Lightweight Antivirus Application (LAVA)

Software Project Plan

Table of Contents

1. **Introduction**

1.1 Purpose & Objectives…………………………………………………..……..3

1.2 Scope…………………………………………………………………….........3

1.3 Definitions, acronyms, and abbreviations……………………………………..4

1.4 References……………………………………………………………………..4

2. **Project Organization & Overview**

2.1 Team Info & Roles…………………………………………………………….5

2.2 Project Deliverables…………………………………………………………...5

2.3 Project Milestones……………………………………………………………..5

2.4 Branching Policies…………………………………………………………….6

2.5 Project Management & Control………………………………………………6

3. **Risk Analysis**

3.1 Project Risks…………………………………………………………………..7

3.2 Product Risks………………………………………………………………….7

3.3 Business Risks………………………………………………………………...7

4. **Hardware and Software Resource Requirements**

4.1 Hardware Requirements……………………………………………………….8

4.2 Software Requirements………………………………………………………..8

5. **Work Breakdown**

5.1 Tasks…………………………………………………………………………..9

6. **Project Schedule**

6.1 Schedule……………………………………………………………………...13

7. **Monitoring and Reporting Mechanisms**

7.1 Reporting………………………………………………………………….……14

7.2 Monitoring……………………………………………………………...….14

7.3 Testing………………………………………………………………..……....14

1. Introduction

1.1 Purpose & Objectives

The purpose of this document is to explain how our team is going to develop and complete this application. There will be breakdowns of exactly what we are doing as well as who is doing each task. There will also be diagrams that lay out the timelines for these tasks and the requirements to carry out the development. Some of the main objectives of LAVA are:

- Provide a lightweight anti malware solution for regular computer users and professionals
- Create an application implementation for Windows 10 by mid Q2 2020

1.2 Scope

LAVA is a supplemental malware scanning tool that allows users to scan locations within their Microsoft Windows-based computer for malicious software using the definitions within the ClamAV open source project. They can opt to scan the full system, frequently infected directories (hereafter to be referred to as a “Quick Scan”), or a set of user-specified directories or files. The stakeholders, basic users, will not have to worry about any of their data being held. LAVA is independent per computer it is installed on, and will only keep a database of viruses to check for. The limitations of LAVA is it will catch viruses that ClamAV will detect, as ClamAV is the core backend. This means as its current state, LAVA will not be as effective as other private options, but it will be Open-Source and free.

1.3 Definitions, acronyms, and abbreviations

Word, Acronym, or Abbreviation	**Definition**
Project Milestone	A milestone is a marker in a project that signifies a change or stage in development.
Git Repo	A **Git repo**sitory is the .git/ folder inside a project. This repository tracks all changes made to files in your project, building a history over time. Meaning, if you delete the .git/ folder, then you delete your project's history.
SDK	A software development kit is a collection of software development tools in one installable package
SRS	Software Requirements Specification
VM	Virtual Machine

1.4 References

LAVA github repo:

https://github.com/kornfeldm/LAVA

ClamAV github repo:

https://github.com/Cisco-Talos/clamav-devel

ClamAVdocumentation:

https://www.clamav.net/documents/

LAVA S.R.S Document:

https://github.com/kornfeldm/LAVA/blob/master/Requrements%20Spec%20Sheet.pdf

2. Project Organization & Overview

2.1 Team Info & Roles

Name	**GitHub Profile**	**E-Mail Address**
Thomas Mulvey	https://github.com/mulvenstein	tmulvey1@pride.hofstra.edu
Dylan Cenotto	https://github.com/dylancen	dcenotto1@pride.hofstra.edu
Mark Kornfeld	https://github.com/kornfeldm/	mkornfeld1@pride.hofstra.edu
James Dempski	https://github.com/JamesDempski	jdempski1@pride.hofstra.edu

2.2 Project Deliverables

The LAVA development team will produce a working anti-malware software application before mid Q2 2020. The final presentation will be given to Professor Jeffreys on or by April 30th, 2020. The master branch of the LAVA repo will produce a working *final* build by this date. The final compiled LAVA program will meet all software requirements listed in the SRS document.

2.3 Project Milestones

The first milestone for LAVA is the main GUI page and main scan page are completed. The estimated date of completion is 3/26/20 with the finalization of task 4.7.4.

The next milestone is reached when all 3 different levels of scanning are completed. This is marked after task 4.1.1 is completed, with an estimated date of 3/27/2020. The third milestone is when most of the backend functionality is completed. The estimated date for completion is 4/20/20 and the task is 4.9.1. The final merge (and the remaining backend code) will be the last milestone, 4.9.2, and will be completed by 4/24/20.

For more task completion and scheduled dates, please refer to sections 5 and 6.

2.4 Branching Policies

Branch policies are an important part of the Git workflow and empower the developers to: decouple work in-progress that is not being done in the master branch, limit who can contribute to specific branches, enforce who can create branches and the naming guidelines for the branches, automatically include the right reviewers for every code change, and enforce best practices with required code reviewers. For LAVA, there will be the master branch where all main builds and releases will come from. Below master, there will be the "development branch". Since LAVA is a short-term project, there will not be any other main branches. Any feature branches will be called "feature/id-description" and will be forked off of develop and merged back into develop, when ready. For example, a feature branch may be called "feature/4.1.1-merging3scans". Any merges into the develop branch will require code approval from one other developer on the team. Bug-fix branches will follow suit, with the prefix "bug/" for the branch name. Merges from develop to master will require approval from two other developers.

2.5 Project Management & Control

Team Meetings: The team will meet once a week in person at the Axinn Library at 10am every Saturday (until the end of the development cycle). Further updates will be done on Discord or GroupMe. The main goal of each meeting is to see where the team is at and the agenda for the upcoming week. The team will decide if any “blockers” will require the reallocation of resources to other tasks at these weekly meetings.

3. Risk Analysis

3.1 Project Risks

The main areas of risk in descending order are implementing ClamAV in the backend, creating the GUI, and the API that translates the GUI events into backend ClamAV commands. The libclamav library is very old and there is little existing documentation for it. Once libclamav is working and implemented, the backend development can really take off. The development team has planned for a high ramp up time for this portion. The ClamAV library has the highest risk of altering the schedule of tasks. To mitigate the risk here we planned for a large ramp-up time for learning the libraries.

3.2 Product Risks

The main product risk is that ClamAV performs worse when detecting malware when compared against other antivirus solutions. ClamAV is the best Open Sourced (free licensed) virus scanner available, which is imperative to LAVA’s development. LAVA being under the GNU General Public License 3.0.

3.3 Business Risks

Although LAVA will be a free product to use (and contribute to), there are still business risks. There are other similar solutions of antimalware products that use ClamAV for scanning such as ClamWin, ClamTK, ClamAV-GUI, and NodeClam. Some of these solutions are inactive, but may still be actively used by others. Additionally, there is the liability associated with a product that either fails to detect a malicious file or returns false positives. Despite how careful development may be, a perfect product free of such errors is unrealistic.

4. Hardware and Software Resource Requirements

Hardware and software resource requirements for LAVA are the hardware and support software required to carry out the development of the application. Any failures to adhere to the requirements below may affect the estimated schedule.

4.1 Hardware Requirements

Developers will need a Windows platform to test on. ClamAV and nuklear (the frontend framework) can run on linux, but as of right now LAVA will only operate on Windows 10. Development in Linux is possible but not recommended nor outlined in this section. If you wish to pursue linux development for LAVA, please start off by following ClamAVs linux development tutorial https://www.clamav.net/documents/clamav-development.

4.2 Software Requirements

For the development of LAVA, Visual Studio 2017 (VS 2017) will be required. Additional requirements include VS 2017 build tools for C++, as well as Windows 10.0.102 SDK. This is required for any changes that may need to be done towards ClamAV.

5. Work Breakdown

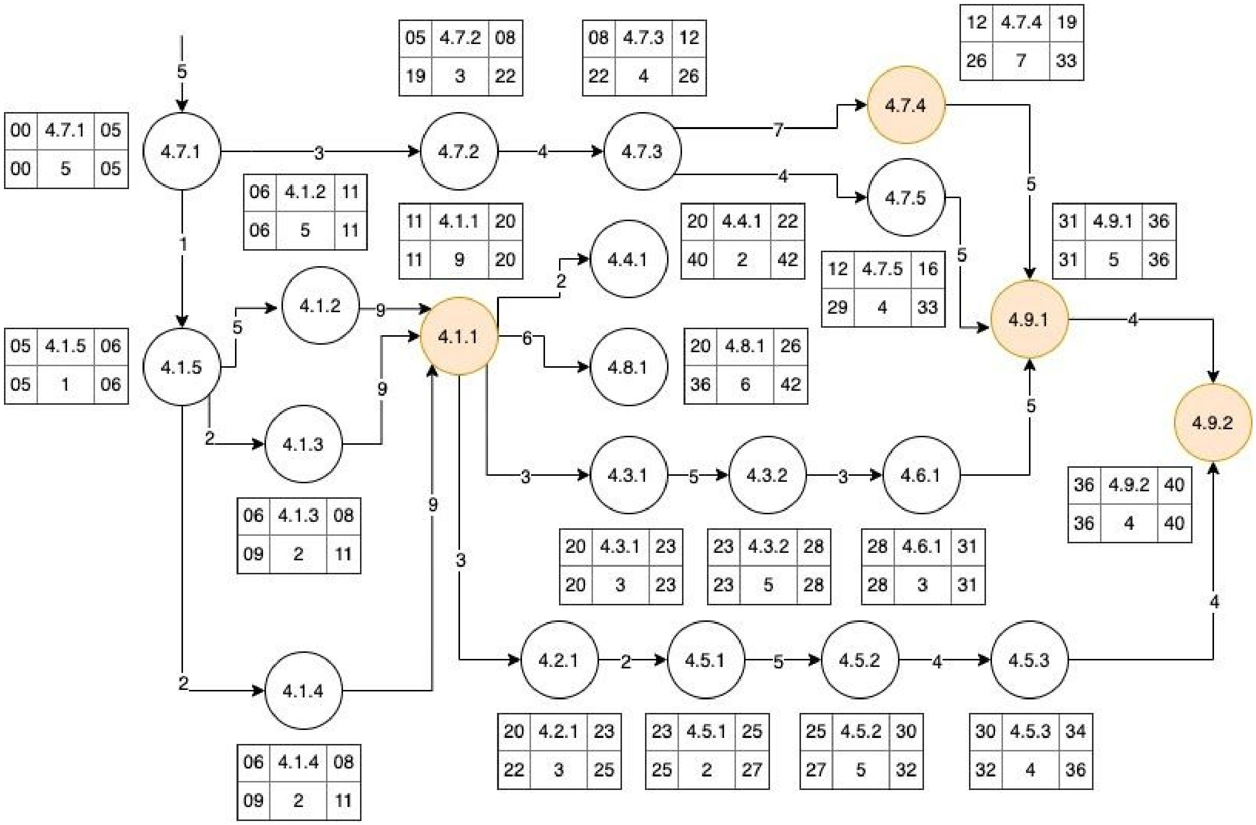
5.1 Tasks

Priority	**Task Name**	**Description**	**Effort (Days)**	**Assignee(s)**
Medium	4.1.1 Different Levels of Scanning	LAVA will have 3 different levels of scans. Quick, Full, and Advanced (also referred to as "custom").	**9**	Mark Tom James Dylan
Medium	4.1.2 Quick Scan Capability	Quick Scan will target critical and previously infected directories.	**5**	James
High	4.1.3 Complete Scan Capability	The Complete Scan will recurse through every file and directory on the machine, including attached storage	**2**	Mark
Medium	4.1.4 Advanced Scan Capability	Scan by specific file, directory or directory sets. Selection will be done through a file explorer	**2**	Dylan
High	4.1.5 Maintain System Integrity	As such, LAVA will not delete or quarantine files that	**1**	Mark

		are essential to the system.		
Medium	4.2.1 Scheduled Scans	LAVA will use Windows Scheduler to run scheduled scans and this option can be configured to occur daily, weekly, or monthly.	**3**	Tom
High	4.3.1 Quarantine	When the scanner finds the malware files, it will move them to a Quarantine Directory.	**3**	Dylan
Medium	4.3.2 Maintaining of an Antibody File	As more scans happen, this Antibody File will populate and these directories will be marked for future, optimized scans such as the Quick Scan.	**5**	James Mark
High	4.4.1 Updates of Virus Bytecodes	LAVA will update its antivirus definition database every time a scan is launched. If there is no valid internet connection, this step will be skipped.	**2**	James
Low	4.5.1 Boot Startup Opt-Out	LAVA will turn on at system boot up and consistently run in the background unless the user	**2**	Mark

		disables that function. This task has been removed.		
Low	4.5.2 Real-Time protection	LAVA will check newly downloaded files for existing virus bytecodes and quarantine them, if necessary. It will alert the user as well. This task has been removed.	**5**	Dylan Tom
Low	4.5.3 External Drive Protection	LAVA will check connected USB drives for any malicious software and quarantine the infected files or eject the drive completely.	**4**	James
Low	4.6.1 Continued Systemed Support	LAVA will require a support section where users can search for help for bugs or usability.	**3**	Mark
High	4.7.1 Install ClamAV	Install ClamAV library and get initial run	**5**	Dylan Mark Tom James
High	4.7.2 Install front end framework	Install the front end framework and get an initial run.	**3**	Tom
Med	4.7.3 Main Page GUI	Create Main Page GUI	**4**	Tom
Med	4.7.4 Subsequent GUI Scan Pages	Create Subsequent Scan page GUI	**7**	Tom James

Low	4.7.5	Create History GUI	**4**	Dylan
Med	4.8.1	LAVA will keep a history of every scan and it’s statistics	**6**	Tom Dylan
High	4.9.1	Merge front end GUI and Backend (discluding Real-Time and USB)	**5**	Dylan Mark Tom James
Med	4.9.2	Final Merge of Front end and all back end	**4**	Dylan Mark Tom James



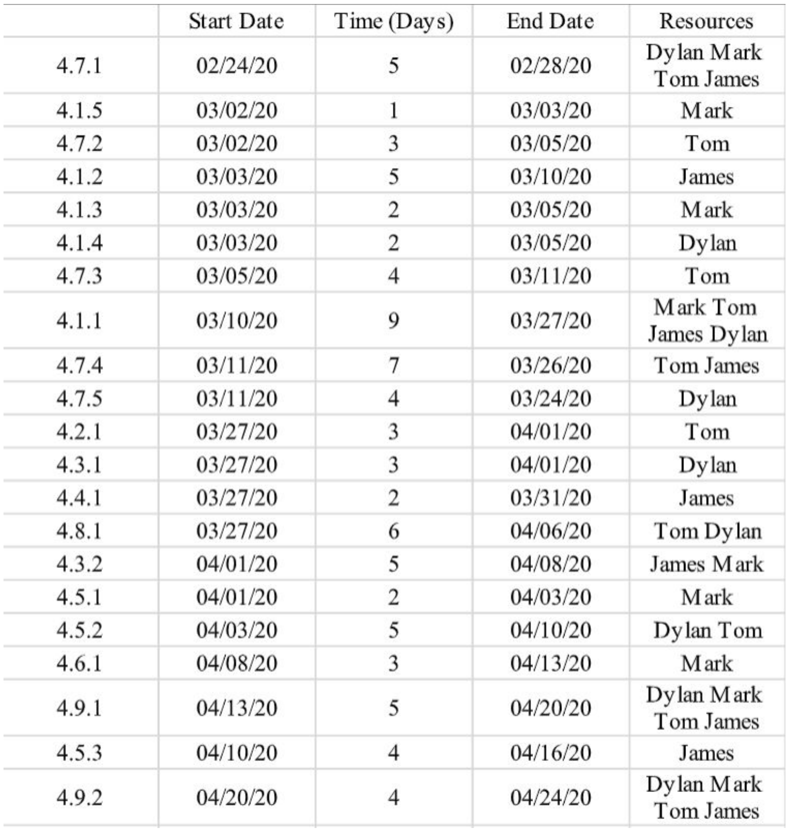
12

6. Project Schedule

6.1 Schedule

The below diagram represents the schedule the development of LAVA will follow. Day 0 is to refer to Monday, February 24th, 2020 with each successive day representing a business day excluding Monday, March 16th through Thursday, March 19th.

The following table represents the schedule with the exact dates and assigned resources:



	Start Date	Time (Days)	End Date	Resources
4.7.1	02/24/20	5	02/28/20	Dylan Mark Tom James
4.1.5	03/02/20	1	03/03/20	Mark
4.7.2	03/02/20	3	03/05/20	Tom
4.1.2	03/03/20	5	03/10/20	James
4.1.3	03/03/20	2	03/05/20	Mark
4.1.4	03/03/20	2	03/05/20	Dylan
4.7.3	03/05/20	4	03/11/20	Tom
4.1.1	03/10/20	9	03/27/20	Mark Tom James Dylan
4.7.4	03/11/20	7	03/26/20	Tom James
4.7.5	03/11/20	4	03/24/20	Dylan
4.2.1	03/27/20	3	04/01/20	Tom
4.3.1	03/27/20	3	04/01/20	Dylan
4.4.1	03/27/20	2	03/31/20	James
4.8.1	03/27/20	6	04/06/20	Tom Dylan
4.3.2	04/01/20	5	04/08/20	James Mark
4.5.1	04/01/20	2	04/03/20	Mark
4.5.2	04/03/20	5	04/10/20	Dylan Tom
4.6.1	04/08/20	3	04/13/20	Mark
4.9.1	04/13/20	5	04/20/20	Dylan Mark Tom James
4.5.3	04/10/20	4	04/16/20	James
4.9.2	04/20/20	4	04/24/20	Dylan Mark Tom James

7. Monitoring and Reporting Mechanisms

7.1 Reporting

The first status report will be due March 3, 2020 and shall be presented in class during a 10 minute presentation period. Further reports will be personally given to Prof. Jeffreys.

7.2 Monitoring

- The project will be monitored by Prof. Jeffreys.
- The project will be built every Friday at 12pm.
- Check-ins to the repository will be done every 48 hours.

7.3 Testing

LAVA will be tested primarily using different virtual machines (hereafter referred to as a "VM"). Each VM will simulate different builds of our target operating system, Microsoft Windows 10. The VMs will also be set to simulate different system states such as storage devices at maximum capacity or system resources utilized at 100%. Additionally, LAVA will be deliberately exposed to malware samples as well as legitimate files on the VM systems to test detection ratio and ensure false-positives are kept to a minimum. In the final stages of testing, LAVA will be installed on physical machines to ensure it behaves as expected. Notably, malware samples will not be tested on physical machines as we unfortunately lack the resources to be able to test such scenarios.

As mentioned earlier in the branching policies, bugs and features branches will be approved by other members before being merged. Once test cases are developed we will publish the code coverage statistics to the GitHub Repo.