Software Requirements Specification

for

“Windows based Malware Prediction System using Deep Learning Techniques”

Prepared by

AATISH KAYYATH	**1MS15CS002**
ABISHEK PADAKI	**1MS15CS005**
ARAVIND P ANIL	**1MS15CS024**
DEVIKA ANIL	**1MS15CS040**

Table of Contents

1. Introduction

1.1 Purpose

Malware refers to malicious software perpetrators dispatch to infect individual computers or an entire organization's network. It exploits target system vulnerabilities, such as a bug in legitimate software (e.g., a browser or web application plugin) that can be hijacked.
A malware infiltration can be disastrous—consequences include data theft, extortion or the crippling of network systems.
In the past 5 years (since 2013), there have been more new kinds of malware created than the ten years before that combined. The need for detection of malware attacks is growing. And of

particular concern are the Windows operating systems, which run on over 75% of desktops today.

There are two main types of defence against malware - malware detection and malware prevention. We focus on the former and remove the problem before it even arises. There are over a billion potential systems in the world that potentially be affected by malware. Designing a technique to prevent attacks from all different types of malware can be extremely difficult as compared to detecting the causes of malware and predicting if a machine will be hit with malware**.**

1.2 Intended Audience and Reading Suggestions

Operating systems are a variety of system software that manage computer hardware and software resources, acting as the intermediary between programs and the hardware. In January 2019, the market share of the Windows operating system like Windows 10, Windows 7, Windows Vista, Windows XP range stood at 75.47 percent. Windows OS's for business uses and servers like Windows Server 2012, Server 2019 etc. also has a major share in the market. Because of it's huge presence in the industry Malware Detection tools are intended for both commercial and industrial users. The tools will also be analysed by security engineers and other people related to this domain.

1.3 Product Scope

Over the last decades, there were lots of studies made on malware and their countermeasures. The most recent reports emphasize that the invention of malicious software is rapidly increasing. Moreover, the intensive use of networks and Internet increases the ability of the spreading and the effectiveness of this kind of software. On the other hand, researchers and manufacturers making great efforts to produce anti-malware systems with effective detection methods for better protection on computers.

Malware is software that is aimed at intentionally causing a system to behave in a way that it should not. Its main objective is to disrupt the system's normal functioning or cause damage to the system itself and its components or both. Malware comes in many different forms, but irrespective of the type, their objective is one - damage to a system. Malware attack is a very large domain of cybersecurity attacks. Cryptanalysts across the world has been in a long drawn out battle which has intensified in the past decade with malware. With increase in the technological capabilities of computers, there is also a distinct sharp increase in the capabilities of what malware can do and more importantly, how a malware can prevent from being detected. This is what our project is aimed at - Malware detection.

1.4 References

[1] Baezner, Marie, and Patrice Robin. *Stuxnet*. No. 4. ETH Zurich, 2017.

[2] You, Ilsun, and Kangbin Yim. "Malware obfuscation techniques: A brief survey." In *2010 International conference on broadband, wireless computing, communication and applications*, pp. 297-300. IEEE, 2010.
[3] Bethencourt, John, Dawn Xiaodong Song and Brent Waters."Analysis-Resistant Malware". In: NDSS (2008)
[4] Ekta Gandotra, Divya Bansal, Sanjeev Sofat (2016) "Zero-Day Malware Detection". In: 2016 Sixth International Symposium on Embedded Computing and System Design (ISED).
[5] Miramirkhani, Najmeh, Mahathi Priya Appini, Nick Nikiforakis, and Michalis Polychronakis. "Spotless sandboxes: Evading malware analysis systems using wear-and-tear artifacts." In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 1009-1024. IEEE, 2017.
[6] Rastogi, Vaibhav, Yan Chen, and Xuxian Jiang. "Catch me if you can: Evaluating android anti-malware against transformation attacks." *IEEE Transactions on Information Forensics and Security* 9, no. 1 (2014): 99-108.
[7] Christodorescu, Mihai, Somesh Jha, Sanjit A. Seshia, Dawn Song, and Randal E. Bryant. "Semantics-aware malware detection." In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pp. 32-46. IEEE, 2005.
[8] Burguera, Iker, Urko Zurutuza, and Simin Nadjm-Tehrani. "Crowdroid: behavior-based malware detection system for android." In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pp. 15-26. ACM, 2011.
[9] Borbely, Rebecca Schuller. "On normalized compression distance and large malware." *Journal of Computer Virology and Hacking Techniques* 12, no. 4 (2016): 235-242.
[10] Burnaev, Evgeny, and Dmitry Smolyakov. "One-class SVM with privileged information and its application to malware detection." In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 273-280. IEEE, 2016.
[11] Bhattacharya, Sukriti, Héctor D. Menéndez, Earl Barr, and David Clark. "Itect: Scalable information theoretic similarity for malware detection." *arXiv preprint arXiv:1609.02404* (2016).
[12] Jhonattan J Barriga, Sang Guun Yoo. "Malware Detection and Evasion with Machine Learning Techniques: A Survey" International Journal of Applied Research 2017
[13] Hardy, William, Lingwei Chen, Shifu Hou, Yanfang Ye, and Xin Li. "DL4MD: A deep learning framework for intelligent malware detection." In *Proceedings of the International Conference on Data Mining (DMIN)*, p. 61. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016.
[14] Kolosnjaji, Bojan, Apostolis Zarras, George Webster, and Claudia Eckert. "Deep learning for classification of malware system call sequences." In *Australasian Joint Conference on Artificial Intelligence*, pp. 137-149. Springer, Cham, 2016.
[15] Bagga, Naman. "Measuring the Effectiveness of Generic Malware Models." (2017).

2. Overall Description

2.1 Product Perspective

The product is a model that's developed to accurately predict the probability that an operating system will be hit by a malware. The subsystems of the product are mainly 2 components. These subsystems are explained in the following subsections.

2.1.1 The Operating System

We work on an operating systems dataset that has over 7 million recorded operating systems and their various features. We create a model that tries to predict which of those operating systems will be hit with any type of malware.

2.1.2 The Prediction Model

- Light GBM is a very fitting model to this dataset due to the size of the data and the effectiveness and memory efficiency of this particular method of Gradient boosting. It grows in a tree wise structure. But it differs from other tree-based algorithms due to its horizontal growth structure.
- Extreme Deep Factorization machine is a relatively new deep learning technique that requires no manual feature engineering. Selection of features is very important and selecting raw features often give suboptimal results. Instead we use a factorization machine before applying a deep neural network for feature selection thus giving highly accurate results.

- Extreme Deep Factorization machine is a relatively new deep learning technique that requires no manual feature engineering. Selection of features is very important and selecting raw features often give suboptimal results. Instead we use a factorization machine before applying a deep neural network for feature selection thus giving highly accurate results.

2.2 Product Features

1. This project aims to predict the existence of malware in a Windows system using its operating system characteristics.
2. The deep learning model using neural networks will be used to understand its accuracy in building such a model.
3. Understanding the accuracy and working of using boosting techniques like LGBM in building such a model.
4. Understanding the accuracy and working of using a hybrid like xDeepFM in building such a model.
5. Deep Neural Networks using factorization methods to understand low and high order feature interactions
6. A simple but efficient graphical user interface to give users the satisfactory analysis of their system.

2.3 Operating Environment

The software will run on the Windows operating system devices. All devices that support this version of the operating system will be able to run the software. The software is developed using Python based django application. The application will run on two or more virtual devices simultaneously.
The back end of the project consisting of the prediction model will be run on Jupyter notebook. This system will provide the necessary functionality to implement the data pulling and pushing services. Jupyter notebook also provides for ease of development of the classification model using python as the scripting language.

2.4 Design and Implementation Constraints

1. Network or internet connections are required to interface between the phone and the real-time database.
2. Devices vary in capabilities / technology supported, and thus we cannot guarantee universal access to our application across all Windows platforms.
3. Windows operating system is regularly updated by Microsoft, and future iterations may not be compatible with current version of the application.
4. The software will not be maintained by the developer following release, which may result in compatibility issues in the future.
5. The analysis is made only for Windows systems and malware detection for other operating systems like mac, or linux systems are currently not included in the product.

2.5 Assumptions and Dependencies

To be noted is that the current prediction model revolves only around the working of Windows operating system. The data features may differ when it involves other operating systems like Linux, Mac etc.

3. External Interface Requirements

3.1 User Interfaces

The user interface is mainly run via a python GUI framework. The user will be allowed to enter the details of their operating system as input into the GUI. The GUI is also responsible for presenting a result to the user which is customized to their input. The GUI component for this project is a basic yet functional unit.

3.2 Hardware Interfaces

The database is very large with over 7 billion rows. This makes hardware interfaces and requirements slightly complex. Training the model directly on a user system is not a good idea as commercial laptops do not have to computational power. The computational ability is divided into two major factors - RAM and GPU. The RAM is required to load the database onto the available memory. Commercial laptops come with 8GB RAM which is below the minimum requirements to run a database of this size. Some laptops may have 16 GB RAM but even those are barely meeting the minimum requirements. When it comes to GPU, commercial systems and home systems are behind by a large margin. The best GPU available in a modern laptop is the Nvidia GTX 1060. This GPU is suited for video rendering and does not have enough CUDA cores or multiprocessing threads to deal with the raw computational power for processing and training a database. The GTX 1060 is no match for a professional processing GPU like the Tesla K80.

Both the above problems are solved by using a cloud based computational resource - which in our case is Google Colab

3.3 Software Interfaces

The product will be comprised of interaction between the following software products:

3.3.1 Jupyter Notebook

The Jupyter notebook application allows client server interactions. It enables running of notebooks via the local browser. The use of Jupyter notebook requires no active internet connection, however since our application is interacting with a real time data hosted on firebase an active internet connection is a must. Jupyter notebooks runs as an interpreter which enables line by line code execution. This makes it very easy to detect errors in the code and correct them. This is especially useful as model construction for classification will often take several minutes to run and using a interpreter will produce faster error detection results.

3.3.2 Python 3.6

Python is an interpreted high-level programming language for general-purpose programming. Python provides a large number of libraries and a variety of inbuilt functions. Utilization of these functions is key to the development of the model we intend to use of the classification. Furthermore, python also provides easy methods to read into csv files and JSON files both of which are file formats that will be commonly used during the course of this project.

3.3.3 Google Colab

Google Colab is a free cloud service. Just like Jupyter Notebook, it enables running of notebooks via the local browser. It runs as an interpreter which enables line by line code execution. This makes it very easy to detect errors in the code and correct them. This is especially useful as model construction for classification will often take several minutes to run and using a interpreter will produce faster error detection results.The most important feature that distinguishes Colab from other free cloud services is Colab provides GPU and is totally free. It helps develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.

3.3.4 Windows OS

Basic operating system requirement that is required to detect possible Malware that might affect it and to interact with the above mentioned softwares.

4. Functional Requirements

4.1 Graphical User Interface Module

4.1.1 Description

This module allows the user to provide the details of his/her operating system via a simple interface. The model will work in the backend to analyse the given input and predict if the system will be hit by a malware.

4.1.2 Input / Output Sequences

The user is allowed to give all the required details of their operating system as input. The input is received via the use of forms in a python GUI framework. The GUI framework also provides an output which indicates the chance of a particular system being hit by malware.

4.1.3 Functional Requirements

Req 1. Must have a working internet connection that allows him/her to connect to a network which enables connection to the trained model to predict the probability of being hit by malware

Req 2. A system that has the software and hardware capabilities to support the python GUI framework

4.2 Light GBM Model

4.2.1 Description

Light GBM is a very fitting model to this dataset due to the size of the data and the effectiveness and memory efficiency of this particular method of Gradient boosting. It grows in a tree wise structure. But it differs from other tree-based algorithms due to its horizontal growth structure.

4.2.2 Input / Output Sequences

The input to the LGBM model is the training data set which is used to calculate and predict the probability of a malware infection. There is also a test dataset of smaller size which is derived from the user.

4.2.3 Functional Requirements

Req-1. The system must have access to the test dataset to provide accurate output to the user. The system must have full connectivity to the user data to produce an output.

4.3 XDeepFM Model

4.3.1 Description

Extreme Deep Factorization machine is a relatively new deep learning technique that requires no manual feature engineering. Selection of features is very important and selecting raw features often give suboptimal results. Instead we use a factorization machine before applying a deep neural network for feature selection thus giving highly accurate results.

4.3.2 Input / Output Sequences

The input to the XDeepFM model is the training data set which is used to calculate and predict the probability of a malware infection. There is also a test dataset of smaller size which is derived from the user.

4.3.3 Functional Requirements

Req-1 The system must have access to the test dataset to provide accurate output to the user. The system must have full connectivity to the user data to produce an output.

4.4 Recurrent Neural Network

4.4.1 Description

Recurrent Neural Network is one of the cornerstones of Deep learning. Being one of the oldest yet simplest and reliable model, it works very well for the presented data as computation power is cheaply available. Even if it is not the most efficient method in terms of power consumption, it gives us accurate results with the right hyper parameter tuning. It does have its own issues such as vanishing gradients where the gradient becomes very small. Since the network is feedforward, this reinforces in the future steps making the gradient smaller with each step until it becomes insignificant. This problem is countered using Long Short-Term Memory.

4.4.2 Input / Output Sequences

The input to the XDeepFM model is the training data set which is used to calculate and predict the probability of a malware infection. There is also a test dataset of smaller size which is derived from the user.

4.4.3 Functional Requirements

Req-1 The system must have access to the test dataset to provide accurate output to the user. The system must have full connectivity to the user data to produce an output.

4.5 Data Conversion

4.5.1 Description

The GUI component for python will take data in JSON form which is provided by the testing data. This data however cannot be used in raw form and must be converted to a usable state. Hence this module converts the data from JSON to CSV so that it can be consumed by the GUI component to perform the operations needed on the training data.

4.5.2 Input / Output Sequences

Input will be the JSON format data acquired from the database which can be extracted as training data. The output is CSV formatted data that can be used by the GUI and supplied to the trained model behind the GUI.

4.5.2 Functional Requirements

Req-1 The system must have access to the test dataset to provide accurate output to the user. The system must have full connectivity to the user data to produce an output.

Req-2 A conversion framework within python which enables the data conversion is required.

4.6 Pushing notifications to users

4.6.1 Description

Once a user's message is classified as an abusive message the application will update the real time database with the user's name and the contents of the abusive message. The application will warn the user that repeating this behavior will result in the moderator banning his/her account from the chat room. Furthermore, a moderator can also individually push notifications to the user. This feature will be achieved using the firebase notifications console which also simple easy to use notifications to be sent to individual users or a group of users as well.

4.6.2 Input/output Sequences

- The averaged-out submissions file will be the input to this feature which will consist of the appropriately classified messages.
- The output will be the notifications being pushed to the users on moderator approval.

4.6.3 Functional Requirements

Req-1: An active internet connection to pull the data from the online real time database

Req-2: An API key allowing access to the firebase database by using an administrator account.

5. Non-Functional Requirements

5.1 Safety Requirements

The details about any form of user authentication used in the graphical user interface are safely stored and will not be used for any other purpose than for the authentication system in the current application. The same applies for the data information obtained regarding the operating system state. It will be solely used for the malware prediction purpose as well as the notification of the presence of such malware. Also the data is not just stored on the machine, and so it can be retrieved on any notion of system crashing.

5.2 Security Requirements

The graphical user interface, running as an application on the Windows device should need no additional information other than collected operating system parameters. The rest should be taken care by the wireless security settings on the device which must allow for the application to connect to the server so it can feed information to and from the server. Otherwise, access to the user's personal information from other apps, i.e. calendar information, email, contacts, photos, other web applications etc. is under no circumstance necessary and should be considered a breach of privacy in the event of it being used for any other purpose.

5.3 Software Quality Attributes

If the internet service gets disrupted while sending information to the analytics module, the information can be send again for verification. The software should run smoothly without crashing or freezing, regardless of any machine learning parameters. It should have a very intuitive interface that is easy to learn so the user can focus mostly on the chat itself. Optimally, the architecture of the application might also be flexible enough to be easily adapted for a device with other operating systems or even android and ios devices. At the end of the project, all source code, documentation, as well as any other material related to the development of the application may be made freely available to other developers, where it may be used as reference or for further development.

5.4 Business Rules

Any individual may have use of this project for academic or personal use. The project is part of the development of the open-source project; the code, documents, or other materials used for this project cannot be used for commercial purposes, without the required permissions from the developer's side. Others wishing to further develop the code after the project's completion are free to do so, with the required permissions from the developer's side.

5.5 Performance Requirements

The objective of this project is to develop a reliable multi-headed model for predicting the vulnerability of getting affected by malware in a Windows system. Also, the analysis is not performed directly on the graphical user interface but using the Flask platform to extract the data from the GUI for analysis by a model that will already be trained. This helps in loading of the application very quickly i.e. the response time is very less. Another aspect of performance requirement included are the mechanisms that will be included to ensure appropriate messages are prompted when the server is unable to process requests. Also, the efficiency of the Machine Learning model used for clarification is the one that has performed the highest in the comparative study in the analysis portion of the project.