ISSN 2278-3091

Volume 9, No.3, May - June 2020

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse194932020.pdf

https://doi.org/10.30534/ijatcse/2020/194932020

Linux Security using Blockchain

Anmol Mishra[1], Dr. Charu Gandhi[2], Mayank[3], Aman Sharma[4]

[1]Jaypee Institute Of Information Technology, India, anmolmishra.jiit@gmail.com

[2]Jaypee Institute Of Information Technology, India, charu.gandhi@jiit.ac.in

[3]Jaypee Institute Of Information Technology, India, mayankapex3@gmail.com

[4]Jaypee Institute Of Information Technology, India, amanharitsh123@gmail.com

ABSTRACT

Blockchain is a secure and distributed ledger structure and each block is a cryptographic hash of some other factors like a timestamp. The chain is formed by linking the blocks and distributed data is present on multiple computers. The blockchain made system are more fault-tolerant and has high availability even in times of database failures. Though Linux is considered a benchmark for security, it is susceptible to various attacks. With the use of blockchain, Linux logs can be made more transparent among multiple root users and secured by hosting the logs on a decentralized Ethereum blockchain which is customized as per the requirement. It has been observed that in a scenario with multiple root users working simultaneously on same system, transparency between multiple root users can be compromised if some appropriate changes are made in few lines containing the history, Hence, with the help of Ethereum blockchain and log monitoring using log monitors, it is much easier to track the intrusions to identify the source of unauthorized access to logs or changes in them.

Key words : Linux Security, Blockchain, Smart Contracts, Ethereum, Linux Vulnerabilities

1. INTRODUCTION

In a decentralized system, data is maintained independent of any central hub, unlike centralized systems that govern the whole network. This leads to a system that is less vulnerable to hacking attempts that can be managed from any specific location, as the system in itself is a cluster rather a specific node. This makes it almost impossible to bring down the entire redundant cluster. Also, any failure to any single node restricted to a specific locations makes the uptime of the whole network more effective. In a decentralized system, data is distributed as well as synchronized on several locations and this redundancy adds up to the increased difficulty to hack the entire system. Due to distributed networks, people don't have to put their focus and faith on a single central authority. Moreover, there is a very less probability of a single point of

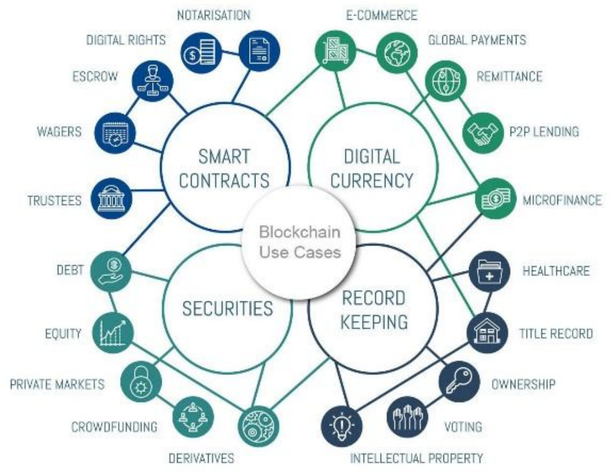


Figure 1: Various applications of Blockchain

failure due to its characteristics which makes the system more integral. In a blockchain [1], individual records, called blocks, are linked together in a single list, called a chain. It is used for recording transactions made with cryptocurrencies, for example Bitcoin, and have many other applications. Figure 1 shows various already existing solutions using blockchain. In a blockchain, each hash is calculated and verified, making it more trustworthy. Since all miners are involved in the evaluation and commit process, the blockchain systems have high data integrity. The underlined peer-to-peer system makes the blockchain a cost effective solution for a distributed environment. An important component of the blockchain is a Smart Contract, an account that is managed by code and easily programmable. Smart Contracts [3] are self-verifying due to automated possibilities, self-enforcing when the rules are met at all stages and tamper-proof, as no one can change what's been programmed. Security and Privacy via Optimised Blockchain [4] also supports for IoT based computing architectures as well. Lightweight scalable blockchain is one of the future application of blockchain as well which can help in communication in a network with limited bandwidth as well. International roaming services optimisation [5] is also an application of blockchain and smart contracts. Procurement fraud [6] is also an application of blockchain as well.

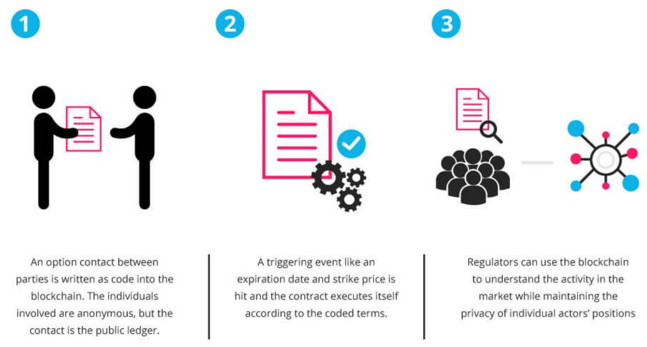

1
2
3
An option contact between parties is written as code into the blockchain. The individuals involved are anonymous, but the contact is the public ledger.
A triggering event like an expiration date and strike price is hit and the contract executes itself according to the coded terms.
Regulators can use the blockchain to understand the activity in the market while maintaining the privacy of individual actors' positions

Figure 2: Working of Smart Contract [7]

Figure 2 show above reflects the basic working of the Smart Contract. One of the popular blockchain based platform these days is Ethereum [8] Blockchain, which allows blocks to have code snippets that can take care of the transactions and helps to control the system more efficiently than the earlier version, Blockchain 1.0. Since, the attack mechanism is evolving in technological ecosystem, even one of the most secure computers that run on Linux is not secured. Vulnerabilities and threats can be found on machines and if they are exploited, they will create huge risks. Any penetration to the system can be made from inside or outside the organization and their attacks can be either active or passive. An attack that attempts to alter the system resources or affect their operation is called an active attack. Such attack compromise the system Integrity and Availability. The denial of service [9] is a very popular attack these days. An attack that attempts to learn or make use of information from the system is called a passive attack. It does not affect the system resources. Hence, it compromises the confidentiality. For example, the Wiretapping using the Wireshark [10]. Thus, the security of the Linux logs can be insured by hosting them on a decentralized ethereum blockchain. This is achieved by using a daemon that continuously monitoring the log files. Whenever a change is made, it creates a block on the existing blockchain. The daemon is immutable, hence, even a root user cannot delete it to evade the effect. The intrusions can be detected by a governing authority by simply checking the various blocks generated. This paper is organized as follows. Literature Survey is described in the Section II. The experimental setup is explained in Section III. Section IV elaborate the experimentation done and various outcomes. Section V concludes the paper and Section VI discusses the future scope of the work done.

2. LITERATURE SURVEY

The various Linux vulnerabilities like privilege escalation, Linux kernel hash algorithm exploitation i.e. time complexity of algorithm shoots up to O(n^2) from O(n) and IP Spoofing have been discussed in [11]. Some Linux distributions have default passwords which are vulnerable in reference to a guest account. This enables attackers to gain unauthorized access via secure shell or telnet service, which leads to the loss of integrity and purge of confidentiality. The process of Operating System hardening helps in securing a system by decreasing the number of accessible vectors of attack. It involves the dismissal of unwanted software like bloatware, unwanted usernames or logins and the disabling of unwanted services. Furthermore, there are some measures of hardening the Linux operating system. It mainly comprises of applying a patch to the kernel into the upstream, closing unwanted open network ports from iptables or other user firewalls, and setting up intrusion-detection/ prevention systems like snort. Various vulnerabilities like kernel omits, access ok checks and miss applying the functionality like get user are discussed in [12]. These do not verify the user entered indexes or other system variables which ensure the boundary of user-space is not crossed and permission check. The exploitation of such vulnerability is directly proportional to the policy adopted by security team, like code execution under presumptions (CVE2010-4347). Memory tagging systems, such as Raksha, can identify the untrusted input in case, the kernel tries to use these input and it is not from the required network or user process. With SecVisor, certain types of exploits are restricted which rely on taking over kernel control flow. In [13] the various common types of Linux vulnerabilities which may lead to attacks like Race Condition, memory corruption, denial of service, infinite loop, Integer Overflow, Null Pointer Dereferences, Divide by Zero, Use After Free, information disclosure, Buffer Overflow are defined using CVSS. The vulnerability score defined by CVSS can be used as reference to analyses the impact and severity of threat. The developers can prioritize their responses accordingly. Segmentation faults are the result of null pointer dereferences or bad memory access inside the kernel space. Divide by Zero is an error thrown by the operating system during division when the divisor is zero. As a reason for this mistake, the program continues to use the free pointer. Denial of service overflow is a common scenario which an attacker tries to generate since the request for a process is never expected to be fulfilled. The race condition is the processing of instructions in incorrect order between processes. Its output is dependent on sequence of execution, which creates loss of integrity. In reference to the Common Vulnerability and Exposures [14] a study of 157 cases mentioned in CVE was conducted. Various dimensions of the file system were taken into consideration to include common vulnerabilities based on causes and methods of implementation and its way of mitigation. A file system is prone to Denial of Service (DoS), system crashes, data leaks and the entire lockdown of the system. The file system in Linux is a composition of inodes that is similar to file pointer reference in Linux kernel which once compromised makes the whole file system easy to break. Moreover, if the memory objects are not freed, situations like memory leaks are easily available to exploit. Since an I/O call scheduler can reorder inode blocks, data rearrangement is easily possible, breaking the integrity of the file system model like JBD2. For optimization, kernel page cache is taken into consideration and page cache can block the file accesses as reported in CVE-2015-8839, page faults are increased by at least 35% than usual degrading the performance. The authors [15] discussed Linux Network Security and various network threats and vulnerabilities in great depth. The Physical and Application Level Security is also a major concern nowadays, along with the secured Operating System and Servers. Network logs help in securing the system as they can provide vital information about the network access. OpenSSH is still vulnerable to attacks and people doing some malicious activities and deleting the bash logs to erase the whole activity is a major concern. Another major problem discussed in this work the DNS Server. Hackers can attempt a DDoS attack on the DNS Servers (they are centralized as of now) which will create websites being down due to lots of traffic. The blockchain solution can provide a distributed DNS Server which will prevent DDoS attacks. The authors [16] presented research on Smart Contract developers and trending topics on analyzing this over StackOverflow. They found various use cases, security threats, blockchain in education and various other smart contract terminology. As the blockchain provides reliability, integrity, audit, and Bitcoin, Ethereum provides the distributed computational platforms, developers are more delving in this field to find the solution to problems using blockchain. Some issues are still there in the smart contract, one of them being some of the security vulnerabilities but as the community is growing most of these vulnerabilities are getting solved. The authors [17] made a hypervisor and named it SecVisor which helps in code integrity for the kernels of the Operating System. For the entire lifetime, hypervisor allows only those code to be executed at kernel which are user-approved. To prevent the system against attacks like code injection, etc this system will be helpful. Users will set own user policy to check against the code entered in the kernel and the SecVisor will check every time user enters a new code in the OS against the policy. The system checks that the code once checked and verified cannot be edited by the attacker at any later stage. There are currently three ways in which an attacker can inject attacks into the system:

1) Modularization Support: Allows privileged users to add code to the existing, running kernels.
2) Software Vulnerability: An attacker can exploit the software vulnerability.
3) Capable Devices: Capable Devices can write into the kernel memory. An attack was demonstrated by Becher.

Due to the increase in the complexity and size of the kernel OS, security vulnerability increases exponentially. The drawback of the SecVisor is that they are not able to prevent attacks done via control-flow. The authors [18] analyzed more than 80,000 security advisories. They collected data and information on known vulnerabilities and analyzed discovery date, disclosure date, exploit date and the patch availability date. They analyzed the data statistically and provided new parameters for distribution functions to extend the study. For security investments, few business decisions need to be taken and a model need to be built to control risk exposure. They also provided a tool to check the security and differentiate between patch exploit and patch-availability. The first security-oriented coding guidelines for Linux were developed in [19]. They also found around 290 Common Vulnerability and Exposures (CVE). The Linux Kernel development team use security practices like precondition validation, ensuring atomicity, capability validation, error handling, freeing resources, zeroing memory, etc. On the basis of the security advisory, the team developed a new mechanism for solving a security issue and to track the patch of the issue. The guidelines included various categories like, System Software Design Guidelines, General Code Guidelines, Privileged Code Guidelines, Concurrent/Parallel Code Guidelines, Performance Coding Guidelines, Resource Management Code Guidelines and Debug/ Log Code Guidelines. After analyzing the literature, the various Linux vulnerabilities as identified are privilege escalation, Linux kernel hash algorithm exploitation, IP Spoofing Vulnerability, Null Pointer Dereferences, Divide by Zero, Use After Free, Infinite Loop, Double Free, Buffer Overflow, Integer Overflow, Race Condition and Array Index Error. The privilege escalation vulnerability has a sub-domain of privilege sharing, which has an undetected vulnerability. It has been observed that in a scenario with multiple root users working simultaneously on the same system, transparency between multiple root users can be compromised if some in-appropriate changes are made in the system and Linux logs are tampered to hide the changes. The solution we proposed will store four types of log files into the smart contract so deleting the logs will not be possible for now. The power to deploy general-purpose computational code into a decentralized and trustworthy system attracted us to use blockchain as the right fit for our work.

3. EXPIREMENTAL DESIGN

The findings from the literature reflect that the current standing of the Linux system has its vulnerabilities and no existing solution is available to keep them in check. Hence by using the ethereum blockchain in conjunction with logs monitors it is possible to secure one of the current vulnerabilities i.e. Privilege Escalation. Here the ethereum blockchain acts analogous to immutable databases. Privilege Escalation is an act of exploiting a design flaw, bug, or configuration oversight in software application or operating system to obtain elevated access to resources that are normally protected from user or an application. The solution to this problem is achieved by developing an application that can support the whole architecture under a hood. Moreover, the application comprises of the main four components:

• Main application - The main application acts as an intermediate between collecting the data and writing it into the blockchain. It initiates the write queue and all the threads that watch over the log files for any new appended logs. It also contains hash addresses of the smart contracts on the blockchain and other important parameters like the path to abi.json file which has been generated using online solidity IDE named Remix. The working of main application is depicted in figure 3.

• Personal Ethereum blockchain - Ganache [20] is a one click Blockchain solution provided by Truffle Suite. After the execution of the Main application, Ganache blockchain is updated by the numerous transactions that are taking place every second. It continues until the total available logs are transferred to the blockchain.

• Log grabbers - Log grabbers are responsible for collecting important logs throughout the system and will place them on a write queue. All the threads monitor different files for the changes made and the log grabbers reflects it. We have used python [21] for log monitoring. It is a method to generate or keep logs in the blockchain to prevent any mistakes.

• Abstract Functions - They act as a wrapper around the web3 API [22] for writing and reading data from the blockchain. This wrapper API is the bridging element between the write queue and Ganache blockchain

4. METHODOLOGY

Main Application: This is a multi-threaded application that is coded in Python3. The application has three main parts:

i) A Daemon [23] for monitoring files and grabbing the latest log. Daemon is a process that runs in the background and performs a specified operation at predefined times or in response to certain events. Here the daemon is coded using python file handling function and multi-threading modules. A thread is initialized by the main application for each log file which writes a newly appended log to the work queue.

ii) Wrapper around web3.py API [22] to read and write data from the blockchain.Web3.py is a python library for interacting with ethereum. Its API is derived from the Web3.js JavaScript API and should be familiar to anyone who has used web3.js.

iii) Menu-driven program to start monitoring and collecting the changes.

A work queue is set up between all the running threads. As soon as new logs are appended threads register new data onto the work queue. Another writer thread continuously monitors the work queue and sends new data to the blockchain. The main application also provides an on-demand functionality of reading data from the blockchain using the initial menu. Figure 3 represents workflow of main application and Figure 4 depicts the Main Application Dashboard. The figure 5 represents the updated logs being written on the blockchain. The blockchain used to realize the solution is created using Ganache [21]. It is a one-click blockchain solution provided by Truffle Suite. It is a personal blockchain for ethereum development that can be used to deploy contracts, develop applications, and run tests. It is available as a desktop application and provides 10 addresses and ethereum test accounts to use with a huge amount of ether. It connects well with Online Remix IDE [22], which makes it a perfect solution to deploy smart contracts on the go. The smart contracts for designing blockchain are created using Solidity [25]. Solidity is used mainly due to its advantages like security, less testing required, easy to read, easy to implement, bounds and overflow checking, strong typing, decorators available. Logs are generated and they act as one of the transaction commodities i.e. any transaction in the block occurs only when the log is changed. The log changes lead to

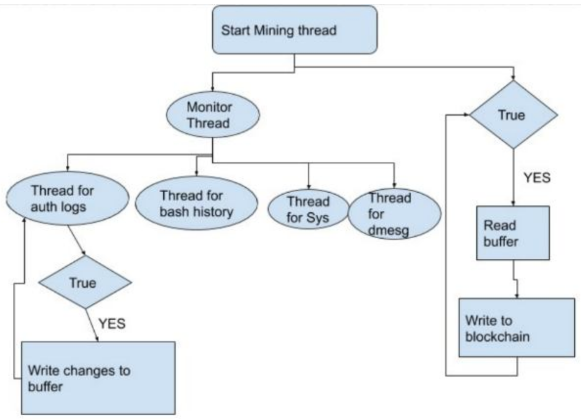
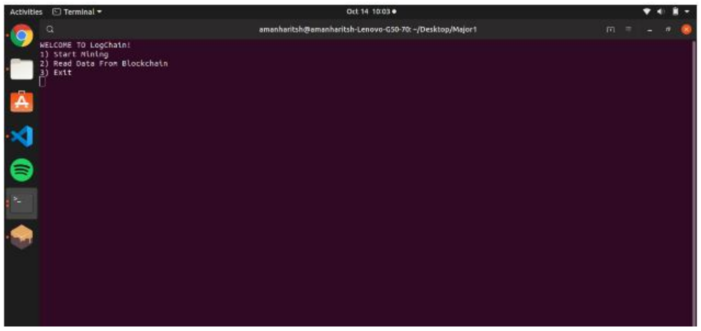
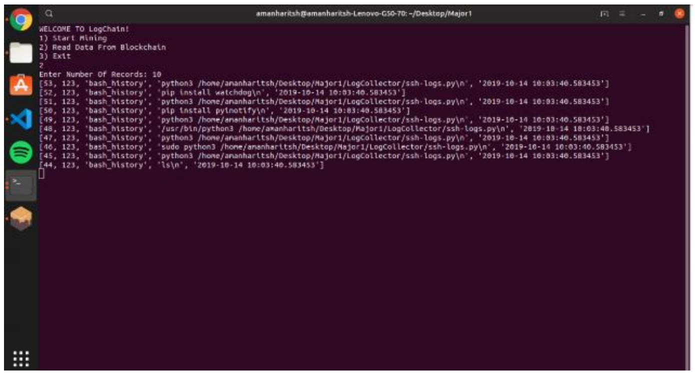


Figure 3: Workflow of Main Application



```
WELCOME TO LogChain!
1) Start Mining
2) Read Data From Blockchain
3) Exit
```

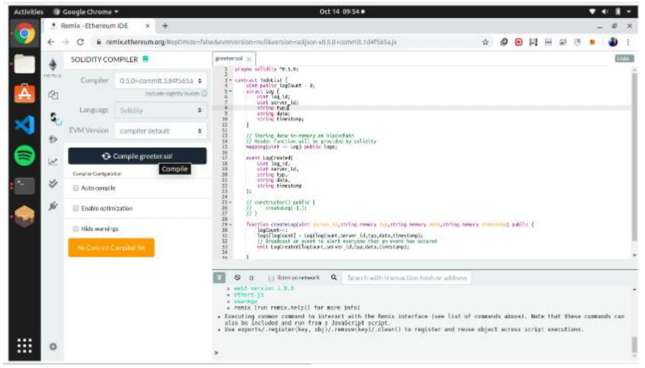
Figure 4: Main Application Dashboard



```
WELCOME TO LogChain!
1) Start Mining
2) Read Data From Blockchain
3) Exit
2
Enter Number Of Records: 10
[53, 123, 'bash_history', 'python3 /home/amanharitsh/Desktop/Major1/LogCollector/ssh-logs.py\n', '2019-10-14 10:03:40.583453']
[52, 123, 'bash_history', 'pip install watchdog\n', '2019-10-14 10:03:40.583453']
[51, 123, 'bash_history', 'python3 /home/amanharitsh/Desktop/Major1/LogCollector/ssh-logs.py\n', '2019-10-14 10:03:40.583453']
[50, 123, 'bash_history', 'pip install pyinotify\n', '2019-10-14 10:03:40.583453']
[49, 123, 'bash_history', 'python3 /home/amanharitsh/Desktop/Major1/LogCollector/ssh-logs.py\n', '2019-10-14 10:03:40.583453']
[48, 123, 'bash_history', '/usr/bin/python3 /home/amanharitsh/Desktop/Major1/LogCollector/ssh-logs.py\n', '2019-10-14 10:03:40.583453']
[47, 123, 'bash_history', 'python3 /home/amanharitsh/Desktop/Major1/LogCollector/ssh-logs.py\n', '2019-10-14 10:03:40.583453']
[46, 123, 'bash_history', 'sudo python3 /home/amanharitsh/Desktop/Major1/LogCollector/ssh-logs.py\n', '2019-10-14 10:03:40.583453']
[45, 123, 'bash_history', 'python3 /home/amanharitsh/Desktop/Major1/LogCollector/ssh-logs.py\n', '2019-10-14 10:03:40.583453']
[44, 123, 'bash_history', 'ls\n', '2019-10-14 10:03:40.583453']
```

Figure 5: Writing Data to Blockchain

a generation of blocks forming a blockchain. Smart Contracts are written using solidity and act as a schema to store data sent from the Main application to the blockchain. Figure 6 depicts Smart Contract created using Solidity. To compile and deploy smart contracts on personal blockchain created using Ganache, an online Remix IDE [24] is used. It generates all necessary information like address of deployed contract, abi.json, etc. after the deployment of the Smart Contract. Figure 7 shows the Remix IDE used to compile and deploy the Smart Contract.



```
pragma solidity ^0.5.0;

contract TodoList {
    uint public logCount = 0;
    struct Log {
        uint log_id;
        uint server_id;
        string typ;
        string data;
        string timestamp;
    }

    // Storing data in-memory on blockchain
    // Header function will be provided by solidity
    mapping(uint => Log) public logs;

    event LogCreated(
        uint log_id,
        uint server_id,
        string typ,
        string data,
        string timestamp
    );

    // constructor() public {
    //     createLog(-1,);
    // }

    function createLog(uint server_id,string memory typ,string memory data,string memory timestamp) public {
        logCount++;
        logs[logCount] = Log(logCount,server_id,typ,data,timestamp);
        // Broadcast an event to alert everyone that an event has occured
        emit LogCreated(logCount,server_id,typ,data,timestamp);
    }
```

```
• web3 version 1.0.0
• ethers.js
• swarmgw
• remix (run remix.help() for more info)
• Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can
  also be included and run from a JavaScript script.
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
```

Figure 6: Solidity-Smart Contract

Figure 8 depicts the process of securing various Linux logs using the main application, blockchain and Smart Contracts as defined above.

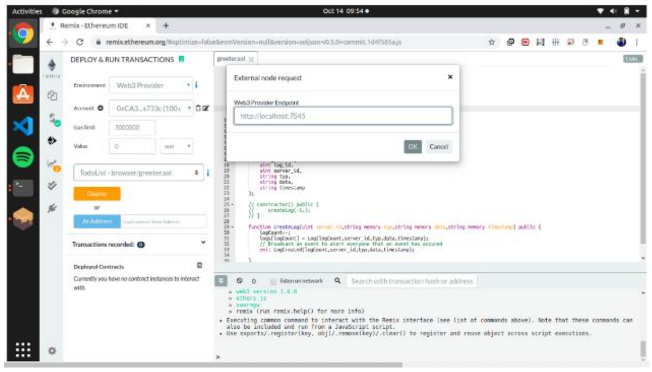

DEPLOY & RUN TRANSACTIONS
Environment Web3 Provider
Account 0xCA3...a733c (100 ether)
Gas limit 3000000
Value 0 wei
TodoList - browser/greeter.sol
Deploy
At Address
Transactions recorded: 0
Deployed Contracts
Currently you have no contract instances to interact with.
External node request
Web3 Provider Endpoint
http://localhost:7545
OK
Cancel

Figure 7: Remix IDE

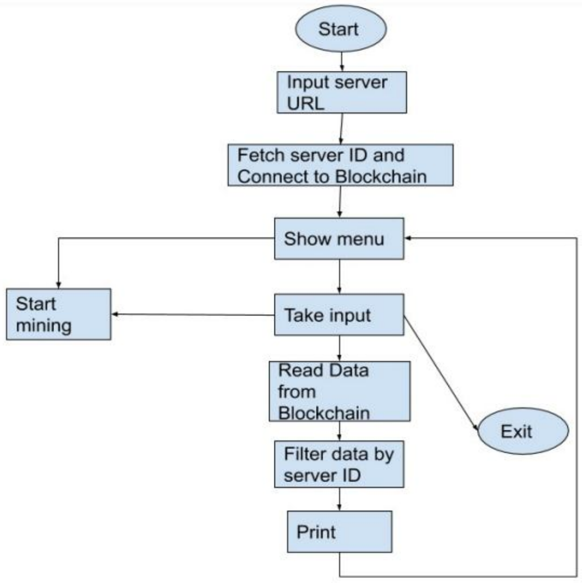


Figure 8: Process of securing logs

From the experiment, the various vulnerabilities in action among multiple root users in a Linux environment were found. The logs stored in /var/log/auth.log, /.bash history, /var/log/httpd/ and /var/log/dmesg were being secured by constant monitoring. Whenever other user make a change in the log file at any point of time, a transaction was generated. Due to fix architecture of files i.e. filesystem in Linux, these files are independent of the Linux distribution and solution can be widely acceptable. The proposed solution makes a transaction in blockchain whenever logs are altered. Thus, preventing any unauthorized user to escalate to the root level and hence, using the privileges of the root user.

5. CONCLUSION

This paper outlines the working of a smart contract based ethereum system and its use to tackle various security vulnerabilities. The application developed in this work can be used for securing various log files like secure shell logs, disk logs, network logs, etc. of a Linux system by moving them on a decentralized blockchain. The proposed solution also improves the transparency of the system.

6. FUTURE SCOPE

Since we can mine the log files and put that on the distributed blockchain solution, our future goal is to find more vulnerabilities present in the Linux system and kernel and try to solve them using blockchain. There is also a scope for securing specific ports and sockets for the system. This will be explored in the future research.

REFERENCES

1. Techterms.com. 2020. Blockchain Definition. [online] Available at: https://techterms.com/definition/blockchain [Accessed 18 March 2020].
2. Medium. 2020. 6 **Emerging Categories For Blockchain Use Cases.** [online] Available at: https://medium.com/@sergiomarrero/6-emergingcategories-for-blockchain-use-cases-4650f824d130 [Accessed 19 March 2020].
3. Blockgeeks. 2020. **What Are Smart Contracts? [Ultimate Beginner'S Guide To Smart Contracts].** [online] Available at: https://blockgeeks.com/guides/smart-contracts/ [Accessed 18 March 2020].
4. Monica Thomas and Dr. Varghese S Chooralil "**Security and Privacy via Optimised Blockchain",** Published in International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), ISSN 2278-309, Volume 8, No. 3, May- June 2019 https://doi.org/10.30534/ijatcse/2019/14832019
5. Mark Renier M. Bailon, Lawrence "**International Roaming Services Optimization Using Private Blockchain and Smart Contracts"**, Published in International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), ISSN 2278-309, Volume 8, No. 3, May-June 2019 https://doi.org/10.30534/ijatcse/2019/32832019
6. A. Thio-ac, A. K. Serut, R. L. Torrejos, K. D. Rivo, J. Velasco "**Blockchain-based System Evaluation: The Effectiveness of Blockchain on E-Procurements**" Published in International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), ISSN 2278-309, Volume 8, No. 5, September-October 2019 https://doi.org/10.30534/ijatcse/2019/122852019
7. Ganpatigraphics.com. 2020. Bitcoin Related Jobs Ethereum Watch Contract – Ganpati Graphics. [online] Available at: http://ganpatigraphics.com/library/nem-market/bitcoin-related-jobsethereum-watch-contract/ [Accessed 19 March 2020].
8. ethereum.org. 2020. Home — Ethereum.Org. [online] Available at: https://ethereum.org/ [Accessed 19 March 2020].
9. En.wikipedia.org. 2020. **Denial-Of-Service Attack**. [online] Available at: https://en.wikipedia.org/wiki/Denial-of-service attack [Accessed 19 March 2020].
10. Wireshark.org. 2020. **Wireshark · Go Deep**.. [online] Available at: https://www.wireshark.org/ [Accessed 19 March 2020].
11. Niu, Shuangxia Mo, Jiansong Zhang, Zhigang Lv, Zhuo. (2014**). Overview of Linux Vulnerabilities**. 10.2991/scict-14.2014.55.
12. Haogang Chen, Yandong Mao, Xi Wang, Dong Zhou, Nickolai Zeldovich, and M. Frans Kaashoek. 2011**. Linux kernel vulnerabilities: state-of-the-art defenses and open problems.** In Proceedings of the Second Asia-Pacific Workshop on Systems (APSys '11). ACM, New York, NY, USA, Article 5, 5 pages.
13. Supriya R., Geetika M., Shagun "**Analysis of Linux Kernel Vulnerabilities** " Published in Indian Journal of Science and Technology (IJST), ISSN: 0974-5645, Vol 9(48), Issue: 10.17485, December 2016. https://doi.org/10.17485/ijst/2016/v9i48/105819
14. Miao Cai, Hao Huang, and Jian Huang. 2019. **Understanding Security Vulnerabilities in File Systems**. In Proceedings of the 10th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys '19). ACM, New York, NY, USA, 8-15.
15. Mukesh Kumar Mishra and Dinesh Goyal, "**Security Analysis in Open Source Linux Network**", International Journal of Computer Science and Network Security, Vol.14, No.8, August 2014.
16. Afiya Ayman, Amna Aziz, Amin Alipour, Aron Laszka,**" Smart Contract Development in Practice: Trend, Issues, and Discussions on Stack Overflow",** arXiv: 1905.08833v1[cs.CY] May 15, 2019.

17. Arvind Seshadri, Mark Luk, Ning Qu, and Adrian Perrig. 2007. **SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes**. In Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles (SOSP ’07). ACM, New York, NY, USA, 335-350. https://doi.org/10.1145/1294261.1294294
18. Stefan Frei, Martin May, Ulrich Fiedler, and Bernhard Plattner. 2006. **Large-scale vulnerability analysis**. In Proceedings of the 2006 SIGCOMM Workshop on Large-scale attack defense (LSAD ’06). ACM, New York, NY, USA, 131-138.
19. Mokhov S.A., Laverdiere MA., Benredjem D. (2008) **Taxonomy of Linux Kernel Vulnerability Solutions**. In: Iskander M. (eds) Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education. Springer, Dordrecht.
20. Truffle Suite. 2020. **Ganache — Truffle Suite**. [online] Available at: https://www.trufflesuite.com/ganache [Accessed 18 March 2020].
21. Python.org. 2020. **Welcome To Python.Org**. [online] Available at: https://www.python.org/ [Accessed 19 March 2020].
22. Web3py.readthedocs.io. 2020. **Web3.Py** — Web3.Py 5.7.0 Documentation. [online] Available at: https://web3py.readthedocs.io/en/stable/ [Accessed 18 March 2020].
23. En.wikipedia.org. 2020. **Daemon (Computing)**. [online] Available at: https://en.wikipedia.org/wiki/Daemon (computing) [Accessed 18 March 2020].
24. Remix.ethereum.org. 2020. **Remix - Ethereum IDE**. [online] Available at: https://remix.ethereum.org/ [Accessed 18 March 2020].
25. Solidity.readthedocs.io. 2020**. Solidity — Solidity 0.6.4 Documentation**. [online] Available at: https://solidity.readthedocs.io/en/v0.6.4/ [Accessed 18 March 2020].