

**Sri Lanka Institute of Information Technology**

**Final Project Report**

**ISP Project Report**

Information Security Project 2023

Submitted by:

|  |  |
| --- | --- |
| IT Number | Name |
| IT21068300 | W.P.Dananjana |

Date of submission

2023-11-05

Abstract

In the growing threat landscape protection of information systems is an ever-growing need, added to the fact most of the critical business operations run on Linux environments highlights the need to have robust protective mechanisms on Linux systems as a whole. One of the most often overlooked aspects in system security tends to be monitoring the integrity of system files. Hence, in this report a product name “FileGuard” was developed as simple, yet powerful file integrity monitoring system designed specifically for Linux systems. Its significance lies in preserving the integrity and security of critical system files and directories. Utilizing the power of python and developing on already present monitoring tools such as ‘Tripwire’, it enforces rigorous file protection by monitoring, detecting, and generating reports on any alterations within the system. In addition to that it provides the users the opportunity to customize the policies to ensure the best usage of their time and resources, by ensuring compliance with defined policies and maintaining the security and integrity of Linux systems. Therefore, “FileGuard” contributes significantly to the overall security and stability of a Linux-based environment.

Link to the Github Repository: <https://github.com/IT21068300/ISP_2023>

Acknowledgement

I would like to express my heartfelt gratitude to all those who have contributed to the development of this product. First and foremost, I am deeply thankful to the lecturer in charge of the ISP module, for their invaluable guidance, support, and mentorship throughout this project. Their expertise and dedication have been instrumental in shaping the direction of this work.

Furthermore, I want to acknowledge the support of my institution, SLIIT, for providing the necessary resources and facilities that made this project possible.

Declaration

I declare that this project report or part of it was not a copy of a document done by any organization, university any other institute or a previous student project group at SLIIT and was not copied from the Internet or other sources.

|  |  |  |
| --- | --- | --- |
| **Reg. No** | **Name** | **Signature** |
| IT21068300 | W.P.Dananjana |  |

Table of Contents

Abstract ii

Acknowledgement iii

Declaration iv

Table of Contents 1

List of Figures 2

1. Introduction 3

1.1 Problem Statement 3

1.2 Product Scope 3

1.3 Product introduction 4

1.4 Information security related Scope of FileGuard 4

2. Methodology 7

2.1 Requirements 7

2.2 Development 9

2.3 Implementation 22

3. Evaluation 24

3.1 Assessment of the Project results 24

3.2 Lessons Learned 24

3.3 Future Work 25

4. Conclusion 26

5. References 27

List of Figures

[Figure 1 9](#_Toc150085112)

[Figure 2 9](#_Toc150085113)

[Figure 3 10](#_Toc150085114)

[Figure 4 12](#_Toc150085115)

[Figure 5 13](#_Toc150085116)

[Figure 6 14](#_Toc150085117)

[Figure 7 15](#_Toc150085118)

[Figure 8 16](#_Toc150085119)

[Figure 9 16](#_Toc150085120)

[Figure 10 17](#_Toc150085121)

[Figure 11 17](#_Toc150085122)

[Figure 12 18](#_Toc150085123)

[Figure 13 18](#_Toc150085124)

[Figure 14 19](#_Toc150085125)

[Figure 15 20](#_Toc150085126)

[Figure 16 21](#_Toc150085127)

# Introduction

## Problem Statement

While working with Linux systems, it was discovered that they do not have an efficient mechanism to monitor, detect or report unauthorized changes to important system files. A problem which can pause a threat to maintaining the security and integrity of data and configurations in Linux systems.

As a solution for those problems, "FileGuard" was created to remedy this issue by utilizing a Python-based file protection mechanism. It will continually watch system files, records their attributes in a seperate database, and detects changes in critical files and generate reports on those changes. Once any changes are detected then those changes are documented into a report. Also, the users are given the additional option of setting up email alerts, to notify of any unauthorized changes.

## Product Scope

## "FileGuard" provides a solution to enhance the security and integrity of Linux systems. Its primary purpose is to protect system files, from unauthorized access, changes and misconfigurations.. "FileGuard" is also a solution with little needs and little user experience and expertise, which is a plus. As shown below, the product's scope covers the following features:

* Continuous monitoring of system files
* Classification of changes to system files
* Database management
* Report generation
* Configurability and customizability

## Product introduction

In this digital age, where data safety has become paramount, the need for strict file security procedures has never been greater. "FileGuard" is a powerful and practical solution for this problem which gives the users a tool to easily secure files on Linux Systems. "FileGuard" was designed with all of this in mind to provides a powerful security tool for corporations, organizations, or normal users alike.

Given below are the key features of the product

1. **Real-time File Monitoring**

FileGuard continuously tracks file attributes and content changes to detect unauthorized modifications.

1. **Classification and Alerts**

It categorizes file changes as additions, modifications, or deletions, sending instant alerts to administrators.

1. **Comprehensive Reporting**

Generate detailed reports to analyze file activity, facilitating quick response to potential threats.

1. **Configurable Policies**

Tailor monitoring to specific needs using a customizable policy configuration file.

1. **Database Management**

A powerful database system stores file information for reference and analysis.

## Information security related Scope of FileGuard

Improving the overall security posture of Linux systems is very important, especially in a scenario when critical operations related to the business are running on those systems. While “FileGuard” does address a wide range of information security concerns, the most prominent is the security and preserving of the integrity of these systems. More unique features of “FileGuard” are discussed below in a detailed description of its information security scope.

* **File Integrity**

File Integrity is ensuring that files are unchanged and correct and is critical for sustaining information security. Monitoring file modifications to detect and rectify errors should be part of file management processes as much as proper backup of data files is critical for file safety. "FileGuard" monitors file integrity and tracks any changes made to them using a mix of "inode" and "MD5 hash" checks. Using MD5 hashes in particular incorporates cryptographic verification of the integrity of file contents [1] [4].

* **Unauthorized Access Detection**

Unauthorized access, to files can result in data breaches the utilization of confidential information and the potential compromise of system resources. Detecting and preventing unauthorized access attempts are very important when maintaining the integrity and confidentiality of critical data. “FileGuard” continuous monitoring mechanism detects unauthorized access. Rather than relying solely on post-event security logs,"FileGuard" actively monitors files, attributes and content. This proactive approach ensures that any suspicious activities are promptly addressed, often even before conventional security logs are generated [1].

* **Configuration Security**

A important part of information security is secure system setup and any modifications to configuration files might cause system vulnerabilities or misconfigurations, potentially allowing unauthorized access or causing system instability. The configuration file in this tool allows system administrators to define crucial file attributes for monitoring, therefore improving configuration security by recording changes to specified attributes that are most important to the system's stability and security.

* **Data Confidentiality**

Data confidentiality is an important aspect of information security. Any file breach or change might jeopardize critical data. Maintaining stakeholder confidence and regulatory compliance need data secrecy. Because of its ability to identify and warn administrators to illegal file alterations, the technology indirectly helps to data secrecy by allowing data breaches to be resolved quickly, decreasing the impact on data confidentiality[1].

* **Reporting and Alerting**

Timely incident reporting and alerting mechanisms allow system administrators to respond quickly to security breaches, lowering the possible damage. Also the feature that enables to receive email notifications to System Administrators about unauthorized modifications allows speedy incident response, providing a unique advantage in the information security sector.

**1.5 User Demographic**

FileGuard caters to any industry and sectors, including finance, healthcare, government, and small businesses, who wants to monitor and safeguard their critical systems which are running Linux operating systems. FileGuard is designed for those who prioritize information security and need a dependable solution to protect their critical systems.

# Methodology

## Requirements

**Environmental Requirements**

* **Programming Language:** This tool is developed using Python 3. Therefore, the development environment should support Python 3 or above. If Python 3 is already installed, it must be the updated version.
* **Database Management System:** The tool relies on SQLite3 as the database management system for storing file information, MD5 hashes, and configuration settings. Therefore, the development environment should support and have SQLite3 already installed. If SQLite3 is already installed, it must be the updated version.
* **Policy Management:** The tool relies on Tripwire security tool for policy management and customization according to admin needs. Therefore, the development environment should support and have Tripwire already installed. If Tripwire is already installed, it must be the updated version [2] [5].

**Functional Requirements**

* **Continuous Monitoring:** The first functional requirement of the tool is to continuously monitor file attributes, including but not limited to permissions, ownership, timestamps, and file size of the system files.
* **MD5 Calculation:** The second requirement is that the tool should be ablet to calculate and store MD5 hashes of each monitored file during the initialization phase. Also it should be able to get the MD5 hashes of each monitored file in later states, to compare with the initial state.
* **Change Categorization:** The third functional requirement of the tool is that it should be capable of categorizing changes in the monitored system files into three classes, which are as follows:
  + **Modified Files:** Detect any changes made to the contents or attributes of existing files.
  + **Deleted Files:** Identify files that have been deleted or are no longer present or visible in the system.
  + **Newly Added Files:** Identify and report files that have been added since the last or initial check.
* **Configuration Management:** The fourth functional requirement of the tool is that it should allow for easy configuration updates by the system admin/user, such as the choice of files to monitor and the email settings for alerting.
* **Alerting Mechanism:** The fifth functional requirement of the tool is the seamless email alerting mechanism. It should send an email notification to the designated system administrators or users when unauthorized file changes are detected along with a full report with details of the incident.

**Non-Functional Requirements**

* **Scalability:** The number of system files that must be observed rises with time, and the tool's workload varies based on the type of system and the functions it performs. Furthermore, the tool should be able to evolve in response to business or user needs. As a result, in order to handle a rising number of monitored files and adapt to changing workloads, the tool should be scalable.
* **Reliability:** The security of critical systems is of high importance, any errors or sudden breakdowns might have catastrophic consequences on a business/users, therefore the tool should be highly reliable, with minimal downtime and good error handling.
* **Performance:** Not only the security the performance of the critical systems must be taken into consideration, which cannot be too compromised for the sake of security only, therefore the tool should have efficient performance, with low overhead on host system resources during continuous monitoring.

## Development

The development of the File Protection System followed a structured and iterative methodology, from requirements analysis through design and implementation.

* **Environment Setup**

The development process starts by first making sure that the building environment has Python 3 and relevant libraries, SQLite and Tripwire already installed. If so start by updating them to the latest version, if not install Python 3 and SQLite3 and Tripwire on the system. Note that the following commands were used to install them on Ubuntu.

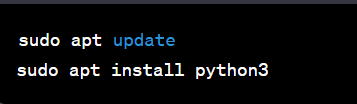
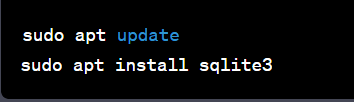


Figure 1

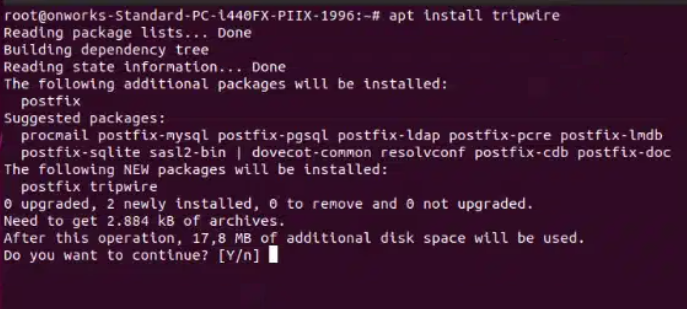


Figure 2

* **Configuration**

Configuring project paths, file paths, database file paths, and any other relevant settings are done by the “Config.py” script. It defines and manages the paths and directories used by the product. For example, the project directory, data storage directory, initial database path, configuration file storage directory, and policy configuration directory.

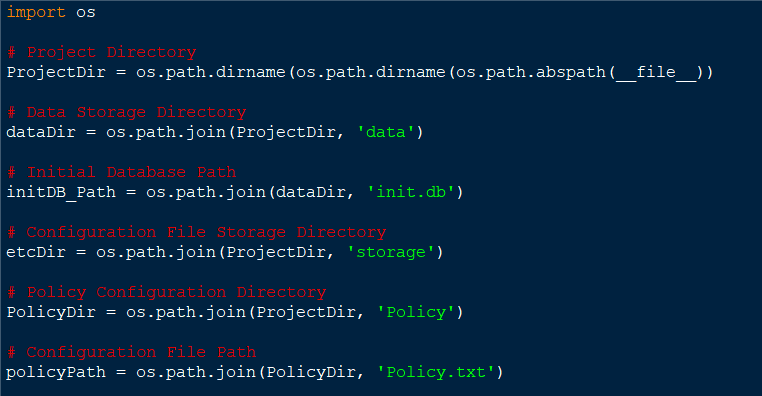
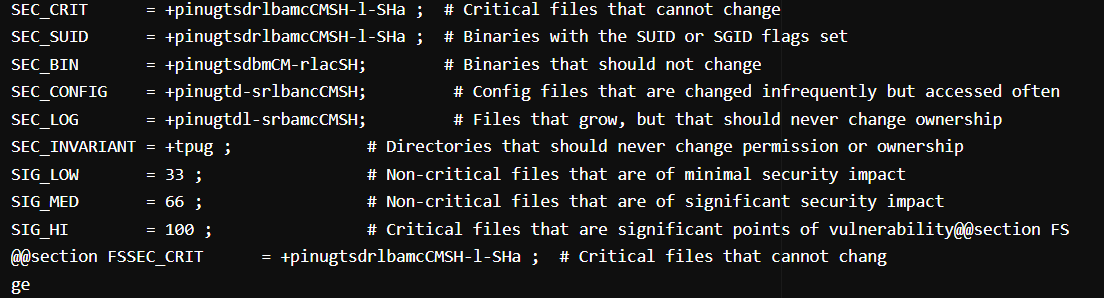
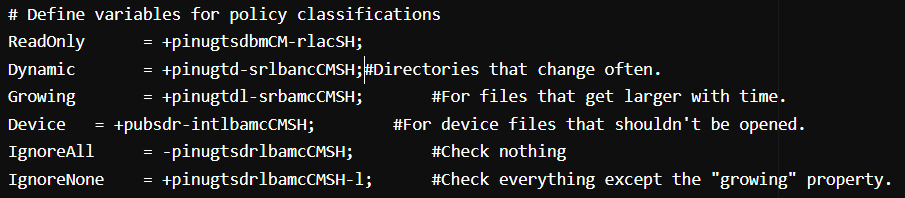


Figure 3

* **Policy Definition and Configuration**

The product follows a set of customizable user defined policies to determine which files and directories should be monitored and what kind of changes are relevant using the security and data integrity tool ‘Tripwire’. These policies are stored in the policy.txt file, which includes the following ;

Specifies different policy classifications and their associated rules are as given below. These classifications include the following, “ReadOnly”, “Dynamic”, “Growing”, “Device”, “IgnoreAll”, “IgnoreNone”, “SEC\_CRIT”, “SEC\_SUID”, “SEC\_BIN, SEC\_CONFIG”, “SEC\_LOG, SEC\_INVARIANT”, “SIG\_LOW”, “SIG\_MED”, and “SIG\_HI”.



Protecting vital system files, monitoring system applications and binaries, protecting control files relevant to security, and guaranteeing the stability of the system boot files are the key areas of concern here. The rules are assigned some severity levels in order to prioritize warnings and handle major security risks.

Once configured, the "twadmin" command was used to generate the Tripwire policy. Then when the policy has been properly built, the "twprint" command was used to show the compiled policy. The "twprint" command should also display the exact rules, paths, and parameters described in the custom Tripwire policy, showing information about file integrity monitoring [3].

* **Database Initialization**

In this product SQLite is used to establish an initial database to store “inode” and “MD5” hash information for files in the system. Database schema and creating tables for storing file information, MD5 hashes, and configuration settings are done using the “Init.py” script.

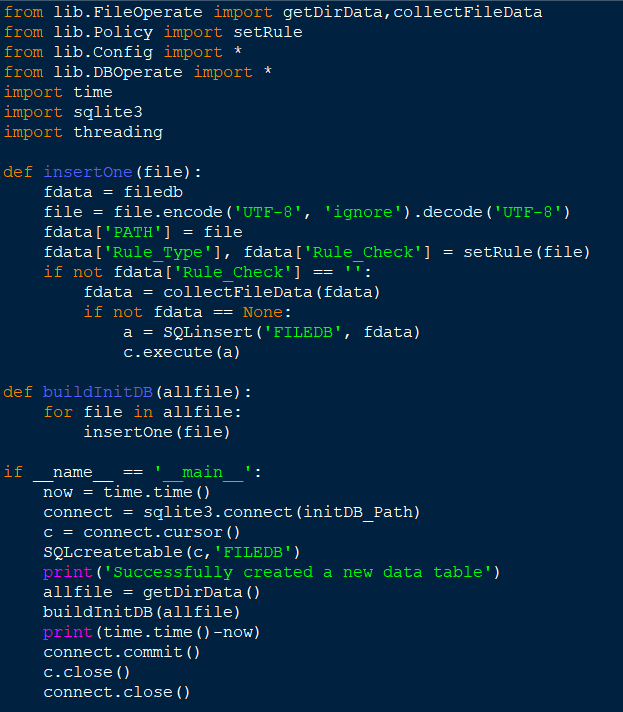


Figure 4

* **Database Operations**

The functions in “DBOperate.py” script are used to manage the database used to store file information.

It gets started by establishing the 'filedb' dictionary, which defines the structure of file records and includes the following attributes: file location, attributes, MD5 hash, rule kinds, rule checks, and record types.

The dictionary'recordmap' is used to convert single-character codes into human-readable file record types for better understanding for the users.

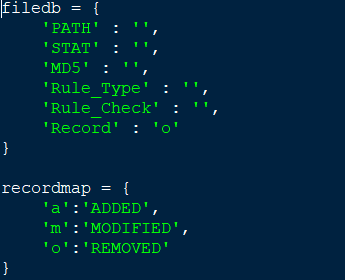


Figure 5

* + 'SQLcreatetable' function : Create new database tables or delete existing ones.
  + 'SQLupdate' function: Create SQL UPDATE statements to modify existing records, 'SQLinsert' function: Generate SQL INSERT statements to insert new data.
  + 'queryFileData' function: Use file path to search the database for file data.

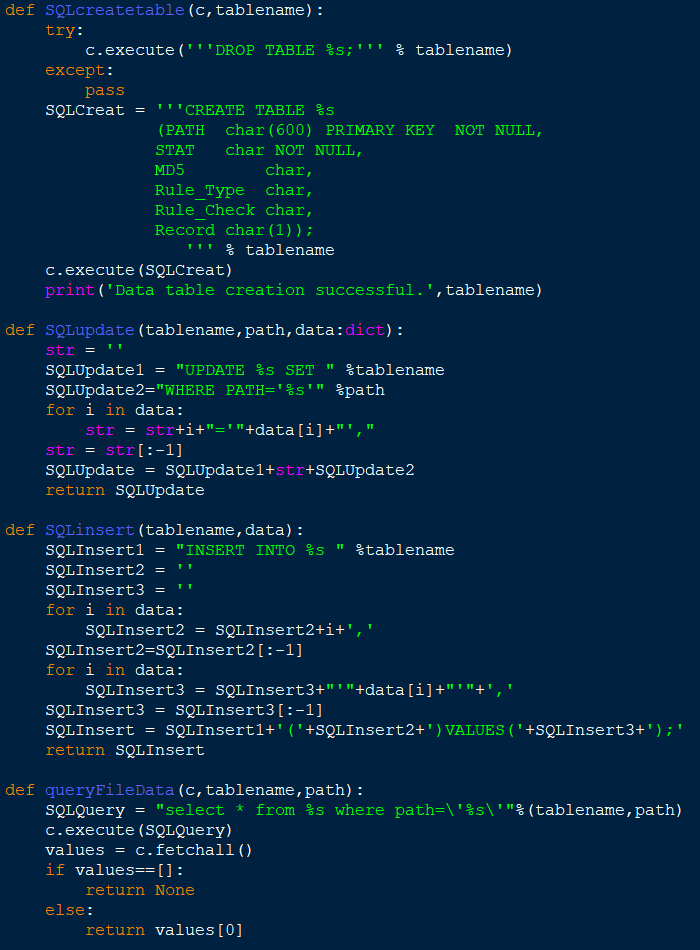


Figure 6

* **File Attribute Collection**

The “FileOperate.py” fulfils the task of collecting data and attributes related to files and directories in the system. It creates a structured repository of file and directory information, which is utilized with the database operations in the “DBOperate.py” script to monitor and record changes within the system.

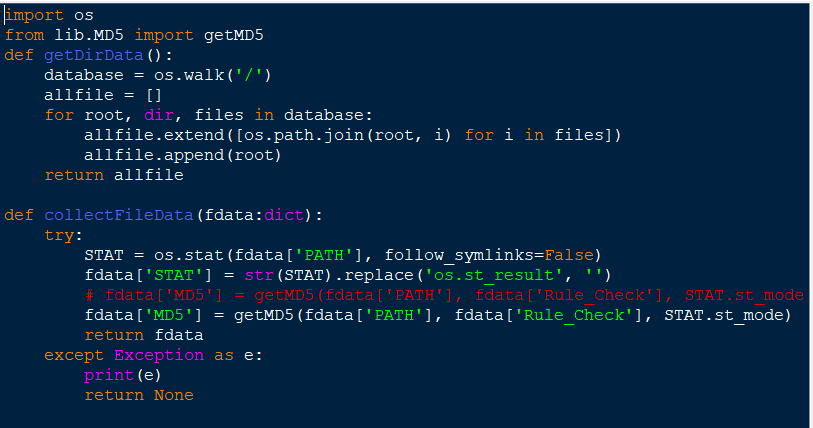


Figure 7

The duties of the other functions,

* + 'getDirData()' function: Utilize the 'os.walk()' method and search the file system and capture files and directories details and constructs and returns a list of paths.
  + 'collectFileData()' function: Obtain specific data for an individual file (attributes, MD5 hash, ect..) based on the information provided in the input dictionary. Stored them within the dictionary and return them.
  + Error handling in this step is done during the data collection process by printing an error message while returning no results.
* **Obtain Statistical File Information**

The main function of “MyStat.py” script is used to capture the statistical information needed for the monitoring process. It is done through the two classes, “Mystat” and “MystatFormated”. The two classes process and format file statistics information obtained from the stat command output. The “Mystat” class parses the input string and extracts a range of statistical attributes (mode, “inode”, timestamps, size, UID, GID, device). The file's type in this is determined by its mode.

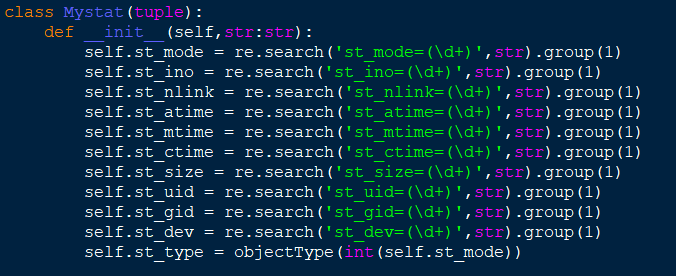


Figure 8

But the “MystatFormated” class extends this functionality of “Mystat” by enhancing the human interpretation of the extracted data by formatting the mode, converting timestamps into user-friendly date and time formats, retrieving user and group names, and translating device numbers into device names.

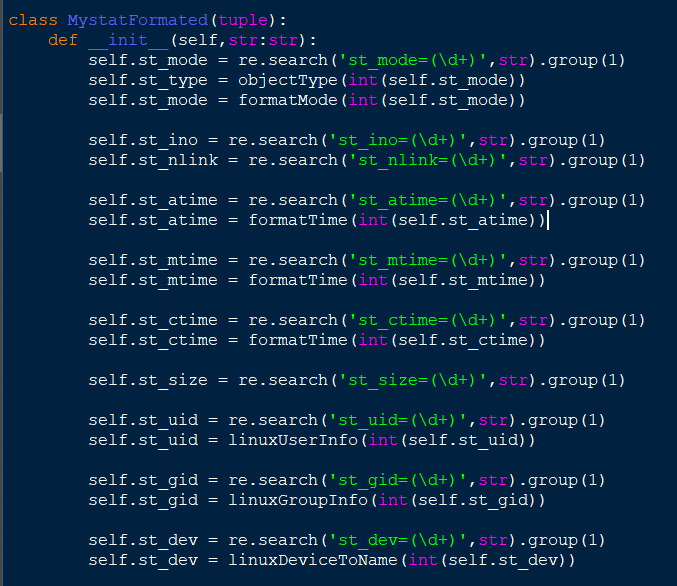
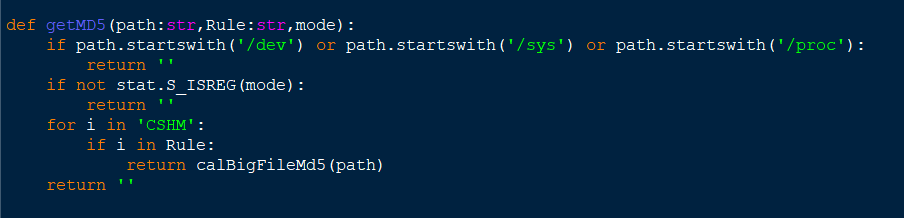


Figure 9

* **MD5 Calculation**

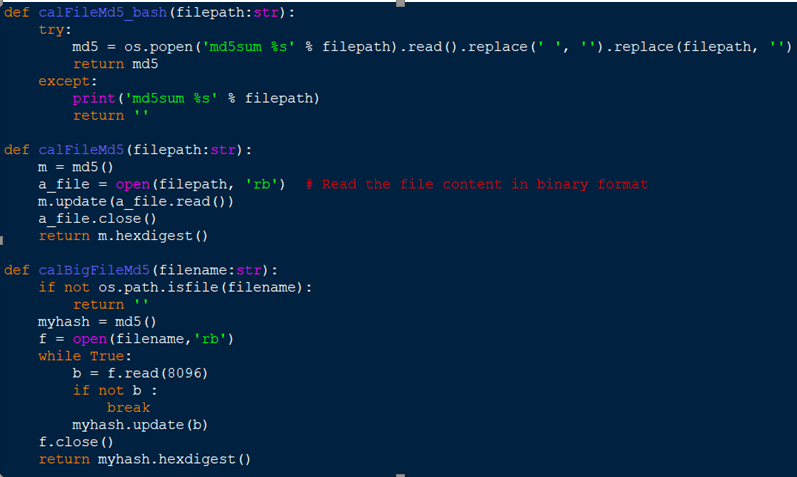
"MD5.py" is a set of functions that compute MD5 hashes for files based on various contexts and rules. The "getMD5" function is the primary way for computing the MD5 hash based on a given rule and the mode of the file. It checks to see if specified directories are prohibited, then uses the proper hash calculation algorithm to ensure the file is a normal file.



Figure

The other functions stand in for other cases when the getMD5 function fails. Their functionalities are given below

* + "calFileMd5\_bash" function: Calculate the MD5 hash of a specified file using the Bash md5sum command.
  + "calFileMd5" function: Read the binary content of a file and calculate the hash using the hashlib library.
  + "calBigFileMd5" function: Break big files into smaller parts to prevent memory issues.

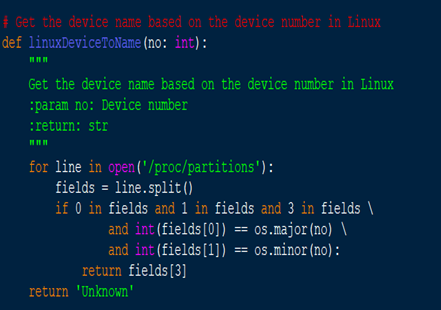


Figure

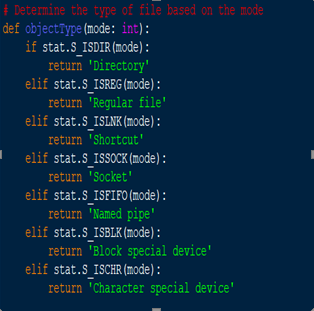
* **Formatting Obtained Attributes**

The FormatInfo.py script contain the utility functions for enhancing file-related data reading and understanding by users/humans. The workings of the functions are as follows,

* + 'objectType' function: determines the file type, depending on the mode.
  + 'linuxDeviceToName' function: Identify device names from device numbers. 'formatTime' function: Translate timestamps to readable date and time formats
  + 'formatMode' function: Transform file modes to readable file permission strings.

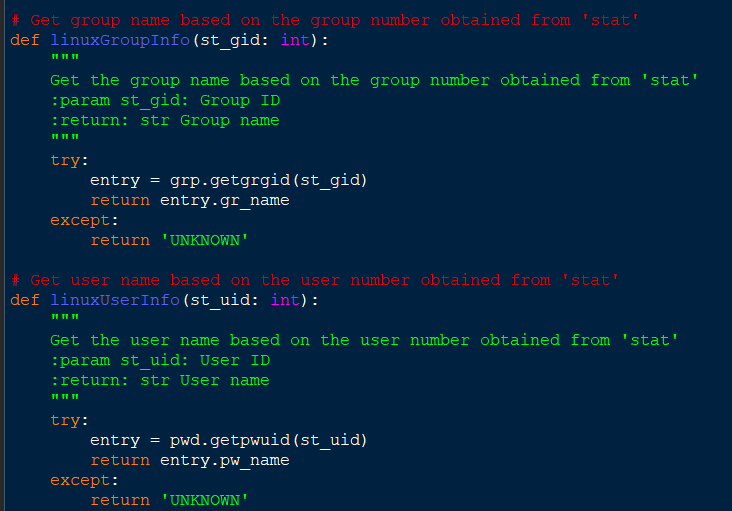


Figure



Figure

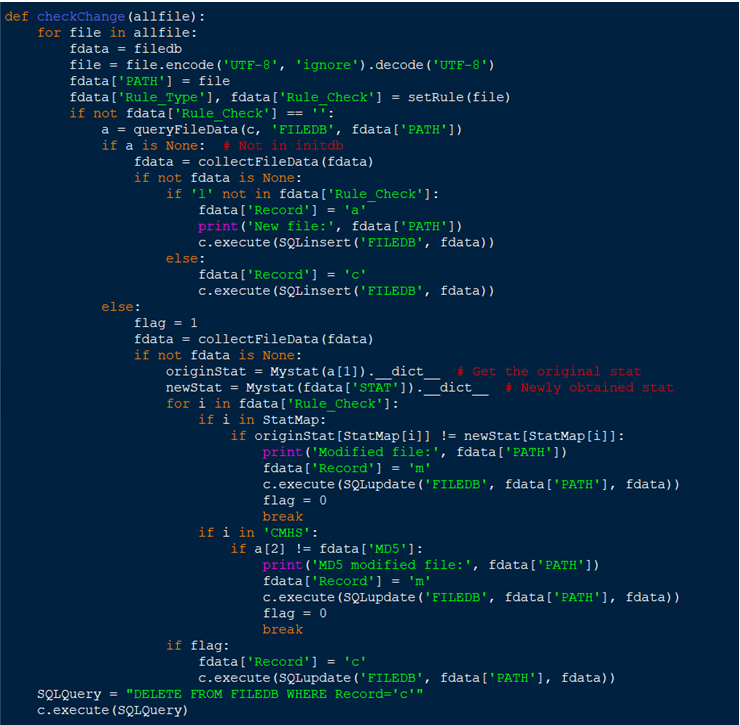
* + 'linuxGroupInfo' function: Transform group IDs into corresponding group names
  + 'linuxUserInfo' function: Associate user IDs with user names and convert integer-based attributes obtained from the "stat" function into human-readable strings.



Figure

* **Monitoring**  
  The component in the "check.py” script are responsible for monitoring and maintaining file integrity based on policies defined in the custom policy document. these tasks are done by the following key functions and processes,
  + "checkChange" function: Scrutinize the list of file and directory paths obtained from the system, employ policy rules, query the database for existing data, and categorizes files as, new ('a'), modified ('m'), or unchanged ('c').

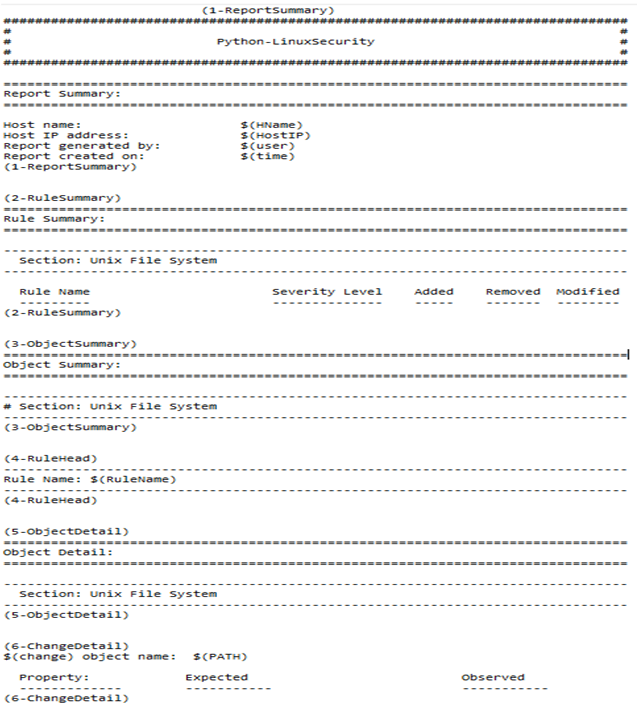
The script creates a database, establishes connections, and copies the baseline data for comparison. It also calculates and records changes while optimizing the database.



Figure

* **Reporting**

To generate a report, use the "PrintReport.py" script. It provides a thorough report that describes the changes seen as a result of the actions in the "check.py" script. It includes data formatting modules such as file statistics and database operations. The script's functions extract essential bits from a template text file, get data from the database, and turn it into a structured report. This final report should include a summary of rule violations, rule details, and information on detailed file modifications. Actual data, such as user and system information, is used to replace placeholders in the report. The main section of the script creates file locations, retrieves data from the database, and then calls the printReport function to create and save the report.



Figure

## Implementation

Once the proper environment is set up for the smooth operation of the product. The process of how this file protection system works is given below from initial configuration until the report creation as the final step.

**Initializing the product**  
The first stage in building the file protection system involves running the “Init.py” script, which creates and loads the database containing details on each file on the Linux operating system. That is started by importing necessary modules and the libraries. Subsequently, it defines the functions aimed at uploading file data in a database and creating the first database. It carries out cycle operation of the entire list of files on the unit and determines the type of rules and searches for certain rules. Data on every file is gathered and loaded into the database using SQL queries. This is followed by establishing a link to the SQlite database where a “FILEDB” table will be used for storing file related information. It records the number of times used during the initialization process which takes such a short time period after which the change committed becomes evident.

**Monitoring the changes in system files**

The script, “Check.py,” is able to go through the annoying task of monitoring by checking for any changes in files. It does this by organizing them into three different categories: “newly added,” “modified,” and “unchanged.” The first thing it does is make a connection with SQLite database that’s named after the current time. Then it compares new file attributes and MD5 hashes with the ones saved on the database to find any change. If either the hash or attribute differ from what they were initially, then it’s flagged as “modified,” if not, then it’s labeled as “unchanged.” For files that weren’t in the original database but are now present, they take on a new label of being “new.” However, if their hash changed or their attribute differs from how it was initially then they’re recategorized as "modified." In another scenario where no changes were found at all, they’re just copied. If there aren’t any changes then there isn’t really a reason to have them still on record so all unchanged files are completely removed from the database.

**Report generation**

The "PrintReport" script gathers all the results generated by the system from file integrity checks. Consolidates them in one location. Its equipped with tools to accomplish this task effortlessly. Initially it retrieves data, from a file for use. Organizes it into objects and their corresponding records. To generate the report this script accesses two databases; one containing the checks and another containing the data. Subsequently it extracts information about files, including their attributes and MD5 hashes. These files are then categorized into three groups; modified or copied based on rule types capturing the discrepancies between original file states. The next step involves constructing a structured report comprising sections such as a summary section displaying system information, an overview of rule objects detailed information, about each rule individually and finally an extensive breakdown of every file change that occurred. Once all these steps are completed successfully the data is organized accordingly. Saved as a text file. This script plays a role, in File Protection Systems as it provides system administrators with a method to identify any security breaches or integrity issues. This allows them to respond promptly and ensure operations on Linux.

# Evaluation

## Assessment of the Project results

The technical document that explains the features of the File Protection System gives an well organized overview of what the product can do. The document does a job of explaining the components of the system such, as "Init.py," "Check.py," and "PrintReport," and goes into detail about their roles and functions. Including code snippets and explanations helps make the document clear. The analysis of the systems capabilities in "Check.py " shows how it can effectively categorize changes to files as "" "modified," or "copied." The ability to customize rules adds another layer of flexibility, to the product. However it would have been helpful to include examples and practical use cases to demonstrate how the system can be applied in real life situations.

The document also analyzes how the system compares file attributes and MD5 hashes to identify changes accurately. Being able to pinpoint file modifications is an accomplishment. However it would have been beneficial to discuss any challenges or limitations faced during development as this would provide a understanding of the project.

In regards, to shortcomings in the outcome it would be advantageous for the document to include a thorough exploration of potential security weaknesses or risks that the system might not mitigate. Furthermore there is a lack of information, on how the system manages file recovery if unauthorized alterations occur. By addressing these aspects we can improve the comprehensiveness of the product.

## Lessons Learned

While developing this product a crucial lesson learned was how important it's to carefully analyze data and keep detailed records when implementing a system to protect files. This product emphasizes the importance of comparing file attributes and MD5 hashes to effectively identify any changes. Moreover the process of generating reports highlights the need, for organized data retrieval ensuring that the reports are informative and actionable. Additionally during the development phase a decision was made not to use the ORM framework, due to it's inability to process larger amounts of data. This highlights how important it is to select appropriate tools when dealing with sensitive data in high volume.

## Future Work

In terms of improvements there is potential, for growth and enhancement in the File Protection System. To begin with designing a user interface (GUI) could greatly enhance the accessibility and ease of use of the product making it more appealing to a broader range of users. Additionally exploring how the system interacts with security tools and platforms could lead to a comprehensive and interconnected security ecosystem. This could involve compatibility with used security information and event management (SIEM) systems thereby enhancing the products capabilities, for enterprise level security. Moreover continued research and development aimed at automating the process of restoring files to their state after changes would provide an added layer of protection and convenience.

# Conclusion

In summary the "FileGuard" File Protection System is an adaptable program that effectively strengthens the security of Linux systems. It continuously. Protects the integrity of system files accomplishing the core goals of gathering and storing file information while also keeping an eye on any changes made to files. The systems flexibility and extensibility allow for customization of security rules and notifications to meet needs.

Users can experience many advantages when utilizing "FileGuard." One of the benefits is its ability to safeguard the integrity of their Linux systems. By identifying and categorizing any file modifications it ensures the security and resilience of their IT infrastructure. Additionally “FileGuards” flexibility allows it to adapt to evolving security standards making it a valuable asset, in todays changing landscape of information security. To further enhance this security solution future efforts should prioritize expanding alerting options and enhancing reporting capabilities, for organizations seeking an user friendly approach.

# References

[1]S. K. Peddoju, H. Upadhyay, and L. Lagos, “File integrity monitoring tools: Issues, challenges, and solutions,” Concurrency and Computation: Practice and Experience, vol. 32, no. 22, 2020.

[2]P. M. Wanigasinghe, “Extending file permission granularity for linux,” Digital Library of University of Colombo School of Computing: Home, 28-Jul-2021. [Online]. Available: https://dl.ucsc.cmb.ac.lk/jspui/handle/123456789/4265. [Accessed: 01-Nov-2023]

[3]R. Fatrika, A. H. Alwis, M. R. Kurniawan, M. F. A. R. Simatupang, and F. Akmaresta, “IDS tripwire implementation: Using a virtual box on linux ubuntu 22.04 LTS,” JComce - Journal of Computer Science. [Online]. Available: https://jurnal.teknologiriau.net/index.php/jcomce/article/view/4. [Accessed: 01-Nov-2023]

[4]“Securing files and file systems,” SpringerLink, 01-Jan-1970. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4302-0005-5\_4#citeas. [Accessed: 01-Nov-2023]

[5]M. S. Javed, “Intrusion detection using tripwire in unix operating system,” [v1] | Preprints.org, 05-Feb-2023. [Online]. Available: https://www.preprints.org/manuscript/202302.0220/v1. [Accessed: 01-Nov-2023]