Software Requirement Specification Document for Malware Detection using Machine Learning

Adham Kamal, Yousef Hany , Yousef Mohammed , Omar Khaled
Supervised by: Dr Nermin Naguib & Assistant supervisor Bassant Kassem

January 16, 2024

Table 1: Document version history

Version	Date	Reason for Change
1.0	9-Jan-2024	SRS First version's specifications are defined.
1.1	11-Jan-2024	Added System Overview Diagram and system scope
1.2	13-Jan-2024	Added System Context Diagram and Functional Requirements
1.3	14-Jan-2024	Added Time Plan and Finalized the SRS Document

GitHub

https://github.com/kamolia1/graduation-project

Contents

1 Introduction **4**

1.1 Purpose of this document 4

1.2 Scope of this document 4

1.3 Business Context 4

2 Similar Systems **5**

2.1 Academic 5

2.1.1 **Intelligent Malware Detection Based on Graph Convolutional Network[1]** 5

2.1.2 **SOMDROID: Android Malware Detection by Artificial Neural Network Trained Using Unsupervised Learning[2]** 5

2.1.3 **Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm[3]** 6

2.2 Business Applications 6

3 System Description **8**

3.1 Problem Statement 8

3.2 System Overview 9

3.3 System Scope 10

3.4 System Context 10

3.5 Objectives 11

3.6 User Characteristics 11

4 Functional Requirements **12**

4.1 System Functions 12

4.2 Detailed Functional Specification 13

5 Design Constraints **16**

6 Non-functional Requirements **17**

6.1 Security 17

6.2 Reliability 17

6.3 Availability 17

6.4 Performance 17

6.5 Usability 17

7 Data Design **17**

8 Preliminary Object-Oriented Domain Analysis **20**

9 Operational Scenarios **20**

10 Project Plan **21**

11 Appendices **21**
11.1 Definitions, Acronyms, Abbreviations 21
11.2 Supportive Documents 23
11.2.1 Survey 23
12 References **24**

Abstract

Malware detection refers to a collection of techniques used to detect all types of malicious software that might harm the victim's system. Ai powered machine learning enhances malware detection processes by analyzing large amount of data by identifying the behavior of a malware to detect it. AI can detect malware through its behavior and signature techniques. We aim to protect computers from malicious infection and to provide a reliable solution, helping users protect their systems from evolving malware threats. The primary feature of the system is to detect and remove malicious code using AI algorithms to prevent malware from causing damage to your computer. The user will be notified when the system detects a malware and it recommends an action to the user, and it alerts users about malicious links. If the malware signature doesn't exist in the database, it will be added as a new signature to the database.

1 Introduction

1.1 Purpose of this document

This document main purpose is to describe the requirement specifications and the crucial functionalities of this system, explaining step by step the process of our system development. The aim of this document is to highlight the overview explanation of the final software product and to illustrate how the process of detecting a malware would be carried out based on our selected techniques.

1.2 Scope of this document

This document will explain the function and non-functional requirements in details as we will clarify the functionalities of the system, also this document provides detailed explanation of how our system works which will be detecting malicious files, notifying the user about any spam/malicious URL's and based on this a recommended action will be given to the user. This document illustrates that our system will be a windows desktop application, despite of our system's specifications our application does not have any concerns with MacOs, Android or IOS operating systems.

1.3 Business Context

Traditional anti-viruses work on the signature only to detect malware which is not enough to secure business organizations nowadays due to the continuous evolution of malware and infection methods. The Business needs are focused on the automation of the detection process intelligently using AI capabilities that will ensure to get high detection accuracy to protect its valuable assets and reputation from severe damage. Our system will solve this problem by using combination of the signature and behavior approaches to enhance the effectiveness of detecting a malware to meet the market requirements, so that the organizations can cooperate with the unstoppable threats and disasters caused by modern malwares.

2 Similar Systems

2.1 Academic

2.1.1 Intelligent Malware Detection Based on Graph Convolutional Network[1]

Li, Shanxi, et al used GCN Algorithm on a dataset with 6688 malicous software and 6984 normal programs. The malicous ones were obtained using VirusTotal and the normal ones were from system applications , they used 80% of the data for training and 20% for testing. Their results have shown that GCN had an accuracy of 98.32% when tested and also had lower false positive rates than other algorithms.

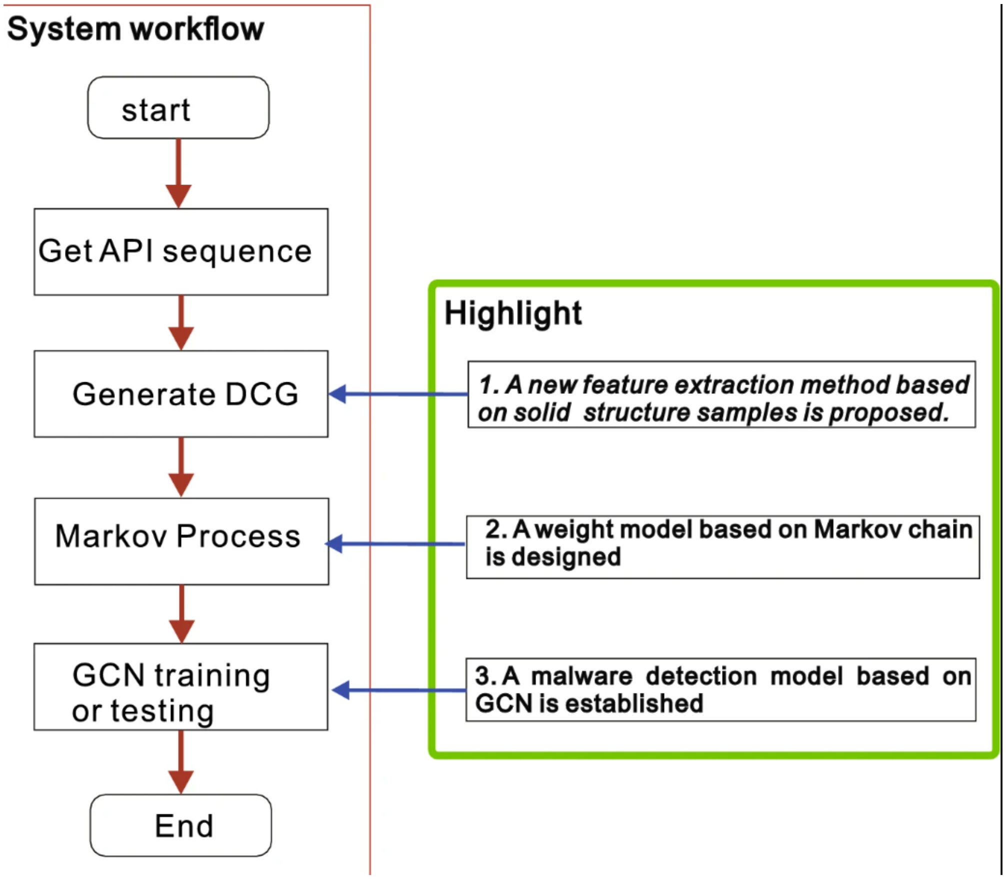

System workflow
start
Get API sequence
Generate DCG
Markov Process
GCN training or testing
End
Highlight
1. A new feature extraction method based on solid structure samples is proposed.
2. A weight model based on Markov chain is designed
3. A malware detection model based on GCN is established

Figure 1

2.1.2 SOMDROID: Android Malware Detection by Artificial Neural Network Trained Using Unsupervised Learning[2]

Mahindru, Arvind, and A. L. Sangal. Developed a framework called SOMDROID, that operates using the unsupervised machine learning algorithm. They used the SOM algorithm on 500,000 .apk files that were obtained through app crawler , the files were categorized into 30 categories of Android. Their framework obtained 98.7% accuracy which they claim is higher by 2% than existing anti-virus scanners.

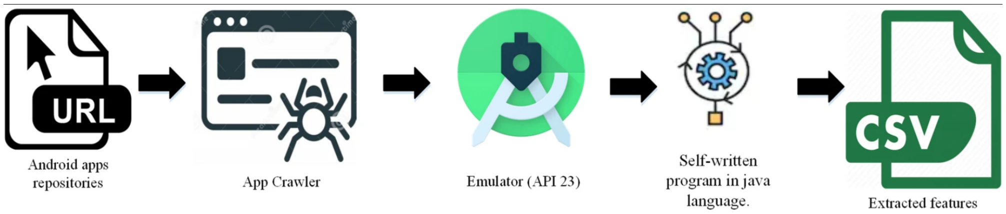

Android apps repositories
URL
App Crawler
Emulator (API 23)
Self-written program in java language.
CSV
Extracted features

Figure 2

2.1.3 Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm[3]

Roseline, S. Abijah, et al. used a visualization approach representing deep learning techniques but with higher performance. They used the dataset Malimg which has 9,339 malware images belonging to 25 malware classes, this model performed better than other techniques scoring an accuracy of 98.65% when testing on the forementioned dataset.

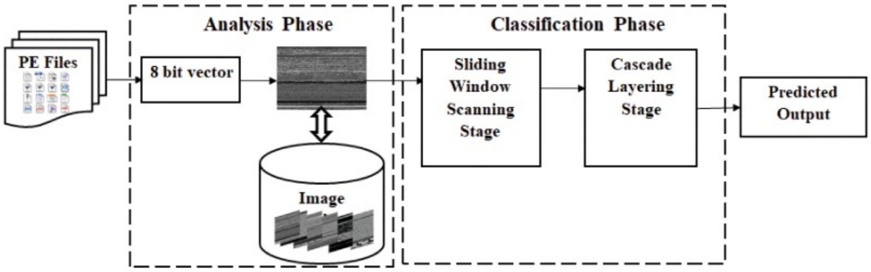


Figure 3

2.2 Business Applications

FortiGuard antivirus service

The FortiGuard Antivirus Service perform automatic updates to protect against polymorphic attacks , spyware, viruses and other threats depending on a patented content pattern recognition language (CPRL). The anti-malware system prevent known and unknown variants of malware. FortiGuard antivirus uses a technology that include signature based detection , behavior based and heuristic utilizing AI analysis. FortiGuard Antivirus help to protect the organization from zero-day attacks. https://www.fortinet.com/support/support-services/fortiguard-security-subscriptions/antivirus

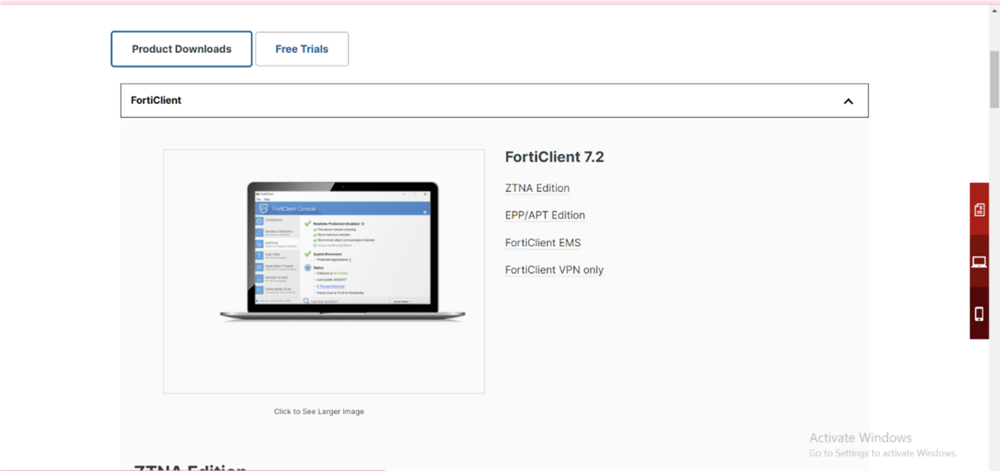
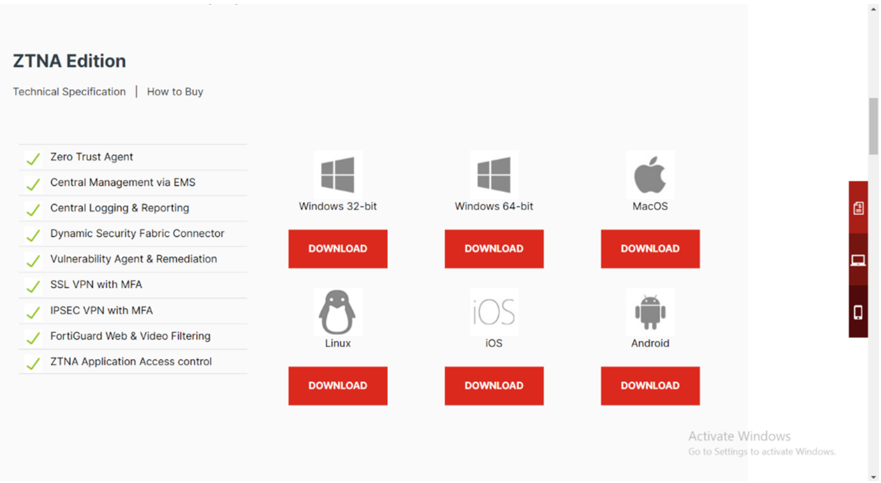


Figure 4



ZTNA Edition

Technical Specification | How to Buy

- ✓ Zero Trust Agent
- ✓ Central Management via EMS
- ✓ Central Logging & Reporting
- ✓ Dynamic Security Fabric Connector
- ✓ Vulnerability Agent & Remediation
- ✓ SSL VPN with MFA
- ✓ IPSEC VPN with MFA
- ✓ FortiGuard Web & Video Filtering
- ✓ ZTNA Application Access control

Figure 5

Crowdstrike https://www.crowdstrike.com/cybersecurity-101/endpoint-security/next-generation-antivirus-ngav/ Next generation antivirus (NGAV) uses a combination of ai, behavior detection and machine learning algorithms so unknown and known threats can be prevented immediately. NGAV is considered a cloud based solution which is deployed in hours and not in months , and the maintenance of software and updating the database that contains the signature is eliminated .Crowd Strike Falcon Prevent, this antivirus stop the attacks using AI/ML and through our threat intelligence, script control , indicator of attacks and advanced memory scanning the antivirus detects and blocks malicious behaviors in it's early stage.

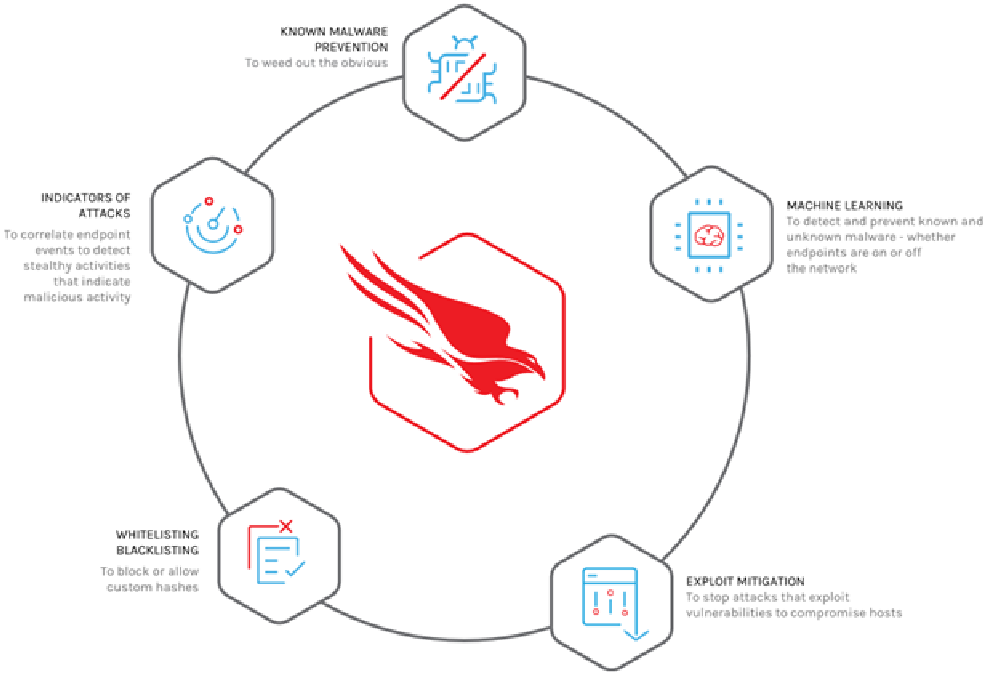

KNOWN MALWARE PREVENTION
To weed out the obvious
MACHINE LEARNING
To detect and prevent known and unknown malware - whether endpoints are on or off the network
EXPLOIT MITIGATION
To stop attacks that exploit vulnerabilities to compromise hosts
WHITELISTING BLACKLISTING
To block or allow custom hashes
INDICATORS OF ATTACKS
To correlate endpoint events to detect stealthy activities that indicate malicious activity

Figure 6

3 System Description

3.1 Problem Statement

Malware is constantly evolving at a speedy rate, cybercriminals continue to develop new techniques to create and distribute malware that will cause financial problems and lack of trust between an organization and its clients, also violates the individual's privacy making it tough for security measures to keep up. Organizations have a critical problem with the severe absence of awareness when it comes to cybersecurity field which will make the proper use of intelligent antivirus software an obstacle to overcome. Traditional malwares are detected faster, and it is required to use AI to improve malware detection process to detect zero-day attacks and brand-new types of malwares with unknown signature.

3.2 System Overview

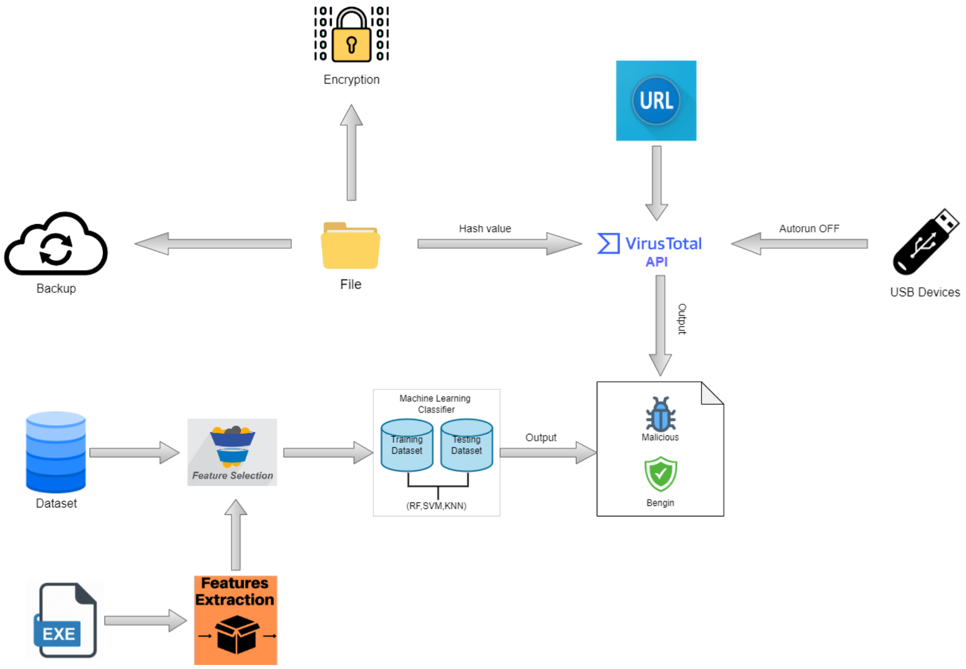


Figure 7: System Overview

- Users can select a specific file to upload to the system for scanning for malicious content using known signatures on VirusTotal.
- The dataset contains malware and benign samples with multiple features. Specific features are selected for the machine learning classifier, and the data is divided into training and testing sets.
- Users can select an exe file, and the selected features will be extracted to the machine learning model to detect malware based on its behavior.
- Users can take a recommended action to delete or quarantine the file when it was detected as a malicious.
- Users have the option to encrypt a selected file to protect its confidentiality from unauthorized access.
- For backup purposes, users can send a file to a secure cloud to ensure it won't be deleted or lost in case of malware infection.
- The system allows users to detect suspicious or malicious URLs by scanning them using the VirusTotal API.
- Users can scan USB devices using the system to identify potential threats that may harm their devices.

3.3 System Scope

Our system detects malware in different types of files using signature-based detection by calculating the hash of selected file and then comparing it through VirusTotal API. The system will detect malware in exe files through its behavior from the memory dumb analysis using AI algorithms. The system will recommend an action for the user to take when a malicious file is detected whether to delete the file or quarantine the file. The system will also detect suspicious URL's that might harm the user's device by checking the URL through VirusTotal API. The system can encrypt selected files that contain confidential information and can also backup certain files that the user selects for future retrieval. The system can scan USB devices to detect malware.

3.4 System Context

• The user would select a file or URL to get scanned.

• The system would compute the hash of the selected file or simply take the URL and send it to VirusTotal API which will give the result.

• The system will take training & testing data from the dataset and send it to the model classifier.

• The system then starts to extract selected features from the exe file uploaded by the user then send the features to the model and get a prediction output whether the file is malicious or not then the system will notify the user.

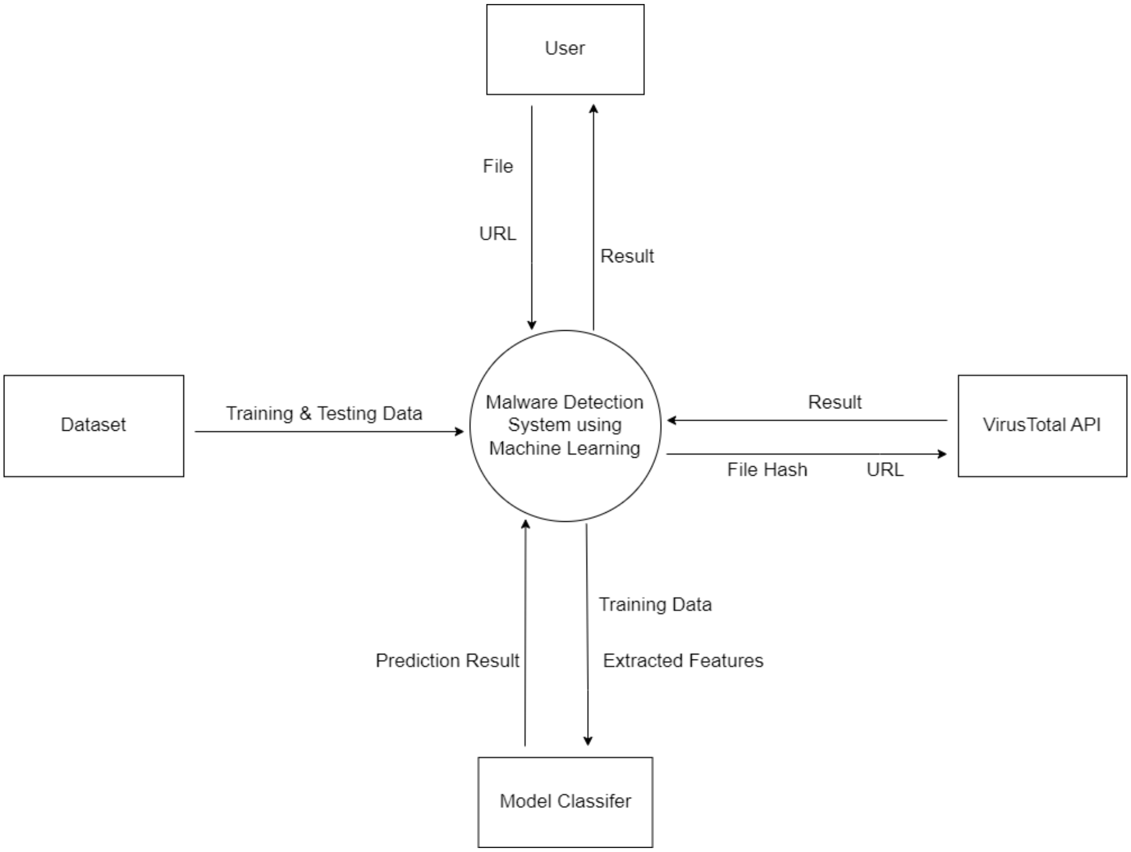


Figure 8: System Context

3.5 Objectives

- Malware detection using a signature and behavior approaches.

- Creating an AI model that detects known and unknown types of malwares.

- Creating an application that ensures fast and reliable protection against malicious infections.

- Creating an application that has additional features which prevent data loss and securing confidential files against unauthorized access.

- Creating an application that alerts the user about any suspicious websites that might cause a severe damage.

3.6 User Characteristics

The user should have simple understanding of windows operating system and to be familiar with how to use it. The user can be any individual person or organization that needs security measures to protect its assets. Users can be of any educational level above elementary school.

4 Functional Requirements

4.1 System Functions

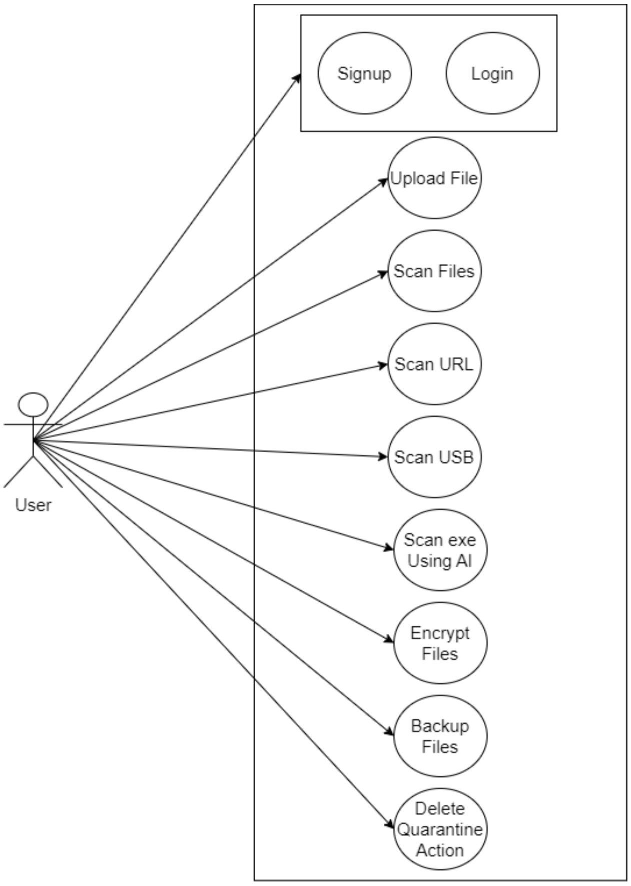


Figure 9: Use Case Diagram

- **FR01 :**The user should be able to sign up to the system.

- **FR02 :**The user should be able to login to the system.

- **FR03 :**The user shall be able to upload a specific file to scan it

- **FR04 :**The user should be able to check if the file is malware or a good ware

- **FR05 :**The user should scan the URL or link to check if it's safe or not

- **FR06 :**The user should be able to scan USB devices attached to the computer.

- **FR07 :**The user should scan an executable file and then selected features of the file behavior will be added to a machine learning model to determine if the file is malicious or not

• **FR08 :**The user should be able to encrypt a selected file to protect confidential information.

• **FR09 :**The user should be able to backup a file to prevent any data loss.

• **FR10 :**The user should be able to take a recommended action whether to delete or quarantine the malicious file.

4.2 Detailed Functional Specification

Table 2: Scan Files

Name	Scan File.
Code	FR04.
Priority	High.
Critical	Detection of malicious files.
Description	Function take the uploaded file then compute its hash and use VirusTotal API to compare the hash to classify the file.
Input	Any type of Files.
Output	Malware or Safe File.
Pre-conditions	User uploads a file.
Post-conditions	User will Know if the file is malicious or not then choose to delete or quarantine the malicious file.
Dependency	The FR04 depends on FR03
Risk	API out of service or Internet connection loss, the user won't be able to scan a file and get false positive detections.

Table 3: Scan URL

Name	Scan URL
Code	FR05
Priority	High
Critical	Detecting malicious Links.
Description	Function take a string input specifying the URL then it sends it to VirusTotal API to check the link and return result if the link is malicious or not.
Input	String input :Link.
Output	Malicious Link or Safe Link.
Pre-conditions	User Enter the Link to be scanned.
Post-conditions	User will know alerted about the malicious link.
Dependency	Depends on user entry.
Risk	API out of service or Internet connection loss, the user won't be able to scan a URL and get false positive detections.

Table 4: Scan executable using AI

Name	Scan executable using AI.
Code	FR07.
Priority	High.
Critical	Detecting Behavior of malware.
Description	Function take an executable file then extracts the selected features from this file through its behavior then the machine learning model will detect the malware.
Input	Executable file.
Output	File malicious or not.
Pre-conditions	User uploads a file.
Post-conditions	User will Know if the file is malicious or not then choose to delete or quarantine the malicious file.
Dependency	The FR07 depends on FR03.
Risk	False positive detections or not being able to detect the malware.

Table 5: Encrypt a File

Name	Encryption.
Code	FR08.
Priority	High.
Critical	Encryption of selected file.
Description	Function take the uploaded file then encrypt to protect confidential information.
Input	Text Files
Output	Encrypted Files
Pre-conditions	User uploads a file.
Post-conditions	User will have an encrypted secure file.
Dependency	FR08 depends on FR03
Risk	encryption method being broken by malicious attackers.

5 Design Constraints

- Operating system: Windows 7, 8, 8.1 ,10 and 11.

- RAM: 2 GB or higher.

- Processor: intel i3 or higher.

- Requires an internet connection:2MB upload and 4MB download minimum.

- Supported languages: English.

- Our system depends on VirusTotal API which has a request rate 4 lookups / min , daily quota 500 lookups / day and monthly quota 15.5 K lookups / month.

- Detection using machine learning can accept only executable files.

6 Non-functional Requirements

6.1 Security

The application can encrypt certain files that the user chooses and no one can decrypt this files as the key is only known to the user. The application allow the backup of certain files to prevent data loss.

6.2 Reliability

The application relies on the VirusTotal API for detecting malware using signature and this Website is trusted by many security vendors, so the application will provide high accuracy in detecting different families of malware.

6.3 Availability

The application can run on any windows operating system and only needs internet connection and the user can select a specific file or a URL to be scanned 24/7.

6.4 Performance

The application works with high efficiency and fast detection process.

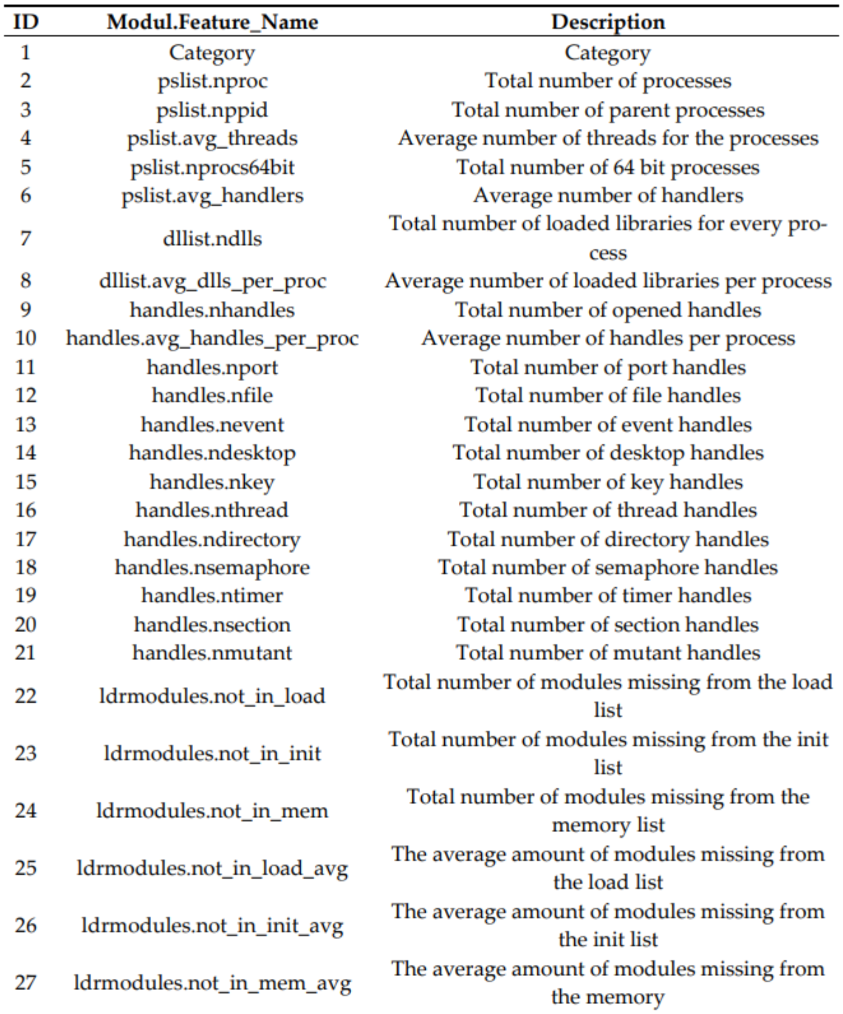
6.5 Usability

The application will have a simple GUI that any type of users can deal with without any problems.

7 Data Design

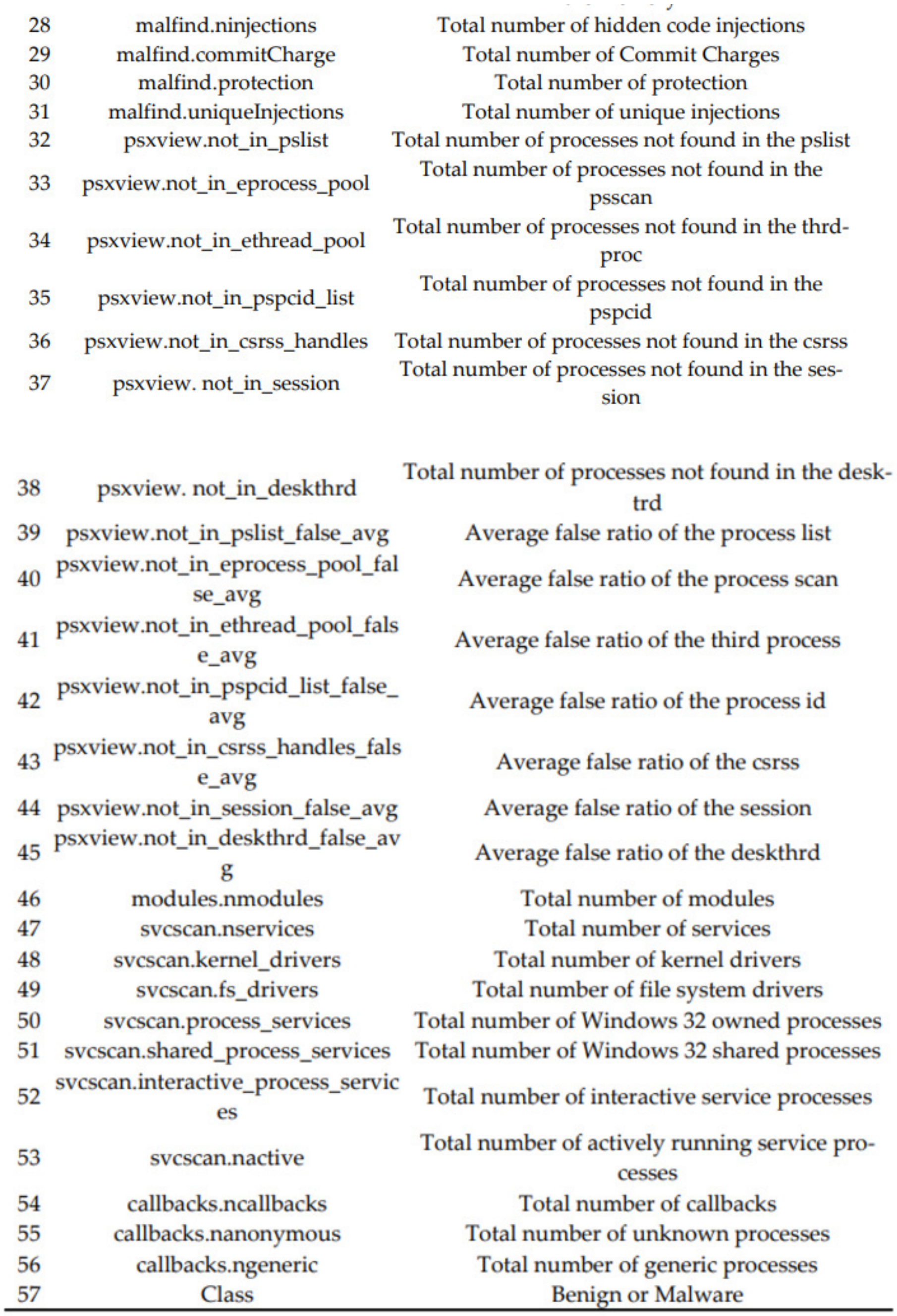
The malware dataset[10] is designed to test malware detection through memory analysis. The dataset tries to mimic a real situation as much as it can using malware that is widespread in the real life. It consists of Trojan Horse, Spyware and Ransomware files, It is a balanced dataset that can be useful in testing malware detection programs.The dataset has 58,596 samples divided into 29,298 not malicious files and 29,298 malicious files.

The Dataset main use is for memory dump processes to evade dumping process to appear in the saved memory dumps and this illustrates how would be an efficient example of what a normal user will be running on the PC when the malware attack happens



ID	Modul.Feature_Name	Description
1	Category	Category
2	pslist.nproc	Total number of processes
3	pslist.nppid	Total number of parent processes
4	pslist.avg_threads	Average number of threads for the processes
5	pslist.nprocs64bit	Total number of 64 bit processes
6	pslist.avg_handlers	Average number of handlers
7	dllist.ndlls	Total number of loaded libraries for every process
8	dllist.avg_dlls_per_proc	Average number of loaded libraries per process
9	handles.nhandles	Total number of opened handles
10	handles.avg_handles_per_proc	Average number of handles per process
11	handles.nport	Total number of port handles
12	handles.nfile	Total number of file handles
13	handles.nevent	Total number of event handles
14	handles.ndesktop	Total number of desktop handles
15	handles.nkey	Total number of key handles
16	handles.nthread	Total number of thread handles
17	handles.ndirectory	Total number of directory handles
18	handles.nsemaphore	Total number of semaphore handles
19	handles.ntimer	Total number of timer handles
20	handles.nsection	Total number of section handles
21	handles.nmutant	Total number of mutant handles
22	ldrmodules.not_in_load	Total number of modules missing from the load list
23	ldrmodules.not_in_init	Total number of modules missing from the init list
24	ldrmodules.not_in_mem	Total number of modules missing from the memory list
25	ldrmodules.not_in_load_avg	The average amount of modules missing from the load list
26	ldrmodules.not_in_init_avg	The average amount of modules missing from the init list
27	ldrmodules.not_in_mem_avg	The average amount of modules missing from the memory

Figure 10: Explanation of the different attributes of the CIC-MalMem-2022 dataset



28	malfind.ninjections	Total number of hidden code injections
29	malfind.commitCharge	Total number of Commit Charges
30	malfind.protection	Total number of protection
31	malfind.uniqueInjections	Total number of unique injections
32	psxview.not_in_pslist	Total number of processes not found in the pslist
33	psxview.not_in_eprocess_pool	Total number of processes not found in the psscan
34	psxview.not_in_ethread_pool	Total number of processes not found in the thrd-proc
35	psxview.not_in_pspcid_list	Total number of processes not found in the pspcid
36	psxview.not_in_csrss_handles	Total number of processes not found in the csrss
37	psxview. not_in_session	Total number of processes not found in the session
38	psxview. not_in_deskthrd	Total number of processes not found in the desktrd
39	psxview.not_in_pslist_false_avg	Average false ratio of the process list
40	psxview.not_in_eprocess_pool_false_avg	Average false ratio of the process scan
41	psxview.not_in_ethread_pool_false_avg	Average false ratio of the third process
42	psxview.not_in_pspcid_list_false_avg	Average false ratio of the process id
43	psxview.not_in_csrss_handles_false_avg	Average false ratio of the csrss
44	psxview.not_in_session_false_avg	Average false ratio of the session
45	psxview.not_in_deskthrd_false_avg	Average false ratio of the deskthrd
46	modules.nmodules	Total number of modules
47	svcscan.nservices	Total number of services
48	svcscan.kernel_drivers	Total number of kernel drivers
49	svcscan.fs_drivers	Total number of file system drivers
50	svcscan.process_services	Total number of Windows 32 owned processes
51	svcscan.shared_process_services	Total number of Windows 32 shared processes
52	svcscan.interactive_process_services	Total number of interactive service processes
53	svcscan.nactive	Total number of actively running service processes
54	callbacks.ncallbacks	Total number of callbacks
55	callbacks.nanonymous	Total number of unknown processes
56	callbacks.ngeneric	Total number of generic processes
57	Class	Benign or Malware

Figure 11: Explanation of the different attributes of the CIC-MalMem-2022 dataset

8 Preliminary Object-Oriented Domain Analysis

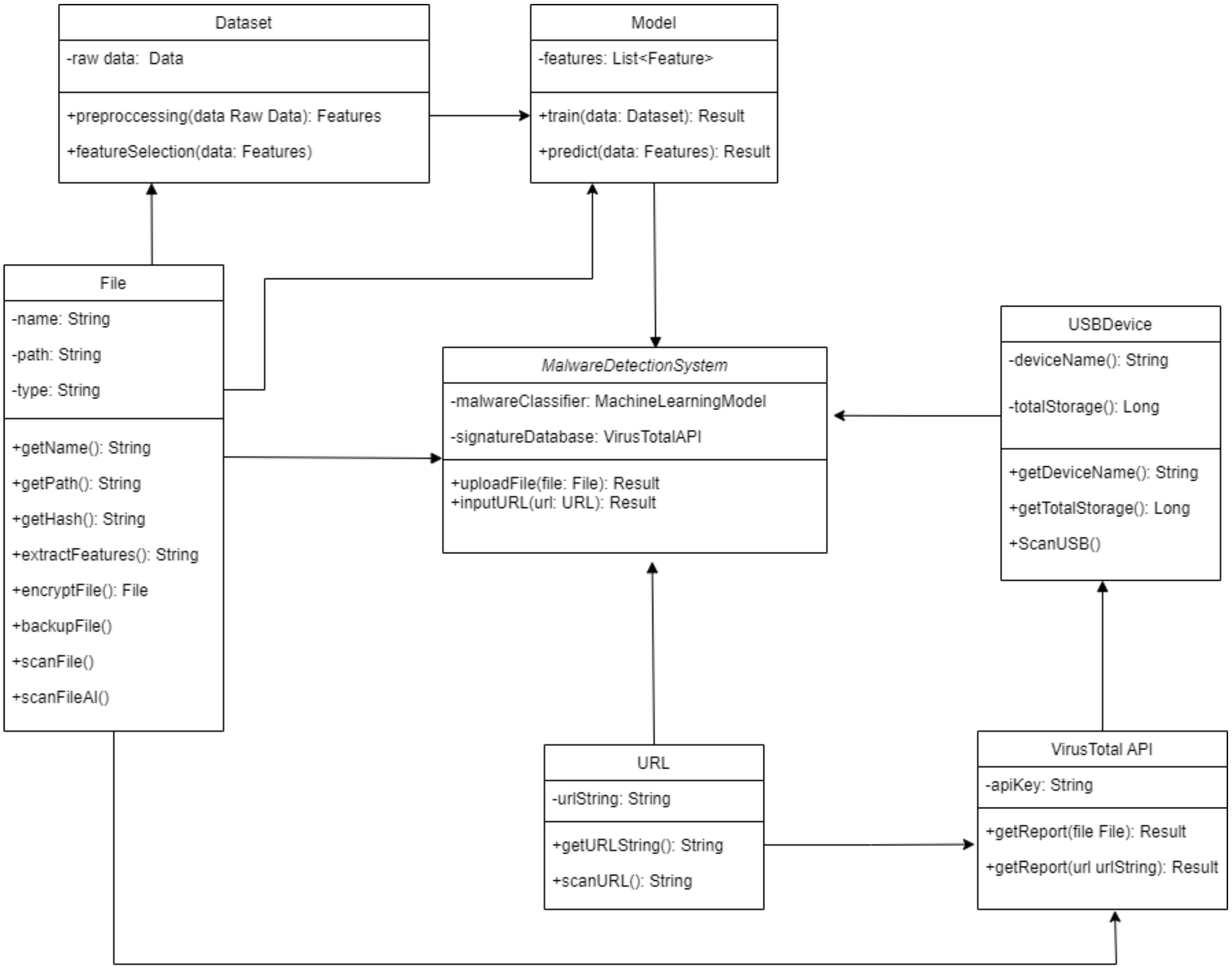


Figure 12: Class Diagram

9 Operational Scenarios

Scenario 1: The user launches the anti-virus and upload a specific file then scan it using the signature based detection which will notify the user if the file is malicious or not and recommend an action for the user to delete or quarantine the malicious file.

Scenario 2: The user can upload an executable file then scan it using the behavior based detection which will detect if the executable file is malicious or not and suggest an action for the user whether to delete or quarantine the malicious file.

Scenario 3: The user can add a URL that he thinks it's malicious and the system will notify him if the website he is visiting is malicious or not.

Scenario 4: The user can choose a certain text document that he wants to encrypt and the system will give him the encrypted file in order to not be accessed by anyone.

Scenario 5: The user should first register and login to his account in order to be able to backup files he choose as the backup will be on the cloud.

Scenario 6: When the user add a USB device to his computer the system will disable the auto run feature and scan files in the USB for malware and then show the malicious files for him.

10 Project Plan

Team members will be assigned by their initials in the time plan:
• Adham Kamal (A)
• Omar Khaled (O)
• Yousef Hany (YH)
• Yousef Mohammed (YM)

11 Appendices

11.1 Definitions, Acronyms, Abbreviations

- **AI:** Artificial Intelligence.
- **ML:** Machine Learning.
- **Malware:** Malicious Software.
- **KNN:** K-Nearest Neighbors is a supervised machine learning algorithm.
- **RF:** Random Forest is a supervised machine learning algorithm.
- **SVM:** Support Vector Machine is a supervised learning algorithm to solve complex classification.
- **AI-based Malware Detection:** Malware detection using Artificial Intelligence capabilities.

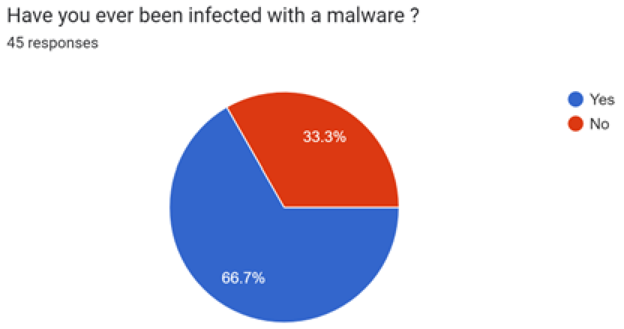
Table 6: Time Plan

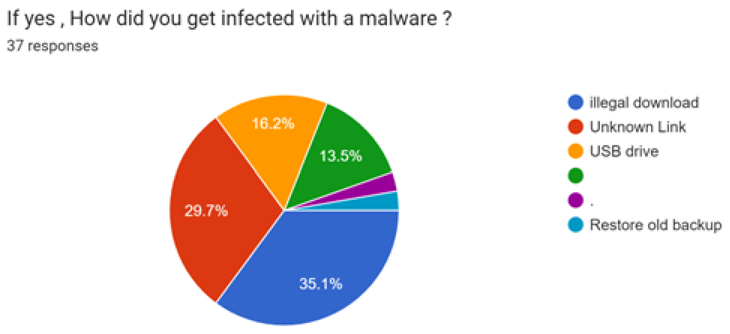
Task	Start Date	End Date	Duration	Assigned To
Proposal Documentation	2/11/2023	16/11/2023	14 days	All
Proposal Survey	2/11/2023	9/11/2023	7 days	All
Proposal Presentation	12/11/2023	16/11/2023	4 days	All
Searching for Dataset	2/11/2023	5/11/2023	3 days	All
Data Preprocessing	5/11/2023	7/11/2023	2 days	A,YH
SRS Documentation	15/12/2023	14/1/2024	30 days	All
SRS Presentation	10/1/2024	14/1/2024	4 days	All
Feature Selection	30/12/2023	16/1/2024	16 days	All
Comparing ML Models	30/12/2023	16/1/2024	17 days	O,YM
Defining Signature for Detection	1/1/2024	5/1/2024	4 days	All
Signature Based File Scanning & API	5/1/2024	14/1/2024	9 days	A,YH
URL Scanning	5/1/2024	14/1/2024	9 days	O,YM
GUI for Demo	5/1/2024	10/1/2024	5 days	All
Files Encryption	N/A	N/A	N/A	N/A
Files Backup	N/A	N/A	N/A	N/A
USB Scanning	N/A	N/A	N/A	N/A
Integrating Classifier with Desktop Application	N/A	N/A	N/A	N/A
Features Extraction	N/A	N/A	N/A	N/A
Behavior Based Executable File Scanning	N/A	N/A	N/A	N/A
SDD Documentation	N/A	N/A	N/A	N/A
SDD Presentation	N/A	N/A	N/A	N/A

- **Signature-based Detection:** Malware detection based on specific patterns or signatures.
- **Behavior-based Detection:** Malware detection based on analyzing the behavior of suspicious software.
- **GCN:** Graph Convolutional Network is a semi-supervised learning on graph-structured data.
- **API:** Application Programming Interface.
- **URL:** Uniform Resource Locator.
- **USB:** Universal Serial Bus.
- **EXE:** executable files.
- **NGAV:** Next Generation Anti-Virus.

11.2 Supportive Documents

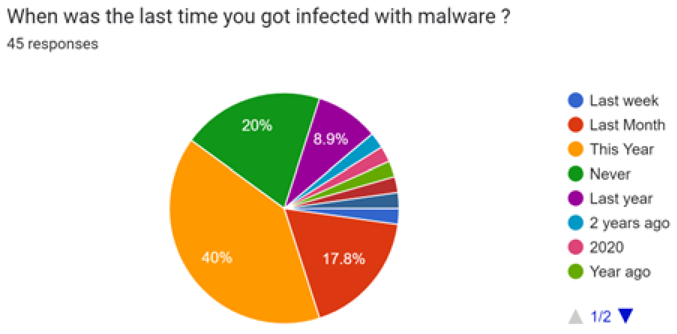
11.2.1 Survey





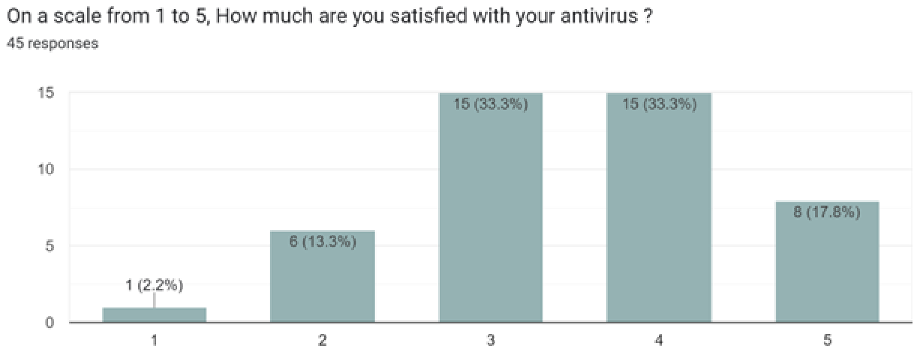
If yes , How did you get infected with a malware ?

37 responses



When was the last time you got infected with malware ?

45 responses



On a scale from 1 to 5, How much are you satisfied with your antivirus ?

45 responses

12 References

References

[1] Li, Shanxi, et al. "Intelligent Malware Detection Based on Graph Convolutional Network." *The Journal of Supercomputing*, 24 Aug. 2021. `https://doi.org/10.1007/s11227-021-04020-y`. Accessed 11 Jan. 2024.

[2] Mahindru, Arvind, and A. L. Sangal. "SOMDROID: Android Malware Detection by Artificial Neural Network Trained Using Unsupervised Learning." *Evolutionary Intelligence*, 12 Nov. 2020. https://doi.org/10.1007/s12065-020-00518-1. Accessed 11 Jan. 2024.

[3] Roseline, S. Abijah, et al. "Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm." *IEEE Access*, vol. 8, 2020, pp. 206303–206324. https://doi.org/10.1109/access.2020.3036491. Accessed 11 Jan. 2024.

[4] Jobair, Md, et al. "Malware Detection and Prevention Using Artificial Intelligence Techniques." 22 June 2022. https://arxiv.org/ftp/arxiv/papers/2206/2206.12770.pdf. Accessed 11 Jan. 2024.

[5] Alkahtani, Hasan, and Theyazn H. H. Aldhyani. "Artificial Intelligence Algorithms for Malware Detection in Android-Operated Mobile Devices." *Sensors*, vol. 22, no. 6, 15 Mar. 2022, p. 2268. https://doi.org/10.3390/s22062268. Accessed 11 Jan. 2024.

[6] Akhtar, Muhammad Shoaib, and Tao Feng. "Malware Analysis and Detection Using Machine Learning Algorithms." *Symmetry*, vol. 14, no. 11, 1 Nov. 2022, p. 2304. www.mdpi.com/2073-8994/14/11/2304. https://doi.org/10.3390/sym14112304. Accessed 11 Jan. 2024.

[7] Sharma, Sanjay, et al. "Detection of Advanced Malware by Machine Learning Techniques." *Advances in Intelligent Systems and Computing*, 31 Aug. 2018, pp. 333–342. https://doi.org/10.1007/978-981-13-0589-4_31. Accessed 11 Jan. 2024.

[8] Senanayake, Janaka, et al. "Android Mobile Malware Detection Using Machine Learning: A Systematic Review." *Electronics*, vol. 10, no. 13, 5 July 2021, p. 1606. https://doi.org/10.3390/electronics10131606. Accessed 11 Jan. 2024.

[9] Djenna, Amir, et al. "Artificial Intelligence-Based Malware Detection, Analysis, and Mitigation." *Symmetry*, vol. 15, no. 3, 8 Mar. 2023, p. 677. https://doi.org/10.3390/sym15030677. Accessed 11 Jan. 2024.

[10] Dener, Murat, Gökçe Ok, and Abdullah Orman. “Malware Detection Using Memory Analysis Data in Big Data Environment." *Applied Sciences* 12.17 (2022): 8604. https://www.mdpi.com/2076-3417/12/17/8604 DOI: 10.3390/app12178604 Accessed 11 Jan. 2024.