---

name: Interactive Security Dashboard overview: Build a terminal-based interactive security dashboard with real-time system monitoring, network analysis, and cybersecurity aesthetics. The dashboard will feature ASCII art, color-coded displays, and modular architecture for easy expansion. todos: - id: setup_structure content: Create project structure with modules directory and **init**.py files status: completed - id: create_config content: Create config.py with refresh intervals, color themes, and display settings status: completed - id: implement_ui_renderer content: Build ui_renderer.py with ASCII art banner, color functions, and layout rendering status: completed - id: implement_system_monitor content: Create system_monitor.py to collect CPU, memory, disk, and process statistics status: completed - id: implement_network_monitor content: Build network_monitor.py for network interfaces, connections, and traffic stats status: completed - id: implement_main_dashboard content: Create main dashboard loop in index.py with real-time updates and keyboard controls status: completed - id: create_requirements content: Create requirements.txt with psutil and colorama dependencies status: completed - id: create_readme content: Write README.md with installation, usage, and feature documentation status: completed

---

Interactive Security Dashboard - Kalizzer

Overview

A terminal-based security dashboard that displays real-time system information, network statistics, and security metrics in a visually appealing red team/ cybersecurity style. Built with Python for maximum portability and GitHub resume appeal.

Architecture

```
flowchart TD
    A[Main Dashboard] --> B[System Monitor]
    A --> C[Network Monitor]
    A --> D[Security Scanner]
```

```
    A --> E[UI Renderer]

    B --> B1[CPU Usage]
    B --> B2[Memory Stats]
    B --> B3[Process Info]

    C --> C1[Network Interfaces]
    C --> C2[Active Connections]
    C --> C3[Traffic Stats]

    D --> D1[Port Scanner]
    D --> D2[Service Detection]

    E --> E1[ASCII Art]
    E --> E2[Color Themes]
    E --> E3[Real-time Updates]
```

Implementation Plan

Core Components

1. **Main Dashboard (`index.py`)**
2. Entry point with menu system
3. Real-time refresh loop
4. Keyboard controls (q to quit, r to refresh)
5. Terminal clearing and cursor management
6. **System Monitor Module (`modules/system_monitor.py`)**
7. CPU usage percentage
8. Memory usage (used/total/percentage)
9. Disk usage statistics
10. Running processes count
11. System uptime
12. **Network Monitor Module (`modules/network_monitor.py`)**

13. Network interface list with IP addresses
14. Active network connections
15. Bytes sent/received per interface
16. Connection states (ESTABLISHED, LISTEN, etc.)
17. **UI Renderer (`modules/ui_renderer.py`)**
18. ASCII art banner/logo
19. Color schemes (red team theme)
20. Progress bars for metrics
21. Formatted tables and sections
22. Terminal size detection
23. **Configuration (`config.py`)**
24. Refresh interval settings
25. Color theme options
26. Display preferences

Features

- **Real-time Updates**: Dashboard refreshes every 1-2 seconds
- **Color-coded Metrics**: Green (normal), Yellow (warning), Red (critical)
- **ASCII Art Banner**: Cybersecurity-themed logo
- **Modular Design**: Easy to add new monitoring modules
- **Cross-platform**: Works on Linux, macOS, Windows
- **Clean Exit**: Proper terminal cleanup on quit

Dependencies

- `psutil` - System and process utilities
- `colorama` - Cross-platform colored terminal text
- Standard library: `os`, `sys`, `time`, `socket`, `subprocess`

File Structure

```
Kalizzer/
├── index.py                # Main entry point
├── config.py               # Configuration settings
├── requirements.txt        # Dependencies
├── README.md               # Project documentation
└── modules/
    ├── __init__.py
    ├── system_monitor.py   # System metrics collection
    ├── network_monitor.py  # Network statistics
    └── ui_renderer.py      # Terminal UI rendering
```

Future Expansion Points

- Port scanner integration
- Vulnerability scanner
- Log file monitoring
- Alert system for anomalies
- Export functionality (JSON/CSV)
- Customizable widgets
- Plugin system for third-party modules

Implementation Details

Key Functions

Main Loop (`index.py`):

- Initialize terminal
- Load configuration
- Enter refresh loop
- Handle keyboard input
- Clean exit

System Monitor:

- Use `psutil` for CPU, memory, disk stats
- Format data for display
- Calculate percentages and trends

Network Monitor:

- Enumerate network interfaces
- Get active connections via `psutil.net_connections()`
- Calculate network I/O statistics

UI Renderer:

- Generate ASCII banner with project name
- Create bordered sections for each metric
- Render progress bars for percentages
- Apply color coding based on thresholds

Visual Design

- **Banner**: Large ASCII "KALIZZER" or security-themed logo
- **Layout**: Multi-column sections for different metrics
- **Colors**:
- Red/Orange for headers and critical alerts
- Green for normal operations
- Yellow for warnings
- Cyan for network info
- **Progress Bars**: Visual representation of CPU/memory usage

Error Handling

- Graceful degradation if modules fail
- Permission error handling for system access
- Terminal compatibility checks
- Network interface fallbacks