****

**Software Requirements**

**Specification**

**for**

**SEng Malware (SEM)**

**Version 0.5**

**Prepared by:**

1. John Dornheim
2. Chris Durand
3. John Schlatter
4. Shelby Mitchell
5. Alex Lopez
6. Jon Harrity

**CSC 190 Term Project**

**Advisor:** Dr. Scott Jeffreys

Spring 2019

*Made with love in Axinn 806*

**Table of Contents**

1. Introduction

1.1 Preface  
 1.2 Versions

1.3 Overview

1.4 Glossary

2. User Requirements

3. System Architecture

3.1 User Interface & Program Entry

3.2 Virus Database

3.3 File Scanner

3.4 Threat Quarantine

4. System Requirements

5. System Models

6. System Evolution

7. Appendix

1. **Introduction**

**1.1 Preface**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reasons For Changes** | **Version** |
| Team | 2/10/19 | Began initial planning + designing of SEng Malware | 0.1 |
| Team | 2/10/19 -- 4/20/19 | Creation and implementation of SEng Malware | 0.2 |
| Team | 3/3/2019 | Changed allowance of user deleting files  Additional details for User Requirements(Scan function, # of user scheduled scans, when database is updated)  Updated technology versions for Windows, Python, and Electron | 0.3 |
| Team | 3/25/2019 | Removed Python, this will no longer be used.  Updated User requirements with additions(2.1, 2.2.1, 2.2.2, 2.3.1, 2.3.2, 2.4.2, 2.4.3, 2.4.4)  Updated System Requirements with additions(4.13/4.14)  Updated System Model and UI pictures | 0.4 |
| Team | 3/29/2019 | Stopped multiple scans from running at the same time. | 0.5 |
| Team | 4/7/2019 | UI Redesign  Update UI requirement 7.1.3, 7.1.4 | 0.6 |

**1.2 Versions**

|  |  |  |
| --- | --- | --- |
| **Version** | **Additions** | **Changes** |
| 0.1 | Initial Requirements | - |
| 0.2 | 1. ClamAV will be used as the engine |  |
| 0.3 | 1. Windows 10, Python, and Electron versions added 2. Users will only be able to schedule 1 scan at a time 3. Virus Signatures Database will update on application start-up | 1. An option for the user to delete a file will no longer be included in the requirements. All flagged malware will be quarantines automatically instead. 2. PyClams library will no longer be used |
| 0.4 | 1. Format of Schedule scans will be MM/DD/YYYY 2. Logs will be stored in YYYY-MM-DDscanlog.txt format | 1. Python will no longer be used, this was an unnecessary step. 2. 2 types of scans will be allowed, “Deep” which will scan the whole drive and “Shallow” which will scan vulnerable drives. 3. System Model/UI changed to reflect actual changes |
| 0.5 | 1. Only one scan will be allowed run at a time. 2. Cancel button will be added that will kill a scan |  |
| 0.6 |  | 1. UI Redesign, scans will be located on the same page, menu will no longer exist, buttons for logs and scheduling scan will be added to the top right corner 2. Updated exclusions to only include the Quarantine folder 3. Program will no longer run at start-up, it will manually start from user .exe |

**1.3 Overview**

In an increasingly connected world, security has become a necessary tool for every-day end-users. According to Norton Security, the U.S. will account for half of all breached data by 2023. In addition, total malware infections have been increasing over the last 10 years. In an effort to combat the globally growing malware problem, scanning for viruses will help mitigate this risk. SEng Malware is a malware scanner that will help the clients of Enterprise Software protect their devices from malicious files. With its constantly-updated malware database, SEng Malware allows users to scan parts or all of a file system for malware, and keeps records of any malicious files it finds. Users can schedule scans of their files for peace of mind, and have the option to quarantine malware that has been found. Our primary objective is to allow end-users to effectively monitor and contain possible malicious files on their Windows devices with an easy-to-use program that even the least savvy of computer users can utilize to its full potential.

**1.4 Glossary**

Throughout this document, many technical terms will be used. Their working definitions as applicable in this document are defined below.

**Back-end**: Or data access layer, is the part of the computer system or application that is not directly accessed by the user, typically responsible for storing and manipulating data.

**Front-end:** Or presentation layer, is the part of the computer system or application in which the user interacts directly.

**Functional Requirement:** A statement of some function of feature that should be implemented in a system.

**Git:** A distributed version management and system building tool where developers take complete copies of the project repository to allow concurrent working.

**Github:** A server that maintains a large number of Git repositories. Repositories may be private or public. The repositories for many open-source projects are maintained on GitHub.

**GUI**: Graphical user interface. A GUI is the interface through which user interact with a program. Specifically, the GUI will be used here to communicate information to the end-user, and accept user input.

**Interface:** A specification of the attributions and operations associated with a software component. The interface is used as the means of accessing the component’s functionality.

**Malware**: Software that is specifically designed to disrupt, damage, or gain unauthorized access to a computer system.

**Nonfunctional Requirement:** A statement of a constraint or expected behavior that applies to a system. This constraint may refer to the emergent properties of the software that is being developed or to the development process.

**Open Source:** An approach to software development where the source code for a system is made public and external users are encouraged to participate in the development of the system.

**Python:** A programming language with dynamic types, which is particularly well-suited to the development of web-based systems.

**Risk:** An undesirable outcome that poses a threat to the achievement of some objective. A safety risk is a measure of the probability that a hazard will lead to an accident.

**Risk Management:** The process of identifying risks, assessing their severity, planning measures to put in place if the risks arise and monitoring the software and the software process for risks.

**Server**: A computer program which manages access to a centralized resource or service in a network.

**Software Architecture:** A model of the fundamental structure and organization of a software system.

**User Interface Design:** The process of designing the way in which system users can access the system functionality, and the way that information produced by the system is displayed.

**Version Control (VC) Systems:** Software tools that have been developed to support the process of version control. These may be based on either centralized or distributed repositories.

1. **User Requirements**

**2.1** The user will be able to scan files on their system for malicious programs.

**2.1.1** The user will be able to choose between two types of scans, a “Deep” scan and a “Shallow” scan

**2.2** The system will update the virus definitions from ClamAV’s virus databases

**2.2.1** The system will update all 3 of ClamAV’s virus databases (main.cld, daily.cld, and byte.cld)

**2.2.2** The system will update the database upon start-up of the application

**2.3** The system keeps a log of malicious files quarantined

**2.3.1** The logs will be stored in the format as follows: YYYYMMDDscanlog

**2.3.2** The logs will be displayed as a list in its own tab

**2.4** The user can schedule scans by date and time.

**2.4.1** User will only be able to schedule 1 scan at a time

**2.4.2** User will only be able to schedule a scan by MM/DD/YYYY

**2.4.3** User will be able to specify the type of scan

**2.4.4** If the application is not open for a scheduled scan, the scan will be

postponed until the next time the application starts

1. **System Architecture**

SEng Malware will have four system modules. There will be a user interface and program entry module, a virus definition module, a file scanner module, and a threat quarantine module.

**3.1** User Interface and Program Entry

**3.1.1** Display scanning results: Users will be shown the results of previous scans. For

each previous scan, the directories scanned will be listed. Any malware detected will

be listed, including the path to the program, the matching virus definition, and the

action taken to handle the issue.

**3.1.2** User Configurations

1. Scheduling scans by date and time of day.
2. History of files quarantined via logs.
3. File exceptions for false positives.

**3.1.3** Settings: Users will be provided options for scheduling a scan. They can choose between a deep and shallow scan, including selecting the folders to scan if a deep scan is selected, and selecting the time and date to scan at.

**3.1.4** Main

**3.2** Virus Database

**3.2.1** Provide button to update antivirus definitions.

**3.2.2** Accesses “signature byte code and patterns” from database.

**3.3** File Scanner

**3.3.1** Scans specified directories, disks, or individual files.

**3.3.2** Flags files that are a “risk”.

**3.3.3** Cancel button will cancel a scan currently running

**3.4** Threat Quarantine

**3.4.1** Logs quarantine history.

**3.4.2** Removal/Quarantine of malicious files.

1. **System Requirements**

**4.1** Users can schedule scans for specific dates and times.

**4.2** Users will only be able to schedule 1 scan at a time

**4.3** Users have the ability to perform a “Shallow Scan” which will scan the Windows and Temp folders

**4.4** The scanner utilizes signature byte code detection and pattern matching.

**4.5** The system will update the virus definitions on a regular basis via the ClamAV server database.

**4.6** Malware definitions will be provided by ClamAV

**4.8** Upon completion of a scan, the software application will quarantine the malicious file(s) by moving it into a specified file.

**4.9** The system shall be developed to run on Windows 10 v1809.

**4.10** Electron will be used to develop the GUI.

**4.11** The software will have the option to run at startup.

**4.12** A log will be generated upon completion of each scan, showing which files were

flagged and quarantined. Format YYYY-MM-DDscanlog.txt.

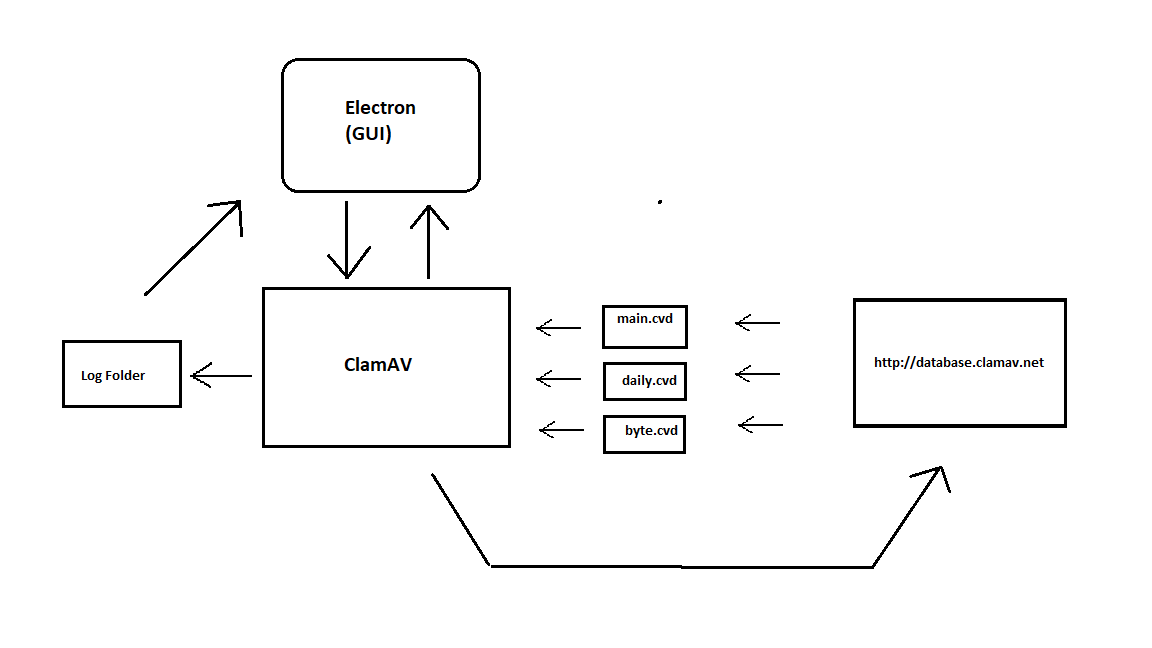
**4.13** The Deep Scan will search the entire C drive of a user’s computer

**4.14** The Shallow Scan will search a user’s C:\Windows folder

**4.15** Only one scan will be allowed to run at a time

1. **System Models**

**5.1 \*\*Diagram Showing the relationships between the system components and its environment\*\***



1. **System Evolution**

**6.1** Heuristic Detection Algorithm Integration - The system will analyze previously deleted malware’s bytecode in order to predict new bytecodes that future malware may use.

**6.2** Easy Installer - Add a general installer to make is easier for end users to install the application on their system, whether it be Windows or Mac.

**6.3** Quality Assurance Management - Set up system to incorporate quality assurance handling from end users.

**6.4** Mac Version - create a version of the application for a Mac environment.

**6.5** Monetize the application - add subscriptions/user accounts

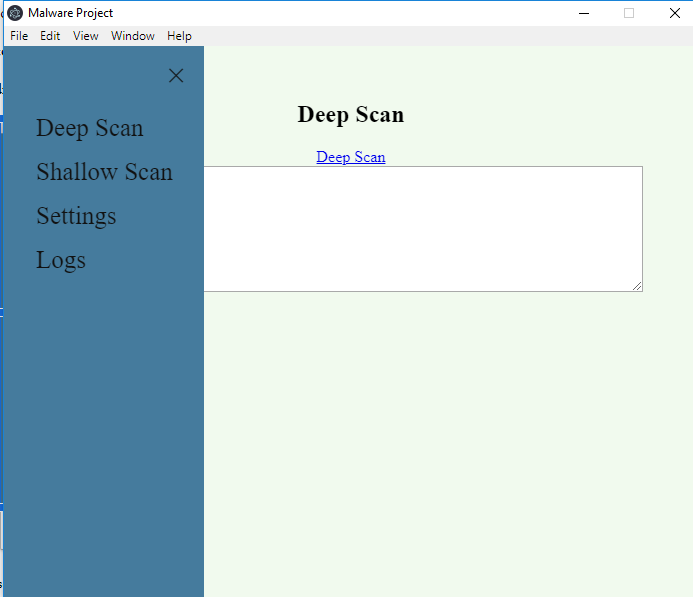
1. **Appendix**

**7.1 Functional Requirements**

**7.1.1** Graphical User Interface (GUI)

**7.1.2** Virus Scanners

1. Users will be able to choose whether they would like to conduct a Deep scan or a Shallow scan to check their computer for malware. The Surface scan will search for the most popular directories where malware is commonly found, whereas the Deep scan will search the entire specified drive
2. Malware that is found will be flagged, highlighted in red and will be automatically quarantined in the QUARANTINE folder

.

**Example of home screen (above)**

**7.1.3** Exclusions

1. Quarantine file will be excluded from the scan

**7.1.4** Run

1. Program will run manually by the user via the executable.

**7.1.5** Scheduled Scans

1. Users will be able to specify a month, day, and year to schedule a scan.

**7.1.6** Quarantine

1. Malware detected on a user’s computer will be quarantined automatically.

**7.1.6** Update Definitions of Viruses

1. Virus definitions taken from an ClamAV’s database main, daily, and byte
2. Definitions will be updated from ClamAV’s server upon application start-up

**7.1.7** Record Quarantine History

1. The names and information surrounding a quarantined strain of malware will be recorded in a log file. The user can access their logs to see the history of the malware scans.

**7.2 Non-Functional Requirements**

**7.2.1** Version Control

1. Version control handled by Github, https://github.com/jonharrity/Seng-Malware

**7.2.2** Operational

1. SEng Malware aims to be up and running 99% of the time to make sure that users always feel secure.

**7.2.3** Technology Choice

1. SEM focuses on a Windows 10 system because the majority of laptop and desktop users run on a Windows OS.
2. Electron v2.0.2
   1. Utilizes node.js and html to create the user interface.
3. ClamAV
   1. Open source antivirus engine.

|  |  |
| --- | --- |
| **Functional** | **Nonfunctional** |
| GUI (electron) | VC (Github) Version Control |
| Variable Scanner (Shallow/Deep)(Single Directories) | Operational (Uptime for Server)(9% uptime) |
| Exclusions (of files) | Technology Choice:  Windows 10 v1809  Backend: Electron 2.0.2 |
| Auto Run |  |
| Timer (Scheduled Scans) |  |
| Quarantine |  |
| Update definitions of viruses |  |
| Record Quarantine (LOGS) |  |

**(Contributions on final page)**

**Contributions**

John Dornheim - Functional/Non-functional Requirements, Formatting, Editing

Chris Durand - User Requirements/System Requirements, Overview

John Schlatter - Overview, Logos, Formatting

Shelby Mitchell - Glossary, Scheduling, Editing, Appendix

Alex Lopez - Systems Models, Functional/Non-functional Requirement

Jon Harrity - Systems Architecture, User Requirements/System Requirements