**Architecture & Design**

**PhishNet**

**Version 2.0**

**Updated 11/25/2024**

**—**

Aarshia Hukmani

Gianmarco Bonventre

Julia Pallazola

AJ Whitman

**Table of Contents**

1. **Introduction**...................................................................................................1

1.1 Purpose

1.2 Scope

1. **Architecture Model**.......................................................................................1

2.1 Architecture Overview

2.2 Architecture Style

2.3 Key Implementation Decisions

2.4 System Objectives and Limitations

1. **Design Patterns**..............................................................................................4

3.1 Adapter Design Pattern

3.2 Interface and Backend Design Patterns

1. **Key Components**............................................................................................5
2. **Interface and Backend Integration/Workflow**............................................8

5.1 User Interfaces

5.2 Third-Party Integrations

5.3 Communication Protocols

5.4 Backend Integration

5.5 Data Formats and Protocols

5.6 Criteria for Integration

5.7 Workflow Diagram

1. **Introduction**

This document is the authoritative guide to Phishnet’s software architectureand design, providing a clear resource for understanding our software's structure and design practices. This document was created to address the need for accurate information, which helps prevent the misallocation of resources and unintentional changes to key features that can occur when relying on outdated or unclear data. By displaying a detailed and up-to-date overview of the software’s architecture, this document aims to simplify development processes and protect the integrity of the software. Furthermore, this document plays a key role in transparently documenting the project’s progress in line with our strategic project plan. A thorough explanation of how the system will be built to meet the specified requirements is included. The precision and clarity in this guide are vital for keeping the project on track and aligned with Phishnet’s goals and objectives, making it a valuable asset for project managers, stakeholders, and developers.

1. **Architecture Model**

**2.1 Architecture Overview**

The architecture overview gives a clear look at core functionalities and key components for our anti-malware application, PhishNet, which sets the foundation for understanding how each piece works together. Our application aims to create an easy and efficient way to scan and remove malware using the ClamAV engine, exclusively on Linux operating systems. The user interface will be developed using the QT framework, which is built for C++ and provides a native performance, ensuring effective and responsive application execution. The backend of our application will be using C++ along with the ClamAV toolkit for reliable malware detection and SQLite database for storing scanning data.

**2.2 Architecture Style**

PhishNet uses a Model-View-Controller (MVC) architecture pattern. This ensures that the user can directly interact with PhishNet and access all of its functionalities through a user-friendly interface. The model manages the virus bytecode database, processes the results of each scan, maintains the state of quarantined files, and maintains backend communication to update virus bytecodes. The view is the front end, where users can interact with the system. All navigation and scanning buttons are included in the view. This is also where scanning results and reports are displayed to the user. The controller processes the inputs coming from the view and sends commands to the model to update the view. For example, the controller handles start scan button inputs and navigation to the different pages through the home menu.

PhishNet uses a client-server architecture split into a front-end GUI and a back-end processing unit. The client is the front-end GUI, which allows the user to initiate scans, schedule scans, and manually update the virus bytecode database. The client is also responsible for displaying the report created at the end of each scan. The server is responsible for using ClamAV to analyze files, maintain the virus bytecode database, and store scanning reports and quarantined files.

PhishNet utilizes a layered architecture to ensure efficient scanning and easy-to-use functionality. This architecture allows for easy maintenance, efficient threat handling, and a smooth user experience while also making sure that changes to one layer won't affect the others. The architecture divides the system into five layers, each overseeing specific tasks. At the top, The user interface layer will provide a user-friendly way to select scan types and manage Quarantined threats. Below the interface, the application layer will perform scanning operations and handle user decisions on detected threats. Next is the security layer, which integrates ClamAV’s scanning engine and signature database using freshclam for updates. The file system layer will handle which directory or files the user wants to scan utilizing ClamAV’s libclamav to do so efficiently. The last layer will maintain records of quarantined files and scan history for all users.

**2.3 Key Implementation Decisions**

*ClamAV Integration:* ClamAV is an open-source antivirus engine and toolkit tailored to Linux systems. It allows for the detection of different viruses. Integrating ClamAV ensures our application is reliable and well-established in all areas of virus detection.

*Qt Framework:* Qt framework for our applications UI allows seamless integration with our backend development and the Linux environment. The framework's capabilities will be used to create a robust interface that aligns with the program and environment's requirements.

*SQLite Database:* SQLite is a lightweight, file-based database system that allows for the storage of malware detection logs without significant system overhead. It will also allow for quick tracking of scanning history with low memory consumption, as the database is directly read and written on the user's local file system. The use of SQLite contributes to the idea of maintaining a fast and easy-to-use anti-malware application.

**2.4 System Objectives and Limitations**

Objectives:

* *Fast/Efficient Malware Detection:* Our anti-malware application's main goal is to give users an efficient and reliable way to scan their Linux system for malware.
* *Minimal System Impact:* Our goal is to prioritize user experience, as it is crucial that the application won't slow down or impact the user’s computer or other applications.
* *System Security:* Our application must enhance the security of Linux systems by ensuring the detection and removal of any malware on a user's device.

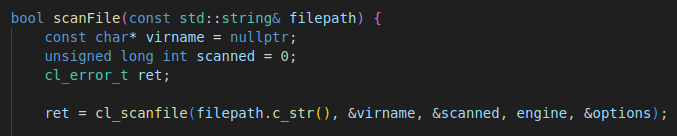
Limitations:

* *Platform Specificity:* PhishNet is currently designed exclusively for Linux systems and is not compatible with operating systems such as Windows or MacOS.
* *Dependency on ClamAV:* The application relies solely on ClamAV’s signature database and its virus detection capabilities, which could result in potential delays in detecting new threats.
* *Database Size and Management:* SQlite’s database will grow over time as the user continues scanning, which could impact performance with an extensive scan history.

1. **Design Patterns**

**3.1 Adapter Design Pattern**

A major design pattern of PhishNet is the backend code for each scan type. While the user may believe that each scan type is configured very differently, each is very similar and varies only by the difference of what is being scanned. With this, a custom scan is used as a base for each other scan type, as each type is essentially a variation of a customized scan. This creates a changeable core that allows for the different scan types expected of the software but a seamless backend code for their implementation.



**3.2 Interface and Backend Design Patterns**

PhishNet uses a repository pattern through the implementation of GitHub to store code that is separated between the frontend UI and backend. This separation allows for organized maintenance and the simultaneous modification of multiple modules. By utilizing GitHub's branch feature, versions can easily be tracked, and code that has been checked out and pushed.

1. **Key Components**

PhishNet comprises multiple essential components, each contributing significantly to the software’s overall performance. Below is an outline and explanation of how each component will be integrated into the final anti-virus software:

* Scanner Engine:
  + ClamAV is the back-end component used to scan files and update virus bytecodes. By calling the existing scanning functions and modifying them to work with new custom scanning methods, the scanner engine pulls from the virus definition database to compare the user's files to the malicious content. The scanner generates a report based on the results displayed to the user. If necessary, the scanner moves infected files to a quarantine database.
* Virus Definition Database:
  + The Virus Definition Database is derived from the ClamAV Signature Database, which houses an extensive array of malware signatures and patterns. Before scans are run, the virus database will update automatically to ensure the most accurate results. When the ClamAV repository is updated, PhishNet will detect that it has changed, and the application virus database will update when the user begins a scan. The update will occur between the user initialization of a scan and the start of the actual scan.
* Quarantine File Manager:
  + The Quarantine File Manager is tasked with overseeing and storing files and processes flagged as potentially harmful periods, it isolates suspicious items to prevent them from jeopardizing system security. After scanning is completed, if the scanner detects any infected files, those files will be moved to a quarantine directory within the PhishNet application. Users can view and manage quarantined files through the user interface. If a user chooses not to delete a file, it will remain in the quarantine folder until it is deleted
* User Interface:
  + The UI provides an intuitive platform for users to engage with the PhishNet application. It serves as the entry point to various features and functionalities. The user interface is broken down into four main pages: scans, schedule scans, quarantine, and history. Users can initiate scans, schedule scans, view and manage quarantine files, and view curated history reports through the interface. The UI will be created using C++ and the Qt libraries. Qt Creator is used to visualize and build GUI components using a graphic interface.
* Scheduler:
  + The Scheduler enables users to establish automated scanning tasks at a predetermined time or interval, ensuring regular system checks for malware without the need for manual intervention. By connecting to the Linux clock using ‘crontab’ the user can select a time, frequency, and scan type to be performed. When the user schedules a scan type, that scan is called at the specified time in the user’s system background. A report is created, and the user can view it in the history section of the application.
* Scan History and Reporting:
  + This component is vital for monitoring application activities, documenting scan results, and generating reports that provide a detailed view of the system's security status. Once the scanner engine has completed a scan and moved the necessary files, a report is generated for the user to view. Through the user interface, the user can view the file paths to the infected files, see what time the scan was initiated, and view the details of individual scans.

Understanding how these components interact and collaborate is essential for grasping the architecture of PhishNet. These components offer robust and user-friendly antivirus solutions with the latest threat detection capabilities. The accompanying diagrams below illustrate the interactions among these components.

**5. Interface and Backend Integration/Workflow**

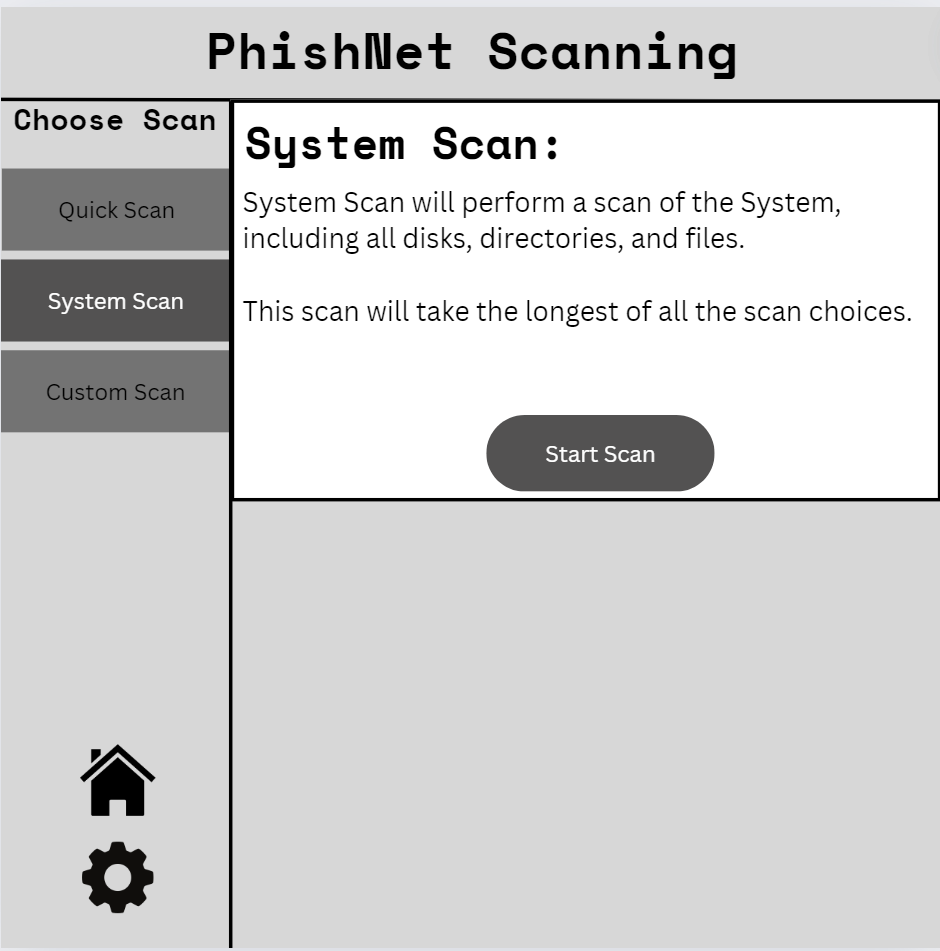
**5.1 User Interfaces**

PhishNet will provide a graphical user interface (GUI) for users to interact with the backend ClamAV system. We used C++ with the Qt library to create the GUI framework. The following is a brief description of the UI elements:

Home Screen: When the PhishNet application is started, this is the first page that the user will see. Each button will navigate to it’s corresponding page.

****

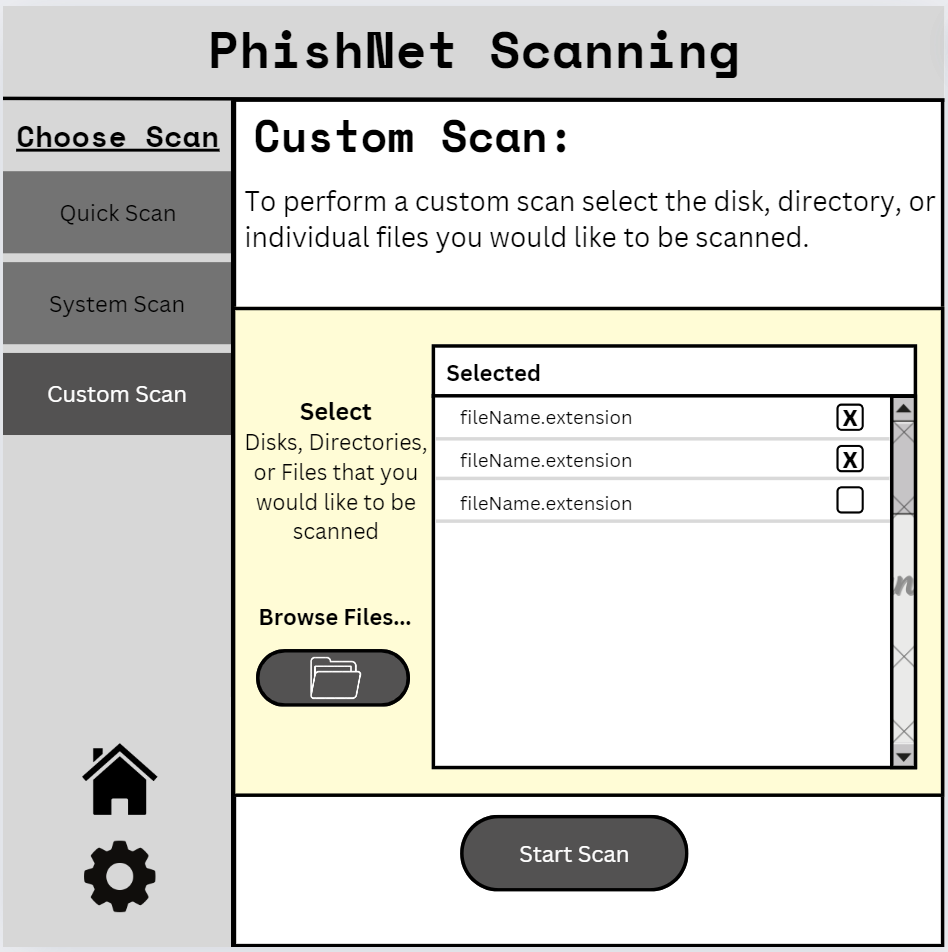
System Scan: This scan type allows the user to perform a scan of all disks, directories, and files.

****

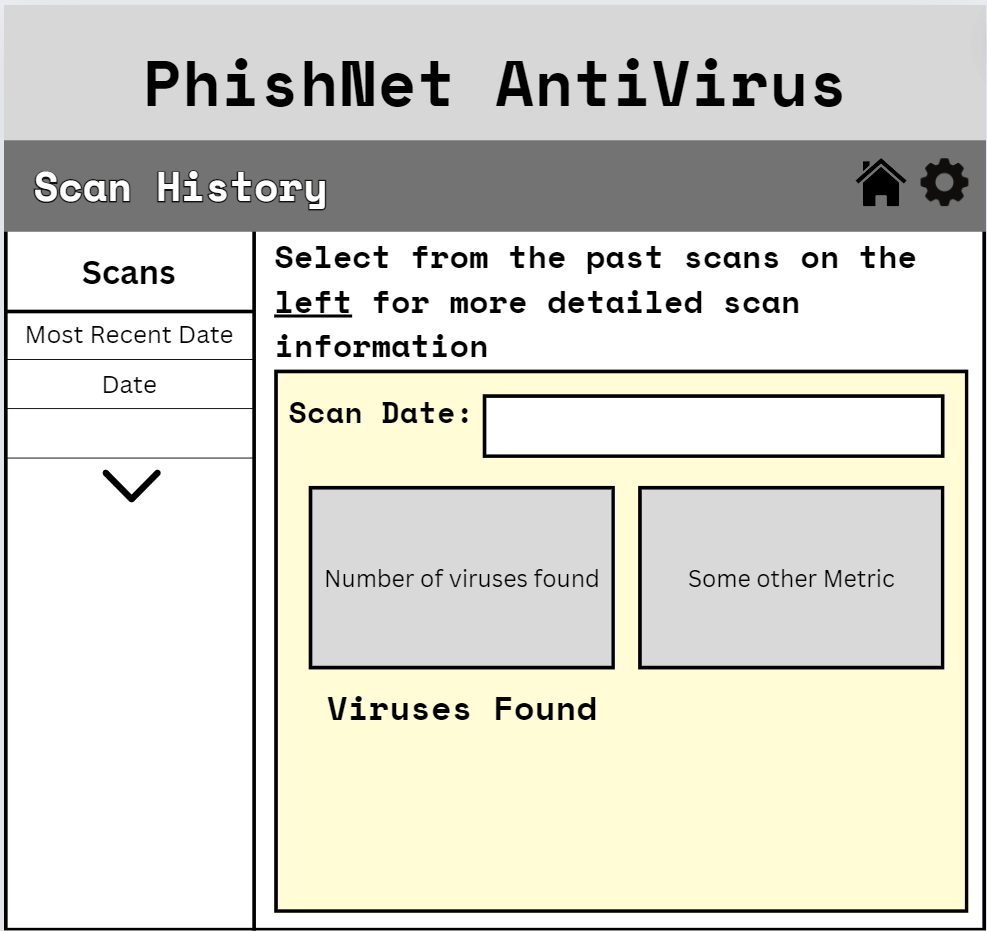
Quick Scan: This option allows the user to perform a scan of critical system areas and flagged directories.

****

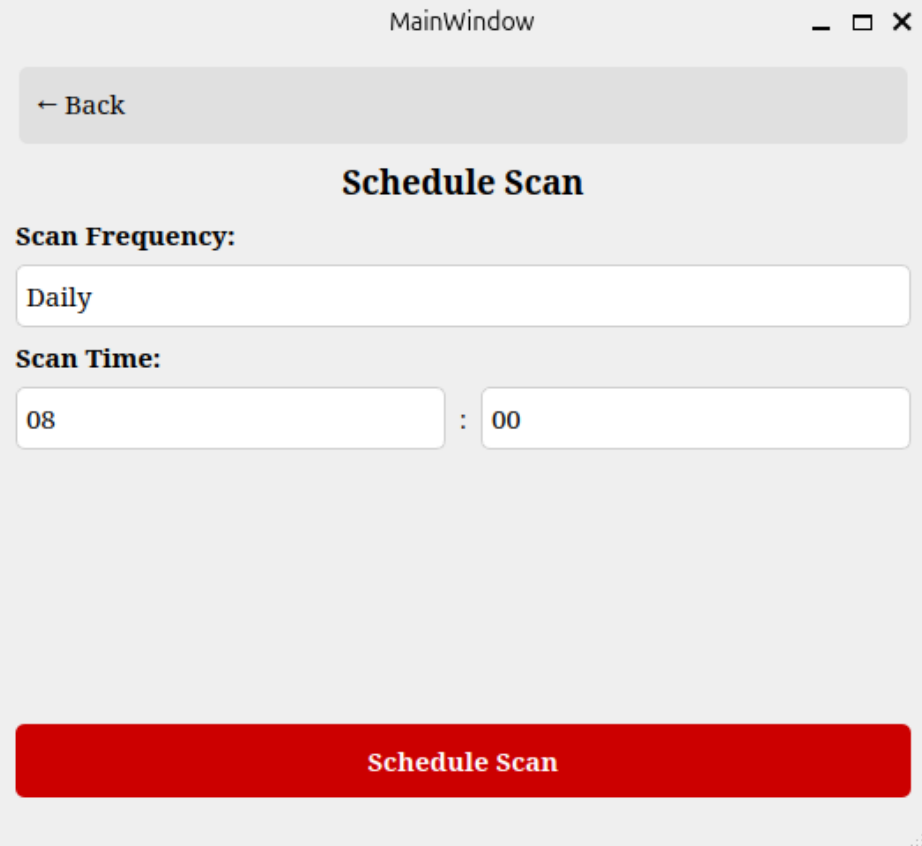
Custom Scan: The user can choose which disks, directories, or files to scan. Once they have chosen the content to be scanned, they can start scanning.

****

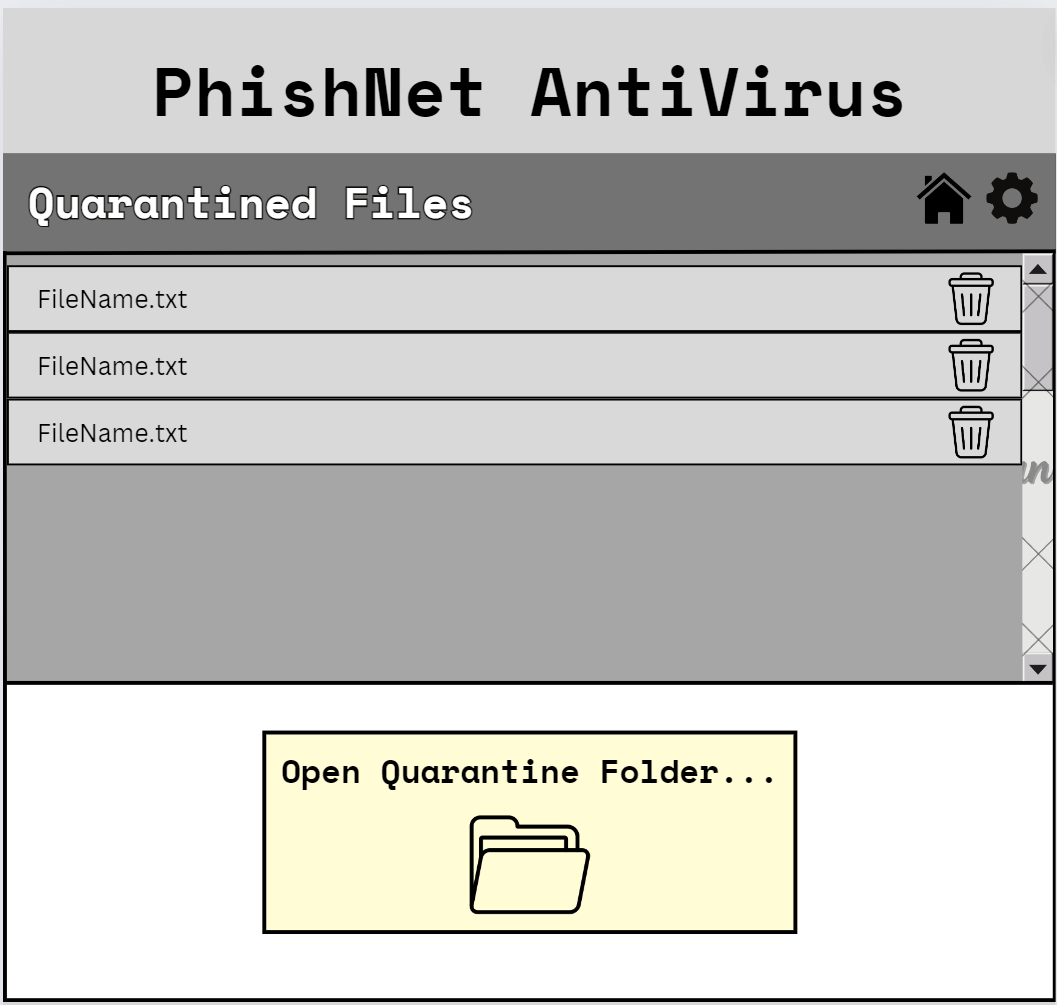
Scan History allows users to access all reports generated through scanning. Users can search by date or by selection in the scroll menu.



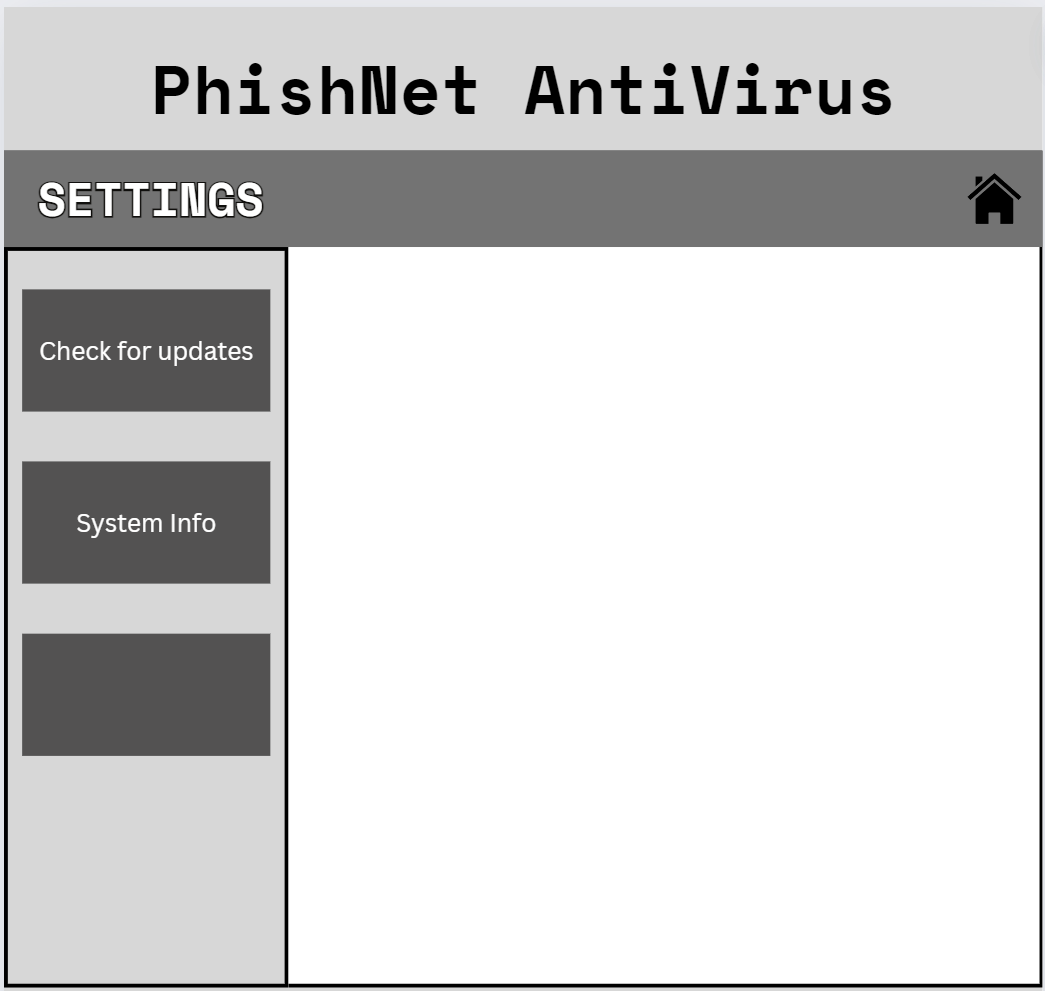
Scheduled Scan: Users will be able to select a time and frequency for a scan (System Scan or Quick Scan) and schedule daily, weekly, or monthly scans.



Quarantined Files: The user can view all current quarantined files before deleting them. They can do this through the file viewer or their own file manager.



Settings: The user can check for updates and also view their system information. When a user checks for updates, they can choose to update their application if an update is available.



**5.2 Third-Party Integration**

PhishNet uses ClamAV for all detection and scanning functionalities. ClamAV is an open-source antivirus engine. Using C++, ClamAV processes are called, and functions are created to integrate other features, such as curating history reports.

**5.3 Communication Protocols**

PhishNet only uses internal communication. ClamAV processes are called with C++, and the user accesses these processes through the GUI.

**5.4 Backend Integration**

PhishNet’s backend consists of C++ functions that will interact with ClamAV to perform scans and other features. ClamAV has predefined functions for different processes that will be directly called by PhishNet. Users will be able to use the GUI to navigate to the features they would like and the functions they want to perform. The results from each function call will be returned to the GUI in a user-friendly format.

*System Scan*: This function will scan the entire system, including all disks and directories. It will utilize the same technique as a custom scan, selecting all files, drives, and directories for ClamAV to scan. After this scan is performed, a log will be generated detailing a report of the findings. This report will be stored in a directory called ‘History’ created within the ClamAV engine and a ‘Last Scan’ directory until the next scan is performed.

*Quick Scan:* This function will scan all essential system directories/files and directories that have been flagged for having malicious files in the past. After scanning, a report will be generated.

*Custom Scan:* This function will scan user-selected drives, directories, and individual files. The user can choose as many files as they would like to scan. After scanning, a report will be generated.

*Schedule Scan:* This function allows the user to schedule scans for a certain day and time. Users select the type of scan they would like performed (Quick Scan by default) and can choose to have a scan occur daily, weekly, or monthly.

*Virus Bytecode Updates:* To keep users most protected, PhishNet will offer automatic updates when they are available through ClamAV. Updates will include the most current virus bytecodes/signatures. Users can also opt out of automatic updates if they prefer to update their applications manually.

*File Quarantine:* Infected files will be secured in a Quarantine directory within PhishNet. Users can access these files through the ‘Quarantined Files’ section or their file manager.

**5.5 Data Formats and Protocols**

Strings for file paths and scan results will manage the data exchange between the PhishNet GUI and the ClamAV backend.

**5.6 Criteria for Integration**

*Consistency:* The user interface will maintain a clean and friendly design.

*Reliability:* PhishNet will set a high standard for reliability by testing each scanning method rigorously in a virtual Linux environment. This will be done with both ClamAV testing files and uniquely curated testing files and directories.

*Scalability:* Building PhishNet with a modular architecture will facilitate incorporating new features and third-party tools.

*Performance:* PhishNet will be a lightweight application that allows scans to occur among other system processes without causing lag or delays. Using the ClamAV engine and a C++ GUI will make our application swift and memory efficient.

**5.7 Workflow Diagram**

