Senior Project Report: NightWatch

**Student Name:** Charlotte Strobl

**Degree and Major:** Bachelors in Cybersecurity

**Project Advisor:** Professor Henderson

**Statement of Purpose**

In a perfect world, network systems would be easy to understand. Instead, they can be complicated and issues can be hard to diagnose. Learning tools like PowerShell can help, but it is time-consuming, complicated, and daunting for beginners. This steep learning curve makes it harder to stop issues before they cause downtime, data loss, or security risks. As networks become more critical, system administrators need tools they can use quickly and confidently.

NightWatch is a network analysis tool meant to bridge the gap between novice users and professional-grade tools. It benefits system administrators and cybersecurity learners by providing system information, network discovery, and port scanning features wrapped in a clean, visual interface. Designed using safe scripting and built for usability, NightWatch makes essential insights easy to access, understand, and act on—without requiring advanced expertise.

**Research and Background**

To build this tool, I researched ways to create a desktop application that could also run PowerShell commands. I discovered Electron, a framework that lets developers build desktop apps using web technologies like HTML, CSS, and JavaScript. It allows me to design a user interface similar to a website while running PowerShell commands in the background. This approach combines the ease of modern web development with the powerful system-level access that PowerShell provides.

**Project Resources**

* Electron - Cross-platform desktop app framework
* Node.js - Backend logic and PowerShell integration
* JavaScript - Back end PowerShell output processing
* PowerShell - Executes diagnostic commands
* HTML/CSS - Front-end layout and interactivity

**Project Requirements**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Function** | **Priority** | **Dep.** | **Description** | **Rationale** | **Fit Criterion** |
| 1.1 | Display Host Disk Information | High | – | Show total size, space remaining, and disk health | Essential system health monitoring | Disk size, space, and health shown |
| 1.2 | Display OS and Version | High | – | Show operating system and version | Inform users about system environment | OS name and version shown |
| 1.3 | Display System Name and IP Address | High | – | Show system name and IP address | Aid device identification | Name and IP in system info |
| 1.4 | Display Last Boot Time | Medium | – | Show last boot time | Indicates uptime and performance | Accurate time shown |
| 2.1 | Port Scanning Identification | High | – | Identify open/closed ports | Diagnose vulnerabilities | List and show port status |
| 2.2 | Port Scanning Visual Representation | High | 2.1 | Grid view of port status | Easier result comprehension | Visual grid with indicators |
| 2.3 | Port Information on Click | Medium | 2.2 | Click port for routing info | Clarify network paths | Show details in side panel |
| 3.1 | Identify Active IPs | High | – | Discover active IP addresses | Network awareness and troubleshooting | Active IPs displayed |
| 3.2 | Display Host Names | Medium | 3.1 | Show hostnames if available | Easier device recognition | Hostname next to IP |
| 4.1 | Side Navigation Panel | High | – | Persistent side navigation | Improves usability | Always visible links |
| 4.2 | Organized Dashboard Sections | High | – | Separate sections for each feature | Quicker access to info | Distinct dashboard areas |
| 5.1 | Minimalistic Dark Theme | High | – | Dark, clean design | Focus and eye comfort | Clean dark layout |
| 6.1 | Seamless Navigation | High | 4.1 | Quick switching between features | Better user experience | One-click transitions |
| 6.2 | Descriptive Explanations | Medium | – | Clear feature descriptions | Help for non-experts | Explanation per feature |
| 6.3 | User-Friendly for All | High | – | Friendly for both novices and experts | Wider usability | Both basic and advanced users benefit |
| 7.1 | Speed and Latency | High | – | Fast task completion | Avoid user frustration | <6 mins for 255 IPs |
| 8.1 | Non-Invasive Commands | High | – | Safe PowerShell commands only | Prevent system harm | Use harmless commands |
| 9.1 | Always Available | High | – | Work offline | Continuous access | No internet needed |
| 9.2 | No Downtime | High | – | No required maintenance downtime | Ensure availability | Zero interruptions |
| 9.3 | Low Maintenance | Medium | – | Minimal updates needed | Long-term usability | No frequent intervention |
| 10.1 | Back-End Error Handling | High | – | Handle backend errors | Ensure continuity | Show error without crash |
| 10.2 | Front-End Error Prompts | High | – | Show specific user-facing errors | Clear user guidance | Relevant messages displayed |
| 12.1 | Single User Support | High | – | One user session at a time | Avoid complexity | One active session only |
| 12.2 | Expandable for Future Scripts | High | – | Allow adding new scripts | Future growth | Seamless script integration |
| 13.1 | Visual Representation Improvements | Medium | – | Enhance visual displays | Better insights | Option to update visuals |
| 14.1 | Low Maintenance | High | – | Rare updates required | Reduce support overhead | Functional for at least 1 year |
| 14.2 | User Support Docs | Medium | – | Provide guides and FAQs | Improve self-service | Include documentation |
| 15.1 | Harmless PowerShell Scripts | High | – | Safe scripting practices | Prevent risk | Use verified scripts |
| 16.1 | Command History Tracking | Medium | – | Log all commands | Track and audit actions | Visible command list |
| 17.1 | No Sensitive Data Collected | High | – | Do not collect personal data | Ensure privacy | No data retention |

**Project Implementation Description & Explanation**

The tool offers several important features to help users better understand their systems and networks. Under system information, users can view the system name and IP address for clear identification, check disk usage including total and available space along with disk health, see connected network adapters, and verify the operating system version. For network discovery, the tool identifies active devices on the local network, matches IP addresses to hostnames when available, displays MAC addresses for better device tracking, and shows ping times to measure network latency. The port scanner functionality allows users to view which ports are open or closed, focus on common ports such as SSH, RDP, and HTTP, and interact with the scan results through an easy-to-use visual grid layout.

To implement these features, the program uses Node.js to run PowerShell commands and wait for their responses. PowerShell returns the output as an unformatted string, which JavaScript then parses into a more usable structure, such as an array or an object. After parsing, JavaScript updates the DOM elements to display the results to the user through the HTML and CSS front end.

**Test Plan**

The test plan focused on verifying both functionality and security. It covered all possible input combinations to ensure the program would not crash or fail under different conditions. The plan also tested the program across multiple networks to confirm that it displayed accurate system and network information. For any text-based inputs, tests were performed to check for cross-site scripting vulnerabilities. Lastly, the test plan involved peer feedback.

**Test Results**

The program successfully passed all but one of the test cases. The failed test involved an input field incorrectly accepting invalid entries if the first half of the input was valid. This issue will be addressed in future updates. Peer feedback was overall positive and provided valuable suggestions for improvement. One user noted that the interface was bland, which led to a redesign featuring a colorful space theme to make the program more engaging. Another user suggested adding the ability to specify a custom network for the network discovery feature. This enhancement will be considered in future updates.

**Challenges Overcome**

During development, several major challenges arose, most of them related to working with Electron. Finding clear and helpful documentation was difficult, so much of the early progress relied on trial and error. One of the biggest hurdles was connecting the front end to the PowerShell backend. I was also new to PowerShell and learning it along the way, which added an extra layer of difficulty. I had to learn a new programming language and a new framework simultaneously.

Originally, I planned to create separate PowerShell backend scripts that Node.js would execute through Electron. However, managing multiple backend connections quickly became complex and unreliable. I decided to restructure the backend design. Instead of calling multiple separate scripts, I established a single connection between JavaScript and PowerShell, which allowed JavaScript to send any command directly. To keep the system organized, I shifted encapsulation to the JavaScript side, using functions to manage backend operations and keeping them closely tied to the front end.

**Future Enhancements**

Looking ahead, there are many ways to expand this project. I would like to add new features, such as a traceroute visualization and a file transfer speed test. I also want to improve the performance and speed of the existing features and allow users to download their results. Based on peer feedback, I aim to add options for users to select a custom network range for the network discovery feature and to choose specific ports to scan in the port scanning feature. Overall, NightWatch has growth opportunity, and future updates will continue to make it more powerful, flexible, and user-friendly.