Vajra: An Indigenous Solution for Endpoint Detection and Management

IIT Bombay

A. Introduction

Continuous monitoring and proactive threat detection enable organizations to stay ahead of the crafty attacks executed by malicious actors. These attacks may be done to extract critical information from a private network or extort money or cause reputational damages to the end target. Linux and Windows-powered servers or desktops are popular choices amongst enterprise customers and various government organizations. There are a limited set of tools and practices to enable continuous monitoring and threat detection on these platforms. Our Vajra tool aims to help organizations and enterprises to develop capabilities to identify, investigate and remediate security threats in the networks. Vajra has the following two main functionalities.

- **Continuous Monitoring**: Process of continually gathering granular system information and recording it to perform historical analysis on the data and to understand activities performed in the past. System logs are critical for security analysis to detect malicious behaviors. Our tool enables gathering contextualized data which is important for getting better visibility of the system activities and correlating them for any malicious activity.
- **Proactive Threat Detection**: This is the process of systematically analyzing the system logs and detecting any malicious activity in real-time. Our tool performs real-time threat detection using MITRE ATT&CK and GTFOBins frameworks and provides alerts. The tool has the capability for live threat hunting and provides remediation capabilities.

B. Vajra Solution and Architecture

The proposed tool will enable security teams to achieve the following capabilities.

1. Collection of contextualized information from endpoints
2. Central logging and storage of data
3. Search and analyze the collected data
4. Create security alerts and real-time monitoring
5. To enroll/on-board devices into fleet management server
6. Remediation and blocking.

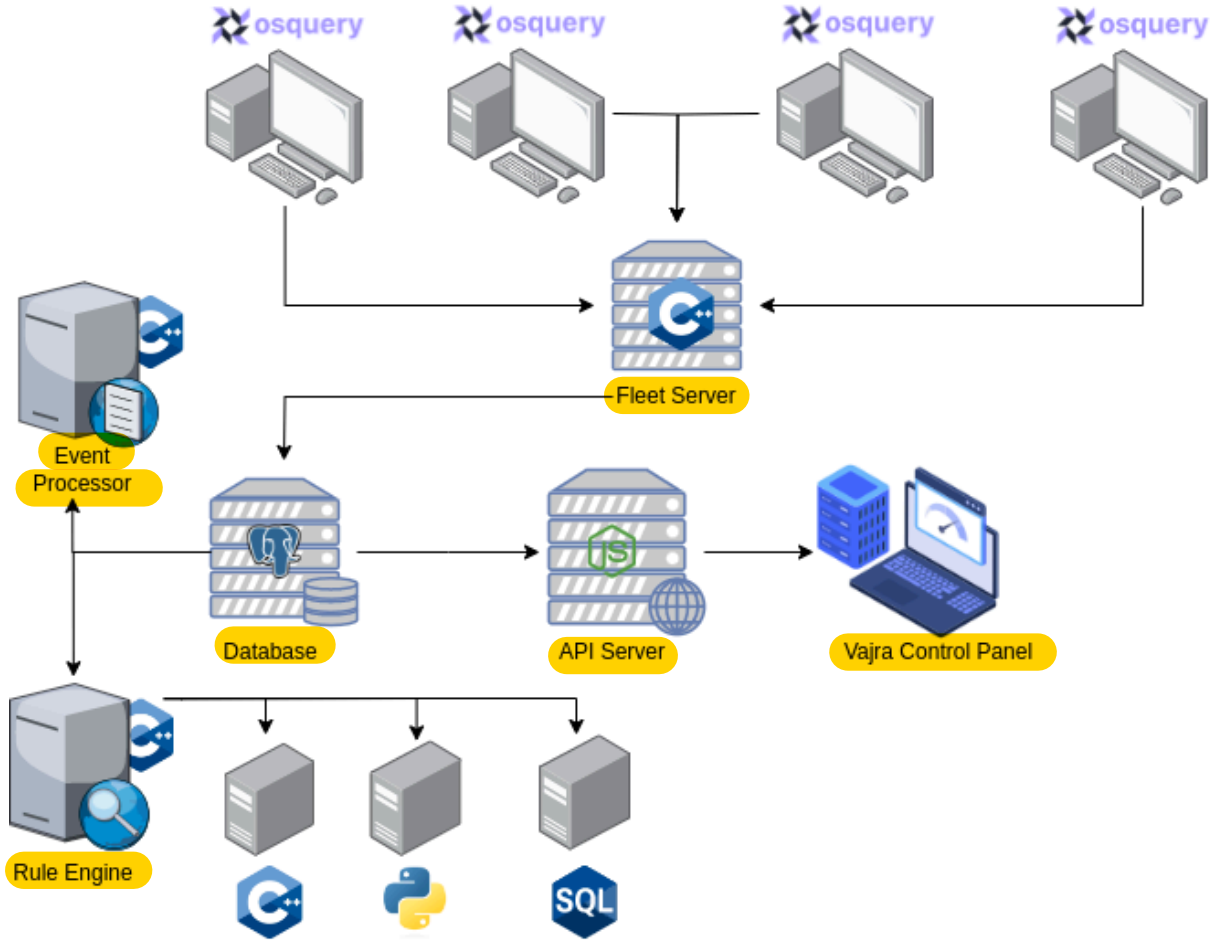

Fig. Vajra architecture

The Vajra system consists of the following main components:

1. Vajra agent for log collection
2. Fleet server
3. Database
4. Event processor
5. Rule Engine
6. Admin console (web front-end)

Below we provide details of each of these components.

1. **Vajra agent (https://github.com/VajraSecurity/vajra-client)**

Vajra agent is powered by Osquery. Vajra agents run on endpoints where security attacks and breaches are to be detected or blocked. All endpoints system logs to a fleet server based on a predefined configuration based on a delta, i.e., only if there is any change in collected logs, they will be forwarded to the fleet server.

Osquery is a framework for instrumenting operating systems for Windows, OS X (macOS), and Linux. The tools make low-level operating system analytics and monitoring more efficient and user-friendly. osquery transforms an operating system into a high-performance relational database. This enables you to create SQL queries to investigate operating system data. SQL tables in osquery describe abstract ideas like operating processes, loaded kernel modules, open network connections, browser plugins, and hardware events. This information is collected in the form of Process events, sockets events and file events. To do this, we have custom-built the osquery by using eBPF for Linux and Kernel drivers for Windows.

Our extension of Osquery allows us to get granular information at intervals that can be customized by the user based on the requirements and resources consumed. Even if the information is gathered periodically, no event is missed as logs are 'evented', i.e., every activity is captured as they happen, and all the information that is captured since the last query is sent when a new query is made. We also considered the possibility that the system would restart and the PIDs would be reset, in that case also, we would be able to relate the information and extract useful information.

The Vajra system is compatible with Linux and Windows. The best part about our Vajra is that it is both agnostic to kernel versions in Linux and for Windows, it has full compatibility with the latest version. Our Vajra agent can be used in stand-alone mode to collect system logs for Linux and Windows systems. The Vajra agent sends logs in JSON format, which can be pushed to any third-party SIEM tools over an HTTPS connection.

Vajra agent installation is as simple as running a single script. Installation of Vajra agent takes care of simplicity to install and set up the agent. Even if the person installing the agent is a non-technical person, the installer script takes care of an easy and hassle-free installation process. The overall installation process takes less than 30 seconds.

Fig. Agent installation process in Windows 10

1.1 System Performance and Requirements:

The Vajra agent is designed to be lightweight and efficient, consuming minimal system resources such as CPU, memory, and disk space. Its performance and resource

requirements depend on factors like query complexity, query scheduling, and real-time query usage. Regular updates and proper query optimization strike a balance between resource usage and data collection. Osquery runs as a background process with low overhead, effectively monitoring endpoints on various operating systems. It communicates securely with a central server, generating logs and storing query results locally before transmission. Overall, osquery is a performant EDR tool, offering effective endpoint monitoring without significant resource impact.

Vajra is developed in a modular approach and is highly scalable. The Vajra system can easily handle a large number of endpoints in a single deployment. The agent consumes around 10−15% of the CPU of the system and would be interrupted and restarted if it consumes more than 25% of resources for more than 9 secs.

2. **Fleet Server**

We are using a custom-built fleet server that can be deployed as a single binary across the environment, which helps us manage individual endpoints, the database, and the API server altogether. This fleet server is a bare bone HTTPs server written in C++, providing maximum scalability and performance.

3. **Database**

Our application is using a relational database that provides database guarantees, enabling the data at any point to be reliable in the application. We are using PostgreSQL for this purpose.

4. **Event Processor**

The event processor is the component of the system that primarily processes raw input in order to construct a process tree. The fundamental job of an event processor is to connect the dots between the processes running in the system to form a tree and to attach relevant information to the event.

5. **Rule Engine**

Rule Engine enables Vajra to analyze incoming logs, which are collected from individual endpoints, and send threat events that match rules to Vajra as alerts. We mapped these attacks with the MITRE ATT&CK matrix. The analysts can then investigate these alerts. They can also take appropriate action to block them in the future.

6. **Admin console (https://getvajra.com)**

Admin console uses Fleet APIs and provides additional capabilities to enable analysts to use Vajra and aid in security investigations easily. The UI also allows for custom Configuration, endpoint enrollment, live machine querying, and much more.

C. Capabilities of Vajra

The main capabilities of Vajra are as follows:

1. Threat Detection
2. Threat Hunting
3. Threat Alert management
4. Node/Inventory management
5. Isolation of malicious nodes
6. Blocking and Terminating malicious processes
7. Container visibility and analysis

1. Threat Detection capabilities:

Threat detection has four components.

- Rule Engine: Rules are written and can be used to identify threat activity. Rules provided to rule engine runs in real-time. These runs cover most of the Tactics, Techniques, and Procedures outlined by the MITRE ATT&CK framework. Also, SOC analysts can add rules in Python language directly from the Web front end. SOC analysts can add custom rules using User Interface without writing any code, just by using UI components.
- Scheduled Queries: SQL-based queries to identify threat data, but these queries are run in a specified time schedule and are not real-time detections.
- Yara Rules: OSquery supports Yara rules out of the box, we can write and validate the memory of running processes to identify threat events.
- Integration to other threat intelligence modules like AbuseDB, NIST database, etc.

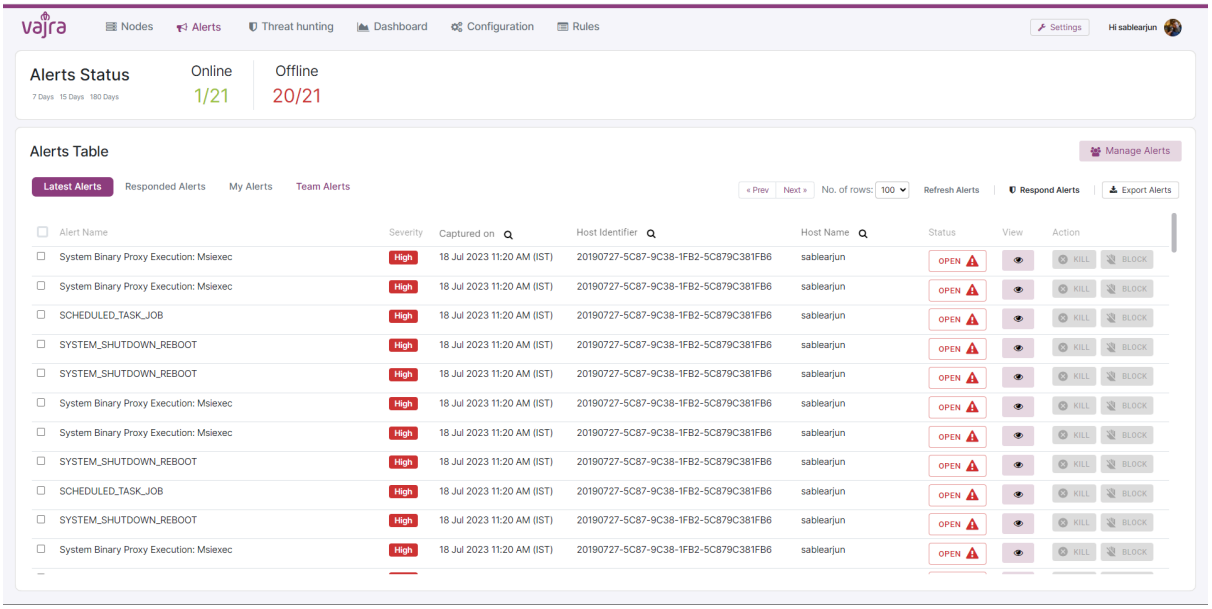

Fig. Threat detection on the alerts page

2. Threat Hunting capabilities:

By allowing analysts to send targeted queries directly to endpoints, they gain real-time access to relevant data from individual devices within the network. This data can include system logs, network traffic information, file activity, and other critical endpoint details.

With this comprehensive visibility into the endpoints, analysts can conduct in-depth investigations to identify suspicious activities or indicators of compromise that might otherwise go unnoticed. They can search for patterns, anomalies, or specific indicators associated with known threats or emerging attack techniques.

The ability to perform such investigations in real-time is crucial as it allows security teams to respond rapidly to potential cyber threats. By identifying and containing security incidents early, organizations can prevent them from escalating into major breaches that could lead to data loss, financial damage, and reputational harm.

Moreover, the proactive nature of threat hunting means that analysts can actively seek out threats rather than relying solely on automated alerts. This helps to overcome the limitations of reactive security measures and significantly reduces the dwell time of attackers in the network.

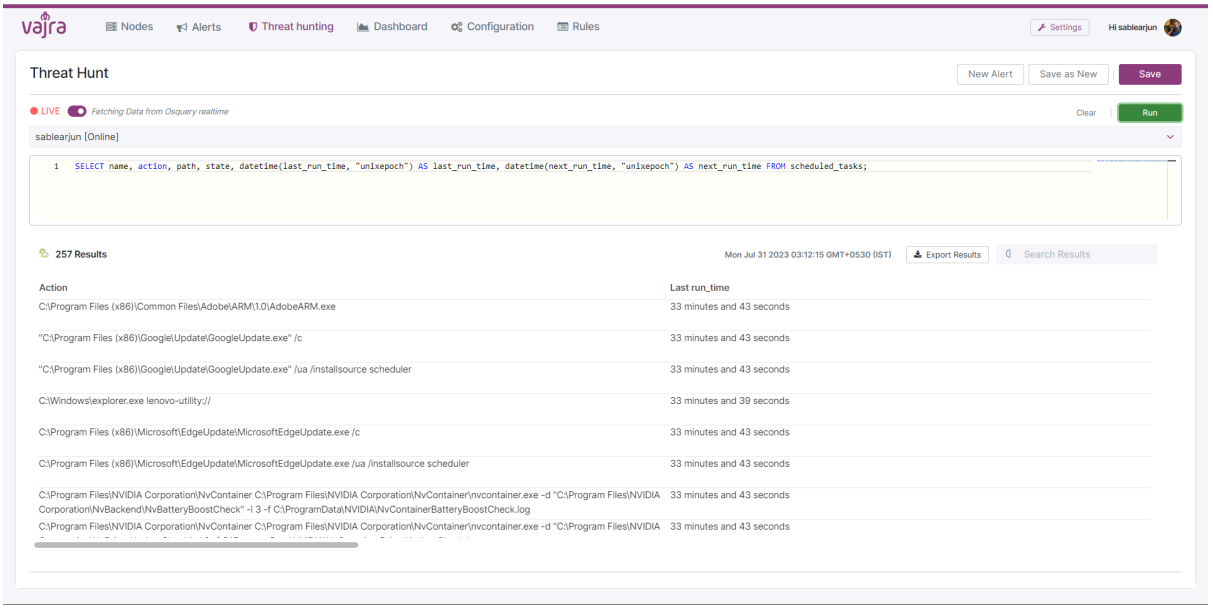

Fig. Threat hunting

3. Threat Alert Management:

Once the raw data is collected, the Vajra system performs correlation analysis, which involves analyzing this data Correlation allows the Vajra system to identify patterns and trends in endpoint behavior over time, helping to establish normal baselines for different activities.

By understanding what is typical behavior for various endpoints, the Vajra system can detect deviations from these established norms. These deviations may indicate abnormal or suspicious activities, potentially suggesting the presence of malicious actors or malware on the network.

To achieve this, the Vajra system allows security analysts to create specific rules or policies that define what constitutes abnormal behavior on an endpoint. These rules can be tailored to match the organization's specific security needs and risk tolerance. For example, rules may flag activities such as unusual file access patterns or network connections to known malicious domains.

When the Vajra system detects endpoint behavior that matches the predefined rules for abnormal activities, it generates an alert. This alert is then sent to the web Front-end, notifying it of the potential security threat or suspicious behavior. The prompt alerts allow security analysts to investigate further, quickly respond to incidents, and take appropriate actions to mitigate potential risks.

When the Vajra system triggers an alert due to the detection of suspicious or abnormal endpoint behavior, it is essential to ensure that the right team or individual with the relevant expertise handles the incident. Different teams within the organization might

have specialized skills and knowledge for specific types of threats or areas of the network. For instance:

- **Security Operations Center (SOC) Team:** The SOC team typically handles the initial triage and investigation of security alerts. They are responsible for analyzing the alerts, determining the severity of the incident, and taking immediate actions to contain the threat or escalate it to the appropriate team for further investigation.

- **Incident Response Team:** This team focuses on in-depth investigation and response to confirmed security incidents. They work to understand the nature and scope of the threat, gather additional evidence, and develop a comprehensive remediation plan.

- **Threat Hunting Team:** This team is proactively engaged in identifying and investigating potential threats before they trigger alerts. They can use this system capabilities to conduct targeted searches for indicators of compromise or unusual activities.

- **IT Operations Team:** The IT Operations team might be involved in addressing non-malicious but abnormal activities related to system health or performance issues. While not security-related, these events could be relevant to ensure the overall health of the network.

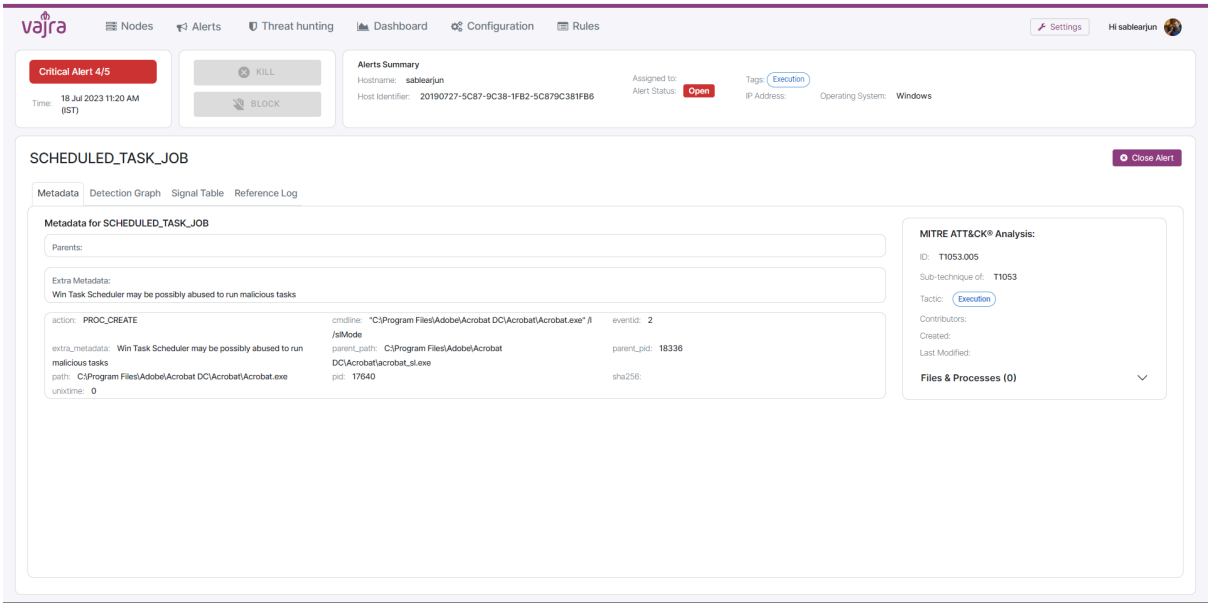

Fig. Threat alert management

4. Node/Inventory management:

Node or inventory management is an essential component of the Vajra tool. It involves keeping track of all the endpoints (nodes) within an organization's network and managing information related to these endpoints. Here's an overview of what node or inventory management entails in the Vajra tool:

- **Endpoint enrollment:** The Vajra tool registers the endpoint in its node inventory. Registration involves recording essential information about the endpoint, such as hostname, IP address, operating system, hardware specifications, and other identifying details.

- **Endpoint discovery:** The Vajra tool identifies all the endpoints connected to the server i.e Windows and Linux systems. To get the list of all the endpoints connected to the server Node management page in the Admin console is useful.

- **Node Configuration:** In Vajra, node configuration refers to the process of setting up and configuring the agent on individual endpoints (nodes) within a network. Each endpoint runs osquery as an agent to collect data, execute queries, and communicate with a central osquery server or management system. The Vajra agent allows for a distributed configuration setup, where the user can have a global configuration (shared across all nodes) and a per-node configuration. The per-node configuration can be modified from the central server.

- **Tracking Endpoint Changes:** The Vajra tool continuously monitors and tracks changes to the endpoints, such as hardware or software modifications, installed applications, configuration changes, and network connections.

- **Software Inventory**: The Vajra tool maintains a software inventory for each endpoint, keeping track of installed applications, their versions, and any changes in software configurations. Also the application list checks for any outdated application.or vulnerable version of software installed.

- **Hardware Inventory:** Similarly, the Vajra tool maintains a hardware inventory, which includes details like CPU, RAM, disk space, and other hardware-related information for each endpoint.

- **Endpoint Status and Health Monitoring:** The EDR tool constantly monitors the status and health of endpoints to ensure they are online, operational, and complying with security policies. The information about the health status of node can be seen on the Node’s health status section.

- **Search and Reporting:** Inventory management features often include search capabilities to quickly find specific endpoints based on criteria like IP address, hostname, or operating system.

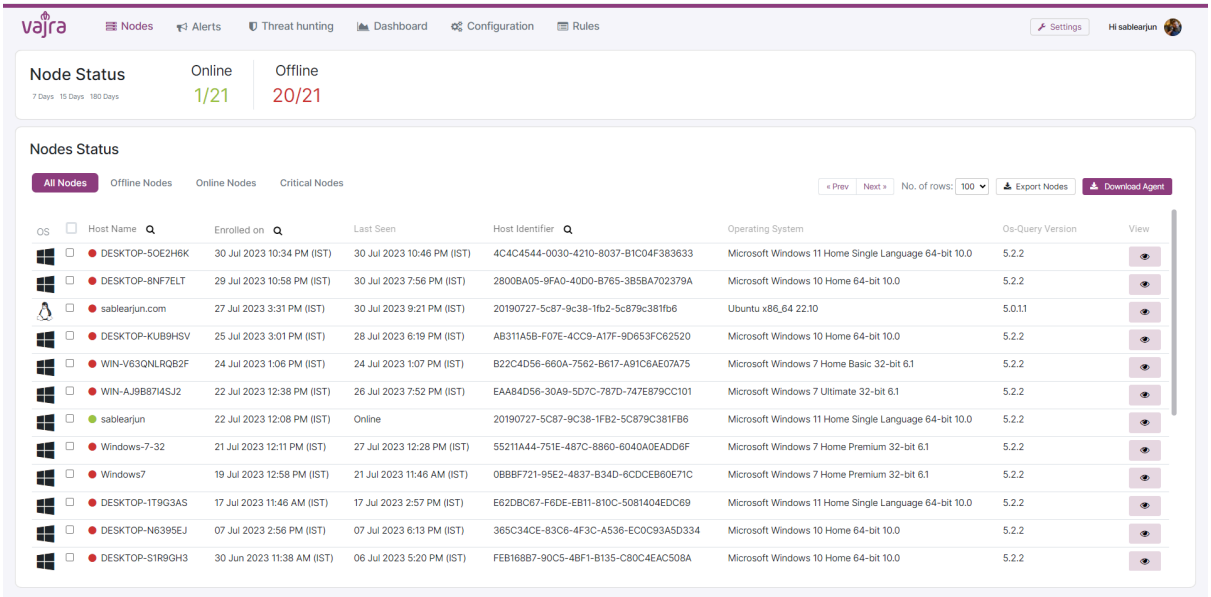

Fig. Node/Inventory management

5. Isolation of malicious nodes:

Vajra EDR's node isolation feature is a powerful capability that provides an additional layer of security to protect endpoints or nodes in an organization's network. When a node is isolated, it effectively cuts off almost all communication with other IP addresses and ports, except for port 1234, which the agent uses to send logs to the central server. This isolation ensures that the node's network activity is limited, reducing the risk of lateral movement of threats and containing potential breaches.

Here's a detailed explanation of how the node isolation feature works in Vajra EDR:

- **Node Isolation Trigger**: The node isolation feature can be triggered manually by an authorized administrator or automatically based on certain conditions, such as detecting suspicious or malicious behavior on the endpoint. For example, if Vajra EDR detects a potential security threat on the endpoint, it may automatically isolate the node to prevent the threat from spreading further.

- **Isolation Mechanism:** When the isolation is triggered, Vajra implements the isolation mechanism on the endpoint. This mechanism can work at different layers of the network stack, depending on the level of isolation required. It can be implemented using firewall rules, network segmentation, or other networking technologies.

- **Limited Communication:** Once isolated, the endpoint's network communication is significantly restricted. All outgoing and incoming

connections to IP addresses and ports other than port 1234 are blocked. This isolation ensures that the node cannot communicate with potentially malicious or unauthorized external systems, minimizing the attack surface.

- **Central Log Collection:** The node continues to function and generate logs, and it can send these logs to the central Vajra server through port 1234. This enables security teams to continue monitoring the isolated node's activity and analyze the logs to identify potential threats or security issues.

- **Recovery and Remediation:** The node remains isolated until an authorized administrator manually restores it or an automated remediation process is triggered. Before lifting the isolation, the cause of the isolation should be investigated and resolved to ensure that the endpoint is secure and ready to be reintegrated into the network.

By isolating a compromised or suspicious node, Vajra can effectively contain security incidents and prevent lateral movement of threats within the network. This containment measure helps to limit the potential damage and gives security teams time to investigate and remediate the issue before restoring normal network communication.

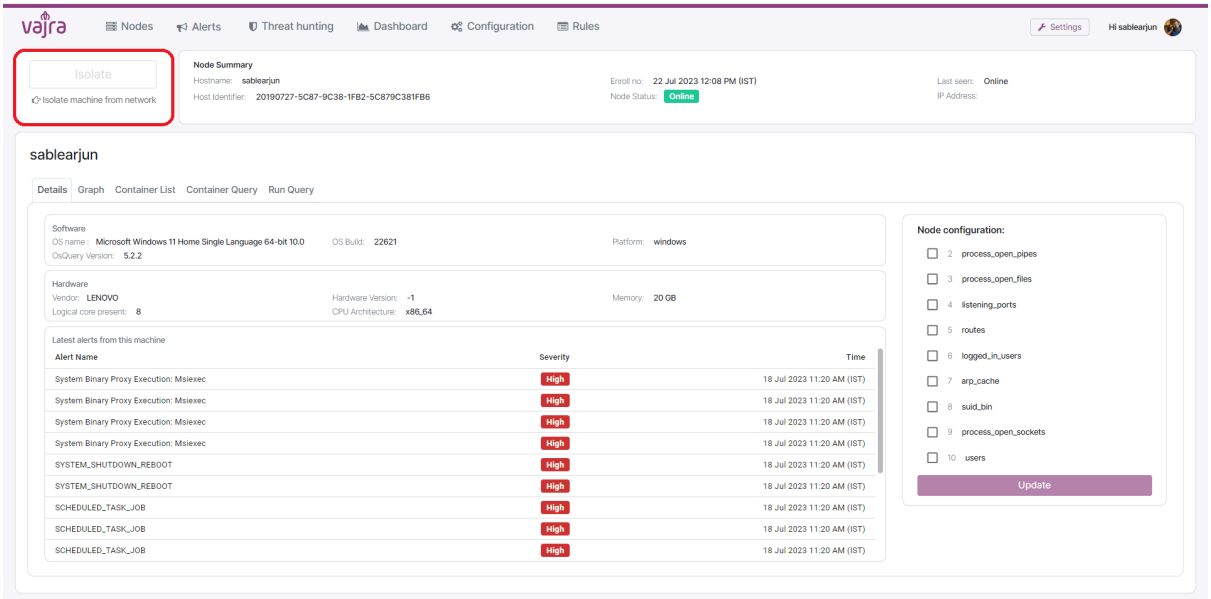

Fig. Isolation of malicious node

6. **Blocking and Terminating malicious processes:**

We offer robust Kill and Block process functionalities. We aim to execute these actions seamlessly and efficiently without encountering any access control issues as well as we want our solution to make a system-wide Impact. One of our key priorities is maintaining optimal performance while maintaining low CPU consumption. So we

have developed Kernel Driver to support our Vajra which will add a low-level component to it. In a production environment, trusting a user-space program for critical tasks like killing or suspending processes can be risky due to its lower privilege level and potential vulnerabilities. To address this, a kernel mode driver offers a superior solution. Operating at the highest privilege level (Ring 0), the kernel driver is inherently trusted by the operating system. Operating at a lower level allows for optimized monitoring and response mechanisms, ensuring that we can protect the system without causing significant performance degradation or introducing unnecessary overhead. By operating at the kernel level, we can monitor and intercept system calls, providing deep insights into the execution of processes, memory access, and other critical system events and act accordingly. Kernel drivers can impact system-wide, applying changes and enforcing policies globally across all processes and users.

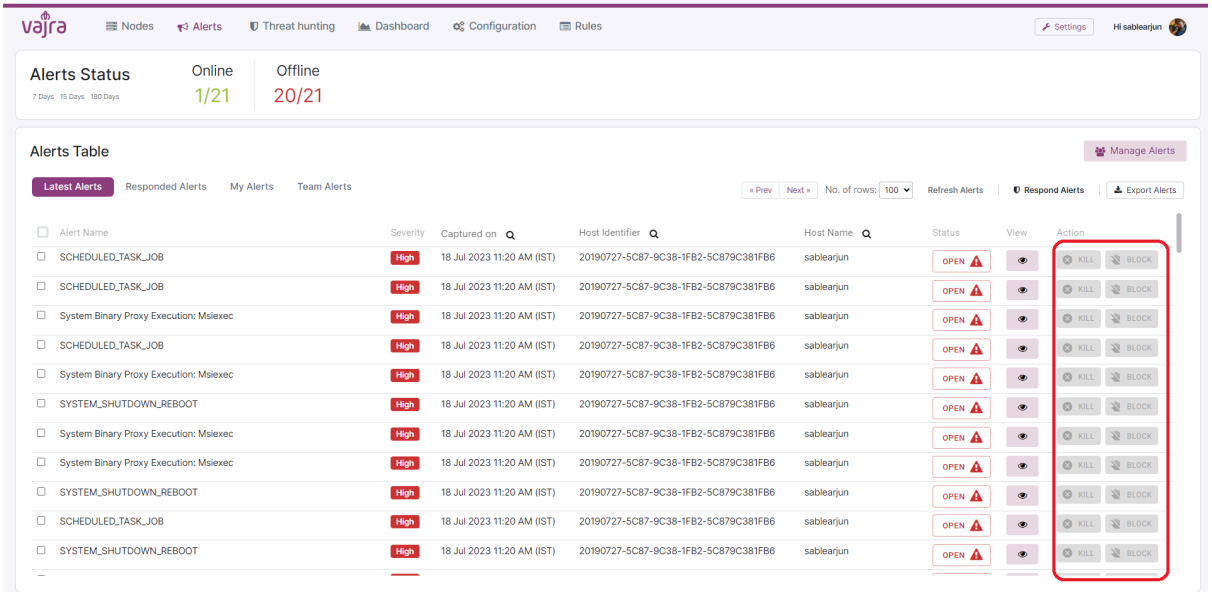

Fig. Blocking and termination of malicious process

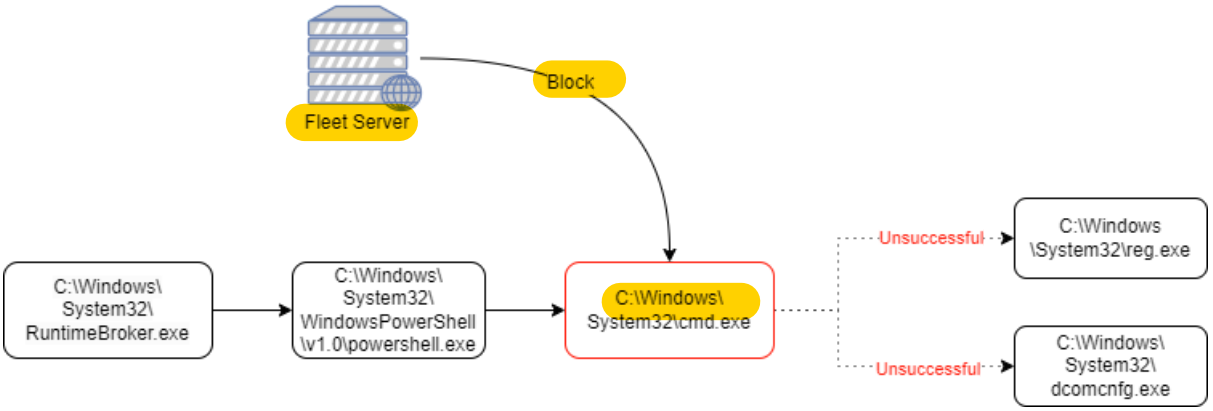

Fig. Active Blocking

7. Vajra Container Visibility and Analysis:

The software development moves away from the monolithic structure and uses modular micro-services architecture. Most of the services are now containerized and share common resources. Typical micro-service architecture with microservices running on containers is shown in the figure below.

Vajra tool offers comprehensive container support, providing organizations with a powerful solution to monitor and protect containerized environments. By harnessing kernel event logs and utilizing an Osquery extension created in-house, Vajra ensures a seamless experience without the need for intrusive probing within the containers themselves. The feature of Vajra for container visibility and analysis are listed below:

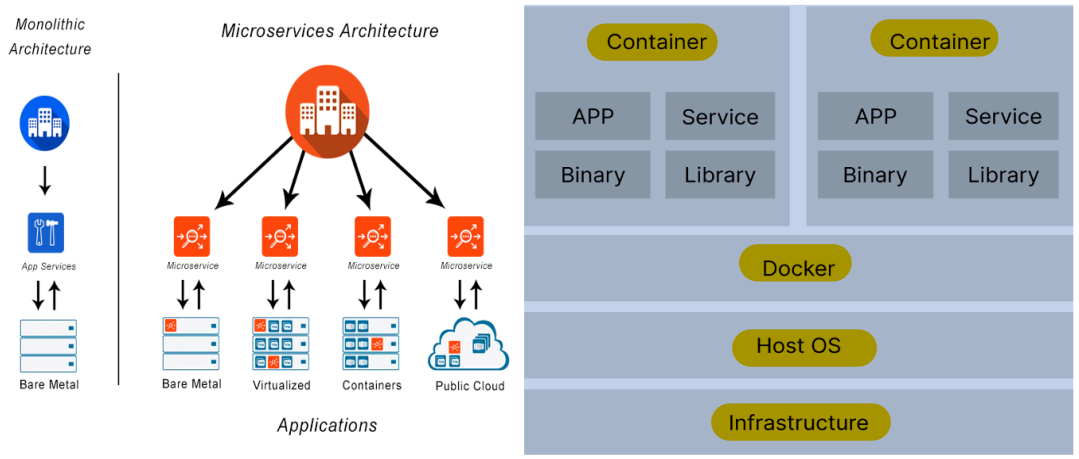

Figure: Microservice architecture. Containerization of services. Microservices run on containers, and they are connected through API gateways.

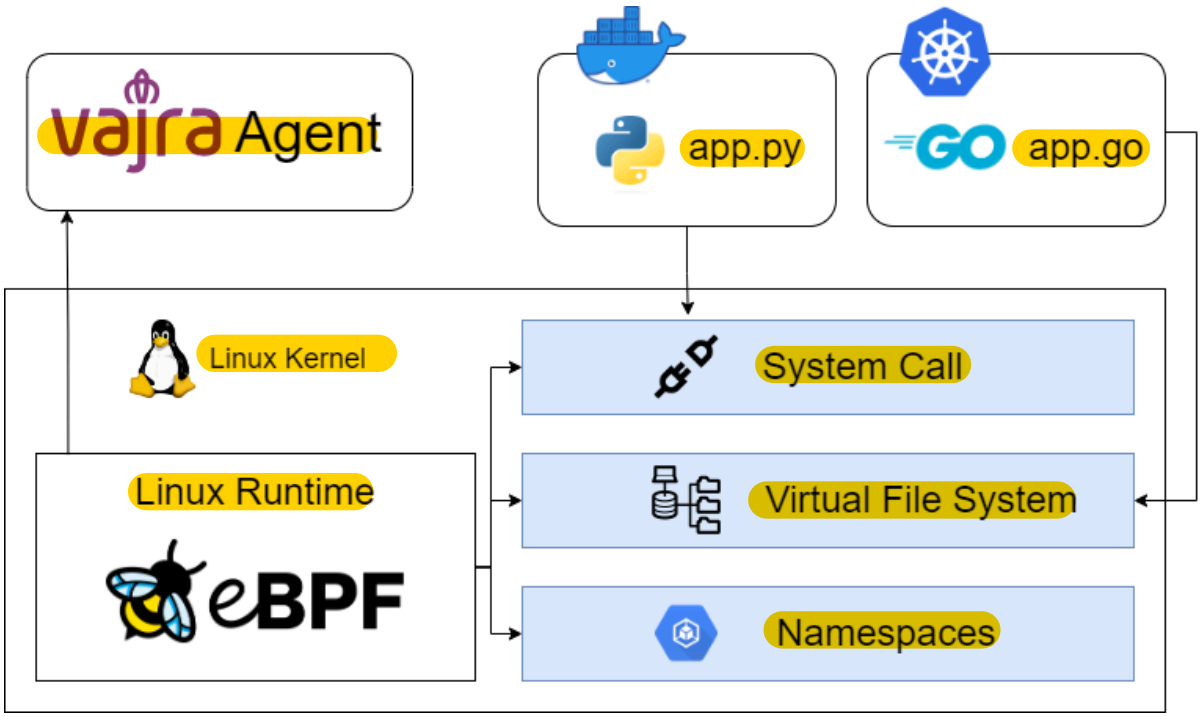

Fig. Container Visibility at Kernel Level

- *Container Visibility at Kernel Level:* Vajra captures vital container process information directly from kernel events. This approach allows us to gain real-time visibility into container activities without requiring any container modifications. By gathering data at the kernel level, Vajra ensures that container monitoring remains efficient and non-invasive, preserving the integrity of the containers' isolation.
- *Agent-Based Deployment for Ease of Implementation:* To get started with Vajra's container support, all that is needed is to install the lightweight agent on the host system. This agent seamlessly integrates with the container runtime, eliminating the need for complex setups or modifications to existing container configurations. The agent-based approach ensures quick and hassle-free deployment, allowing organizations to secure their containerized environments rapidly. This is a one-click installation and plug-and-play type system.
- *Comprehensive Container Process, File, and Socket Event Tracking:* Vajra provides extensive coverage by tracking not only container process events but also container file events and socket events. This holistic monitoring approach allows security teams to gain insights into file access patterns, network connections, and process interactions within the container. The comprehensive event tracking ensures that no suspicious activity goes unnoticed.
- *Container Information with Osquery Extension:* Vajra has gone a step further in supporting docker monitoring by creating an Osquery extension. This extension empowers organizations to gather all relevant information about the docker containers, including their configuration details, container state, and network configurations, etc. The collected information is invaluable in understanding the

container's normal behavior, making it easier to identify anomalies and potential security threats.

- *Exclusive container process events extension:* This innovative Osquery extension empowers users to gain comprehensive visibility inside containers without the need for heavy backend correlation. This extension simplifies container monitoring, enhances security, and allows organizations to efficiently do real-time process monitoring within containers and address potential threats within containerized environments.

D. Features for Malware analysis on 5G ecosystem in regular shorter intervals using minimal cost

Vajra offers various analytics features to analyze malicious activity as follows:

1. **Finding the Root Cause of The Alert. (Back Tracking):**

   Every process that runs on an operating system has a unique process identifier (PID) assigned to it. The PID is a numeric value that the operating system uses to track the process and allocate system resources to it.

   In addition to the PID, every process, except the initial process, also has a parent process identifier (PPID). The PPID is the PID of the process that created the current process. When a new process is created, the operating system sets the PPID of the new process to the PID of the process that created it. This establishes a parent-child relationship between the two processes.

   The parent-child relationship between processes is an important concept in understanding how processes behave on an operating system. The parent process creates child processes and can also terminate them. The child process inherits some of the properties of the parent process, such as its security context and environment variables.

   Task managers, system administrators, and developers use PIDs and PPIDs to identify process relationships and understand how different processes interact with each other. For example, if a particular process is causing problems, identifying its PPID can help identify the parent process that spawned it. This information can be used to troubleshoot issues and understand the root cause of problems.

   In summary, the PID and PPID are crucial identifiers that help track and manage processes on an operating system. The parent-child relationship between processes established by the PPID is an important concept that helps understand process behavior and troubleshoot issues in the system.

2. **Analysing Attacker Behaviour with the MITRE ATT&CK Framework:**

The MITRE ATT&CK framework is a powerful tool for analyzing attacker behavior and understanding the different tactics and techniques they use during an attack. The framework provides a standardized way to categorize and describe these tactics and techniques, making it easier for organizations to identify and respond to attacks. One of the key benefits of using the MITRE ATT&CK framework is that it provides a comprehensive view of the different stages of an attack.

This allows organizations to better understand the attack lifecycle and the different tactics and techniques attackers use at each stage. By understanding these tactics and techniques, organizations can develop more effective security controls and mitigation strategies to better defend against attacks.

Another important benefit of the MITRE ATT&CK framework is that it helps organizations develop better threat intelligence. By tracking the use of specific tactics and techniques across different attacks and campaigns, organizations can gain insights into the tactics and techniques favored by different threat actors. This can help organizations anticipate and respond to future attacks more effectively.

The following figure shows the Tactics, Techniques, and Procedures (TTPs) of the framework

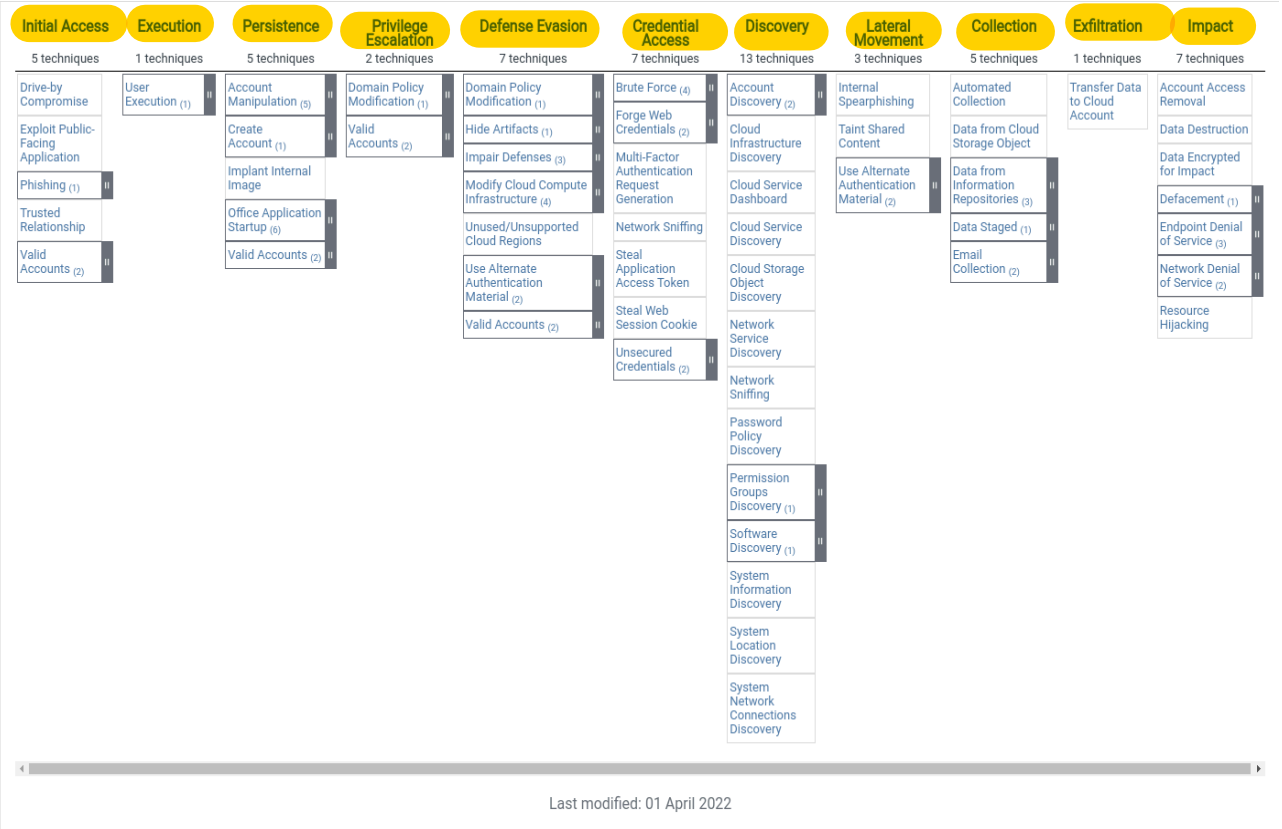

Fig. MITRE ATT&CK framework matrix

3. Tracking Children of Malicious Event (Forward Tracking):

Once a suspicious activity alert is generated, it is important to track the child processes of the suspicious process to understand the full scope of the attack. This can be done by checking all the child processes recursively of the suspicious process alert.

To do this, we can use Osquery to check the suspicious alert event PID in different event tables such as Socket events table and File events table. If the PID of the alert event matches the PPID of any events in these tables, a parent-child relationship is formed, which we can call forward tracking.

Forward tracking helps us to understand how the malicious process spawned child processes and what those child processes were doing. This information is crucial for identifying the extent of the attack and determining the appropriate remediation actions.

The concept of parent-child relationships and tracking child processes can be applied to both Linux and Windows OS. In Windows, instead of using PIDs and PPIDs, we use Process GUIDs (PGUIDs) and Parent Process GUIDs (PPGUIDs) to track processes and their relationships.

In the Figure below, we can see an alert is generated for the process D and we have used backtracking to find the root cause of the alert. To track the children of the process D, we'll do forward tracking where D's PID will be checked as PPID in all evented tables of Osquery, and if it matches, we'll generate the forward process tree of it.

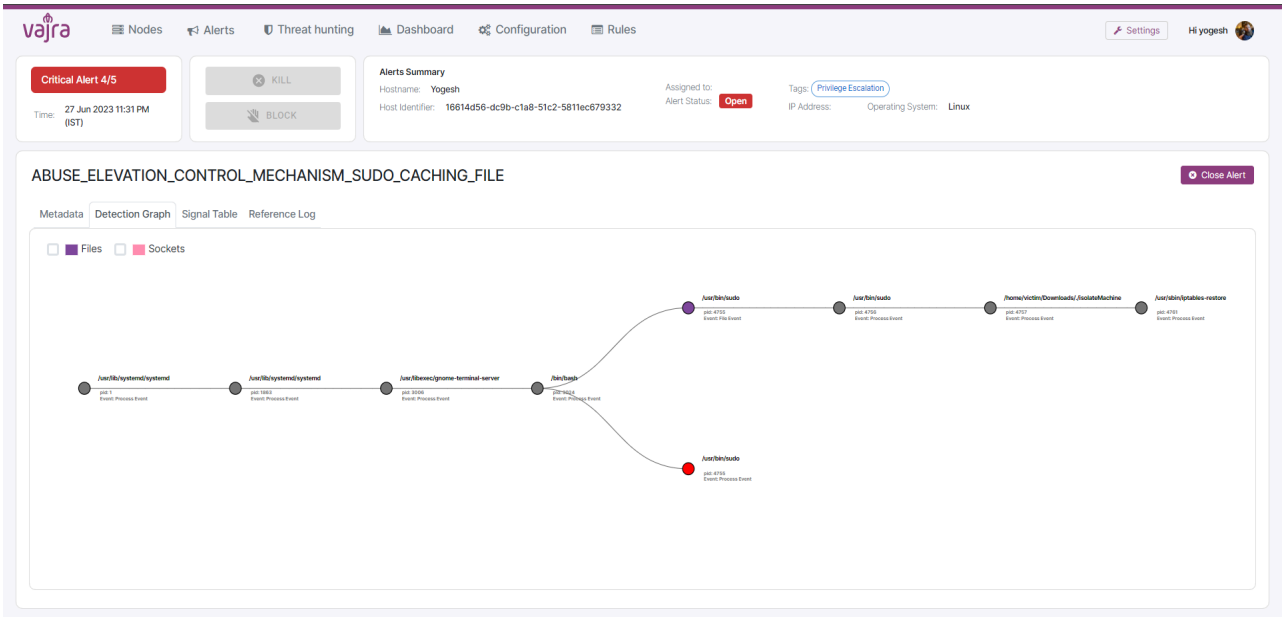

Fig. Forward and backward tracking of the potentially malicious event

Overall, tracking child processes and understanding the parent-child relationships between processes is a crucial step in investigating and mitigating suspicious activity on a system. By using Osquery and examining different event tables, we can gain a comprehensive understanding of the scope of an attack and take appropriate actions to remediate the issue.

E. Other features of Vajra:

Vajra has several other features. Some of them are listed below.

- Full support for Bharat Operating System Solutions (BOSS) Linux operating system
- Can be deployed on-prem or cloud
- No internet connection is required for installation or update.
- Provides configurable data retention policy