International Research Journal of Education and Technology
Peer Reviewed Journal
ISSN 2581-7795

Endpoint Security and Threat Intelligence Project (EDR)

ABISHEK P[1]

[1] *Student, Dept. of Electronic and instrumentation Engineering, Bannari amman institute of technology, INDIA*

Abstract - The Endpoint Detection and Response (EDR) system is an advanced cybersecurity solution designed to provide continuous monitoring, detection, and automated response to potential threats targeting endpoint devices. It operates in real time, using a combination of signature-based detection, heuristic analysis, and behavioral monitoring to identify both known and unknown threats, such as malware, ransomware, and fileless attacks. The EDR system employs powerful techniques like API hooking to monitor system-level activities, including file access, process creation, memory usage, and network communications. By analyzing system behaviors and correlating them with threat intelligence, the EDR system can quickly detect anomalies, suspicious patterns, and unauthorized actions that might indicate a security breach. It provides comprehensive visibility into endpoint activities, helping security teams identify threats early and mitigate damage effectively. The system also integrates with external threat intelligence sources to stay updated on the latest vulnerabilities and exploits, ensuring proactive defense against emerging threats. Equipped with a user-friendly interface, the EDR allows administrators to configure scan settings, view real-time logs, and manage security responses. Whether performing routine scans or responding to active incidents, the system empowers security teams with the tools needed to maintain endpoint integrity and minimize the impact of cyberattacks.

Key Words: Endpoint Detection and Response (EDR), Cybersecurity Monitoring, Threat Intelligence, Behavioral Analysis, API Hooking, Heuristic Analysis

1. INTRODUCTION

In today's rapidly evolving cybersecurity landscape, safeguarding endpoints has become a critical priority for organizations. Endpoints, such as desktops, laptops, and servers, often serve as entry points for cyberattacks, making them vulnerable to threats like malware, ransomware, and advanced persistent threats (APTs). To address these challenges, the Endpoint Detection and Response (EDR) system has emerged as an essential cybersecurity solution.The EDR system continuously monitors and protects endpoints against a wide spectrum of security threats. By combining signature-based detection, heuristic analysis, and behavioral monitoring, the system can effectively identify and mitigate both known and unknown threats. It operates in real time, focusing on system activities such as file access, process execution, memory utilization, and network traffic to detect anomalous or malicious behaviors.Through advanced techniques like **API hooking**, the EDR system delves into low-level system operations to identify abnormal activities and intervene before attacks escalate. This capability allows the EDR solution to monitor critical events, including file modifications, unauthorized process creation, and suspicious network communications. Beyond robust detection capabilities, the EDR system empowers security teams with actionable insights and automated responses. By leveraging external threat intelligence sources and continuously analyzing endpoint data, the system ensures organizations stay ahead of emerging threats. Its proactive defense mechanisms and deep endpoint visibility form a cornerstone of any modern cybersecurity strategy, enabling rapid detection and response to potential breaches.

1.1 Client-Server Architecture

The EDR solution is built on a scalable **client-server architecture**, where the **client (PlanqX Sensor)** operates on endpoint devices, and the **server** acts as a centralized hub for data aggregation, analysis, and threat management. The client collects real-time data, monitoring activities like file access and network traffic, and forwards this information to the server. The server correlates these findings with threat intelligence, performs deeper analysis, and provides actionable commands for mitigation.

1.2 PlanqX Sensor (Client-Side Component)

The **PlanqX Sensor** is a lightweight agent installed on endpoint devices, responsible for real-time monitoring and data forwarding. It hooks into system APIs to track critical activities such as process creation, file modifications, and memory usage. Despite its advanced capabilities, the sensor operates with minimal resource consumption, ensuring uninterrupted endpoint performance. When it detects

International Research Journal of Education and Technology
Peer Reviewed Journal
ISSN 2581-7795

suspicious activities, it can trigger alerts, block threats, or initiate automated responses to neutralize potential risks.

1.3 Server Architecture

The **server component** functions as the brain of the EDR system, processing data received from client sensors. It correlates findings with threat intelligence, performs behavioral analysis, and generates actionable insights. Designed for scalability, the server can support a large number of endpoints, making it suitable for organizations of all sizes. Administrators can interact with the server through a **command-line interface (CLI)** or a graphical user interface (GUI) to view logs, configure settings, and monitor security events.

1.4 Installation and Requirements

Deploying the EDR system requires the installation of both client and server components.

Client Installation: The PlanqX Sensor is deployed on endpoint devices using automated scripts, ensuring a streamlined setup process.
Server Installation: The server can be hosted on a dedicated machine or a cloud-based infrastructure. It requires a modern operating system, sufficient processing power, and adequate storage to manage data from connected endpoints. Internet connectivity is essential for integrating external threat intelligence feeds, and the server-client communication is secured over encrypted channels.

1.5 Add-ons and Configurations

The EDR system supports flexible configurations and add-ons to cater to diverse security requirements.

Accounts Management: Role-based access control ensures that sensitive data and system functions are accessible only to authorized personnel, such as administrators and security analysts.

Logging and Reporting: Comprehensive logs capture every detected threat, system activity, and response action. These logs are stored locally and centrally, with options to forward them to external systems for long-term analysis or compliance reporting.

2. CLIENT ARCHITECTURE (PLANQX SENSOR)

The PlanqX Sensor is the client-side agent that resides on each endpoint device, performing real-time monitoring and data collection. Its primary purpose is to continuously observe system activities, detect potential threats, and forward critical data to the server for further analysis and response. The sensor is lightweight and designed to operate with minimal impact on system performance while maintaining robust protection against cyber threats.

Table -1: Feature Comparison Table

Feature	Your EDR System	Traditional Antivirus	Basic IDS/IPS
Real-Time Monitoring	✔	✔	Limited
Behavioral Analysis	✔	✘	✔
API Hooking	✔	✘	✘
Network Monitoring	✔	✘	✔
Phishing Detection	✔	✘	✘
Exploit Detection	✔	Limited	✔
Advanced Logging	✔	Limited	✔
Customizable Responses	✔	✘	✔

2.2 Role of Sensor

The sensor's primary role is to monitor system activities at the endpoint level, including file access, process creation, network traffic, and registry changes. It collects this data through a combination of signature-based and behavioral monitoring techniques. The sensor is responsible for sending relevant information back to the server for deeper analysis and correlation with other endpoint data. In the event of suspicious activity, it can trigger alerts, block harmful processes, and take corrective actions based on predefined rules or commands from the server.

2.3 Design Goals

The PlanqX Sensor was designed with the following key goals in mind:

Real-Time Threat Detection: It continuously monitors system behavior to detect potential threats, such as malware or ransomware, as soon as they occur.

Minimal Resource Consumption: The sensor is designed to use as few system resources as possible to avoid impacting the user experience or performance.

Remote Management: The sensor can receive updates, configurations, and commands from the server without requiring manual intervention on each endpoint.

Stealth and Evasion: The sensor operates discreetly without alerting potential attackers, ensuring its presence is undetected by malicious software.

2.6 Performance Optimization

To minimize system resource consumption, the PlanqX Sensor is optimized for performance:

.The sensor's operations are highly optimized to minimize CPU and memory usage, ensuring that the endpoint's primary functions remain unaffected by the monitoring process.

Not all events are equally critical. The sensor filters out irrelevant data, ensuring that only high-priority or potentially malicious activities are transmitted to the server.

The sensor uses a compressed and encrypted format for transmitting data to the server, ensuring efficient bandwidth usage while maintaining the security and integrity of the data.

2.7 Logging and Alerting

The PlanqX Sensor is equipped with robust logging and alerting capabilities:

Local Logging: The sensor logs all events and detected activities on the endpoint, which can be stored locally for future reference or forensic analysis. These logs are encrypted to ensure confidentiality.

Centralized Logging: The data is forwarded to the server for centralized logging and analysis. This provides security teams with a unified view of all endpoint activities across the network.

Real-Time Alerts: The sensor triggers real-time alerts based on predefined thresholds, such as unusual file modifications or unauthorized process executions. These alerts can be sent to the server, which then notifies the administrators.

3. SERVER SIDE ARCHITECTURE

3.1 Core Components

The Server-Side Architecture serves as the heart of the EDR system, processing data received from the PlanqX Sensors on the endpoints. The core components of the server architecture include:

Data Ingestion and Storage: The server collects data from the client-side sensors, including event logs, alerts, and system behavior reports. This data is securely stored in a centralized database that allows for quick retrieval and correlation.

Threat Intelligence Integration: The server integrates with external threat intelligence feeds to keep the system updated on emerging threats. These feeds provide additional context for the alerts and data collected from the endpoints.

Analysis Engine: The analysis engine processes incoming data, analyzing it against known threat patterns, signatures, and behaviors. It uses advanced algorithms to correlate endpoint data, looking for potential threats and patterns of suspicious behavior.

Alert Management System: This component handles the classification, prioritization, and notification of detected threats. It ensures that administrators are promptly alerted about critical security incidents.

Communication Layer: The communication layer manages the secure exchange of data between the server and the endpoint sensors, ensuring that information is transmitted efficiently and safely.

3.2 Administrator Dashboard – CLI

The Administrator Dashboard provides a user interface for security professionals to manage, monitor, and respond to threats. For advanced users and automated operations, a Command-Line Interface (CLI) is available, allowing administrators to interact with the server through scripted commands.

Real-Time Alert Monitoring: Administrators can view real-time alerts and security incidents as they occur on the network.

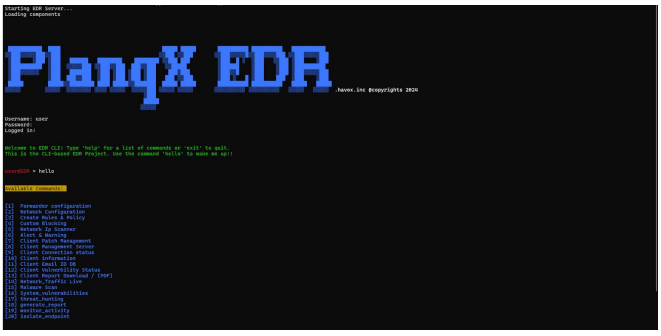
Endpoint Status Overview: The dashboard provides an overview of all endpoint devices, their current status, and their communication with the server.

Incident Response Management: Administrators can take immediate action based on alerts, such as isolating endpoints, stopping malicious processes, or running diagnostic scans.

Threat Analysis: The CLI can be used to perform detailed threat analysis, allowing security teams to dig deeper into alert details and endpoint behaviors

International Research Journal of Education and Technology
Peer Reviewed Journal
ISSN 2581-7795

.**Fig -1**:EDR CLI interface



3.3 Telemetry Collection

The server plays a crucial role in Telemetry Collection, gathering and analyzing system behavior data from the endpoints. Telemetry data includes logs, system events, user activities, and metadata, which provide insights into the overall health and security posture of the network.

Continuous Monitoring: The server continuously collects telemetry data from the sensors, ensuring real-time visibility into endpoint activities.

Granular Data Collection: Telemetry is collected at a granular level, including process executions, file system changes, network traffic, and system resource usage.

This telemetry helps provide a comprehensive view of the security environment, enabling proactive threat hunting and incident response.

3.5 Threat Detection and Correlation

The Threat Detection and Correlation engine is one of the most critical components of the server architecture. It uses advanced techniques to detect, prioritize, and correlate potential threats across endpoints in real-time.

Signature-Based Detection: The server compares data collected from sensors against known malware signatures and behaviors to identify threats based on predefined patterns.

Heuristic and Behavioral Analysis: The engine identifies suspicious activities based on their behavior, even if they don't match known signatures. For example, it may detect ransomware-like behavior, such as mass file encryption, or unusual system activities that deviate from normal patterns.

Event Correlation: The server correlates data from multiple endpoints to identify coordinated attacks, such as lateral movement or malware spreading across the network. By analyzing related events from different systems, the server can piece together a broader view of the attack.

4. NETWORKING AND MONITORING ON SERVER

Networking monitoring on the server side focuses on overseeing the traffic flow, detecting suspicious activities, and ensuring that communications between endpoints and external systems are secure. This is achieved using Windows Networking Libraries such as WinPCap, WFP (Windows Filtering Platform), and NIDPS. These libraries allow for comprehensive traffic capture and analysis at both the network and transport layers.

By integrating encrypted proxy servers, the system can route all traffic through a secure channel, ensuring no data is leaked or tampered with. It provides insight into real-time traffic from endpoints, including the communication with external servers, allowing for the detection of malicious activities such as phishing attacks, C2C (Command and Control) communications, and data exfiltration.

4.2 Traffic Analysis

Traffic analysis on the server involves the constant observation of all inbound and outbound network traffic from the endpoints. By leveraging Windows Network Libraries, the server captures and analyzes packet data, looking for abnormal patterns or known attack vectors. The implementation uses custom proxy servers to capture traffic, encrypt it, and analyze it using pattern matching, frequency analysis, and anomaly detection.

Real-time Traffic Inspection: Continuous monitoring of the flow of data allows the detection of suspicious spikes or drops in traffic. For example, when an endpoint suddenly begins sending large volumes of data, it could signal a data breach or exfiltration attempt.

Phishing Detection: Traffic analysis is enhanced with IOC to detect suspicious URLs, IP addresses, or domains, providing an extra layer of protection against phishing attempts.

4.3 Packet Inspection

Packet inspection involves analyzing the individual network packets that flow through the system. Using the Windows Packet Capture (WinPCap) library, the EDR server inspects packets for patterns such as malicious payloads, port scanning, or unexpected data. The proxy server ensures that all traffic is inspected, even if encrypted, by intercepting and decrypting traffic before passing it to the endpoint.

Layered Inspection: All data from endpoints is captured and analyzed at the application, transport, and network layers.

Encrypted Traffic Handling: The proxy server decrypts encrypted traffic (e.g., HTTPS) between endpoints and external servers to inspect for malicious content, ensuring that malicious communications aren't missed.

4.4 Anomaly Detection

Anomaly detection involves identifying irregular behaviors in network traffic patterns. By using statistical methods, the server can automatically flag anomalies based on typical network activity and historical data.

Behavioral Analysis: Identifies any deviation from baseline traffic behaviors, such as an endpoint connecting to unusual or suspicious IP addresses or domains. For example, if a device typically only communicates with internal systems and suddenly contacts multiple external domains, it could be a sign of a compromise or botnet activity.

Threshold-Based Alerts: The system automatically triggers alerts when certain thresholds for traffic volume or frequency are surpassed. This enables prompt response to potential threats like DDoS attacks or data exfiltration.

4.5 Protocol-Specific Monitoring

The system is capable of monitoring traffic across various network protocols such as HTTP, HTTPS, DNS, FTP, and SMB. The proxy server inspects these protocols in real-time, checking for protocol-specific anomalies, such as DNS tunneling, command-and-control messages in HTTP requests, or suspicious SMB communications.

DNS Monitoring: The server monitors DNS queries, checking for domain fluxing (a common tactic used by malware), detecting unauthorized DNS requests and filtering malicious domains.

HTTP/HTTPS Requests: Inspects HTTP/HTTPS traffic to identify common attack patterns, such as SQL injection, cross-site scripting (XSS), or attempts to exploit vulnerabilities in web applications.

4.6 Tampered Domain, IP, Subdomain Monitoring

This component tracks and monitors domains, IP addresses, and subdomains that may be associated with malicious or unauthorized activities. The server cross- references endpoints' traffic with threat intelligence feeds, blocking known malicious IPs, domains, and subdomains.

Real-Time Blocking: When traffic attempts to reach a known compromised domain or IP address, it is immediately flagged, logged, and blocked.

IOC (Indicators of Compromise) Database: The server can hold up to 10,000+ IOCs (e.g., suspicious IPs, domains, or URLs), continuously comparing and updating known threat indicators in real-time.

5. USER & KERNEL MODE DETECTION

5.1 User Mode Overview

User-mode detection is essential in identifying malicious activities originating from applications or user-space programs. The EDR system uses Windows internal APIs to intercept system calls, monitor processes, and flag abnormal behaviors indicative of malicious activity.

5.1.1 User Mode Detection

At the core of user-mode detection, the EDR leverages internal Windows APIs like **NtQuerySystemInformation, CreateProcess**, and **NtReadFile**. These APIs allow the EDR to monitor the creation of processes, track system calls, and inspect file system accesses in real-time. For example, when a suspicious process tries to open a file, **NtCreateFile** API is monitored for unusual access patterns.

5.1.2 Techniques and Methods

The EDR uses several key techniques:

API Hooking: Through SetWindowsHookEx and Detours API, the EDR system can intercept API calls and inspect parameters or return values

File and Process Monitoring: The system hooks the NtOpenProcess and NtQueryDirectoryFile to track processes and file operations, enabling detection of malicious file system modifications.

5.1.3 API Hooking

API hooking in user-mode is implemented using the Detours library or Inline Hooking techniques to monitor critical APIs like **NtWriteFile** and **CreateFile**. When an API call such as NtCreateFile is made, the EDR inspects the parameters (like file paths) for abnormal behavior, such as accessing system-sensitive files or files outside standard directories. If malicious activity is detected, the EDR can block the process or alert the administrator.

5.1.4 Dynamic-Link Library (DLL) Injection Monitoring

Monitoring DLL injections is done through intercepting calls to LoadLibrary and **GetProcAddress**, which are frequently abused by malware to inject code into l**egitimate processes.** The EDR tracks when a process loads an untrusted DLL and blocks it if it's not signed or part of the trusted list. It also looks for **RtlCreateUserThread** to detect thread injection attempts that malware might use for persistence.

5.1.5 Process Monitoring

Process monitoring is achieved through hooks placed on APIs like **NtQuerySystemInformation** and ZwQueryInformationProcess, which help track process execution and identify any process that behaves abnormally or tries to bypass security mechanisms. Suspicious processes, such as those trying to modify system files or escalate privileges, are flagged and terminated if necessary.

5.1.6 File System Monitoring

The file system is monitored by hooking APIs such as **NtCreateFile** and **NtWriteFile**, which are responsible for file creation and modification. The EDR inspects these calls to detect malicious files, especially those attempting to alter critical system files. If a suspicious file is detected (e.g., **ransomware** trying to encrypt files), it is quarantined or blocked.

5.1.7 Event-Based Monitoring

Event-based monitoring is achieved through Windows Event Tracing for Windows (ETW) or ZwNotifyChangeDirectoryFile to capture system-level events. These events, like user login failures or unexpected file access patterns, are logged and analyzed in real-time to identify potential security threats.

5.2 Kernel Mode Overview

Kernel mode monitoring focuses on the system's most privileged level of execution. Malicious activities in kernel mode can subvert many security measures, so detecting and preventing them is critical. The EDR uses Windows internals to hook into kernel routines and track low-level system interactions.

5.2.1 Kernel Hooking

Kernel hooking is implemented by intercepting calls to kernel-level functions using techniques like Inline Hooking and Kernel Modules. Key functions such as **ZwOpenFile** and **ZwSetInformationFile** are hooked to detect abnormal behavior, like unauthorized access to kernel memory. This allows the EDR to block activities like rootkit installation before they can compromise the system.

5.2.2 Driver Monitoring

Driver monitoring is accomplished by checking loaded drivers using **PsLoadedModuleList** and hooking into **IoCreateDevice** and **IoCreateSymbolicLink**. The EDR watches for unsigned or suspicious drivers that could potentially give attackers access to privileged kernel resources. If a malicious driver is detected, it is flagged, and the system may prevent it from loading.

5.2.3 ELAM (Early Launch Anti-malware)

ELAM is integrated into the EDR to protect against early-stage malware that loads before the operating system has fully initialized. By monitoring driver and module loading during boot using **NtLoadDriver** and integrating with Secure Boot, **ELAM** ensures that malicious code is blocked before it can interact with the system.

5.2.4 Direct Kernel Object Manipulation (DKOM) Detection

DKOM detection works by monitoring kernel data structures for unusual modifications. The EDR tracks changes in structures like PsActiveProcessHead to detect hidden processes or rootkits trying to remain undetected. Any tampering with these structures triggers an alert for further investigation.

CONCLUSION:

In conclusion, the development and implementation of this Endpoint Detection and Response (EDR) system will significantly enhance the security capabilities of Windows-based environments. By leveraging advanced detection techniques, including real-time monitoring, user and kernel mode detection, network traffic analysis, and integration with threat intelligence, the system will provide a comprehensive defense against evolving cyber threats. The use of client-server architecture ensures scalability and flexibility, making it suitable for deployment across various network sizes, from small enterprises to large organizations.

ACKNOWLEDGEMENT

The authors express their sincere gratitude to the faculty and staff of Bannari Amman Institute of Technology for their unwavering support and guidance during this project. Special thanks are extended to the Smart India Hackathon committee for providing a platform to innovate in cybersecurity. The authors also acknowledge the contributions of the open-source community, whose tools and resources were instrumental in the development of this project.

REFERENCES

[1] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed., New York, NY: Wiley, 1995.

[2] G. Kim and J. Love, "Evolution of Endpoint Security: From Antivirus to EDR," IEEE Security Privacy, vol. 19, no. 4, pp. 57–65, Jul. 2021, doi:10.1109/MSEC.2021.3103301.

[3] J. Smith, "Behavioral-based detection in modern EDR systems," Cybersecurity J., vol. 8, pp. 22–30, Sep. 2023.

BIOGRAPHIES

Threat Researcher, Malware analyser, Pentesting.