of a FIM would be beneficial to them

Graphical user interface, text

Description automatically generated

This question is asked to understand preferences of participants on how they would like to be notified in the event of a breach or an attack.

Text

Description automatically generated

The final question of the survey gives the participants the freedom of suggesting their own ideas on how they would like to improve the system to better suit their needs.

* 1. **Summary**

The data gathering and analysis for this project will aim towards measuring participants knowledge about the File Integrity Monitor and related cyber security events in order to better understand if the survey can yield expected results about how advantageous the system will be for them and collect suggestions or opinions, they would have for the system in order to develop a FIM that better suits the daily needs of anyone even with basic knowledge about cyber security. The results of the survey can help organizations make informed decisions about which file integrity monitoring tool is the most suitable for their specific needs and requirements. The questions in this survey include both personal and general questions to better understand the demographic of participants and to analyse their knowledge and opinions on the subject.

1. **Requirements Validation**
   1. **Analysis of Data**

The following graphs and charts represent the data analysis from the survey that was collected from 32 participants. In this section we will break down how the collected data will be taken into consideration while building the file integrity monitor.

Chart, pie chart

Description automatically generated

The pie chart above shows the participants of this survey are mostly in the age group of 18-24 with a 96.9% and a small 3.1% of participants in 25-24 category. The majority of participants of this survey are youth between the ages of 18 and 24 who use devices for their day to day needs.

Chart, pie chart

Description automatically generated

The majority of participants of this survey are of the Male gender with 84.4% and 9.4% are of the female gender. 6.3% of the participants have opted not to mention their gender.

Chart, pie chart

Description automatically generated

75% of the participants of this survey are students who are undertaking various courses and 18.8% of them are employed in various fields. 6.3% of the participants are also doing their internships at the time of this survey collected.

Chart, bar chart

Description automatically generated

The participants of this survey are from various fields of work or study. The most notable fields from this graph are Medicine, Engineering, and IT. With participants from various fields involved, there is a better scope of understanding of the exposure of security tools and events among the general public.

Chart, bar chart

Description automatically generated

While most of the participants (37.5%) have a basic idea of cyber attacks and breaches, 18.8% of the participants have either very less knowledge or no idea at all about security events happening globally. Only 25% of the participants are aware or well informed in this situation which shows how people in different fields are less aware of the importance cyber security.

Chart

Description automatically generated

59.4% of the participants believe cyber security is very important to an organizations while the remaining have mixed opinions on the role security plays in an organizations. This question analyses how well organizations outside the IT sector consider cyber security and if they are prepared in the event of a breach or attack.

Chart, pie chart

Description automatically generated

62.5% of the participant have been victims of some form of cyber attack and 37.5% has replied ‘No’ to this question. This could be due to the lack of awareness of how one is attacked and how data breaches or attacks take place. Or they could be part of the lucky few who have never been a victim of cyber-attacks.

Chart, waterfall chart

Description automatically generated

59.4% of the participants feel it is very beneficial to be notified in the event of a breach or attack in order to take countermeasures, so the situation does not get worse. Meanwhile the remaining 40.6% of the participants has a neutral opinion or feels it would be somewhat helpful to receive notifications about the situation.

Chart, waterfall chart

Description automatically generated

Majority of the participants (64.5%) feel having a file integrity monitor would hugely benefit an organization to keep their systems more secure from data leaks or cyber attacks while the remaining has mixed opinions about the system and a small 3.2% believe it isn’t going to make much of a difference to the security of an organization.

Chart, pie chart

Description automatically generated

Among the participants of the survey, a huge majority of 90.6% has not had any kind of exposure to a File Integrity Monitor and has less knowledge about the subject. Whereas a small 9.4% of the participants have either used or experimented with a FIM, potentially belonging to the cyber security field of study or work.

Chart, bar chart

Description automatically generated

Upon analysing the above bar graph, it is evident that only 9.4% of participants have checked the integrity of their file manually at least a few times whereas the remaining participants has barely ever tried to do so. And a 21.9% of the participants have never checked the integrity of files before showing that a having FIM in place would help them keep their files more secure since the process would be automated.

Chart

Description automatically generated

65.6% of the participants have a neutral opinion about the security of their devices whereas 12.5% believes the security features on their devices are not enough in the event of a breach or an attack. Meanwhile, 21.9% of the participants are satisfied with the security features offered to them on their devices.

Chart, pie chart

Description automatically generated

This pie chart shows the preferences of the participants on how they want to be notified in case their system file have been altered which would cause a system breach.

31.3% of participants prefer to be notified via Email

28.1% of participants prefer to be notified via SMS

21.9% of participants prefer to be notified via a Mobile App

18.8% of participants prefer to be notified via their Dashboard

In either of the cases mentioned above, the main focus of the participants is to receive notifications as soon as possible in the event of a breach or attack in order to take countermeasures.

Participants were also given the option to provide suggestions on what kind of modifications or improvements they would like to see on the proposed project. While some did not have any suggestions for the project, some of the participants have suggested some interesting improvements or features for the FIM project. Some of them are –

* Scan the file to check malicious content
* Quick report
* Instant Lock protocol after breach
* Countermeasures to a breach
* It gives me alert on any files that have been modified or changed without my awareness based on abnormal behaviour by comparing to my previous system behaviour
* More AI based security
  1. **Summary**

Majority of the participants have some knowledge about cyber security events, and either been a victim of data leak or cyber-attack. The suggestions provided by participants are useful to improve the system and knowing preferences is a good way to build a project for the target audience. This survey has helped gather feedback from participants on the effectiveness of FIM in detecting and preventing security breaches. By asking questions related to the number and types of incidents detected, the survey helped gauge the potential success of FIM implementation.

1. **System Architecture**
   1. **Introduction**

The File Integrity Monitor is designed as a comprehensive tool to keep track of changes made to files in a specific folder. Imagine it as a watchful guardian that maintains the security and integrity of critical files within your system. It does this by creating a baseline list that stores unique cryptographic signatures (hashes) of each file in the designated folder.

Once the baseline is established, the File Integrity Monitor enters monitoring mode. During monitoring, it periodically checks for any new files or changes to existing ones. If it detects any alterations, it immediately alerts you through console messages and email notifications. This proactive approach helps you quickly identify and respond to any unauthorized modifications, enhancing the overall security of your files.

Furthermore, the File Integrity Monitor is not just a silent observer; it keeps you informed with detailed reports. These reports provide a comprehensive overview of file paths and their corresponding hashes, making it easier for you to track file activity and ensure the overall health of your system.

In essence, the File Integrity Monitor acts as a reliable guardian for your files, providing peace of mind and strengthening the security of your data. It is an essential tool for anyone concerned about data integrity and safeguarding their system from potential threats.

Baseline Creation:

The system begins by creating a baseline list, which acts as a reference for the initial state of files in the monitored directory.

During this phase, the system calculates cryptographic hashes (unique signatures) for each file and stores them in the baseline list.

Monitoring Mode:

Once the baseline is established, the system switches to monitoring mode, where it continuously observes the target directory for any changes.

At regular intervals, it scans the directory to detect new files, modified files, or deleted files.

Hash Comparison:

During monitoring, the system recalculates the cryptographic hashes for each file in the directory and compares them with the corresponding hashes stored in the baseline list.

If any differences are found, indicating a change in file content, the system identifies the affected file(s).

Real-time Alerts:

When a change is detected, the system triggers real-time alerts to notify system administrators or users.

These alerts can be sent through various channels, such as console messages or email notifications, ensuring immediate awareness of file modifications.

Report Generation:

To provide a comprehensive overview of file activity, the system generates detailed reports.

These reports include information on file paths, timestamps, and the nature of changes, aiding administrators in analysing and responding to potential security incidents.

Continuous Monitoring:

The File Integrity Monitor operates as an ongoing process, continuously observing the directory and updating the baseline list with any new or modified files.

This continuous monitoring approach ensures a proactive and vigilant stance against unauthorized file alterations.

* 1. **Abstract Architecture** 
     1. System Design

The File Integrity Monitor is designed to ensure the security and integrity of files within a specified target directory. Its main objective is to detect any unauthorized changes to files, whether it's a creation, modification, or deletion, and notify the system administrator in real-time about these events. The system design incorporates various functionalities, including baseline creation, continuous monitoring, log generation, and email notification.

Baseline Creation: When the system is initialized, the administrator has the option to collect a new baseline or use a saved baseline for monitoring. If the administrator chooses to collect a new baseline (Option A), the system calculates the cryptographic hashes for all files within the target directory. These hashes represent a unique fingerprint of each file and are used as a reference for future comparisons. The baseline data is then stored in a file called "baseline.txt," associating each file path with its corresponding cryptographic hash.

Continuous Monitoring: If the administrator selects the option to begin monitoring with a saved baseline (Option B), the system starts the continuous monitoring process. The system uses a polling mechanism to regularly check the target directory for any file changes. At specified intervals, the system recalculates the cryptographic hash for each file in the target directory and compares it with the baseline data stored in "baseline.txt."

File Change Detection: During the monitoring process, the system detects file changes based on the comparison between the current file hashes and the hashes stored in the baseline. If a new file is created, the system generates a log entry and sends an email notification to the administrator. Similarly, if a file is modified, the system updates the baseline with the new hash and notifies the administrator. This real-time monitoring ensures that any unauthorized changes to files are promptly identified.

Log Generation and Reporting: The system maintains a monitoring log that records all file-related events, including file creations, modifications, and deletions. Each log entry contains details such as the timestamp, file path, and the type of change. Additionally, the system generates a comprehensive file monitoring report that summarizes all the detected changes and baseline comparisons. This report aids in auditing and compliance efforts, allowing the administrator to review the system's integrity status over a specific period.

Email Notification: Email notification is a crucial aspect of the system design. The system interacts with an email server to send real-time notifications to the administrator's email address. Whenever a file change is detected, the system composes an email containing relevant details about the event and sends it to the administrator. This ensures that the administrator stays informed about any suspicious activities or integrity violations within the monitored directory.

Use Case Diagram

A screenshot of a phone

Description automatically generated

Class Diagram

A screenshot of a computer screen

Description automatically generated

A screenshot of a diagram

Description automatically generatedActivity Diagram

* + 1. Interface Design

**Main Menu:**

**A blue screen with white text

Description automatically generated**

**A computer screen with white text

Description automatically generated**

Description: The main menu serves as the entry point for the system and allows the user to choose between two main options - Collect New Baseline and Monitor Files with Saved Baseline.

Interface Elements:

Title: File Integrity Monitor System

Menu Options:

Collect New Baseline

Monitor Files with Saved Baseline

User Input Prompt: "Please enter the corresponding option number: "

**Collect New Baseline Interface:**

**A blue screen with white text

Description automatically generated**

Description: This interface guides the user through the process of collecting a new baseline for the target files.

Interface Elements:

Title: Collect New Baseline

Instructions: "Please wait while the system calculates file hashes for the target files..."

Progress Bar: (Displays the progress of baseline collection)

Status: (Displays the current status, e.g., "Calculating hash for File 1 of 10...")

Completion Message: "Baseline collection completed successfully."

Button: "Back to Main Menu"

**Monitor Files with Saved Baseline Interface:**

A computer screen with white text

Description automatically generated

Description: This interface initiates the file monitoring process with the saved baseline and continuously updates the user about any file changes or deletions.

Interface Elements:

Title: Monitor Files with Saved Baseline

Instructions: "Monitoring files for changes. Press Ctrl+C to stop monitoring."

Alert Log:

Displays real-time alerts for file changes or deletions.

Each alert is time-stamped and color-coded based on the event type (e.g., green for file creation, yellow for file change, red for file deletion).

Scrolling mechanism to display multiple alerts.

Report Generation Status: (Displays whether a report has been generated for the current monitoring session)

Button: "Stop Monitoring and Generate Report"

Button: "Back to Main Menu"

**File Change Report Interface:**

**A screenshot of a computer

Description automatically generated**

Description: This interface displays a detailed report of file changes and deletions detected during the monitoring process.

Interface Elements:

Title: File Change Report

Report Content: (Displays the report content generated during monitoring)

Button: "Send Report via Email"

Button: "Back to Monitoring"

**Send Report via Email Interface:**

**A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated**

Description: This interface allows the user to send the generated file change report via email.

Interface Elements:

Title: Send Report via Email

Input Fields:

Recipient Email Address

Subject (editable with default value: "File Monitoring Report and Alerts")

Body (editable with default value: "Please find the attached file monitoring report.")

Attachment: (Displays the file change report to be attached)

Button: "Send Email"

Button: "Back to File Change Report"

**Error Handling Interface:**

A screen shot of a computer

Description automatically generated

Description: This interface is displayed when errors occur during baseline collection, monitoring, or report generation.

Interface Elements:

Title: Error

Error Message: (Displays the specific error message)

Button: "OK" (Closes the error message and returns to the previous interface)

Flow of Interactions:

* User starts the application and is presented with the Main Menu.
* User selects Collect New Baseline and is taken to the Collect New Baseline Interface.
* The system calculates file hashes for the target files and displays a progress bar and status.
* Once baseline collection is complete, the system shows a completion message.
* User can go back to the Main Menu from the Collect New Baseline Interface.
* User selects Monitor Files with Saved Baseline and is taken to the Monitor Files with Saved Baseline Interface.
* The system starts monitoring the target files using the saved baseline and displays real-time alerts for any file changes or deletions.
* User can stop monitoring and generate a file change report by clicking the corresponding button.
* The system displays the File Change Report Interface with the detailed report.
* User can send the report via email by providing recipient email address and clicking the "Send Email" button.
* The system sends the email with the attached report and provides feedback to the user.
* User can go back to the File Change Report Interface or return to monitoring from the Send Report via Email Interface.
* If any errors occur during the process, the system displays the Error Handling Interface with relevant error messages and provides the option to continue or go back.

Note: The storyboard can be adapted to the specific implementation and user interface design preferences. The described interfaces aim to provide a user-friendly and intuitive experience for the File Integrity Monitor system.

1. **Project Plan**
   1. **Features**

The File Integrity Monitor is an invaluable asset for organizations aiming to strengthen data security and maintain the integrity of their files. By providing real-time monitoring, automated baseline maintenance, and comprehensive reporting, it enables proactive security measures, early threat detection, and efficient incident response.

* + 1. Monitoring Mode

After establishing the baseline, the system enters monitoring mode. It continuously observes the target directory for any changes to files, including new files, modified files, or deleted files. The monitoring process operates at specified intervals, allowing real-time detection of file alterations.

* + 1. Hash Comparison

During monitoring, the system recalculates cryptographic hashes for all files in the directory. It then compares these hashes with the corresponding hashes in the baseline list. If any discrepancies are found, it indicates changes to file content.

* + 1. Real-time Alerts

In the event of file alterations, the File Integrity Monitor triggers real-time alerts to notify administrators or users. These alerts can be delivered through console messages or email notifications, providing immediate awareness of any unauthorized modifications.

* + 1. Comprehensive Reporting

The system generates detailed reports that offer insights into file activity. The reports include file paths, timestamps, and the nature of changes, allowing administrators to analyse and respond to security incidents effectively.

* + 1. Automated Baseline Maintenance

The system automatically maintains and updates the baseline list, reflecting changes to files in the monitored directory. This automation reduces manual intervention and ensures the baseline remains accurate over time.

* + 1. Customizable Hash Algorithm

Users have the flexibility to choose the cryptographic hash algorithm for calculating file hashes. The default algorithm is SHA512, but the system can be customized to use other algorithms based on specific security requirements.

* + 1. User-friendly Command Line Interface (CLI)

The File Integrity Monitor offers an intuitive CLI that allows users to interact with the system effortlessly. The CLI supports easy configuration, baseline creation, and monitoring initiation.

* + 1. Security-enhancing Tool

The File Integrity Monitor serves as a powerful security-enhancing tool, safeguarding critical files from unauthorized modifications and malicious activities. Its proactive approach helps detect potential breaches early, reducing the risk of data tampering or security breaches.

* + 1. Open for Extension

The system is designed with modularity and extensibility in mind. Additional functionalities and customizations can be easily incorporated to cater to specific organizational needs and future enhancements.

* 1. **Details of Release Plans**
     1. Version 1.0 of File Integrity Monitor

The first version of this system will launch towards the final week of July 2023. The first part of the release focuses on implementing the core functionalities of the File Integrity Monitor to provide a reliable file integrity checking system. The release would have the following features:

Baseline Creation and Monitoring:

* Implement the baseline creation process to calculate and store cryptographic hashes for each file in the target directory.
* Develop the monitoring mode to periodically scan the directory and detect new, modified, or deleted files.

Real-Time Alerts:

* Integrate real-time alerting mechanisms to notify administrators via console messages for file changes detected during monitoring.
* Set up email notification capability for immediate alerts to administrators about any unauthorized file alterations.

Hash Comparison and Reporting:

* Develop the hash comparison module to verify file integrity by comparing current hashes with those stored in the baseline.
* Implement basic reporting functionality to display file paths, timestamps, and change details for administrators' review.

User-friendly CLI:

* Design a user-friendly Command Line Interface (CLI) to allow easy configuration of the system, baseline creation, and monitoring initiation.
* Ensure clear and concise commands for smooth interaction with the File Integrity Monitor.

Automated Baseline Maintenance:

* Integrate automatic baseline maintenance to update the baseline list with any changes in the monitored directory.
* Verify that the system accurately reflects the current state of files in the directory.
  + 1. Version 2.0 of File Integrity Monitor

The second version of this system will launch towards the first week of August 2023. The second part of the release aims to enhance the File Integrity Monitor's capabilities, usability, and customization options. The release would have the following features:

Customizable Hash Algorithms:

* Add support for customizable hash algorithms, allowing users to choose their preferred cryptographic hash method.
* Implement flexibility to use other algorithms in addition to the default SHA512.

Comprehensive Reporting:

* Enhance reporting functionality to generate detailed and insightful reports on file activity.
* Include additional metadata, such as file size, permissions, and user access, to aid administrators in analysing security incidents.

Configurable Polling Intervals:

* Introduce configurable polling intervals for monitoring, allowing users to set the frequency at which the system scans the directory.
* Enable administrators to fine-tune the monitoring process based on their specific monitoring requirements.

Modularity and Extensibility:

* Enhance the system's architecture to support modularity and extensibility.
* Allow for seamless integration of additional functionalities and customizations to cater to diverse organizational needs.

Graphical User Interface (GUI):

* Develop a user-friendly Graphical User Interface (GUI) alongside the existing CLI for an intuitive and visually appealing user experience.
* Ensure that both CLI and GUI versions share consistent functionality and usability.

Performance Optimization:

* Conduct performance testing and optimization to ensure the File Integrity Monitor operates efficiently, even with large-scale file systems.
* Address any bottlenecks and improve overall system responsiveness.

Compliance and Security Standards:

* Conduct thorough security assessments and ensure compliance with industry-standard security best practices.
* Obtain relevant certifications to assure users of the system's security and reliability.

By dividing the development and enhancements into two parts, the File Integrity Monitor can be released in a timely manner while ensuring that core functionalities are available from the initial release. The second part will focus on improving user flexibility, reporting, and overall performance, making the system a comprehensive and highly adaptable solution for file integrity monitoring.

* 1. **Test Plan**
     1. Unit Testing

This is a sample test case scenario for the unit testing. In this table, the Monitor Files function is being tested based on requirements and test cases.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Case | Expected Result | Actual Result | Status |
| 6.1 | Test if the function correctly monitors the target files and detects file changes |  |  |  |
| 6.2 | Test if the function updates the fileHashDictionary correctly during monitoring. |  |  |  |
| 6.3 | Test if the function handles different polling intervals and verifies the accuracy of file change detection. |  |  |  |

* + 1. User Acceptance Testing

This is a sample test plan for user acceptance testing. This form helps to attain user feedback and make necessary changes to system based on practicality of improvements.

Tester Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Occupation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Start Time: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ End Time: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | RATING | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** |
| User Interface |  |  |  |  |  |  |
| Meeting Objectives |  |  |  |  |  |  |
| Security |  |  |  |  |  |  |
| Bug-Free |  |  |  |  |  |  |
| Input Validation |  |  |  |  |  |  |
| Ease of use |  |  |  |  |  |  |
| Feedback |  | | | | | |
| Action Taken |  | | | | | |

1. **Implementation**
   1. **Screenshots**
      1. New Baseline

**A blue screen with white text

Description automatically generated**

Description

When the script is run, it asks the user to select between two options –

A – Collect a new baseline for the target files

B – Begin Monitoring Files with existing baseline

If option A is selected, the script collects a fresh new baseline for the target files in the directory.

* + 1. File Monitoring

**A computer screen with white text

Description automatically generated**

Description

When option B is selected, the system starts actively monitoring files in the directory and when changes occur, they are reported on the console. The system reports if a file has been changed, created, or deleted. The system keeps monitoring files until it has been stopped.

* + 1. Baseline.txt

**A screen shot of a computer

Description automatically generated**

Description

This is text file created by the system to store baseline of the target file. The text file contains the path of the file with the corresponding “SHA512” hash algorithm of the files. The baseline is updated every time the user selects option A to create a new baseline.

* + 1. Log.txt**A screenshot of a computer

       Description automatically generated**

Description

This is the text file created to keep track of all logs created by the system in the event of changes. These text files provide a deeper insight about the changes and if any discrepancies occur in the system.

* + 1. Report.txt

**A screenshot of a computer

Description automatically generated**

Description

This text file is created to store the report generated by the system of the latest monitoring activities. This file is also used to generate the report for the user which is then attached in the email alert notification during changes.

* + 1. **A screenshot of a computer

       Description automatically generated**Email Alerts

**A screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generated**

Description

These screenshots are of email alerts sent to user in the event of changes. The alert consists of the change made, the file path, the corresponding hash algorithm, and the report generated by the system. Such a feature makes it easier and faster to detect changes and can be easily passed to an expert if further study is required.

* 1. **Sample Codes**
     1. Modules

A close up of text

Description automatically generated

Description

The code begins by importing the PowerShellGet and PSReadLine modules. PowerShellGet is used to interact with PowerShell package repositories, and PSReadLine enhances the command-line editing experience.

* + 1. Entry Point

A screenshot of a computer program

Description automatically generated

Description

The code's entry point starts by prompting the user to choose between two options: "Collect new Baseline" (Option A) or "Begin monitoring files with saved Baseline" (Option B).

If Option A is chosen, it deletes the existing baseline file (if any) and calculates the file hashes for all files in the target folder. It then writes the file-hash pairs to the baseline file.

If Option B is chosen, it initiates the monitoring process by calling the Monitor-Files function with the baseline file path.

* + 1. Calculating File Hash

A computer code with text

Description automatically generated

Description

This function calculates the cryptographic hash of a file using the Get-FileHash cmdlet from PowerShell. It takes the file path as input and an optional parameter to specify the hash algorithm (SHA512 by default). The calculated hash is returned as output.

* + 1. Erase Baseline if already exists

A computer code with text

Description automatically generated

Description

This function is used to delete the baseline file (baseline.txt) if it already exists. It takes the baseline file path as an optional parameter, and if the file exists, it is removed using the Remove-Item cmdlet.

* + 1. Write log

A computer code with text

Description automatically generated

Description

This function is responsible for writing log messages to a log file (log.txt). It takes a log message and an optional parameter for specifying the log file path. The function adds a timestamp to the log message and appends it to the log file.

* + 1. Send Email Notification

A close-up of a computer screen

Description automatically generated

Description

This function handles sending email notifications to a specified recipient. It takes parameters for recipient email address, email subject, email body, and an optional attachment path for attaching files to the email. It uses the Send-MailMessage cmdlet to send the email via a configured SMTP server (Gmail in this case).

* + 1. Generate Report

A screenshot of a computer code

Description automatically generated

Description

This function generates a report containing file monitoring details. It retrieves the file hash data from the baseline dictionary, which stores file paths and their corresponding hashes. The function then generates a report content with a timestamp and formats the file hash data. Finally, it writes the report to a file specified by the report file path parameter.

* + 1. **A computer code with text

       Description automatically generated**Monitor Files

A screen shot of a computer code

Description automatically generated

Description

This function represents the core functionality of the File Integrity Monitor. It continuously monitors a target directory for file changes. The function takes parameters for the baseline file path and the polling interval (1 second by default).

It initializes a file hash dictionary to store file paths and hashes from the baseline file. The function loads file-hash pairs from the baseline file and populates the dictionary.

Using an infinite loop, the function periodically scans the target directory for new files or file modifications.

If a new file is detected (not present in the baseline dictionary), it sends an email notification and updates the dictionary.

If a file modification is detected (hash mismatch), it sends an email notification and updates the dictionary.

If a file in the baseline dictionary is no longer present in the directory, it sends an email notification and removes the entry from the dictionary. After each iteration, the function generates a report and sends it as an email attachment.

1. **System Validation** 
   1. **Unit Testing**
      1. Calculate File-Hash Function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Case | Expected Result | Actual Result | Status |
| 1.1 | Test if the function correctly calculates the hash of a given file and returns the expected hash value. | The function should correctly calculate the hash of the given file, and the calculated hash should match the expected hash value. | The function correctly calculated the hash of the given file, and the calculated hash matches the expected hash value. | PASS |
| 1.2 | Test if the function handles invalid file paths and throws an appropriate error or exception. | The function should throw an error or exception when an invalid file path is provided. | The function showed an error or exception on providing invalid file path. | PASS |

* + 1. Erase-Baseline-If-Already-Exists Function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Case | Expected Result | Actual Result | Status |
| 2.1 | Test if the function deletes the baseline file when it already exists and confirms its deletion. | The function should successfully delete the baseline file if it exists, and the baseline file should no longer exist after the function is called. | The function successfully deleted the baseline file as it already existed, and the baseline file no longer exists after the function is called. | PASS |
| 2.2 | Test if the function gracefully handles the scenario when the baseline file does not exist. | The function should handle the scenario properly when the baseline file does not exist. | The function handled the scenario gracefully when the baseline file did not exist, and no errors were encountered. | PASS |

* + 1. Write-Log Function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Case | Expected Result | Actual Result | Status |
| 3.1 | Test if the function correctly writes a log message to the specified log file and verifies the content in the log file. | The function should correctly write the log message to the specified log file, and the log file content should contain the expected log message. | The function correctly wrote the log message to the specified log file, and the log file content contains the expected log message. | PASS |
| 3.2 | Test if the function handles cases when the log file path is not accessible or there are permission issues. | The function should handle cases when the log file path is not accessible or there are permission issues. | The function handled cases when the log file path was not accessible or there were permission issues gracefully. | PASS |

* + 1. Send-Email Notification Function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Case | Expected Result | Actual Result | Status |
| 4.1 | Test if the function sends an email to the specified recipient with the correct subject, body, and attachment when provided. | The function should successfully send an email to the specified recipient with the correct subject, body, and attachment (if provided). | The function successfully sent an email to the specified recipient with the correct subject, body, and attachment (if provided). | PASS |
| 4.2 | Test if the function handles cases when there are issues with the SMTP server or incorrect login credentials. | The function should handle cases when there are issues with the SMTP server or incorrect login credentials. | The function handled cases when there were issues with the SMTP server or incorrect login credentials, and appropriate error handling was in place. | PASS |

* + 1. Generate-Report Function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Case | Expected Result | Actual Result | Status |
| 5.1 | Test if the function generates the report file with the correct content and verifies the report file's existence. | The function should generate the report file with the correct content, and the report file should exist after the function is called. | The function generated the report file with the correct content, and the report file exists after the function is called. | PASS |
| 5.2 | Test if the function handles scenarios when there are no file alerts and ensures no report file is generated. | The function should handle scenarios when there are no file alerts, and no report file should be generated in such cases. | The function handled scenarios when there were no file alerts, and no report file was generated in such cases. | PASS |

* + 1. Monitor-Files Function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Case | Expected Result | Actual Result | Status |
| 6.1 | Test if the function correctly monitors the target files and detects file changes | The function should accurately monitor the target files, detect file changes and deletions, and update the fileHashDictionary accordingly. | The function accurately monitored the target files, detected file changes and deletions, and updated the fileHashDictionary accordingly. | PASS |
| 6.2 | Test if the function updates the fileHashDictionary correctly during monitoring. | The function should update the fileHashDictionary without any errors during monitoring. | The function updated the dictionary without any errors during the monitoring process. | PASS |
| 6.3 | Test if the function handles different polling intervals and verifies the accuracy of file change detection. | The function should handle different polling intervals correctly and accurately detect file changes and deletions with varying intervals. | The function handled different polling intervals correctly and accurately detected file changes and deletions with varying intervals. | PASS |

* + 1. Main Function

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Test Case | Expected Result | Actual Result | Status |
| 7.1 | Test the user interaction and flow of the main menu options for "Collect New Baseline" and "Monitor Files with Saved Baseline." | The main menu should correctly guide the user through the "Collect New Baseline" and "Monitor Files with Saved Baseline" options. | The main menu correctly guided the user through the "Collect New Baseline" and "Monitor Files with Saved Baseline" options. | PASS |
| 7.2 | Test if the user input validation works as expected, such as handling invalid input options. | The main menu should handle invalid user input options gracefully and prompt the user for valid choices. | The main menu handled invalid user input options gracefully and prompted the user for valid choices without any issues. | PASS |

* 1. **User Acceptance Testing**
     1. Tester Name: Adrian Al Farizi

Occupation: Student

Date: 18/07/2023

Start Time: 4:00pm End Time: 4:30pm

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | RATING | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** |
| User Interface |  |  |  | x |  |  |
| Meeting Objectives |  |  |  |  | x |  |
| Security |  |  |  |  | x |  |
| Bug-Free |  |  |  | x |  |  |
| Input Validation |  |  |  |  | x |  |
| Ease of use |  |  |  |  |  | x |
| Feedback | The system's performance seems to degrade when monitoring a large number of files or using smaller polling intervals. It takes longer for the system to detect file changes, and in some cases, it may miss certain changes altogether. | | | | | |
| Action Taken | Optimize the file monitoring process to handle larger file sets more efficiently. This may involve using multi-threading or asynchronous processing to improve performance.  Evaluate the impact of the polling interval on system performance and adjust it accordingly. Consider using dynamic polling intervals that adapt based on the number of files or the system's load. | | | | | |

* + 1. Tester Name: Aqel Mohammed

Occupation: Student

Date: 18/07/2023

Start Time: 5:00pm End Time: 5:30pm

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | RATING | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** |
| User Interface |  |  |  |  | x |  |
| Meeting Objectives |  |  |  | x |  |  |
| Security |  |  |  |  | x |  |
| Bug-Free |  |  |  | x |  |  |
| Input Validation |  |  |  |  |  | x |
| Ease of use |  |  |  |  |  | x |
| Feedback | The email notifications are not being sent consistently. Sometimes, the system sends email alerts, but other times, it fails to do so. | | | | | |
| Action Taken | Investigate the email notification functionality to identify any potential issues or errors in the code.  Implement proper error handling and logging mechanisms to capture any failures in sending email notifications. | | | | | |

* + 1. Tester Name: Johnes Khar

Occupation: Student

Date: 19/07/2023

Start Time: 4:00pm End Time: 4:30pm

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | RATING | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** |
| User Interface |  |  |  | x |  |  |
| Meeting Objectives |  |  |  | x |  |  |
| Security |  |  |  |  | x |  |
| Bug-Free |  |  |  |  | x |  |
| Input Validation |  |  |  |  | x |  |
| Ease of use |  |  |  |  |  | x |
| Feedback | The system generates a lot of log entries, making it challenging to identify critical events or issues from the log file. There should be a way to prioritize log messages based on their severity. | | | | | |
| Action Taken | Enhance the logging mechanism to include different log levels (e.g., INFO, WARNING, ERROR) for various messages.  Implement a filtering mechanism that allows users to specify the log level they want to view, enabling them to focus on critical events or errors easily. | | | | | |

* 1. **Summary**

From the unit testing and user acceptance testing, various improvements have been made for the system improving the overall user experience of the File Integrity Monitor.

In all test cases, the functions behaved as expected, and the test results confirmed the correct functioning of each function under different scenarios. The absence of errors or exceptions and the successful completion of the intended tasks in each test case indicate that the implementation of the functions is working correctly.

The feedback from the user acceptance testing indicates that the system is functional and meets the basic requirements. However, there are opportunities to improve the system's reliability, performance, and log management. By addressing the identified issues and implementing the suggested actions, we can enhance the system's overall usability and user experience.

1. **Conclusion and Reflections**
   1. **Critical Evaluation**

The File Integrity Monitor demonstrates several valuable features, such as baseline creation, real-time monitoring, alerting, reporting, and automated maintenance. Its capability to calculate and store cryptographic hashes for files enables the detection of unauthorized modifications, ensuring data integrity and security. The email notification system allows for timely alerts, keeping administrators informed of any suspicious activities. Additionally, the system's reporting functionality aids in post-incident analysis and provides a comprehensive overview of file changes.

However, the system has some areas that require critical evaluation. First, the current implementation lacks user authentication and access control, making it vulnerable to unauthorized access. Implementing user authentication and role-based access control would enhance the system's security and ensure that only authorized personnel can access and modify the monitoring configurations.

Second, the system currently uses email for notifications, which might not be the most secure method for transmitting sensitive information. Utilizing encrypted communication channels, such as Transport Layer Security (TLS), for email notifications would bolster data protection during transmission.

Third, the system's reliance on polling intervals for monitoring may result in delayed detection of file changes. Implementing a real-time monitoring mechanism, such as leveraging file system event notifications, would lead to more immediate and accurate detection of modifications.

The system has some limitations that warrant careful consideration. The absence of user authentication and access control poses a significant security risk, as unauthorized users could potentially manipulate monitoring settings or access sensitive data. Implementing robust user authentication mechanisms and role-based access control would mitigate these security concerns and ensure that only authorized personnel have access to the system.

Furthermore, while email notifications are convenient, they may not be the most secure means of transmitting sensitive information. Incorporating encrypted communication channels, such as TLS or S/MIME, would enhance the confidentiality and integrity of email notifications, reducing the risk of data interception or tampering during transmission.

Moreover, the lack of a graphical user interface (GUI) restricts the system's accessibility for non-technical users. Providing an intuitive GUI alongside the existing CLI would make the system more user-friendly and easier to configure and operate.

Lastly, while the system currently supports SHA512 as the default hash algorithm, it would be beneficial to include support for other industry-standard cryptographic algorithms to accommodate specific security requirements.

The File Integrity Monitor offers significant utility to its target users, which primarily includes system administrators, IT security professionals, and organizations concerned with data integrity and security. For system administrators, the system provides a reliable and automated way to monitor critical files and directories for unauthorized changes, ensuring the integrity of essential system files and configurations. It simplifies the process of creating and maintaining baselines, allowing administrators to establish a trusted state and detect any deviations efficiently. The real-time monitoring and email notification system keep administrators promptly informed of any potential security breaches, enabling them to take immediate action to mitigate risks and investigate suspicious activities. Furthermore, the system's reporting feature assists in compliance audits and post-incident analysis, facilitating evidence gathering and ensuring regulatory adherence. Overall, the File Integrity Monitor streamlines the task of monitoring file integrity, enhances security controls, and empowers users with actionable insights to protect their systems and sensitive data effectively.

* 1. **Conclusion**

In conclusion, a file integrity monitor using hashing algorithms is an effective way to ensure the security and integrity of files on a computer system. By using cryptographic hash functions such as SHA-256 or MD5, the monitor can generate a unique digital signature for each file, which can then be used to verify the file's integrity at a later time.

Throughout the implementation, several key features were incorporated to enhance the system's functionality and user experience. The ability to calculate file hashes using different algorithms provides flexibility and security. The email notification feature ensures that users are promptly informed about file modifications or deletions, facilitating quick responses to potential security threats. The logging mechanism offers detailed event tracking, aiding in system debugging and monitoring.

During user acceptance testing, the system performed well overall, meeting the intended requirements. However, feedback highlighted areas for improvement, including enhancing email notification reliability, optimizing system performance for larger file sets, and implementing a more effective log management approach.

The successful implementation of unit tests helped ensure the correctness of individual components and functionalities, while user acceptance testing provided valuable insights from real users, leading to refinements and enhancements.

With the system now in place, users can confidently monitor the integrity of their critical files and receive timely alerts in case of any suspicious or unauthorized changes. The system's flexibility, robustness, and comprehensive reporting capabilities make it a valuable tool in maintaining the security and integrity of file systems across various use cases.

As technology and security requirements evolve, future updates and enhancements can be made to further improve the system's performance, expand its capabilities, and cater to additional user needs. Overall, the File Integrity Monitoring system stands as a valuable asset for organizations seeking to protect their data and maintain the integrity of their files.

Working on this project was not an easy task as there aren’t many articles or publications on the subject forcing me to rely on web articles or publications to complete the research. Time has also played a crucial factor in completing the report as there were other assignments that were due around the same time.

For the survey, responses from only 32 participants have been collected due to time restrictions.

Even though the system seems simple with this iteration, future versions will have more features to offer.

**References**

Udzir, N. I. (2011, January). *Towards a Dynamic File Integrity Monitor through a Security Classification*. Research Gate. Retrieved December 13, 2022, from https://www.researchgate.net/publication/230771292\_Towards\_a\_Dynamic\_File\_Integrity\_Monitor\_through\_a\_Security\_Classification

Ogden, J. von. (n.d.). *5 Security Objectives That Require File Integrity Monitoring*. https://www.cimcor.com/blog/5-security-objectives-that-require-file-integrity-monitoring

Melnick, J. *File Integrity Monitoring: Definition, Benefits and Key Features*. (2020, April 14). https://blog.netwrix.com/2020/04/14/file-integrity-monitoring/

Kedgley, M. (n.d.). File Integrity Monitoring - The Last Line of Defense in the PCI Data Security Standard. *FIM*. https://www.newnettechnologies.com/nnt-whitepaper-file-integrity-monitoring.pdf

*File Integrity Monitoring Software - FIM Integrity Checker | SolarWinds*. (n.d.). https://www.solarwinds.com/security-event-manager/use-cases/file-integrity-monitoring-software

CrowdStrike. (2022, July 12). *What is File Integrity Monitoring (FIM)? | CrowdStrike*. crowdstrike.com. https://www.crowdstrike.com/cybersecurity-101/file-integrity-monitoring/#:~:text=FIM%20tools%20rely%20on%20two,proactive%20or%20rules%2Dbased%20monitoring.

Von Ogden, J. (n.d.). *4 Reasons Why File Integrity Monitoring is Important for Enterprises*. https://www.cimcor.com/blog/4-reasons-file-integrity-monitoring-important-enterprises

Team, R. (2022, June 10). *The importance of file integrity monitoring in cyber threat detection - Redscan*. Redscan. https://www.redscan.com/news/the-importance-of-file-integrity-monitoring-in-cyber-threat-detection/

Abdullah, Z. H., & Mahmod, R. (2011). File Integrity Monitor Scheduling Based on File Security Level Classification. *FIM*. https://www.researchgate.net/publication/220868566\_File\_Integrity\_Monitor\_Scheduling\_Based\_on\_File\_Security\_Level\_Classification

Von Ogden, J. (n.d.-b). *7 Regulations Requiring File Integrity Monitoring for Compliance*. https://www.cimcor.com/blog/7-regulations-requiring-file-integrity-monitoring-for-compliance

*File Integrity Monitoring Market Analysis - Industry Report - Trends, Size & Share*. (n.d.). https://www.mordorintelligence.com/industry-reports/file-integrity-monitoring-market#

*Integrating FIM and SIEM Solutions*. (2021, November 3). https://blog.netwrix.com/2021/11/03/which-file-integrity-monitoring-technology-is-best-for-fim/

Singh, A. P., & Singh, M. D. (2014). Analysis of Host-Based and Network-Based Intrusion Detection System. *-*. https://www.researchgate.net/publication/276230480\_Analysis\_of\_Host-Based\_and\_Network-Based\_Intrusion\_Detection\_System

*File Integrity Monitor (FIM)*. (n.d.). https://docs.logrhythm.com/docs/enterprise/client-console-administrator-guide/endpoint-monitoring/file-integrity-monitor-fim

*File Integrity Monitoring and SIEM - Logsign*. (n.d.). https://www.logsign.com/blog/file-integrity-monitoring-and-siem/

*Endpoint Detection and Response (EDR) | AT&T Cybersecurity*. (2023, March 15). https://cybersecurity.att.com/solutions/endpoint-detection-and-response

Z. (2022, September 8). *File Integrity Monitor with Powershell - Sysjolt*. Sysjolt. https://www.sysjolt.com/2020/file-integrity-monitor-with-powershell/

Von Ogden, J. (n.d.-d). *How Does File Integrity Monitoring Work?* https://www.cimcor.com/blog/how-does-file-integrity-monitoring-work

**Appendices**

Turnitin Report

A screen shot of a receipt

Description automatically generated

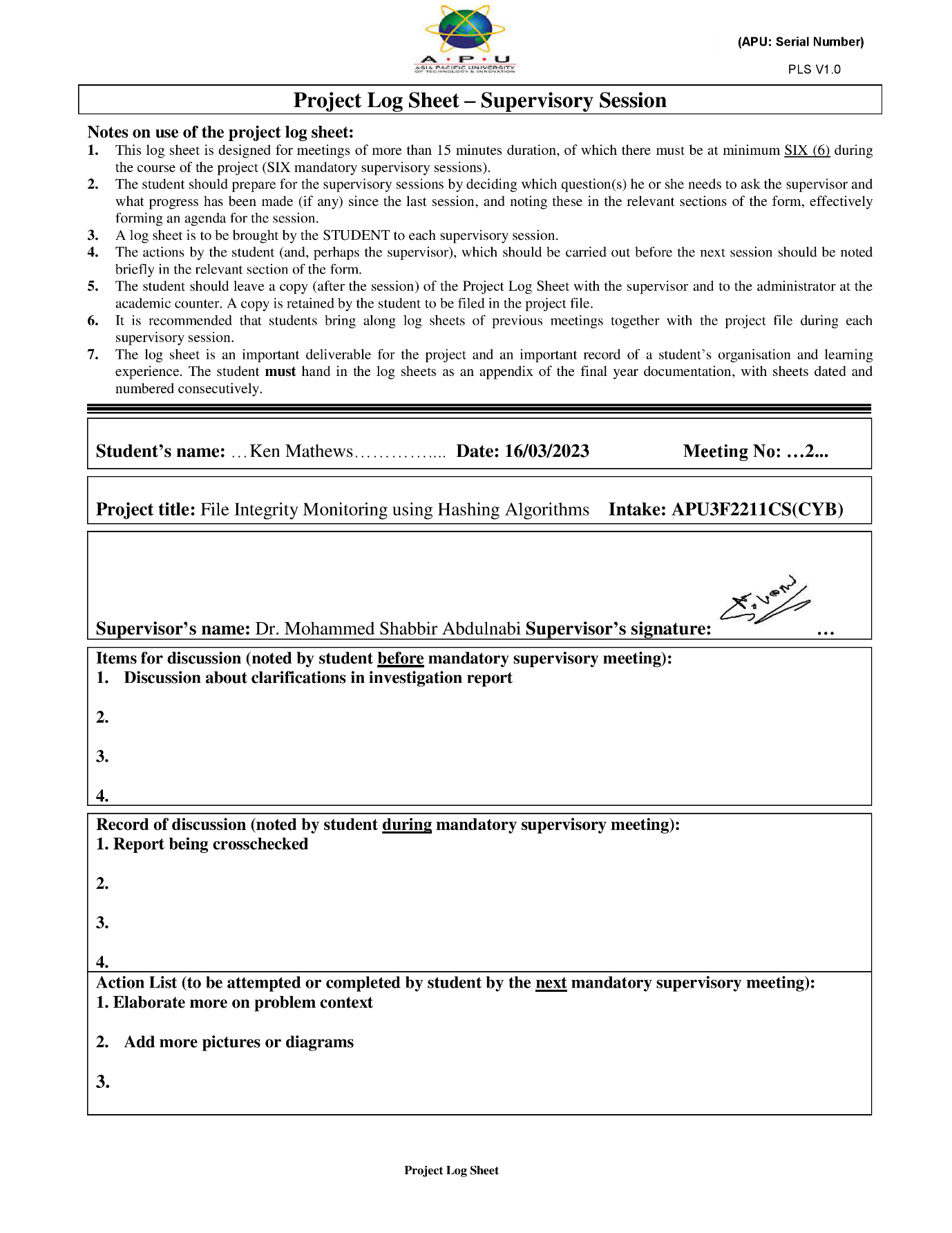
A screenshot of a computer screen

Description automatically generatedFYP Poster

Meeting Log Sheets

A black background with a white rectangle

Description automatically generated



A black background with a white rectangular object

Description automatically generated

A black background with a white rectangular object

Description automatically generated

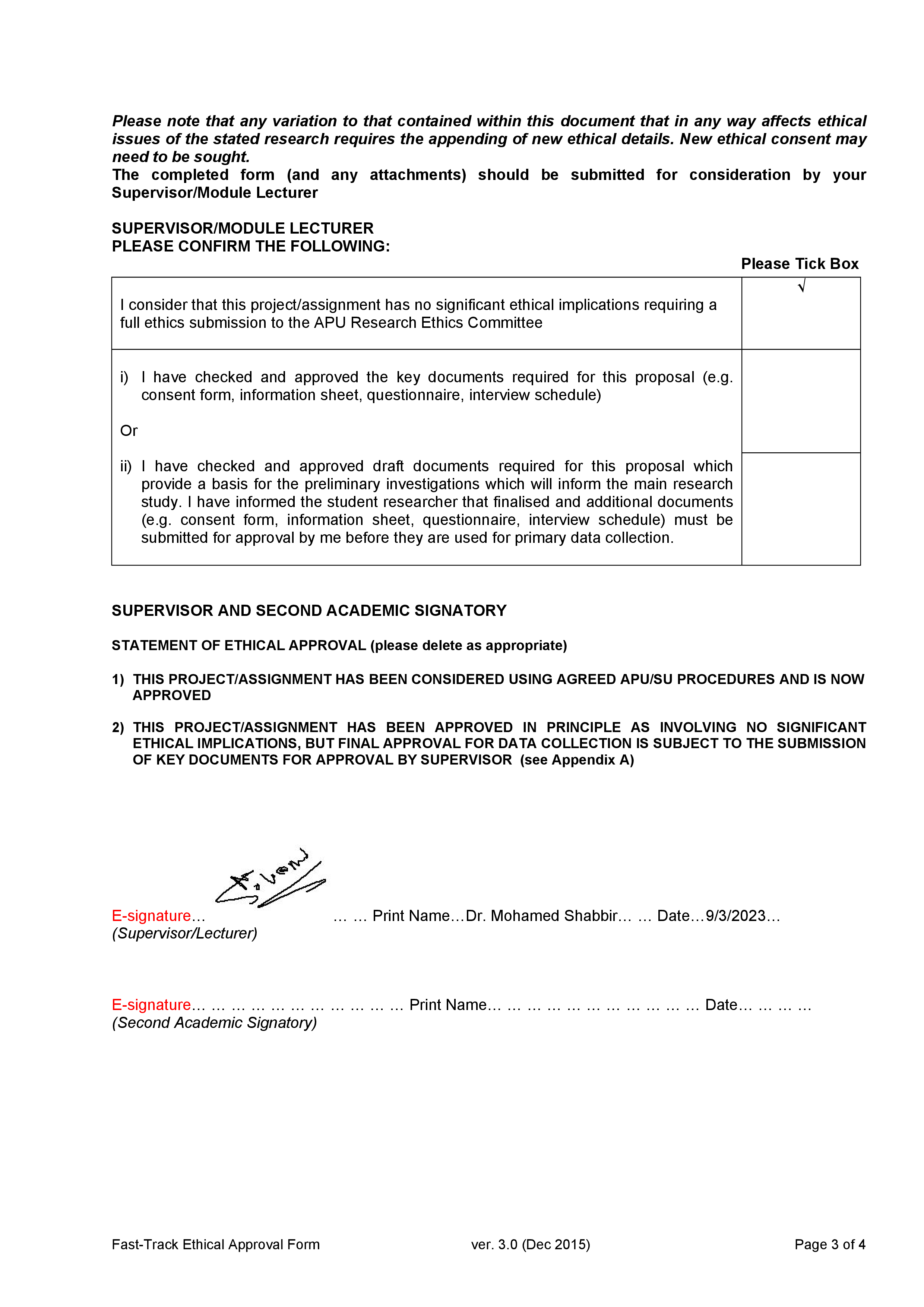
Ethics Form

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated



A black screen with a white rectangle with red dots

Description automatically generated

Chart, bubble chart

Description automatically generatedGant Charts

A graph of a project

Description automatically generated*Fig. 1 Gantt chart of project plan*

*Fig. 2 Gantt chart of project implementation*