**VARC PROJECT OVERVIEW**

**Methodology of Execution of the Project**

**Introduction**

The cybersecurity dashboard application is designed to collect, analyze, and present security-related data in a structured manner. The primary objective is to assess system security risks using a systematic approach that involves real-time data collection, forensic analysis, and risk scoring. The methodology is structured into four key phases: data collection, risk assessment, visualization, and reporting.

**Phase 1: Data Collection**

The system collects security-related data from multiple sources using Python libraries and system commands. The key data collection methods include:

1. **Process Scanning**: Using psutil, the application retrieves all running processes along with their names, process IDs (PIDs), CPU/memory usage, and execution paths.
2. **Network Port Analysis**: The application utilizes nmap to scan for open ports and running services.
3. **Registry & File System Monitoring (Windows)**: Using winreg (for Windows), startup entries and suspicious file modifications are analyzed.
4. **Event Log Analysis**:
   * **Windows Logs**: The win32evtlog module extracts logs from the Security, System, and Application event logs.
   * **Linux Syslogs**: System authentication logs (auth.log and syslog) are scanned for failed logins, privilege escalations, and unauthorized access attempts.
5. **Digital Signature Verification**: Binary files and executables are verified for tampering or unsigned modifications.

**Phase 2: Risk Assessment & Scoring**

Once the data is collected, the system processes the findings to assign a risk score based on the severity of detected threats. The risk assessment is performed as follows:

1. **Risk Classification**:
   * Each event is categorized into predefined threat levels (low, medium, high) based on known security best practices.
   * Failed logins, unauthorized access, and unverified processes increase the risk score.
2. **Threshold Evaluation**:
   * Suspicious processes (>5) = +5 points per process.
   * More than 10 open ports = +3 points.
   * Failed logins (Windows 4625, Linux SSH) = +2 points per attempt.
   * Audit logs cleared (Event ID 1102) = +5 points.
3. **Final Risk Calculation**:
   * The total risk score is computed and categorized into severity levels:
     + 0-10: Low Risk
     + 11-30: Medium Risk
     + 31+: High Risk

**Phase 3: Data Visualization & Dashboard**

Once the risk score is computed, results are displayed on a Flask-based web interface. The dashboard features:

* **Risk Score Overview**
* **Detailed Event Log Breakdown**
* **Historical Risk Trends**
* **Suspicious Process & Port Information**

**Phase 4: Forensic Reporting & Evidence Storage**

For forensic investigations, the application allows exporting logs, risk assessments, and suspicious activity into structured JSON or CSV formats. Additionally, evidence like memory dumps and registry modifications can be archived for future analysis.

**Step-wise Algorithm & Complexity Analysis**

**Step-wise Algorithm (Pseudocode)**

1. Initialize system scan

2. Collect security data:

a. Scan processes: O(n)

b. Scan open ports using nmap: O(m)

c. Retrieve system logs:

- Windows Event Logs: O(log n)

- Linux Syslogs: O(log n)

d. Check registry entries (Windows): O(k)

e. Verify digital signatures: O(f)

3. Assess risk score:

a. If suspicious processes > 5, add 5 pts per process (O(n))

b. If open ports > 10, add 3 pts (O(1))

c. If failed logins exist, add 2 pts per instance (O(n))

d. If audit logs cleared (Event ID 1102), add 5 pts (O(1))

4. Compute total risk score (O(1))

5. Categorize risk level:

a. If score <= 10 → Low Risk (O(1))

b. If 11 ≤ score ≤ 30 → Medium Risk (O(1))

c. If score > 30 → High Risk (O(1))

6. Store results & generate report (O(n))

7. Display findings on dashboard (O(1))

**Complexity Analysis**

* **Process Scanning**: O(n) where n is the number of running processes.
* **Port Scanning**: O(m) where m is the number of scanned ports.
* **Log Retrieval**: O(log n) since event logs grow logarithmically.
* **Risk Assessment Conditions**: O(1) since each condition is checked in constant time.
* **Overall Complexity**: **O(n + m + log n)** (linear with log component).