Software Design Document

Table of Contents:

Introduction:

1.1 Purpose:

- The purpose of the GUI application is to provide a comprehensive tool for conducting internal security audits within the organization. It aims to facilitate the following:
- Generate detailed security reports highlighting system details, compliance status etc.
- Enable administrators to manage and adjust security settings as per organizational policies.
- Facilitate the generation and analysis of logs related to security events and actions.
- Provide functionality to clear or delete memory caches on systems for security and performance reasons.

1.2 Scope:

The scope of the GUI application includes the following functionalities

- Security Reporting: The GUI application aims to generate comprehensive system reports, encompassing critical information such as:
    - IP addresses used by the system.
    - Status of antivirus software and recent scans.
    - History of USB devices connected to the system.
    - Compliance status with organizational security policies.
    - Status of specified services crucial for system operation.
    - Listing of folders present on the desktop.
    - Timestamp of the last Windows update applied to the system.
    -
- These reports provide detailed insights into system health, security posture, and compliance with organizational standards, aiding administrators in proactive management and security auditing.

- Settings Management: Enable administrators to disable or enable specific security settings based on audit findings.

- Logs: Capture and analyse logs related to security events, system changes, and user actions for auditing purposes.
    - Security logs
    - Display DNS
    - System logs
    - Application logs
    - USB logs

- Memory Cache Management: Provide functionality to clear or delete memory caches on systems to improve security and performance.
    - recycle bin
    - temp files
    - %temp files
    - prefetch files
    - The above-mentioned cache files will be cleared.

- The GUI application is designed to cater specifically to internal security auditing needs within the organization, ensuring compliance with security policies and proactive management of system vulnerabilities.

- This concise definition of purpose and scope should effectively communicate the goals and functionalities of your GUI application for internal security auditing. Adjust the details as per your specific implementation and organizational requirements.

Architectural Overview:

2.1 System Architecture

The GUI application leverages a hybrid architecture combining frontend technologies with backend processing capabilities to deliver comprehensive internal security auditing functionalities:

Frontend Components:

- Developed using **Electron** framework, combining HTML, CSS, and JavaScript (JS) to create a cross-platform desktop application.
- Provides a responsive and intuitive user interface (UI) for administrators to interact with various auditing features.

Backend Components:

- Python Scripts: Responsible for collecting system information such as IP addresses, antivirus status, USB device history, service statuses, desktop folder contents, and Windows update logs.
- Batch Files: Used for executing python scripts as administrator since, the application needs administrator privileges in order to perform all the functionalities that are mentioned earlier in the document.
- Node.js: Handles backend server operations and communication between frontend and Python scripts.

Integration and Communication:

- Electron facilitates seamless communication between the frontend and backend components, ensuring real-time updates and data synchronization.
- Data collected by Python scripts and batch files are processed and displayed through the Electron UI, enabling administrators to perform security audits efficiently.

2.2 User Interface Design

The user interface (UI) of the GUI application is designed to optimize usability and functionality for administrators conducting internal security audits:

Electron Window Management:

Utilizes Electron's window management capabilities to create multiple windows for different audit functionalities, enhancing user workflow and multitasking.

- Dashboard: Contains 6 buttons and a dropdown box and a result section.
- About Button: It includes detailed instructions on its usage, along with specific codes that indicate results and their corresponding meanings.
- Manul Button: It describes the settings that need to be done manually with clear process and implementation.
- Run Audit: Allows generation of detailed reports on system parameters such as IP addresses, antivirus checks, USB device history, service statuses, desktop folder contents, and Windows update logs.
- Audit Report Button: Its purpose is to display the audit report generated upon completing the audit run.
- Automate Services Button: Used to disable certain settings and also configure certain settings
    - Under Services RDP services will be disabled.
    - Secpol settings will be configured but few are not [6/9].
    - Telnet will be unchecked.
    - Firewall will be enabled.
- Logs Generate: Generates logs including Security logs, Display DNS logs, System logs, Application logs, and USB logs. These logs are saved as text files in a directory named "Windows Logs" located on the desktop.

This architecture and UI design ensure the GUI application effectively supports internal security auditing needs, leveraging Electron for cross platform desktop deployment and integrating Python, batch files, and Node.js for robust backend processing and frontend interaction.

Detailed Design

3.1 User Interface Details

3.1.1 Screens and Components

The GUI application features a screen with the following components:

- Dashboard:
  - Contains 6 buttons and a dropdown box.
  - Includes a result section for displaying outputs and audit reports.

- About Button:
  - Provides detailed instructions on application usage.
  - Displays specific codes with corresponding meanings to interpret results.

- Manual Button:
  - Guides users through manual settings adjustments with clear processes and implementations.

- Run Audit Button:
  - Initiates the audit process to generate comprehensive reports on system parameters:
    - IP addresses
    - Antivirus checks
    - USB device history
    - Service statuses
    - Desktop folder contents
    - Windows last update date etc.

- Audit Report Button:
  - Displays the audit report generated upon completing the audit run.

- Automate Services Button:
  - Allows users to automate service configurations:
    - Disables RDP services.
    - Configures Secpol settings (6 out of 9 settings).
    - Unchecks Telnet service.
    - Enable firewall.

- Logs Generate Button:
    - Generates logs including:
        - Security logs
        - Display DNS logs
        - System logs
        - Application logs
        - USB logs
    - Saves these logs as text files in the "Windows Logs" directory on the desktop.

This consolidated screen provides all necessary functionalities and controls for users to conduct internal security auditing, manage settings, and review detailed logs efficiently.

3.2 Implementation Details

3.2.1 Technologies Used

The application is developed using the following technologies, frameworks, and tools:

- Frontend:
    - HTML, CSS, JavaScript (JS)
    - Electron framework for desktop application development
- Backend:
    - Python for scripting and data processing
    - Batch files for running python scripts as administrator.
- Integration:
    - Node.js for backend server operations and communication.
- Folder structure:

    ├———Backend

    │ └———bat

    └———styles

    - CAS.jpeg [logo of the GUI application204]
    - GUI3.py [For Backup if front end doesn’t work]
    - index.html [front end of the GUI]
    - main.js [used to run the application]
    - packagelock. Json [default file initiated when a command npm init is executed]
    - package. Json [configuring the required dependencies]
    - renderer.js [for back end and frontend communication]

Under Backend we have some JS files and bat folder:

- logs.js [used to call and execute logs_generation.bat file]
- percent_temp.js [used to call and execute per_temp.bat file]
- prefetch.js [used to call and execute clearRecycle.py file]
- recycle.js [used to call and execute clearRecycle.py file]
- retrieval.js [used to call and execute retrieval.bat file]
- services.js [used to call and execute services.bat file]
- temp.js [used to call and execute temp.bat file]

Under bat folder we have some python and batch files:

- clearRecycle.py [used to clear recycle bin (unwanted deleted folders)]

- logs_generation.bat [calls and executes logs.py and runs it as administrator]

- logs.py [used to generate logs and save them in windows logs folder in desktop]

- per_temp.bat [calls and executes clear_Temp.py and runs it as administrator]
- clear_Temp.py [used to clear %temp folder (cache) and called using per_temp.bat]

- prefetch.py [used to clear prefetch folder (cache)]

- retrieval.bat [calls and executes retrieva.py and runs it as administrator]
- retrieval.py [used to run the internal audit and generate audit report]

- services.bat [calls and executes services.py and runs it as administrator]
- services.py [used to configure and disable RDP services, Secpol services and disable Telnet and enables firewall and called using services.bat]

- temp.bat [calls and executes temp.py and runs it as administrator]
- temp.py [used to clear temp folder (cache) and called using temp.bat]

Under styles folder we have css files for the frontend:

button.css	dropdown.css	
card.css	light.css	toggle.css
dark.css	loader.css	table.css

3.2.2 Coding Standards

The development adheres to the following coding standards:

- Consistent naming conventions for variables, functions, and files.
- Modular and reusable code components.
- Use of version control (e.g., Git) for collaborative development.

Deployment

4.1 Deployment Environment

- The GUI application is intended for deployment on Windows operating systems. It is designed to run on desktop environments where administrators can access system configurations and execute auditing tasks.

4.2 Installation Instructions

- To install and configure the application:

  - Download the Executable (exe) File: Obtain the executable file of the GUI application.

  - DoubleClick to Run: Simply DoubleClick on the executable file to launch the application.

  - Follow Onscreen Instructions: The application will guide you through its functionality using the dashboard interface.

  - No Installation Required: Since it's an executable file, no formal installation process is necessary. Users can run the application directly from its location.

Notes:

1. System Requirements: Ensure that the target system meets the minimum requirements to run the GUI application, including Windows version compatibility.
2. Executable Security: Depending on security settings, you may need **administrative privileges** or permission to run the application.

Conclusion

5.1 Summary

- This Software Design Document (SDD) outlines the architecture, functionality, and deployment considerations of the GUI application developed for internal security auditing. Key components of the application include a consolidated dashboard with various buttons and a result section, facilitating tasks such as running audits, generating detailed reports, managing system services, and logging activities. The application is designed to operate efficiently on Windows desktop environments, utilizing technologies like Electron framework for frontend development and Python, batch files, and Node.js for backend operations.

5.2 Future Considerations:

- Looking ahead, several enhancements and features could be considered for future iterations of the application:

- Enhanced Reporting Capabilities: Introduce more advanced reporting features, such as graphical representations of audit results and trend analysis.

- Automated Scheduling: Implement scheduling options for audits and automated tasks, improving operational efficiency.

- Realtime Monitoring: Enable real-time monitoring of system events and security metrics, providing administrators with immediate insights.

- Integration with Cloud Services: Support integration with cloud-based storage solutions for secure and scalable log management.

- Enhanced User Management: Introduce user authentication and role-based access control (RBAC) features to manage application access and permissions securely.

- These enhancements aim to further empower administrators in managing and auditing internal security measures effectively while enhancing overall usability and scalability of the GUI application.