Software Requirements Specification

for

Malgone: An Anti-Malware Scanning Application

Version 1.0

Prepared by: Travis Shen, Tyrone Harmon, Yuliang Xue, Kunal Patel, Kelly Rose, and

Kestrel Bridges

Table of Contents

Requirements Specification - Anti-Malware Scanning Application

1. Introduction

In modern day computing, many users are faced with malware that affects their device. These endpoint devices (Linux, PC, Mac, Android, IOS) serve as a vulnerability and can corrupt files. Our goal is simple, to keep their devices malware-free by allowing for users to scan their device for malware and detain them if found and have the option of deleting them. The team members include the following; Kelly Rose, Kestrel Bridges, Travis Shen, Tyrone Harmon, Yuliang Xue and Kunal Patel. This team of six is dedicated to develop an efficient algorithm and interface to allow users to use this software without conflict. The Anti-Malware Scanning Application provides security as it scans, detects, and removes. The full details of the purpose, scope and Intended Audience is listed in this document.

1.1 Purpose

The purpose of this document is provide an overview on the requirement that will be deliver at the end of this term. We will deliver a desktop application which has the following functions: scan, detect, quarantine, and delete.

1.2 Scope

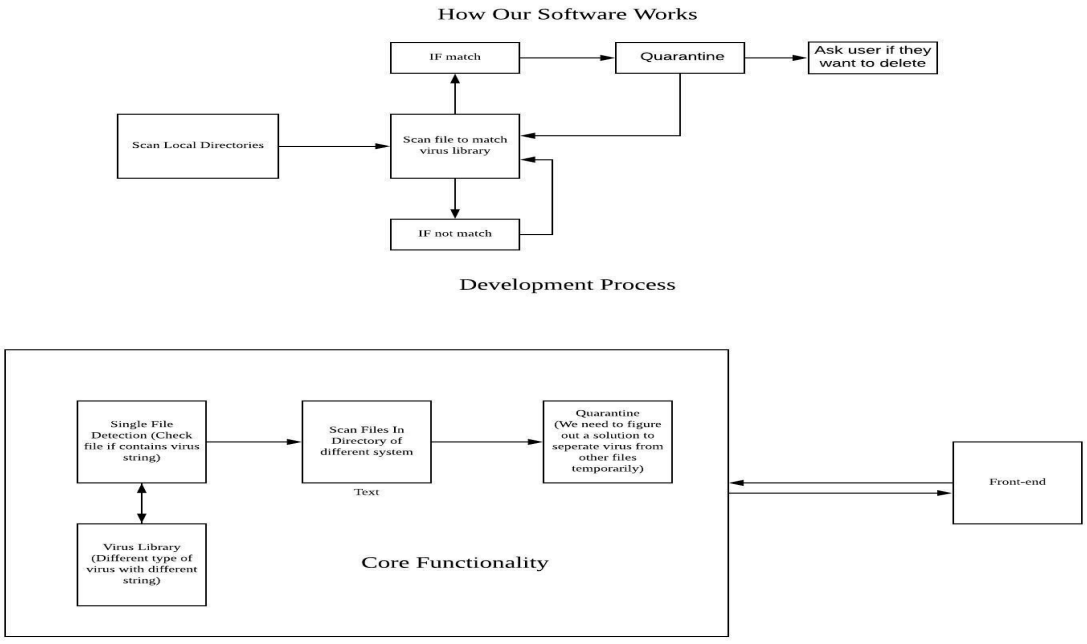
The purpose of this anti-malware software is to provide a reliable and user-friendly tool for ensuring security on the user's physical endpoint device. We will be creating a malware detection algorithm, utilizing open-source libraries of malware and virus codes against which the software will check suspected files. Our software will then give the user the option to quarantine or delete the affected file.

1.3 Intended Audience

This document is meant to serve as a resource for the team creating the anti-malware software, as well as the professor advising. End users are also encouraged to read through, beginning with product overview and paying special attention to User Requirements.

2. Product Overview

2.1 Product Perspective



2.2 Product Features

From the user end, the product resulting from this project will have the basic feature of scanning the designated areas, such as specific disks, directories and/or individual files for malware signatures. From the back end, this product will scan the files in the designated against the malware libraries for malware matches, if there are any matches. If a match or matches are found, it can perform one of the following actions:

a.) Quarantine the affected file

- It will move the affected file to a safe location that is managed by the antivirus software but not delete or clean the file.
- The main advantage of quarantining affected files compared to deleting them is that files that impact the normal features and functionality of the operating system or programs that users use are not accidentally removed.
- It also provides necessary time for the software to identify and categorize the types of malware present in the file(s).

b.) Delete the affected file

- Deleting will completely remove the affected file(s) from the computer.
- The main advantage is that it will get the malware out of system in a timely manner, given that it is true malware.
- The major downside is that if the files affected are needed for the operating system or the programs that are being used, then deleting those files might be fatal to the OS and/or the programs therein.

Sub-features that will be included in the final product consist of: scheduled scanning of designated areas, selection of areas to be checked, updating of the malware libraries and recording history of quarantining.

c.) The product will offer a function in which user can select scheduled scanning of files.

- It will be offered in monthly, weekly, and/or daily.
- The product will perform a monthly, weekly and/or daily scanning of the designated areas per user's choice.

d.) User will be able to choose the specific areas for scanning

- This feature will be combined with scheduled scanning feature so that users can choose the areas for scanning at the same time of choosing the frequency of scanning

e.) The product will update the malware libraries every 12 hours

- It will run the current libraries against the latest malware libraries to make sure that all the newly archived malware samples are recorded in the libraries

f.) The product will also record the history of quarantining and deletion of malware-infected files

- It will create a list of malware found from the beginning of the software's installation until present day

- The list will have the categories: time of detection, location of the malware, name, location, size and type of the files affected by the malware, and actions taken
- It will also provide a sorting feature that could list the items in ascending or descending order of one of the categories.

2.3 Product Environment

Operating System(s): Windows

3. User Requirements

The program we are producing gives the user the experience of security. Our program grants the user the ability to:

1	Scan	The software will look for viruses and malicious data contained in files
2	Quarantine	Any documents that are found during the search to have malicious variables inside will then be contained
3	Notify	Once a file with a virus or bug is found the user will be notified about which documents are now in question And ask what to do with it
4	Deletion	Documents that are now contained will be deleted

The user will also be given the option of how they wish to run this program. They can either choose to:

1	Select certain files of their choosing and run them against our program to find viruses or bugs
2	Run a full system scan that checks all files on the computer

The program will work on Windows OS. We hope in the future to provide functionality on other operating systems, such as Mac OS and Linux

4. System Requirements

4.1 Language

- For developing an anti-virus software, we need machine level access to files. We can use Java for this project, but it requires us to write in JNI (Java Native Interface) to access machine level code. C# has .NET framework which has such ability but with limited capability. We decided to write the program in Python to avoid complex infrastructure for our software, and also build up our scanning engine. For the virus library, we adapted libclamav (ClamAV signature database) with clamd (ClamAV multi-threaded daemon). We first set up configuration for Freshclam (Virus database update tool), and run Freshclam to update ClamAV virus database every time when the program starts. The process of updating virus library is before the loading of ClamAV daemon, so the daemon can be based on the newer version of virus library. For the front-end production, we will be using

an open source GUI software called Tkinter, a cross platform service written in Python and Cython

4.2 Detection

- Detection is the core functionality of our software. An anti-virus software should be capable of detecting malicious code in different files. It's a challenge for us to come up with detection algorithms and strategies to identify virus in the system. We need to have basic understanding of what make malware unique from other files, so we can detect and eliminate the virus from spreading.

4.3 Quarantining

- After detect the virus, we need to quarantine the virus and leave user a choice if they want to keep it or not. It's a challenge for us to determine a method to quarantine virus file. We need to move it into our secured and closed data storage to prevent it from spreading in system data storage.

4.4 Virus Library

- Anti-virus software should be able to detect different type of virus in the system. It requires us to build a virus library for the system to recognize various type of virus. We could use API or existed library for this purpose. In this case, the challenge is the implementation of certain API or library.

4.5 Monitoring Process

- Besides virus detection in data storage. A anti-virus software should also have the ability to monitor process and mark abnormal behavior of malicious process. This is additional feature we may work on after finishing the basic functionality of the

software such as detection and scanning. For this feature, we need to set up permission and monitoring end-points in certain folders such as system folder to decrease possibility of virus gaining access to important data.

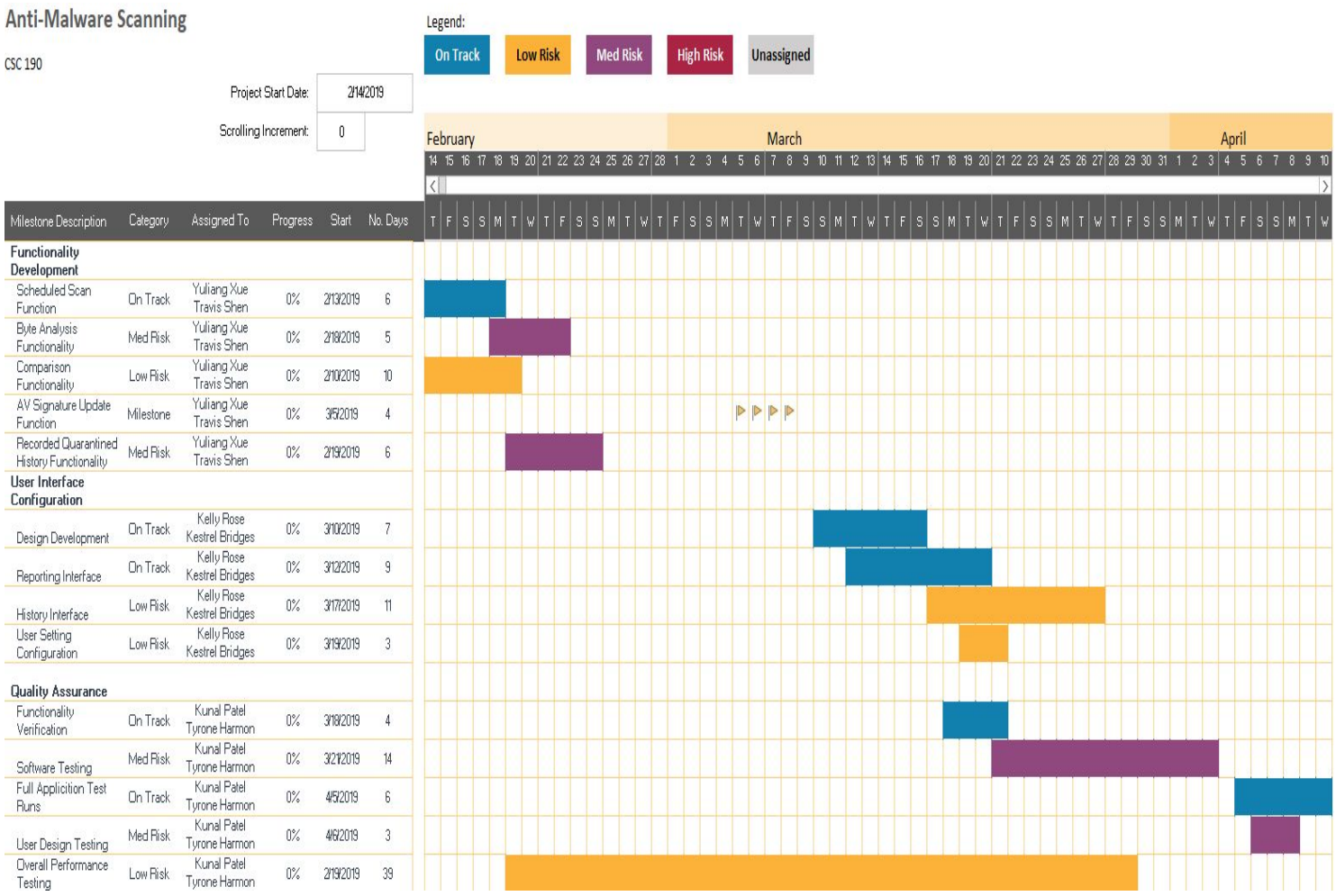
4.6 User experience

- User experience is critical and important for any software. Most users who tend to use anti-virus software are concerned about their security. A user-friendly interface became important as part of the user experience. It should provide users with quick access to different features and make them feel security by using our software.

5. Expected Development

5.1 Project Timeline:

Our predicted time for completion is nine (9) weeks, allowing for flexibility within the schedule. Our team is composed of six (6) members: Yuliang Xue and Travis Shen will be working on functionality development, Kelly Rose and Kestrel Bridges will be developing the user interface configuration, and Kunal Patel and Tyrone Harmon will be facilitating quality assurance and overall performance testing throughout the process. Attached below is a visual representation of our activity dependency chart.



Milestone Description	Category	Assigned To	Progress	Start	No. Days
Functionality Development					
Scheduled Scan Function	On Track	Yuliang Xue Travis Shen	0%	2/13/2019	6
Byte Analysis Functionality	Med Risk	Yuliang Xue Travis Shen	0%	2/18/2019	5
Comparison Functionality	Low Risk	Yuliang Xue Travis Shen	0%	2/10/2019	10
AV Signature Update Function	Milestone	Yuliang Xue Travis Shen	0%	3/5/2019	4
Recorded Quarantined History Functionality	Med Risk	Yuliang Xue Travis Shen	0%	2/19/2019	6
User Interface Configuration					
Design Development	On Track	Kelly Rose Kestrel Bridges	0%	3/10/2019	7
Reporting Interface	On Track	Kelly Rose Kestrel Bridges	0%	3/12/2019	9
History Interface	Low Risk	Kelly Rose Kestrel Bridges	0%	3/17/2019	11
User Setting Configuration	Low Risk	Kelly Rose Kestrel Bridges	0%	3/19/2019	3
Quality Assurance					
Functionality Verification	On Track	Kunal Patel Tyrone Harmon	0%	3/18/2019	4
Software Testing	Med Risk	Kunal Patel Tyrone Harmon	0%	3/21/2019	14
Full Applicition Test Runs	On Track	Kunal Patel Tyrone Harmon	0%	4/5/2019	6
User Design Testing	Med Risk	Kunal Patel Tyrone Harmon	0%	4/6/2019	3
Overall Performance Testing	Low Risk	Kunal Patel Tyrone Harmon	0%	2/19/2019	39

5.2 Timeline Constraints

This timeline provides some degree of flexibility, as the project is under several constraints. The project must be completed by the hard deadline of April 30th. All of the developers are also engaged in other independent projects, limiting the time they have to devote solely to this software. Additionally, parts of the project are dependent on the earlier stages, meaning some sections cannot be worked on in parallel. This staggers the development time. We have made provisions, however, to ensure downtime is minimized and the software develops on time.