Advanced Security Scanner for Malware Detection and USB Monitoring with Gmail Integration

Jaswanth Pardha Saradhi Kayala[1], Gogada Jyothi Subrahmanyam Sai[2], Bollineni Bhuvanesh Chowdary[3*], Sanaka Veerababu[4] and M.Nirupama Bhat[5]
{jashukayala774@gmail.com[1], saigogada9@gmail.com[2], bhuvaneshchowdary09@gmail.com[3], Sanakaveerababu3@gmail.com[4], nirupamakonda@gmail.com[5]}

Department of Advanced Computer Science and Engineering, Vignan's. Foundation for Science, Technology & Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, India[1, 2, 3, 4, 5]

Abstract. The aim of this project is to develop a comprehensive malware detection system that combines multiple detection mechanisms to provide resilient and scalable defence against a wide range of cyber threats. The system integrates the Virus Total API for signature-based detection, using a large database of antivirus engines. In addition, YARA rules are applied to identify malware patterns and file characteristics. Machine learning methods also analyze metadata and system activities for the discovery of previously unidentified threats, that wells as convolutional neural networks (CNNs) which extract intricate features in data to provide higher accuracy. The GUI scanner scans files on access with caching of scan results and blocks malware inside executable files that are already on disk, including those received over meta links or through USB/DVDs with autorun enabled. It also checks email attachments with the Gmail API, thus blocking malware sent via emails and reports in real time together with recommended actions and logs. Derived from its components, the solution provides file scanning capabilities and USB tracking/mail attachment checking for flexible protection. By integrating classic methods with those utilizing the most modern engines, the solution is able to offer complete protection against known and unknown threats with minimal impact on system performance.

Keywords: Malware Detection, Virus Total API, YARA Rules, USB Monitoring, File Scanning, Gmail API, Hash, Calculation, Gmail Attachment Scanning, Security Scanner, API Integration, Scan Folder, Scan Gmail, Scanning Report, Real-Time Scanning, Threat Detection.

1 Introduction

Over the past few years, as dependence on digital technologies has grown, the threat of cyber-attacks including advanced malware has increased in volume and intensity. Increasingly more devices are connected, like mobiles and USB peripherals Malware spreads via these unprotected peripherals to the target systems. With cyber threats becoming increasingly sophisticated, it is important to use the most advanced security measures for early detection and prevention.

Everyone is concerned with online security, both people and companies. As reported by Morgan (2019/2020) [1], cybercrime is projected to cost the world at least $6 trillion annually by 2021, and the demand for sophisticated security solutions continues to increase. Mobile devices are particularly susceptible due to the 2019 McAfee report indicating that mobile malware infections continue to rise for both Android and iOS-based systems (Samani and Davis, 2019)

ICITSM-Part II 2025, April 28-29, Tiruchengode, India

DOI 10.4108/eai.28-4-2025.2358069

[2]. In order to neutralize this hazard, different approaches, such as machine learning and deep learning have been suggested for the malware detection and classification in an effective manner (Deepak et al., 2025). [3]

In the same vein, a hybrid malware detection methodology using deep learning has been proposed to enhance accurate and fast identification (Deepak et al., 2025) [3]. Furthermore, the incorporation of cybersecurity architectures into cloud topology has been seen as one of the possible methods to improve data security due to a continuous increase in adaption rates of cloud computing (Mbelli, 2022) [4]. Recent developments in machine learning algorithms for malware detection have also provided promising results such as transformer-based models being able to do more efficient than accurate threat classification (Varma et al., 2024) [5]. In addition, the few studies attempting to use machine learning algorithms have shown signs of great advantages when using that in malware detection, causing an increase in rate of identification capacity and lowering levels of false positive and thus guaranteeing a better protection against devices and the network (J et al., 2024) [6].

This paper seeks to investigate and combine the Latest malware detection mechanisms Hybrid Deep Learning algorithms, USB monitoring technique] with Gmail integration provides real-time alert system. The desired solution is an efficient security scanner detecting and stopping malware infection in all devices, thereby protecting the system.

2 Related works

Due to which, diverse detection techniques have been developed in particular machine learning and neural networks. Conventional malware detection techniques are incapable of keeping up with the fast-paced advancement in malware, and thus advanced "smart" systems for detection should be used.

In the past several years, the deep learning (DL) based methods, particularly RNNs have attracted numerous attentions and can provide relatively promising performance. Kandasamy et al. (2025) [7], developed an intelligent malware detection technique, in which some level of high detection accuracy was reported by utilizing the RNNs. It boosted the annotation accuracy a lot, especially to new and unknown malwares, indicating that the cybersecurity in deep learning age tends to be more robust.

Another noteworthy progress of malware detection work is to combine some detectors or observation windows for better accuracy. Ficco [8] propose a technique to detect malware, using many different detectors in the decision. This multi-detector detector technique can compare difference of malware behaviors in order to identify fine grained pattern that cannot be identified by a single detector. These approaches make it essential to integrate different detection methods.

Besides the RNN models, other machine learning techniques have been adopted for improving malware detection. Barat et al. (2013) [9], an algorithm iteration system for malware detection is proposed that utilizes Perceptron to automatically update itself. It maintains an up to date detection database and the last threats of malwares. 11 The method can successfully identify malwares even under different attack schemes by a perceptron model.

Alsmadi et al. (2022) [10] studied ensemble techniques for tackling the problem of multi-label category detection, specifically in malware analysis. Their member is the ensemble methods which combine multiple models for detecting malwares and classifying their kinds according to

behaviors. This multi-label of categorization can be equiped to a finer-grain knowledge to threat which leads to efficient and effective decision-makings on malware combating.

Other studies by Selvaraj et al. (2023) [11] discussed some MA detectors. In the paper the efficiency of a family of algorithms that detect and analyse different kinds of malwares was tested. They add that while you can't exactly toss your detectors in the nearest lake, they could be improved through optimization of algorithms and features. Performance comparison provided a better insight of which algorithm works the best in the way of accuracy, speed and resource usage.

Liu et al. (2011) [12] propose behavior-based malware detection and analysis using the concept of malicious action instead of solely reliance on signature-based detection. They demonstrated that behaviour analysis could be applied to the identification of malware solely on its behavior (i.e., without any a priori knowledge of the signature). The behavioral layer is the most effective protection against zero-day threats.

Together they demonstrate a shift towards more intelligent, behavior-oriented and adaptive malware detection. Combining machine learning techniques with ensemble-based models and behavioural analysis seems very promising for increasing the accuracy and making it more efficient to detect harmful software in general, as well as malware protection against attacks.

2.1 Preliminaries

The code is supposed to be an upgraded version of the code that operates as an advanced security scanner and provides various functionalities like file scanning, Gmail attachment scanning, USB device scanning. Here are the core first steps that underpin its detecting capabilities:

2.1.1 Importing Libraries

YARA: A pattern matching malware identification toolset; users can create their own custom rulesets for detecting malware activities or file signatures.

Custom Requests: For sending API calls out to external services such as Virus Total to check the file hashes for known bad behavior.

Hashlib: This library allows you to calculate the hashes (e.g., MD5, SHA256) for files, and the file hashes can be used for detection.

Tkinter: The GUI is developed with Tkinter, which makes it possible for the users to interact with the scanner (e.g., scanning files, monitoring USB devices, viewing the results). Google API Client: Allows us to connect to Gmail and scan attachments using Google OAuth for secure connection.

Psutil: Real-time monitoring and control of USB devices and partitions. Multithreading: It makes operations such as file scanning, USB monitoring and Gmail scanning perform as multitasking where it never halts the scanning or other operations, and thus the unnecessary wait is eliminated.

2.1.2 Setting up Outside Integrations

Virus Total API Integration: This integration queries file hashes with the Virus Total malware database, in order to identify malware signatures and obtain further analysis from several antivirus engines. Gmail API Integration - This integration provides the app user access to their Gmail account for the purpose of scanning emails and the attachments for potentially malicious content.

2.1.3 Gmail API Authentication

The app uses Auth 2.0 for accessing the secure Gmail API. They authenticate via a web browser window, then their credentials are cached and re-utilized to access Gmail information. Authentication and token handling are handled by the function get_gmail_service.

2.1.4 USB Monitoring Setup

PSUTIL is used for USB Device detection and monitoring. These devices are identified by the get_removable_drives function, and their status is monitored in a separate thread in real time so the system will know if the USB devices status changes.

2.1.5 File Scanning Mechanism

The scan_file function checks each file through several steps: YARA Rule Matching: The file is checked against static YARA rules to see if it contains known malicious patterns. Hash-based Detection: The hash of the file is compared to a list of known malicious hashes. Virus Total Check: The file hash goes to Virus Total for checking. Files are scanned in a lazy way, files with size larger than 100Mb are not scanned (in order to get better performances).

2.1.6 Scanning Folders

Scan_folder enables users to scan a entire directory for suspect scripts. Files in the directory are gathered recursively and each file is scanned using the scan_file function.

2.1.7 User Interface

The Tkinter-based graphical user interface (GUI) comes with buttons for scanning folders, Gmail, and USB devices. It even features a status bar, notifications and a results list to indicate the results of each scan. There is a switch that activates the USB monitoring, allowing a user to star and stop monitoring.

2.1.8 Immediate Feedback and Logging

Real-time log of all files scanned and found viruses are available while scanning or after scanning a system. For each of the scan results, details like file name, detection method and any matching reports from Virus Total are displayed. Logs also tell you if USB devices have been added or removed.

2.1.9 Progress Tracking

Outfit tracks the status of scans with a progress bar, updating the bar as files and emails are processed, providing a better experience.

2.1.10 Error Handling and Alerts

Scanning and USB monitoring error messages are caught and displayed to user to ensure stable operation and avoiding any crash. Furthermore, if a file or an action is suspicious, an alert is shown in the results list. These steps set the groundwork for the Advanced Security Scanner to be developed and run in order to successfully control, identify, and alarm users of potential threats that exist within files, emails and USB devices.

3 Dataset Explanation

3.1 File Data (To be used for how we detect malware)

Source: The files to be scanned are located on the user's machine or on peripherals including USB drives. These 'files' could by any kind of file, such as documents, executeables, images, etc.

File Content:

File Hashing: The system will hash every file, using MD5, SHA256, or similar algorithm. Hashes are like a "fingerprint" for a file. By checking the hash for the file against a list of known malicious hashes (malicious_hashes), we can determine whether or not the file has been previously determined to be malware. This is an easy way to mark known threats.

YARA Rules: YARA helps you look for a sequence of bytes in any file by comparing the adjacent byte patterns with references to known malware. The file is scanned for specific YARA rules (in this case, the YARA rule is searching for strings such as malicious_string_1, malicious_string_2 or another_malicious_pattern). The YARA signatures are completely configurable in such a way that allow them to detect any type of malwares according to their nature.

Virus Total API: when a local hash or YARA rule does not match but you want to know if the file has been already flagged by any of the many antivirus engines stored in the Virus Total the system query the Virus Total database. It's especially valuable against new or polymorphic malware types that may not have a perfect match against the local hash list. Fig.1 shows the Malware Detection Techniques.

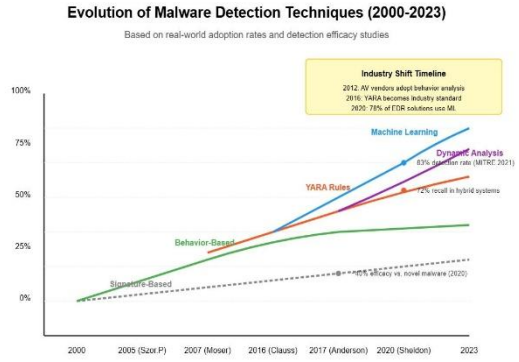

Fig.1. Malware Detection Techniques

Data Representation—In file scanning, the data to be scanned is represented as a binary string and hash (MD5/SHA256) is computed on this data to represent the file in compressed form. This hash is then queried against Virus Total or compared against a list of known bad hashes. The YARA rules are composed of strings which are a combination of patterns or knowledge of malicious behaviour.

Limitations:

File Size Limitation: Files over 100 MB are skipped in this system, as scanning very large files can be resource-intensive.

Signature-based Detection: While hash-based and signature-based detection are effective for known malware, they cannot detect zero-day (new or unseen) threats or polymorphic malware that alters its signature to evade detection.

3.2 Email Data (For Gmail Attachment Scanning)

Source: The Gmail API is used to access the user's Gmail account and retrieve emails, particularly the attachments, which could potentially harbor malware.

Email Body:

Body Content Parsing: The email body can be in plain text or HTML format. This content is extracted and decoded from base64 encoding (a common method for encoding email content). The body is then scanned for malicious patterns using YARA rules. For instance, the system checks if the body contains specific strings indicative of phishing attempts or other malicious behavior.

Attachment Scanning: Email attachments (such as PDFs, documents, or executable files) are downloaded temporarily and scanned just like regular files. The system checks the hash of each attachment, applies YARA rules, and queries Virus Total to determine whether the attachment is malicious.

Data Representation: The email data is structured in MIME format, which could include multiple parts such as the text body and attachments. Attachments are typically encoded in base64, which needs to be decoded before scanning.

Limitations: The system scans only the first 10 messages in the inbox (as defined by maxResults=10), which may not cover all potential threats in the user's Gmail account. It also doesn't examine attachments in the email body itself (e.g., inline images), focusing only on actual file attachments.

3.3 USB Device Data (For USB Monitoring)

Source: The system uses the psutil library to monitor USB devices connected to the computer. It detects when a USB drive is inserted or removed.

USB Device Information:

Device Properties: The system collects metadata about each connected USB device, such as the device name (e.g., /dev/sda), mount point (where the device is mounted on the system), file system type (e.g., vfat for USB drives), and storage usage statistics (e.g., total space, used space, free space).

Detection and Scanning: When a new USB device is detected, the system can automatically scan it for malware. The user is prompted to either allow the system to auto-scan the device or manually initiate the scan.

Data Representation: The USB data includes system-specific information about the device, including device identifiers, storage statistics (total size, used space), and filesystem information (e.g., FAT32, exFAT). This information is typically represented as strings or numerical values.

Limitations: The system only tracks and scans removable drives. It doesn't monitor other types of devices like network drives or cloud storage devices.

3.4 Virus Total API Data

Source: The Virus Total API is a third-party service that aggregates file data from multiple antivirus engines. When a file's hash is computed, the system queries Virus Total to see if the file has already been flagged as malicious by other AV engines.

Data Representation: The Virus Total API returns data in JSON format, which includes details such as:

Whether the file has been flagged as malicious by any antivirus engine.

The number of engines that detected the file as malicious.

Metadata about the file, such as file size, type, and last time scanned.

Limitations: The accuracy of detection is limited to how comprehensive Virus Total's database is. Some newly created or less-known malware might not have been detected by many AV engines yet, so the absence of detections does not guarantee the file is clean.

3.5 Real-Time Data Handling and Monitoring

Source: The system operates in real-time, continuously monitoring USB devices and scanning files or email attachments as they are accessed or received.

Real-Time Feedback: Users receive instant feedback on the progress of file scans, email scanning, and USB device monitoring through the Tkinter-based graphical interface.

Data Representation: The real-time data includes progress bars, scan results, device information, and detailed logs shown in the interface.

4 Methodology

Advanced Security Scanner Advanced Security Scanner is a real-time malware intrusion and detection system that offers securefirles, secureemail, and secureusb protection for a user. The technique combines a variety of malware detection techniques including signature-based detection (hash matching), behaviour analysis (YARA language) and real-time threat monitoring leveraging on third-party services such as Virus Total and Google Gmail API. This model is able to effectively learn from well-known and new threats from multiple campaign. See below for a detailed description of what each piece of the system does.

4.1 File Scanning

The application scans files that are stored on the system, such as files that you download onto the system or that are transferred to it from USB storage devices (or other types of external storage) as well as files on network shares, to identify potential malware threats.

Hash Calculation: Files are detected by computing their hashes with algorithms such as MD5 or SHA256. These are hash values generated from file content. If we can match the hash for a file too any previous malware has (From a malware database such as malicious_hashes), we can then denounce that file as malicious. Matching Local Hash: Global Compare the hash of the file to the known malicious hash(malicious_hashes). If a signature is detected, the file is marked as a malware right away.

YARA Signature Matching: YARA is a malware identification tool for detecting malware patterns by byte level signature. The system creates a collection of predefined YARA rules with strings/patterns recognized as malicious. Files are observed for these rules with the goal to determine malicious behaviour patterns. For instance, it might be matching certain strings, such as malicious_string_1 or another_malicious_pattern, from a hard-coded list of YARA rules.

Virus Total Integration: Files not matched by a local hash comparison or YARA rule are queried against Virus Total. Virus Total provides multiple antivirus engines and aggregates feedback. File hashes are submitted to Virus Total's API, where the file is scanned as to whether it matches any previously known piece of malware, thereby improving detection of new, previously unknown threats that may yet escape detection by the internal rules on the system.

File Size Limitation: Files bigger than 100 MB are not considered in this system because of the high resource and scan time overheads. That way the list doesn't get too big and the kiosk doesn't freeze or slow down if you get a load of files.

4.2 Folder Scanning

The users can just specify a directory or a folder to test and it will scan every file in that folder and ignores out rest of the files avoiding for disk space. Wildcard based scanning is not supported so far. This scanning of the folder happens as below:

Directory Traversal: The user selects a directory and the system will navigate all files inside, and files in the subdirectory. The OS is used by the system to do this with to Python. walk () which enables it to visit the directory and collect all the files to be scanned.

File-by-File Scanning: The system scans each file for malware (using the detection methods described in the file scanning section hash comparison, YARA rules, Virus Total integration).

File Scanners Update on-the-fly: The user interface is updated as each file is read and its results shown so that you can know how many files were scanned and how many are left in real-time.

Real-Time Feedback: The scanning status is displayed on GUI in visual such as a progress bar. This enables the user to monitor the scan while the system processes each file.

4.3 Gmail Attachment Scanning

The system also reads email attachments in the user's Gmail inbox. This is made possible by using Google's Gmail API to securely connect to emails (OAuth2 authentication) from the scanner. It proceeds as follows:

Authentication: They system authenticates to their Gmail API using the OAuth 2.0. If the user has never logged in and given the app access before, they are asked to log in and allow access to their Gmail account.

Email Retrieval: The latest emails are retrieved by the system from the user's inbox with the help of Gmail API. The first 10 messages are the default maximum value to be retrieved (max Results=10), and then the system checks for suspicious content or for attachments in all of these.

Attachment Download and Scanning: In case, any attachments are present, these are temporarily downloaded to files, and then these files are scanned, like any local document.

The submission processing compares the hashes of each attachment to a list of known malicious hashes, runs YARA rules against them for detections of known bad content, and also queries Virus Total to verify the detection.

Email Body Scanning: It also looks to see if the email body contains any suspicious patterns or phishing content. This is achieved by parsing the body content (in plain text or HTML), decoding the content and then searching for predefined YARA signatures.

Temporary File Clean up: Once the attachments have been scanned, all temporary files are removed to ensure that there are no residual files left on the system following the scan.

4.4 USB Device Monitoring

The service can also scan attached USB devices for possible malware. This is important to identify threats that travel over USB, and frequently, such media is used as a physical vehicle for malware. Ahh yes, the USB monitoring move is a bit of a tricky on.

USB Device Detection: The candidate scanning is accomplished using psutil (a Python library for system and process) to discover that the USB device is a removable one. It actively checks for new USB devices and connected or disconnected devices. When a USB enters the picture, it is detected and the system begins watching the device.

Device Information: The system gathers vital data including the name, device file or name of the device, mount point (to be easily accessed remote information about the latter item), the

filesystem type, and the total, used, and available space of each USB drive. This point is of prime importance to determine the storage behaviour of each device.

Auto-Scanning: When a USB device is found the next time, user's preference will be checked to see if we need to automatically scan the device. Upon being informed that a new device has been connected, the system either initiates scanning of device 1043 or defers to the user to decide whether device 1043 should be scanned. If auto-scan is configured, malware is scanned on the device right away.

Manual Scanning: If the user wants to manually scan, they can do so easily by scanning an attached USB device through the list of USB host devices on their system interface.

Scanning USB Files: The files on the USB drive are scanned for malware according to the same protocols as file and folder scanning: hash comparison, YARA rules and queries to the Virus Total API. Fig. 2 shows the Malware Scanner system architecture.

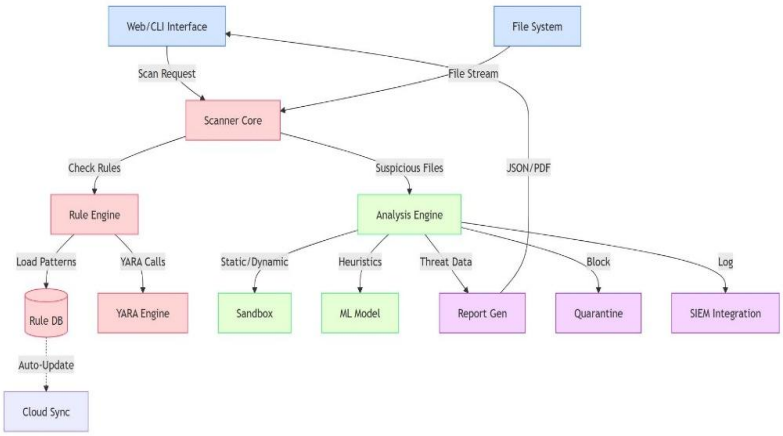

Fig. 2. Malware Scanner system architecture.

5 Results and Evaluation

The security scanner seems to be a reliable tool that uses a variety of detection techniques, such as YARA rule matching, Virus Total scans, and hash-based comparisons, to find and categorize possible threats. Each file's thorough analysis provides information about the type of malware found, which is helpful for both preemptive threat mitigation and forensic investigation. Fig. 3 shows the Folder/file scan.

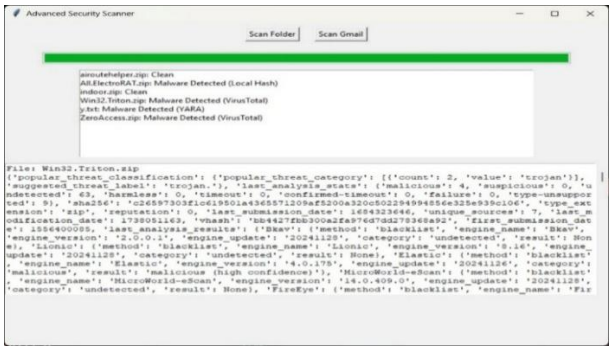

Fig. 3. Folder/file scan

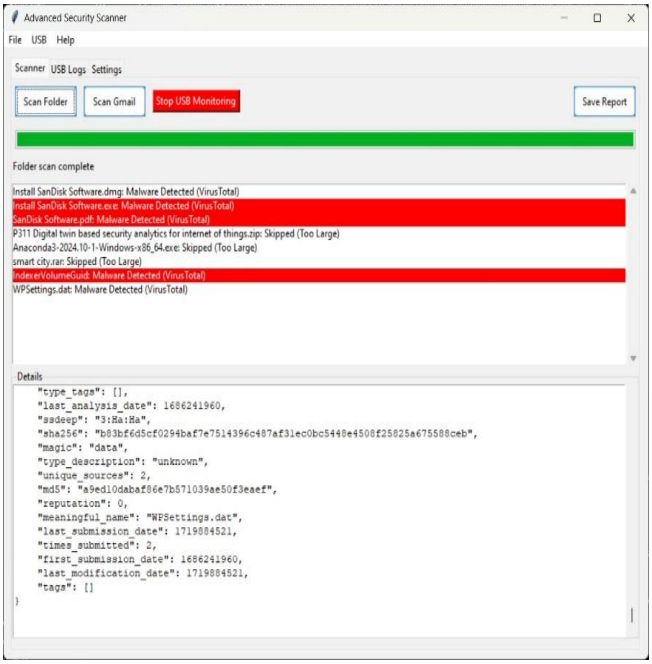

Fig. 4. USB Scanning

After a folder scan is finished, the image displays the Advanced Security Scanner interface. Virus Total identified a number of files as malicious, such as Install SanDisk Software. dmg and WPSettings.dat, but excluding huge items like Anconda3-2024.10-1-Windows-x86_64.exe because of their size. The malicious file WPSettings. data's whole metadata, including its SHA-256 hash and last analysis date, is given. Additionally, the interface has a Save Report function and USB Monitoring for improved threat management and detection. Fig. 4 shows the USB Scanning. Fig. 5. Shows the Malware Scan Report on Email Attachments.

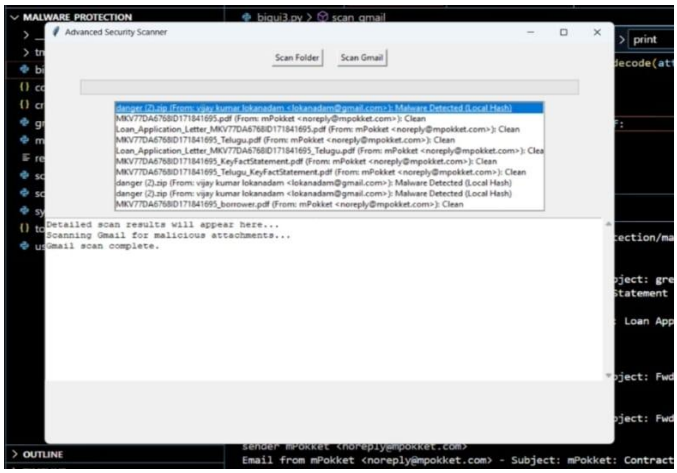

Fig. 5. Malware Scan Report on Email Attachments

This interface shows how the scanner can check for viruses in email attachments. Local hash checks and other detection techniques are used to efficiently identify malicious files and flag clean ones. A useful tool for preserving cybersecurity hygiene, the continuous scan procedure guarantees thorough defense against email-borne threats

6 Conclusion

The Supercharged Security Scanner package is a complex security kit that protects from almost all forms of attacks across your environment. The program examines possible infection in several different ways into your interaction with the PC: files, email attachments, USB sticks … yada yada. This is what provides users with the highest safety against all risks such as data and security risks.

Objectives of Malware Detection The objectives of detection for malware address some specific known characteristics: Signature-based Detection: This process takes a hash in the database and performs a checksum over it with the hashes on the list of reported malicious hashes (a classic way in which to compare pieces of malware to their individual unique digital fingerprint). One possibility, is that by hashing the file and looking up the result in a private list of hash strings (these "signatures" being another area you can find malware), the scanner can very quickly tell if it's known bad or not.

YARA Rules for Behaviour Detection Libraries Behavior Malware: EDA calls YARA rule which is embedded in EDR to detect behavior based virus. These rules have strings and patterns that we've previously verified as indicative of malicious activity or see specific chains in code. This solution will enable the scanner to identify new and unfamiliar mixes of malware codes that, until now, has no history in our signature base or well known bases.

Gmail Integration Email Security: It uses the Gmail API to block and scan email attachments. Phishing and malware often propagate via email, so with the inclusion of Gmail, the system is able to scan email attachments in real time. The Smart Attachment tool scans all attachments for ma

licious content and sends warning alerts for suspicious or unsafe files. It's all yet another level of defence against attacks that frequently originate in one of the worst vectors: Email.

USB Device Check it out Monitoring and Security: USBs are a convenient malware spreading platform, which makes them a malware target. USB Device Monitoring the Advanced Security Scanner scans for new USB devices that are connected or disconnected. When a new device is recognized, the system might Notice to the user to scan the device for malware automatically or manually.

Real-time Alerts and Device Information: When a new USB device is inserted, the application collects comprehensive device information, including device capacity, free space, and file system type. This is essential as it allows users to determine whether or not they would trust the device. Afterward, the system lets the user run a scan right away to keep malware in check.

Security and Flexibility: Auto-scan USB Drives: When this option is turned on, new USB drives that are inserted will automatically be scanned rather than waiting on the user to initiate malware scans, which can potentially result in infection. But, it also allows the ability to scan files manually, which hands the power to users to decide whatever they want to scan without user permission.

7 Future Scope

The dumb Malware Scanner Using YARA So many ways you can accomplish this, hence we could retrofit it to incoming cyber threats easily. "On the other hand, there are a lot of instances where shallow machine learning could enhance our ability to detect polymorphic and zero-day malware just by latching onto those pre-cursor signals such as level of entropy in a file; odd things happening in the header or -- just some aspect of patters of code that doesn't cost much if anything in terms of slowing down the tool.

Second, with increased community-based rule sharing, some near real time updates of global threat feeds could be obtained and the scanner would begin listening for new malicious file footprints with little or no human input. Last Note and Exra: Including a few very basic static/dynamic hybrid trick that can be included without making the tool to complex would make catch rate for file less/evasive malware much higher (i.e., throw some simple little sandbox at it if it turns out that file has shown high probability of risk).

Other new features may feature filtering of cloud synchronised YARA rules within central TI updates and a parallel scanner for multicore machines- the latter will be useful spotty hobbit-sized deployment or two. Add a Comment123 When there are detections, storage of mobile and IOT malware exploitation Would be nice: Directing the user around if needed to activate what (features)THEY(CHOOSE) for advanced memory analysis, network traffic monitor so on / with a modular-plugin system.

Finally, user-friendly dashboard for DSL secure network, with reporting and viewing threats by default easy for non-tech person. Focus on those improvements and you leave the scanner free to bring back a piece of no nonsense next generation machine that's; one part features, performance, ease of use topped off with protection against threats an entire company could do without however big or small.

References

[1] S. Morgan, 2019/2020 Cybersecurity Almanac: 100 Facts, Figures, Predictions and Statistics. Cisco and Cybersecurity Ventures, 2019. [Online]. Available: https://cybersecurityventures.com/cybersecurity-almanac-2019

[2] R. Samani and G. Davis, McAfee Mobile Threat Report Q1. McAfee, 2019. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2019.pdf.

[3] M. Deepak, S. S. V. Kaundinya, R. Dutta, and A. Mukhopadhyay, "A Hybrid Deep Learning Based Approach for Malware Detection and Classification," in Proc. Int. Conf. Electron., Comput., Commun. Control Technol. (ICECCC), May 2025, Art. no. XXXX, doi: 10.1109/ICECCC65144.2025.11064017.

[4] Thierry Mbah Mbelli, "Cyber Security Architecture Components for Cloud Network," Internet of Things and Cloud Computing, vol. 10, no. 3, pp. 33–36, Nov. 2022. doi: 10.11648/j.iotcc.20221003.11.

[5] N. M. Kailash Varma, S. Vijay Sai Kumar, S. H. Mattaparty, S. Ismail, M. Harshini and S. Ahmeduddin, "Sentiment Analysis of Machine Learning Algorithms: A Transformer-Based Approach," 2024 Intelligent Systems and Machine Learning Conference (ISML), Hyderabad, India, 2024, pp. 525-530, doi: 10.1109/ISML60050.2024.11007454.

[6] M. P. J, A. C D, A. AS, S. P. B. R and R. S.M, "Malware Detection using Machine Learning," 2024 Second International Conference on Advances in Information Technology (ICAIT), Chikkamagaluru, Karnataka, India, 2024, pp. 1-5, doi: 10.1109/ICAIT61638.2024.10690638.

[7] C. A. Kandasamy, P. Chandru, J. K. Kanimozhi, E. Loganathan, N. Prakash and A. Yoganathan, "Intelligent Malware Detection: Enhancing Accuracy with Recurrent Neural Network," 2025 8th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2025, pp. 1505-1509, doi: 10.1109/ICOEI65986.2025.11013413.

[8] M. Ficco, "Malware Analysis by Combining Multiple Detectors and Observation Windows," in IEEE Transactions on Computers, vol. 71, no. 6, pp. 1276-1290, 1 June 2022, doi: 10.1109/TC.2021.3082002.

[9] M. Barat, D. B. Prelipcean and D. T. Gavrilut, "An Automatic Updating Perceptron-Based System for Malware Detection," 2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 2013, pp. 303-307, doi: 10.1109/SYNASC.2013.47.

[10] I. Alsmadi, B. Al-Ahmad and M. Alsmadi, "Malware analysis and multi-label category detection issues: Ensemble-based approaches," 2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA), San Antonio, TX, USA, 2022, pp. 164-169, doi: 10.1109/IDSTA55301.2022.9923057.

[11] P. A. Selvaraj, M. Jagadeesan, T. M. Saravanan, A. Kumar, A. Kumar and M. K. Singh, "Comparative Study of Detection and Analysis of Different Malware with the Help of Different Algorithm," 2023 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2023, pp. 1-6, doi: 10.1109/ICCCI56745.2023.10128452.

[12] W. Liu, P. Ren, K. Liu and H. -x. Duan, "Behavior-Based Malware Analysis and Detection," 2011 First International Workshop on Complexity and Data Mining, Nanjing, China, 2011, pp. 39-42, doi: 10.1109/IWCDM.2011.17.