MALWARE DETECTION AND ANALYSIS USING MACHINE LEARNING

Submitted in partial fulfillment of the

requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

by

MANOJ SIRIGIRI (Reg. No - 39110604)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHISALAI,

CHENNAI - 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **MANOJ SIRIGIRI (39110604)** who carried out the Project Phase-2 entitled **"MALWARE DETECTION AND ANALYSIS USING MACHINE LEARNING"** under my supervision from Jan 2023 to April 2023.

Internal Guide
Ms. R. YOGITHA M.E., (PH.D.,)

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.,

Submitted for Viva voce Examination held on 19-04-2023

Internal Examiner **External Examiner**

DECLARATION

I, **MANOJ SIRIGIRI (Reg.No- 39110604),** hereby declare that the Project Phase-2 Report entitled "**MALWARE DETECTION AND ANALYSIS USING MACHINE LEARNING"** done by me under the guidance of **Ms. R. Yogitha M.E., (Ph.D.,)** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 19-04-2023 Manoj sirigiri

PLACE: Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, **Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. R. Yogitha M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-1 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The increasing prevalence of malware is one of the most significant threats to the security of computer systems and the Internet. Malware is any software that has the intention of performing malicious activities on a targeted system, including stealing data, disrupting operations, and causing financial losses. There are many types of malwares, and attackers can use various communication strategies to infect systems and spread malware, including Trojans, keyloggers, port forwarding, application format converters, and social engineering tactics. To improve malware detection and classification, I studied various malware types and communication strategies and applied machine-learning algorithms to classify malware as malicious or non-malicious. The machine learning algorithms I used in our study were the Random Forest, K-Nearest Neighbor, and Support Vector Machine algorithms. I applied these algorithms to a dataset containing a large number of malware samples and a set of features that describe each sample's behavior and characteristics. To evaluate the performance of the machine learning models, I analyzed the resulting classification report, accuracy, and f1 score metrics. The classification report provided detailed information about the classification results, including the precision, recall, and f1- score for each class. The accuracy metric measured the percentage of correctly classified samples, while the f1 score combined precision and recall to evaluate the classifier's accuracy. The Random Forest algorithm performed the best for classifying both malicious and non-malicious software, achieving the highest f1-score for the validation dataset. The results also showed that the K- Nearest Neighbor and Support Vector Machine algorithms performed well but did not achieve the same level of accuracy as the Random Forest algorithm. By using machine learning algorithms to classify malware, I aim to improve the identification and classification of malware, thus enhancing online privacy for individuals. In my analysis, I have demonstrated how applying machine learning techniques can identify and prevent malware attacks, protecting computer systems and the internet. This study highlights the importance of understanding various malware types and communication strategies and the potential impact of malware attacks on organizations and individuals. By using machine learning algorithms to classify malware, I can effectively detect and prevent malware attacks, thus enhancing online privacy and security.

INDEX

LIST OF FIGURES

LIST OF ABBRIVATIONS

S.NO	ABBRIVATION	FULL FORM
1	AI	ARTIFICIAL INTELLIGENCE
2	SVM	SUPPORT VECTOR MACHINE
3	ML	MACHINE LEARNING
4	KNN	K-NEAREST NEIGHBOR
5	HTTPS	HYPERTEXT TRANSFER PROTOCOL SECURE
6	HTTP	HYPERTEXT TRANSFER PROTOCOL
7	URL	UNIFORM RESOURCE LOCATOR
8	NSIS	NULLSOFT SCRIPTABLE INSTALLSYSTEM
9	CMD	COMAND PROMPT
10	GUI	GRAPHICAL USER INTERFACE
11	RAM	RANDOM ACCESS MEMORY
12	ROC	RECEIVER OPERATING CHARACTERISTIC
13	AUC	AREA UNDER THE CURVE
14	UI	USER INTERFACE
15	CPU	CENTRAL PROCESSING UNIT
16	EXCEL	ELECTRONIC SPREADSHEET FORMAT
17	PDF	PORTABLE DOCUMENT FORMAT

CHAPTER 1

INTRODUCTION

1.1 CYBERSECURITY

In recent years, networks have evolved from a mere means of communication to a present computational infrastructure. Networks have become larger, faster, and highly dynamic. The pervasive use of computer and network technologies in all walks of life has turned cybersecurity issues into national security issues. sequence, cyber-attacks against apparently “non-critical” services may produce unforeseen side effects of devastating proportions. First of all, the Malware is a malicious technique which are created by any software programmer with an intention to cause damage to a computer or server. The malware does the damage after it is implemented or introduced in some way into a target’s computer and can corrupt the data of target’s computer. Generally, viruses are sent mostly through emails as attachment files. When someone opens a mail, it appears to be similar to trusted companies or friend’s mail, attachment or through link. It has become very easy to compromise an email account in modern world. Mostly the attachment would be like a document, picture etc. as soon as the clicks it the attacker gets a connection with victim. Victim notices that the media attached was of some kind of irrelevant or blank and closes the concerned window. A virus is computer program which can copy itself and infect a system. The virus is created in such a way that it sends similar mails to others nodes in network. The virus injected in to the victim’s system does the task as programmed by the attacker like compromising personal, company’s details, manipulation of system configuration etc. The problem to be examined involves the high spreading rate of computer malware (viruses, worms, Trojan horses, rootkits, botnets, backdoors, and other malicious software) and conventional signature matching-based antivirus systems fail to detect polymorphic and new, previously unseen malicious executables. Malware are spreading all over the world through the Internet and are increasing day by day, thus becoming a serious threat. The manual heuristic inspection of static malware analysis is no longer considered effective and efficient compared against the high spreading rate of malware.

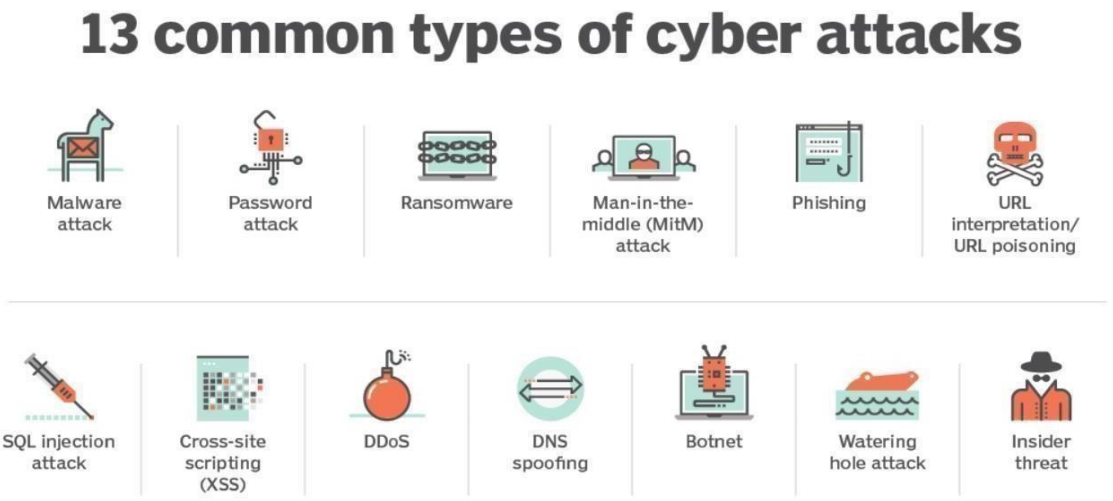


Fig 1.1: Most common Cyberattacks

The manual heuristic inspection of static malware analysis is no longer considered effective and efficient compared to the high spreading rate of malware. Nevertheless, researchers are trying to develop various alternative approaches to combating and detecting malware.

1.2 MACHINE LEARNING

One proposed solution is to use behavior malware analysis combined with data mining tasks such as machine learning classification techniques to achieve effectiveness and efficiency in detecting malware. With machine learning, cybersecurity systems can analyse pattens and learn from them to help prevent similar attacks and respond to changing behaviours. It can help cybersecurity teams be more proactive in detecting threats and responding to active attacks in real time. It can reduce the amount of time spent on routine tasks and enable organizations to use their resources more strategically. In short, machine learning can make cybersecurity simpler, more proactive, less expensive, and far more effective. But it can only do those things if the underlying data that supports the machine learning provides a complete picture of the environment. As they say, "garbage in, garbage out. Machine learning is about developing patterns and manipulating those patterns with algorithms. In order to develop patterns, you need a lot of rich data from everywhere because the data needs to represent as many potential outcomes from as many potential scenarios as possible. It’s not just about

the quantity of data; it’s also about the quality. In this project, I will discuss how an attacker creates and send his malware or trojans using different social engineering techniques, and how python is used to create a keyloggers and how a normal user should get protected from these types of payloads. A Trojan is a program which contains malicious or harmful code wrapped with apparently harmless programming or data in such a way that it can enter the victim’s computer undetected, providing the attacker unrestricted access to the data stored on that computer and causing immense damage to the victim. Trojans have the capability to replicate, spread and get activated upon certain predefined actions performed by the victim. With the help of a Trojan, an attacker gets access to the victim’s computer resources, stored passwords and it would enable him/her to read personal documents, delete important files or the whole drive, display pictures, and/or show messages on the screen. For example, a user downloads a music file or a video from the internet, but when he/she runs it, it triggers a dangerous program that may erase the user’s disk or send his/her credit card numbers to a stranger. In another aspect, a victim may also be used as an intermediary to launch attacks on others without letting the victim know about this.

1.3 MALWARE AND ITS TYPES

1.3.1 TROJAN

- Monitoring user behavior through keyloggers: Keyloggers can be used to record every keystroke made by a user, including passwords and other sensitive information.
- Display monitoring: Attackers can remotely monitor and view the display of a targeted system, allowing them to see everything that is being displayed on the victim's screen.
- Accessing confidential information: Trojans can be used to access confidential information stored on a system, such as login credentials, financial data, or other sensitive data.
- Activating a system's webcam and recording video: Attackers can remotely activate a system's webcam and record video of the user without their knowledge.
- Taking screenshots: Trojans can be used to take screenshots of a system's display,allowing the attacker to see what the user is doing on their computer.

- Distributing viruses and other malware: Trojans can be used to distribute viruses and other malware to other systems on the network, causing further damage and spreading the infection.
- Getting full control on browser, CMD, and file manager: Attackers can gain full control over the browser, Command Prompt, and file manager, allowing them to execute arbitrary commands on the system.
- Formatting drives: Attackers can use Trojans to format hard drives, erasing all dataon the targeted system.
- Deleting, downloading, or altering files and file systems: Trojans can be used to delete, download, or alter files and file systems on a targeted system, causing significant damage to the victim's data and system.

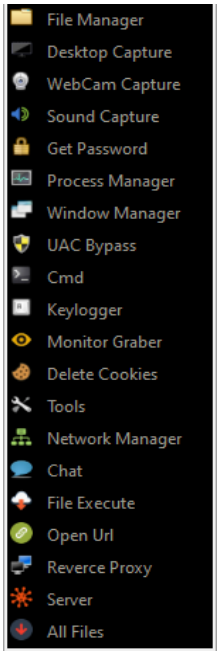


Fig 1.2: Options in a Trojan

1.3.2 KEYLOGGER

Coming to keylogger Keyloggers are a type of monitoring software designed to record keystrokes made by a user. Data captured by keyloggers can be sent back to attackers via email or uploading log data to predefined websites, databases, or FTP servers. A keylogger is a type of malicious software that is designed to record every keystroke made on a computer or mobile device keyboard, including passwords, credit card numbers, sensitive personal information, and other confidential data. The attacker can use this information to steal your identity,

access your financial accounts, or perform other nefarious activities. In some cases, keyloggers are installed as part of a larger attack, such as a phishing scam or a malware infection. Once installed, the keylogger can silently monitor all keyboard activity, without the user even knowing that their actions are being recorded. The pynput library in Python allows attackers to control and monitor the keyboard on a targeted machine remotely. With this tool, the attacker can listen to every keystroke, and collect sensitive information from the victim's computer. This information is then sent to a remote server, where it can be accessed by the attacker. Cybercriminals who use keyloggers are often highly skilled and well-funded, and they can use this stolen information to commit a wide range of crimes. They might use it to steal money from bank accounts, make unauthorized purchases, or even impersonate the victim to gain access to other sensitive information.

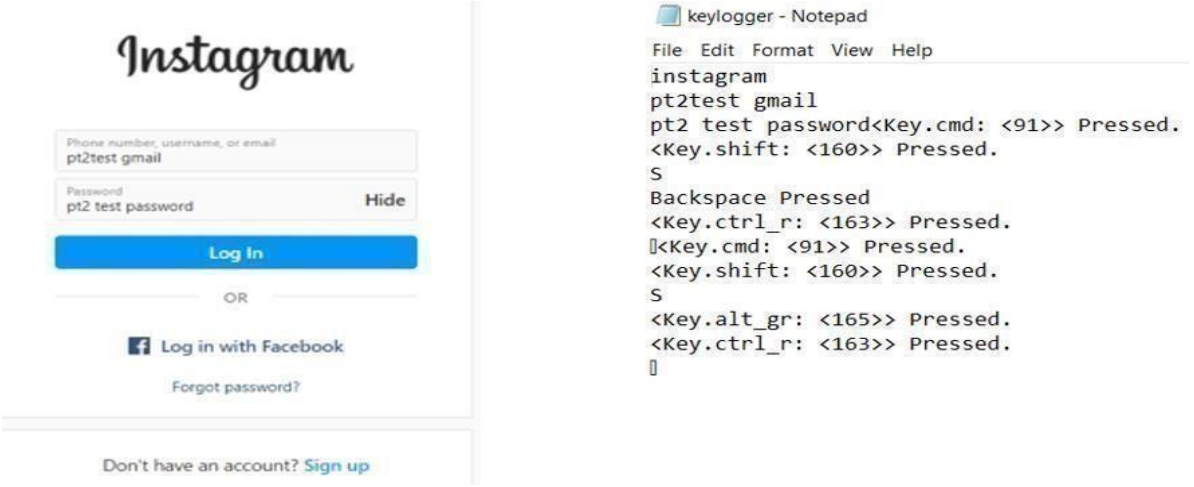


Fig 1.3: Working of keylogger

1.3.3 SOCIAL ENGINEERING

Social engineering is a sophisticated and cunning form of cyber-attack that has been around for decades. It is a type of attack that relies on exploiting human behavior, rather than technical vulnerabilities, to gain access to sensitive information or systems. By using psychological manipulation techniques, attackers can trick users into giving away valuable data or making security mistakes that can be used to gain unauthorized access to networks, computers, and other devices. There are different types of social engineering attacks, and they can take many forms. Some of the most common tactics used by attackers include phishing,

pretexting, baiting, and spear-phishing. In a phishing attack, for example, attackers will send emails or messages that appear to be from a legitimate source, such as a bank or a social media platform. The message will typically ask the recipient to click on a link or download an attachment, which can then install malware or steal sensitive information. In a pretexting attack, the attacker will create a false pretext or scenario to trick the victim into divulging sensitive information. For example, an attacker may pretend to be a technical support representative and call a victim claiming that there is a problem with their computer that requires access to sensitive data. Baiting attacks, on the other hand, involve offering something of value to a victim in exchange for their sensitive information or access to their systems. This can include offering a free software download or a USB drive with malicious code installed. Finally, spear-phishing attacks are targeted attacks that are designed to trick a specific individual or group of individuals. The attacker will typically use information that they have gathered through social media or other sources to create a personalized message that is more likely to be effective. In conclusion, social engineering attacks can be extremely dangerous and damaging to individuals and organizations alike. It is important for users to be aware of the different types of attacks and to take appropriate measures to protect themselves, such as using strong passwords, avoiding suspicious emails or messages, and keeping their software up to date. By remaining vigilant and informed, users can help to reduce the risk of falling victim to these insidious attacks.

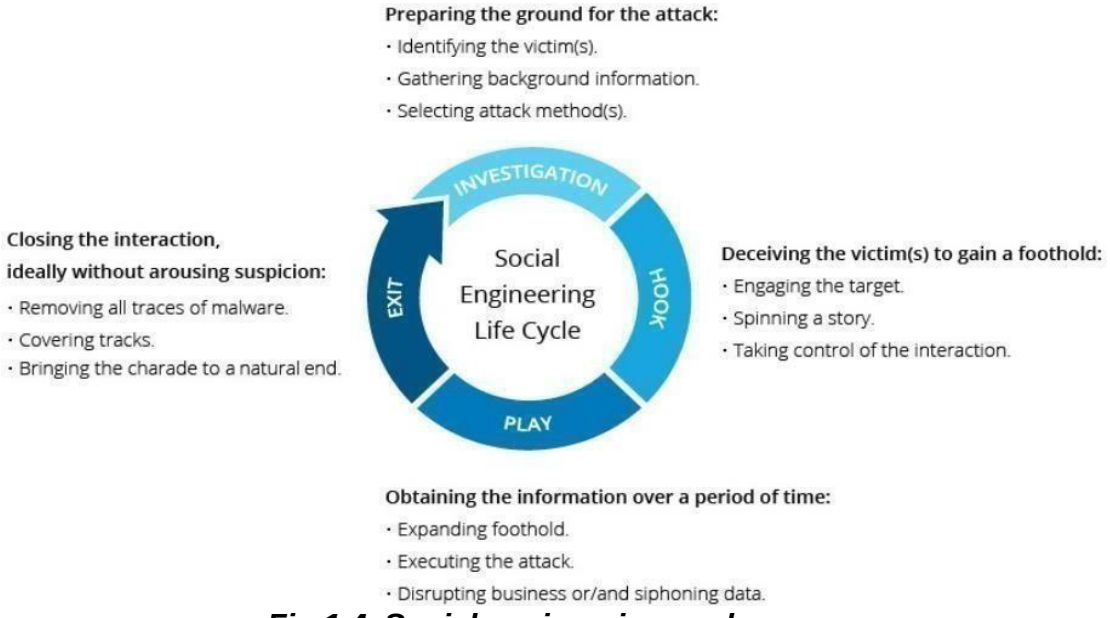


Fig 1.4: Social engineering cycle

CHAPTER 2

LITERATURE REVIEW

1. Research on technology of trojan horse detection

Yu, W., Yalin, Y., & Haodan, R. (2019, October). Research on the technology of trojan horse detection. In 2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA) (pp. 117-119). IEEE.

- In this paper, the characteristics and principles of trojans are analyzed and the detection methods are compared
- This paper includes detection methods like
- Sandbox testing
- Heuristic-based testing.

2. Review of Signature-based Techniques in Antivirus Products

Al-Asli, M., & Ghaleb, T. A. (2019, April). Review of signature-based techniques in antivirus products. In 2019 International Conference on Computer and Information Sciences (ICCIS) (pp. 1-6). IEEE.

- This paper revisits existing research on virus detection using signature-based algorithms
- hybridization of cybercrime investigation models with existing antivirus products to make an extension to their benefits to the entire community.

3. Comprehensive Review on Malware Detection Approaches

Aslan, Ö. A., & Samet, R. (2020). A comprehensive review on malware detection approaches. IEEE Access, 8, 6249-6271.

- This paper briefs different malware detection techniques
- Evolution of malware detection and the history of malware discussed in this paper

4. Malware Detection Techniques

Idika, N., & Mathur, A. P. (2007). A survey of malware detection techniques. PurdueUniversity, 48(2), 32-46.

- In this paper, they discussed what malware is followed by thecategories of malware

- The paper includes future malware threats and techniques

5. cyber security at a glance

Asish, M. S., & Aishwarya, R. (2019, March). Cyber security at a glance. In 2019 Fifth International Conference on Science Technology Engineering and Mathematics(ICONSTEM) (Vol. 1, pp. 240-245). IEEE.

- This paper described various hacking techniques and their countermeasures invarious aspect
- Learned about different types of malware and after-effects of the installation of that malware
- Discussed all kinds of malware from old to new to describe the evolution of malware
- Explained how Classical Defense mechanisms (like signature-based malware detection) used by anti-virus will fail to cope up new age malware challenges.

6. Malware detection using machine learning and deep learning

Rathore, H., Agarwal, S., Sahay, S. K., & Sewak, M. (2018, December). Malware detection using machine learning and deep learning. In International Conference onBig Data Analytics (pp. 402-411). Springer, Cham.

- In this paper, they have modeled malware analysis and detection as machine learning and deep learning problem.
- They used best practices in building these models (like cross-validation, fixingclass imbalance problems, etc.).

7. The world of Malware: An overview

Namanya, A. P., Cullen, A., Awan, I. U., & Disso, J. P. (2018, August). The world of Malware: An overview. In 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud) (pp. 420-427). IEEE.

- This paper presents an overview of the world of malware with the intent of providing the underlying information for the intended study into developing malware detection approaches.
- This paper reviews the foundational information of malware and anti-malware systems.
- They presented summaries of works found in literature about malware

evolution, malware analysis techniques, malware evasion techniques and existing malware detection methods.

8. Keyloggers: silent cyber security weapons

Bhardwaj, A., & Goundar, S. (2020). Keyloggers: silent cyber security weapons. Network Security, 2020(2), 14-19.

- Explained the destruction caused by keyloggers in past and how advance theybecame
- Keyloggers aka silent cyber weapon is deadly and undetectable in most cases
- Described the economic damage caused by keyloggers
- The privilege level at which keyloggers execute is higher than typical malware,which makes them almost impossible to detect and remove.

9. Identification and prevention of social engineering attacks on an enterprise

Parthy, P. P., & Rajendran, G. (2019, October). Identification and prevention of social engineering attacks on an enterprise. In 2019 International Carnahan Conference on Security Technology (ICCST) (pp. 1-5). IEEE.

- This paper classifies the various social engineering attacks based on the perspective of an attacker.
- Explained about all types of enterprise attacks and classified them
- Many new social engineering techniques were discussed likes reverse social engineering.
- This paper helps the reader to gain insight into how social engineering can be used against enterprises.

10. Social media: A new vector for cyber attack

Kunwar, R. S., & Sharma, P. (2016, April). Social media: A new vector for cyber-attack. In 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring) (pp. 1-5). IEEE.

- This paper provides an in-depth detail of threats, security risks and different types of attacks using social media.
- Discussed many things about spam and malicious ads in social media
- Finding detailed information about the number of users who uses social media

and how many are lured into these cyber-attacks gave a clear ratio of much more information.

11. The strange world of Keyloggers - an overview, Part I

Creutzburg, Reiner (2017). "The strange world of keyloggers - an overview, Part I". Electronic Imaging. 2017 (6): 139–148.

- provided a summary of the relevant hard-, software, and mobile keyloggers that are available and in use, as well as a bibliographic overview of keyloggers. Keyloggers' capabilities, accessibility, and detection potential are examined and detailed.

12. A Novel Approach of Unprivileged Keylogger Detection.

Wajahat, A., Imran, A., Latif, J., Nazir, A., & Bilal, A. (2019). A Novel Approach of Unprivileged Keylogger Detection. 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies(iCoMET).doi:10.1109/icomet.2019.8673404.

- This research focused on detecting the most common indigent user space keylogger. They demonstrated code snippets in this study that allow the client to deal with keylogger spyware without jeopardizing security.

13. A Comprehensive Study on Malware Detection and Prevention Techniques used by Anti-Virus.

Rohith, C., & Kaur, G. (2021, April). A Comprehensive Study on Malware Detection and Prevention Techniques used by Anti-Virus. In 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM) (pp. 429-434). IEEE.

- The purpose of this study is to describe and debate the cutting-edge technologies employed by anti-virus. They talked about how malware that lives on a system might permanently destroy its hardware components.

14. Cybersecurity Analytics to Combat Cyber Crimes

Nallaperumal, K. (2018). CyberSecurity Analytics to Combat Cyber Crimes. 2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). doi:10.1109/iccic.2018.8782430

- This paper gives an introduction to the cyber-crimes and their impacts.

15. **Design and Analysis of Machine Learning Based Technique for Malware Identification and Classification of Portable Document Format Files**

Alshamrani, S. S. (2022). Design and Analysis of Machine Learning Based Technique for Malware Identification and Classification of Portable Document FormatFiles. Security & Communication Networks, 2022.

- This study presents a Machine Learning (ML) model, which can recognize JavaScript and malicious API call assaults in PDF files.

2.1 INFERENCE FROM LITERATURE SURVEY

The review provides a comprehensive overview of the history of malware and its detection methods, highlighting the evolution of cyber threats over time. It emphasizes the need for effective detection methods that can keep up with the rapidly changing cybersecurity landscape. The literature review discusses the pros and cons of various detection methods, including signature-based, heuristic-based, behavior-based, and machine learning-based methods. It emphasizes the need for a multi-layered approach to cybersecurity, where multiple detection methods are used in conjunction to provide a comprehensive defense against cyber threats. The review also discusses new cyber threats like ransomware, crypto-jacking, and phishing, which have emerged as significant threats in recent years. It highlights the need for continuous research and development in cybersecurity to keep pace with the evolving threats and develop new detection methods that can effectively detect and mitigate these threats. Furthermore, the review emphasizes the significance of social engineering techniques used by attackers to exploit human vulnerabilities, emphasizing the importance of cybersecurity awareness training. It also discusses the role of artificial intelligence and machine learning in detecting and preventing cyber threats.

The review provides an understanding of the importance of vulnerability assessment and penetration testing in identifying security gaps and addressing them before attackers can exploit them. It also emphasizes the need for end-to-end encryption and secure communication channels to protect sensitive data. Additionally, the literature review highlights the importance of backup and disaster

recovery plans in mitigating the damage caused by cyber-attacks. It also discusses the significance of cybersecurity insurance in providing financial protection to organizations in case of a cyber-attack. The review provides insights into various compliance regulations like GDPR and HIPAA, emphasizing the need for organizations to comply with these regulations to avoid legal repercussions. It also discusses various cybersecurity frameworks like NIST and ISO 27001, which can help organizations establish a robust cybersecurity posture. The literature review underscores that cybersecurity is not just about technology, but it also involves people and processes. Organizations must develop a comprehensive approach to cybersecurity that includes training employees, implementing policies and procedures, and investing in the latest technologies to protect their systems from cyber threats. Moreover, a multi-layered approach to cybersecurity is necessary to mitigate the risks posed by cyber threats effectively. This approach involves using multiple detection methods in conjunction to provide a comprehensive defense against cyber-attacks. It is also essential for organizations to prioritize cybersecurity and invest in continuous research and development to keep pace with the evolving threats. As new threats emerge, organizations must be proactive in identifying and mitigating them before they cause significant damage.

In conclusion, the literature review highlights the need for a holistic approach to cybersecurity that involves people, processes, and technology to effectively mitigate the risks posed by cyber threats. Organizations that prioritize cybersecurity and implement a multi-layered approach will be better equipped to protect their sensitive data and maintain business continuity.

CHAPTER 3

ARCHITECTURE DESIGN AND PROPOSED SYSTEM

Malware is a term used to describe any software that is intentionally designed to cause harm or damage to a computer system or network. Malware can be created and distributed by cybercriminals using various social engineering techniques to trick users into downloading and installing the malicious software. One of the most common methods of distributing malware is through phishing emails. Cybercriminals will send emails that appear to be from a legitimate source, such as a bank or a social media platform, and will include a link or attachment that, when clicked, will download the malware onto the victim's computer. Another method is through the use of malicious websites. Cybercriminals will create fake websites that appear to be legitimate, such as online shopping sites or banking sites, and will trick users into entering their login credentials or downloading a file, which is actually malware. Malware can also be distributed through social media platforms. Cybercriminals will create fake profiles or pages and will post links or attachments that, when clicked, will download malware onto the victim's device.

Once themalware is installed on the victim's computer or network, it can perform a range of malicious activities, such as stealing sensitive information, encrypting files and demanding a ransom, or using the victim's computer to launch attacks on other systems. To detect and prevent malware, cybersecurity products use machine learning algorithms to analyze pre-execution and post-execution phase data. In the pre-execution phase, the malware recognition module analyzes the file's format, code, binary data statistics, and text strings to determine if it is likely to be malicious. In the post-execution phase, the module analyzes the behavior and events caused by the processing activity of the software to determine if it is behaving in a malicious manner. In summary, malware is created and distributed using various social engineering techniques, and to detect and prevent malware, cybersecurity products use machine learning algorithms to analyze pre-execution and post-execution phase data.

3.1 WHAT IS A TROJAN

A Remote Access Trojan (RAT) is a type of malware that allows an attacker to remotely access and control a victim's computer or network. It is a type of backdoor malware that can give the attacker complete control over the infected system, allowing them to steal sensitive data, install additional malware, and even use the victim's computer as part of a larger botnet for launching attacks on other systems. RATs typically enter a victim's system through malicious email attachments, phishing scams, or by exploiting vulnerabilities in software or operating systems. Once installed, the RAT will establish a connection with the attacker's command and control (C&C) server, enabling the attacker to remotely control the victim's computer.

3.2 TROJAN WORKING

Trojan viruses work by taking advantage of a lack of security knowledge by the user and security measures on a computer, such as an antivirus and antimalware software program. A Trojan typically appears as a piece of malware attached to an email. The file, program, or application appears to come from a trusted source. Trojans create a virtual “backdoor” to a computer that allows hackers remote access to the computer. As such, hackers can download user data and easily steal it. **Even worse, a backdoor allows a attacker to upload additional malware to the device.**

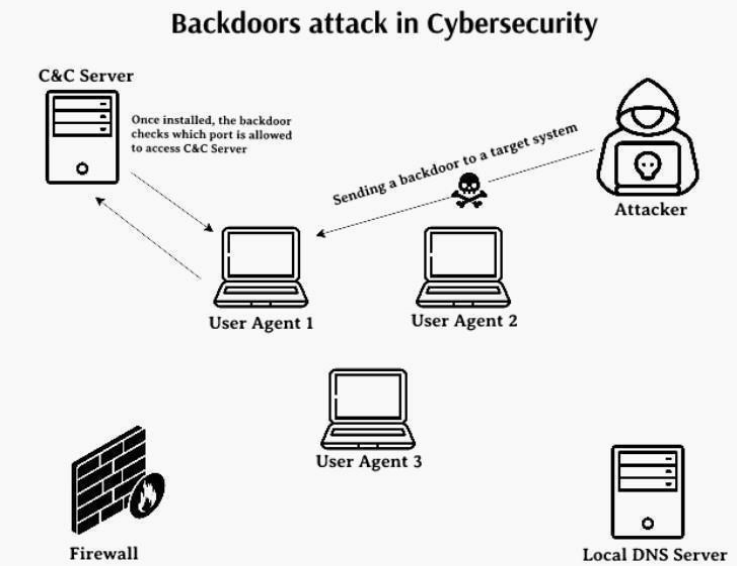


Fig 3.1: Back door in malware

3.3 CAPABILITIES OF TROJAN

Trojan viruses can carry out a wide range of malicious activities on a computer system. Here are some of the most common activities that Trojans can perform: Monitoring user behavior through KEYLOGGERS: Some Trojans have keylogger functionality, which allows them to record a victim's keystrokes and monitor their activity on the computer. This can give attackers access to sensitive information such as login credentials and other confidential information. Display monitoring: Some Trojans can monitor a victim's screen activity, capturing screenshots and recording video. This allows attackers to see everything the victim does on their computer, including sensitive information such as banking details and login credentials. Accessing confidential information: Trojans can be used to steal confidential information, such as credit card numbers and social security numbers, from a victim's computer. Distributing viruses and other malware:

Once a Trojan has infected a victim's computer, it can be used to distribute other malware or viruses to other systems, creating a larger network of infected devices. Getting full control on browser, CMD, file manager: Some Trojans can give attackers full control over a victim's computer, including control over their browser, command prompt, and file manager. Formatting drives: Trojans can be used to format drives, destroying all data on the drive and rendering it unusable. Deleting, downloading, or altering files and file systems: Trojans can be used to delete, download, or alter files and file systems on a victim's computer, causing significant damage to the system and potentially exposing sensitive information. In addition to having up-to-date antivirus and antimalware software, there are other measures that individuals and organizations can take to protect against Trojans. One effective measure is to maintain a robust firewall that blocks unauthorized access to your computer or network. Regularly updating your operating system and software is also important as it helps to patch vulnerabilities that attackers may use to exploit your system. It is also advisable to use strong passwords and enable two-factor authentication on all accounts to prevent unauthorized access. It's also essential to be aware of phishing scams and other social engineering tactics that attackers use to trick users into divulging sensitive information. online can go a long way in protecting your privacy and security.

3.4 SILENT KEYLOGGERS

A keylogger is a type of monitoring software that records every keystroke made by a user on a computer or mobile device. Here are some key points to understand about keyloggers:

Purpose: Keyloggers are typically used by cybercriminals to steal sensitive information, such as login credentials and financial data. They can also be used by employers to monitor employee activity on company computers.

Data capture: Keyloggers capture all keystrokes made by the user, including usernames, passwords, and other sensitive information. Some keyloggers can also capture screenshots, record audio, and track internet activity.

Data transmission: Data captured by keyloggers can be sent back to the attacker via email or uploaded to predefined websites, databases, or FTP servers. If the keylogger is bundled within a larger attack, an attacker might simply remotely log into a machine to download keystroke data.

Detection: Detecting a keylogger can be difficult, as many are designed to operate silently in the background. However, some antivirus and antimalware programs can detect and remove keyloggers.

Backdoors and port forwarding are commonly used techniques by attackers to gain unauthorized access to a victim's machine. A backdoor is essentially a secret entry point into a system that is not visible to the regular user. Attackers can use these backdoors to gain access to a system without being detected by security measures. Backdoors can be created intentionally by software developers, such as in the case of remote access tools that are used for legitimate purposes, but they can also be installed by attackers as a means of maintaining access to a compromised system. Port forwarding, on the other hand, is a technique that allows attackers to redirect network traffic from one port to another, usually on their machine. This can be done by modifying network settings on the victim's machine, or by compromising network infrastructure such as routers. Once the attacker has redirected the traffic, they can then intercept and analyze it to gain sensitive

information or control over the victim's machine. Remote access trojans (RATs) often use backdoors and port forwarding to maintain a constant connection with the victim's machine. By doing so, attackers can perform a range of malicious activities such as stealing sensitive information, installing other malware, or even taking full control of the victim's machine. It is important for users to be aware of these techniques and to take appropriate measures to prevent unauthorized access to their systems, such as using strong passwords and keeping their software and security measures up to date.

Keyloggers are a type of malicious software that can record keystrokes on a keyboard without the user's knowledge or consent. These keystrokes can include sensitive information such as passwords, credit card numbers, and personal messages. Once captured, the information can be used for nefarious purposes such as identity theft, financial fraud, or espionage. One of the best ways to protect against keyloggers is by using reputable antivirus and antimalware software. These software programs can detect and remove keyloggers from a computer system. It is also important to keep the software updated with the latest security patches and updates to ensure maximum protection. Another way to protect against keyloggers is by being cautious when entering sensitive information on public or unfamiliar computers. It is recommended to avoid using public computers or unsecured Wi-Fi networks when accessing sensitive information. When using a public computer, it is advisable to use a virtual keyboard to input sensitive information instead of a physical keyboard. Additionally, it is essential to avoid suspicious links and downloads that may contain keyloggers. Emails from unknown senders, pop-ups, and attachments from unknown sources should be avoided to prevent downloading malicious software. By taking these precautions, users can significantly reduce the risk of falling victim to keyloggers and protect their sensitive information

Most antivirus and antimalware software can detect and remove many types of keyloggers, but there are some keyloggers known as "silent keyloggers" that can evade detection. Silent keyloggers are designed to run undetected in the background and can be difficult to detect using traditional antivirus methods. These keyloggers can capture sensitive information, including usernames, passwords,

and credit card numbers, without the user's knowledge. To protect against silent keyloggers, users should use reputable antivirus and antimalware software, keep their software up-to-date, and avoid downloading software or clicking on links from unknown sources. It's also important to be cautious when entering sensitive information on public or unfamiliar computers and to use virtual keyboards when possible. It is important to protect against keyloggers by using reputable antivirus and antimalware software, avoiding suspicious links and downloads, and being cautious when entering sensitive information on public or unfamiliar computers.

3.5 HOW TO CONVERT PYTHON SCRIPT TO AN EXE?

I utilized the Nullsoft scriptable install system (NSIS) to convert a Python script containing a malware payload into an executable application. The purpose of this conversion was to make it easier to deliver the malware to unsuspecting users through various social engineering techniques. To convert the script into an executable, I first compressed the Python script into a zip file, which was then used by NSIS to create the executable file. This process helped to keep the malware payload hidden and undetectable by antivirus software and other security tools. Once the executable file was created, I could distribute itto targets disguised as a legitimate application or software update. When the target downloads and runs the file, the malware payload is executed, and it can carry out its malicious activities, such as stealing sensitive data or providing unauthorized access to the victim's system.

In conclusion, the use of NSIS to create executable files from Python scripts is a technique that can be used by attackers to deliver malware to unsuspecting targets. Users need to remain cautious when downloading and executing any executable files from unknown sources to avoid falling prey to such attacks.

Fig 3.2 NSIS

3.6 ANTIVIRUS DETECTION METHODS (POST-INSTALLATION METHODS)

The project discusses various post-detection methods used by anti-virus systems to prevent the spread of malware and mitigate the damage caused by cyber-attacks. In anti-virus software, various types of scans can be scheduled to run at different intervals. These include full scans, targeted scans, and quick scans.

A full scan is a comprehensive scan that examines all files and directories on the system for any signs of malware. It is a time-consuming process that can take several hours to complete. Full scans are typically scheduled to run regularly, such as once a week or once a month, to ensure that the entire system is thoroughly scanned for malware.

A **targeted scan**, on the other hand, is a scan that is focused on a specific area of the system, such as a particular file or folder. This type of scan is useful when there is a suspicion of malware in a specific location. Targeted scans are typically quicker than full scans and can be scheduled to run as needed.

A **quick scan** is a fast scan that examines only critical areas of the system, such as the system files and running processes. Quick scans are useful for detecting and removing malware that may be actively running on the system. They are typically scheduled to run more frequently than full scans or targeted scans, such as once a day or once every few hours.

schedule scans to ensure that the system is protected from malware. The frequency of scans depends on the usage and vulnerability of the system. For example, if the system is used for sensitive data, then more frequent scans may be necessary to ensure the highest level of protection. It is also important to ensure that the anti-virus software is up to date to ensure the latest virus definitions are being used during the scan.

.

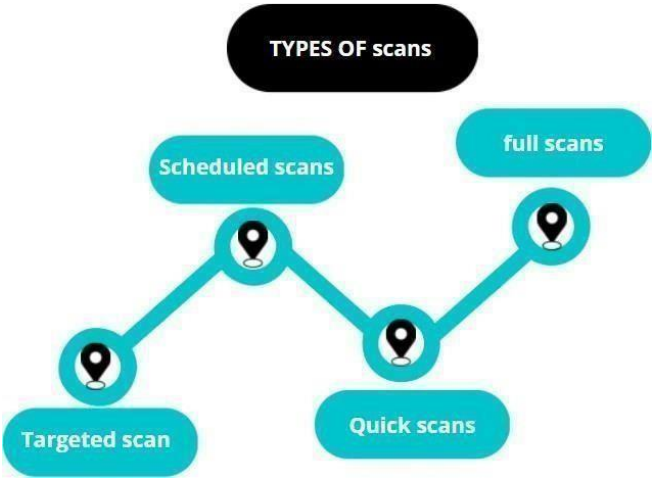


Fig 3.3: Types of scanning in an antivirus

Signature-based detection: Signature-based detection is commonly used. A method in antivirus software to detect and prevent the spread of malware. It works by identifying the unique binary patterns, or signatures, associated with known malware strains. When a file is scanned by an antivirus program, it checks the file's signature against a large database of known virus signatures. If the signature matches a known malware pattern, the antivirus software will immediately alert the user and take appropriate action to quarantine or remove the file. The advantage of signature-based detection is its ability to quickly and accurately identify known malware strains. It is a tried and tested method that has been used by antivirus software for decades, and it is still effective against many common forms of malware. However, signature-based detection has limitations. It is only effective against known malware strains, which means that it may not detect new or unknown threats. In addition, malware creators can easily modify the signature of their malware to evade detection by antivirus software. As a result, modern antivirus software often employs a combination of signature-based detection and other methods, such as behavioral analysis and machine learning algorithms, to provide more comprehensive protection against malware.

Heuristic-based detection: is a type of malware detection method that relies on analyzing the behavior and patterns of code to identify potential threats. Unlike signature-based detection, which relies on a database of known virus signatures, heuristic-based detection is designed to detect new and unknown malware strains.

When a file is scanned using heuristic-based detection, the antivirus software analyzes the code for any suspicious behavior or patterns that are indicative of malware. For example, the software might look for code that attempts to modify system settings, access sensitive data, or communicate with external servers. If the antivirus software identifies any suspicious behavior, it may run the code in a runtime virtual environment to test it further. This allows the software to observe the behavior of the code without risking damage to the user's system. If the code is determined to be malicious, the antivirus software will take appropriate action to quarantine or remove the file. The advantage of heuristic-based detection is its ability to detect new and unknown malware strains that may not be included in traditional signature-based databases. This method is particularly useful for identifying zero-day exploits and other new forms of malware that have not yet been identified. However, heuristic-based detection can also produce false positives, as legitimate code may sometimes exhibit behavior that appears suspicious. To address this, antivirus software typically includes a range of configurable sensitivity settings that allow users to adjust the level of heuristic analysis that is applied to their files.

Behavioral-based detection. Behavioral-based detection is a malware detection method that relies on monitoring the behavior of software and programs running on a computer system. This approach focuses on identifying suspicious behavior that deviates from the normal operations of software applications. In behavioral-based detection, antivirus software continuously monitors the activity of all programs running on a computer system. This monitoring allows the software to detect any behavior that is outside the normal range of activity for a particular application. For example, if a program suddenly starts asking for additional read and write permissions or starts attempting to modify critical system files, the antivirus software will identify this behavior as suspicious and may issue a warning to the user. Similarly, if a program attempts to communicate with external servers or perform other unusual actions, the antivirus software may flag it as potentially malicious. One of the advantages of behavioral-based detection is its ability to identify new and previously unknown threats that may not be caught by traditional signature-based or heuristic-based detection methods. Because this approach focuses on the behavior of software rather than specific code patterns or

signatures, it can detect new and evolving threats in real time. However, behavioral-based detection can also produce false positives, as some legitimate software may exhibit behavior that appears suspicious. To address this, antivirus software typically includes configurable sensitivity settings that allow users to adjust the level of monitoring and analysis that is applied to their system.

Sandbox detection: Sandbox detection is a technique used by antivirus programs to analyze and test suspicious files or programs in a virtual machine environment. The virtual machine or sandbox is an isolated and secure environment that allows the file or program to run without affecting the actual system. When an antivirus program is unable to determine whether a file is malicious or not, it may use sandbox detection to observe the behavior of the file. If the file performs any malicious activities, such as attempting to modify system files or accessing sensitive data, the antivirus program can quickly detect it and flag it as a threat. Sandbox detection is an effective way to detect unknown malware that may not be included in the antivirus database. By analyzing the behavior of the file, it can identify new types of malware and create a signature that can be added to the antivirus database, allowing it to detect and block future infections. Overall, sandbox detection is an essential feature of modern antivirus programs and plays a vital role in keeping computer systems protected from new and emerging threats.

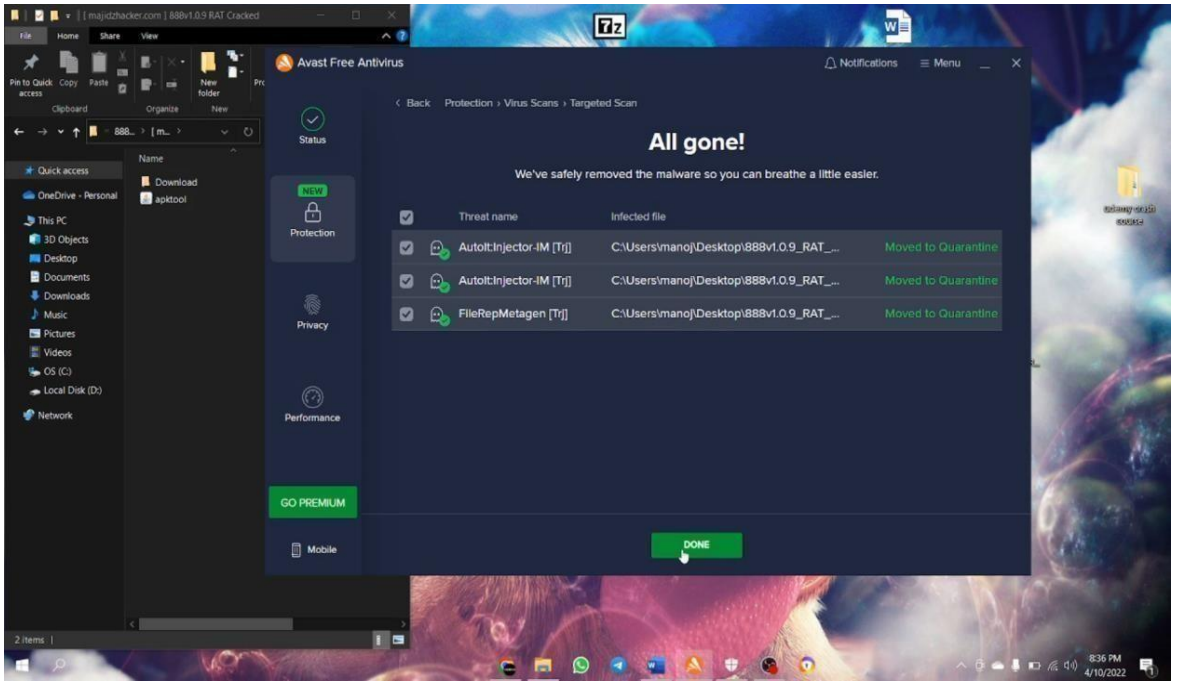


Fig 3.4: Malware found and deleted

3.7 PROPOSED SYSTEM

The project's approach to developing a machine-learning model for pre-installation detection of malware is crucial in combating the growing number of malware attacks that exploit vulnerabilities in software and hardware systems. By identifying the patterns followed by malicious malware, the project can contribute to the development of new and effective cybersecurity technologies that can prevent malware infections before they occur. The use of machine learning algorithms can help identify patterns and trends in malware behavior, allowing for real-time detection and response to threats. The project also aims to develop effective prevention strategies by analyzing the patterns used by attackers to send malware to victims. These strategies can include educating users about the risks associated with opening emails or downloading software from untrusted sources. Additionally, the project can inform the development of more secure software and hardware systems that are less vulnerable to exploitation by malware and other cyber threats. In addition to the technical aspects of the project, there are ethical considerations to be taken into account. The creation and use of malware, even for research purposes, can have unintended consequences if the malware falls into the wrong hands. Therefore, the project must adhere to ethical guidelines and ensure that appropriate security measures are in place to prevent the accidental release of the malware. The project must also be transparent about its goals and methods to ensure that it does not contribute to the proliferation of malware and cyber threats.

In conclusion, the project's focus on developing a machine-learning model for pre-installation detection of malware and analyzing the patterns used by attackers to send malware to victims is critical in the ongoing fight against cyber threats. The knowledge gained from this project can contribute to the development of more secure software and hardware systems and inform the development of effective prevention strategies to protect individuals and organizations from malware infections.

3.8 ARCHITECTURE DIAGRAMS

PHASE 1

Phase 1 of the project involves creating a trojan and using social engineering techniques to deliver it to a targeted device. The trojan is designed to be a program that appears useful or interesting to the victim but contains hidden code that can perform malicious actions on their device. Once the trojan has been created, it is bound with a payload and forwarded to an IP address and port. This allows the attacker to remotely control the infected device and perform a variety of malicious actions on it. The trojan is then delivered to the targeted device using social engineering techniques, which involve tricking the victim into downloading and installing the trojan. This can be done through a variety of methods, such as sending an email with a malicious attachment or creating a fake website that appears to be legitimate but contains the trojan. Once the trojan has been installed on the targeted device, the attacker can begin controlling it remotely. This allows them to perform a variety of malicious actions, such as stealing sensitive data, installing additional malware, or taking control of the device. Overall, phase 1 of the project involves creating a trojan, binding it with a payload, delivering it to a targeted device using social engineering techniques, and then controlling the device remotely.

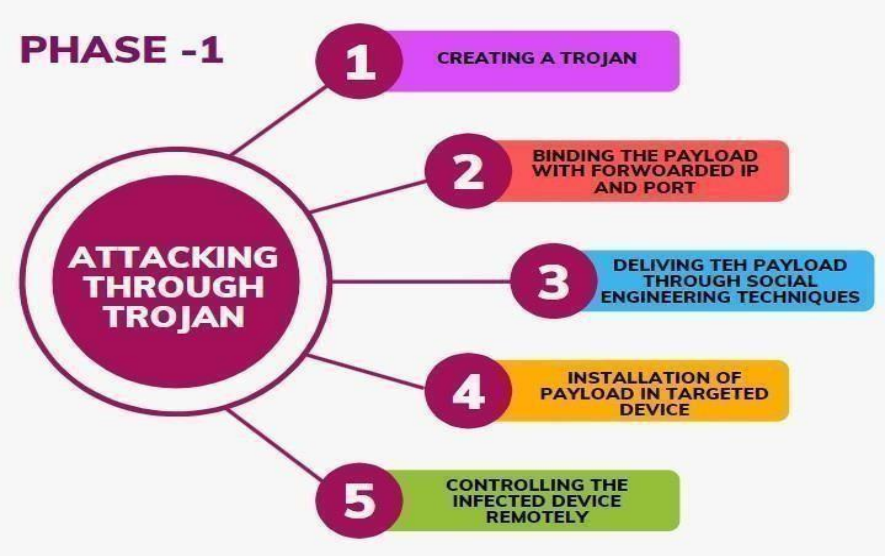


Fig 3.5: Phase 1 architecture

PHASE 2

Phase 2 of the project involves creating a keylogger using the Python library pynput. A keylogger is a type of malware that records every keystroke made on a device and sends the data back to the attacker. This allows the attacker to steal sensitive information such as passwords, credit card numbers, and other confidential data. The keylogger is created using the pynput library in Python, which allows the developer to monitor and record keystrokes in real-time. Once the keylogger is created, the Python script is converted to an executable file using NSIS (Nullsoft Scriptable Install System) for source code hiding and better delivery. Once the keylogger is installed on the targeted device, it begins recording all keystrokes made by the user. This includes everything typed on the keyboard, such as passwords, login credentials, and other sensitive information. The keylogger then sends the recorded keystrokes back to the attacker, who can then use this information for malicious purposes. For example, the attacker may use the stolen credentials to access the victim's online accounts, steal their identity, or commit other types of fraud. Overall, phase 2 of the project involves creating a keylogger using the pynput library in Python, converting the Python script to an executable file using NSIS, and then using the keylogger to steal sensitive information from the victim's device. By understanding how keyloggers work and how they can be used to steal sensitive information, researchers can develop more effective anti-malware tools and techniques to protect users from these types of threats.

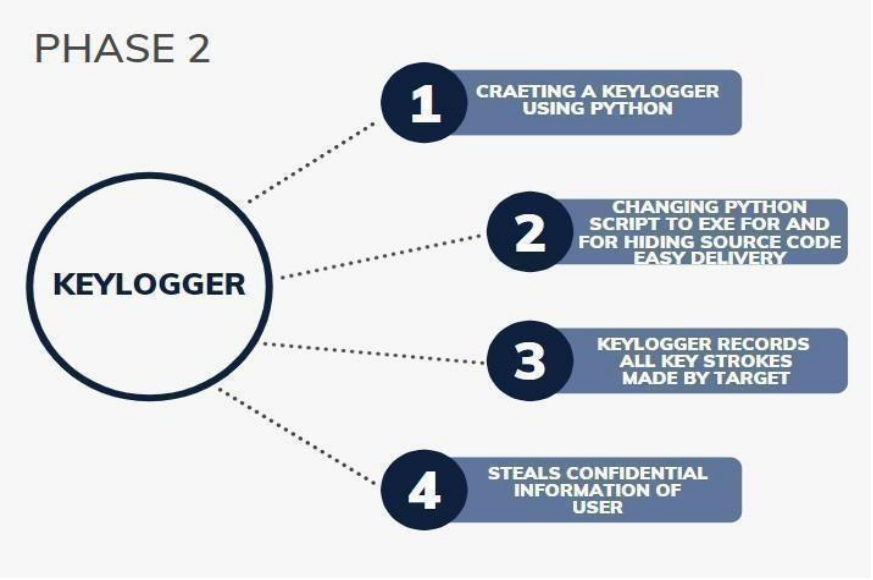


Fig 3.6: Phase 2 architecture

PHASE 3

Phase 2 of the project involves creating a keylogger using the Python library pynput. A keylogger is a type of malware that records every keystroke made on a device and sends the data back to the attacker. This allows the attacker to steal sensitive information such as passwords, credit card numbers, and other confidential data. The keylogger is created using the pynput library in Python, which allows the developer to monitor and record keystrokes in real-time. Once the keylogger is created, the Python script is converted to an executable file using NSIS (Nullsoft Scriptable Install System) for source code hiding and better delivery. Once the keylogger is installed on the targeted device, it begins recording all keystrokes made by the user. This includes everything typed on the keyboard, such as passwords, login credentials, and other sensitive information. The keylogger then sends the recorded keystrokes back to the attacker, who can then use this information for malicious purposes. For example, the attacker may use the stolen credentials to access the victim's online accounts, steal their identity, or commit other types of fraud. Overall, phase 2 of the project involves creating a keylogger using the pynput library in Python, converting the Python script to an executable file using NSIS, and then using the keylogger to steal sensitive information from the victim's device. By understanding how keyloggers work and how they can be used to steal sensitive information, researchers can develop more effective anti-malware tools and techniques to protect users from these types of threats.

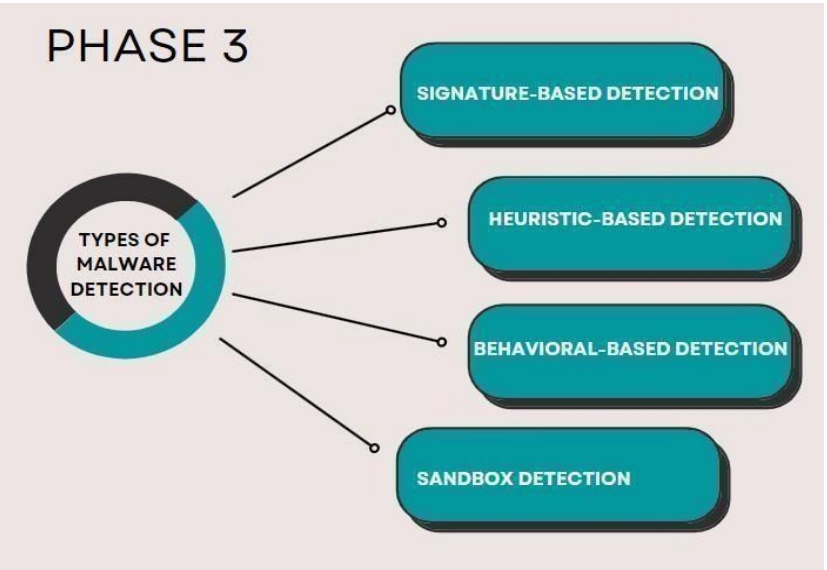


Fig 3.7: Phase 3 architecture

PHASE 4

In phase 4 of the project, the focus is on developing a machine learning model that can predict the probability of an application being malware using physical features of the application. Physical features of an application can include attributes like file size, file type, file permissions, and RAM and CPU usage etc. By analyzing these physical features, a machine learning model can learn to differentiate between malicious and benign applications. The goal of this phase is to develop a model that can accurately classify new and unknown applications as either malware or benign. This can help in detecting and preventing malware attacks before they can cause any harm to the system. To develop this machine learning model, a dataset of known malware and benign applications is required. The dataset is then split into training and testing sets. The physical features of each application in the dataset are extracted, and a machine learning algorithm is trained on the training set to learn the patterns and features that distinguish malware from benign applications. Once the algorithm is trained, it is tested on the testing set to evaluate its accuracy and performance. The model is then fine-tuned and optimized to improve its performance and minimize false positives and false negatives. Overall, developing a machine learning model for malware detection is a promising approach as it can detect new and unknown malware by analyzing physical features of an application. This can provide an additional layer of security to traditional signature-based detection methods.

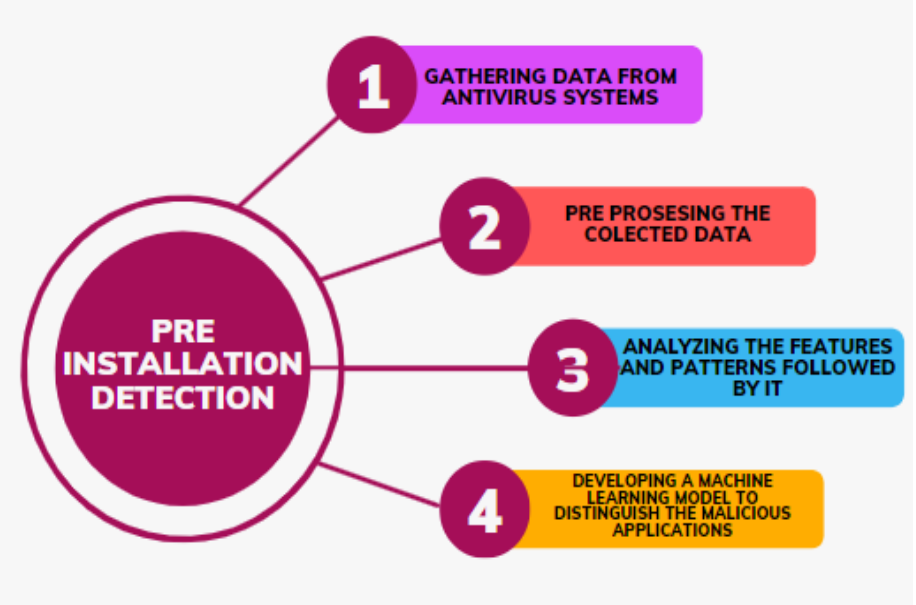


Fig 3.8 Phase 4 architecture

CHAPTER 4

PRE-INSTALLATION DETECTION THROUGH MACHINE LEARNING

4.1 DATA SET

The process of building a machine learning model for malware detection typically involves identifying a set of attributes or features that are indicative of safe or malicious applications. These features may include a variety of factors, such as the name of the application, the file type, and the behavior of the device after download. In order to train a machine learning algorithm to differentiate between safe and malicious applications, a large dataset of labeled examples is needed. The dataset should contain a significant number of malware samples, as well as a representative sample of safe applications. Once the dataset has been prepared, a machine learning algorithm can be trained to recognize patterns in the data that are associated with safe or malicious applications. The algorithm can then be used to make predictions about new applications based on their attributes or features. The features mentioned in the given context, such as RAM usage, CPU usage, downloaded time, and the source of the downloaded program, are all potentially useful in distinguishing between safe and malicious applications. By analyzing these features, a machine learning algorithm can identify patterns that are indicative of malware and use them to make accurate predictions.

	app_name	file_format	ram_usage	download_time	behaviour_of_the_device	automatic_or_manual	permissions_taken	verified_by	digital_signature_found
0	Enmeet	PDF	48	Saturday, December 11, 2021	Abnormal	Automatic	User	microsoft store	no
1	Man Mobile	EXE	97	Saturday, February 20, 2021	Normal	Automatic	Others	not known	no
2	Ytrap	ZIP	77	Wednesday, August 11, 2021	Normal	Manual	Others	McAfee	yes
3	Andes Shop	EXE	57	Monday, October 25, 2021	Abnormal	Manual	User	McAfee	yes
4	Orty	EXE	56	Monday, August 2, 2021	Abnormal	Automatic	Admin	not known	no

behaviour_of_the_device	automatic_or_manual	permissions_taken	verified_by	digital_signature_found	browser	passed_browser_firewall	website_name	malicious
Abnormal	Automatic	User	microsoft store	no	Firefox	no	youtube	0
Normal	Automatic	Others	not known	no	Edge	no	SnapFiles	0
Normal	Manual	Others	McAfee	yes	Yahoo	yes	nytimes.com	0
Abnormal	Manual	User	McAfee	yes	Tor	yes	FileHorse	0
Abnormal	Automatic	Admin	not known	no	Firefox	yes	fr.wikipedia.org	0

Fig 4.1: Data set containing information about apps.

4.2 ALGORITHMS USED

Machine learning algorithms are a subset of artificial intelligence that enable computers to learn from data, recognize patterns and make decisions based on the insights gathered from that data. These algorithms have become increasingly important in many different industries because of their ability to provide accurate predictions and insights based on vast amounts of data. The importance of machine learning algorithms lies in their ability to automate and optimize decision-making processes. By using these algorithms, businesses can analyze vast amounts of data to identify patterns and insights that would be difficult or impossible to detect using traditional analytical methods. This can help organizations make more informed decisions, improve efficiency, and reduce costs. Some of the key roles of machine learning algorithms include:

Predictive analytics: Machine learning algorithms can be used to predict future outcomes based on past data. For example, businesses can use these algorithms to predict which products are most likely to sell, which customers are most likely to churn, or which financial investments are most likely to yield a return.

Image and speech recognition: Machine learning algorithms can also be used to recognize patterns in visual or auditory data. This can be used for tasks such as facial recognition, speech recognition, and object detection.

Natural language processing: Machine learning algorithms can help computers understand and interpret human language. This can be used for tasks such as chatbots, sentiment analysis, and language translation.

Recommender systems: Machine learning algorithms can be used to recommend products, services, or content based on a user's past behavior. For example, Netflix uses machine learning algorithms to recommend movies and TV shows to users based on their viewing history.

Overall, machine learning algorithms are an important tool for businesses and organizations looking to improve their decision-making processes and gain valuable insights from large amounts of data.

4.2.1 GRADIENT BOOST HISTORY OF GRADIENT BOOST

Gradient Boosting is a popular machine learning algorithm used for regression and classification problems. It was first introduced by Jerome Friedman in 1999, in a paper titled "Greedy Function Approximation: A Gradient Boosting Machine." The algorithm has since undergone several improvements and variations, including the widely used XGBoost and LightGBM. The idea behind Gradient Boosting is to combine several weak learners to create a strong learner. Weak learners are models that perform slightly better than random guessing, such as decision trees with a single split. In each iteration of Gradient Boosting, a new weak learner is trained to predict the residuals of the previous learner. The residuals are the differences between the actual values and the predictions made by the previous learner. The new learner is then added to the ensemble, and the process is repeated until a desired level of performance is achieved. The term "gradient" in Gradient Boosting refers to the use of gradient descent optimization to minimize the loss function. The loss function is a measure of the difference between the predicted values and the actual values. Gradient descent involves iteratively adjusting the parameters of the model to minimize the loss function, by following the direction of steepest descent of the gradient. One of the key advantages of Gradient Boosting is its ability to handle non-linear relationships between the input features and the target variable. It can also handle missing data, outliers, and categorical variables.

However, Gradient Boosting can be sensitive to overfitting, especially when the number of iterations or the depth of the trees is too high. Regularization techniques, such as shrinkage and early stopping, can be used to prevent overfitting and improve generalization performance. Overall, Gradient Boosting has become a popular and powerful algorithm in the field of machine learning, with applications in areas such as finance, healthcare, and e-commerce. The machine learning boosting system known as gradient boosting represents a decision tree for large and complex data. It is predicated on the idea that the next model will lower the overall prediction error when combined with the previous set of models. Decision trees are used to make the most accurate predictions. The gradient boosting method is also known as the statistical predictive algorithm. Even though

it allows for the generalization and optimization of divergent loss functions, it still functions largely in the same way as earlier boosting methods. Processes for classification and regression frequently use gradient boosting.

```
Classification Report for train
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     31276
           1       1.00      0.99      1.00     31652

    accuracy                           1.00     62928
   macro avg       1.00      1.00      1.00     62928
weighted avg       1.00      1.00      1.00     62928

Classification Report for test
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     13399
           1       0.99      0.51      0.67       180

    accuracy                           0.99     13579
   macro avg       0.99      0.76      0.84     13579
weighted avg       0.99      0.99      0.99     13579

Train ROC:  0.9996191890292259
Test ROC:  0.9987111285458915

Train Accuracy Score:  0.9968535469107551
Test Accuracy Score:  0.9934457618381324

Train F1 Score:  0.9968535188271423
Test F1 Score:  0.835341138930501
```

Fig 4.2: Results of gradient boost algorithm

4.2.2 LOGISTIC REGRESSION HISTORY OF LOGISTIC REGRESSION

Logistic regression is a popular statistical model used for binary classification problems, where the goal is to predict a binary outcome (e.g. yes/no, 1/0). It was first introduced by David Cox in 1958, in a paper titled "The Regression Analysis of Binary Sequences." Cox's paper introduced the concept of maximum likelihood estimation (MLE) for estimating the model coefficients in logistic regression. MLE is a statistical method for estimating the parameters of a model that maximizes the likelihood of the observed data, given the model. Cox's paper demonstrated how logistic regression can be used to model the probability of an event occurring, given a set of explanatory variables. In the years that followed, logistic regression was further developed and refined by statisticians and researchers. The development of computing power and software made it easier to apply logistic regression to large

datasets and complex problems. In the 1970s and 1980s, logistic regression gained popularity in the field of epidemiology, where it was used to model the relationship between risk factors and the occurrence of diseases. Logistic regression was also used in social sciences, marketing, and finance to model binary outcomes such as whether a person will buy a product, whether a loan will default, or whether a candidate will win an election. In the 1990s and 2000s, logistic regression was incorporated into machine learning algorithms, such as support vector machines and neural networks, as a component of binary classification models. This integration helped to improve the accuracy and predictive power of these algorithms. Today, logistic regression is widely used in many fields, including medicine, biology, finance, and social sciences. It is a versatile and flexible model that can be used to model the relationship between binary outcomes and a wide range of explanatory variables, such as demographic variables, behavioral variables, and genetic variables. Logistic regression has become an essential tool for researchers and practitioners in many fields who need to make predictions or decisions based on binary outcomes. Logistic regression is a widely used algorithm in machine learning for classification tasks. It is a type of supervised learning method that uses a set of predictor variables to predict the probability of a binary outcome, which is represented by a categorical variable. The goal of logistic regression is to estimate the relationship between the input variables and the probability of the binary outcome. Logistic regression is a widely used statistical method for binary classification problems where the goal is to predict a binary outcome based on input variables. The logistic regression algorithm uses the sigmoid function, which is an S-shaped curve that maps any real-valued input to a value between 0 and 1. In logistic regression, the sigmoid function is used to map the input variables to the predicted probability of the binary outcome.

To train a logistic regression model, the algorithm takes a set of input variables and their corresponding binary outcomes as the training data. The algorithm then estimates the parameters of the logistic function that best fits the training data by minimizing a cost function such as the cross-entropy loss function. The optimized parameters of the logistic function are then used to predict the probability of the binary outcome for new input data. Logistic regression is a powerful and interpretable method for binary classification problems. The coefficients of the

model indicate the direction and magnitude of the effect of each input variable on the probability of the binary outcome. However, logistic regression assumes a linear relationship between the input variables and the binary outcome, and may not perform well when there are complex interactions between the input variables. Nonetheless, logistic regression remains a widely used and effective tool for binary classification problems in many fields.

```
Classification Report for train
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     30995
           1       1.00      0.98      0.99     31933

    accuracy                           0.99     62928
   macro avg       0.99      0.99      0.99     62928
weighted avg       0.99      0.99      0.99     62928

Classification Report for test
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     13286
           1       1.00      0.32      0.48       293

    accuracy                           0.99     13579
   macro avg       0.99      0.66      0.74     13579
weighted avg       0.99      0.99      0.98     13579

Train ROC:  0.9996191890292259
Test ROC:  0.9987111285458915

Train Accuracy Score:  0.9920703025680142
Test Accuracy Score:  0.9852713749171514

Train F1 Score:  0.9920698620754638
Test F1 Score:  0.7371973967078743
```

Fig 4.3 Results of the logistic-regression algorithm

4.2.3 RANDOM FOREST HISTORY OF RANDOM FOREST

Random forest is a popular machine learning algorithm that was first introduced by Leo Breiman and Adele Cutler in 2001. It is a type of ensemble learning method that combines multiple decision trees to create a more accurate and robust model. The algorithm works by randomly selecting a subset of the input variables and a subset of the training data to create each decision tree, thus introducing variation and reducing the risk of overfitting. Each decision tree in a random forest makes a prediction based on the subset of input variables it was trained on. The final prediction of the random forest is obtained by aggregating the predictions of all the

individual trees, either by taking the majority vote in classification problems or by taking the average in regression problems. The random forest algorithm has several advantages, including its ability to handle high-dimensional data and its robustness to noisy or missing data. Random Forest is widely used in various applications, such as image classification, bioinformatics, and finance, due to its versatility and high accuracy. It also has the advantage of being able to handle imbalanced datasets, which is a common issue in many real-world scenarios. Random Forest is also known for its scalability, as it can handle large datasets efficiently by parallelizing the computation. Another benefit of Random Forest is that it is relatively easy to interpret, as it provides measures of feature importance that can help identify which variables have the most impact on the model's predictions.

```
Classification Report for train
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     31343
           1       1.00      1.00      1.00     31585

    accuracy                           1.00     62928
   macro avg       1.00      1.00      1.00     62928
weighted avg       1.00      1.00      1.00     62928

Classification Report for test
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13434
           1       0.94      0.60      0.73       145

    accuracy                           1.00     13579
   macro avg       0.97      0.80      0.86     13579
weighted avg       1.00      1.00      0.99     13579

Train ROC:  0.9996191890292259
Test ROC:  0.9987111285458915

Train Accuracy Score:  0.9980771675565726
Test Accuracy Score:  0.9952868399734884

Train F1 Score:  0.9980771604472876
Test F1 Score:  0.8643575112065629
```

Fig 4.4: Results of random forest algorithm

4.3 MACHINE LEARNING WORKFLOW

To complete this project, I have followed several steps. First, I imported necessary libraries such as pandas, numpy, seaborn, and learn. These libraries are used for data manipulation, data visualization, and machine learning.

Then, I read the dataset (in Excel format) using the pd.read_excel() function and printed information about the dataset using df.info() and df.isnull().sum() functions. The df.info() function provides information about the dataset such as the number of rows and columns, column names, data types of columns, and memory usage. The df.isnull().sum() function provides the number of missing values in each column.

Next, I explored the data using df.describe() and df['malicious'].value_counts() functions to understand the distribution of data and the number of malicious and non-malicious instances. The df.describe() function provides summary statistics such as mean, standard deviation, minimum value, maximum value, and quartiles for each numerical column. The df['malicious'].value_counts() function provides the count of each unique value in the 'malicious' column.

I defined two functions, binary_encode() and category_encode(), to transform categorical and binary data into numerical form. The binary_encode() function encodes binary data (with only two possible values) into 0 and 1. The category_encode() function uses LabelEncoder to transform categorical data into numerical form. LabelEncoder assigns a unique integer value to each unique category in a column.

Using seaborn visualizations such as sns.barplot() and sns.violinplot(), I visualized relationships between different features and the target variable (malicious). sns.barplot() is used to show the average value of a numerical variable for each category in a categorical variable. sns.violinplot() is used to show the distribution of a numerical variable for each category in a categorical variable.

I preprocessed the data by dropping the 'download_time' column, binary encoding binary data, and category encoding categorical data. The preprocessed data were

split into training and testing sets using the train_test_split() function with a 70:30 ratio and stratified sampling. Stratified sampling ensures that each class in the target variable is represented proportionally in both training and testing sets.

I scaled the features using the StandardScaler() function, which standardizes the data by subtracting the mean and dividing it by the standard deviation. StandardScaler ensures that each feature has zero mean and unit variance. I fit the scaler on the training data and transformed both training and testing data.

To prevent bias towards the majority class during model training, I used the SMOTE algorithm to balance the data by oversampling the minority class (malicious). SMOTE generates synthetic samples for minority class instances by interpolating between neighboring instances.

I defined a function called calculate_metrics() to evaluate the performance of different machine learning models. This function calculates various metrics such as accuracy, precision, recall, F1 score, and ROC AUC score for each model.

4.4 CHALLENGES FACED

As a machine learning developer, I faced several challenges during the development of this project. One of the first challenges was to identify the appropriate libraries to use for data manipulation, data visualization, and machine learning. I had to research and experiment with various libraries such as pandas, numpy, seaborn, and sklearn to find the best tools for the project.

Another challenge I faced was data preprocessing. I had to identify missing values, outliers, and imbalanced data and come up with strategies to handle them. For instance, I used the train_test_split() function with stratified sampling to ensure that each class in the target variable was represented proportionally in both training and testing sets. I also used the SMOTE algorithm to oversample the minority class and balance the data. Choosing the right machine learning model was another challenge. I had to experiment with various models such as logistic regression, decision trees, random forests, and SVMs to find the best one for the project. I evaluated the performance of each model using various metrics such as

accuracy, precision, recall, F1 score, and ROC AUC score and selected the one with the best performance.

Finally, I faced challenges related to model optimization and deployment. I had to tune the hyperparameters of the selected model to improve its performance and ensure that it could handle new data effectively. I also had to deploy the model to a production environment and ensure that it worked correctly and efficiently. Designing a user-friendly interface: Streamlit offers a wide range of built-in components such as sliders, dropdown menus, and text fields, but designing an interface that is easy to use and understand for non-technical users can be challenging. It is important to consider the target audience and design an interface that is intuitive and easy to navigate.

Handling user input: The user interface allows users to input data and see the results of the model's predictions in real-time, making it crucial to handle user input efficiently. Ensuring that the user inputs the correct information in the right format can be challenging, and it is important to provide clear instructions and error messages to guide the user through the input process. Integrating the machine learning model: Integrating the machine learning model with the user interface can be challenging, especially if the model is complex or has large data sets. It is important to ensure that the model runs smoothly and efficiently, and that the interface provides clear and accurate results. Debugging and testing: Debugging and testing the user interface and machine learning model together can be a time-consuming process. Ensuring that the interface and model work seamlessly and accurately requires thorough testing and debugging to identify and fix any errors. Deployment: Deploying a machine learning model can be challenging, particularly when it comes to integrating it with a user interface that can handle multiple users and requests. This is especially true for large models that require significant computing resources, as it can affect the performance and speed of the application. Therefore, it is essential to ensure that the interface and model are deployed correctly and efficiently to meet the users' needs. Proper deployment of the user interface and machine learning model requires careful planning and testing to avoid any potential issues that may arise during deployment. For instance, the interface must be user-friendly, easily navigable, and responsive,

allowing users to interact with the model effectively. The model, on the other hand, should be optimized for deployment, taking into account factors such as computing resources, scalability, and maintainability. It is also essential to ensure that the model is secure and that it can handle multiple requests from different users without compromising the system's performance.

4.5 USER INTERFACE WITH STREAMLIT

Streamlit is a powerful tool that allows developers to create interactive data science applications with just a few lines of code. With Streamlit, you can quickly build and deploy custom user interfaces that enable users to interact with your machine learning models and other data science tools in a seamless and intuitive way. The Streamlit interface is built on top of Python, which makes it an ideal tool for data scientists and developers who are already familiar with Python and its libraries. With Streamlit, you can easily create custom dashboards, visualizations, and other interactive tools that enable users to explore and analyze complex data sets in real-time.

One of the main advantages of Streamlit is that it is incredibly easy to use. You don't need to be an experienced developer or have a deep understanding of web development technologies to get started with Streamlit. Instead, you can simply write Python code and use Streamlit's simple syntax and built-in components to create custom user interfaces that enable users to interact with your machine learning models and other data science tools. To use Streamlit, you start by installing the Streamlit package and importing it into your Python script. From there, you can use a variety of Streamlit's built-in components to create custom user interfaces, including sliders, dropdown menus, text boxes, and more. You can also incorporate data visualizations and other interactive elements into your Streamlit interface, such as charts, maps, and interactive tables.

Once you have created your Streamlit interface, you can deploy it using a variety of hosting platforms, including Heroku, Google Cloud, and AWS. Streamlit also offers a variety of tools and resources to help you optimize your interface and improve its performance, including a built-in profiler that can help you identify performance bottlenecks and optimize your code for speed and efficiency.

Overall, Streamlit is an incredibly powerful and versatile tool for data scientists and developers who want to create custom user interfaces for their machine learning models and other data science tools. With its simple syntax, built-in components, and robust hosting and deployment options, Streamlit is an essential tool for anyone working in the field of data science and machine learning. malware detection tool that I created using machine learning technology to thoroughly analyze the physical features of a downloaded application before it is installed on a device. The tool requires manual input from the user about the physical features of the downloaded application, such as the file size, name, format, verified source, permissions required during installation, RAM usage after download, behavior of the device post-download, if the application has a digital signature, if it passed Chrome's firewall, and from which browser it was downloaded.

This comprehensive approach to analyzing a downloaded application's physical features allows the tool to detect potential threats that traditional antivirus software may miss. This proactive approach ensures that the device is protected before any harm can be done. The manual input gives the user more control over the security of their device. The user-friendly interface created with Streamlit makes it easy for anyone to use and understand the security status of a downloaded application. The tool provides clear and concise information, making it accessible for users of all technical backgrounds. Streamlit is a Python library used to build interactive web applications for data science and machine learning projects. It allows for quick and easy creation of user interfaces and data visualizations using simple Python scripts. In this project, Streamlit was used to create an intuitive interface for the malware detection tool, which allows for easy manual input from the user and displays the results of the machine learning analysis in a clear and concise manner. The combination of machine learning technology and Streamlit's user-friendly interface makes this malware detection tool a must-have for anyone looking to protect their device from potential threats. The tool is easy to use, effective, and always up-to-date, making it the perfect solution for keeping devices safe and secure at all times. By giving the user the ability to provide manual input, I empower them to take control of their device's **security.**

4.5.1 HOME PAGE

The home page of my malware detection tool interface serves as the main entry point for users. It provides an introduction to the tool, its purpose, and its importance. The home page also includes a brief overview of the different sections of the interface and what each section entails.

The home page of the tool is specifically designed to provide a user-friendly and informative introduction to the tool's features and functions. The language used on the page is intentionally kept simple and easy to understand, so that even users with limited technical knowledge can grasp the concepts presented. By providing users with a clear understanding of what the tool does and how it works, the home page aims to instill confidence in users and encourage them to use the tool regularly. In addition to explaining the purpose of the tool, the home page also includes detailed instructions on how to use it. This section guides users through the process of inputting the necessary information about a downloaded application into the tool. The instructions cover a wide range of parameters, including file size, name, format, verified source, permissions required during installation, RAM usage after download, behavior of the device post-download, if the application has a digital signature, if it passed Chrome's firewall and from which browser it was downloaded. By providing such a comprehensive guide, the home page ensures that users can use the tool effectively and accurately.

Additionally, the home page includes information about the tool's machine learning technology, which is used to thoroughly analyze the physical features of a downloaded application before it is installed on the user's device. This information helps users understand the advanced technology behind the tool and provides reassurance that it is capable of detecting potential threats that traditional antivirus software may miss. The interface's user-friendly design and intuitive layout make it easy for users to navigate through the tool's different sections. The home page serves as an essential guide for users who are new to the tool, providing a clear understanding of what the tool does and how it works.

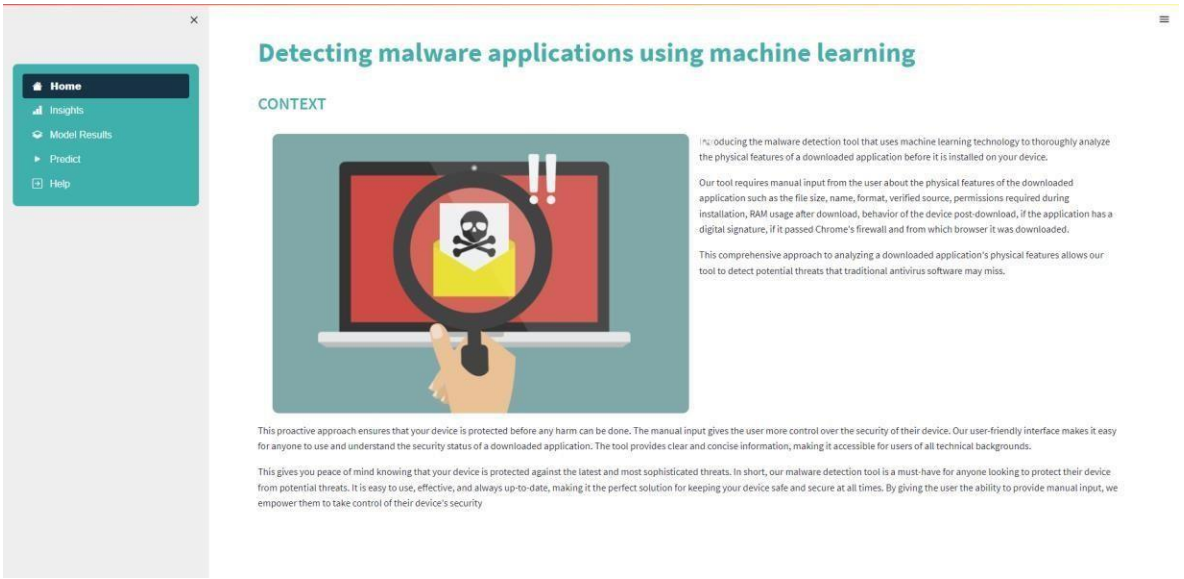


Fig 4.5: Home page in streamlit user interface

4.5.2 INSIGHTS PAGE

The Insights page of my Streamlit interface is designed to provide users with an in-depth analysis of the data used in the machine learning model. It includes exploratory data analysis and data visualizations to help users understand the correlations and patterns in the data. The page begins with a brief introduction to the data used in the machine learning model. I explain the different features and their importance in detecting malware. This section helps users understand why certain features are important and how they contribute to the overall performance of the model. Next, the page displays a set of data visualizations to help users understand the relationship between the different features. The visualizations include scatter plots, histograms, and heatmaps, which make it easy to identify patterns and correlationsin the data.

For example, one of the visualizations is a scatter plot of file size versus CPU usage. This plot shows how larger files tend to have higher CPU usage, which could indicate that they are more likely to be malware. Similarly, another visualization shows the relationship between file format and RAM usage, which can help users understand which file formats are more likely to cause issues on their devices. In addition to the data visualizations, the Insights page also includes a section on feature importance. This section ranks the importance of each feature in themachine learning model. This information can be used to understand which

features are most important in detecting malware, and to identify any potential gaps in the data that may need to be addressed in future versions of the model. Overall, the Insights page is a powerful tool for users who want to understand the data used in the machine learning model. It provides clear visualizations and in-depth analysis of the data, making it easy for users of all technical backgrounds to understand the correlations and patterns in the data.

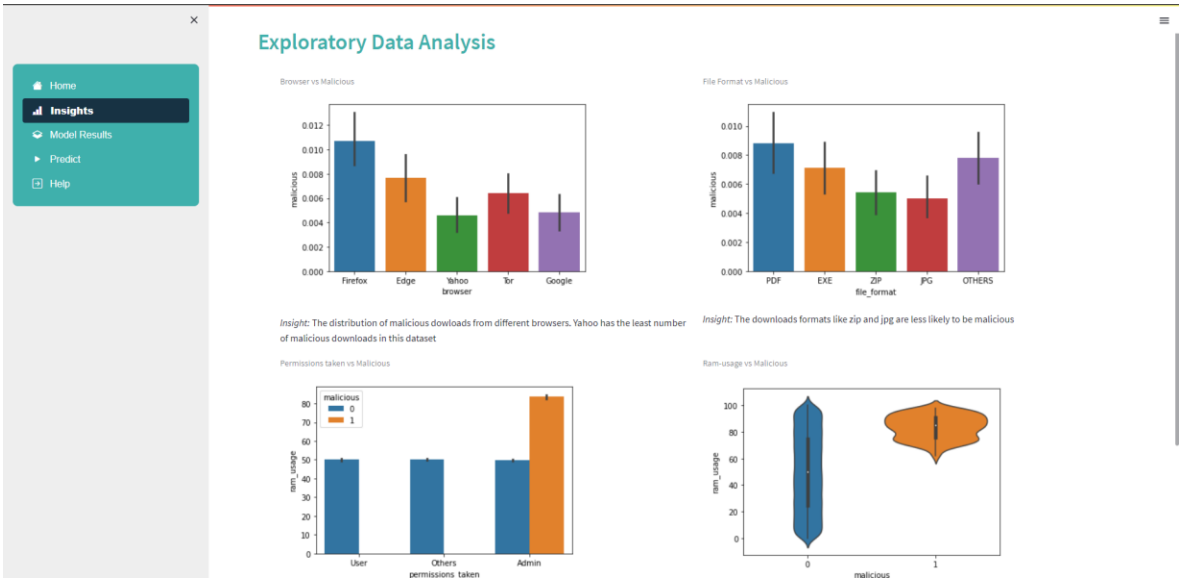


Fig 4.6: Insights page in streamlit user interface

4.5.3 MODEL RESULTS PAGE

The Model Results page in my Streamlit interface displays the results of the machine learning model. It shows whether the application is classified as malware or benign based on the physical features entered by the user.

The page displays a clear and concise message indicating whether the application is classified as malware or benign. Additionally, it shows the probability score of the classification, which provides insight into how confident the model is in its classification. In addition to the classification results, the page also displays the physical features of the application that were used as input to the machine learning model. This allows users to see which features contributed to the classification and provides insight into how the model is making its decisions. To help users understand the results of the model, the page also includes a brief description of how the model works and how it is able to detect malware. This information can be helpful for users who are unfamiliar with machine learning and may not understand

how the model is making its decisions.

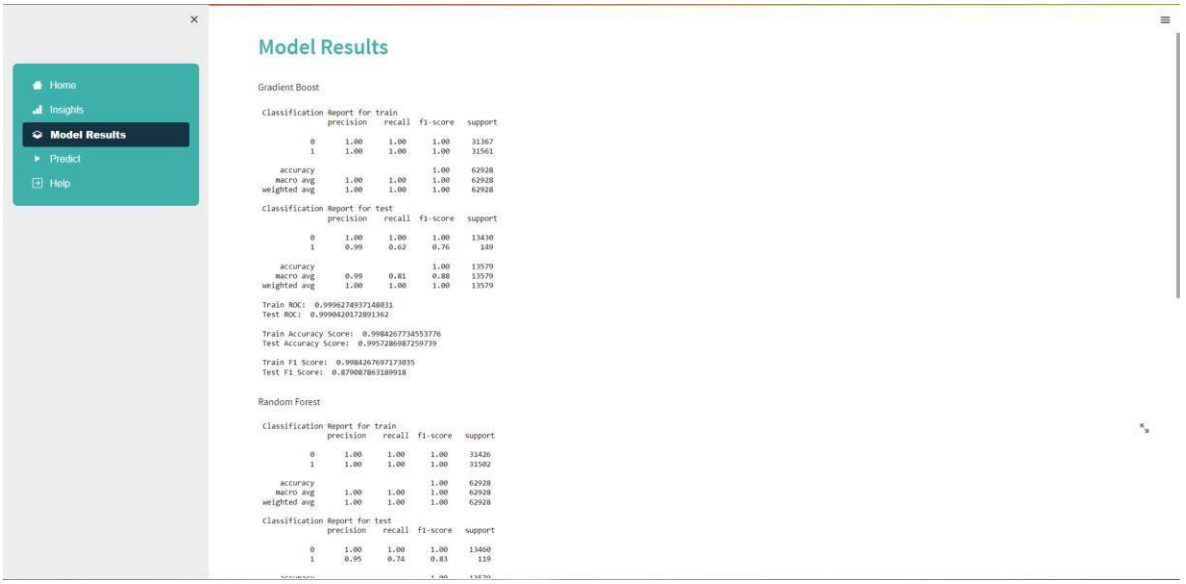


Fig 4.7: Model results page in streamlit user interface

4.5.4 PREDICT PAGE

Predict page The Predict page of my Streamlit software is where users can input the physical features of a downloaded application and get a prediction on whether or not it is likely to be malware.

The page includes a form where users can manually enter the relevant physical features, such as the file size, name, format, verified source, permissions required during installation, RAM usage after download, behavior of the device post-download, if the application has a digital signature, if it passed Chrome's firewall and from which browser it was downloaded. Once I have entered all the necessary information, I can click the "Predict" button to get a prediction on whether or not the application is malware. The prediction is based on the machine learning model that was built using the data provided during the model training phase. The page also includes a section that displays the results of the prediction. This section provides information on the likelihood of the application being malware, as well as any relevant details on the physical features that contributed to the prediction. For example, if the prediction is that the application is likely to be malware, the section might highlight the file size, format, and behavior of the device post-download as key contributing factors. This information can be useful for me if I want to better understand why the prediction was made and to make

informed decisions about whether or not to install the application on my device.

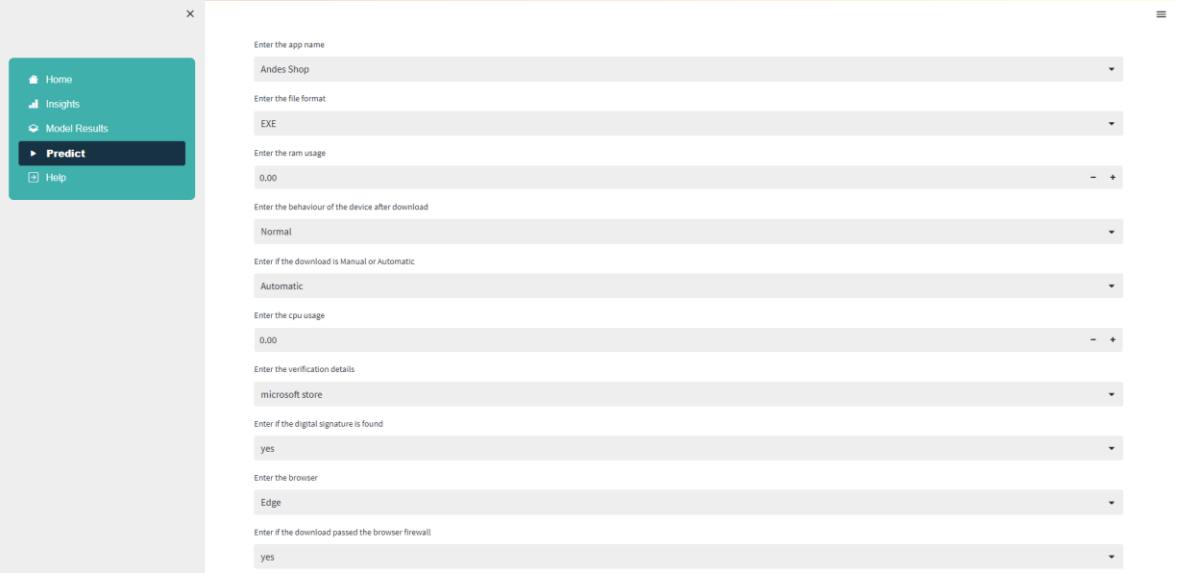


Fig 4.8: Predict page in streamlit user interface

4.5.5 HELP PAGE

The help section is a crucial component of our Streamlit interface, as it ensures that users have all the information, they need to successfully use the software. It provides a detailed guide on how to use each feature of the interface and what information needs to be provided. This helps to avoid any confusion or mistakes that could affect the accuracy of the predictions made by the machine learning model.

The help section includes several subsections, each providing detailed instructions on a specific feature or component of the software. For example, there may be a subsection that explains how to download an application and extract its physical features or a subsection that provides tips on how to interpret the results of the prediction. The help section in the interface includes detailed photos on how to provide the necessary input information. Even users with limited knowledge can easily follow these visual aids to understand the process and provide the required information accurately.

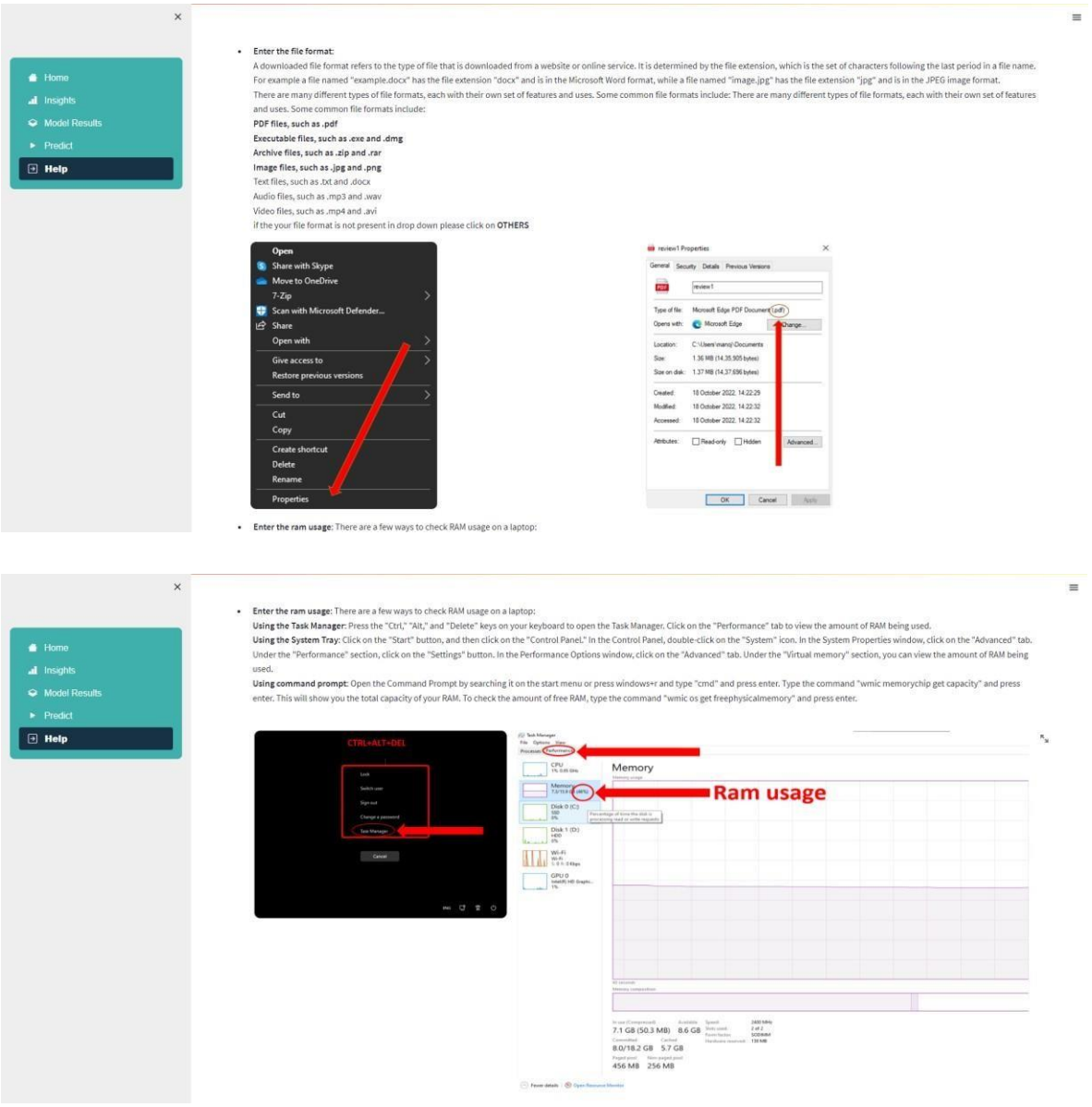


Fig 4.9: Help page in streamlit user interface

CHAPTER 5

RESULTS AND CONCLUSION

The use of machine learning algorithms for cybersecurity has become increasingly important as cyberattacks become more prevalent and costly. The development of a machine learning model that accurately predicts the threat level of an application with 99% accuracy is a significant achievement. The random forest algorithm has been shown to be particularly effective in this regard, as it demonstrated the highest f1-score for the test dataset.

The model's use of specific information on the streamlit interface enables the detection of malicious applications without even installing them. This is particularly useful as it saves time and resources compared to traditional methods, which can be expensive and time-consuming. By using advanced technologies such as machine learning algorithms, we can strengthen our defenses against cyberattacks and safeguard our digital assets.

This project not only demonstrates the effectiveness of machine learning in cybersecurity but also raises awareness about various hacking methods and their defenses. By addressing security vulnerabilities in computer operations, users can effectively protect their sensitive data. It is essential to stay vigilant and proactive in preventing cyberattacks, and this project highlights the potential of advanced technologies such as machine learning to help achieve this goal.

In conclusion, the use of machine learning in cybersecurity offers a promising solution for preventing cyberattacks and protecting sensitive data. The development of the machine learning model discussed in this project provides a concrete example of how this technology can be effectively applied. By continuing to explore and incorporate advanced technologies into our security measures, we can stay ahead of the constantly evolving threat landscape and ensure the safety of our digital assets.

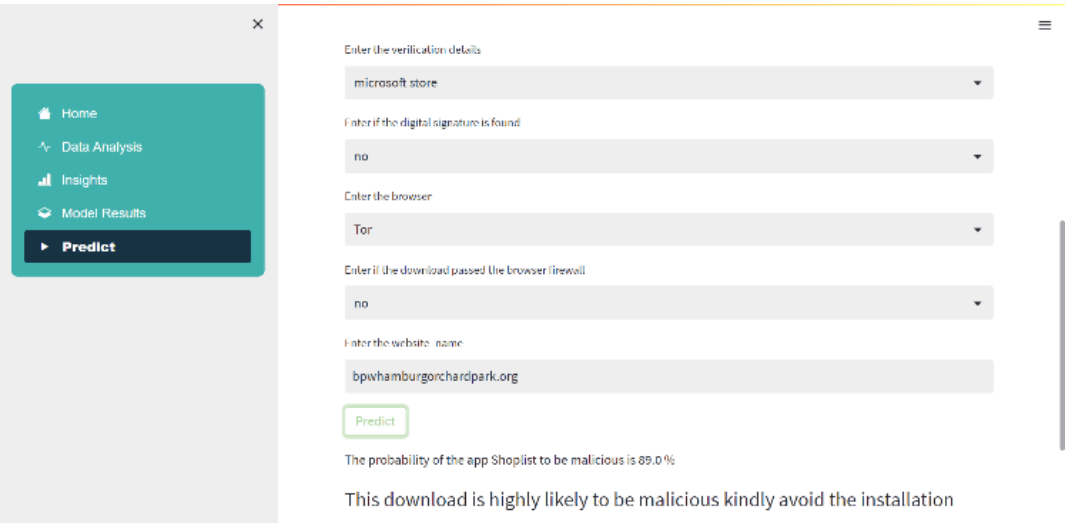


Fig 5.1 final result of the proposed system

5.1 FUTURE SCOPE

The future scope for the project can involve several aspects that could enhance the effectiveness and applicability of machine learning algorithms in cybersecurity. One of the critical areas for future development could be improving the accuracy of the model. This could be achieved by incorporating more data sources and features into the algorithm, allowing it to identify patterns and trends more accurately. Additionally, the algorithm could be trained on a more extensive dataset to enhance its ability to detect new and emerging threats.

Another important aspect of future development could be integrating the machine learning model into existing cybersecurity systems and tools. This could enable real-time threat detection and response, providing organizations with a more comprehensive and proactive approach to cybersecurity. For instance, integrating the model into a security information and event management (SIEM) system could enable automated response and remediation, significantly reducing the time to detect and respond to cyber threats. Furthermore, the project's future scope could include exploring other applications of machine learning in cybersecurity, such as anomaly detection, intrusion detection, and behavioral analysis. Anomaly detection could be useful in identifying unusual activity on a network that may indicate a cyberattack. Intrusion detection could be used to detect and prevent unauthorized access to a system or network. Behavioral analysis could be applied to identify abnormal user behavior and detect potential insider threats.

Lastly, future development could involve exploring the use of machine learning in other areas of cybersecurity, such as threat intelligence and vulnerability management. Machine learning algorithms could be used to analyze threat intelligence feeds and identify potential threats, enabling organizations to take proactive measures to prevent cyberattacks. Additionally, machine learning could be used to analyze vulnerability data and prioritize remediation efforts, helping organizations to manage their cybersecurity risk more effectively.

In summary, the future scope for the project is vast and could involve many potential avenues for further research and development. By continuing to explore the capabilities of machine learning in cybersecurity, we can stay ahead of the constantly evolving threat landscape and protect our digital assets more effectively.

CHAPTER 6

REFERENCES

[1] Yu, W., Yalin, Y., & Haodan, R. (2019, October). Research on the technology of trojan horse detection. In 2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA) (pp. 117-119). IEEE.

[2] Al-Asli, M., & Ghaleb, T. A. (2019, April). Review of signature-based techniques in antivirus products. In 2019 International Conference on Computer and Information Sciences (ICCIS) (pp. 1-6). IEEE.

[3] Asish, M. S., & Aishwarya, R. (2019, March). Cyber security at a glance. In 2019 Fifth International]Conference on Science Technology Engineering and Mathematics (ICONSTEM) (Vol. 1, pp. 240-245). IEEE.

[4] Rathore, H., Agarwal, S., Sahay, S. K., & Sewak, M. (2018, December). Malware detection using machine learning and deep learning. In International Conference on Big Data Analytics (pp. 402-411). Springer, Cham.

[5] Namanya, A. P., Cullen, A., Awan, I. U., & Disso, J. P. (2018, August). The world of Malware: An overview. In 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud) (pp. 420-427). IEEE.

[6] Bhardwaj, A., & Goundar, S. (2020). Keyloggers: silent cyber security weapons. Network Security, 2020(2), 14-19.

[7] Parthy, P. P., & Rajendran, G. (2019, October). Identification and prevention of social engineering attacks on an enterprise. In 2019 International Carnahan Conference on Security Technology (ICCST) (pp. 1-5). IEEE.

[8] Kunwar, R. S., & Sharma, P. (2016, April). Social media: A new vector for cyber- attack. In 2016 International Conference on Advances in Computing,

Communication, & Automation (ICACCA)(Spring) (pp. 1-5). IEEE.

[9] Aslan, Ö. A., & Samet, R. (2020). A comprehensive review on malware detectionapproaches. IEEE Access, 8, 6249-6271.

[10] Idika, N., & Mathur, A. P. (2007). A survey of malware detection Purdue University,48(2), 32-46.

[11] Creutzburg, Reiner (2017). "The strange world of keyloggers - an overview, Part I".Electronic Imaging. 2017 (6): 139–148.

[12] Wajahat, A., Imran, A., Latif, J., Nazir, A., & Bilal, A. (2019). A Novel Approach ofUnprivileged Keylogger Detection. 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET).doi:10.1109/icomet.2019.8673404.

[13] Rohith, C., & Kaur, G. (2021, April). A Comprehensive Study on Malware Detection and Prevention Techniques used by Anti-Virus. In 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM) (pp. 429-434). IEEE.

[14] Nallaperumal, K. (2018). CyberSecurity Analytics to Combat Cyber Crimes. 2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). doi:10.1109/iccic.2018.8782430

[15] Alshamrani, S. S. (2022). Design and Analysis of Machine Learning Based Technique for Malware Identification and Classification of Portable Document FormatFiles. Security & Communication Networks, 2022.

APPENDIX

SOURCE CODE

❖ Machine learning source code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from category_encoders import CountEncoder

from sklearn.preprocessing import LabelEncoder from sklearn.preprocessing
import StandardScaler from sklearn.model_selection import train_test_split

from imblearn.over_sampling import SMOTE
from sklearn.ensemble import GradientBoostingClassifierfrom
sklearn.linear_model import LogisticRegression from sklearn.ensemble import
GradientBoostingClassifierfrom sklearn.svm import SVC

from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, roc_auc_score, recall_score, f1_score import pickle


# read the dataset
df = pd.read_excel('dataset.xlsx')df.info()

# null values check
df.isnull().sum()

# observation: no null values in the dataset

# summary if numerical columns
df.describe()

# value count check for the target variable
df['malicious'].value_counts()

# observation: the count of "1" value is less compared to the "0"'s -> class
imbalance

# code to convert string to numerical
```

```
    # code to convert binary variables to numerical
    def binary_encode(df, columns_with_positive_values):df = df.copy()
    for column, positive_value in columns_with_positive_values:
    df[column] = df[column].apply(lambda x: 1 if x == positive_value else 0)
    return df

    # count or frequency encoder
    def category_encode(df,cols):
for col in cols:print(col) encoder = LabelEncoder()
    df[col] = encoder.fit_transform(df[col])

    #                           df =  CountEncoder(cols=[col]).fit(df).transform(df)
    output = open(col+'.pkl', 'wb')pickle.dump(encoder, output)output.close()
    return df df['verified_by'].value_counts()EDA


    df.columns sns.barplot(x='digital_signature_found',y='malicious',data=df)


    sns.barplot(x='cpu_usage',y='ram_usage',hue='malicious',data=df)


    sns.barplot(x='browser',y='malicious',data=df)


    sns.barplot(x='verified_by',y='malicious',data=df)


    sns.barplot(x='file_format',y='malicious',data=df)


    sns.violinplot(y='ram_usage',x='malicious',data=df)

    Preprocessing


    def preprocessing(df):df = df.copy()

    # dropping date column
    df = df.drop(['download_time'], axis=1)
```

```
 # Binary encode the columns
df = binary_encode(df,
 columns_with_positive_values=[('behaviour_of_the_device','Normal'),
 ('automatic_or_manual','Manual'),('digital_signature_found','yes'),
 ('passed_browser_firewall','yes')
 ])

 # category encode
 cols = df.columns[df.dtypes=='object'].to_list()df = category_encode(df,cols)

 return df


 df = preprocessing(df)


 # all columns are converted to numerical format
 df.head()
```

Train_test_split and Standardisation

```
 # Split df into X and y
 y = df['malicious'].copy()
 X = df.drop('malicious', axis=1).copy()

 # Train-test split
 X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,
 random_state=101,stratify=y)

 X_train_temp = X_train.copy()X_test_temp = X_test.copy()

 # Scaling X with a standard scalerscaler = StandardScaler() scaler.fit(X_train)


 X_train.columnsX_train.values


 scaler = StandardScaler()scaler.fit(X_train.values)

 output = open('scaler.pkl', 'wb')pickle.dump(scaler, output) output.close()
```

```
# with open(r'scaler.pkl', 'rb') as f:
#                     temp = pickle.load(f)
#                     app_name =
temp.transform(np.array([10,1,2,2,1,2,2,2,1,1,1]).reshape(-1,11))

X_train = pd.DataFrame(scaler.transform(X_train.values),
columns=X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)

print(X_train.head())

y_train.value_counts()y_test.value_counts()

# applying smote to overcome the class imbalance
sm = SMOTE(random_state=101)
X_train, y_train = sm.fit_resample(X_train, y_train.ravel())

pd.DataFrame(y_train).value_counts()
```

Modelling

```
def calculate_metrics(model,X_train, X_test, y_train, y_test):y_pred_train =
model.predict(X_train)
y_pred_test = model.predict(X_test)

print('Confusion matrix for train:\n',confusion_matrix(y_pred_train,y_train),'\n')
print('Confusion matrix for test:\n',confusion_matrix(y_pred_test,y_test),'\n')

print('Classification Report for train\n',classification_report(y_pred_train,y_train))
print('Classification Report for test\n',classification_report(y_pred_test,y_test))

prob = model_gb.predict_proba(X_train)[:,1] print('Train ROC:
',(roc_auc_score(y_train,prob)))

prob = model_gb.predict_proba(X_test)[:,1] print('Test ROC:
',(roc_auc_score(y_test,prob)))

print('\nTrain Accuracy Score: ',(accuracy_score(y_train,y_pred_train)))print('Test
Accuracy Score: ',(accuracy_score(y_test,y_pred_test)))

print('\nTrain F1 Score: ',(f1_score(y_train,y_pred_train,average = 'macro')))
print('Test F1 Score: ',(f1_score(y_test,y_pred_test,average = 'macro')))


model_gb =
```

```
GradientBoostingClassifier(n_estimators=100,max_features='sqrt',min_samples_leaf=1)


# training
model_gb.fit(X_train,y_train) calculate_metrics(model_gb,X_train, X_test, y_train,

y_test)

# logistic regression
model_lr = LogisticRegression(penalty='l2') model_lr.fit(X_train,y_train)
calculate_metrics(model_lr,X_train, X_test, y_train, y_test)

# picking random forest as Champion model
from sklearn.ensemble import RandomForestClassifiermodel_rf =
RandomForestClassifier(n_estimators=100,max_depth=15,max_features='sqrt')
model_rf.fit(X_train, y_train)

output = open('model.pkl', 'wb')pickle.dump(model_rf, output) output.close()

feat_imp =
pd.DataFrame({'variable':X_train.columns,'Importance':model_rf.feature_importances_})
print(feat_imp.sort_values('Importance',ascending=False).head(10))

calculate_metrics(model_rf,X_train, X_test, y_train, y_test)
```

❖ **STREAMLIT CODEAPP.PY**

```
import streamlit as st st.set_page_config(layout="wide") from
streamlit_option_menu import option_menuimport sysfrom homeDemo import
Homefrom insights import EDA from help_file import helpsys.path.insert(0,
r'C:\Users\manoj\Desktop\FINAL\LIVE\streamlit\application_analysis_malicious_or
_not\Model')
from Model_results import resultsfrom Model_results import predict


with open("style.css") as f: st.markdown(f'<style>{f.read()}</style>',
unsafe_allow_html=True)


def main():
with st.sidebar:
selected = option_menu(menu_title = None,
options = ["Home","Insights","Model Results","Predict","Help",],
        icons = ["house-fill","bar-chart-line-fill","layers-half","play-fill","arrow-right-
square"],
styles={
"container": {"background-color": "#3FB0AC"},"icon": {"color": "white"},
        "nav-link": {"color":"white", "text-align": "left", "margin":"0px", "--hover-color":
"#BCCBDE"},
"nav-link-selected": {"background-color": "#173243"},
}
)
if selected == "Home":Home().run()if selected == "Help":help().run()
if selected == "Insights":EDA().run()
if selected == "Model Results":results().run()if selected == "Predict":predict().run()
main()
```

HOME.PY

```
import streamlit as stclass Home():
def run(self):
st.title('Detecting malware applications using machine learning')


st.subheader('CONTEXT')


   _, col1, col2, _ = st.columns([0.05, 1.5,1.5, 0.01])with col1:


col1.image('images\image1.png')


with col2:
st.markdown('''Introducing the malware detection tool
        that uses machine learning technology to thoroughly analyze the physical features of a downloaded application before it is installed on your device.
'''                                , unsafe_allow_html=True)
        st.markdown('''Our tool requires manual input from the user about the physical features of the downloaded application such as the file size, name, format, verified source,
        permissions required during installation, RAM usage after download, behavior of the device post-download,
        if the application has a digital signature, if it passed Chrome's firewall and from which browser it was downloaded.
'''                                , unsafe_allow_html=True)


        st.markdown('''This comprehensive approach to analyzing a downloaded application's physical features allows our tool to detect potential threats that traditional antivirus software may miss.
'''                                , unsafe_allow_html=True)


      st.markdown('''This proactive approach ensures that your device is protected before any harm can be done. The manual input gives the user more control over the security of their device.
```

Our user-friendly interface makes it easy for anyone to use and understandthe security status of a downloaded application. The tool provides clear and concise information, making it accessible for users of all technical backgrounds.
''' , unsafe_allow_html=True)

st.markdown('''This gives you peace of mind knowing that your device is protected against the latest and most sophisticated threats.

In short, our malware detection tool is a must-have for anyone looking to protect their device from potential threats. It is easy to use, effective, and always up-to-date, making it the perfect solution for keeping your device safe and secureat all times. By giving the user the ability to provide manual input, we empower them to take control of their device's security''' , unsafe_allow_html=True)

INSIGHTS.PY

import streamlit as stclass EDA():def run(self): print("insights loaded")
st.header('Exploratory Data Analysis')

_, col1, col2, _ = st.columns([0.05, 1,1, 0.01])with col1:a st.caption('Browser vs Malicious')st.image('plots\Browser.png')
st.write('*Insight:* The distribution of malicious dowloads from different browsers. Yahoo has the least number of malicious downloads in this dataset')

with col2:
st.caption('File Format vs Malicious')st.image('plots\FileFormat.png') st.write('*Insight:* The downloads formats like zip and jpg are less likely to
be malicious')
_, col1, col2, _ = st.columns([0.05, 1,1, 0.01])

with col1:
st.caption('Permissions taken vs Malicious')st.image('plots\PermissionTaken.png')
st.write('*Insight:* The downloads that take admin access are themalicious')

with col2:
st.caption('Ram-usage vs Malicious')st.image('plots\Ramusage.png')

st.write("*Insight:* The dowloads with heavy ram-usage are more likely to be malicious compared to the one with least ram-usage")

MODEL RESULTS.PY

```
import pickle
import streamlit as stimport numpy as np import pandas as pd

class results(): def run(self): st.header('Model Results')

st.write('Gradient Boost') st.image('plots\GradientBoost.png') st.write('Random

Forest') st.image('plots\RandomForest.png') st.write('Logistic Regression')

st.image('plots\LogisticReg.png')

# st.write('Precision Vs Recall curve')# st.image('plots\curve_pr.png')

# st.write('accuracy_score: 0.9441922675415704')
# st.write('f1_score: 0.8937189921276614')

class predict():def run(self):
with open('Model/pickle/model.pkl', 'rb') as f:model = pickle.load(f)

with open(r'Model/pickle/scaler.pkl', 'rb') as f:scaler = pickle.load(f)def
data_preprocess(app_name,file_format,ram_usage,
behaviour_of_the_device, automatic_or_manual,
verified_by,digital_signature_found,browser,passed_browser_firewall,website_nam
e,cpu_usage):

with open('Model/pickle/dataset.xlsx', 'rb') as f:df_temp = pd.read_excel(f)

with open(r'Model/pickle/app_name.pkl', 'rb') as f:temp = pickle.load(f)app_name =
temp.transform(np.array(app_name).reshape(-1,1))
```

```
    with open(r'Model/pickle/file_format.pkl', 'rb') as f:temp = pickle.load(f)file_format =
        temp.transform(np.array(file_format).reshape(-1,1))

# with open(r'Model/pickle/permissions_taken.pkl', 'rb') as f:# temp = pickle.load(f)#
                                        permissions_taken =
        temp.transform(np.array(permissions_taken).reshape(-1,1))

    with open(r'Model/pickle/verified_by.pkl', 'rb') as f:temp = pickle.load(f)verified_by =
        temp.transform(np.array(verified_by).reshape(-1,1))

    with open(r'Model/pickle/browser.pkl', 'rb') as f:temp = pickle.load(f)browser =
        temp.transform(np.array(browser).reshape(-1,1))


        with open(r'Model/pickle/website_name.pkl', 'rb') as f:temp = pickle.load(f)

      if website_name not in df_temp['website_name']:website_name = "Others"

        website_name = temp.transform(np.array(website_name).reshape(-1,1))

        # encoding the binary columns
        if behaviour_of_the_device=='Normal':behaviour_of_the_device =1
        else: behaviour_of_the_device=0


      if automatic_or_manual=='Manual':automatic_or_manual =1else:
        automatic_or_manual=0


      if digital_signature_found=='yes':digital_signature_found =1else:
        digital_signature_found=0


      if passed_browser_firewall=='yes':passed_browser_firewall =1else:
        passed_browser_firewall=0
```

```
            new_values = np.array([app_name,file_format,ram_usage,
        behaviour_of_the_device, automatic_or_manual,
        verified_by,digital_signature_found,browser,passed_browser_firewall,website_nam
        e,cpu_usage]).reshape(-1,11)
        print(new_values)
        data = scaler.transform(new_values)print("after scaler",data) predict_class(data)
        def predict_class(data):

data = np.array(data).reshape(-1,11) result = model.predict_proba(data)[0,1]
        pd.DataFrame(data,columns=['app_name','file_format','ram_usage',
        'behaviour_of_the_device', 'automatic_or_manual',
        'verified_by','digital_signature_found','browser','passed_browser_firewall','website_
        name','cpu_usage']).to_csv('Model/data.csv',index=False)
        if result>=0.5:
            st.write("The probability of the app ", app_name,"to be malicious is ",str
        (round(result,2)*100) +' %')
            url="This download is highly likely to be malicious kindly avoid the
        installation"
          st.markdown(f'<p style="background-color:#FFFFFF;color:##FF0000;font-
        size:24px;border-radius:2%;">{url}</p>',
        unsafe_allow_html=True)else:
            st.write("The probability of the app ", app_name,"to be malicious is ",str
        (round(result,2)*100) +' %')
        st.markdown("**Please enter the details of the App to find the probability of itto
        be a malicious**")
        st.markdown(f'<p> click on <a
        href=r"C:\\Users\\manoj\\Desktop\\FINAL\\LIVE\\streamlit\\application_analysis_ma
        licious_or_not\\help_file.py">help</a> section in case you got stuck</p>',
        unsafe_allow_html=True)
        app_name = st.selectbox('Enter the app name',('Andes Shop','Avatar
        Mobile','Baywtch','BeReady','Bookoread','Caint','Crayze','DeskGet','Enmeet','Fdea
        m','Gniteem','Gusto Mobile','Ingocal','Koob','Kulop','LilyGrate','Lopping','Man
        Mobile','MemoMe','Mettcal','MyGameNight','Nime','OrTravel','Orty','Others','Page
        One Shop','Paize','Partnip','Plalitaire','Prayons','ReadABook','Redsky','Resinagro',
```

```
'ROF','Rook','Run24','ShopFactory','Shoplist', 'Shost','Sist','Solaplay','Solo
Mobile','SolSol','Tracking Mobile','Trafor','Vrigo',
'Yalp','Yola','Ytrap',))
file_format = st.selectbox('Enter the file format',('EXE','PDF','ZIP','JPG','OTHERS'))
ram_usage = st.number_input('Enter the ram usage')


    behaviour_of_the_device = st.selectbox('Enter the behaviour of the device
after download',('Normal','Abnormal'))
    automatic_or_manual = st.selectbox('Enter if the download is Manual or
Automatic',('Automatic','Manual'))
cpu_usage = st.number_input('Enter the cpu usage')
    verified_by = st.selectbox('Enter the verification details',('microsoft store','not
known','App Store','Play Store'))
    digital_signature_found = st.selectbox('Enter if the digital signature is
found',('yes','no'))
    browser = st.selectbox('Enter the
browser',('Edge','Firefox','Google','Tor','Yahoo'))
    passed_browser_firewall = st.selectbox('Enter if the download passed the
browser firewall',('yes','no'))
website_name = st.text_input('Enter the website_name')


if st.button("Predict"): data_preprocess(app_name,file_format,ram_usage,
behaviour_of_the_device, automatic_or_manual,
verified_by,digital_signature_found,browser,passed_browser_firewall,website_nam
e,cpu_usage)
```

HELP.PY

```
   import streamlit as stimport pandas as pdclass help(): def run(self):


df = pd.read_excel('Datasets/train_dataset.xlsx')
st.write('The columns available in the dataset: ')st.markdown("'<ul>
                <li><b>Enter the app name</b>: <br>An application name is the
name given to a softwarae program or application.
```

some examples of application names include "Microsoft word","Adobe photoshop","spotify".
</br>

Enter the file format:
A downloaded file format refers to the type of file that is downloaded from a

website or online service. It is determined by the file extension, which is the set of characters following the last period in a file name.

For example a file named "example.docx" has the file extension "docx" and is in the Microsoft Word format, while a file named "image.jpg" has the file extension "jpg" and is in the JPEG image format.

There are many different types of file formats, each with their own set of features and uses. Some common file formats include:
There are many different types of file formats, each with their own set of features and uses. Some common file formats include:

PDF files, such as .pdf

Executable files, such as .exe and .dmg

Archive files, such as .zip and .rar

Image files, such as .jpg and .png

Text files, such as .txt and .docx

Audio files, such as .mp3 and .wav

Video files, such as .mp4 and .avi

if the your file format is not present in drop down please click on OTHERS
</ '',unsafe_allow_html=True)

_, col1, col2, _ = st.columns([0.05, 1.5,1.5, 0.01])with col1:

col1.image('images\image2.png')
with col2: col2.image('images\image3.png')

st.markdown('''
Enter the ram usage: There are a few ways to check RAM usage on a laptop:

Using the Task Manager: Press the "Ctrl," "Alt," and "Delete" keys on your keyboard to open the Task Manager. Click on the "Performance" tab to view the amount of RAM being used.

Using the System Tray: Click on the "Start" button, and then click on the "Control Panel." In the Control Panel, double-click on the "System" icon. In the System Properties window, click on the "Advanced" tab. Under the "Performance" section, click on the "Settings" button. In the Performance Options window, click on

```
the "Advanced" tab. Under the "Virtual memory" section, you can view the amount
of RAM being used.
<br><b>Using command prompt</b>: Open the Command Prompt by searching it
on the start menu or press windows+r and type "cmd" and press enter. Type the
command "wmic memorychip get capacity" and press enter. This will show you the
total capacity of your RAM. To check the amount of free RAM, type the command
"wmic os get freephysicalmemory" and press enter.</li></br>
'",unsafe_allow_html=True)
_, col1, col2, _ = st.columns([0.05, 0.75,1.5, 0.01])with col1:
col1.image('images\image4.png')with col2:
col2.image('images\image5.png')st.markdown("'<ul>
                    <li><b>Enter the behaviour of the device </b>: <br>There are
several signs that can indicate that a system's behavior is abnormal, such as:
<br><b>Slow performance</b>: If a system is running slowly or freezing, it may be
a sign that something is wrong.
<br><b>High CPU or RAM usage</b>: If the CPU or RAM usage is consistently
high, it may indicate that a process or program is using too many resources.
<br><b>Error messages and crashes</b>: If a system is frequently displaying error
messages or crashing, it may indicate a problem with the software or hardware.
<br><b>Abnormal Network behavior</b>: If you notice any strange network activity
such as sudden high network traffic or unknown connections, it could be an
indication of a malware or intrusion.
<br><b>Overheating</b>: If the system is overheating, it may be a sign of a
hardware problem or that the cooling system is not working properly.
<br><b>Unexpected pop-ups and ads</b>: If you are seeing unexpected pop-ups
or ads, it could be a sign of malware or adware.
<br><b>If you face any of the above issues after the download please click on
Abnormal</b></li><br>
'",unsafe_allow_html=True)
_, col1, col2, _ = st.columns([0.05, 2.5,1.5, 0.01])with col1:
col1.image('images\image6.png') st.write('High CPU or RAM usage')
col1.image('images\image8.png') st.write("Unexpected pop-ups and ads")
with col2: col2.image('images\image7.png') st.write("Error messages and crashes")
col2.image('images\image9.png') st.write("Overheating")
```

st.image('images\image10.png') st.write("Abnormal Network behavior") st.markdown("'

Enter if the download is Manual or Automatic:
Whether a download is automatic or manual refers to the process by which a file is downloaded from the internet.

Automatic downloads occur when a file is automatically transferred from a website or online service to the user's computer without any intervention by the user.Automatic downloads are often considered to be a higher risk when it comes to malicious software. These types of downloads occur without the user's intervention and can result in potentially harmful files being transferred to the user's computer. It is important to be aware of automatic downloads and to take precautions to protect against malicious software when downloading files in this manner.

Manual downloads, on the other hand, require the user to initiate the download process. This is typically done by clicking on a download button or link on a website or online service. Manual downloads give the user more control over the download process, allowing them to choose the location where the file is saved, for example.

In the context of malware detection, it is important to consider whether a file was automatically or manually downloaded. This information can be used by the machine learning model to help determine whether the file is potentially malicious, as malicious files are often automatically downloaded without the user's knowledge.

Overall, whether a download is automatic or manual is a crucial piece of information in understanding the origin and potential risk of a file.

Enter the permissions taken for the download:
Entering the permissions taken for a downloaded application is a crucial aspect of malware detection. Permissions are the specific actions that a software application is allowed to perform on a user's device, and they can provide valuable information about the behavior and potential risk of the application.

In the context of the user interface, there may be options to specify the type of permissions taken for the downloaded application. The options could include "admin," "user," or "others (if not known)."

Choosing "admin" would indicate that the application has been

granted administrative privileges, which give it full access to the user's device and the ability to make system-level changes. This is a high-level of access and should only be granted to trusted applications.

Choosing "user" would indicate that the application has been granted standard user permissions, which limit its access to the user's device and the ability to make changes. This is a more limited level of access and is typically granted to most applications.

Choosing "others (if not known)" would indicate that the user is unsure of the specific permissions granted to the application, or that the application is not yet installed and therefore the permissions are not yet known.

By entering the correct information about the permissions taken for a downloaded application, the machine learning model can make a more informed decision about the risk of the application and help to prevent the installation of malicious software.
"',unsafe_allow_html=True) st.markdown('''
Enter if the digital signature is found:One can typically check if a particular file has a digital signature by using the file properties or information available in the operating system. For example, in Windows, a user can right-click on a file and select "Properties." In the Properties window, there may be a "Digital Signatures" tab that displays information about the digital signature, including the signer and the date and time of the signature.

For PDF and ZIP files, a user can try to open the file properties to see if there is information available about a digital signature.

However, it's important to note that the absence of a digital signature does not necessarily indicate that a file is malicious. The absence of a digital signature simply means that the authenticity and integrity of the file cannot be verified.
</br>
''',unsafe_allow_html=True)
_, col1, col2, _ = st.columns([0.05, 2.5,1.5, 0.01])with col1:
col1.image('images\image11.png')with col2:
col2.image('images\image12.png')st.image('images\image13.png')
st.markdown('''
Enter Enter the browser:This information can be used as a factor in determining the safety of the downloaded application. Some browsers have built-in security features and filters that help prevent users from

downloading malicious files. By including the browser information as an input to your model, you can incorporate this factor into your malware detection algorithms and potentially improve the accuracy of your model in identifying potentially dangerous applications.

'',unsafe_allow_html=True)st.markdown('''
Enter if the download passed the browser firewall:A browser firewall is a security feature built into certain browsers, designed to protect users from downloading malicious applications. If the application passes the browser firewall, it means that the browser has determined that the application is not a security threat. On the other hand, if the application does not pass the browser firewall, it could indicate that the browser has identified the application as potentially harmful. By including this information as an input to the model, the algorithm can take into account whether the browser considers the application to be safe or potentially dangerous. This information can help improve the accuracy of the malware detection algorithms and better protect users from downloading harmful applications.

'',unsafe_allow_html=True)

_, col1, col2, _ = st.columns([0.05, 2.5,1.5, 0.01])with col1:

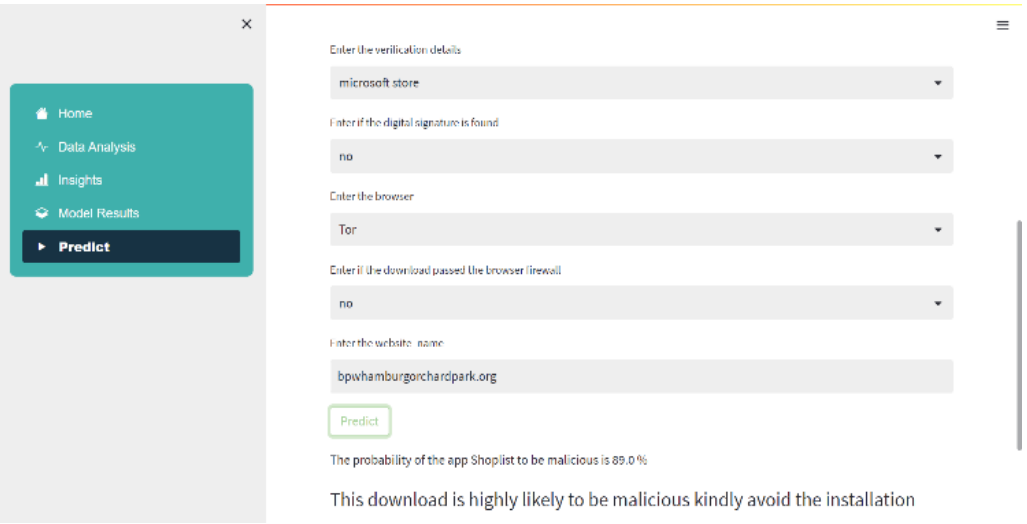
col1.image('images\image14.png')with col2:

col2.image('images\image15.png')

SCREEN SHOTS

```
Confusion matrix for train:
 [[31343     0]
 [  121 31464]]

Confusion matrix for test:
 [[13428     6]
 [   58    87]]

Classification Report for train
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     31343
           1       1.00      1.00      1.00     31585

    accuracy                           1.00     62928
   macro avg       1.00      1.00      1.00     62928
weighted avg       1.00      1.00      1.00     62928

Classification Report for test
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13434
           1       0.94      0.60      0.73       145

    accuracy                           1.00     13579
   macro avg       0.97      0.80      0.86     13579
weighted avg       1.00      1.00      0.99     13579

Train ROC:  0.9996031468393585
Test ROC:  0.9987091352402093

Train Accuracy Score:  0.9980771675565726
Test Accuracy Score:  0.9952868399734884

Train F1 Score:  0.9980771604472876
Test F1 Score:  0.8643575112065629
```

FINAL RESULT OF MACHINE LEARNING MODEL



FINAL RESULT OF STREAMLIT INTERFACE

RESEARCH PAPER

MALWARE DETECTION AND ANALYSIS USING MACHINE LEARNING

MANOJ SIRIGIRI
Department of Computer Science and Engineering Sathyabama Institute of Science and Technology Chennai, India
manojsirigiri16@gmail.com
(Author)

DIVYA SIRIGIRI
Data Scientist
Brillio Technologies
Bangalore, India
divyasirigiri7@gmail.com
(Author)

R. AISHWARYA
Department of Computer Science and Engineering Sathyabama Institute of Science and Technology Chennai, India
aishwarya.cse@sathyabama.ac.in
(Assistant Professor)

R. YOGITHA
Department of Computer Science and Engineering Sathyabama Institute of Science and Technology Chennai, India
yogitha.cse@sathyabama.ac.in
(Assistant Professor)

***Abstract*— Online privacy for people is getting worse every day. Computer malware is tainting the data records of some well-known companies. Hackers can gain access to a network and change data, once inside. This work discusses several types of malware and communication strategies, such as Trojans, keyloggers, port forwarding, source code obfuscation, application format converters, and social engineering techniques. Three different machine learning algorithms are applied in this work to derive meaningful insights from the data. As a result of the work proposed, given malware can be categorized as a malicious or a non-malicious application. This is executed by analyzing the classification report, accuracy and f1 score metrics. The Random Forest is selected as a champion model based on the highest f1-score for the validation dataset.**

Keywords*— Computer malware, Trojan, Keyloggers, Port forwarding, Social Engineering, Pre-installation detection, post-installation detection

I. INTRODUCTION

Malware, often known as malicious software, is a general word for programs or applications that are intended to harm or infect electronic devices. Your device might be held hostage, have important and private information stolen from it or have malicious software installed that watches your internet activity. All kinds of electronic gadgets, including smartphones, laptops, tablets, smart TVs, and even game consoles, are susceptible to malware infection. Malware often reproduces itself in order to spread. The virus can continue to function unnoticed inside the device's files as long as it is executing that file. Malware might be inactive and wait for a file to be opened before it begins to function and do damage. Once activated, malware might spread. First of all, malware is a term used to describe malevolent practices designed by software programmers with the purpose of harming a computer or server. The virus can corrupt a target's computer after being implemented or otherwise introduced into the target's machine. In olden times, the malware used to be simply detected using many anti-virus applications due to a lack of proper code hiding and social engineering techniques, but now that these things have gotten more advanced, results in causing more damage to the victims. People being more involved in the internet gives cybercriminals a better chance to lure more people into their trap. The evolution of malware increased rapidly, which gave a better challenge to the modern malware protection system. The capacity to fend off and recover from cyberattacks can be referred to as "cybersecurity. Antivirus software plays a significant part in cyber security. The primary role of virus scanners is to protect your computer against malicious programs and other malware. It achieves this by evaluating each threat it encounters to a "blacklist". This list includes each malware that the security software is knowledgeable of. Antivirus detects malware that has already been installed on a machine (post-installation detection) using various methods of scanning based on the user's interest. Machine learning is an area of computing algorithms that is rapidly expanding and aims to mimic human intelligence by learning from its past and its surroundings. Applications utilize machine learning algorithms to respond adequately to cyberattacks. Huge data sets of potential attacks will be analyzed in order to determine patterns and the physical features of applications and their malicious behaviour and assist with this. When comparable events are found, machine learning comes into action so that the trained model can distinguish between malware and a safe one.

II. Related Work

The history of cyberattacks and their prevention attacks should be known to one who wants to deal with advanced malware. This section contains some related work regarding my research. **Yalin, Y., & Haodan, R et.al [1]**. In this paper, the characteristics and principles of trojans are analyzed and the detection methods are compared. This paper includes detection methods like Sandbox testing and Heuristic-based testing. **Al-Asli, M., & Ghaleb et.al [2]**. This paper shares previous work on malware detection using

signature-based algorithms. Cybercrime investigation models are being combined with existing antivirus software in order to extend their benefits to society. **Aslan et.al [3].** This paper briefs different malware detection techniques Evolution of malware detection and the history of malware are discussed in this paper. **Idka et.al [4].** This study has analyzed the malware and its types. Future malware dangers and techniques are also covered in this study. The paper includes future malware threats and techniques. **Asish**, **M. S., & Aishwarya,R. et.al [5]**. In many aspects, this study described several hacking tactics and countermeasures. Learned about different types of malwares and the after-effects of the installation of that malware. Discussed all kinds of malware from old to new in order to describe the evolution of malware. Explained how traditional defence mechanisms (such as signature-based malware identification) utilized by antivirus would struggle to meet the difficulties of modern malware. **Rathore et.al [6]**. Malware analysis and detection have been modelled as machine learning and deep learning problems in this paper. These models are constructed by using different practices (cross-validation, correcting class imbalance issues, etc.). **Namanya et.al [7]**. To provide background knowledge for the intended study on developing malware detection systems, this study provides an overview of the malware world. The fundamental knowledge of ransomware and anti-malware programs is reviewed in this study. This study has provided abstracts of articles about the development of malware, malware analytical techniques, virus evasion tactics, and currently used malware detection approaches. **Bhardwaj et.al [8]**. Explained the destruction caused by keyloggers in past and how advanced they became. Keyloggers aka silent cyber weapons are deadly and undetectable in most cases. Described the economic damage caused by keyloggers. Keyloggers function at a higher privilege level than ordinary malware, making them extremely difficult to identify and eliminate. **Parthy et.al [9]**. This document categorizes numerous social engineering assaults based on the perpetrator's perspective. Explained all types of enterprise attacks and classified them. Many new social engineering techniques were discussed likes reverse social engineering. This paper will assist the reader to understand how social engineering could be used as opposed to businesses. **Kunwar et.al [10]**. This article delves into the threats, security concerns, and many sorts of social media cyberattacks. Discussed many things about spam and malicious ads in social media. Finding detailed information about the number of users who uses social media and how many are lured into these cyber-attacks gave a clear ratio of much more information. **Reiner Creutzburg et.al [11]** provide a summary of the relevant hard-, software, and mobile keyloggers that are available and in use, as well as a bibliographic overview of keyloggers. Keyloggers' capabilities, accessibility, and detection potential are examined and detailed. **Wajahat et.al [12]** This research focused on detecting the most common indigent user space keylogger. They demonstrated code snippets in this study that allow the client to deal with keylogger spyware without jeopardizing security. **Rohith et.al [13]** The purpose of this study is to describe and debate the cutting-edge technologies employed by anti-virus. They talked about how malware that lives on a system might permanently destroy its hardware components. **Nallaperumal, K [14]** This paper gives an introduction to the cyber-crimes and their impacts. **Alshamrani, S. S [15]** This study presents a Machine Learning (ML) model, which can recognize JavaScript and malicious API call assaults in PDF files.

III. TYPES OF MALWARE AND METHODOLOGIES

A. Trojan

By using social engineering, Trojans are tricked into running on users' computers. Once a Trojan has been loaded, hackers can use it to monitor victims, steal their personal data, and acquire remote access with a back door to the targets' systems. The activities could include tampering with the victim's data, monitoring target actions such as spying on target displays, and so on. For instance, the most dangerous and widely used Trojan is known as 888 Remote Access. This Trojan allows attackers to violate victims' privacy by doing various things, like gaining access to secret information by monitoring user activities with built-in keyloggers. Display monitoring allows attackers to see the live feed of what victims are seeing on their screens. An attacker can see his targets using a system's webcam in real-time. Spreading viruses and other malware by obtaining complete control over the browser, CMD, file management, drive formatting, and deleting, downloading, or modifying files and file systems. Remote Trojans are typically hidden within a user's requested program, for example, a game, or delivered as an e-mail attachment. This directly runs encoded malicious code to give the administrator control over the application.

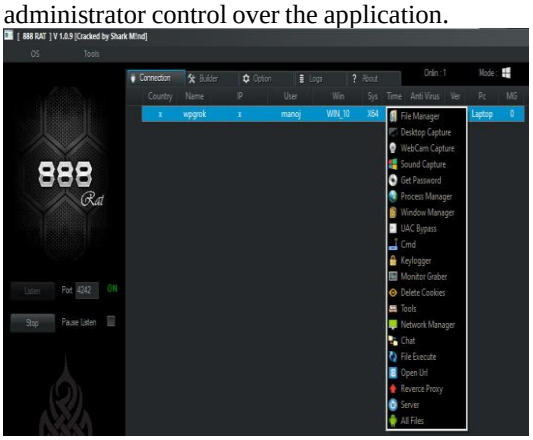


Figure 1: trojan interface

B. Keylogger

The Keystroke logger software, sometimes known as keyloggers, is a category of malware that has the ability to covertly track keyboard input from users in order to steal important information. Thus, keyloggers pose a serious risk to both personal and professional activities including payments, internet banking, mail, and small talk are all available. The keyboard is the main target. due to the fact that it is the most common method for users to connect with computers and because it makes it possible for keyloggers to collect user input from the system. Software keyloggers and handset keyloggers are the two types of keyloggers available today. Software keyloggers are much more prevalent, easier to set up, and more likely to do significant damage. Keyloggers do two tasks: they guide the customer input records keystrokes using a physical circuit that is located between both the keyboard and the machine. It stores a record of every keystroke on the keyboard in internal memory, which may be retrieved by keying in a sequence of characteristics that have been predefined. They are frequently designed to appear innocent and to mix in with the other equipment or cables. They may also be added or modified to a keyboard's internal circuitry, or the keyboard may be designed with this functionality. Software keyloggers are programs that need to be installed on a computer to collect keystroke data. Keyloggers use the Pynput library to record keystrokes. This library enables the management and observation of the keyboard. They are the most typical technique used by hackers to obtain user keystrokes. Attackers use python SMTP libraries or backdoors to track these keystrokes from keyloggers.

keylll.py - C:\Users\manoj\Desktop\FINAL\LIVE\TROJAN AND KEYLOGGER\keylogger\keylll.py (3.9.13)

File Edit Format Run Options Window Help

```
from pynput import keyboard
def write(text):
    with open("keylogger.txt",'a') as f:
        f.write(text)
        f.close()


def on_key_press(Key):
    try:
        if(Key == keyboard.Key.enter):
            write("\n")
        else:
            write(Key.char)
    except AttributeError:
        if Key == keyboard.Key.backspace:
            write("\nBackspace Pressed\n")
        elif(Key == keyboard.Key.tab):
            write("\nTab Pressed\n")
        elif(Key == keyboard.Key.space):
            write(" ");
        else:
            temp = repr(Key)+" Pressed.\n"
            write(temp)
            print("\n{} Pressed\n".format(Key))

def on_key_release(Key):
    #This stops the Listener/Keylogger.
    #You can use any key you like by replacing "esc" with the key of your choice
    if(Key == keyboard.Key.end):
        return False

with keyboard.Listener(on_press= on_key_press,on_release= on_key_release) as listener:
    listener.join()
```

Figure 2: keylogger source code

The software could operate like a basic keylogger to obtain all user data from clients by recording their keyboard strokes and mouse actions without revealing the clients' names. As a result, the client is unaware of what is going on in the foundation. The application can keep track of information, save it in a specific location and email it to the owner if needed. The program will conceal itself from the operating system while it is running in the background. I recognize that the bar for monitoring data and collecting it either for legal or illegal purposes has been greatly raised by this technique.

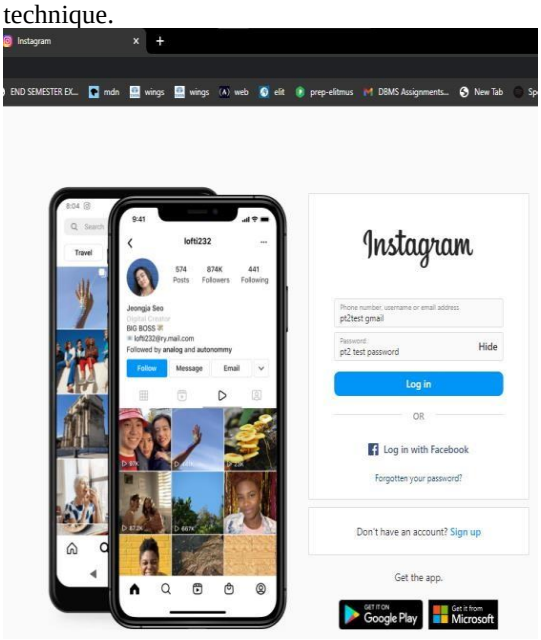


Figure 3: user typing confidential information

After being installed, the keylogger records each keystroke you make on the operating system you're using and looks at the routes they travel. This enables a keylogger piece of software to keep track of and log every keystroke you make.

keylogger - Notepad

File Edit Format View Help

```
instagram
pt2test gmail
pt2 test password<Key.cmd: <91>> Pressed.
<Key.shift: <160>> Pressed.
S
Backspace Pressed
<Key.ctrl_r: <163>> Pressed.
<Key.cmd: <91>> Pressed.
<Key.shift: <160>> Pressed.
S
<Key.alt_gr: <165>> Pressed.
<Key.ctrl_r: <163>> Pressed.
```

Figure 4: keystrokes recorded

C. Backdoor

Any method by which permitted or not permitted users may get around common security controls

and get high-level access rights on a group of computers or software applications is referred to as a backdoor. Once inside, fraudsters could use a backdoor to take control of equipment, install further malware, and steal financial and personal data. If both the target and the attacker are on the same network, a back door can be easily opened. Otherwise, the attacker employs port mapping to establish an unattended communication bridge between the attacker's and the target's systems. Back doors enable attackers to re-enter the victim's network at any time.

D. Port Forwarding

If the victim's computer is not linked to his local network, the attacker utilizes port forwarding to gain access to it. Computers or services on private networks can connect with other publicly or privately accessible computers or services on the internet thanks to port forwarding, also known as port mapping. The goal of port forwarding is to establish a connection between a router's public, WAN, and secret LAN, Internet protocol addresses for a device on that secure network. Several software and applications can be used for port forwarding like port map.io. port forwarding is just like call forwarding.

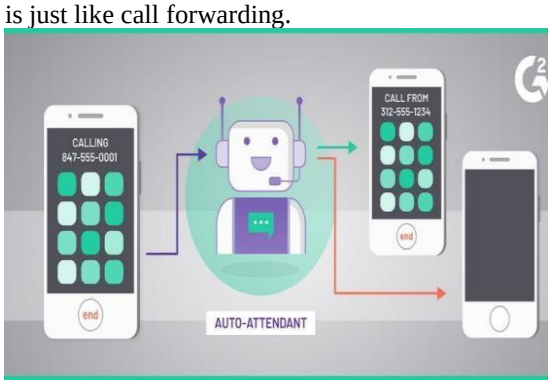


Figure 5: port forwarding

E. Malware Formatting

Malware comes in a variety of formats, including EXE, PDF, zip, and JPEG files. These file types make it simple for hackers to distribute malware or other harmful software via numerous social media platforms and mail file formats. assists attackers in hiding their source code and renders them invisible to various antivirus software. Several software's are present to change a piece of code into a specific format like a null soft script. Changed python keylogger script into an exe file for source code hiding and better malware delivery. Nullsoft Scriptable Install System) is a free software for creating Windows installers. It is meant to be as tiny and adaptable as practical, making it ideal for online distribution.

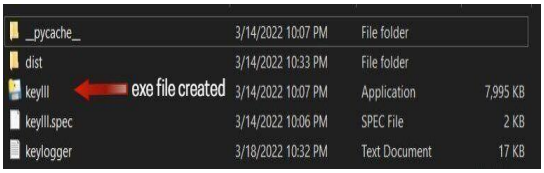


Figure 6: file formatting

F. Social Engineering

An extensive range of malicious acts carried out via interactions with other individuals is referred to as "social engineering." Users are psychologically manipulated into disclosing important data or committing security blunders. A payload can be sent to a target in a variety of formats by an attacker.

- Exe (commonly found in third-party applications and mod applications; by creating a cracked version of a paid software and encoding it with a payload, users will easily install them because they are free and simple to install, but they will overlook their security.)
- HTML format (in the name of cookies, it is possible to hack into victims' devices)
- Jpg (photo format) (Photo has been tainted with malicious code. The code will be executed if you click on it.)
- PDFs containing malicious code Opening it will cause the code to run.

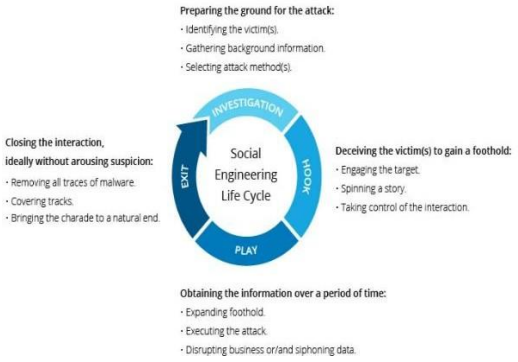


Figure 7: social engineering

IV. Post-installation detection

Post-installation detection is an application or software detecting Malware malicious applications using the data post installation detection is an application of software detecting Marlboro malicious applications using data

A. Anti-Virus

An antivirus program is a sort of application that is used to protect against, detect, and eradicate malware from a machine. After installation, most antivirus programs run on autopilot to provide real-time protection against viral threats. Effective malware protection systems can also include extra security capabilities such as configurable firewalls and Traffic shaping to protect both data and hardware from infections like worms, Trojans

and adware. Antivirus software begins to function by evaluating the data and apps on the machine to a database containing known malware types. It will also analyze PCs for any possible threats from a novel or unidentified malware kinds because hackers are always creating and disseminating new malware. Heuristic detection, generic detection, and specific detection are the three detection techniques used by the majority of programs. A malware file is normally isolated and/or marked for removal by the application when it is discovered, reducing the risk to your device and making the file unusable. A traditional antivirus program isn't as effective at thwarting threats on its own as it should be. These factors have led to the adoption of methodologies by many antivirus software vendors today, including global scanning, human threat analysis, industry collaboration, and cloud integration.

B. Types of anti-virus detection

Malware is computer software that was purposefully created to enter or harm a computer without the owner's permission. This contains, among other things, Trojan horses, worms, and viruses. Malware detection is the process of identifying whether a certain application is harmful or benign or identifying the presence of viruses on a host system.

- **Signature-based detection**: Every reliable piece of software has a digital signature. A binary pattern is all that a signature is. When a file is scanned by an antivirus tool, its signature is evaluated and compared with a vast database of digital signatures. The antivirus will immediately alert you if it has a history of being harmful.
- **Heuristic-based detection**: To identify the contamination status of your file or program, code behaviour and patterns will be examined. Any suspicious code is executed in a virtual environment during runtime to further test it. You can use this strategy to find new viruses that haven't yet been uploaded to the antivirus databases.
- **Behavioral-Based Detection**: Your antivirus software is constantly scanning your computer. If one of your software starts behaving strangely, doing serious harm and requesting more read and write permissions, your antivirus will detect this and notify you.
- **Sandbox detection**: If your antivirus software is unsure if a program or file is infected with malware and suspects it, it will execute it in a sandbox environment. This detection will run your software in a virtual system to examine how it acts. Examining system failures and examining startup applications to see if they are consuming excessive amounts of RAM and network bandwidth.

C. Types of scans in an antivirus:

- **On-access scanning:** This scanning begins when the antivirus program is launched Real-Time Scanning, sometimes referred to as On-Access Scanning, is the process where your security program continuously scans the data that users access while using your machine.
- **Manual scanning:** Manual scanning allows you to start a malware scan whenever you want.
- **Scheduled scanning:** Some software lets you schedule scans for specified days or weeks These are an essential component of maintaining your system's integrity.
- **Quick scans:** Quick scans look at frequently used areas of a system like temporary files, the OS directory, and computer cache memory.

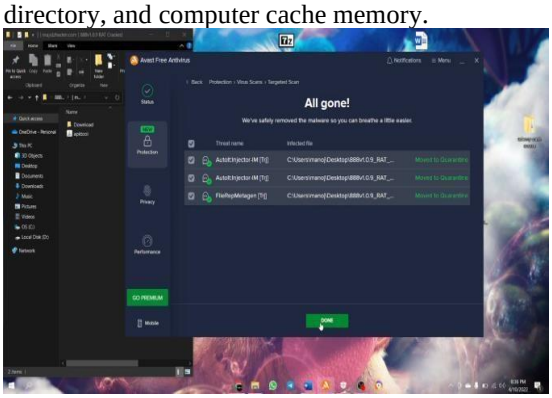


Figure 8: working of antivirus

V. PRE-INSTALLATION DETECTION

Preinstallation data is any information you can provide about a file without actually running it. Among other related data, this could comprise descriptions of the executable file format, the actions of the application, code descriptions, statistics for binary data, text strings, and information obtained through code emulation. The information collected can help us distinguish whether a download is malicious or not by training a machine learning model on the same. The model can help us understand the probability of a particular file to a malicious one and thereby the user can avoid installing the file on the device. This can help the user from falling into the trap of running harmful files and hence protecting the data and information security. Antivirus software is quite expensive to operate and takes up a lot of space on the device. It consumes RAM and memory on a regular basis in order to keep the deity healthy. Whereas these machine-learning algorithms predict if an application is harmful or not based on its physical traits and behaviour, the user must configure the information he knows about the specific program, and the trained model does the rest. The data set contains information such as app name, downloaded time, ram usage, the behaviour of the computer after download, permissions requested during installation, the presence or absence of a digital signature for the downloaded application, the source of the download, whether it is verified by several

genuine applications or not, and finally, whether the application is malicious or not. Numerous antivirus apps will gather this data using various post-installation detection methods, such as signature-based detection, behaviour-based detection, heuristic-based detection, and sandbox detection.

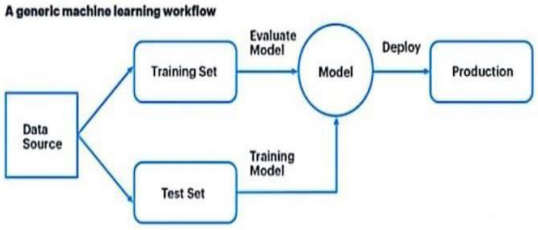


Figure 9: machine learning model process

A. Data set

The data set contains a wealth of information about the applications that are understandable to the average user and contains a large number of malware samples. Machine-learning algorithms will differentiate between safe and malicious applications based on these attributes and the features that assist us in the process include the Application name, downloaded file type, RAM use, downloaded time, the behaviour of the device after download, automated or manual installation, permissions requested during installation confirmed by and sources, whether or not a digital signature was detected, the browser, passes or fails the browser firewall, website name, or source of the downloaded program, and, malicious or not being the target

	app_name	file_format	ram_usage	download_time	behaviour_of_the_device	automatic_or_manual	permissions_taken	verified_by	digital_signature_found
0	Entreet	PDF	48	Saturday, December 11, 2021	Abnormal	Automatic	User	microsoft store	no
1	Man Mobile	EXE	97	Saturday, February 20, 2021	Normal	Automatic	Others	not known	no
2	Yinap	ZIP	77	Wednesday, August 11, 2021	Normal	Manual	Others	McAfee	yes
3	Andes Shop	EXE	57	Monday, October 25, 2021	Abnormal	Manual	User	McAfee	yes
4	Orly	EXE	56	Monday, August 2, 2021	Abnormal	Automatic	Admin	not known	no

behaviour_of_the_device	automatic_or_manual	permissions_taken	verified_by	digital_signature_found	browser	passed_browser_firewall	website_name	malicious
Abnormal	Automatic	User	microsoft store	no	Firefox	no	youtube	0
Normal	Automatic	Others	not known	no	Edge	no	SnapFiles	0
Normal	Manual	Others	McAfee	yes	Yahoo	yes	nytimes.com	0
Abnormal	Manual	User	McAfee	yes	Tor	yes	FileHorse	0
Abnormal	Automatic	Admin	not known	no	Firefox	yes	fr.wikipedia.org	0

variable.

Figure 10: data set containing information about apps

B. Algorithms used

I implemented several machine algorithms in order to improve efficiency and accuracy like

- **Gradient boost**

The machine learning boosting system known as gradient boosting represents a decision tree for large and complex data. It is predicated on the idea that the next model will lower the overall prediction error when combined with the previous set of models. Decision trees are used to make the most accurate predictions. The gradient boosting method is also known as the statistical predictive algorithm. Even though it allows for the generalization and optimization of divergent loss functions, it still functions largely in the same way as earlier boosting methods. Processes for classification and regression frequently use gradient boosting.

```
Classification Report for train
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     31367
           1       1.00      1.00      1.00     31561

    accuracy                           1.00     62928
   macro avg       1.00      1.00      1.00     62928
weighted avg       1.00      1.00      1.00     62928

Classification Report for test
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13430
           1       0.99      0.62      0.76       149

    accuracy                           1.00     13579
   macro avg       0.99      0.81      0.88     13579
weighted avg       1.00      1.00      1.00     13579

Train ROC:  0.9996274937148031
Test ROC:  0.9990420172891362

Train Accuracy Score:  0.9984267734553776
Test Accuracy Score:  0.9957286987259739

Train F1 Score:  0.9984267697173035
Test F1 Score:  0.879087863189918
```

Figure 10: Results of gradient boost algorithm

- **Logistic regression**

Logistic regression is a well-known Machine Learning algorithm from the Supervised Learning method. It forecasts the categorical variables based on a set of unconventional variables. A categorical dependent variable's output is predicted using logistic regression. As a result, the outcome must be categorical or discrete. It can be zero or one, true or False, and so on, but rather than presenting the actual values like 0 and 1, it presents the probability values that fall between zero and one.

```
Classification Report for train
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     31254
           1       1.00      0.99      1.00     31674

    accuracy                           1.00     62928
   macro avg       1.00      1.00      1.00     62928
weighted avg       1.00      1.00      1.00     62928

Classification Report for test
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     13383
           1       1.00      0.47      0.64       196

    accuracy                           0.99     13579
   macro avg       1.00      0.74      0.82     13579
weighted avg       0.99      0.99      0.99     13579

Train ROC:  0.9996274937148031
Test ROC:  0.9990420172891362

Train Accuracy Score:  0.9966628527841342
Test Accuracy Score:  0.992414758082333

Train F1 Score:  0.9966628156194084
Test F1 Score:  0.8198826009727318
```

Figure 11: Results of logistic-regression algorithm

- **Random Forest**

Random Forest is a popular supervised machine learning technique. It can be used in machine learning to solve classification and regression problems. It is based on the concept of ensemble learning, which is also the method of combining multiple classifiers to address a complex problem and improve the model's performance. Random Forest is a classifier that averages different decision trees on different subsets of a given dataset to improve the dataset's projected accuracy. Rather than relying solely on one decision tree, the random forest forecasts the correct outcome by taking into account the predictions made by each tree.

```
Classification Report for train
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     31426
           1       1.00      1.00      1.00     31502

    accuracy                           1.00     62928
   macro avg       1.00      1.00      1.00     62928
weighted avg       1.00      1.00      1.00     62928

Classification Report for test
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13460
           1       0.95      0.74      0.83       119

    accuracy                           1.00     13579
   macro avg       0.97      0.87      0.91     13579
weighted avg       1.00      1.00      1.00     13579

Train ROC:  0.9996274937148031
Test ROC:  0.9990420172891362

Train Accuracy Score:  0.9993961352657005
Test Accuracy Score:  0.9973488474850872

Train F1 Score:  0.9993961350454996
Test F1 Score:  0.9144263369506308
```

Figure 12: Results of random forest algorithm

C. User interface with Streamlit

Streamlit is an open-source Python framework that allows users to construct and share visually appealing, one-of-a-kind online apps for data processing and artificial intelligence. With the aid of this software, complex data apps may be designed and published.

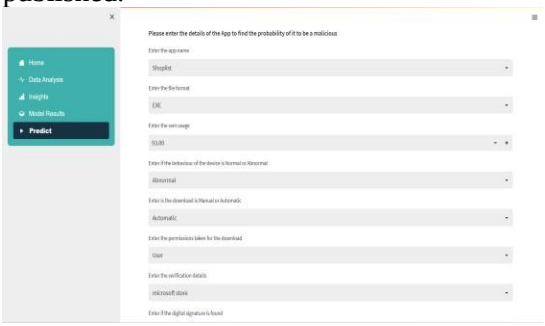


Figure 13: the user interface of running with a machine learning model

VI. Results

Developed a machine learning model that predicts malicious applications with an accuracy of 99%. The random forest algorithm is chosen as the champion model based on the highest f1-score for the test dataset. To determine how safe the downloaded application is, the user must provide specific information on the streamlit interface built for the same.

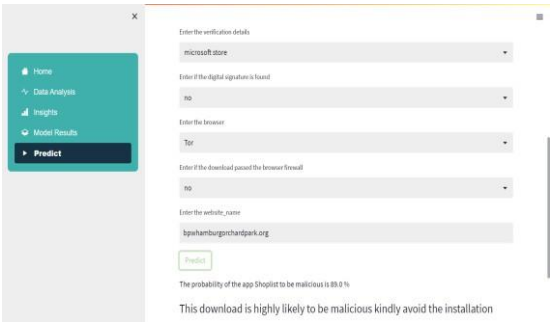


Figure 14: final result of the proposed system

VII. Conclusion

This paper described numerous hacking methods and their defenses from diverse angles. Hackers must be kept out of the network in order to secure sensitive data. The existing system is expensive and time-consuming compared to the proposed system. Machine learning plays a key role in predicting the threat level of an application without even installing the application. I believe this paper helps readers enhance their understanding of cybersecurity and address security vulnerabilities in their computer operations. It also assists in the transmission of knowledge about emerging security concerns. Prevention is preferable to cure. Keep an eye out for cyber criminals.

VIII. References

[1] Yu, W., Yalin, Y., & Haodan, R. (2019, October). Research on the technology of trojan horse detection. In 2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA) (pp. 117-119). IEEE.

[2] Al-Asli, M., & Ghaleb, T. A. (2019, April). Review of signature-based techniques in antivirus products. In 2019 International Conference on Computer and Information Sciences (ICCIS) (pp. 1-6). IEEE.

[3] Asish, M. S., & Aishwarya, R. (2019, March). Cyber security at a glance. In 2019 Fifth International]
Conference on Science Technology Engineering and
Mathematics (ICONSTEM) (Vol. 1, pp. 240-245). IEEE.

[4] Rathore, H., Agarwal, S., Sahay, S. K., & Sewak, M. (2018, December). Malware detection using machine learning and deep learning. In International Conference on Big Data Analytics (pp. 402-411). Springer, Cham.

[5] Namanya, A. P., Cullen, A., Awan, I. U., & Disso, J. P. (2018, August). The world of Malware: An overview. In 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud) (pp. 420-427). IEEE.

[6] Bhardwaj, A., & Goundar, S. (2020). Keyloggers: silent cyber security weapons. Network Security, 2020(2), 14-19.

[7] Parthy, P. P., & Rajendran, G. (2019, October). Identification and prevention of social engineering attacks on an enterprise. In 2019 International Carnahan Conference on Security Technology (ICCST) (pp. 1-5). IEEE.

[8] Kunwar, R. S., & Sharma, P. (2016, April). Social media: A new vector for cyber-attack. In 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring) (pp. 1-5). IEEE.

[9] Aslan, Ö. A., & Samet, R. (2020). A comprehensive review on malware detection approaches. IEEE Access, 8, 6249-6271.

[10] Idika, N., & Mathur, A. P. (2007). A survey of malware detection Purdue University, 48(2), 32-46.

[11] Creutzburg, Reiner (2017). "The strange world of keyloggers - an overview, Part I". Electronic Imaging. 2017 (6): 139–148.

[12] Wajahat, A., Imran, A., Latif, J., Nazir, A., & Bilal, A. (2019). A Novel Approach of Unprivileged Keylogger Detection. 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET).doi:10.1109/icomet.2019.8673404.

[13] Rohith, C., & Kaur, G. (2021, April). A Comprehensive Study on Malware Detection and Prevention Techniques used by Anti-Virus. In 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM) (pp. 429-434). IEEE.

[14] Nallaperumal, K. (2018). CyberSecurity Analytics to Combat Cyber Crimes. 2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). doi:10.1109/iccic.2018.8782430

[15] Alshamrani, S. S. (2022). Design and Analysis of Machine Learning Based Technique for Malware Identification and Classification of Portable Document Format Files. *Security & Communication Networks*, *2022*.