Self-Healing Intrusion Detection in Networks Using Blockchain

By

Ahmed Salman

A thesis submitted to the Department of Information Security, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfillment of the requirements for the degree of MS in Information Security

Oct 2018

CERTIFICATE

This is to certify that **Ahmed Salman**, Student of **MSIS-15** Course Reg. No: **00000201399** has completed his MS Thesis title **"Self Healing Intrusion Detection In Networks Using Blockchain"** under my supervision. I have reviewed his final thesis copy and am satisfied with his work.

_____

Thesis Supervisor

(Brig. Imran Rashid, PhD)

Dated: _____

ABSTRACT

Host based Intrusion Detection Systems rely on a database to indentify threats to the computer system. The intrusion detection process in a protected system depends entirely on the integrity of this database and is therefore a preferred target of attackers. Existing techniques to secure this database either lack flexibility or the desired level of protection. Hence, there is a need to explore new techniques to protect the database in a better way. In this research thesis, we have explored the blockchain technology to secure the IDS database by storing the system checksum on blockchain immutable ledger. Secure updates for operating system are also proposed to be tackled through the blockchain distributed consensus. Since blockchain is a very new technology, therefore this research is a first attempt to secure IDS database integrity in a distributed manner.

The research work explores various IDS types and techniques along with existing blockchain projects related to scalability, storage and malware detection. In our proposed IDS design, we identified some integrity related threats and expect that the proposed system will be secure against such attacks through use of blockchain technology.

ACKNOWLEDGMENTS

I would like to extend sincere gratitude to my supervisor, Brig Imran Rasheed PhD, whose insight on technical matters and research approaches made this work achievable. His gracious supervision and professional opinion have been very useful through all stages of research work. I also feel indebted to Maj Muhammad Faisal Amjad (retd) PhD, for extensive deliberations and important suggestions which contributed to a great extent in improving my work. This thesis also benefited greatly from the valuable suggestions made by Asst Prof Mian Waseem Iqbal and Major Asad Khan Sadozai. I take this opportunity to thank them.

Special thanks are due to my family, for their continuous support and understanding, for putting up with unusual study hours and their patience. This thesis has been written during my stay at the Military College of Signals for the Masters Degree, which defines a wonderful phase of my life. Here I got the opportunity to improve my knowledge and understanding of information systems and security. I would also like to express thanks to my institute for providing excellent working conditions and study environment.

Lastly, all praise to Almighty Allah to give me the strength and knowledge to embark on this research work. I am ever indebted to Allah for his countless blessings upon me.

TABLE OF CONTENTS

4. BLOCKCHAIN INTRODUCTION AND LITERATURE REVIEW

5. PROPOSED IDS DESIGN

6. FUTURE WORK AND CONCLUSION

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Introduction

With the exponential increase in use of computer systems in all fields of technology, the number of attacks has also increased. In recent years, cyber security products and services have seen increasing use in all business and processes to secure against such threats. With every passing day the threats are evolving and gaining new forms. This necessitates that the security apparatus is also upgraded according to new threat patterns and techniques. Intrusion detection systems at both Network and Hosts level form an essential part of a network's security posture.

Intrusion detection systems come in many shapes and forms, each using different techniques and analysis mechanisms. Systems at network and host level have their own significance. In any network, intrusion detection should be implemented at both levels for wholesome security. Intrusion detection systems at host level are commonly known as host-based intrusion detection systems or HIDS. These systems generally have a database of signatures or a profile to carry out the detection analysis. In all host-based systems, the detection is completely dependent upon the integrity of the database. An attacker can easily defeat the IDS if he is able to modify the database as per his own designs.

Different techniques are used to secure the databases in hosts based IDSs. Some experts suggest storing the signature DB remotely, some keep it in write protected media etc. However, each solution has its advantages and weaknesses. In this work, we will focus on achieving database integrity through use of blockchain distributed ledger. Blockchain is a new technology to store value or information in an immutable form. In its existing form, blockchain is still facing some limitations with regards to storage and scalability. We will endeavor to arrive at a solution where balance between blockchain efficiency and IDS integrity is achieved. Furthermore, secure updates in systems with installed IDS are also a cause of concern due to malware injected updates. We will also try to develop a framework to use blockchain to analyze updates for presence of malwares traces in a collaborative manner. We will focus on file integrity checking HIDS for our framework due to their simpler design and requirements.

1.2 Motivation

The motivation behind this work has two distinct facets. First is that intrusion detection is a very important field in this digital age. Every legacy system is converting into a

digital platform with multiple sensors e.g. it is expected that by year 2020, around 30 billion IoT devices will be integrated in digital systems. Many critical systems will depend on the correct value inputs from such devices, therefore it is imperative to keep the inputs safe from compromise. In such a scenario, intrusion detection systems gain even more importance. Therefore, a study on IDS can provide new avenues to secure existing and new networks.

Secondly, blockchain is an emerging technology and being hailed as the next internet revolution. There has been a quantum increase in blockchain research the world over in last few years. However, blockchain research landscape in Pakistan is not very active and requires a lot of attention to remain in step with the world. Blockchain is deemed to provide solutions to all threats related to centralization in digital world. This warrants that blockchain research initiatives must be generated and encouraged. In this work, it will be endeavored to undertake detailed study of blockchain research landscape and reach upon a possible solution to IDS integrity as stated earlier.

1.3 Problem Statement

The problem statement for our research thesis is as follows: -

A HIDS which uses files integrity checksums to detect intruder activity, can be evaded by tempering the checksum DB **[1], [2]** ***and updates in such hosts are difficult due to increase in false positive rate.***

1.4 Research Objectives

The purpose of this research work is to recommend a framework for an Intrusion Detection System based on Blockchain with following objectives: -

a. Anomaly based intrusion detection by comparing system states on Blockchain distributed ledger.
b. Self-healing in hosts if anomaly is found.
c. Malware detection in updates using a suitable Blockchain consensus model.

1.5 Contributions

With the stated motivation and research objectives above, we aim to contribute following to the research landscape in Pakistan: -

a. Detailed study of Blockchain research and applications.
b. Study of blockchain usability to secure intrusion detection systems in computer systems.
c. Study of blockchain based malware detection mechanisms.
d. Study of intrusion detection design with system recovery capability.

1.6 Research Methodology

The research has been carried out in following phases: -

a. ***Phase 1*** - Study of publications and surveys on HIDS and its Database protection mechanisms.

b. ***Phase 2*** - Study of publications on Blockchain technology and applications.

c. ***Phase 3*** - Study of blockchain projects related to intrusion detection, malware detection, data storage and blockchain scalability.

d. ***Phase 4*** - Designing a novel IDS architecture levering blockchain technology for DB security.

1.7 Thesis Outline

The thesis is organized in five chapters, brief description of next chapters is as follows: -

a. ***Chapter 2*** - Presents introduction to Intrusion Detection System and its types. Advantages and Disadvantages of each type are also given. Lastly, some research considerations for Host based intrusion detection systems are also given.

b. ***Chapter 3*** - It focuses on File integrity checking HIDS and its types. Basic introduction to attacks on such HIDs and existing protection mechanisms for its databases are also discussed.

c. ***Chapter 4*** - Basic introduction to Blockchain technology is given along with various application scenarios. The chapter ends with a brief study of existing work related to our thesis objectives.

d. ***Chapter 5*** - This chapter presents our IDS framework based on blockchain technology to achieve the stated objectives.

Chapter 2

INTRUSION DETECTION

2.1 Introduction

Intrusion detection systems (IDS) are a significant part of security in a computer network or a standalone system. These systems analyze events in a network or a computer system for indicators of security breaches. Due to recent explosion of cyber attacks on computer systems and networks, IDSs have become an integral part of computer security paraphernalia. Hackers or attackers attempt intrusions to compromise confidentiality, integrity and availability of systems. Attackers may attempt an attack from outside the network or they may be insiders trying to gain additional privileges or misuse their privileges. IDS are the software or hardware products used to detect or prevent such attacks from succeeding using various techniques.

2.2 Reasons to use Intrusion detection systems

One can find numerous compelling reasons to use IDS in their systems to ensure security, some are listed below: -

a. To deter potential intruders by enhancing risk of detection and subsequent punishment.
b. To detect new security attacks and threats.
c. Detect pre-attack scanning activities like probes.
d. Log encountered attacks.
e. Compliance with standard security design especially in large enterprises.
f. Provide useful data for analysis and improve security architecture for future attacks.

2.3 Types of Intrusion Detection Systems

Present day IDSs come in different forms and types. These systems are based on various techniques dependent upon their use. Before going into types of IDSs, we will briefly describe the basic IDS model: -

2.3.1 Basic IDS Model

Most IDS systems can be conveniently described in following three modules: -

a. *Source of Information* - All intrusions lead to generation of events in network and hosts. These events are used by IDSs to detect intrusion attempts. These events are compiled from different levels of system i.e. network, host or application etc.

b. *Analysis engine* - This part of the intrusion detection system performs the important task of performing analysis on events information and detecting signs of intrusion attempts. Analysis engines use various techniques i.e. signature and anomaly detection.

c. *Response to intrusion attempts* - These form the actions unleashed on detecting an intrusion. Two types are actions are taken on signs of intrusion i.e. active and passive response. Active response involves automated system on part of detection system to stop the intrusion. Active response systems are generally known as intrusion prevention systems (IPS). Passive response systems generate alerts or reports and involve human intervention to further stop the detection or removing threat.

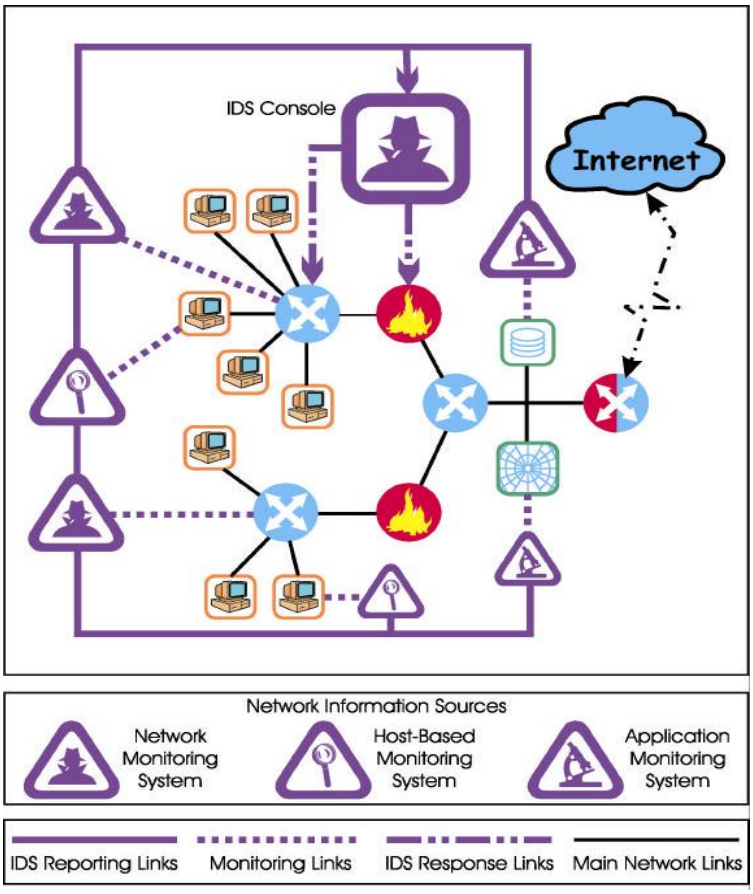

Figure 1.1 Deployments of Intrusion Detection Systems in a Network

2.3.2 Types based on information source

This is the most common method to classify IDSs. Some detection systems use packets captured from network segments to analyze and detect signs of intrusion while others use operating system events inside a host or at application to detect threats. Detail is as under: -

2.3.2.1 **Network Intrusion Detection Systems**

Most detection systems used in commercial entities are network intrusion detection systems. These systems capture data packets from various locations in a network and analyze them for threats. A network intrusion detection system monitoring a network segment will protect all host systems connected on that network segment from malicious data traffic. Network detection systems may consist of sensors performing a single task or multiple hosts at various points of network to monitor traffic. Such hosts capture traffic and perform analysis locally. Threats detected are then reported to a central management system for further action. Events or traffic sensors are generally initiated in stealth mode and remain hidden from attackers. As these sensors are only communicating with the IDS, so securing these is easier than other network elements.

2.3.2.1.1 **Advantages**

a. A large network can be monitored for threats by using some well-located sensors.

b. The deployment such IDSs carry no impact on normal working of the network. The network IDSs are passive devices which only sniff data traffic without effecting the network communications. Thus, such systems can be incorporated in a network with minimal modifications to network design.

c. These systems can be secured against attacks by hiding from attacker's view.

2.3.2.1.2 **Disadvantages**

a. Network IDS may fail to detect an attack in high traffic networks or in periods of high traffic. These systems may face failure when faced with processing high volume of traffic owing to limited processing resources. A possible solution to this problem is by implementing the system in hardware completely to provide more processing power and memory resources. This is relatively much faster but expensive solution. Researchers are endeavoring to achieve high detection rates with minimal resources by using various detection techniques.

b. Network IDSs do not provide information if an attack detected was a success or not. Merely attack detection information is provided by the system and then an administrator must manually see the effects of attack or extent of penetration on the victim host.

c. Some systems face difficulty in detecting more advanced attacks involving fragmented data packets. Generally, these packets interact with IDS in a hostile way and cause these systems to crash or become unstable.

d. Modern switched data networks sometimes do not allow network IDS to function at maximum potential. These switches divide network in small local networks and reduce the range of network IDSs. Some switches also do not provide a universal monitoring port and limit the traffic monitoring to single host segment. The encrypted communication between network elements also cannot be analyzed by IDS, even with a universal port. With exponential increase in use of VPNs, this problem is becoming more pronounced and increases the chances of intrusion manifolds.

2.3.2.2 **Hosts Intrusion Detection Systems**

Host based intrusion detection systems make use of internal system events and information to detect attacks. Being closest to an OS, these systems are best placed to determine the exact processes and profiles involved in an intrusion. Also, these systems can also monitor the effects of an intruder attack being closely located. This ability is not available to network IDSs.

Generally, these systems use two types of information from within a host to perform intrusion analysis i.e. system logs and audit trails. Audit trails generated in kernel of a system, provide best details of an intrusion as this is the inner most level of an OS. These are also better protected than logs files. On the contrary logs are less complex and smaller and are very easy to analyze. Some Host IDSs are connected to a single management console to control these many hosts, while some are designed to output reports compatible with a Network based management system.

2.3.2.2.1 **Advantages**

a. These have better visibility on attacks on a host than network IDSs.

b. Host IDSs are also in a better position to analyze encrypted traffic than network IDSs. Such systems can analyze the suspicious traffic before it is encrypted in a host and on decryption on the receiver host.

c. Host IDSs are not limited by network switches due to their independence from such network elements.

d. Host IDSs can even detect Trojan horse attacks within a system by analyzing the OS audit trails and checking files for integrity. The discrepancies found in the process execution can alert a host-based IDS.

2.3.2.2.2 **Disadvantages**

a. These systems are a little complex to manage as configuration must be done in every host.
b. Host based IDSs are more prone to attacks as these depend on information from within a system and sometimes even the analysis engine is in the same host. A targeted attack may be able to block or modify the information sources or even shut down the analysis engine.
c. Host IDS cannot be used for network level detection scans as these systems can only detect traffic coming into their respective systems.
d. A denial of service attack can effectively shut down the Host based intrusion protection.
e. Audit trails are massive and require additional storage within the protected system.
f. The protected system's resources are used by the host IDS to perform analysis on the audit trails and logs thus taking an undesirable toll on system performance.

2.3.2.3 **Application level IDS**

Such systems may be considered a type of Host intrusion detection systems. These are meant to analyze events happening within applications i.e. application transactions logs. The database of such system is populated with application and domain specific data and effectively detects suspicious application's behavior due to users misusing their authorization.

2.3.2.3.1 **Advantages**

a. Application based IDSs analyze user - application interaction and can detect any malicious activity with user identity.
b. As with most Host IDSs, application IDSs can also operate in encrypted system environment as these interact with application and analyze information presented to the user in unencrypted form by the respective application.

2.3.2.3.2 **Disadvantages**

a. Application based IDS are more prone to attacks than host based IDSs as these depend on application transaction logs which are less protected than system audit trails. An intruder might find a way to feed fake transaction logs to the IDS and bypass such systems.
b. Such systems operate at the highest level in an OS and thus cannot find any threats based on Trojan horses or malwares based at lower levels i.e. kernel. Therefore, these systems must be used in conjunction with Host and Network level IDSs.

2.3.3 **Types based on analysis engine**

Events in a host or network are analyzed by using two types of analysis approaches i.e. misuse and anomaly detection. Most enterprise systems use misuse detection approach and analyze any event perceived to be malicious. Such event will be in signature database of the detection engine and provide detection capability. On the contrary, anomaly detection approach tries to find traces of abnormal behavior by a process etc. The anomaly DB consists of event profile known to be good. Any deviation is considered to be bad. Anomaly detection approach relatively less used than misuse detection. Each type has advantages and weaknesses, but it appears that mostly misuse approach is used in systems and networks with some elements using anomaly approach.

2.3.3.1 **Misuse detection**

These systems look for system activity against known patterns of malicious behavior. These patterns are known as signature and as stated earlier, these systems are also known as signature detection systems. Common misuse detection systems used in enterprises look for separate signatures in single pattern of events in an attack but there are also advanced systems which can detect groups of attacks by analyzing a single signature (state-based detectors).

2.3.3.1.1 **Advantages**

a. Rate of false detections in a misuse detection system is very low. These are very fast and diagnose an attack if a known pattern is found. This can provide a lot of breathing space to administrators to stop an ongoing attack.

b. These systems allow users to quickly identify security issues due to exact matching of mal activity pattern. The user can initiate the counter measure more efficiently at any level.

2.3.3.1.2 **Disadvantages**

a. Misuse detectors cannot detect new attacks or attacks for which these do not have any known pattern. Therefore, the DB must be updated quickly.

b. These systems might not be able to detect variants of known patterns of mal activity as these are mostly trained to find exact matches of signatures. However, state-based detectors might be able to overcome such threats but not commonly used.

2.3.3.2 **Anomaly Detection**

As against misuse detection engines, these systems detect anomalous or unusual behavior in a system or network. These make use of the fact that attacks generate different

activities and event than a normal system or network and can be easily and distinctly detected by anomaly systems. These systems use historical data from a system or network to build profiles for normal users and network. Then the current events and activities are collected and analyzed for anomalous behavior against these constructed profiles. Various techniques are used in anomaly detection approach, some are listed as under: -

a. *Statistical measures* - It uses parametric data and non-parametric data. Parametric data is where distribution of the documented attributes of an entity is taken to match a single pattern. Non-parametric data is where the said distribution is learned from historic data collected over time.

b. *Threshold detections* - It segregates selected user behavioral attributes into permissible and non-permissible levels. The attributes are expressed in terms of repetitions or counts i.e. file access frequency, failed login attempts, CPU usage by a process etc. The threshold may remain static or be heuristic, which change over time with change in observed patterns.

c. *Rule based detection* - It is quite similar to non-parametric detection, however it differs in the way the values are analyzed. In non-parametric detection, values are in expressed in numeric quantities but these systems, value observed are expressed in terms of rules.

d. *Misc* - Certain *other* detection models include genetic algorithm, neural networks and immune systems.

Only the first two models are commonly used in enterprise or commercial solutions. Anomaly detection systems can detect new forms or zero-day attacks due to perceived deviation from known attack patterns, but these also produce a lot of false positives due to natural usage patterns of users. At the same time, opposite is true for signature or misuse detection systems. Sometime both types of systems can be used in combination. An anomaly system based on threshold detection can produce a user profile giving access frequency of a user. The same data can then be used by a misuse detection engine to generate an alarm if the frequency exceeds from this built profile.

Generally, we do not find systems using anomaly detection other than a few. Whenever used, it is generally utilized for port scanning or analyzing network based anomalous behavior.

2.3.3.2.1 **Advantages**

a. These can detect unusual behavior and thus can detect new forms of attacks without specific knowledge about such attacks.

b. The produced information about newly detected attacks can then be used by misuse detectors for fast detection.

2.3.3.2.2 **Disadvantages**

a. These systems have a great false positive rate due to deviations in natural user patterns and network traffic.

b. A lot of training is required to make profiles in these systems from historical data and a lot of data is required to build these profiles.

2.3.4 **Types based on Response options**

After an intrusion is detected using different detection methods, IDSs generate responses in systems and networks to stop or eliminate these attacks. Some generate alerts and update logs for the intrusion records, while some might have more active responses to deal with the threat. Response of IDS has great bearing on the normal working of an attacked system. Enterprise level systems might have passive response or active response. Some might even have a combination of the two. Details are as under: -

2.3.4.1 **Active response systems**

These systems have automated responses, which are taken when some intrusion is detected. Following three types of active responses exist: -

a. *Collecting more information* - Harmless but sometimes most effective response is to collect additional information on a suspected attack. It is always better to look more deeply into the suspected attack to confirm if an actual attack is taking place. This can include enhancing the sensing level to get more data i.e. capturing more data packets through a network monitor, increasing quantum of events recorded for the audit trail etc. Extra information gathered like this can also assist the user to investigate and finally apprehend the attacker and support legal and criminal proceedings.

b. *Environment Change* - In this type of response, an attack is halted and ability to intrude again is blocked by changing the environment. Normally, the access is not blocked instead the IP address launching the attack packets is detected and blocked. An expert attacker can be very difficult to block, however some actions can effectively hamper the attacker's movement i.e. Terminate connection by injecting TCP reset packets, Block attacker's IP address, opened network ports, protocols and services by reconfiguring firewalls and routers. In extreme conditions, all network connections can be terminated temporarily.

c. *Acting against the intruder*

This includes launching against the attacker after gaining information about its location and where about. However, this is not advisable by moist experts due to legal considerations. Many users use fake spoofed IP addresses, therefore probability to identify the true attacker is very low. It might happen that a counter attacker is launched on a wrong user and sites. Also, an attacker can be provoked to intensify its attack following this option. Only if a confirmed location and identity is found, this option can be exercised with an expert supervision only.

2.3.4.2 **Passive response systems**

These systems only provide attack information to alert users and expect users to act against the attack. Most enterprise systems use this response only. Details are as follows: -

a. *Alerts and notifications* - These are generated to inform the user about a detected attack. Most typical is a pop up or an alarm on a system screen. A user can configure the IDS for alerts to a great deal. The information in an alert can also vary to a great deal depending upon the configuration settings by the user. Some IDSs can also send remote alerts through SMS or emails to incident response team members. This feature is extremely useful in large enterprises and commercial establishments. Email alert is not advisable as the only alert method as an attacker might block the administrator emails.

b. *Plug-in and SNMP traps* - *Some* advanced IDS are configured to generate alerts for a network NMS. SNMP traps are used to launch these alerts in NMS, where such alerts are handled by the security staff. The biggest advantage of this kind of alert scheme is that the entire infrastructure can be configured to respond to the attack. Other advantages are shifting the processing load from the attacked system to a safe system and ability to use the existing communications infrastructure for reporting using known protocols.

2.3.4.3 **Archiving and Reporting capability**

Almost all enterprise level IDS have capability to produce information and reports on a routine basis. Some can provide report of events in a system or about intrusion detection data over a certain reporting period. Some also produce logs or statistics in suitable forms for inclusion in IDS DB for future detection reference.

2.3.4.4 **Failsafe features in IDS response**

Failsafe features are meant to protect the IDS process from being bypassed by an attacker. Several types of functions require failsafe features. These include reliable and silent monitoring to remain hidden from attacker by avoiding sending alerts and messages in plain

text. Otherwise it can alert the attacker and IDS itself can be attacked. IDS communication can be protected by using encrypted tunnels and cryptographic steps and enhances the system reliability.

2.3.4.5 **Misc detection tools**

Some other tools are also regarded as intrusion detection system as these perform similar work. These include Honey pots, vulnerability scanning systems, padded cells etc. These systems can be used in addition to traditional IDS to enhance the security posture of an enterprise.

2.4 **IDS Strengths and Weaknesses**

While selecting IDS for security in an organization, it is vital to know about its capability as it may not be able to perform all security functions. Selecting IDS for security therefore, depends largely on company's needs and its capability. Detail is as under: -

2.4.1 **Strengths of IDS**

a. Keep track of security configurations and detect any changes.
b. Detecting patterns of known attacks.
c. Detecting different variants of known attacks.
d. Management of OS logs and audit data.
e. Provide default information on security policies.
f. Provide a convenient way to non-experts to monitor & protect the system.
g. Alerting security staff about a detected attack.
h. Monitoring and analyzing system events for information on attack patterns.
i. Tracking any changes to baseline security settings of a system.

2.4.2 **Limitations of IDS**

a. IDSs cannot perform functions of missing or weak necessary security peripherals like firewalls, access management systems, Virus detection and elimination and link encryption.
b. These systems cannot detect report and respond to an attack instantaneously under periods of heavy processing and network loads.
c. Some systems cannot detect new attacks or previous attack variants.
d. May have limited response to expert hackers.
e. Automatic investigation of attacks without human intervention may not be possible.
f. May fall victim to bypass or circumventing attacks.

2.5 **Additional literature on Host intrusion detection system**

As our thesis is primarily focused on host intrusion detection systems and their security, therefore some additional discussion is necessary before proceeding further on security issues.

2.5.1 **Reasons for less research on Host intrusion detection system**

Authors in [3] provide an in-depth analysis on why Network intrusion detection systems are the focus in latest researches. Authors have enhanced the argument put forth by J. Hu in [4] and have given following reasons for less study on Host intrusion detection systems: -

a. *Resource constraint and real time* - The most desirable feature of an intrusion detection system should be the real time detection capability. However, typical HIDS techniques like periodic integrity checking and log analysis cause undesirable delays in detection and slow down the system.

b. *Ground truth factor* - An attacker can easily feed wrong information to a user level HIDS if he takes control of the underlying OS. An attacker who succeeds to get the administrator access can easily configure any aspect of the system including kernel and programmable hardware. Systems logs can also be altered to hide attacker's activity.

c. *Network factor* - These days, most applications are hosted from a centralized location to the entire network and hence become a desirable attack vector for hosts. This in view, most efforts to protect hosts are made at network level by securing these network-based applications.

d. *Deployment factor* - Finding a standard intrusion detection tool for an entire network of heterogeneous hosts is very difficult. Therefore, it is thought to be attackslogical to deploy a network level intrusion detection system.

e. *Single system attack limitation* - Detecting some attacks is only possible by observing multiple hosts and collating the information for observing pattern of attack i.e. horizontal port scans). As HIDS are deployed only on one system, therefore detecting this kind of attack is not possible. Network IDS are better suited for this purpose but HIDS deployed in collaborative form can sometimes also detect this attack.

2.5.2 **Arguments to support development of Host IDS**

Authors in [3] also provide arguments to support development and research in area of host intrusion detection systems, which are as follows: -

a. *Network traffic analysis and interpretation* - if the ground truth issue can be solved than the network information generated at host level is more integral

and authentic than network level data and traffic. This more authentic information can lead to an accurate analysis of any malicious activity happening within or originating from the target system. HIDS interpret data packets in same way as system applications. Techniques like data obfuscation, encryption is not possible against a host IDS.

b. *Semantic data* - Alerts in HIDS can be very precise as the detection engine is located very close to the source of malicious activity by the operating system or application. It can precisely generate the exact name of malicious process or user.

c. *Best effort factor* - In some cases, a detected attack may not be published by network security administrator to protect against and in such a case, HIDS can provide minimal level of system protection.

2.5.3 **Data sources used in HIDS**

Authors in [5] have produced a valuable survey on classification of HIDS as per data sources. Various kinds of data sources, which can be used by HIDS for detection may include following: -

a. *System audit information and logs* - Audit data consists of files which are produced because of activity by a user or an application i.e. successful or failed authentication logs, system calls and user command logs. Similarly, log files produced because of system operation are known as system logs. These records save detailed sequence of events in an OS. The data is then analyzed for traces of intrusion. A log file produced by OS usually contains events happening at system level i.e. warnings, errors and system failures. Data at both system and application level records sequence of events of systems and applications and provide valuable information about a possible intrusion.

b. *System Calls* - This is the primary source of information on any activity within the kernel. System calls invoked by a process can give valuable information about the authenticity of the process. The system call artifact is a produced at OS level and hence very close to the source. Popular system calls include open, close, write, read, wait exit etc. However, analyzing system calls for detection have performance overhead.

c. *Windows Registry* - For windows systems, a registry holds data on configuration settings on both software and hardware components in form of key value pair. All processes in a windows system use registry. Therefore, it is also prime target for malwares to hide their tracks.

d. *File System* - File system in a computer has access to executable files, file Meta data and stored data which is used to support file system operations and requests. Often, malware seeks to add malicious files or modify old files or file's metadata to achieve their intents. File system is an effective way to monitor files and their integrity. File integrity IDS are separated from OS and thus harder to turn off. Their drawback is limited visibility of the file system.

Chapter 3

FILE INTEGRITY CHECKING IDS

3.1 Introduction

This chapter will focus around the study of File integrity checking IDSs. As stated in section 2.5.3, file system is a useful type of Host intrusion systems. However, it may not be an area of active research. The reasons for less research and arguments to perform greater research on Host intrusion detection systems are already given at section 2.5.1 and 2.5.2 respectively. As our research deals with security of HIDS Databases, we have selected file integrity HIDS for our research. The reasons to select file integrity HIDS against other types include: -

a. Simpler design requirements which can aid in focused research on integrity protection aspect.

b. Requires no training dataset or period as the DB must be populated from the file system.

c. No external data is required to update the DB i.e. malware signatures.

File integrity checking may not be a glamorous research area but nevertheless important. We will discuss some additional literature on importance of file integrity IDS before going into more detail about DB protection etc.

3.1.1 Importance of File integrity IDS

File integrity IDS have been considered as an essential part of information security best practices. Almost all security standards promote use of integrity checkers, but PCI DSS is the most popular of them all. PCI DSS recommends using file integrity monitoring in its clause no 11.5 to protect against unauthorized modifications of critical files. There is a constant debate that an anti-virus can perform the same functions of file integrity checker however, the argument is not that simple. Although an anti-virus is quite effective in removing a malware on introduction to a system but only when a valid signature or a profile is available. In contrast, a file integrity checker doesn't need a profile or signature DB. Instead, it can call upon its indigenous integrity DB to detect any change to file system. Hence file integrity IDS is better to detect zero-day malwares.

Using a file integrity checker also implements the concept of defense in depth. The authors point out that if a machine without an integrity checker is compromised, then a careful intruder may never be detected due to latest hiding techniques. However, some files will be changed in such a case and can easily be detected by a file integrity IDS.

3.1.2 **HIDS Functionality and issues**

Authors in [6], List down HIDS functionality into three broad categories. These include perimeter control, host access control and Change control mechanism. The authors further note that all these functions can be easily be bypassed by intruders given certain conditions. In solutions to these attacks, the authors recommend using file integrity IDS ensure integrity of the attacked system. They also refer such systems as ***Change Auditing Systems***.

3.1.3 **Fundamental Qualities of File Integrity IDS**

Authors in [7] have carried out a very detailed survey on types of File integrity monitors in the market. Before surveying the types, they also give a list of fundamental qualities in a FIM inspired by work in [8], which are: -

a. *Checks on Meta information* - File integrity monitors must also check the Meta data of a file in addition to file data as a trusted non-super user executable may fall in the category of a non-trusted executable for the root or super user.

b. *Automated process* - The system must not depend on user input to initiate response action, or significant damage space is given to the malicious activity to occur.

c. *Self-Protected* - If system files are being modified by an attacker then it means that he has gained escalated privileges on the machine. In that case, modifying the IDS DB is trivial task. Therefore, mechanisms to maintain the IDS DB must be in place to ensure secure detection of malicious activity.

d. *Relevance* - With automation comes, great deal of output info. This info is later used by users to analyze attacks and reach other important decisions. The more the quantum of output, the more the chance of mistakes on part of human user. Therefore, the output must only be relevant for the human user and in a convenient form.

e. *Repetitive Checking* - Most File Integrity IDSs are periodic IDS, which leave an opportunity gap for the attacker to modify files and take actions to hide himself. Such systems are less effective than systems which are real time and can handle access requests based on security policy.

f. *Upgradeable Systems* - A system is upgraded with new updates to secure against new vulnerabilities. However, if the related File IDS is not upgraded as well with new file integrity DB, even trusted executables will not be executed.

Therefore, a file integrity checker must be upgradeable with little or no human intervention.

Authors in [9] differentiate between 'weak' and 'strong' forms of intrusion. When an intrusion can colonize a system, then a strong intrusion has taken place and on the contrary, a weak intrusion is the one when the intruder is unable to completely take over the system. A good file checker with above qualities continuously reduces a strong intrusion to the level of weak intrusion.

3.2 Types of File Integrity IDS

While discussing various File integrity HIDS, we will use the language of Linux OS to describe the functions and level of operations. Linux is open standard and be easily modified, therefore a rewarding discussion can be generated to move towards a perfect file integrity monitor.

Authors in [7] have carried out a good survey of File integrity HIDS types as stated earlier. The Integrity HIDS have been segregated with respect to the location of File integrity checker operation. Following types have been defined: -

3.2.1 User-space Checkers

Such systems operating at user space are much more vulnerable the systems at kernel space as these are more prone to modifications by malicious intruders. These rely heavy on the safety mechanisms of the system they are protecting. Therefore, valid security issues exist for such systems. Furthermore, these systems must carry out integrity check periodically as these can't intercept calls at lower levels of OS. Also, this kind of periodic checking has a performance toll on the system on every check. However, they are much easier to implement due to libraries and resources available at this level and operate just like an application. Most popular FIM at user level is Tripwire [10]. Other examples are YAFIC, AFICK, and AIDE etc.

3.2.2 Nonresident Checkers

Such systems are located at a separate system and check integrity of one or more systems at other locations. These systems are secure in the sense that hosts have no method to intrude into the server or remote IDS system. However, there are few problems associated with it like actions to be done by the host if the contact between hosts and server is severed or period between checks provide opportunity gaps to intruders and nevertheless securing the client program from compromise. Popular nonresident checker is Osiris but other examples are Radmind, Veracity and Samhain.

3.2.3 Kernel Space checkers

These systems are more complex and very difficult to develop. They also have significant performance overhead if not properly designed. However, such systems are best suited for integrity checking than any other type. Since the system is located at the lowest level of OS, therefore it is not entirely dependent on system protections. It can provide real time integrity monitoring as all file requests must process through the kernel. Furthermore, it can allow or deny access to file depending upon the security policy. Examples are I3FS, DigSig, WLF, SOFFIC umbrella etc.

3.2.4 **Kernel and User space hybrid checkers**

Such systems check files in kernel space and user space as well. One example of such a system is available as described in [11]. A hybrid checker can have advantages of both types of file integrity checkers i.e. checking files in different executable formats like ELF etc. However, it is difficult to maintain an updated DB to check all formats. Furthermore, security of communications between kernel and user space can be compromised.

3.3 **Attacks against File Integrity HIDS**

Authors in [12] have given very precise challenges for File integrity checkers and how to counter these. Detailed challenges to HIDS security have also been given but here we will focus only on file integrity HIDS. The authors have segregated the file integrity HIDS in three modes i.e. initialize mode, check mode and update mode. Attacks have been given against each as follows: -

a. *Attacks against Initialize mode* - If a system is already compromised then in initialize mode a compromised DB will be created and used in subsequent phases of operation. The counter measure could be to run the HIDs right after installing the OS and in offline mode to ensure absence of any compromised system component.

b. *Attacks against Update mode* - In base level integrity checking programs, an attacker can run the update mode on modified system or simply modify the DB. But this can occur if DB is stored in the same system. The solution to this problem may be to protect the update mode with a password, encrypt DB or ideally the DB can be stored at a remote location. This can protect against insider and physical attack.

c. *Attack against Check mode* - Attacks in this mode are found in many forms. Some are listed here: -

i. *Trojan Binary* - An intruder can replace the existing standard system binary to check integrity with a malicious one to report correct results of certain files. Counter measure is using own hash checking binary.

ii. *Loadable Kernel module* - An intruder can load his own LKM to hijack system calls and direct them to open a malicious program stored elsewhere. It can be countered by implementing more checks, which can cause to attacker to remap greater number of system calls which is not trivial.

iii. *Faked report* - Attacker can hijack the reporting channel between the checking module and the reporting module and feed a fake report. Furthermore, he can also replace the system binary with an ok report generator. This can be stopped by using secure channel between modules or checking the system binary for integrity as well.

iv. *Checking gap attack* - An astute attacker can quickly do his work periodic checking IDS and replace the malicious file with the original after completion of work. It is not logical to perform periodic checks in short intervals therefore, this is likely avenue of attack. Counter this attack by using kernel level checkers to intercept the file access request and check integrity on call. Also, daemon checker like samhain also exist, which keep on running in back ground.

3.4 **Database protection**

As we have seen so far that location of HIDS itself, HIDS DB and DB integrity are of great importance for the wellbeing and correct operation of the system. Author in [7] propose to place the integrity checker at kernel level and check selective files for efficiency. For DB security, it gives few options like storing the DB offline on external write only media or on the hard disk. External media is safer but complex in terms of updates whereas later is vice versa. Authors in [13] deal with this subject of DB protection more elaborately. They define three techniques to protect the DB integrity, which are as follows: -

a. *Read only Media* - Storing the DB on a write protected CD or floppy can force the user to have physical possession of the DB media for operation. It is useful in computers with very less changes in files DB, however this technique is very tedious when updating in highly frequent changing environment.

b. *Use of Signature* - The file information DB can be protected by using hash function or a digital signature to ensure integrity of the DB. This is more efficient with respect to file updates, but it increases system complexity due to encryption overhead.

c. *Remote DB Storage* - The DB can be stored on a remote server and accessed via a secure channel. This also provides operational advantages since all hosts

can be updated for policy from a single server and protected. However, it still provides a single point of failure if the server is compromised itself.

3.5 Using Blockchain for DB protection

After going through the above discussion, it is quite clear that none of the solution to protect DB is flawless and has some compromises. In this research we recommend the novel approach of using Blockchain for ensuring DB integrity in File integrity checking HIDS. We will discuss DB security in File integrity checking HIDS as a test case, however we think that this technology can be implemented for all HIDS or NIDS which maintain some sort of DB for detection. To confirm our approach, we will use the method given by [14], to know the exact type of blockchain solution suitable for our Host based integrity checkers.

Detailed discussion on Blockchain will follow in the next chapter. However, we will use this space to find out the suitable type of blockchain for our research. Authors give the result through a flowchart which is reproduced herein: -

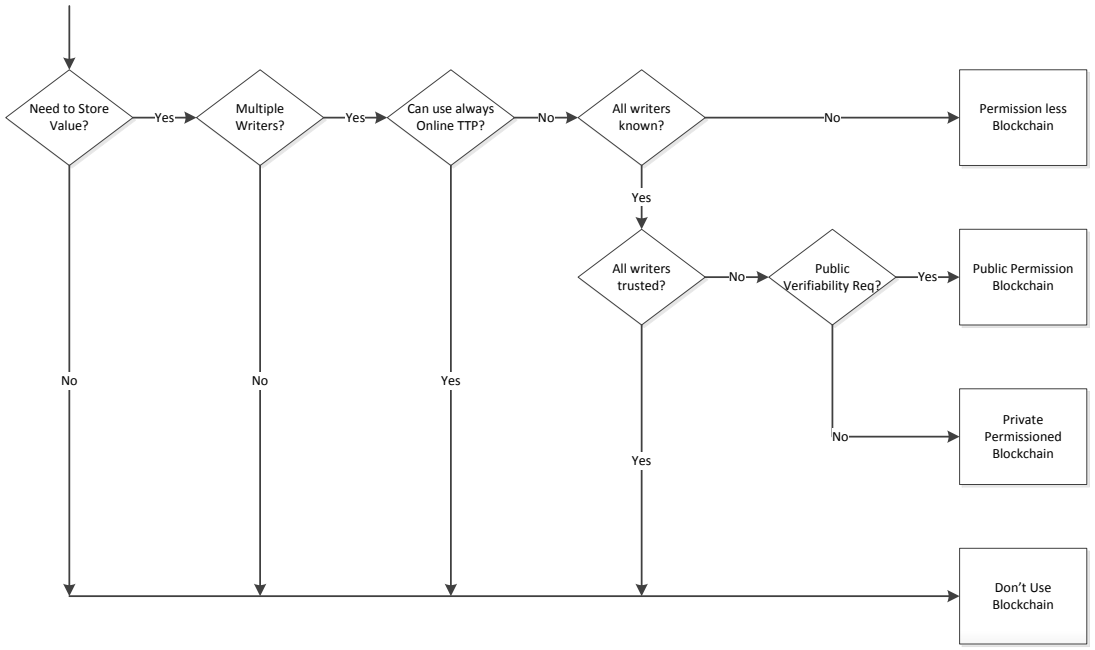

Figure 3.1 Selection Method for Blockchain [14]

Most people take blockchain as a storage technology, whereas it is infact the worst DB in the world. The prime advantage of using blockchain is ***Immutability*** and ***Decentralization***. We will delve in this detail in the next chapter. As we can see from the above shown flowchart, we can use blockchain for HIDS DB integrity protection. For the initial test case, we will use ***Private Permissioned Blockchain*** for our discussion in later chapters to avoid complexity of

online blockchain presence. A comparison of Permissioned, Permission less blockchains and a Central DB is given below to put technical aspects in perspective.

	Permission less Blockchains	***Permissioned Blockchains***	***Centralized Database***
Throughput	Low	High	Very High
Latency	Slow	Medium	Fast
Readers	High	High	High
Writers	High	Low	High
Un-trusted writers	High	Low	0
Consensus protocol	PoW, some PoS	BFT protocols [15]	Nil
Central Management	No	Yes	Yes

Table 3.1 Comparison of Blockchain and Central DB [14]

So, we can conclude that blockchain at present suffer from low throughput to match high transaction environments. However, it is worthwhile to experiment with this technology for securing the HIDS DBs. We will take our discussion on Blockchains further in the next chapter.

Chapter 4

BLOCKCHAIN INTRODUCTION AND LITERATURE REVIEW

4.1 Introduction

The internet revolution in recent times has brought about a paradigm shift in how man and machines communicate with each other. Digital transactions, may it be for business or educations etc. are being carried out through the internet with an ever increasing rate. Digital transactions or communications in coming time is expected to take a new shape with the full blown advent of IoT, where huge number of intelligent devices will swarm the digital landscape. Even today, digital communication has reached a well developed stage but requires a central agency or authority for maintaining trust in the system. Electronic payments and e-commerce are obvious examples of such communications with central authorities maintaining trust. However, the central agency can be easily compromised. Further, Satoshi [16] maintains that in present financial models, the cost of mediation by a central financial institution to establish trust inevitably increases transaction costs and some amount of fraud between participants is taken as unavoidable. The cost and required trust in the central institution can be reduced or eliminated by using a true de-centralized trust mechanism i.e. trust distributed among the participants. Same is true for non-financial sectors as well.

In [17], the author points out that endeavors to achieve a truly de-centralized currency were started as early as 1980s but every model proposed required a central agency as in [18], [19] .Then models like Karma [20], B-money [21] and Bitgold [22] used concept of a cryptographic puzzle to generate digital currency and replace a bank but still required some central involvement to maintain possession records. The first truly de-centralized system was proposed by Satoshi Nakamoto [16] and announced on Cryptography mailing list [23] in 2008. Satoshi combines research in recent times in a novel way to achieve a de-centralized trust environment in Bitcoin system.

The Bitcoin uses as a core element, *Blockchain technology* to maintain trust in the system, by validating records and keeping track of all digital transactions in a distributed manner. Bitcoin can be called the first implementation of *Blockchain technology* and understanding Bitcoin is essential to understand Blockchain. In [24], authors explain that although Bitcoin enables multibillion dollar global market of transactions, it is still the most controversial application of Blockchain Technology due to lack of governmental control and oversight. However, the underlying technology of Blockchain is not contentious and has operated perfectly to achieve a de-centralized trust model in Bitcoin. But Blockchain is much

more than Bitcoin. For the first time in digital history, there is an understanding among thinkers that online P2P transactions or communications may be possible without a central institution. These may include online contracts, digital records keeping, email services, intelligent device identity and privacy protection, hence it can be applied to both financial and non-financial applications and the opportunities are limitless. Authors in [24] mention Marc Andreessen, a Silicon Valley capitalist, listing blockchain distributed trust model as the most important development after Internet. They also mention Johann Palychata, BNP Paribas, who equates invention of Blockchain to that of a combustion engine in Quintessence magazine and hints that it has the potential to transform the digital world. Briefly speaking, [24] a blockchain is a distributed ledger available publicly of all digital transactions shared among participating parties. Every entry in the ledger is confirmed by a majority consensus mechanism among participants. Information confirmed and introduced in the chain cannot be erased. So, in a nutshell, a Blockchain contains a certifiable copy of every single transaction ever made in the system. One of the most dynamic applications of Blockchain technology is deemed to "smart contracts", computer codes which can execute the contract terms themselves without mediating oversight. These smart contracts can be applied in the field of notary, asset transactions involving properties, cars etc. and ownership of money and shares.

As authors in [25] mention, despite its capability to maintain de-centralized trust in a system, Blockchain technology, at present faces multitude of challenges in mass scale adoption. Due to its envisaged role in future internet based transactions, these challenges along with implementation scenarios merit further study to pave way for embracing Blockchain at a large scale. In this paper, a holistic view of Blockchain technology is intended to be taken resembling a survey by studying its challenges and areas of application. The inspiration for research methodology comes from [25], in which a systematic mapping process [26] is used to study and recognize pertinent publications in the Blockchain domain. This systematic research methodology uses a structured approach, where papers are identified against a criterion. Furthermore, as in [25], focus of research will remain on Blockchain technical and application domains only and will exclude Bitcoin except for technology reference. From here the paper leads to Section 2 which explains Blockchain technology briefly. In Section 3, challenges and advantages of blockchain will be discussed. Section 4 will give research landscape of Blockchain along with discussion on major application areas. Section 5 will indicate the area of future research and concludes the paper.

4.2 **Blockchain technology**

As already stated blockchain is a distributed ledger system where each event or transaction is recorded and can be used later for verification and auditing. A blockchain defines a chain of blocks, where each block holds several events recorded in it. This chain of blocks is formed by linking each new block with a previous block by using a cryptographic hash function. Each transaction in a block is verified and included in a block by nodes known as miners. Further, a block is included in a Blockchain only after a certain mathematical proof is achieved. As mentioned before, Blockchain is intrinsically related to Bitcoin and understanding Bitcoin technology is necessary for understanding Blockchain. Infact, no defined standard for Blockchain components exists yet but we can generally take the Bitcoin technology of distributed trust as *THE BLOCKCHAIN* technology. Now since the trust or record of transaction is recorded in a Blockchain after some computational work is carried out by miners, there must be some incentive for the miners. In Bitcoin, this incentive comes in the shape of getting 12.5 Bitcoins on reaching the mathematical solution of a block first. Hence, a ***block is appended to blockchain*** periodically when a miner reaches the required mathematical solution for a block and Bitcoin ***is generated in the system***. Briefly a Bitcoin Blockchain passes through following processes [16] as depicted in Fig 1: -

a. Events or transactions are broadcast in the network on occurrence.
b. Each node forms a new block by accumulating received transactions in said block.
c. Each node endeavors to reach the mathematical proof of the block or *proof of work* first to receive the incentive described above.
d. On finding the proof, a node broadcasts it's found block to entire Network.
e. The broadcast block is agreed to by other nodes if every transaction in it is valid.
f. If accepted, all nodes try to form new blocks using the hash of this newly accepted block.
g. Hence all the accepted blocks form a blockchain.

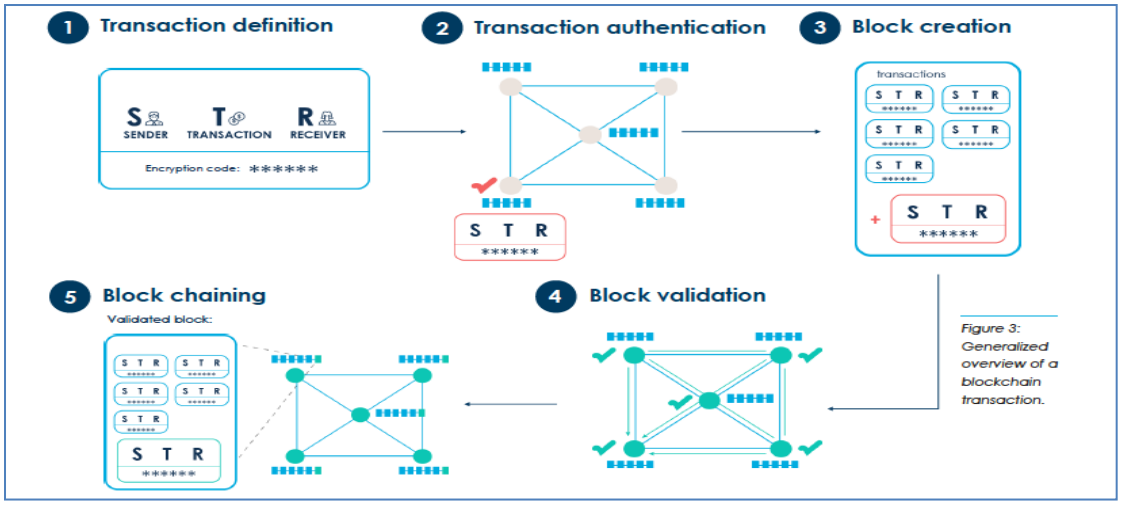

Fig. 4.1 Step by Step Blockchain processing [27]

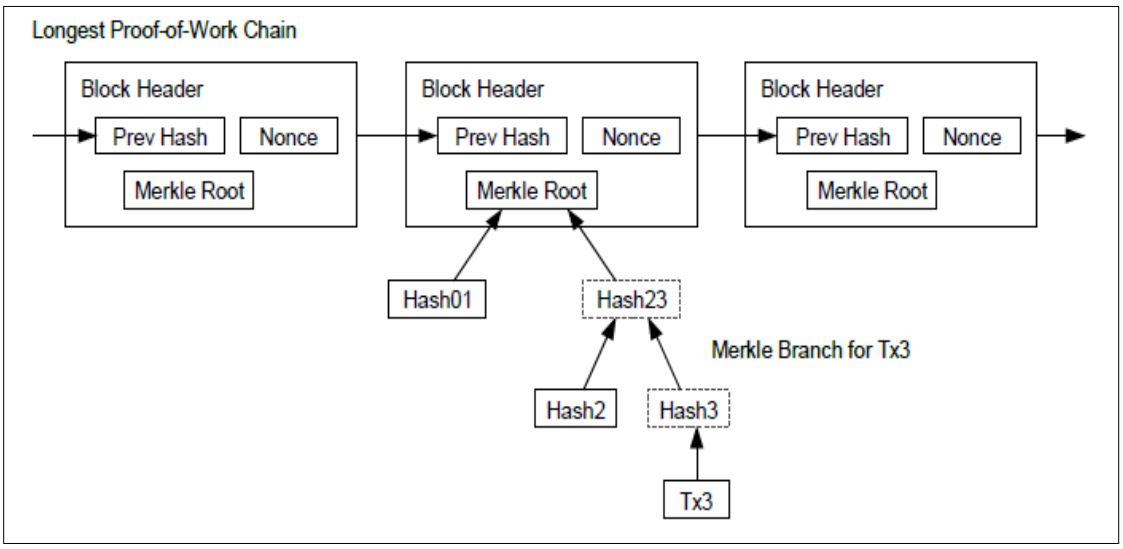

Fig. 4.2 New Block Formation in Blockchain [16]

In Fig 1, formation of a blockchain has been further elaborated. Every new block will be formed with three major inputs: Hash of previous or last validated block, a nonce using which solution to the mathematical proof will be reached and a Merkle root of all the transactions of this new block. When the right nonce is identified, the block with trace of its transactions and the same nonce is broadcast in the network for validation and acceptance by other nodes and this process continues forming a chain.

4.3 Components of Blockchain

Although no specific structure of blockchain has been defined but most researchers agree that a blockchain may be divided into following components [28]: -

a. *User application* - The major component in a blockchain system is the computer application or software which will enable a user to participate in the desired ecosystem. This application will define the rules of the ecosystem and

how users will interact with each other on the chain. Most pertinent example can be Bitcoin wallet, which is the mandatory application software required to carry out trading on a bitcoin blockchain.

b. *Events ledger or Blockchain* - This is the core component of a blockchain system, which is primarily a logical entity. It is populated inside a node as the life of a blockchain system progresses. This is the component which provides the immutability in a blockchain ecosystem and provides the vital functionality in a distributed trust environment. All transactions ever performed can be verified using this historic ledger. Each new application will require a new or separate ledger. A node may run many different applications based on blockchain at any single time, but all will have their own events ledger.

c. *Consensus Mechanism* - The consensus mechanism plays an important role in blockchain ecosystem. This functionality guides the ecosystem to reach a consensus and how to form the chain of blocks. Any conflicts are also resolved with this mechanism. This component of a blockchain system will probably see many mutations in future as the research in blockchain technology progresses. In Bitcoin, this mechanism is sufficient to deal with the quantum of transactions but as we try to introduce blockchain in other system, it is the consensus mechanism that will be modified the most to fit into the desired ecosystem. We might take example of lightening networks, in which two communicating parties after achieving trust through blockchain do not broadcast all transactions in the system and only interact with blockchain on establishing and terminating interaction. Other prominent examples are proof-of-stake, proof-of-work and proof-of-elapsed-time.

d. *Virtual machine* - The last module in blockchain ecosystem is the virtual machine. It is also a logical component which collaborates with user application to participate in a blockchain ecosystem. It is a logical representation of a machine created by the blockchain application. Again, we can see Ethereum as an example. It implements a virtual machine in its blockchain ecosystem. Its virtual machine can understand a wider range of instructions making it possible to manage the status of a digital contract in contrast to Bitcoin, which is not so flexible due to underlying application.

These generic components will always be present in a blockchain ecosystem taking some form dependent on the desired application. However, there will

always be other factors, which will have some bearing on the effective performance of the desired blockchain application. These factors may include the communications network, underlying hardware for the virtual machine etc.

4.4 Classification of Blockchain

As stated earlier, there are no established standards available for the Blockchain yet. However, endeavors are at hand to formulate a standard to guide future work on the technology. International experts from over 30 countries have recently participated in the opening session of ISO TC 307 blockchain and electronic distributed ledger technologies chapter, held in Australia in April 2017.The goal was to discuss standardization in Blockchain Technology. Five groups for development of standards have been formed: taxonomy and ontology, reference architecture, use cases, identity and smart contracts, security and privacy.

Few Blockchain classification schemes have been defined using different perspectives. Many have classified Blockchain on the permission model. Authors of [29] classify blockchain in following permission types: -

a. *Public blockchains* - Such blockchain systems are open to users for active participation. Generally, a public blockchain system allows anyone to write to the blockchain. There can be two aspects to it: Any entity can write to the blockchain on being granted permission by someone or any entity can read from the blockchain on being granted permission by someone.

b. *Private Blockchains* - Such blockchains are held by private entities or groups and do not require establishing trust among themselves: for example, a group of companies owned by a single bigger company. In such a case, the processes will differ widely from a public blockchain. It might not need the consensus mechanism at all to write to the blockchain. The incentive might also take a different form.

Authors in [30] propose blockchain classification based on Authority and Incentive Dimensions. Following classifications have been proposed: -

a. *Blockchains without an authority and market-based* - This type of blockchain is un-regulated and gets the momentum from the market to extract some incentive for miners. The market can help in establishing a price for the Coinbase or the prevalent incentive form. In a blockchain which does not receive regulation from any authority, it is necessary to have an established market.

b. *Blockchains under an authority and market-based* - Such ecosystem can benefit from the market to incentivize their miners or the participants for carryout the block validation work for the system. Further, the ecosystem is regulated in some way by an authority. Authors in [30] suggest that a market for circulation of currency or the Coinbase can be established for a blockchain ecosystem under an authority like systems without an authority such as Bitcoin. The currency will have a value which will be the main incentive for joining the blockchain. However, the characteristics of the ecosystem will be heavily regulated by the authority.

c. *Blockchains under an authority and non-market-based* - In a non-market based blockchain there needs to be a compelling alternative incentive to join and remain in the system. Without a market surrounding it —without a price mechanism— it would be preferable for consensus to be based on some other grounds besides POW or similarly costly algorithms. Rather, it is more sensible to take an approach based on other methods of agreement between the parties that are more energy efficient compared to POW consensus. In this case, the authority can facilitate the implementation of these alternative methods by exerting control over participation in the system.

d. *Blockchains without an authority and non-market based* - Without a market to provide the incentive of price of the currency, and without an authority to enforce and administrate participation in the system through contracts, the long term continuance of a blockchain becomes complicated. Therefore, this last type of blockchains is perhaps difficult to contemplate. In its early years Bitcoin could have fit this classification. However, without the monetary incentive and without the possibility of enforcing regulations, incentivizing participation would remain a challenge, although ideological, social or ethical considerations can play a part as incentives.

In the later classification, we can equate authority to permission and market to incentive. This classification may be useful in understanding the machinations of a blockchain protocol but fundamentally it is like permission model. Incentive of participating in blockchain proof of work will always be there. However, it might take different shapes under varying scenarios i.e. financial incentive or attainment of organizational or ideological aims. As per [31], another type of blockchain follow a hybrid architecture, which allows for different blockchains (public or private) to communicate with each other, enabling transactions between participants across blockchain networks.

4.5 Challenges of Blockchain

4.5.1 Technical Challenges

Blockchain still faces daunting challenges to achieve mass scale adoption. Authors in [25] have carried out some useful research on technical challenges identified by Swan [31] identifying the current research directions in following challenge areas: -

a. *Throughput* - The possible throughput of transactions, a Bitcoin network can handle is 7 tps (transaction per sec). Examples of transactions handling networks are Twitter (5000 tps), VISA (2000 tps). There are still questions about how the blockchain with current architecture will behave once the transactions level will reach the same magnitude as VISA or Twitter. There will most likely be an emergent requirement of increasing the transactions throughput of the desired network.

b. *Latency* - As we have discussed earlier, the proof of work mechanism establishes the authenticity of a block after reaching a mathematical solution to problem. This ensures security in the ecosystem against a double spending attack where a user might be able to spend the same coins twice. A single block takes around 10 minutes in bitcoin network to get validated. As the block validation capability grows, the difficulty of mathematical proof is also adjusted. This is necessary to achieve security in the system. Double spending is a consequence of successfully transferring same coin more than one time [32]. Bitcoin protects itself against such an attack by verifying every transaction sent to the blockchain. All this comes at the price of latency in the ecosystem. Validating a block and confirming the contained transactions should be less time consuming while keeping the required security intact. If we look at contemporary systems, VISA network requires only few seconds, which is a great advantage as compared to Bitcoin or Blockchain.

c. *Bandwidth and Size* - As of December 2016, size of Bitcoin blockchain is 100GB. Imagine the scenario when the transaction rate will increase from 7 tps to that like VISA or Twitter, the blockchain is expected to grow at an enormous rate of 214 PB every year. The Bitcoin and blockchain community suggests that the size of each bitcoin block is 1 MB, and new block is formed every ten minutes [33]. Therefore, each block is limited to handle a prescribed no of transactions (average five hundred transactions) [34]. This scalability issue of blockchain must be resolved if wide spread adoption of blockchain is to be undertaken.

d. *Security* - Blockchain faces few security challenges as well. These include the 51% attack, where any party controlling most of mining capability can control the ecosystem and can validate the blocks. Another type of attack is block withholding attack where a selfish miner can create a longer blockchain to race the main blockchain, creating an opportunity for manipulating the main blockchain. Such issues also require due research for adoption of blockchain in other fields.

e. *Wasted resources* - It is estimated that Bitcoin mining wastes USD 15 million worth of energy daily due to its Proof of Work protocol. Proof of Stake protocol is being promoted as a useful alternative [35]. This issue of resource wastage must be solved before efficient mining of Bitcoin can be carried out further.

4.5.2 Non-technical challenges

There are also some non - technical challenges hindering mass adoption of Blockchain. Some of the non-technical challenges identified by [36] are as follows: -

a. *Uncertain regulatory status* - A big advantage of blockchain bitcoin is its un-regulated status but at the same time, it is probably its biggest hurdle in widespread adoption. National institutions are still skeptical about how a blockchain ecosystem will turn out without any regulation and monitoring.

b. *Energy utilization* - The bitcoin mining process consumes enormous amount of energy on annual basis. The miners attempt approximately four hundred and fifty thousand trillion solutions to the proof of work problem each second to validate event or transaction, expending substantial computer power. A more efficient validation mechanism can reduce this enormous energy consumption.

c. *Integration concerns* - The lack of any blockchain standard leads to integration issues with present systems. To move a business or process on blockchain, complete or significant changes or replacement to existing systems are required. Formation of a blockchain standard is therefore, needed to be expedited.

d. *Cultural adoption* - Blockchain presents a complete departure from a centralized and regulated enterprise. This will require quite some motivation on part of users and operators to start adopting blockchain for their respective operations.

e. *Cost* - Where blockchain offers great savings on transaction costs by eliminating the middle man, it does incur great initial expense for adoption of a Blockchain ecosystem. This could be a deterrent for the time being as users or enterprises might not immediately be interested in expending so much capital without any visible motivation. [37]

4.5.3 Advantages

Authors in [37] have summarized the advantages of blockchain as following: -

a. *Trustless and Decentralized exchange* - Parties interact or transact without any regulator or middle man, hence completely removing the third party.

b. *User Control* - Users control all their data or transactions. Data privacy is achieved.

c. *Quality of data* - Data in blockchain is timely, accurate, consistent, widely available and complete.

d. *Durability, longevity and Reliability* - No single or central point of failure exists in a blockchain ecosystem. Danger of malicious attack altering the information is greatly reduced.

e. *Process Reliability* - Users remain assured that their trust in the system remains intact and transactions are executed as protocol dictates.

f. *Immutability* - Any change in public blockchains can be viewed by all users providing complete transparency, and transactions are indisputable due to intrinsic cryptographic functions, meaning these cannot change or be deleted.

g. *Faster transactions* - Presently, Bitcoin blockchain can transfer any capital in little over an hour whereas, interbank transactions take few days to clear and finalize a single settlement. This provides a fundamental advantage to blockchain in financial sector as the long transfer channel is significantly shortened bringing down the time consumed.

h. *Lower transaction costs* - By eliminating third party or the bank, transfer costs or overheads are reduced significantly for exchanging holdings or assets.

4.6 Research Landscape

Research on blockchain has really taken off in the last three years. Bitcoin was the area of primary research on blockchain in the early years i.e. since 2008. This new-found interest in blockchain has given rise to an increased number of publications in the last few years. Below we discuss few popular areas of active research in Blockchain: -

a. *Internet of Things* - It is one of most researched areas of present era. It has enormous potential to revolutionize the digital world. In [38], four main

challenges to IoT facing today are: Security, Privacy, Connectivity and Compatibility. A blockchain can be an effective method to address security and privacy concerns in IoT in particular. In [39], IBM has identified IoT for Blockchain implementation trends in 2017. Most of research publications analyzed are related to IoT security and privacy. Some other aspects of IoT have also been touched but these are very limited. Blockchain in its classical form i.e. bitcoin is resource intensive itself and cannot be used for IoT. In [40], the authors have attempted to comprehend whether the blockchain and P2P mechanisms can be used to build a decentralized IoT. As a first step, they embarked upon a Literature Review to accumulate data on the present implementation scenarios of blockchain technology and observe its current level of adaptability, integrity and anonymity. They discovered eighteen scenarios of blockchain out of which four were exclusively related to IoT. They also observed few issues in the anonymity, integrity and adaptability of a blockchain. They found that blockchain just ensures pseudonymity.

b. *Adaptability of Blockchain* - Authors in [16] have proposed a lightweight blockchain design for IoT which practically eliminates the overhead of a traditional Blockchain, while keeping its security and protection benefits. IoT gadgets use a private ledger which works like a Blockchain however it is centrally managed to reduce energy expenditure. High resource entities make an overlay network to execute a permission less or public Blockchain that ensures end to end protection and security. The design utilizes decentralized trust to reduce time required for block verification. Analysis of a smart home use case has been carried out to assess the proposed framework. Results show that processing overhead is reduced significantly when compared with a traditional Blockchain used in Bitcoin.

In [42], same authors have continued their work in [41] and outline the core elements and their functionalities in a smart home environment. Every smart home will have an always alive device with greater resources, called a miner which will act as a gateway to handle all traffic in and out of smart home. This miner also maintains a private blockchain, which will aid in controlling and auditing all traffic. The authors also conclude that their proposed blockchain smart home is safe, by analyzing the performance considering fundamental goals of confidentiality, availability and integrity. They also presented results

to conclude that overhead in the proposed framework, are insignificant as compared to its achieved privacy and security gains.

In [43], author has recommended a framework that securely integrates the blockchain with smart IoT devices in smart city to provide a secure communication.

Further in [44], authors describe a user-centric framework to share the data from IoT devices with other users and enterprises. This approach uses access lists, access rights and capabilities. They describe an auditable, decentralized, transparent, distributed mechanism and their automation in the IoT scenario. Using blockchain, it makes users in control of their sensors' data and chooses whoever accesses it. The storage is again based on blockchain technology with access layer based on publish and subscribe methodology.

c. *Secure updates and firmware* - Broadcasting updates and firmware securely are another potential application area for blockchain in IoT. With the distributed consensus mechanism in Blockchain, we can be sure of using the authentic copy the desired update or firmware. In [45], authors have investigated how a blockchain can securely deploy updates for IoT devices. They observed that a blockchain improves availability of updates significantly due to its immutability and intrinsic protection against DoS.

Another scheme to update firmware based on blockchain technology is proposed in [46]. It observes the firmware's version, verify the firmware's integrity, and retrieves the latest firmware version for its devices. The design encompasses asking collaborating nodes for latest firmware version. If own version is modified, then latest version is downloaded from nodes by P2P file transfer mechanism.

In [47], authors again propose a blockchain based framework for configuration of IoT devices. They have used a blockchain platform for smart contracts, Ethereum, to manage keys in RSA public key cryptosystems. In this design public keys are stored in Ethereum platform and private keys are stored in respective devices. Using smart contracts, IoT devices' configuration is easily managed and a key management system is built. They have used a few IoT devices instead of a full system for proof of concept of their proposed design.

d. *Data privacy in IoT* - Connected devices in IoT, will exchange highly insightful data about a person or entity. Routing such data through regulatory

companies presents a serious risk to user's privacy. Economic interests could result in undesirable use of personal information. Authors in [48] have proposed to develop software systems for IoT, which are decentralized private-by-design. The basic idea is to safely store sensitive data produced by numerous IoT sensors or devices in a distributed system. Such a system is designed to ensure privacy of data, thus making people the real owner of their data. To achieve this goal, they propose a P2P storage network in combination with a blockchain.

Another approach has been proposed in [49], in which an IoT device is registered in a cloud eco-system. The device proves its OEM origin without any third party's involvement. The design introduces the *Chain anchor* architecture which supports this mechanism for registration.

Another angle to P2P communications is in e-commerce using Blockchain [50]. In this article, the authors: 1) propose a framework for IoT based e-businesses, 2) traditional e-business elements are redesigned, 3) Use blockchain to evaluate property transactions in IoT.

e. *Secure service sharing* - In [51], authors discuss how sharing services based on blockchain can add to development of smart cities using a theoretical design. A similar proposal is presented in [52] which forwards the idea of using virtual resources with permissioned blockchain to use IoT services.

f. *Device authentication* - Authors in [53] have embarked on a novel approach to ensure trust worthiness of Devices and Data. Verifiers use remote attestation to see if a Trusted Computing Base (TCB) under study is trustworthy. However, the prevalent remote attestation has cannot be deployed in IoT due to limitations. To address such limitations, authors have presented a trustworthy administration of TCB measurements known as TM-Coin leveraging the blockchain. The TM-Coin use blockchain technology for TCB measurements of IoT elements. Using TM-Coin, data attestation by verifiers for IoT devices is carried out remotely using the blockchain without attesting to the TCB.

g. *Blockchain as a service* - Due to its distributed and decentralized organization, researchers are contemplating using blockchain in IoT for device configuration, storage of data from sensors and to support payments at micro level. In [54], the authors evaluate cloud and fog platforms as hosting

environments. The performance analysis of both systems is carried out and clearly indicates that network latency plays an important part when both used for hosting the blockchain. Finally, the fog platform outperforms the cloud.

h. *Other Areas of Blockchain research* - Extensive research is being carried out in the areas of Healthcare, Governance, Smart contracts, Social Networks, Supply chain, Cloud computing, DNS and DHCP, Smart Energy, Big Data, Asset and ID management and Critical Information Infrastructure Protection.

4.7 Blockchain study for DB protection

As stated in chapter 2, our aim is to secure HIDS by using blockchain for DB security. After careful consideration, we have shortlisted following three areas of blockchain research to achieve our aim: -

a. *Blockchain Storage* - The purpose to study blockchain based storage is to gauge its suitability for storing HIDS DBs. We have observed that storage for blockchain based applications is one of the biggest hindrances for mass scale blockchain adoption. However, many startups are working on the blockchain storage issue and we may find many good projects available for blockchain based storage in next couple of years. Presently, we have concluded that storing entire DB on blockchain may be a farfetched idea and only some values can be stored on few available platforms like Ethereum. Let's discuss some of the good projects we have studied for storage. Most notable is Bluezelle [55]and their aim is to develop a blockchain system to store data for decentralized blockchain applications. They propose making swarms of nodes to store data. These swarms join to form a meta swarm. Data is distributed among various swarms and replicated within a swarm. Downloading of data is done from nearest swarm. They have proposed the use of SHARDING for efficient and failsafe data storage and retrieval. Another good storage project is Phantasma [56]. Basically, it is decentralized content sharing platform to enable communications and data exchange between decentralized applications. It uses concept of relay nodes to send data across the network. Resource intensive tasks like data transfer is done off chain while only locations are stored on chain.

At present, no storage application is ready for client usage and is in development phase.

b. *Blockchain Antivirus* - We studied this category of papers to find a solution for secure updates in a system. As with blockchain storage, these systems are

mere proposal and in development or trial stage. However, we found Polyswarm [57] to be promising. It aims to provide a decentralized market place for security analysis and malware detection. It combines clients with geographically diverse detection engines through detection offers. It also supports a system of reputation to assess the result and build confidence vote for the correct detector. Another proposed system is [58]. It is more pertinent to our desired system. An update from one system is broadcasted to all systems in a network for individual analysis. The result is compiled, and a consolidated decision based on majority vote is reached. We will consider this system in our design to securely update the OS. Rest of these papers proposed use of blockchain for collaborative detection, where a threat detected by one system will be communicated to all joined systems through blockchain.

c. *Blockchain Scalability* - Along with storage, Scalability is also one of the biggest obstacle in blockchain mass adoption. Present Blockchain platform do not support large transaction rates, which can cause slow response of applications. Some noteworthy projects targeting greater transactions rates are Quarkchain and Zilliqa. Quarkchain [59] is trying to achieve a transaction throughput of 100,000 tps. Transactions are distributed in side chains for confirmation and root chain is used for confirmation of blocks from side chains. There are no transactions in root chain. Zilliqa [60] is quite like Quarkchain and aims to increase the transaction rate. Again, transactions are divided among miners to achieve efficiency and faster transaction rates.

In the next chapter, we will propose an IDS design to achieve our objectives as stated in chapter 1.

Chapter 5

PROPOSED IDS DESIGN

5.1 Introduction

In previous chapters, we discussed that IDSs based on File integrity monitoring make use of a distinct fingerprint of protected files calculated from a file's contents [61]. This digital fingerprint is also referred to as file checksum or hash value. This unique file information is recorded in a database and retrieved only to validate the protected file's integrity against change or modification. Hashing or checksum generation algorithms have very important place in such an IDS. These algorithms must be resistant to collision generation and pre-image attacks, failing which integrity of the entire system can be jeopardized. Well known algorithms with fixed length outputs are MD5 and SHA etc. [62].

Any intruder, who has already gained entry in a protected system, can modify system utilities, modify file attributes or contents or install backdoors etc. Such attacks can be easily detected by an integrity checking IDS and thus significantly enhance system security. However, all such systems may be bypassed by modifying the file integrity database or system binary. This vulnerability exists not only in user level programs but also in kernel level programs. Generally, file restoration mechanism if an infected file is found, is also not found in majority of programs baring some research projects. Furthermore, file and operating systems updates hinder operational response of these systems.

In this chapter, we recommend a novel design for a File integrity-based IDS to overcome above mentioned weaknesses in existing systems. Following broad contours for the proposed system have been set to carry out further research: -

a. Proposed system should provide real time detection through implementation at kernel level.
b. Backup of protected critical files should be maintained in protected storage or off the system.
c. System should be able to restore critical protected files from backup if modification is detected.
d. System integrity is checked against modification by using tamper proof record of system fingerprint stored on Blockchain.
e. Occasional file updates be checked against presence of malware through collaborative Blockchain detection mechanism.

We propose to implement our system at user and kernel level to ensure enhanced system security as compared to similar systems implemented at user level only. Most well-known of such IDSs is tripwire, which is a user level system. User level systems have following disadvantages: -

a. Any intruder can tamper such a system by operating at a lower level of system.
b. Serious degradation issues when such systems carryout periodic checks and interfere with systems performance.
c. These systems are not real time thus providing sufficient space to intruder to perform their activity.

These issues are appropriately catered for by implementing the system at kernel level. The kernel checks against security constraints during system operations like file access. Normal kernel operation can be enhanced and provided with a mechanism to check and validate file digital finger prints or checksums before access is granted. There are various ways to achieve kernel extension as described above. Few of these are as follows: -

a. *System calls* - Normal calls are modified or changed with new custom system calls ensuring secure access. However, this approach has few weaknesses [63]. It has race conditions, may also need code duplication and has limited security context expression.
b. *Stackable file system* - It is an efficient approach to implement secure file access in kernel space. An additional layer is added between an existing file system and virtual file system for security. It intercepts file system calls from virtual file system and passes them on to actual file system at a lower level. However, in this case, the file integrity checking module and kernel is separated and hence may be fed malicious data.
c. *Linux Security Modules (LSM)* - LSM [63] is a Linux framework designed to provide access control to kernel objects. It is executed as a loadable kernel module (LKM) and uses hooks to intercept system calls for access. These hooks call Linux security functions to validate or invalidate the access request to an object. It is used in SELinux [64] and Apparmor.

We suggest using LSM as it provides a well-constructed API providing easy integration of modules with Linux kernel for security functions. It also ensures that any hooked module cannot be removed by the intruder as the extension module is hooked to the Linux kernel.

5.2 Background and related work

5.2.1 Typical File Integrity Checking IDS

Figure 5.1 shows operation of a typical Integrity checking IDS. Most of these systems follow same basic steps to achieve the goals. Common steps in file integrity checking IDSs are as follows: -

a. Scan configuration data to check about file protection information.

b. Calculate and store digital fingerprints of protected files and store in a database.

c. To check file integrity, calculate integrity of accessed file again and compare against value stored in database.

d. Grant/ Deny access to file or initiate appropriate action based on comparison result. Actions may include alerting the administrator, generating a report, logging the action or shutting the system down.

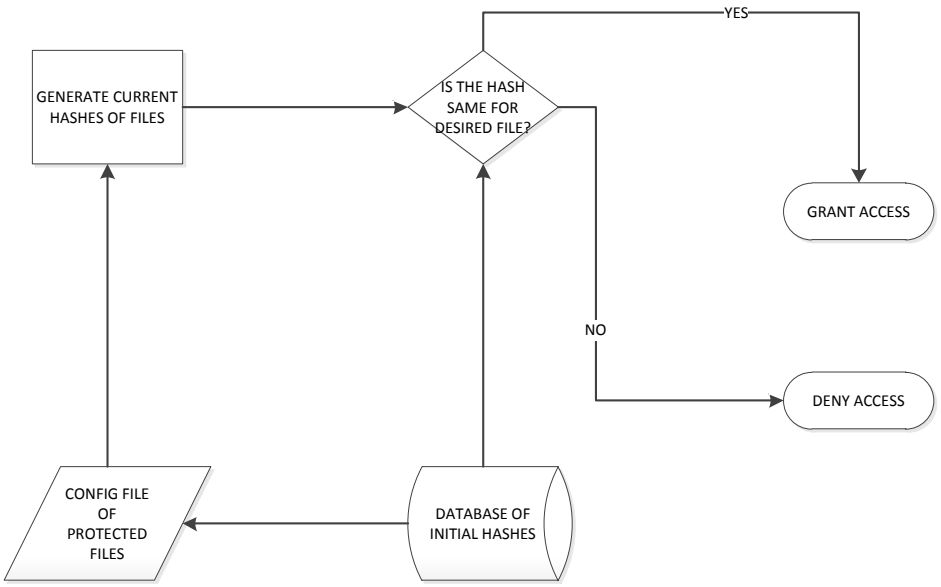

Figure 5.1 Typical Integrity Checking IDS

As we have already discussed that the Linux OS has user and kernel level space. This notion has a lot of bearing on file integrity checking as some tools run in user space with escalated or administrator privileges and some tools are executed in kernel space.

Codes at kernel level give better control of system resources as these include applications for process management, file operations and memory management. While user level codes generally include routine applications, GUIs, text applications and user OS commands.

A usual hazard for system security due to user level code is an error like buffer overflow which can be used to take administrative control of the system. Today's large number of applications is a great obstacle to deal with such a threat due to their distinctive complex architecture. Codes at user level are also run periodically so real time detection is not possible and are not meant for prevention from intrusions. Examples are Tripwire [65],AIDE, Veracity and Integrit [66].

As discussed earlier, kernel level file integrity IDSs can be based on file systems, custom system calls or LSM. I3FS [67] is based on a stackable file design used for kernel-based systems. It blocks access to modified files and alerts the user. It implements four Berkeley databases due to their encrypted nature for database security. The databases are (i) File policy DB, (ii) File data fingerprint or hash DB (iii) File Meta data DB, and (iv) Access frequency DB.

SOFFIC [68], Secure on the Fly File Integrity Checker manages file access based on modified system calls. It main weaknesses are its storage of policy and file Hash DB in regular files, which can easily be modified or deleted by an intruder. Needless to say, that despite being based at kernel level both I3FS and SOFFIC are vulnerable to database modification or OS kernel.

Another example of kernel level IDS system is a research project ICAR. It uses LSM for file access validation, real time integrity checking and provides a system to store backup copies of OS kernel and protected files in write protected media. However, storage of protected files / DBs on write protected media is a big obstacle in updating the system.

In a nutshell, balance between protection and system efficiency must be struck to design a smooth system capable of protecting selected files with recovery opportunity.
Both ICAR and I3FS are closest to our objectives and few of their features can be used in designing own system with similar objectives.

5.3 System Design

5.3.1 Functional layers

Potential targets of an intruder in a functioning operating system can be files and processes both. Current security mechanisms in Linux provide sufficient protection for running processes, therefore we will only focus on file protection in our design. Important files to protect include system and configuration files. System is proposed to be implemented in Linux as it is open source with readily available source and binary codes of all distributions. It provides a convenient way to extend and modify existing functionality. The File integrity IDS is proposed to be implemented at three layers (Fig 5.2) as follows: -

5.3.1.1 Kernel Layer

It is the actual functional part of the proposed system, which is responsible to validate protected files' integrity along with some cache functions. During startup, it is loaded in RAM with the Kernel module. Details are as follows: -

a. *Hashing* - SHA 256 is good option to be used for hashing function at this layer but aspect of CPU performance must also be kept in mind in choosing a suitable hash function. It is used to calculate fresh hashes of protected files for integrity checks.

b. *Cache* - Numerous cache as shown in figure have also been implemented in this layer. These include Blockchain, Read, policy and logging caches. The caches can significantly improve system efficiency by storing results of already performed computations and avoiding same computations again.

c. *Authentication function* - This module has following functions: -

- Granting write access to encrypted databases.
- Permitting updates to files protected by policy.
- Secure mounting of Kernel and IDS backup media.
- Check user roles for IDS operation.
- Check system integrity by comparing Blockchain value and system Merkle value.

d. *User Roles* - This module will check file access against user privileges of read and write permissions and grant / deny access to protected files.

e. *Hash checking* - It is responsible to compare stored and freshly calculated hashes of protected files.

5.3.1.2 Database and Backup layer

Stores numerous databases for system functionality and backup copies of selected files. Included databases are Policy, file info, frequency, Merkle root, Blockchain detection, Authentication, Hash info, user roles Data Bases. Backup of selected critical files is also stored at this layer to initiate recovery of such files. The OS kernel with IDS module may also be backed up in write protected media for recovery if required. The databases are encrypted being Berkeley DBs and require authentication on startup to grant access.

5.3.1.3 User Layer

It will provide user with an interface to interact with system and handle policy configuration, authentication and file updating tasks. It will also allow the administrator to monitor the system for protection and simplify work for intrusion detection in short.

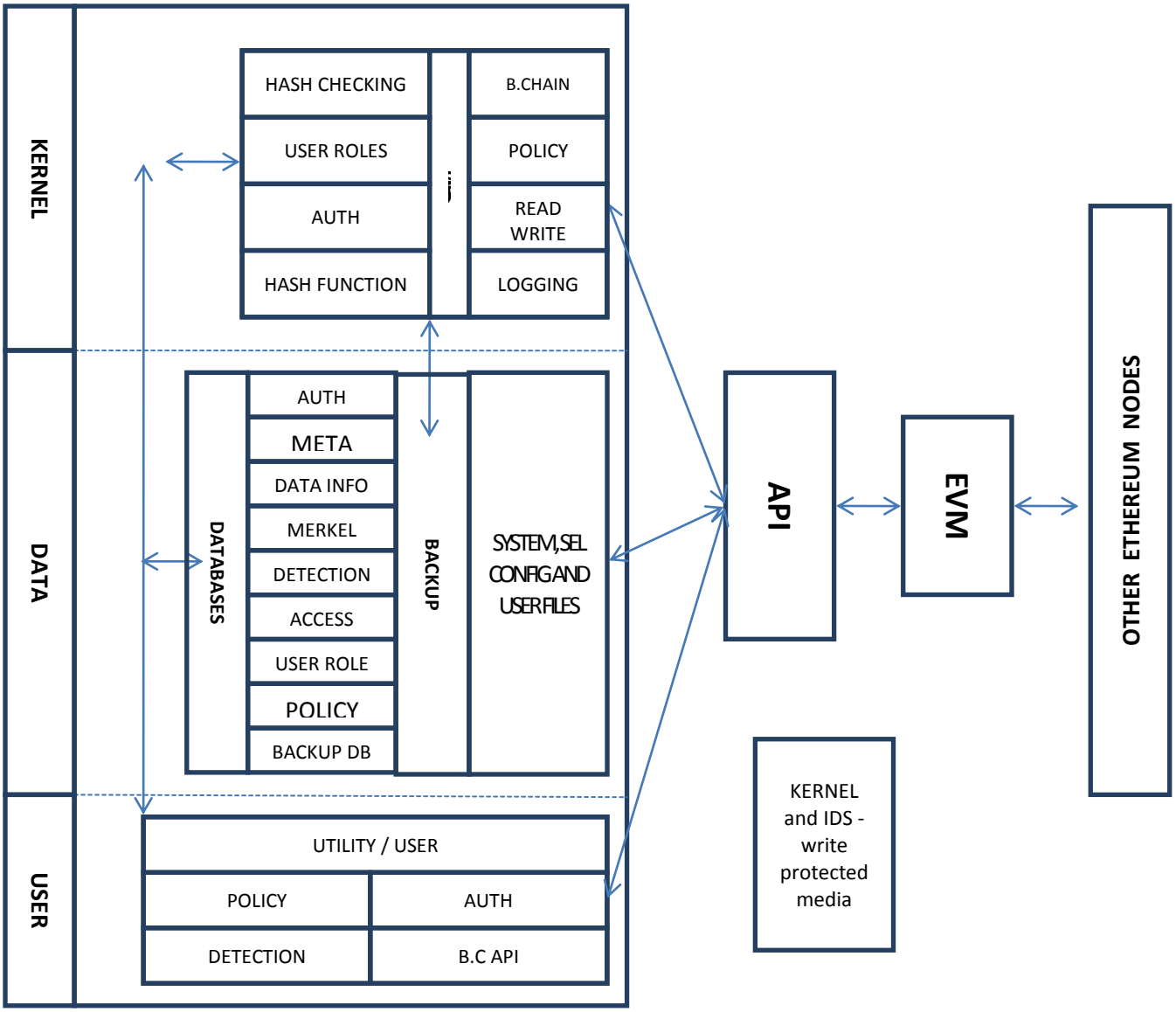

Figure 5.2 Proposed Design-Layers

5.3.2 **Threat Model**

Proposed IDS is mainly designed to detect following threats: -

- Detect replacement of critical files in system i.e. / bin folder to check integrity of useful programs like ps or ls. These may be replaced with Trojans for malicious purposes.
- Detect modification of critical files by malicious intruders.
- Prevent malicious OS updates to critical files before these are executed in the system.
- Detect attacks against kernel / IDS integrity to ensure system in a true state.

5.3.3 **Databases**

Proposed design's configuration data is stored in nine Berkeley Databases [69]. These are extremely high performance and scalable DBs and stores Key, value pairs. DBs use following schema for operation: -

Database	Key	Value
Policy db	Inode #	Policy bits, freq #
Meta db	Inode #	Hash value
Data db	Inode #, page #	Hash value
Access db	Inode #	Counter value #
Merkle db	Respective Timestamp	Hash value, Comparison result
Detection db	Inode #	Hash value, Detection Result
User roles db	Inode #, usernames	Roles, permissions
Back up db	Timestamp	Restore pts
Authentication db	Usernames	Hash value of Password

Table 5.1 Databases and Schema

We propose to use separate schemas for various operations to achieve efficiency and better performance (Back up DB is a separate DB). Detail of DBs is as under: -

a. *Policy Database* - It will hold the policy options related to protected files along with the permitted check frequency before re-calculation of fresh hash. In most schemas, we will use inode number as the key to do away with an additional step of string comparison. Data can be in the form of 8 bytes of value to represent policy options.

b. *Merkle Database* - Hash value of OS kernel including IDS module and optional protected files will be hashed together to obtain the value of Merkle Hash of the protected system. This Merkle value is critical in the entire scheme of IDS design as it will be calculated on each startup and compared with the same stored on immutable Blockchain. This will ensure that the

proposed IDS system is not tempered with and is in true state. The result of comparison along with timestamp will be stored in this DB for audit purpose only.

c. *Detection Database* - It will hold compiled results of malware analysis from Blockchain, updated from time to time. Each OS update file submitted or analyzed for presence of malware by each node, will have its entry in this DB for quick reference. This DB is critical in securely updating the OS.

d. *Access freq Database* - It will hold a counter representing the opening frequency of a file after it was checked last for integrity violation. This frequency option increases the system efficiency as relatively less important files can be checked after a number is achieved. Frequency for each file will be indicated in policy DB through policy bits.

e. *Metadata Database* - Checking metadata for protected files is more efficient than checking entire file data. Any changes in a file will be reflected in the metadata and calculating hash for metadata only is far less time consuming. However, policy configuration will give option to use full data hash checking if desired by user.

f. *Data Database* - Although less efficient than metadata checking but still gives an option to the user to get the entire data hash calculated. However, it will involve more Input/output operation and less efficient utilization of cache as file data will also be called along with metadata.

g. *User roles Database* - It will hold privilege / permission data for users other than file owner and administrator. On requesting read or write access to a protected file, values from this database will be used to compare the requested permission and granted / denied access to file.

h. *Authentication Database* - This DB will hold passwords hash value for each user for DB access and updating protected files. Being encrypted, the DBs require a password to grant access to DB data. File updating is also password protected to safely ensure if the user is authorized for such action or not. The password will be asked as part of startup authentication mechanism. Only during Blockchain detection operation, authentication is required during IDS process.

i. *Backup of Database* - It is important to backup DBs to protect the system from failure if these DBs are deleted or inaccessible. This back up DB will be a

separate DB and not a schema of previously mentioned DB to ensure security through segregation.

5.3.4 Securing Databases and OS Kernel module

Securing databases is a critical operation for the wellbeing and correct operation of the proposed system. As stated before, all integrity checking IDSs face common problem of DB security. Furthermore, an authorized user might update a file and requires write access to same. In this case, the system must follow a secure channel to allow the write operation and segregate authorized and unauthorized requests and update the related DBs.

Our design proposes use of Berkeley DBs, which are encrypted by AES using 128 bits key. We use Berkeley DBs as these are convenient to configure and encrypt by using an inbuilt API. A password is required from user to gain access to the encrypted DBs. This password is acquired on system initialization through a user level application and stored in authentication DB in checksum form. Furthermore, OS kernel with IDS may be backed up in a write protected device like USBs separately. Keeping backup in the same machine will require more storage and less secure, hence a separate device may be used. Since OS updates are not very frequent, therefore using an external device for safety makes sense.

To further ensure security, the restoration from backup will be protected by requiring administrator level authorization on system startup.

5.3.5 Caching

Generally, the protected files are much less than unprotected files in an IDS secured system. Furthermore, protected files having frequent access requests should not have repeated hash calculation. These factors can be easily used to enhance system efficiency by constructing a cache. Our proposed system will have following cache operations as part of system kernel: -

a. *Policy* - It will hold results of previous policy look up for protected files. If policy exists in cache, then this cache will also hold the policy details of the file to avoid looking up in the DB and hence increase system efficiency.

b. *Read data* - On getting positive policy result, a file is granted access. As stated earlier, frequent access must not cause recalculation of hash values, therefore files present in read cache will not be re-calculated for hash unless they meet the frequency number. Each file write operation to such a file will remove the entry from the read cache.

c. *Blockchain result* - Secure updates to OS are handled in a unique way in the proposed system. All updates are broadcasted to every blockchain node in the IDS network to analyze for presence of malware. Detail of this mechanism

will be given later. However, each node will submit its result on the blockchain and cache the result for quick reference, if same file is submitted by another node again.

5.3.6 **Authentication**

Authentication is required due to following reasons: -

a. *Granting access to encrypted Databases* - As stated earlier, the DBs at data layer are encrypted and require a password to gain access. The authentication at this layer ensures that no malicious user can access the DBs without going through a secure channel. Password acquired on each startup is compared to the user provided password stored during initialization.

b. *Allowing secure OS and file updates* - Users should be able to securely update the protected files and saving of new OS updates should be done through a secure channel only. In this case, our design allows updates to take place only after authenticating the valid user. This is a necessary step to avoid re-initializing the system for updates to take effect.

c. *Secure restore from external Backup media* - Kernel and IDS module binaries are optionally backed up in a write protected external media like a USB, CD-R/DVD-R. If required, recovery of OS can be done from such a media. The restoration will be initiated on administrator authentication, after the Kernel and IDS binary fail the Blockchain comparison test.

d. *Check user roles* - On system start up, users will be asked to enter their credentials and compared with data in authentication DB. Once authenticated, each file access will be granted as per user permissions in user roles DB.

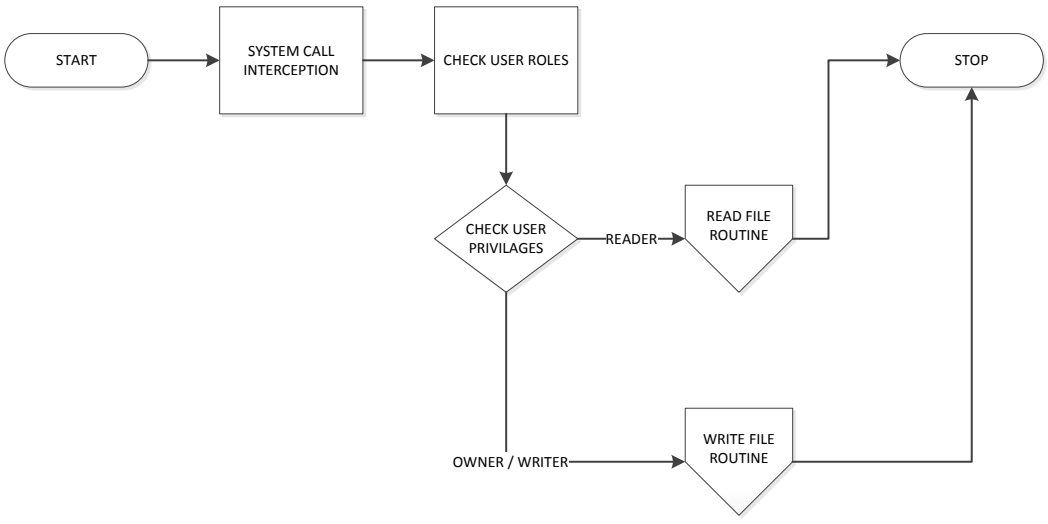

Figure 5.3 Checking User Roles

e. *Check system integrity* - System integrity will be checked on each startup by calculating the Merkle value of kernel, IDS and selected files. This value will be compared with Merkle value from blockchain. System will only boot completely if test is successful or will prompt the user to authenticate for system restore from external media.

5.3.7 **Secure Updates**

Due to scalability issues, we are limiting ourselves to design IDS system for a server, which must undergo very few updates. Normally there are two types of updates. OS updates and file updates by users. Our design proposes a novel approach to ensure that updates are not malicious and are carried out through a secure channel. Details are as under: -

5.3.7.1 **OS updates**

Our novel approach to analyze each OS update is based on use of blockchain. Normally, updates in a network are sent to client systems from a FTP server. However, this arrangement is not very secure as any intruder can infect the centralized server for malicious updates and infect all systems in a network. Authors in [58] propose a novel method to check updates for malware.

a. Any update file in a system or blockchain node is broadcast to all other nodes to analyze for presence of malware. Here we are not using any specific detection engine, but all nodes may use different techniques for detection. The result from all nodes is added to the blockchain and is compiled. Majority vote principal is applied to the update file to know if it carries malware or not. Blockchain provides diversification and decentralization to this process ensuring better result probability and avoiding single point of failure.

b. Update is applied to OS if blockchain majority vote indicates that no traces of malware were present in the update file.

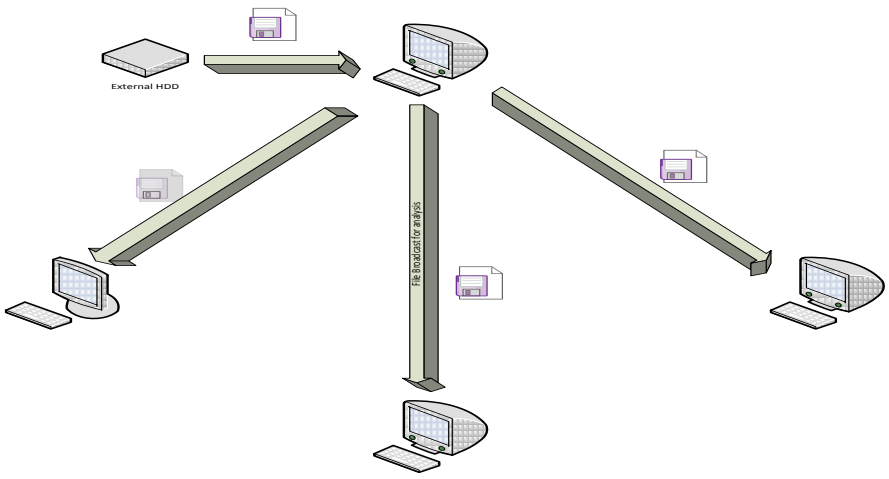

Figure 5.4 Collaborative Malware Detection

5.3.7.2 **User updates to protected files**

Users will be checked for roles to obtain read and write privileges to protected files. Users will only be able to write to files after authentication on startup. Such files can also be broadcasted to blockchain nodes before saving, however such an arrangement is deemed to be very resource intensive as user file updates may be relatively frequent than OS updates. Therefore, policy option will be given to a user to check file for written traces of malware or just use authentication mechanism to assess user roles for protection. Later option is proposed to be the default option.

5.4 **Blockchain Integration with IDS system**

Existing file integrity IDSs face common threats to DB security and managing secure updates to OS. In our design, we have endeavored to address both issues using the new blockchain technology. Blockchain not only provides immutable record for a system state but can also be used for maintaining a decentralized malware detection network. In the start of this research, we wanted to use blockchain for following purposes: -

a. Ensuring true system state by using Merkle root value of all files, stored on blockchain. The decentralized and immutable nature of blockchain ensures that recorded Merkle value is true and cannot be hacked.
b. Ensuring safe updates to OS and few cases of file updates by collaborative malware analysis.
c. Storing backup copies of protected files and OS on blockchain nodes for recovery if intrusion detected.

First two uses have been incorporated in the design to some extent, however the third use may not be possible at this technological age of blockchain due to scalability and storage issues. Large scale blockchain adoption is currently hampered due to these issues of mass storage and transaction speed. Several blockchain startups are working on increased transaction speeds and scalability like Quarkchain or Zilliqa and many more are also researching to provide mass storage for blockchain applications i.e. Bluezelle or Phantasma. We propose to use Ethereum smart contract platform to implement the system. Ethereum is a second generation blockchain and provides users with smart contract writing facility. We propose to use Ethereum as it is widely used in blockchain world with several online resources available. Almost all other smart contract platforms are in development stage baring few like Neo, but these have very less resources available online. Furthermore,

Ethereum development is quite efficiently integrated with Microsoft Azure platform for testing and deployment.

Ser	Specification	Bitcoin	Ethereum
1.	Block time	10 minutes	15 seconds
2.	TPS max	2000 tps	2000 tps
3.	Blockchain size	173 Gb	> 1 Tb
4.	Block size	1 Mb	Variable
5.	Monthly growth	4.4 Gb	187 Mb

Table 5.2 Comparison of Bitcoin and Ethereum

5.4.1 Blockchain Merkle root

Due to scalability issues and simplicity, only Merkle root value from OS kernel with IDS module will be stored on the blockchain along with timestamp as a default. The root is a resource intensive step and will be carried out once on initialization. Files will not be included in Merkle root on file updating as new Merkle value will have to be calculated and updated on blockchain. The new Merkle value calculated will be non-persistent and not stored on the system.

OS updates are less frequent, so Merkle root value updating on blockchain after authentication will not be required often. To check system integrity on startup, OS kernel and IDS module root value will be recalculated and compared with same value from blockchain as stated earlier. Since recalculating Merkle root for every startup may be time consuming, therefore an option in policy can be constructed to carry out this step after a set frequency or on start up by an administrator.

5.4.2 Value exchange through API

A custom API will be used to transfer values between blockchain virtual machine and the IDS module. This API will have following functions: -

a. Provide Merkle root value from IDS module to blockchain and vice versa.

b. Provide files data to blockchain for nodes involved in malware detection.

c. Getting detection results of each node from blockchain and calculating majority vote.

d. Updating majority vote result on blockchain.

e. Updating detection DB by getting all previous results of majority votes from blockchain.

f. P2P transfer of files between detection nodes for malware analysis.

5.4.3 Ethereum smart contract process

Developing exclusive blockchain for the proposed IDS is not viable due to development costs involved. Therefore, we will use Ethereum platform of smart contracts to develop the required blockchain application. This will eliminate all development steps for nodes, consensus protocol, value storage etc. The Ethereum web application developed through a smart contract, will just be used as an online storage platform by custom API for IDS functionality.

5.4.4 File data exchange for Malware detection

As stated earlier, a custom API will exchange values between the IDS module and the blockchain network. What values are exchanged for an operation, is critical. Detail about values is as under: -

5.4.4.1 System validation check on start up

Only Merkle root value calculated from OS kernel, IDS module along with timestamp is stored on the blockchain on system initialization. On each startup, the calculated value less timestamp will be compared with the blockchain value.

5.4.4.2 Malware detection

a. File metadata and hash sent to blockchain and vice versa.

b. Result of malware analysis to blockchain.

c. Result of majority vote result to blockchain.

d. List of available nodes for malware detection from blockchain.

e. Confirmation of P2P file reception to blockchain.

f. Complete list of analyzed files with result from blockchain to update detection DB.

g. Results of individual malware analysis by nodes from blockchain.

5.5 Selection of Files for protection

It is imperative that we build the file protection list with care. The selection depends largely on the system's tasks and objectives. Almost every research publication recommends protecting system and configuration files, however there may be other files which might be important for correct system operation. The file selection is important not only from security point of view but also for balanced performance. It is not possible to guard all files in a system or frequently changing files i.e. log files due to performance issues. But at the same

time, there might be some data files like personal or financial information which require protection. Frequency of file access and security significance are the main factor to select a file for protection.

5.5.1 Types of files

In modern OSs, all system components such as instructions, drivers and data are stored as files. The quantity of files in an OS is huge and these are the prime target of an attacker to intrude into the operating system. Attacks on operating systems are carried out by modifying or replacing the original files. It is important to understand the importance of each file type along with its effect on system security. Detail is as under: -

a. *OS, program and app files* - These files are vital for correct operation of a system and only change during a patch or upgrade implementation, hence must be protected. If an intruder is successful in replacing one of these files, he can conveniently hijack the system or install a malware. In every intrusion, a malware is copied into the target system behaving like a system file i.e. Trojan. A file integrity checker can easily detect such a threat. Examples of system files may include executables, libraries supplied with OS and directories of other applications.

b. *Configuration files* - These are also important to secure but their modification may not result in a security emergency. Configuration settings restrict access to services and objects through user privileges and should be monitored for modification. These may be required to modify frequently, therefore to ensure better system performance, only selected configuration files be protected. The selection may depend on system tasks and software type being configured.

c. *Temporary files and User data* - These files update frequently and have reduced privileges, therefore may not be selected for protection. Although these may be attacked by an intruder but due to reduced permissions, do not pose a grave threat to system functionality. A user may select a data file for protection if deemed necessary, when configuring the IDS policy. Few file classification algorithms can aid a user in selecting a data file for protection.

5.5.2 File classification algorithms

As stated earlier, system files are prime focus of intruders and must be protected for security. There are other files as well, which might be important for a user such as files related to online banking services, medical records, military sensitive data files or files related to web hosting. It is useful to classify these files for protection as per every user's

unique requirements. In [70], authors have classified files as per their security weight. The formula is: -

$$wi = (\alpha * fi) + \{\beta i * di\ (\alpha + \beta = 1)\}$$

Where,

wi= weight of file i

fi= access frequency of file i

di=significance of the directory containing the file i

α and β are related with proportion of directory significance and frequency

Microsoft also provides File Classification Infrastructure (FCI) in [71] Windows Server edition 2008 R2. It is to aid in selecting user files for protection. It is more focused towards selecting business data files than system files and depends on business impact due to intrusion.

Another algorithm in [72] is provided which is more related to security posture of a system. Files are classified into three types; read only, write free and log-on files and are related to security levels. The security levels are also classified into high, medium and low levels. High security files require real-time checking i.e. system files. Medium level files may be checked for integrity periodically and low-level files may be ignored for checking.

In [73], authors provide another angle to classify files. Threshold values for sensitive and significant files are defined by tsig and tsen. If wi is greater than threshold values, then the file is placed in the related group i.e. ssig or ssen, otherwise it belongs to ordinary files group sord. Files in significant group can be read or executed by a user but not modified even by a root i.e. executables. Files in sensitive group can be modified but the activity must be logged i.e. configuration files. All other files like temporary files fall under ordinary group and do not require integrity checking for intrusion.

5.5.3 **Recommendation**

After analyzing different protection models and rationale, we arrive that all system files must be protected. Few important configuration files be selected depending on the system role. Lastly all other files may be analyzed by user for business or security needs by himself or above mentioned algorithms.

5.6 **IDS Algorithms**

5.6.1 **System Start up**

The system may face a threat of kernel or IDS modification. This threat is averted with a novel solution by checking the system integrity on start up. Figure 5.5 below presents the startup protection algorithm. After the OS is loaded in RAM by Linux loader, the system

undergoes integrity check as shown in the flowchart. As the Merkle checksum value is non-persistent, it is not saved in the same system. Therefore, a fresh Merkle value is calculated as per configured security policy and compared with the value from blockchain.

System boot process will resume only if the value is correct otherwise the system will ask the user to undergo administrator authentication to restore system from external backup. The system will shut down if user fails to enter the administrator password correctly.

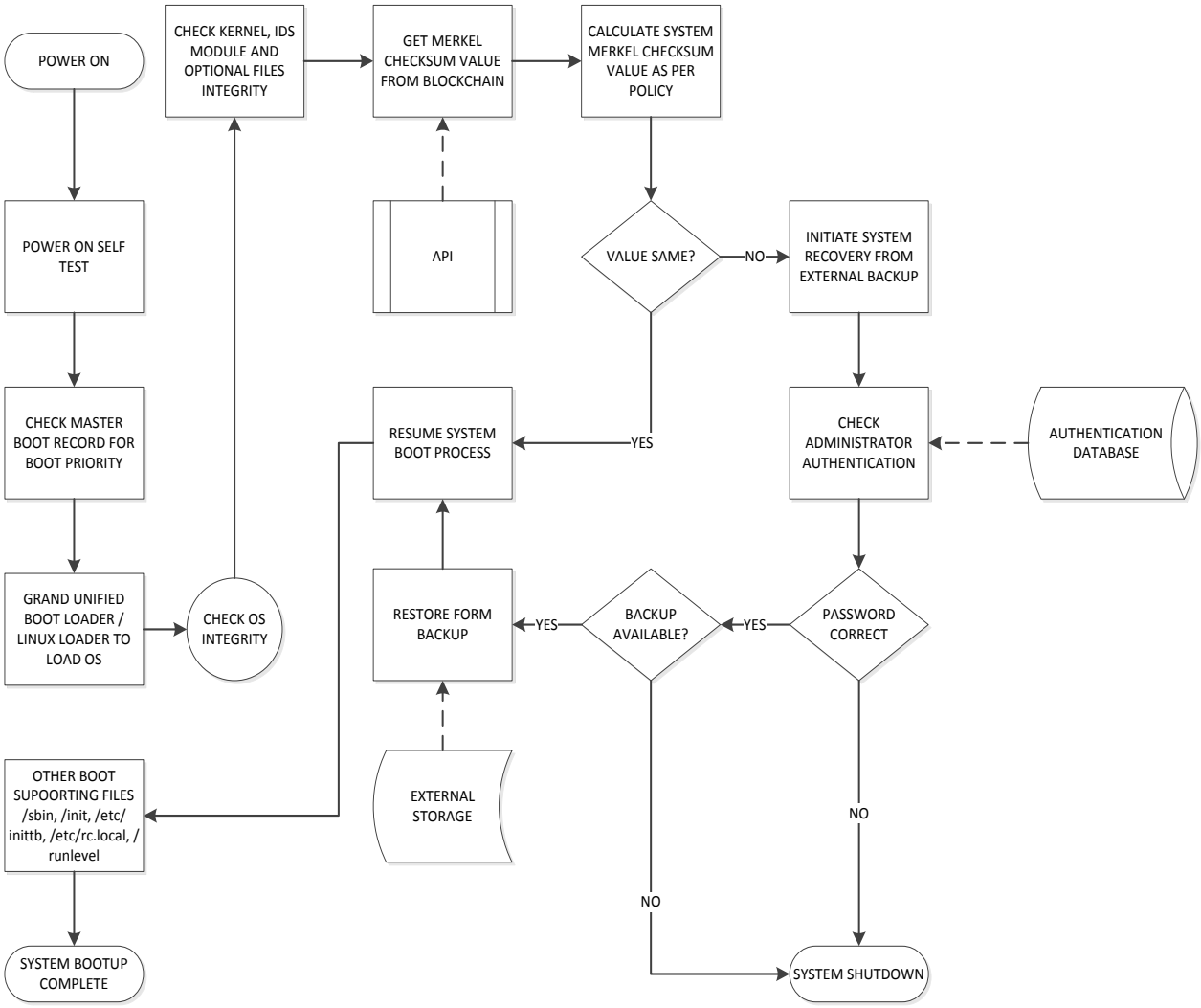

Figure 5.5 System Start up Algorithm

5.6.2 **File Read**

After file operation are segregated as per user permission as explained in section 5.3.6 (d), the file read or write processes will initiate. Figure 5.6 presents file read operation. File integrity is checked during file read request if a related protection policy is found in the protection DB. Access permission against each user is detected and file access is granted or denied after updating the file access logs for later auditing.

To eliminate repetitive calculation of file checksum for integrity, it is checked if the file has been read before by scanning the read cache. Access is instantly granted if an entry for the same file is found in the cache. On the contrary, absence of file entry will show that the file is being checked for the first time and system will initiate the checksum calculation mechanism. Access to file is controlled based on result of Hash comparison from meta or data DB (depends on policy) and fresh value. It is important to lock and unlock cache to avoid race conditions in multi user environment.

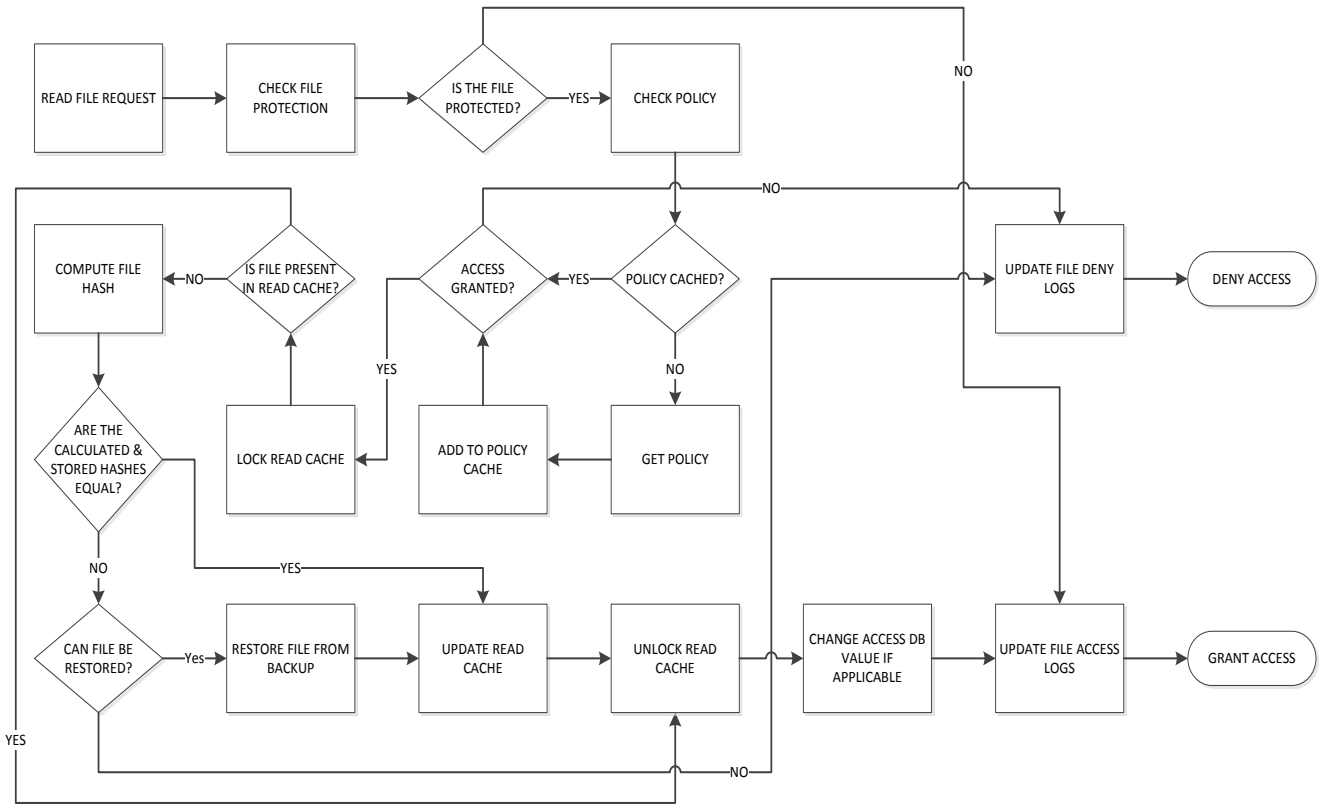

Figure 5.6 Read File Algorithm

5.6.3 File Write

Figure 5.7 gives the file write algorithm. This algorithm is critical to permit writing to protected files. After segregating owners and writers, if a write request is received then policy will be checked and access to file will be granted as configured.

It is important to consider that the file is going to be written to, so its checksum will change. Therefore, the file entry from read cache will be removed to perform the hash check on next read operation. Furthermore, the checksum and other related DBs are also updated after the file write operation is completed or the file is saved.

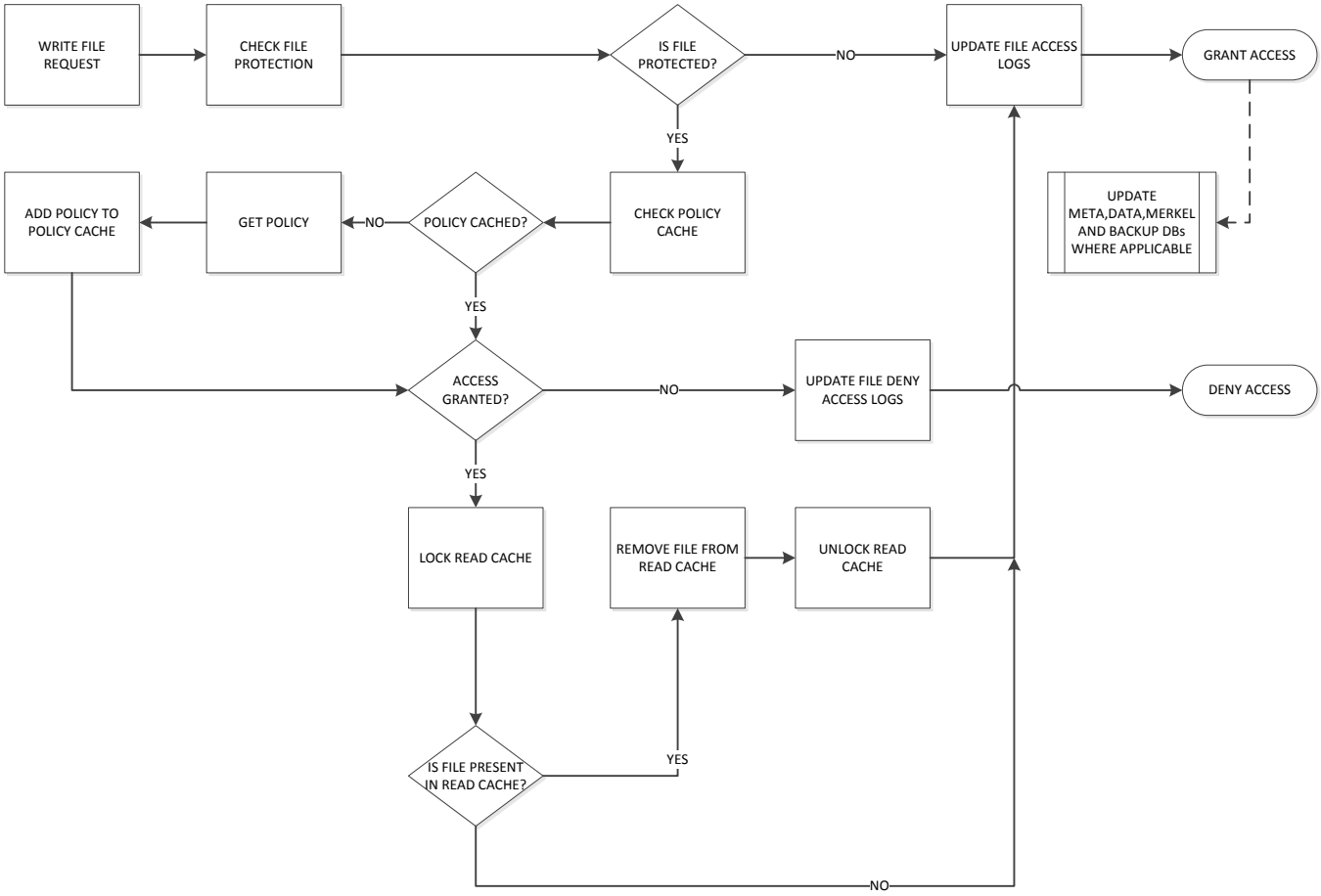

Figure 5.7 Write File Algorithm

5.6.4 **BC Query**

Our design follows a novel approach to check OS updates for presence of malware. Authentication is required after receiving the file open request to check if the user wants to undertake the blockchain detection mechanism. This step is required as the user might be confident of having the update copy from authentic source and not require detection.

If detection is required, then the file is broadcast to all nodes in blockchain network to carryout detection. On receiving results from all nodes, a majority vote is calculated, and result is also uploaded on blockchain for record of other nodes. File open is controlled based on majority vote.

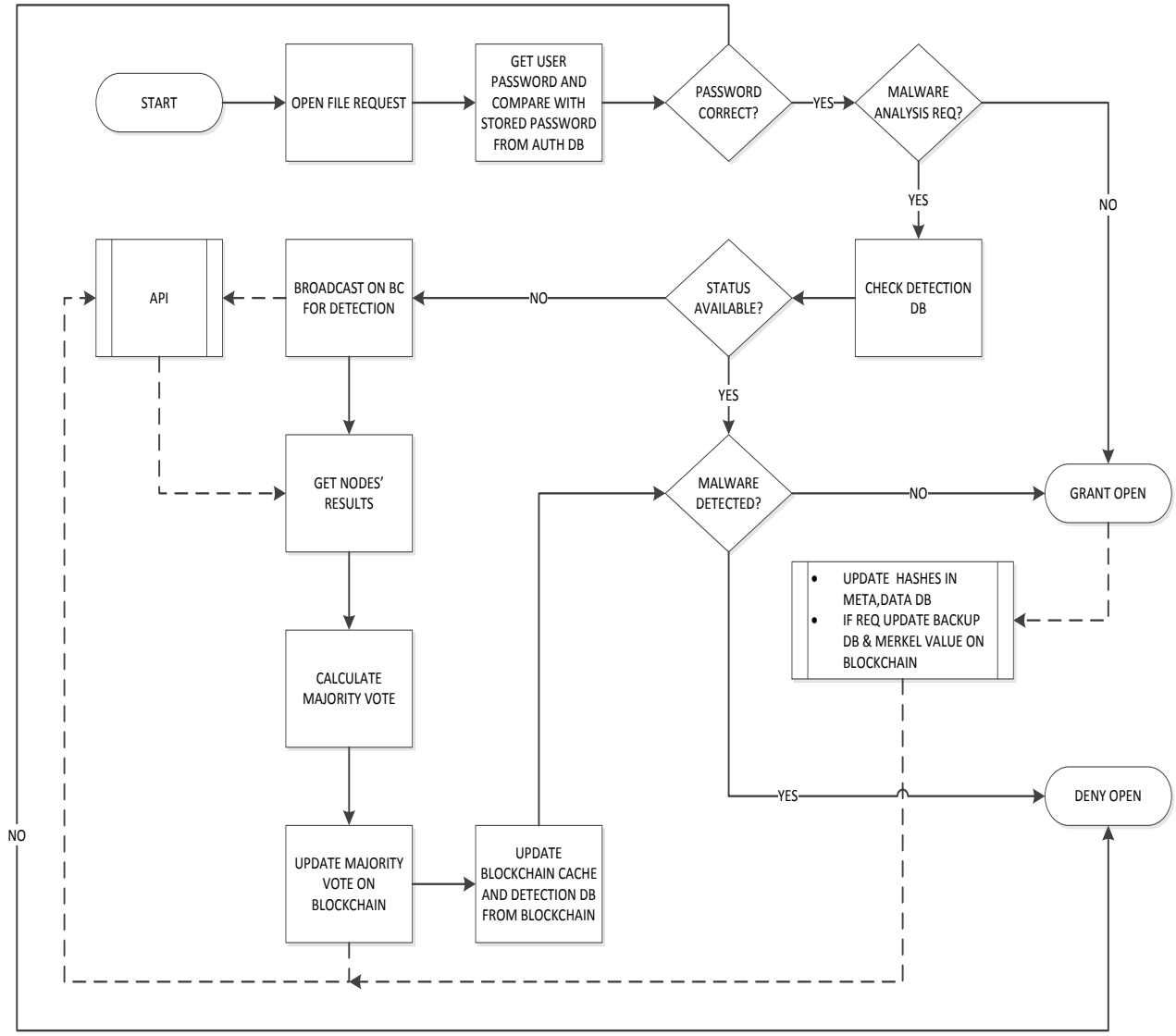

Figure 5.8 Blockchain Query Algorithm

5.6.5 **BC Collaborator**

A blockchain collaborator node aids other nodes to detect if the OS update is malware free. On receiving the file from the query node, each collaborator node will use own detection engine and upload result on blockchain. An entry about the result will also be made in the blockchain cache. The file will be deleted after the analysis is complete. Figure 5.9 presents the collaborator mechanism.

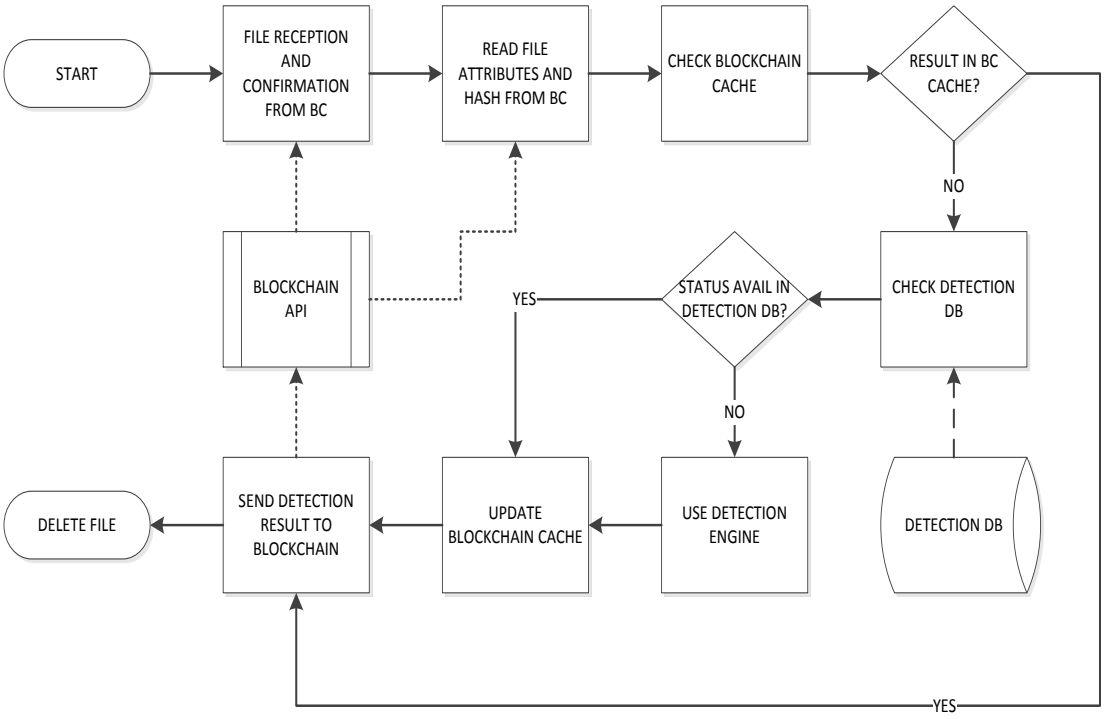

Figure 5.9 Blockchain Detection Algorithm

5.7 Challenges

The proposed system comes with several challenges. Few are listed below: -

a. *Scalability* - Integrity based IDS face a big problem of scalability. Performance overhead increase with the increase in the number of files to be protected. If OS kernel is added, the performance is even more degraded. Additional features will further reduce the system efficiency. We feel that our design will be resource intensive and will require simplification to reduce performance overhead. I3FS [67] causes an overhead of 4%, whereas ICAR [74] causes a performance overhead of 10% on the system.

b. *Files selection* - Selecting files for protection is complex as different applications have different critical files. We do not want to select all files for protection due to scalability reasons, therefore limited files only will be protected for integrity. Due to difference in relevance of files for different applications, it is difficult to establish a default pattern for selecting protected files.

c. *Blockchain integration* - Blockchain is still in its infancy. A great solution to secure IDS DB would have been backup storage on blockchain, however blockchain storage is not currently at the level of required scale. At the

moment, blockchain can only be used for minimal value storage and consensus.

d. *Linux extensibility* - LSM is an access control mechanism for files in Linux OS. It uses hooks to call functions to validate user access permissions to Linux objects. However, recently the hooks have been removed due to security reasons. Therefore, our entire real time call interception design will have to be custom developed.

Chapter 6

FUTURE WORK AND CONCLUSION

6.1 Future Work

As stated earlier, Blockchain is an emerging technology. A lot can be done to improve the proposed design with the use of Blockchain. Possible roadmap for the future work related to this thesis is given below: -

a. Development of smart contract for system integrity check and malware detection in Ethereum.
b. Development of proposed IDS system in Linux.
c. Selection of an efficient Hashing algorithm for proposed design.
d. System design validation by carrying out intensive benchmarking tests.
e. Leveraging Blockchain technology to improve system design by incorporating Blockchain based storage for backup.
f. Incorporate Blockchain based collaborative DDoS protection mechanism.
g. Study development of a private Blockchain platform for IDS operation only.
h. Using the same framework for other types of IDS for database protection.

6.2 Conclusion

The idea behind this research work was to asses new Blockchain technology for securing the integrity of IDSs database and operating system updates. During our research, we studied various existing IDS solutions and established that available techniques to secure the databases and updates suffer from either flexibility or security issues. In our research we developed a novel IDS framework levering Blockchain technology and predicted that protection algorithms thus developed will provide the desired protection levels in IDSs. We used file integrity checking IDSs for our research due to their simplicity, but we also envisage that our framework and related algorithms will also work for other IDS techniques.

We established a threat pattern specific to IDS security and developed our algorithms to counter these threats. We are positive that these algorithms and framework will deliver the required protection from these threats.

We plan to build a prototype of the proposed system to establish benchmarks and will also use other IDS techniques to validate the viability of the framework design.

REFERENCES

[1] Kim, Gene H, and Eugene H. Spafford. "The design and implementation of tripwire: A file system integrity checker", Proceedings of 2nd ACM Conference on Computer and Communications Security. ACM, 1994.

[2] Ken, Bypassing Tripwire and MD5 Hash Checking", internet: http://caffeinesecurity.blogspot.com/2013/05/bypassing-tripwire-and-md5-hash.html, Jan 5, 2013 [Jan 27,2018].

[3] Bukac V, Tucek P, Deutsch M. (2012), "Advances and Challenges in Standalone Host-Based Intrusion Detection Systems". In Fischer-Hübner S., Katsikas S, Quirchmayr G. (eds) Trust, Privacy and Security in Digital Business. TrustBus 2012. Lecture Notes in Computer Science, vol 7449. Springer, Berlin, Heidelberg.

[4] Hu, J: Host-Based Anomaly Intrusion Detection. In Handbook of Information and Communication Security, pp. 235–255. Springer, Heidelberg (2010).

[5] Swapnil Patel , Gopalan Sivathanu, Anand Kashyap, Erez Zadok, FS : "An In-Kernel Integrity Checker and Intrusion Detection File System", Proceedings of the 18th USENIX conference on System administration, November 14-19, 2004, Atlanta, GA.

[6] Mosley, D. (2006). "Implementation of a File Integrity Check System". East Carolina University.

[7] Motara, YM and Irwin, BVW. "*File Integrity Checkers: State of the Art and Best Practices"*. Information Security South Africa (ISSA). Sandton Conference Centre, Johannesburg. July 2005.

[8] Gerco Ballintijn, William A and Leendert van Doorn. Arbaugh. "Design and Implementation of Signed Executables for Linux". Technical Report CS-TR-4259, June 2001.

[9] Ivan Visconti and Luigi Catuogno. "An Architecture for Kernel-Level Verification of Executables at Run Time". The Computer Journal, 47(5), September 2004.

[10] Tripwire, Inc. Tripwire for Servers Datasheet. Technical report, Tripwire, Inc., 2005.

[11] Gerco Ballintijn, William A and Leendert van Doorn. "Design and Implementation of Signed Executables for Linux". Technical Report CS-TR- 4259, June 2001.

[12] A. Chuvakin, "Ups and Downs of UNIX/Linux Host Based Security Solutions", login: The Magazine of USENIX & SAGE, Apr. 2003, pp. 57-62, vol. 28, No. 2.

[13] Del Armstrong. (2003). "*An Introduction to File Integrity Checking on Unix Systems"* (Global Information Assurance Certification Paper). Retrieved from http://https://www.giac.org/paper/gcux/188/introduction-file-integrity-checking-unix-systems/104739.

[14] K. Wüst, A. Gervais, "Do you need a blockchain?", pp. 375, 2017, [online] Available: http://eprint.iacr.org/2017/375.

[15] Barbara Liskov and Miguel Castro. "Practical byzantine fault tolerance". In Proceedings of the Third Symposium on Operating Systems Design and Implementation, OSDI '99, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association.

[16] "Bitcoin: A peer-to-peer Electronic Cash System", Available: https://bitcoin.org/bitcoin.pdf.

[17] Björn Scheuermann and Florian Tschorsch, "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies," in IEEE Communications Surveys & Tutorials, vol. 18, 2016.

[18] D. Chaum, "Blind signatures for untraceable payments," in Proc. 2nd Conf. Adv. Cryptol, Aug. 1982, pp. 199–203.

[19] S. Sabett, L. Law and J. Solinas, "How to make a mint: The cryptography of anonymous electronic cash," Amer. Univ. Law Rev, vol. 46, no 4,1996.

[20] S. Chandrakumar, V. Vishnumurthy and E. G. Sirer, "KARMA: A secure economic framework for peer-to-peer resource sharing," in *Proc.1st Workshop Econ. Peer 2 Peer Syst. (P2PEcon'03),* Jun. 2003.

[21] W.Dai (1998)." *B-Money"*,[Online],Available: http://www.weidai.com/bmoney.txt.

[22] H. Finney. (2004). "*Rpow"*, [Online]. Available : http://cryptome.org/rpow.html.

[23] S. Nakamoto. (2008). "*Bitcoin P2P e-Cash Paper",* [Online].Available: https://www.mail-archive.com/cryptography@metzdowd.com/msg09959.html.

[24] "Blockchain Technology: Beyond Bitcoin", Available: scet.berkeley.edu/wp-content/uploads/BlockchainPaper.pdf.

[25] Ko D, Choi S,Yli-Huumo J, Park S and Smolander K, "Where Is Current Research on Blockchain Technology? A Systematic Review". PLoS ONE 11(10): e0163477. doi:10.1371/journal. pone.0163477, 2016.

[26] Charters S and Kitchenham B. "Guidelines for performing Systematic Literature Reviews in Software Engineering",2007.

[27] "Blockchain: Powering the Internet of Value". Available: https://www.evry.com/globalassets/insight/bank2020/bank2020blockchainpoweringtheinternetofvaluewhitepaper.pdf.

[28] "The logical components of Blockchain", Available: www.linkedin.com/pulse/logical-components-blockchain-tony-willenberg.

[29] "A Gentle Introduction to Blockchain Technology", Available: bravenewcoin.com/assets/Reference-Papers/A-Gentle-Introduction/A-Gentle-Introduction-To-Blockchain-Technology-WEB.pdf.

[30] Shigeichiro Yamasaki, Hitoshi Okada and Vanessa Bracamonte," Proposed Classification of Blockchains Based on Authority and Incentive Dimensions" . ICACT2017 February 19 ~ 22, 2017.

[31] Swan M. "Blockchain: Blueprint for a New Economy" O'Reilly Media, Inc.; 2015.

[32] "Double-spending", 2016. Available: https://en.bitcoin.it/wiki/Double-spending.

[33] Bitcoinwiki; 2015. Available: https://en.bitcoin.it.

[34] Antonopoulos AM. "Mastering Bitcoin: unlocking digital cryptocurrencies", O'Reilly Media, Inc, 2014.

[35] "Proof-of-Stake",2016. Available :https://en.bitcoin.it/wiki/Proof_of_Stake.

[36] Dhiren Patel, Jay Bothra and Vasudev Patel "Blockchain exhumed "ISEA Asia Security and Privacy (ISEASP),2017.

[37] "Blockchain technology: 9 benefits & 7 challenges", Available: http://www2.deloitte.com/nl/nl/pages/innovatie/artikelen/blockchaintechnology-9-benefits-and-7-challenges.html.

[38] "4 major technical challenges facing IoT developers", Available :www.sitepoint.com/4-major-technical-challenges-facing-iot-developers/.

[39] IBM Blockchain trend report 2017, Available: www.slideshare.net/HorizonWatching/blockchain-trend-report-2017.

[40] Marco Conoscenti, Antonio Vetro and Juan Martin," Blockchain for the Internet of Things: A systematic literature review", IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA) , 2016.

[41] Ali Dorri, Salil Kanhere and Raja Jurdak-Praveen Gauravaram "Blockchain for IoT security and privacy: The case study of a smart home",IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops) ,2017 .

[42] Ali Dorri, Salil Kanhere and Raja Jurdak,” Towards an Optimized Blockchain for IoT", Proceedings of the Second International Conference on Internet-of-Things Design and Implementation - IoTDI '17 , 2017.

[43] Kamanashis Biswas and Vallipuram Muthukkumarasamy ," Securing Smart Cities Using Blockchain Technology", IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016.

[44] Sayed Hashemi, Faraz Faghri, Paul Rausch and Roy Campbell ," World of Empowered IoT Users " , IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI) , 2016.

[45] Aymen Boudguiga, Nabil Bouzerna, Louis Granboulan, Alexis Olivereau, Flavien Quesnel, Anthony Roger and Renaud Sirdey ," Towards Better Availability and Accountability for IoT Updates by Means of a Blockchain", IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2017.

[46] Boohyung Lee, Jong and Hyouk Lee ," Blockchain-based secure firmware update for embedded devices in an Internet of Things environment", The Journal of Supercomputing , 2016.

[47] Seyoung Huh, Sangrae Cho and Soohyung Kim , " Managing IoT devices using blockchain platform", 19th International Conference on Advanced Communication Technology (ICACT) , 2017.

[48] Marco Conoscenti, Antonio Vetro and Juan Martin ," Peer to Peer for Privacy and Decentralization in the Internet of Things", IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C) , 2017.

[49] Thomas Hardjono and Ned Smith ," Cloud-Based Commissioning of Constrained Devices using Permissioned Blockchains ",Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security - IoTPTS '16 , 2016.

[50] Yu Zhang and Jiangtao Wen ," The IoT electric business model: Using blockchain technology for the internet of things", Peer-to-Peer Networking and Applications , 2016.

[51] Jianjun Sun, Jiaqi Yan and Kem Zhang ," Blockchain based sharing services: What blockchain technology can contribute to smart cities ",Financial Innovation , 2016.

[52] Mayra Samaniego and Ralph Deters ," Using Blockchain to push Software-Defined IoT Components onto Edge Hosts", Proceedings of the International Conference on Big Data and Advanced Wireless Technologies - BDAW '16 , 2016.

[53] Jaemin Park and Kwangjo Kim ," TM-Coin: Trustworthy management of TCB measurements in IoT", IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops) , 2017.

[54] Mayra Samaniego and Ralph Deters," Blockchain as a Service for IoT ", in Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (Smart Data), 2016.

[55] Bains.P , Murarka.N (2017) . *Bluzelle: A Decentralized Database for the Future.* [Online] Available from: https://crushcrypto.com/wp-content/uploads/2018/01/BLZ-Whitepaper.pdf .

[56] Flores.S, Ferreira. M, Paixão.A, Pinho,B (2018). *Phantasma Protocol.* [Online] Available from: https://phantasma.io/phantasma_whitepaper.pdf.

[57] PolySwarm (2018). *PolySwarm: A Decentralized Cyber Threat Intelligence Market.*[Online] Available from: https://polyswarm.io/polyswarm-whitepaper.pdf.

[58] S. Raje, N. Wilson, S. Vaderia and R. Panigrahi, " Decentralized firewall for malware detection," *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*, Mumbai, 2017, pp. 1-5.

[59] Quarkchain Foundation (2018). *Quarkchain: A High Capacity Peer to Peer Transactional System.* [Online] Available from: https://quarkchain.io/QUARK%20CHAIN%20Public%20Version%200.3.4.pdf.

[60] The Zilliqa Project (2017). *Zilliqa: A Secure, Scalable Blockchain Platform* .[Online] Available from: https://docs.zilliqa.com/whitepaper.pdf.

[61] Bace R, Mell P . "NIST special publication on intrusion detection systems". DTIC Document, 2001.

[62] Preneel, B. "State of the art ciphers for commercial applications", Computer Security, 1999, 18, (1), pp. 67–74.

[63] Wright C, Cowan C, Smalley S, Morris J, Kroah and Hartman G."Linux security modules: General security support for the Linux kernel". In: Proc. 11th USENIX Security Symposium, San Francisco,CA, 2002, vol. 2, p. 44.

[64] Z.C, McGill and Payne C., "Empowering end users to confine their own applications: the results of a usability study comparing SELinux, AppArmor, and FBAC-LSM", ACM Trans. Inf.Syst. Secur. (TISSEC), 2011, 14, (2), p. 19.

[65] Kim, G.H. and Spafford E.H.: 'The design and implementation of tripwire: a file system integrity checker'. In: Proc. Second ACM Conf. Computer and Communications Security, ACM, 1994, pp. 18–29.

[66] Rocke, A.J and DeMara: "CONFIDANT: Collaborative object notification framework for insider defense using autonomous network transactions". Agents Multi-Agent Syst., 2006, 12, (1), pp. 93–114.

[67] Patil, S., Sivathanu, G., Kashyap and Zadok E."I3FS: An in-kernel integrity checker and intrusion detection file system". In: Proc. 18th Annual Large Installation System Administration Conf. (LISA04), 2004.

[68] Da Silveira Serafim and Weber R.F."Restraining and repairing file system damage through file integrity control", Comput. Secur., 2004, 23, (1), pp. 52–62.

[69] O. Yigit and Seltzer M, ‘‘A New Hashing Package for UNIX,’’ Proceedings of the Winter USENIX Technical Conference, pp. 173-84, http://www.sleepycat.com , January 1991.

[70] Jin, H. "A guest-transparent file integrity monitoring method in virtualization environment" Computers and Mathematics with Applications, 2010. 60(2): p. 256-266.

[71] "Microsoft. File classification infrastructure, technical white paper". 2009; Available from: http://www.microsoft.com/windowsserver2008/en/us/fci.aspx.

[72] R. Mahmod, Z. H. Abdullah, N. I. Udzir and K. Samsudin. 2011. "Towards a dynamic file integrity monitor through a security classification". Internal Journal of New Computer Architectures and Their Applications (IJNCAA) 1, 3, 766-779.

[73] Jin, H., Xiang G., Zou D., Zhao F., Li M. and Yu C. " A guest-transparent file integrity monitoring method in virtualization environment". Computers & Mathematics with Applications 60(2), 256–266 (2010).

[74] Wrobel MR and Kaczmarek J. "Operating system security by integrity checking and recovery using write-protected storage". IET Information Security. 2014; 8(2):122–31.