****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**A System Design Document on**

**Windows based Malware Prediction System using Deep Learning Techniques**

**Submitted by**

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

# ***Bachelor of Engineering in Computer Science & Engineering***

**Under the guidance of**

**Dr. S. Rajarajeswari**

**Associate Professor**

**RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054, 2017-2018, www.msrit.edu**

**Table of Contents**

[Introduction](https://docs.google.com/document/d/1YmTUwMaVsK8qdfliShlW1vm97aMwNzHLzF7scRMjfmM/edit#heading=h.30j0zll) 2

[Purpose](https://docs.google.com/document/d/1YmTUwMaVsK8qdfliShlW1vm97aMwNzHLzF7scRMjfmM/edit#heading=h.1fob9te) 2

[Design Constraints](https://docs.google.com/document/d/1YmTUwMaVsK8qdfliShlW1vm97aMwNzHLzF7scRMjfmM/edit#heading=h.2et92p0) 3

[Roles and Responsibilities](https://docs.google.com/document/d/1YmTUwMaVsK8qdfliShlW1vm97aMwNzHLzF7scRMjfmM/edit#heading=h.tyjcwt) 3

[System Architecture](https://docs.google.com/document/d/1YmTUwMaVsK8qdfliShlW1vm97aMwNzHLzF7scRMjfmM/edit#heading=h.1t3h5sf) 4

[Hardware and Software Detailed Design](https://docs.google.com/document/d/1YmTUwMaVsK8qdfliShlW1vm97aMwNzHLzF7scRMjfmM/edit#heading=h.2s8eyo1) 6

[System Security and Integrity Controls](https://docs.google.com/document/d/1YmTUwMaVsK8qdfliShlW1vm97aMwNzHLzF7scRMjfmM/edit#heading=h.3rdcrjn) 7

[Project References](https://docs.google.com/document/d/1YmTUwMaVsK8qdfliShlW1vm97aMwNzHLzF7scRMjfmM/edit#heading=h.3dy6vkm) 8

**1. Introduction**

Malware is software that is aimed at intentionally causing a system to behave in a way that it should not. It’s 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. So objective of this project - rather than inspecting malware, we can get much better results by analysing the system and its own vulnerabilities against targeted attacks from malware. The outcome is a predictive model that outlines the probability of a system with many parameters being affected in the near future or not. A generic solution that fits all the systems will help plug holes in many security systems - primarily windows machines as they are the most used operating system on the planet currently. This is what our project is aimed at - Malware detection.

**2. 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.

**3. Design Constraints**

* Network or internet connections are required to interface between the phone and the real-time database.
* Devices vary in capabilities / technology supported, and thus we cannot guarantee universal access to our application across all Windows platforms.
* Windows operating system is regularly updated by Microsoft, and future iterations may not be compatible with current version of the application.
* The software will not be maintained by the developer following release, which may result in compatibility issues in the future.
* 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.

**4. Roles and Responsibilities**

As the project team size was limited to only three members the roles and responsibilities of the project members were well cut out from the beginning. Additional responsibilities were given to team members as when new requirements appeared. Development tasks were well distributed in a span of 5 months for the development of the project.

| **Name** | **Role** | **Responsibilities** |
| --- | --- | --- |
| Devika Anil | Product Developer/ Project team Lead | 1) Project management and scheduling.  2)Ensuring smooth communication of project status with team guide  3)Development and understanding of one prediction model.  4) Ensuring that deadlines were met by strictly sticking to schedule |
| Aravind P Anil | Product Developer | 1) Back end development of application  2)Application integration with firebase  3)Application backend integration with the machine learning models  4)Development and understanding of one prediction model.  5) Maintaining database and modules related to working with online database. |
| Aatish Kayyath | Product Developer | 1) Application front end development  2)Testing and debugging application.  3)Development of a hassle free UI for good user experience  4)Development and understanding of one prediction model. |
| Abishek Padaki | Product Developer | 1)Performed comparative study of ML models using data visualisation  2)Testing and debugging application  3)Development of a hassle free UI for good user experience  3)Application backend integration with the machine learning models |

Table 1.2 : Roles and Responsibilities

**5. System Architecture**

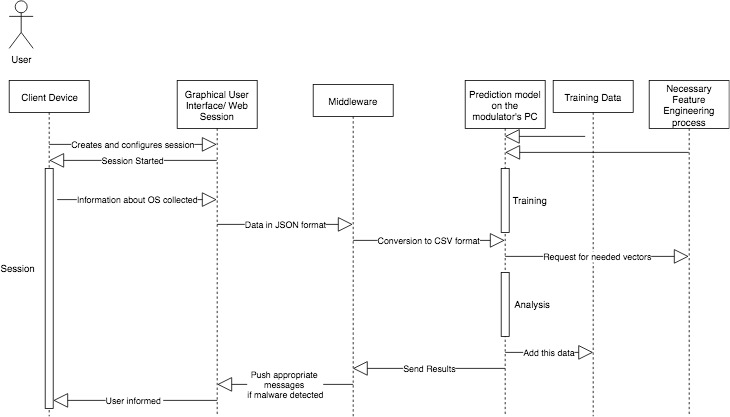
****

Figure 6.1 : Sequence Diagram

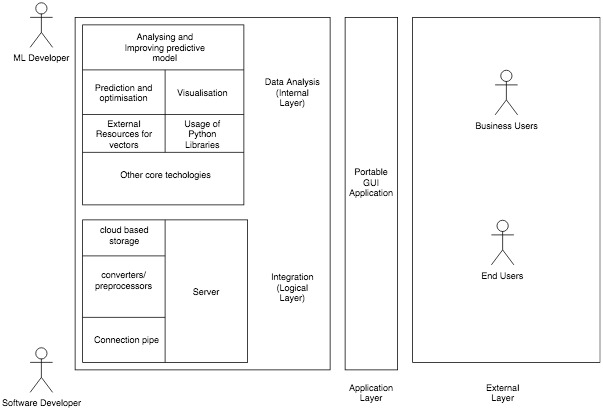


Figure 6.2 : System Architecture Diagram

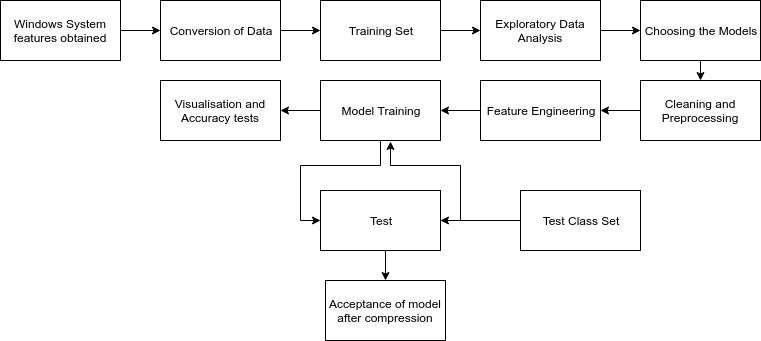


Figure 6.3 : Data Flow Diagram

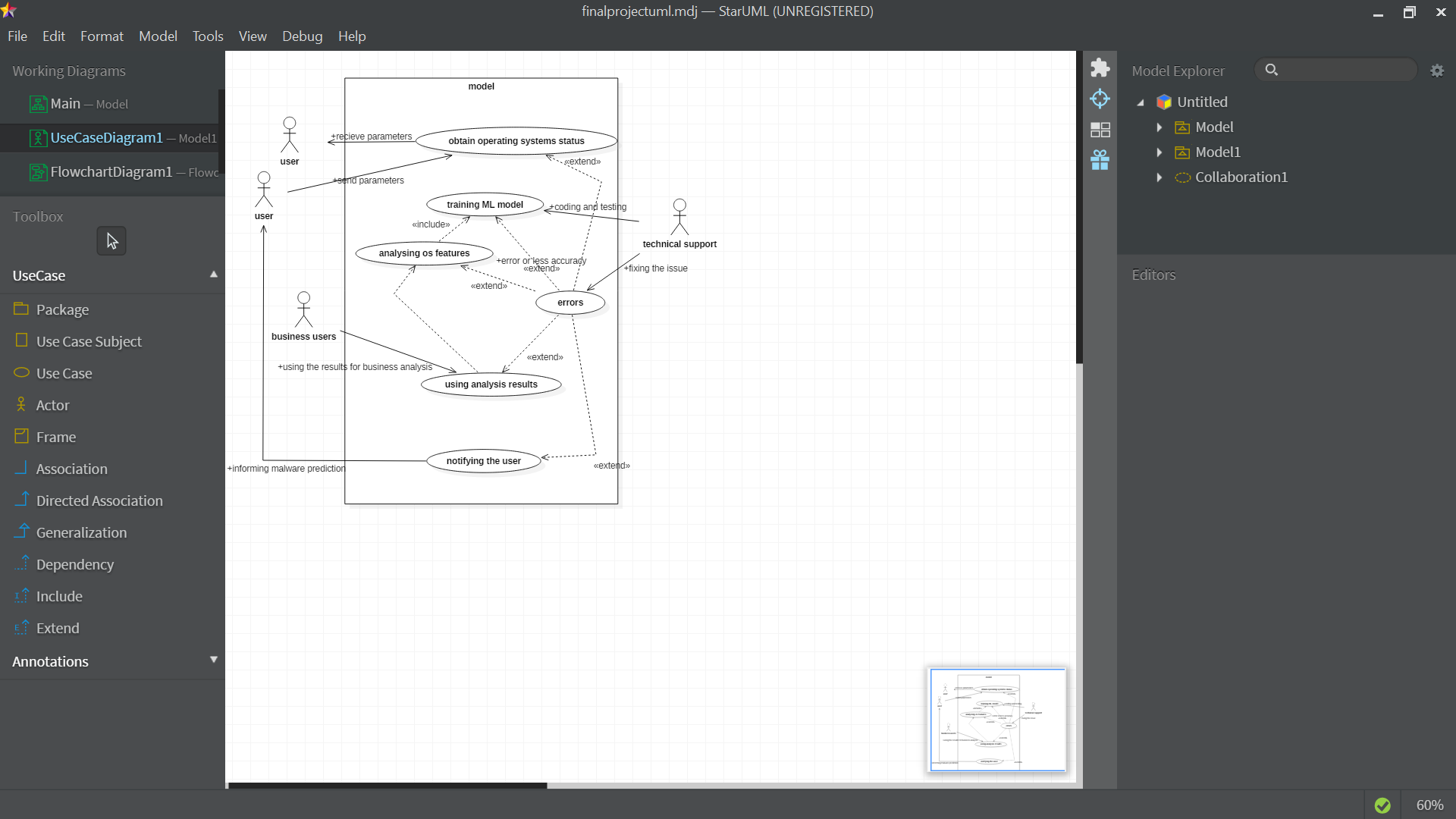


Figure 6.4 : UML Diagram

**6. Hardware and Software detailed design**

**6.1 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.

**6.2 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.

**6.3 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.

**6.4 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.

**7. System Security and Integrity Controls**

The system works with a very large database which contains details of Operating Systems. Whether these are actual instances or generated by permutations and combinations of various options is unclear. However, data security is a high priority task and must be ensured. At no point is the data uploaded to an arbitrary cloud or any anonymous storage service. The only portion where data reaches the internet is during the initial training of the model where the code is run on a powerful machine so as to get faster results.

The codebase is also not stored on any cloud or virtual machine instance at any point of time. The GUI is run completely on a local machine and leak of data is considered a breach of privacy even if it seems highly improbable. The lack of clarity about the extent of sensitivity of data should not interfere with the security of the entirety of the system as the system security should be given high priority. There are no instances or points on the timeline of the system where authorization, identification or authentication is required. Hence, the probability of a breach of data occurring is miniscule. Even if the three modules seem separate, the system works as one functional unit. The python GUI blends in with the models and presents us with an integrated working environment for smooth functioning. The database sits in the backend and does not interfere with the models. However, it still remains a vital part of the system and is linked to the frontend because the input to the GUI is eventually a part of the database

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).

[16] Danny Hendler, Shay Kels and Amir Rubin (2018) “Detecting Malicious PowerShell Commands using Deep Neural Networks”. In: arXiv:1804.04177, Cornell University, 2018.