****

**Architecture + Design Overview**

**for**

**SEng Malware (SEM)**

**Version 0.2**

**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. Overview
   1. Purpose
   2. Roles and Responsibilities
2. User Interface
   1. Menu
   2. Deep Scan Page
   3. Shallow Scan Page
   4. Settings Page
   5. Log Page
   6. Help Page
3. System Architecture
   1. Box and Line Diagram
   2. Layered Architecture Pattern
4. Design Consideration
   1. Assumptions
   2. Constraints
   3. Risks
5. Current Revision Log of SEng Malware
6. **Overview**

1.1 Purpose

The purpose of this document is to provide an in depth description of SEng Malware, which is a malware scanner for Windows 10. SEng Malware was created for clients who might not be the most tech-inclined, but would like to keep their desktop safe from malicious files. This Architecture and Design overview will review the components of SEng Malware, its prototypes for the user interface, its design considerations, and its current revision log. This living document will continue to be updated until this project is completed.

1.2 Roles and Responsibilities

The following is a list of all SEng Malware team members and their roles within this project:

Shelby Mitchell — Project Manager, Scheduler, UI builder

Jon Harrity — Head of Configuration Management, UI builder

Alex Lopez — Head creator of the GUI

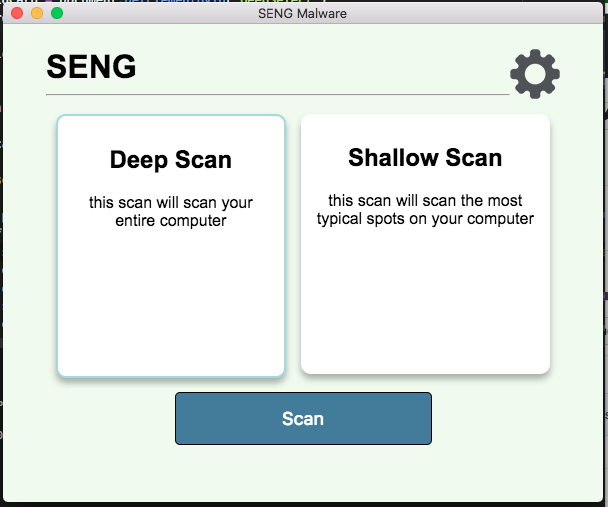
John Dornheim — Head of ClamAV functionality

Chris Durand — ClamAV functionality, UI builder

John Schlatter — UI builder, software tester

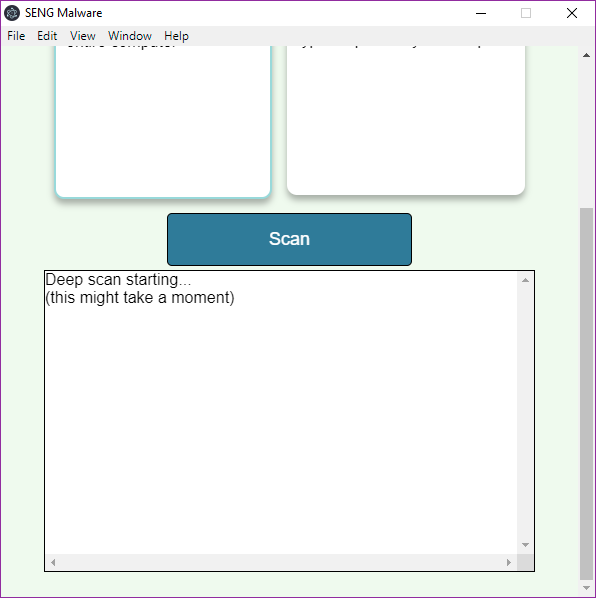
1. **User Interface**

2.1 Menu



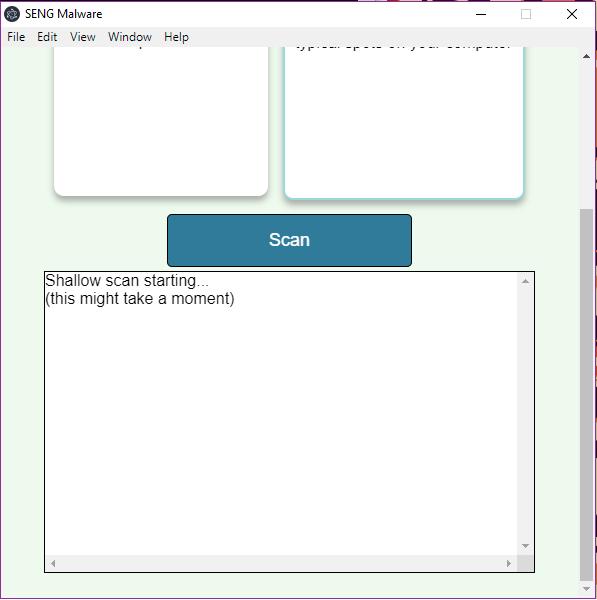
Menu Functions

1. Menu will show these options: Deep Scan, Shallow Scan, Settings, and the Scan button.
2. This application is defaulted to Shallow Scan, so even if the user only pressed the Scan button, the application would automatically run the Shallow Scan.

 2.2 Deep Scan Page

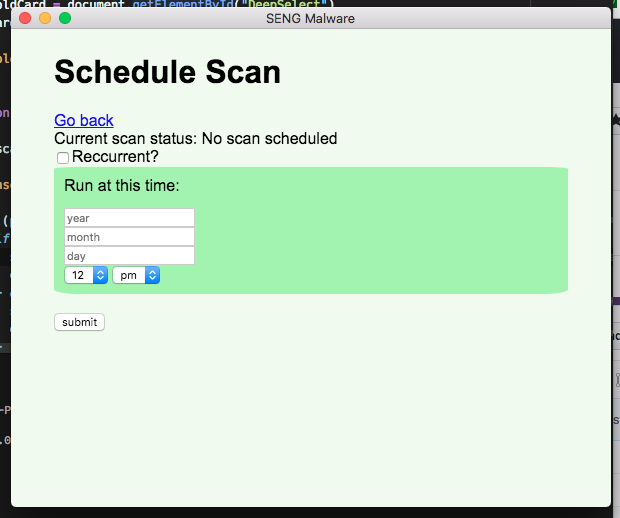
Deep Scan Functions

1. Once the Scan button is clicked on, the above screen is presented to the user
2. The Deep Scan will scan the C: directory of the current device
3. Text box will display what is currently being scanned. Files that are not malicious will be labeled “OK” while malware will be labeled “Found”
4. Empty files will also be tagged
5. Log will be generated from the results

2.3 Shallow Scan Page

Shallow Scan Functions

1. Shallow Scan button will initialize the shallow scan
2. The shallow scan will scan the Windows and Users Folders
3. Text box will display what is currently being scanned. Files that are not malicious will be lable “OK” while malware will be labeled “Found”

2.4 Scheduling Page

Settings Functions

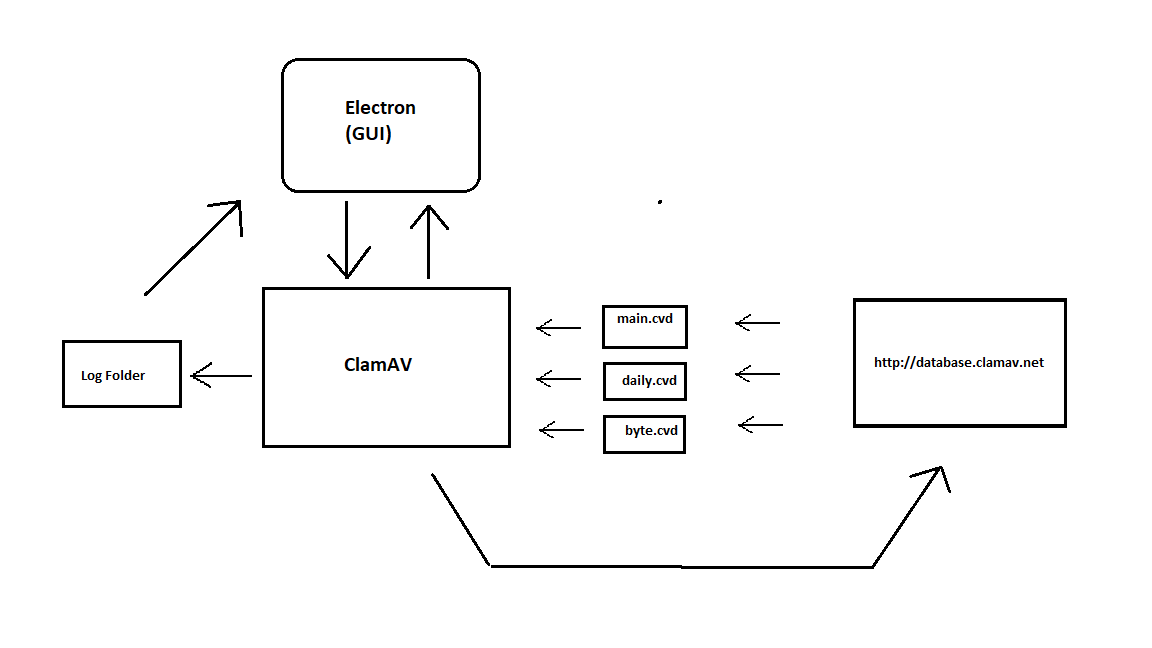
1. User will be able to schedule a scan for a future date
2. Simple Month/Day/Year/Time input with a schedule button
3. User will be able to manually update virus databases with button

2.5 Log Page

Log Page Functions

1. Displays a list of previous virus scans
   1. Initially will display the date of the scan and infected files were found.
   2. Clicking on a log will open the entire log of the scan. This will include the files scanned, number of files scanned, number of viruses checked for, number of infected files, and number of quarantined files.

**2. System Architecture**

2.1 Box and Line Diagram

2.1.1 Descriptions

ClamAV - anti-virus engine

Electron - GUI. Utilizes node.js

Log Folder - contains the virus logs generated by the ClamAV engine

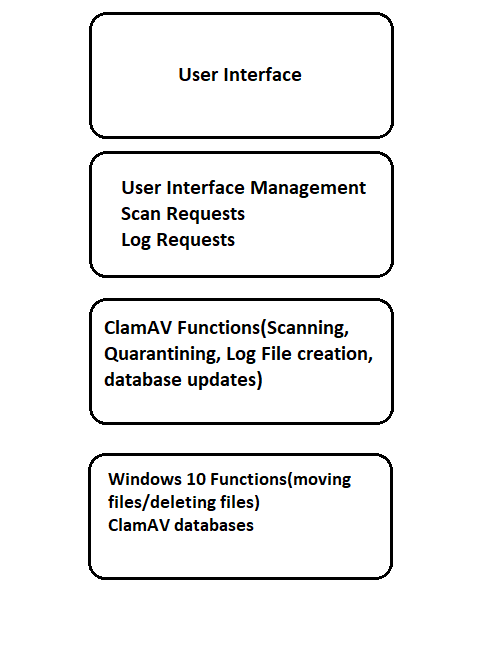
Main.cvd - main virus database

Daily.cvd - daily virus database

Byte.cvd - byte virus database

<http://database.clamav.net> - ClamAV virus database server

The box and line diagram describes the flow of information between components. This is in reference to how individual components talk to each other, as well as interactions with the database files and the internet.

2.2 Layered Architectural Pattern 

**4. Design Considerations**

4.1 Assumptions

So far for the project, there is no doubt about how many of the components will work. Development for the user interface, for example, is expected to go very smoothly. We have multiple team members with Node.JS experience and other front-end development experience. For the project, we decided to utilize the existing open-source ClamAV software. ClamAV will do a lot of heavy lifting for us, and we have begun to integrate ClamAV into our project. While we’ve been able to successfully install and use ClamAV to some extent, we’ve not yet been able to fully test the software. As a result, we are thus far making some assumptions about ClamAV.

Based off of our existing knowledge of ClamAV, we believe and assume the software will be able to successfully identify malware on a user’s computer. We have not been able to prove that ClamAV is able to do this. Furthermore, by malware, we refer solely to the malware already in ClamAV’s database. Research suggests that ClamAV’s database includes samples from all the major malware, but we do not have a listing for each individual sample in the ClamAV database. We therefore are assuming that ClamAV’s database is extensive and will cover the common malware that exists out in the wild.

As we continue to integrate ClamAV into our product and further test the scanning component, we will assume less about these two points as we gain more information.

4.2. Constraints

1. Time
   1. SEng Malware has a implementation time between March 5, 2019, and April 30, 2019. The projected time of finishing this project is April 20, 2019, with 10 days of slack. However at the latest, SEng Malware must be completed by April 30, 2019.
2. Version
   1. SEng Malware may only be run on Windows 10 computers. Any other OS, such as Mac or Linux, will **not** be compatible with our software.

4.3 Risks

1. Risk of not showing all files (hidden files). Can be mitigated by testing to make sure all files are returned and evaluated.
2. Limited Experience comparing file signatures. Can be mitigated by focusing on specific types of malware and by testing ClamAV functions for this
3. Risk of flagging necessary system-file and other false positives. Can be mitigated by creating exceptions for the files that we do not want to flag.
4. Risk of components not working together. Can be mitigated by sufficient unit and integration testing
5. Risk of ClamAV not being able to successfully identify malware. Mitigated by testing ClamAV scan functions on different malware samples
6. Risk of performance issues, especially if there are a lot of files to scan. Can be mitigated by making the application lightweight and including tests for large file systems in our test suite.