

# Threat Detection on Linux Systems Using OSquery

Yogesh Chandrakant Jadhav<sup>1,\*</sup>, Vaibhav Kisan Kadam<sup>1</sup>, Shubham Ramesh Kanase<sup>1</sup>,  
Srijita Bhattacharjee<sup>2</sup>

## Abstract

We have made an EDR tool for Linux Systems using Facebook open-source project OSquery. Making our own EDR tool rather than using a commercial EDR tool helps us gain knowledge about the platform and the security aspect of the platform and gives us the capabilities to detect and investigate security events. In our method, we are collecting the logs on the central server and then we are using these logs to generate the correlation between events which are happening on different Linux endpoints. These events are different events which are taking place in the Linux system like file events, socket events, process events, etc. These events are automatically detected and categorized into different attack vectors to remediate in future. Due to continuous monitoring, we get these events after a specific interval which makes the detection real-time. Users can provide on-the-fly configuration which makes the tool more responsive and accurate and does not collect the garbage data which are not required. We are also providing container security which is a new feature in the open-source tools. In our method, we have designed a system in such a way that we can scale the system and add a scalable amount of nodes in a single deployment.

**Keywords:** Endpoint detection and response, OSquery, Fleet server, Linux, malware

## INTRODUCTION

Continuously monitoring the system and detecting the threats helps the organization to be ahead of the malicious attacks executed on their organization. These attacks include the attacks crafted for accessing confidential information which can help the malicious actor to defame the organization or to extract the key insights from the data which he/she can use against the organization. This information is extracted from non-public networks and private nodes. Linux is a very popular choice among the enterprise customers and various government bodies. We do not have many choices for continuous monitoring and threat detection on Linux powered machines. Our tool is made in such a way that it can assist organizations and government affiliates to catch, block, remediate and investigate threats who are targeting Linux platforms.

### \*Author for Correspondence

Yogesh Chandrakant Jadhav

<sup>1</sup>Student, Department of Computer Science and Engineering,  
Pillai HOC college of Engineering and Technology, Rasayani,  
Maharashtra, India

<sup>2</sup>Assistant Professor, Department of Computer Science and  
Engineering, Pillai HOC College of Engineering and  
Technology, Rasayani, Maharashtra, India

Received Date: April 26, 2022

Accepted Date: April 29, 2022

Published Date: May 04, 2022

**Citation:** Yogesh Chandrakant Jadhav, Vaibhav Kisan Kadam, Shubham Ramesh Kanase, Srijita Bhattacharjee. Threat Detection on Linux Systems Using OSquery. Journal of Advances in Shell Programming. 2022; 9(1): 1–5p.

## Continuous Analysis

This is the process in which we try to continuously gather the information of the system and save it in the data server to perform the historical analysis afterwards to gain the information about activities in the past.

## Proactive Threat Detection

In this method we try to search through logs and create detection for malicious activities before they are performed in the environment by a malicious actor.

System logs tell us about the events that take place in the system along with the information like

who initiated it and at which time it had been executed. But the system logs are unable to provide us with the granular information about the events taking place in the system. This information is very important for the security analyst for detecting the malicious behavior happening in the environment. We have made a tool which can help analysts to do informed security investigations on contextualized data and invent remediation for the threat in future.

## EXISTING TOOLS AND LIMITATIONS

In the current scenario, there are very few tools available which assist system administrator and security team to log the information to a specific central server to store it and analyze it in future. These tools are classified as Security Investigation and Event Management (SIEM).

There are many utilities in Linux systems which help to collect the information about the activities and events that took place in the system but mostly used are system logs or Auditd.

**Linux System Logs:** Linux logs provide a timeline of events for the Linux operating system, applications and system and are a valuable troubleshooting tool when you encounter issues. When issues arise, analyzing log files is the first thing an administrator needs to do. Following are the logs which are made available by the Linux system logs which help us to analyze the types of activities being performed within the system.

- *Syslog and messages:* Global system activity, including startup messages.
- *Authentication logs:* Security modules like Pluggable authentication module.
- *Kernel logs:* Includes the logs of kernel events, errors, and warnings.
- *Cron logs:* information about executing cron jobs.
- *Application specific logs:* logs present at `/var/log/*`.
- *Auditd Logs:* The Linux Auditing System is a native feature to the Linux kernel that collects certain types of system activity to facilitate incident investigation. It helps us to get even the log specific syscall usages. If we want specific items according to our interest from the system, we can write specific rules for that and include it in the `auditd.conf` file. Auditd custom logging.
- *Control Rules:* These rules are associated with controlling auditing daemon.
- *File Watch Rules:* These rules are associated with file monitoring; these files may include binaries also.
- *System call rules:* Allow logging of system calls that any specified program makes.

Both of these techniques have their own set of problems:

- Linux System Logs have a limitation that this utility is unable to contextualize user information and it does not tell us specifically who has performed specific action. If the user is logged in as a normal user and if he/she escalated their privileges to root, then all the activities are logged as the root user had performed these activities. This makes blind-spot, which leads to not getting the actual information about who has performed which action.
- Auditd Logs: they are better than Linux system logs; they are very system aware and facilitate to write our own custom rules to audit the system which helps in detecting threat. It does tell us about which user has performed which activity but it fails to contextualize the information, which analyst can use to detect the threat.

Organizations need to depend on some Linux utilities like Syscall, polling by log server or need to rely on commercial tools for parsing and collecting the logs. Due to this, log interruptions happen and that leads to decrease the capabilities of the logging system to optimally collect and analyze the logs that are generated by the system and figure out the malicious behavior from it.

## PROPOSED SOLUTION AND SYSTEM ARCHITECTURE

Our tool provides following features:

- Information collected from nodes is contextualized.
- Information is logged and stored in a central location.
- Capability of searching and analyzing through pre-logged data.

- Triggers security alerts and provides capability of real-time monitoring.
- Capability to on-board devices into a central management server.
- It also features remediation and blocking capabilities.

### **Important Components of the System**

Let us analyze the architecture of our system. The most important components of the system which are shown in Figure 1 are:

- Endpoints/clients which contains OSquery,
- Central server,
- Central Database,
- Event processor,
- Rule Engine, and
- Web UI.

EDR client-side application powered by OSquery runs on the Linux systems on which threat is to be detected or blocked [1, 2]. All Linux endpoints send data to the central fleet server. All Linux endpoints send data to the central logging server that is the fleet server. OSquery only sends the data when a change in collected logs is observed. Fleet server helps to query data as per our requirement. When the data is collected, the rule engine processes this data against threat activities on the basis of rules written.

### **Endpoints**

Endpoints are the end nodes of the system which have OSquery running in the background. This is the machine on which continuous monitoring is going on. OSquery is installed on every endpoint, and this facilitates EDR to gather all the logs and system information to perform real time analysis to detect and block threats. Every endpoint sends the logs when it detects the change and not on interval of time.

### **Fleet Server**

Fleet server is an in-house built central logging server which can be deployed as a binary in any system. Fleet server helps us to manage endpoints, databases and the UI API server all together. Fleet server is a bare bone HTTPs server made using C++ which provides maximum scalability and performance.

### **Database**

Database is a central database which stores all the information and processed logs which help us analyze the data in future. The database used ensures that anyone reading data from the database will always receive consistent results. Database offers us to query it at any point of time which makes it reliable within the application.

### **API Server**

API server works on request and serves mechanism and helps analysts to interact with the central logging server fleet. API server provides capabilities to the developer as well as analyst to query data from database, manage endpoints, and enroll/re-enroll and query ad-hoc data directly from the Linux endpoints.

### **UI Application**

UI application provides the GUI web front-end for the analyst to detect threat and respond to it, this front-end provides capabilities of configuring the system as per needed for this purpose; UI application uses Fleet APIs.

### **Rule Engine**

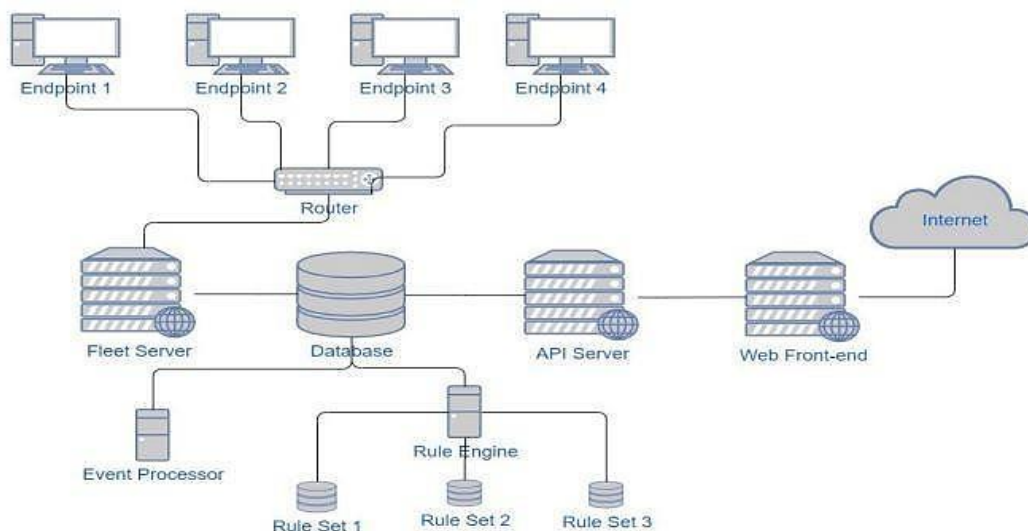
Rule engine uses the incoming logs which are extracted from Linux endpoints and processes them to generate threat alerts for the EDR. These attacks are mapped with MITRE ATTACK matrix [3]. Analyst uses this alert for investigation purposes to block them in future.

### System Performance and Requirements

Our approach to make this EDR is highly scalable and made using modular methods. System is made in this way so that it can handle a large amount of enrolled machines in a single deployment. The EDR Client-side agent consumes around 10–15% of CPU of the system and if this CPU consumption hits the line of 25% for more than 9 sec, then the agent will restrict and restart itself.

### Threat Detection Capabilities

- *Rule Engine for threat detection:* Custom rules can be written and used to identify threat activities. These custom rules can be fed to the engine in real time.
- *Scheduled Queries for threat analysis:* We can pass the SQL based queries to the system to identify malicious behavior, but these queries are scheduled for specific time and are not part of continuous monitoring.
- *Yara Rules [4]:* We have provided support of Yara rules out of the box, which can help analysts to write and validate the memory of the running process to identify the malicious behavior of the process.



**Figure 1.** System architecture.

### TYPE OF ATTACKS

When we are analyzing the behavior of the malicious attack, some attacks seem to behave abnormally which leads to their easy detection. But in this process of malware behavior analysis, we witness some malware which do not show any suspicious behavior like botnet and minor. In these two techniques, the agent allows the attacker to use the CPU and memory bandwidth of the victim's machines. However, we can detect the initial movement as we can classify it into suspicious behavior. To get more information about the attack, we need to search for the signature of the known botnets and miners in the threat database.

In the following part we will discuss the known vulnerabilities and how our tool helps to detect it.

**CVE-2019-2725 [5]:** This CVE provides the information about the well-known vulnerability in Oracle Web logic server which allows an attacker to gain control of a vulnerable host by executing arbitrary OS commands. The attacks start by sending malicious HTTP POST requests, the payload includes the information which performs malicious action, that grants access to the control server's local shell. When we try to analyze this attack behavior, we get to know about following events: Local shell, that is bash/dash/zsh/sh initiated by an internet server Local shell performing socket transactions, for e.g., giving reverse shell, bind/accept events are triggered for bind-shell, here we can predict that something bad has happened by looking at each event that is triggered. Because in the normal routine of the online server it does not spawn shells, and we also know that an online server is

not supposed to do network activities [6]. In the post exploit stage, if the user tries to escalate its privileges, then we get a user event where change in previous UID and current UID is seen. This change in UID, points towards the situation in which the attacker had gained access to different users. If the UID gets changed to 0 then it confirms that the user has gained access to the root privileges. If the attacker tries to access persistence, we get to know about this by getting various events depending upon how he tries to do the persistence [7–9].

*Event associated with Cron job:* File events will get triggered, which tells us about the cron file having changed. Process events will get triggered which tells us that crontab command has been used.

*Event associated with SSH configuration:* File events will get triggered which tells us about “.ssh/\*” or sshd config has been changed.

*Events associated with Bashrc:* File events will get triggered, which tells us about the file “.bashrc” has changed. Each event is sufficient to tell us that something suspicious is happening in the system. Many malwares are made in this way; they are polymorphic in nature. Our approach should be into behavior aspect of the malicious binary, and hence in our approach, we are expecting to detect the fair amounts of malware operating in the system [10–12].

## CONCLUSION

We have designed a new system for threat detection in Linux machines. The system we designed is built in-house and uses open-source tools. This tool is useful for organizations and government bodies to detect threats to their systems. Our tool has capabilities of detecting threats like lateral movement and pivoting.

## REFERENCES

1. Facebook. (2019). SQL powered operating system instrumentation, monitoring, and analytics. [Online]. Available from: <https://github.com/osquery/osquery>
2. Rapid7 Blog. (2019 Jul). Introduction to osquery for threat detection and DFIR. [Online]. Available from: <https://www.rapid7.com/blog/post/2016/05/09/introduction-to-osquery-for-threat-detection-dfir/>
3. MITRE. (2021) ATT&CK. [Online]. Available from: <https://attack.mitre.org/>
4. YARA. (2021). [Online]. Available from: <https://yara.readthedocs.io/en/stable/>
5. Oracle. (2019). Oracle Security Alert Advisory-CVE-2019-272. [Online]. Available from: <https://www.oracle.com/security-alerts/alert-cve-2019-2725.html>.
6. Reed T, Grenier M. (2017 Nov 2). Osquery—Windows, MacOS, Linux Monitoring and Intrusion Detection. [Online].
7. Kieseberg P, Neuner S, Schrittwieser S, Schmiedecker M, Weippl E. Real-Time Forensics through Endpoint Visibility. In: Matoušek P, Schmiedecker M, editors. Digital Forensics and Cyber Crime. ICDF2C 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Cham: Springer; 2018; 216. [https://doi.org/10.1007/978-3-319-73697-6\\_2](https://doi.org/10.1007/978-3-319-73697-6_2)
8. Hurless C. (2019 Sep 10). Exploring Osquery Fleet and Elastic Stack as an Open-source solution to Endpoint Detection and Response. [Online]. SANS Institute Reading Room site.
9. Yin H, Song D, Egele M, Kruegel C, Kirda E. October. Panorama: capturing system-wide information flow for malware detection and analysis. In Proceedings of the 14th ACM conference on Computer and communications security. 2007; 116–127.
10. Gržinić T, González EB. Methods for automatic malware analysis and classification: a survey. Int J Inf Comput Secur. 2022; 17(1–2): 179–203.
11. Kazdagli M, Caramanis C, Shakkottai S, Tiwari M. Early and Robust Malware Detection in Enterprise Networks.
12. Maxwell Dondo, Madeena Sultana, Grant Vandenberghe. Malicious activity detection: An analysis of current tools and methodologies for network defence in operational networks. Reference Document DRDC-RDDC-2021-D078. Canada: Defence Research and Development Canada; 2021.