Threat Matrix – Network Security Monitoring System

Project Overview

Threat Matrix is an advanced network security monitoring system built with a **cyberpunk-inspired design**. It provides **real-time visualization of threats, system resources, and security events** through an immersive and futuristic interface.
The system demonstrates the fusion of **cybersecurity concepts and real-time graphics programming**, offering both functional monitoring and engaging user interaction.

Executive Summary

The project implements a **comprehensive network security simulation platform** capable of monitoring, detecting, and visualizing network threats. Its features include:

- **Particle-based network visualization**
- **Interactive monitoring dashboard**
- **Threat alert and classification system**
- **Resource usage and traffic simulation**
- **Immersive cyberpunk UI with animations**

This project bridges **computer science, cybersecurity, and design**, making it both technically valuable and visually captivating.

Technical Specifications

Core Technologies

- **Frontend:** PyGame (Python-based rendering engine)
- **Architecture:** Object-Oriented Programming (OOP)
- **Graphics:** Real-time 2D particle system simulation
- **Platform:** Cross-platform (Windows, macOS, Linux)

System Requirements

- **Python:** 3.7+
- **PyGame:** 2.0+
- **Memory:** 512 MB minimum
- **Display Resolution:** 1400×900 (recommended)

Architecture & Design

Core Components

1. **Main Application (ThreatMatrix Class)**
   - Manages system state, rendering loop, and event handling.
   - Coordinates visualization, metrics, and UI layers.
2. **Particle System (Network Simulation)**
   - Represents network nodes as particles.
   - Features:
     - Boundary collisions & dynamic movement
     - Threat states: *Normal, Infected, Warning, Critical*
     - Infection spread simulation
     - Visual connections between nodes
3. **User Interface Layer**
   - Interactive **buttons** with hover effects.
   - **Threat alerts** with severity levels.
   - **Panels and dashboards** for metrics and logs.
4. **Visualization Engine**
   - Dynamic network graph.
   - Real-time animated metrics.
   - Special effects: scanlines, neon glow, pulses.

Key Features

1. **Real-time Network Visualization**
   - Particle-based node representation.
   - Color-coded threat indicators.
   - Infection spread & network graph dynamics.
2. **Monitoring Dashboard**
   - **System Resources:** CPU, Memory, Network Load.
   - **Network Stats:** Active nodes, infection spread, bandwidth, uptime.
3. **Threat Management System**
   - Classification levels:
     - 🔴 **Critical** – Immediate action
     - 🟠 **Warning** – Suspicious activity
     - 🔵 **Info** – General alerts
   - Real-time detection & historical tracking.
4. **Interactive Controls**
   - Initialize scan.
   - Pause/Resume monitoring.
   - Quarantine infected nodes.
   - Full system reset.
5. **Terminal & Logging**
   - Real-time log feed with timestamps.
   - Command prompt interface.
   - Persistent historical logs.
6. **Data Visualization**
   - Activity graphs.
   - Animated bars and indicators.
   - Color-coded status updates.

Technical Implementation

Rendering Pipeline

1. Background & layout.
2. Network graph (particles & connections).
3. UI panels & buttons.
4. Special effects (glow, scanlines, neon animation).
5. Optimized rendering pipeline for smooth performance.

Animation Effects

- **Pulse indicators** for threats.
- **Scanline sweep** across screen.
- **Glowing highlights** for active nodes.
- **Gradient progress bars** for resource usage.

Data Simulation

- **Sine-wave metrics** for CPU & memory.
- **Probability-driven infection spread.**
- **Dynamic network traffic simulation.**

User Interface Design

Layout

```
                      HEADER
 Logo | Status Indicators | System State

 LEFT      CENTER              RIGHT
 PANEL     DISPLAY             PANEL

   TERMINAL LOGS     |    NETWORK GRAPH
```

Color Palette

- **Primary (Active):** Cyan `#00d9ff`
- **Background:** Deep black & dark blue
- **Threats:**
  - Critical → Red `#ff0066`
  - Warning → Orange `#ffaa00`
  - Info → Blue `#00aaff`
  - Normal → Neon Green `#00ff41`

Algorithm Analysis

Particle System (Simplified Pseudocode)

```
for particle in particles:
    # Move node
    particle.position += particle.velocity

    # Bounce at boundary
    if boundary_collision:
        particle.velocity *= -1

    # Infection propagation
    if particle.infected:
        for neighbor in nearby_nodes:
            if distance < threshold and chance():
                neighbor.infect()
```

Threat Level Formula

```
Threat Level = 30 + (Infected Nodes / Total Nodes) * 40 + Random(-3, 3)
```

Optimization Techniques

- Connection distance culling.
- Limited node count (80 optimal, 150 max).
- Batch rendering for efficiency.

Security Simulation Features

- **Threat Detection:** Malware, unauthorized access, suspicious patterns, port scans, DDoS attempts.
- **Response Actions:** Auto-quarantine, manual isolation, full system scan.
- **Alert System:** Timestamped, severity-coded, log + visual notifications.

Performance Metrics

- **Frame Rate:** 50–60 FPS (target 60).
- **Memory Usage:** 50–80 MB.
- **Scalability:** Supports up to ~150 nodes before lag.

Code Quality Assessment

✅ **Strengths:**

- Modular & extensible design.
- Strong OOP structure.
- Consistent styling & animations.

- Good documentation.

⚠️ **Areas to Improve:**

- External configuration (colors, node limits).
- Unit testing framework.
- Accessibility (color-blind mode, screen readers).
- Advanced profiling for scaling.

Educational Value

- **Computer Science:** Real-time rendering, particle simulation, UI/UX design.
- **Cybersecurity:** Threat visualization, incident response, monitoring strategies.
- **Software Engineering:** Modular architecture, performance tuning, extensibility.