International Journal of Synergy in Engineering and Technology
IJSET
Journal homepage: https://tatiuc.edu.my/ijset/index.php/ijset/ e-ISSN: 2735-0622

Avoiding Data Loss and Corruption (Towards File Integrity Monitoring)

Noor Suhana Sulaiman[1], Muhammad Afiq Hilmi Mohd Farid[1]

[1]University College TATI, Jalan Panchor, Telok Kalong, 24000 Kemaman, MALAYSIA

*Corresponding author email: suhana@uctati.edu.my

KEYWORDS

File Integrity
Information Security
Host Based Intrusion Detection System (HIDS)
Cyber Security
Verification

ABSTRACT

Cybercrimes, encompassing a range of malicious activities such as phishing, identity theft, and denial of service attacks, pose significant threats to individuals and organizations worldwide. In Malaysia, these crimes have been on the rise, with reported losses reaching RM539 million in 2019 and escalating to RM560 million in 2021 with *four-fold* increase recorded in 2023. These cases highlight the pressing need for effective cybersecurity measures. Moreover, the reliance on digital evidence in legal proceedings underscores the importance of accurate analysis. However, the potential for evidence tampering complicates the interpretation of digital artifacts. For instance, alterations to browser history can mislead investigators, emphasizing the necessity of robust forensic techniques. To address these challenges, a proposed project, the File Integrity Checker (FIC), aims to enhance data security through continuous monitoring of file integrity. By comparing current file states to trusted baselines using cryptographic techniques, FIC detects unauthorized modifications or corruption. Leveraging File Integrity Monitoring (FIM) principles, FIC enables proactive identification of integrity issues and facilitates forensic auditing, thereby bolstering overall cybersecurity resilience. Incorporating tools like OSSEC, the File Integrity Checker project aligns with fundamental information security principles and contributes to a comprehensive risk mitigation strategy against data breaches and system vulnerabilities. Through proactive integrity verification, organizations can safeguard data reliability and foster trust in their digital operations, mitigating the escalating threats posed by cybercrimes in Malaysia.

Received 28 May 2024; Revised 10 September 2024; Accepted 30 September 2024; Published 01 October 2024.

1.0 INTRODUCTION

Cyberattacks, malware, and insider threats are among the methods through which digital files can be altered without authorization. Unauthorized access to a system can allows hackers to manipulate files to disrupt operations, pilfer confidential data, or inject malicious code. Similarly, hostile insiders may deliberately modify files to harm the company or advance personal agendas. The repercussions of such actions extend to individuals, organizations, and entire systems, resulting in data tampering, damage to reputation, and service disruptions. Meanwhile, files are a crucial aspect of computer

systems, serving as both input and output for many applications. Within operating systems, the file system stands out as a vital component requiring protection to uphold the integrity and availability of services. Safeguarding the integrity of files is now more essential than ever, given the vast amount of instructions and data present in modern computing environments [1].

As example of file tampering is the Pineapple attack, linked to the Wi-Fi Pineapple device, which exploits users' trust in wireless networks through a Man-in-the-Middle (MitM) attack [2]. In this scenario, the rogue Wi-Fi Pineapple establishes a fake access point, masquerading as a legitimate network and broadcasting deceptive signals to lure nearby devices into connecting. Devices configured to automatically connect to known or open networks may unwittingly link to the rogue Pineapple. Subsequently, the attacker intercepts and manipulates data exchanged between the device and the internet, which could involve eavesdropping on unencrypted data, injecting malicious content into web pages, redirecting users to counterfeit login pages, hijacking active sessions, and pilfering login credentials. The consequences of file manipulation can be extensive and diverse. It can compromise system and application integrity by causing loss or corruption of vital data. Financial losses, damage to reputation, and identity theft are potential outcomes of data breaches involving sensitive information like financial records or personal data. Furthermore, instances of file tampering can disrupt business operations, lead to legal ramifications, and necessitate costly recovery efforts to restore affected systems to their original state.

This paper introduces a software-based approach to file integrity monitoring, which dynamically assesses related files based on their sensitivity or security requirements. Sensitive files are those whose absence or improper modification can lead to unintended consequences for system services and operations. The subsequent sections of the paper are structured as follows: In Section 2, we review related literature and compare our proposed techniques with existing approaches. Section 3 elaborates on our proposed system, emphasizing the file security classification algorithm and FIM scheduling, and highlights the distinctions from previous FIM methods. Section 4 provides an analysis of the initial implementation and performance evaluation of our approach. Finally, Section 5 presents the concluding remarks and discussions.

2.0 LITERATURE REVIEW

In the operating system environment, every component, including instructions, device drivers, and other data, is stored in files. Modern operating systems contain a vast number of files, making them a primary target for attackers seeking to compromise system security. Attackers may attempt to modify or alter existing files, delete, add, or conceal related files using various techniques, underscoring the importance of file protection as a critical task. To address these security concerns, the implementation of File Integrity Monitoring (FIM) and other relevant system security tools is essential.

2.1 File Integrity Monitoring (FIM)

File Integrity Monitoring (FIM) stands as a crucial security protocol crafted to pinpoint illicit alterations in files or file systems. It entails the continuous surveillance and authentication of the integrity of pivotal system and application files, guaranteeing they remain unaltered in any unsanctioned or malevolent manner. Initially, a reference point is established, representing the recognized, secure condition of the files. FIM tools subsequently contrast the present file status with this benchmark, identifying any adjustments, additions, or removals. Upon detecting unauthorized modifications, the FIM mechanism triggers alerts, furnishing particulars regarding the nature of the alteration, the impacted file, and the timestamp of the adjustment. This facilitates swift responses and inquiries by security personnel. FIM further aids in enforcing policies, enabling administrators to

outline permissible alterations and distinguish between routine system updates and potential hazards. Furthermore, FIM is frequently amalgamated with other security solutions. Regulatory compliance requirements often necessitate the adoption of FIM, rendering it an invaluable asset in upholding the security and integrity of sensitive data and systems.

FIM serves as a pivotal tool in bolstering file security, alongside other related tools such as file integrity verification, file integrity tools, and file integrity checking. These tools share a common objective of ensuring the integrity of files involved. Additionally, integrity tools targeting aspects like kernel [3-6], application [7,8] and memory [9,10] aim to uphold the integrity of the operating system. FIM operates within host environments as part of a Host-Based Intrusion Detection System (HIDS). Its primary function lies in monitoring file integrity, detecting changes in file content, access control, privileges, groups, and other properties, whether initiated by authorized or unauthorized users. The overarching aim of these integrity tools is to alert system administrators to any alterations, deletions, or additions detected in the monitored system [11].

Fundamentally, file integrity tools ascertain the current checksum or hash value of monitored files against their original value to identify any modifications in file content. FIM can generally be categorized into offline and online monitoring schemes [12]. Offline monitoring involves periodic integrity checks according to user settings, whereas online monitoring monitors FIM-related files in real-time as they are modified. Recent solutions emphasize online or real-time monitoring [12-14] to bolster detection capabilities against malicious modifications. However, real-time checking often encounters performance degradation, rendering it impractical for real-world deployment. Moreover, deploying new integrity verification technologies, such as hardware-based protection mechanisms utilizing the Trusted Platform Module (TPM), demands a significant investment. This entails embedding TPM chips into computer hardware and implementing additional software for efficiency. The primary objective of FIM remains safeguarding file integrity within the operating system environment from intruders and unintended alterations by authorized users. Given its critical role in the operating system environment, ensuring the integrity of system files is paramount. Nonetheless, real-time monitoring of all system files proves challenging and costly, especially in multi-host and multi-operating systems environments.

File integrity monitoring, categorized as part of the functions of Host-Based Intrusion Detection Systems (HIDS), can be divided into offline and online integrity monitoring.

2.1.1 Offline File Integrity Monitoring

Tripwire is a [11] widely recognized file integrity monitoring tool that serves as a catalyst for the development of more robust FIM tools. It operates based on four processes: initialization, checking, updating, and testing. The core principle of FIM tools like Tripwire involves comparing the current hash values of files with their baseline values.

However, relying on the baseline database necessitates higher maintenance costs due to frequent system updates or patches [15]. Additionally, offline FIM requires scheduling to verify the integrity of related files, often resulting in delays in detecting modifications. Other tools such as Samhain [16], AIDE [17], and Osiris [18] follow a similar approach to Tripwire, thus inheriting similar issues. The main challenges with offline FIM include inspection frequency and modification detection effectiveness. To address this, we propose a dynamic inspection schedule by categorizing related files into groups and varying the inspection frequency among these groups. This approach allows FIM to maintain its effectiveness with a more acceptable performance impact on the system. Wu et al. introduce BinInt [19] as a new security model for binaries that prevents unauthorized execution of binaries. However, this model may not adequately cover data files that undergo more frequent

changes. We address this concern by focusing on the integrity of system files while ensuring our technique can also be applied to data files.

2.1.2 Online File Integrity Monitoring

Online file integrity monitoring (FIM) is proposed as a solution to address the delay in detection encountered with offline FIM methods, by actively monitoring security events involving system files in real-time [20]. However, operating in real-time necessitates access to low-level (kernel) activities, which in turn requires kernel modifications. This kernel-dependent approach is incompatible with other kernels and platforms. Moreover, real-time applications require scheduling guarantees from the operating system to function effectively, and load balancers are needed to minimize latency in multi-host environments. For instance, I3FS [21] proposes a real-time checking mechanism utilizing system call interception in kernel mode. However, this approach also necessitates kernel modifications on the protected machine, resulting in performance degradation due to constant checksum monitoring in real-time. Although I3FS offers a policy setup and update option for customizing the frequency of integrity checks, it relies on manual configuration by the system administrator. Several online FIM [22] and security tools leverage virtual machine introspection (VMI) [23] to monitor and analyze the virtual machine state from the hypervisor level. VMI, initially introduced in Livewire, has been applied in other tools such as intrusion detection in HyperSpector [24] and malware analysis in Ether [25].

2.2 Open Source Security (OSSEC)

OSSEC stands out as a cost-free, open-source host-based intrusion detection system (HIDS). Its creator, Daniel Cid, often refers to it as a log-based intrusion detection system (LIDS) within the log analysis segment of the program. Intrusion detection log analysis, the process of utilizing collected events to detect intrusions within a specific environment, is a key function of OSSEC. OSSEC identifies potential file manipulation by detecting inconsistencies between the baseline value and the current checksum or hash. Subsequently, it issues alerts [13] or messages to inform administrators or security personnel [28] of the detected activity. OSSEC is intended to monitor and analyse system logs, file integrity, and other security-related events on hosts running Linux. It works by collecting and analysing data from multiple log sources, identifying potential security incidents, and generating real-time alerts or notifications. OSSEC is built on a client-server model, with agent software installed on individual hosts that sends data to a central server for analysis. Log analysis, rootkit detection, file integrity monitoring, active response mechanisms, and policy monitoring are all capabilities. OSSEC is well-known for its adaptability and extensibility, allowing users to tailor rules and configurations to specific security needs. Organisations frequently use the tool to improve their intrusion detection capabilities and ensure the integrity of critical systems and data. OSSEC can also be integrated with other security tools, making it an important part of a comprehensive [26] cybersecurity strategy . Detection systems (IDS) play a crucial role in preempting potentially harmful intrusion attempts, intercepting assaults before they commence. An array of Open-Source IDS options are currently accessible, each offering distinct functionalities. Open Source Host-based Intrusion Detection System (OSSEC) stands out for its diverse capabilities, including log analysis, integration with other security tools, and file integrity monitoring. OSSEC comprises Agents tasked with gathering and transmitting event logs to a Manager responsible for analyzing and evaluating them against predefined rules. Within the Manager, when certain events align with specific rules, predefined actions are triggered within the Agents [29], such as blocking or unblocking a designated IP address.

2.3 Related Research

As depicted in Table 1, related to existing research, deploying real-time intrusion detection for insider threats within industrial control system workstations, this study utilizes file integrity monitoring (FIM) enhanced by behavior-based analysis. This approach surpasses conventional checksum-based monitoring by identifying anomalous activities tailored to the classification of each file. To expedite [13] responses to issues concerning highly sensitive data, files are categorized based on their security classification, such as "public," "internal," or "confidential". In the study titled "An Affordable and Practical Security Solution for Enhancing Log Management and File Integrity Monitoring," there is a focus on providing enterprises with a cost-effective means to enhance their cybersecurity stance without exceeding budgetary constraints. It emphasizes the necessity of a viable, economical security solution that integrates efficient log management and file integrity monitoring. Log management entails the collection and analysis of logs from diverse network and security devices to identify abnormal activities. In the article titled "Addressing Issues and Challenges of File Integrity Monitoring Tools," a common issue highlighted is the occurrence of numerous false positives, where valid modifications trigger alerts, leading to alert fatigue and potentially overlooking genuine security threats. Particularly for large enterprises managing extensive file volumes, scalability poses another challenge. To tackle these issues, robust database management and distributed monitoring capabilities are deemed crucial for File Integrity Monitoring (FIM) systems. Advanced FIM solutions leverage machine learning algorithms and baseline [27] comparisons to enhance accuracy and reduce false alarms, thereby mitigating the issue of excessive false positives.

Table 1: Related Existing Research

Year	Author	Title	Description	Tool	Methodology
2023	Al-Muntaser et al.	Real-Time Intrusion Detection of Insider Threats In Industrial Control System Workstations Through File Integrity Monitoring	By watching file activity and access patterns, this FIM goes beyond conventional checksum-based monitoring by utilizing behaviour-based analysis to identify anomalous behaviour unique to each file's categorization. To guarantee that situations involving highly secret data are attended to right away	N/A	Files are organized into groups according to the security classification they belong in, such as "public," "internal," or "confidential."
2020	Chen et al.	A Practical Low-Cost Security Solution For Log Management And File Integrity Monitoring	A practical low-cost security solution for log management and file integrity monitoring is invaluable for organizations seeking to enhance their cybersecurity posture without breaking the bank.	N/A	This approach combines two essential elements: efficient log management and file integrity monitoring. Log management involves collecting and analysing logs from various network and security devices to detect unusual activities

2020	Peddoju et al.	File Integrity Monitoring Tools: Issues, Challenges and Solutions	File Integrity Monitoring common challenge is the potential for a high volume of false positives, where legitimate changes trigger alerts, leading to alert fatigue and a risk of missing actual security threats. Additionally, scalability can be a challenge, especially for large organizations with a vast number of files to monitor. To tackle this issue, FIM solutions should be designed with efficient database management and distributed monitoring capabilities.	N/A	To address the issue of high volume of false positives ,advanced FIM tools use baseline comparisons and machine learning algorithms to reduce false alerts and improve accuracy.
2019	Tanner	Monitoring With OSSEC	Open Source Security (OSSEC) is a free, open-source, host-based intrusion detection system (HIDS). Daniel Cid, the author of OSSEC, often refers to it in the log analysis portion of OSSEC as a log-based intrusion detection system (LIDS). Log analysis for intrusion detection is the process of using the recorded events to detect attacks on a specific environment.	OSSEC	If OSSEC detects a difference between the current checksum or hash value and the baseline value, it interprets it as a possible indication of file tampering. After that, OSSEC generates an alert or notification to notify administrators or security personnel.
2019	Teixeira et.al.	OSSEC IDS Extension To Improve Log Analysis And Override False Positive Or Negative Detections	Intrusion Detection Systems (IDS) are used to prevent attacks by detecting potential harmful intrusion attempts. Currently, there are a set of available Open-Source IDS with different characteristics. The Open Source Host-based Intrusion Detection System (OSSEC) supports multiple features such as log analysis ,integration with other security tools and file integrity monitoring	OSSEC	consists of Agents that collect and send event logs to a Manager that analyses and tests them against specific rules. In the Manager, if certain events match a specific rule, predefined actions are triggered in the Agents such as to block or unblock a particular IP address.

3.0 METHODOLOGY

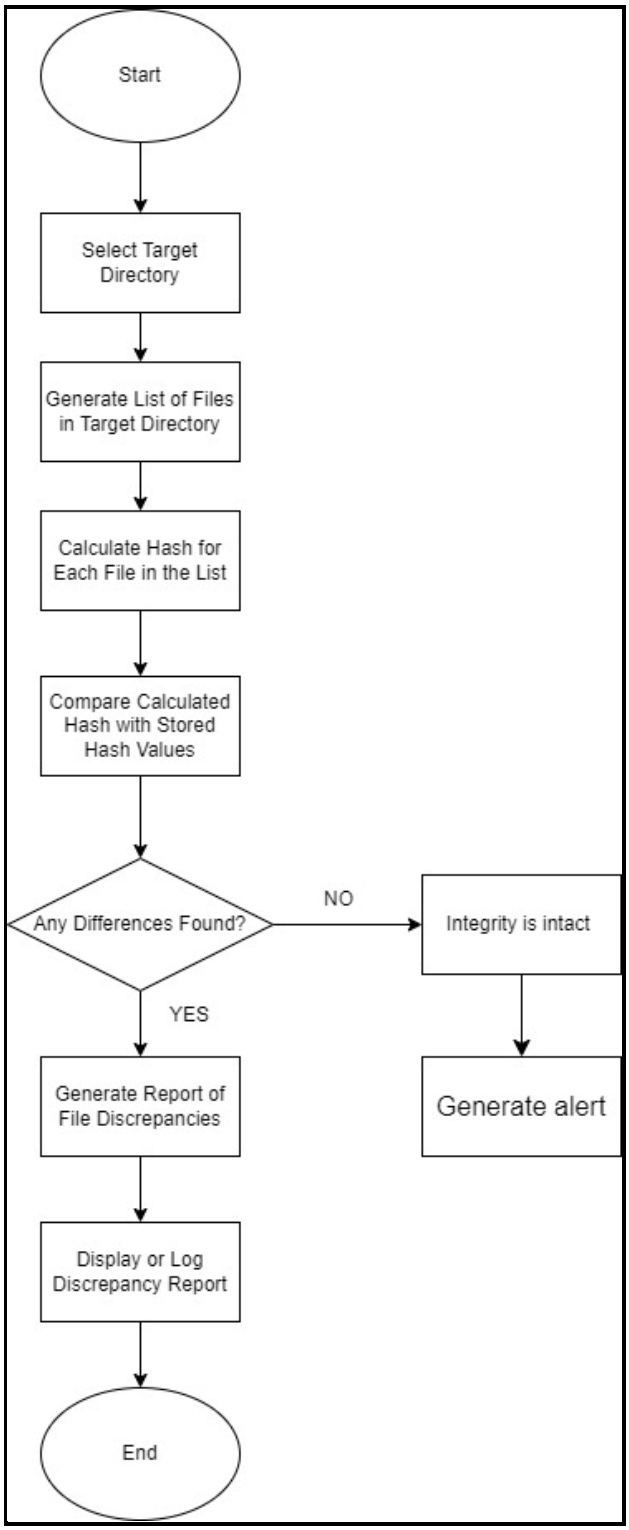
The proposed project is going through the actual development or construction of the project deliverables. The key objective is to convert the conceptual and design aspects into a tangible product, system, or solution. In this particular project, the development phase entails building the File Integrity Checker (FIC) using software such as OSSEC on Ubuntu Linux OS version 10, ensuring its functional and effective operation. Deploying OSSEC involves a series of steps, the intricacy of which depends on factors like network size and specific security

needs. Typically, the initial step involves installing the OSSEC server and agents on the systems slated for monitoring. Subsequently, configuration follows, where specifications are set for monitored files, directories, logs, and rules for file integrity and log analysis. This setup is done through XML files, with OSSEC offering comprehensive documentation for guidance. Effective communication between OSSEC agents and the central server is crucial, often requiring firewall and network configuration adjustments. While default rules cover common scenarios, customization of rules allows adaptation to precise security needs. Additional configuration might be necessary if OSSEC is to be integrated with other tools or systems. It's advisable to test the deployment in a controlled setting before implementing it in production to ensure proper functionality. Continuous monitoring and maintenance involve reviewing alerts, addressing incidents, and updating configurations to maintain system efficacy. Depending on your familiarity with security concepts and the specifics of your network environment, the overall complexity can range from moderate to high.

As illustrated in Figure 3.1, the process commences with the file integrity checker prompting the user to designate a target directory for verifying file integrity. Upon specification of the target directory, the system compiles a comprehensive list of files contained within it. Subsequently, the file integrity checker computes a unique hash value for each file in the compiled list, serving as a digital fingerprint for the content of each file.

These calculated hashes are then compared against previously stored hash values to identify any discrepancies. At a critical juncture, the system determines whether any differences have been detected. If no differences are found, indicating that the calculated hashes match the stored values, the file integrity checker concludes that the files' integrity remains intact. However, if differences are detected, the file integrity checker generates a detailed report highlighting the discrepancies among files.

This report is then presented to the user through an interface display or logged for future reference. Finally, after conducting a thorough inspection and reporting on the integrity of the files within the designated directory, the file integrity checker terminates its operation.



```
3- Configuring the OSSEC HIDS.

 3.1- Do you want e-mail notification? (y/n) [y]: y
  - What's your e-mail address? user@email.com

  - We found your SMTP server as: mx01.mail.com.
  - Do you want to use it? (y/n) [y]: y

  --- Using SMTP server:  mx01.mail.com.

 3.2- Do you want to run the integrity check daemon? (y/n) [y]: y

  - Running syscheck (integrity check daemon).

 3.3- Do you want to run the rootkit detection engine? (y/n) [y]: y

  - Running rootcheck (rootkit detection).

 3.4- Active response allows you to execute a specific
      command based on the events received. For example,
      you can block an IP address or disable access for
      a specific user.
      More information at:
      http://www.ossec.net/en/manual.html#active-response

  - Do you want to enable active response? (y/n) [y]: y

    - Active response enabled.

  - By default, we can enable the host-deny and the
    firewall-drop responses. The first one will add
    a host to the /etc/hosts.deny and the second one
    will block the host on iptables (if linux) or on
    ipfilter (if Solaris, FreeBSD or NetBSD).
  - They can be used to stop SSHD brute force scans,
    portscans and some other forms of attacks. You can
    also add them to block on snort events, for example.

  - Do you want to enable the firewall-drop response? (y/n) [y]:
```

Figure 3: Configuring OSSEC HIDS

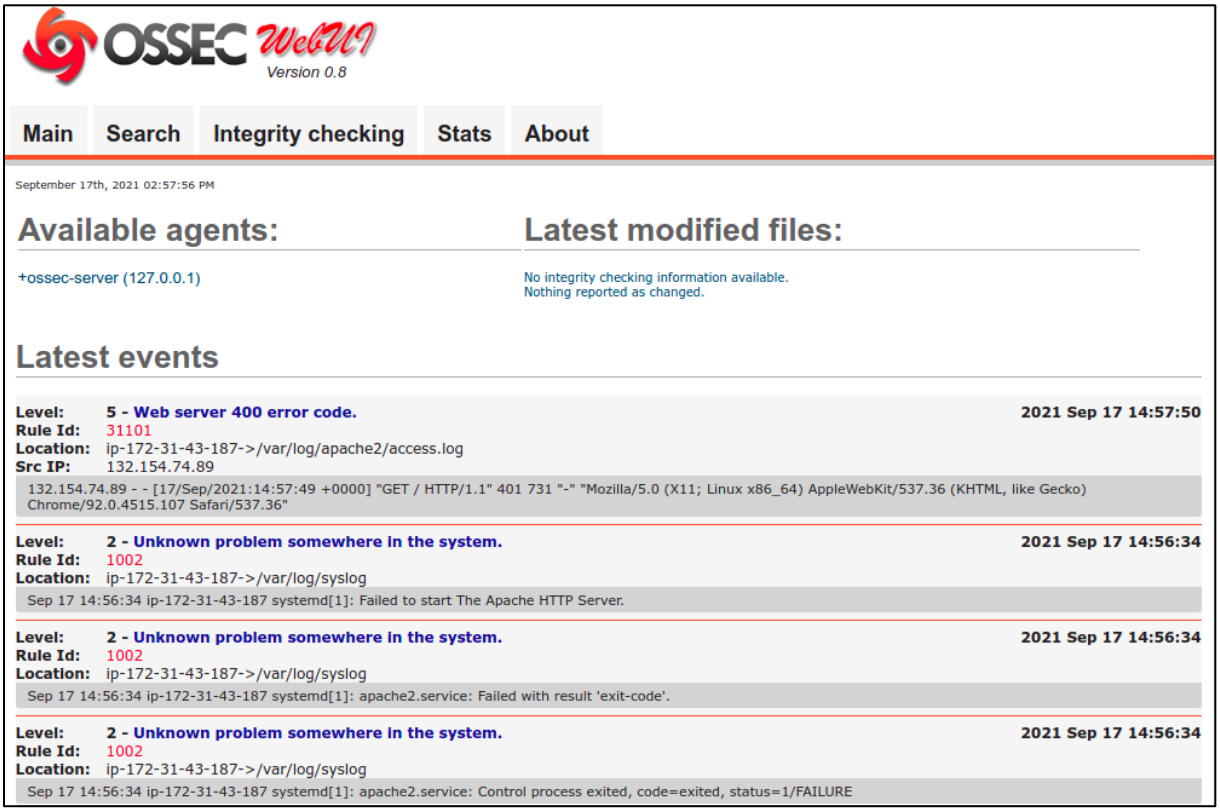


Figure 3: OSSEC Interface

4.0 CONCLUSIONS

In conclusion, the escalating threat of cybercrimes, evidenced by increasing losses in Malaysia, underscores the urgent need for robust cybersecurity measures. The imperative for effective defense mechanisms is evident. Additionally, the reliance on digital evidence in legal proceedings highlights the critical importance of accurate analysis, complicated by the potential for evidence tampering. To address these challenges, the File Integrity Checker (FIC) project proposes continuous monitoring of file integrity to enhance data security. By employing cryptographic techniques to compare current file states with trusted baselines, FIC detects unauthorized modifications or corruption, aligning with fundamental information security principles. Leveraging File Integrity Monitoring (FIM) principles, FIC facilitates proactive identification of integrity issues and supports forensic auditing, contributing to comprehensive risk mitigation against data breaches and system vulnerabilities. By incorporating tools such as OSSEC, the File Integrity Checker project offers proactive integrity verification, enabling organizations to safeguard data reliability and in still trust in their digital operations amidst the escalating cybercrime threats in Malaysia.

Author Contribution

Muhammad Afiq Hilmi Mohd Farid: Methodology, investigation, visualisation, writing and editing. Noor Suhana Sulaiman: Investigation, supervision, writing, and editing. Author: Methodology, writing and editing.

Conflict of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

5.0 REFERENCES

[1] Stallings, W. (2008). Operating Systems: Internals and Design Principles. Sixth ed.: Prentice Hall Press Upper Saddle River, NJ, USA.
[2] Maryam T. (2021). Man In the Middle Attack in Wireless Network. Conference: Man in the Middle Attack at Palestine.
[3] Nick L. Petroni, J., et al. (2006). An Architecture for Specification-Based Detection of Semantic Integrity Violations in Kernel Dynamic Data, Proceedings of the 15th conference on USENIX Security Symposium - Volume 15. USENIX Association: Vancouver, B.C., Canada.
[4] Xu, M., et al. (2007). Towards a VMM-Based Usage Control Framework for OS Kernel Integrity Protection, Proceedings of the 12th ACM symposium on Access Control Models and Technologies. ACM: Sophia Antipolis, France.
[5] Xuan, C., J. Copeland, and R. Beyah (2009). Shepherding Loadable Kernel Modules through On-demand Emulation, in Detection of Intrusions and Malware, and Vulnerability Assessment. p. 48-67
[6] Seshadri, A., et al. (2007). SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes, Proceedings of twenty-first ACM SIGOPS Symposium on Operating Systems Principles. ACM: Stevenson, Washington, USA.
[7] Lifu, W. and P. Dasgupta (2007). Kernel and Application Integrity Assurance: Ensuring Freedom from Rootkits and Malware in a Computer System. Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference.

[8] Li, N., Z. Mao, and H. Chen (2007). Usable Mandatory Integrity Protection for Operating Systems, Proceedings of the 2007 IEEE Symposium on Security and Privacy. IEEE Computer Society.
[9] Dewan, P., et al. (2008). A Hypervisor-Based System for Protecting Software Runtime Memory And Persistent Storage, Proceedings of the 2008 Spring Simulation Multiconference. The Society for Computer Simulation, International: Ottawa, Canada.
[10] Akritidis, P., et al. (2008). Preventing Memory Error Exploits with WIT. Security and Privacy. SP 2008. IEEE Symposium on. 2008.
[11] Kim, G.H. and E.H. Spafford (1994). The Design and Implementation of Tripwire: A File System Integrity Checker, Proceedings of the 2nd ACM Conference on Computer and Communications Security. ACM: Fairfax, Virginia, United States.
[12] Jin, H., et al. (2010). A Guest-Transparent File Integrity Monitoring Method in Virtualization Environment. ComputMath. Appl.. 60(2): p. 256-266.
[13] Feng, Z. (2009). VRFPS: A Novel Virtual Machine-Based Real-time File Protection System. In Software Engineering Research, Management and Applications, ACIS International Conference. IEEE Computer Society.
[14] Junghan, K. (2010). NOPFIT: File System Integrity Tool for Virtual Machine Using Multi-byte NOP Injection. International Conference on Computational Science and Its Applications.
[13] Al-Muntaser, B., Mohamed, M. A., & Tuama, A. Y. (2023). Real-Time Intrusion Detection of Insider Threats in Industrial Control System Workstations Through File Integrity Monitoring. International Journal of Advanced Computer Science and Applications, 14(6), 326–333.
[15] Quynh, N.A. and Y. Takefuji (2007). A Novel Approach for A File-System Integrity Monitor Tool of Xen Virtual Machine, Proceedings of the 2nd ACM Symposium on Information Computer and Communications Security. ACM: Singapore.
[16] http://www.la-samhna.de/samhain/. The SAMHAIN file integrity / host-based intrusion detection system. 2006.
[17] Lehti, R., M. Haber, and R.v.d. Ber. The AIDE manual. 20 May 2011; Available from: http://aide.sourceforge.net/stable/manual.html.
[18] Wotring, B. and B. Potter, Osiris (205). Host Integrity Monitoring Using Osiris and Samhain. Syngress: Burlington. p. 141-239.
[19] Wu, Y. and R.H.C. Yap (2011). Towards a binary Integrity System for Windows, Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. ACM:Hong Kong, China.
[20] Lee, M., et al. (2010). Supporting Soft Real-Time Tasks in The Xen Hypervisor, Proceedings of the 6th ACM SIGPLAN/SIGOPS International Conference on Virtual execution environments. ACM:Pittsburgh, Pennsylvania, USA.
[21] Patil, S., et al. (2004). I3FS: An In-Kernel Integrity Checker and Intrusion Detection File System, Proceedings of the 18th USENIX Conference on System Administration. USENIX Association: Atlanta, GA.
[22] Pfoh, J., C. Schneider, and C. Eckert (2009). A Formal Model for Virtual Machine Introspection, Proceedings of the 1st ACM Workshop on Virtual Machine Security. ACM: Chicago, Illinois, USA.
[23] Tal Garfinkel, M.R. (2003). A Virtual Machine Introspection Based Architecture for Intrusion Detection in Proc. Network and Distributed Systems Security Symposium.
[24] Kourai, K. and S. Chiba (2005). HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection, Proceedings of the 1st ACM/USENIX international Conference on Virtual Execution Environments. ACM: Chicago, IL, USA.
[25] Dinaburg, A., et al. (2008). Ether: Malware Analysis Via Hardware Virtualization Extensions, Proceedings of the 15th ACM conference on Computer and Communications Security. ACM: Alexandria, Virginia, USA
[26] Chen, L., Yang, M., Wimmer, H., & Wilbert, B. (2020). A Practical Low-Cost Security Solution for Log Management and File Integrity Monitoring. International Conference on Mobile Multimedia Communications (MobiMedia), 2020-Augus.
[27] Peddoju, S. K., Upadhyay, H., & Lagos, L. (2020). File integrity monitoring tools: Issues, challenges, and solutions. Concurrency and Computation: Practice and Experience, *32*(22), 1–8.
[28] Tanner, N. H. (2019). Monitoring with OSSEC. Cybersecurity Blue Team Toolkit, 57–66.
[29] Teixeira, D., Assunção, L., Pereira, T., Malta, S., & Pinto, P. (2019). OSSEC IDS Extension to Improve Log Analysis and Override False Positive or Negative Detections. Journal of Sensor and Actuator Networks, 8(3).

