Vulnerabilities Under Learned Network (V.U.L.N)

Team 4: Hibah, Diego, Kenneth, Justin



Project Management Document

Anti-Malware Scanning Application

Version 1.0

**Table of Contents**

**1. Introduction**

1.1 Purpose…………………………………………………………………………………..pg.3

1.2 Product Overview…………………………………………………………………...…..pg.3

1.3 Dependencies………………………………………………………………..…………..pg.3

1.4 Definitions…………………………………………...…………………………………..pg.4

1.5 References…………………………...…………………………………………………..pg.5

**2. Product Organization**

2.1 Team Info………..……..………………………………………………………………..pg.5

2.2 Project Deliverables..……………………………………………………………………pg.6

2.3 Project Milestones……………...…………………………………………………..……pg.6

2.4 Project Management & Control………………………...……………………………….pg.6

**3. Risk Analysis**

3.1 Project Risks………………………………………..…………………………………pg.6-7

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

3.3 Business Risks……….………………………………………………………………..pg.7-8

**4. Hardware & Software Resource Requirements**

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

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

**5. Work Breakdown**

5.1 Tasks……………...………………………………………………………………….pg.8-11

5.2 Configuration Management……...…….………………………………………………pg.12

**6. Project Schedule**

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

6.2 Project Constraints.....………………….…………………………………………...pg.13-14

**7. Monitoring & Reporting Mechanisms**

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

7.2 Monitoring……….…………………………………………………………………pg.14-15

7.3 Quality Assurance Testing…………..……..…………………………………………..pg.15

**Introduction**

**1.1 Purpose**

The overall purpose of this project plan is to document, elaborate and effectively breakdown our team’s objectives in regard to our V.U.L.N application and describe who was involved/responsible for particular tasks in detail. Diagrams will be available to accompany our descriptions as demonstrations.

* *V.U.L.N will run on the Linux operating system.*
* \*\*\* *V.U.L.N’s prime intention is to distribute a user-friendly malware detection application which provides the user with plenty of leeway over what they choose to do when malware is detected*. \*\*\*

**1.2 Product Overview**

V.U.L.N utilizes the ClamAV engine’s utilities to commence different types of scanning protocols issued by the user to detect malware. Such protocols include a Full System Scan, a “Quick” Scan which implies scanning certain directories which affect processes such as performances (files which are usually targeted); or to scan a set of files/directories specified by the user. Information of the malware detected will be stored in a database, which leads to the decision of whether the user prefers to quarantine the file(s) or purge them. The information given will be applied to the improvement of the program’s functionality.

**1.3 Dependencies**

As mentioned prior, the ClamAV Engine will be the backend to develop our “Scanner”. Its utilities such as its libraries, definitions as well as an API will also be integrated/developed. The following will be integrated via a Linux Virtual Machine. All changes/versions of the program will be pushed to a git repository (github).

**1.4 Definitions**

| **Terms/Acronyms** | **Definitions** |
| --- | --- |
| Git Repository | A **Git repository** is a central storage location for managing and tracking changes in files and directories. |
| V.U.L.N | Vulnerabilities Under Learned Network |
| Virtual Machine | The virtualization or emulation of a computer system. **Virtual machine**s are based on computer architectures and provide the functionality of a physical computer. |
| ClamAV Engine | **ClamAV** is a powerful antivirus software that can scan your email and your server for malicious files. |
| Malware | Malicious Software |
| SRS | Software Requirements Specification |
| GUI | The **graphical user interface**, or GUI, is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicators |

**1.5 References**

ClamAV Github Repository: <https://github.com/Cisco-Talos/clamav>

ClamAV Documents: <https://docs.clamav.net>

V.U.L.N Github Repository:

<https://github.com/Kenny4103/V.U.L.N>

Software Requirements Document:

<https://docs.google.com/document/d/1LeAPi1TWdfp8kiimud1kIo7MRg-QTOYy8NIRYhrUoj8/edit?usp=drivesdk>

**Product Organization**

**2.1 Team Info**

| **Name** | **GitHub Profile** | **EMail Address** |
| --- | --- | --- |
| Hibah Arshad | <https://github.com/HibahArshad> | harshad2@pride.hofstra.edu |
| Kenneth | <https://github.com/Kenny4103> | kgavino1@pride.hofstra.edu |
| Justin | <https://github.com/JustinHennis1> | jhennis1@pride.hofstra.edu |
| Diego | <https://github.com/DoryuzuUchiha> | despada1@pride.hofstra.edu |

**2.2 Project Deliverables**

V.U.L.N's team of developers will create a functional anti-malware program. Professor Jeffreys will receive the presentation for our software by November 30th, 2023. This carries the program's executable files, source code, guidelines, and any supporting materials for the program. All software requirements stated in the SRS document will be satisfied by the final implementation of the application.

**2.3 Project Milestones**

The milestones for this project would start with the creation of the base scanner that we will use ClamAV for. When this is done it will open up the project for work to be done. The next major milestone is the finishing of the database communicating manually with the scanner. This will be the time in which the first demo of the product will be done. Then comes the finishing of the automated scan and database updates. This will signify the product almost being completely done and a good amount of risk being taken away from the project.

**2.4 Project Management & Control**

From now till the conclusion of the development cycle, the team is scheduled to meet through Zoom every week. The team will meet in person, if needed, at C. V. Starr Hall on Mondays at 1:00 pm. For everyday communication, GroupMe is used. Each meeting's primary objective is to assess the team's current situation and plan the coming week's schedule. The team will also determine whether any setbacks call for shifting efforts towards different tasks in our activity plan.

**Risk Analysis**

**3.1 Project Risks**

The primary areas of concern include the implementation of ClamAV in the backend, the creation of the GUI, and the development of the API responsible for translating GUI events into backend ClamAV commands. Notably, the libclamav library poses a notable challenge due to its age and limited existing documentation. The successful integration of libclamav is pivotal, as it serves as a catalyst for advancing backend development. Recognizing the complexity and potential setbacks associated with the ClamAV library, the development team has strategically allocated a substantial ramp-up time to familiarize themselves with the libraries as well as establishing a github for this project that all members of V.U.L.N can access, aiming to mitigate potential risks and ensure a smoother development trajectory. Additionally, a learning period will be allocated for the configuration of a backend database that will work with the linux environment that the V.U.L.N team creates.

**3.2 Product Risks**

The main risks associated with this application is the fact that ClamAv is open sourced and any uncaught V.U.L.Nerabilities belonging to this source code could limit the effectiveness of V.U.L.N’s application. The fact that this engine and its libraries are old could impact its effectiveness in responding to newer threats, knowing that the V.U.L.N development team will add a way to create new scans in an attempt to modernize our application.

**3.3 Business Risks**

Various antimalware products, including ClamWin, ClamTK, ClamAV-GUI, and NodeClam, employ ClamAV for scanning purposes. While some of these solutions may no longer be actively maintained, they could still be in use by certain individuals or organizations. Furthermore, there exists a liability concern associated with a product that either fails to detect a malicious file or generates false positives. Even with meticulous development practices, achieving a flawless product devoid of such errors is an impractical expectation. It is important to learn why previous implementations have failed so that mistakes can be avoided when implementing V.U.L.N.

**Hardware & Software Resource Requirements**

Hardware and software resource requirements for V.U.L.N 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**

V.U.L.N developers will need to test their software on Linux OS Ubuntu Version 22. Although Windows supports ClamAV, the V.U.L.N team believes Linux provides a simpler filing system.

**4.2 Software Requirements**

Software requirements include considerations for the internal structure, modularity, or architectural decisions.

For V.U.L.N to be developed, Visual Studio Code is required along with any of the extensions for C/C++ compiling and debugging. Flutter and Dart SDK’s will also be required for frontend UI development.

**Work Breakdown**

**5.1 Tasks**

| **Task #** | **Priority** | **Task Name** | **Task Description** | **Requirements** | **Time(Days)** | **Appointee** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Low | Create Base UI | A simplistic and easy to navigate UI is designed and integrated into the program. | NONE | 3 | Hibah |
| 2 | High | Selection System  -for disk directories and files | V.U.L.N will incorporate a system capable of having the user choose specific files for scanning | NONE | 5 | Kenneth |
| 3 | High | Create Scanner  (requires integration of ClamAV) | Will Scan the Computer for viruses and notify that the scanning is done. | 2 | 7 | Justin  Hibah  Kenneth  Diego |
| 4 | Medium | Create a Database | Creates a base database to store data. | NONE | 2 | Justin |
| 5 | Medium | Quarantine Infected Files | This will hold a file that has been deemed infected and wait for user instruction stopping interaction with the file. | 3 | 5 | Justin  Kenneth |
| 6 | Medium | Purge Infected File(s) | This will delete the quarantine file and remove it from the system | 3 | 5 | Kenneth |
| 7 | Low | Save info to Database | Will take the data/information on which files that were quarantined and save it for future use. | 4,5,6 | 4 | Justin |
| 8 | Low | Create a Finished UI | The finalized UI that was designed prior | 1 | 3 | Hibah |
| 9 | High | Create Update Structure to Database | V.U.L.N will integrated process where the Database will update when given information of common malware | 7 | 2 | Diego  Justin |
| 10 | Medium | Setup Automatic Scanning | This process provides a functionality of having scheduled scanning automatically given certain circumstances | 9 | 2 | Diego  Kenneth |
| 11 | Medium | Setup Automatic Database Update | Consists of a functionality where the process prior is conducted automatically | 10 | 2 | Justin |
| 12 | Medium | Setup Scheduling | How often will the software scan the device given directives from the user? | 11 | 2 | Diego |

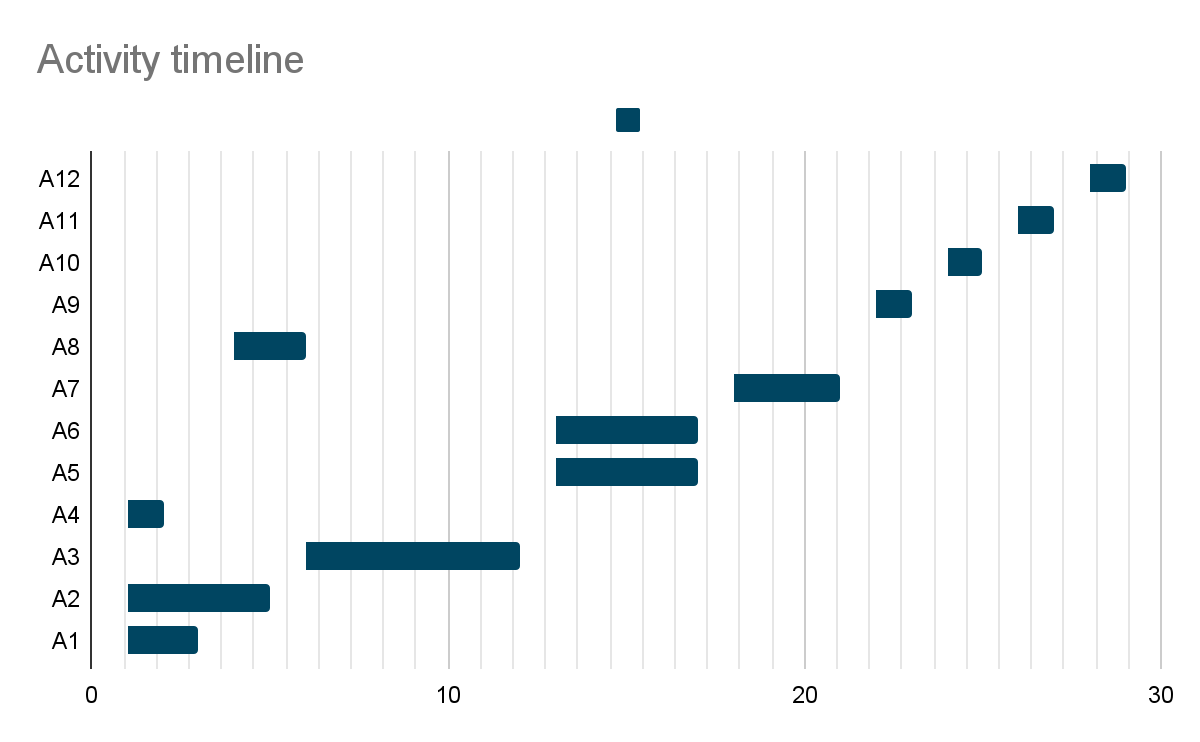
****

**5.2 Configuration Management**

V.U.L.N’s github is up and running along withdocuments done up till now. GitHub offers V.U.L.N the ability to share, analyze, and contribute to the software using this centralized repository. In order to separate modifications, bug fixes, and new features from the main source, GitHub enables organized branching methods. As a result of this, the development team for V.U.L.N can collaborate on several projects at once without sacrificing stability.

**Project Schedule**

**6.1 Schedule**

****

A1- A12 represent the 12 activities in our project plan and the bars on the graph represent when the activity will start and when it will end showing the parallels within the Project Plan.

**6.2 Project Constraints**

Testing Challenges:

The V.U.L.N. project expects to run into constraints while carrying out testing. Considering the goal of our software is to successfully scan for malware, we must be ready to handle any difficulties that may arise, especially when it comes to false positives. The integration of the ClamAV engine, a new tool for the V.U.L.N. team has a learning curve that contributes to this problem. The complexity of this integration necessitates noteworthy consideration.

Time Constraints:

Furthermore, it's important to recognize that undergoing this complicated project inevitably faces time restrictions.

Coding Proficiency:

The C++ programming language was specifically chosen by V.U.L.N. to execute the project, a choice prompted by the team's combined experience. The bulk of the team members are highly skilled in C++, which allows them to operate effectively in the selected framework. It's important to note that one member is not as efficient in C++.

**Monitoring & Reporting Mechanisms**

**7.1 Reporting**

Presentations of the project's status updates are planned for October 31 and November 7, 2023. Documentation will act as an in-depth repository for details on V.U.L.N's development, supplementary to these presentations. The documentation will comprise an overview of the project, important dates, task updates, schedule changes, information on any delays observed, and their related options. In accordance with our project timetable, we will carefully monitor scan speed and update speed to assess the project's overall efficacy.

**7.2 Monitoring**

V.U.L.N will strictly follow the project's timeline and task distribution described in sections 5& 6 as we consistently track the development of our software. Tasks with a higher priority or those where team members are experiencing delays will receive additional emphasis to guarantee the completion of the project. Team members will be able to inform the group on the progress of the tasks or problems they are responsible for during our weekly project-related meetings. These meetings will allow us to correctly assess performance and quickly highlight areas in need of more assistance, permitting task modifications as required. Monitoring will also include:

* Scan Speed: A built-in benchmark tool for evaluating scan speed is included in ClamAV.
* Malware Detection: ClamAV engine will detect threats against known malware.
* Updates: Automatic updates from ClamAV will be used

**7.3 Quality Assurance Testing**

The application’s system will be tested regularly as the project continues. The EICAR file will be utilized to verify the functionality of the interfaces. Cybersecurity and antivirus organizations utilize the EICAR (European Institute for Computer Antivirus Research) file as an accepted test file to evaluate the performance of antivirus programs, notably engines like ClamAV. EICAR is a secure and harmless file made specifically to be tested and not a virus. To show that it can identify harmful data when testing, our antivirus tool should scan a file and identify the EICAR test file, raising a warning or taking the necessary steps (quarantining or destroying the file). The handling of different file types, sizes, and locations will also be evaluated for scanning functions.