IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 March 2024, accepted 14 May 2024, date of publication 20 May 2024, date of current version 29 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3402956

RESEARCH ARTICLE

Enhancing Cybersecurity With P-Code Analysis and XGBoost: A Novel Approach for Malicious VBA Macro Detection in Office Documents

CANDRA AHMADI, JIANN-LIANG CHEN, (Senior Member, IEEE), AND YI-CHENG LAI
Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei 106335, Taiwan
Corresponding author: Candra Ahmadi (D11007809@mail.ntust.edu.tw)

ABSTRACT In the evolving landscape of cybersecurity, the prevalence of malicious Visual Basic for Applications (VBA) macros embedded in Office documents presents a formidable challenge. These macros, while integral to automation, have become potent vehicles for cyber-attacks, necessitating advanced detection techniques. This study introduces a comprehensive framework employing P-Code Analysis and XGBoost, a leading-edge machine learning algorithm, to address this issue. The proposed solution synergizes static analysis of VBA source code with dynamic P-Code structural analysis, enhanced by Natural Language Processing (NLP) techniques for effective feature extraction. By integrating these methodologies, our model adeptly distinguishes between benign and malicious macros, achieving an unprecedented detection accuracy of 98.70% and an F1-score of 98.81% in rigorous testing environments. The core contribution of this research lies in its innovative approach to malicious macro detection, offering a robust framework that significantly improves upon existing methods. Additionally, the utilization of XGBoost for machine learning analysis introduces a novel application in cybersecurity defenses against macro-based threats. The results underscore the efficacy of combining P-Code analysis with machine learning for cybersecurity, marking a significant stride in the detection of sophisticated cyber threats. This study not only advances the domain of cybersecurity but also lays the groundwork for future research, advocating for the exploration of further optimizations and the adaptation of our model to combat evolving attack vectors. Recommended terms: Cybersecurity, Malicious VBA Macro Detection, P-Code Analysis, XGBoost, Machine Learning.

INDEX TERMS Cybersecurity, malicious code detection, natural language processing (NLP), office documents security, P-code analysis, visual basic for applications (VBA), XGBoost algorithm.

I. INTRODUCTION

The inception of the internet and the subsequent explosion of data have created a fertile environment for the emergence and evolution of cybersecurity challenges over the past decades. This period has been characterized by a dramatic increase in smart devices, widespread adoption of social media, and the extensive use of cloud-based services. These developments have not only paved the way for innovative digital solutions but have also introduced a complex and intricate landscape for ensuring digital security [1], [2].

This evolving threat landscape has necessitated the development of sophisticated detection mechanisms that leverage machine-learning techniques to distinguish between benign and malevolent documents with a high degree of accuracy [3]. The integration of dynamic analysis, as well as the adoption of heuristic and signature-based detection methods, have become pivotal in identifying and mitigating the threats posed by these macros. Moreover, the proliferation of such macros underscores the need for continuous education and awareness efforts aimed at end-users to prevent the initial execution of these malicious payloads [4]. This research encompasses various types of cyber threats leveraging VBA macros, including viruses, trojans, and other forms of malware. The proposed approach is designed to detect and prevent these threats with

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed Elhoseny.

high accuracy. This research focuses not only on viruses but also includes all forms of attacks leveraging VBA macros, such as trojans and other types of malware.

Given that over 57% of malware downloaders still rely on office files for dissemination, identifying malicious VBA macros becomes critical [5]. This task is further complicated by techniques such as VBA stomping, which cleverly hides these malicious macros by exploiting the _VBA_PROJECT stream and the sequence of P-code instructions [6]. Such deceptive methods underscore the sophisticated landscape of cyber threats, where attackers continuously innovate to bypass detection mechanisms. As these challenges mount, the cybersecurity community must advance its defensive strategies, employing both technological and educational tools to counteract these evolving threats effectively.

The contemporary landscape of cybersecurity countermeasures often centers on scrutinizing the VBA source code, a method that, despite its partial effectiveness, does not fully encapsulate the detection of all malevolent VBA scripts. This limitation is particularly pronounced in light of the agile modifications employed by adversaries, such as VBA stomping, designed to elude detection by obfuscating the malicious macros through manipulation of the _VBA_PROJECT stream and P-code instruction sequence [7], [8]. The persistence of such evasion techniques illuminates a significant gap in the robustness of current detection frameworks, necessitating a recalibration of our approach to uncovering these veiled threats.

Attackers are constantly refining their tactics, notably through the exploitation of Office documents and the employment of techniques like VBA stomping to conceal their nefarious activities [5]. This evolution, especially their utilization of cloud applications as vectors for malware dissemination, underscores a critical deficiency in existing security defenses. It emphasizes the urgent need for detection systems that are not only comprehensive and adaptable but also sufficiently robust to proactively counter the advanced strategies of cyber adversaries. The revelation that nearly half of all malware downloads in January were facilitated via cloud apps further accentuates the demand for security solutions capable of navigating the evolving digital threat landscape and effectively monitoring traffic originating from these platforms.

In this investigation, we unveil an advanced detection architecture aimed at significantly elevating the precision in pinpointing malicious VBA macros embedded within office documents. By seamlessly integrating analyses of both VBA source code and P-code structures, our objective is to address and surpass the constraints observed in prevailing detection strategies. Our focus extends beyond the conventional examination of suspicious function calls, behaviors, and signatures to include a detailed exploration of intricate code structures that often remain undetected by current systems.

We have intricately designed a framework that meticulously identifies 85 unique features, employing the sophisticated XGBoost algorithm for the accurate categorization of documents as either benign or malicious. This framework stands out due to its innovative use of Natural Language Processing (NLP) techniques, which skillfully process sequences of P-code instructions. This enhancement not only broadens our feature base but also significantly boosts the system's detection accuracy.

Upon conducting an exhaustive series of analyses and processes, our model showcased commendable performance in the domain of static analysis-based detection. It marks a substantial improvement over traditional methods, providing a detection system that is both more comprehensive and precise.

This methodology does more than just enhance the detection capabilities through a thorough examination of both VBA and P-code components; it also merges the power of feature extraction with the advancements of machine learning. This fusion establishes a robust groundwork for future research endeavors and practical deployments in the field of cybersecurity, offering a formidable tool against the continuously evolving spectrum of cyber threats.

II. GUIDELINES FOR MANUSCRIPT PREPARATION

This section introduces the related study of office documents, VBA macros, malicious macro threats, and Investigating Malevolent VBA Macros.

A. OFFICE DOCUMENTS

Before 2007, Microsoft Office predominantly relied on a proprietary binary file format known as the Compound File Binary Format (CFBF), which is also recognized as Object Linking and Embedding (OLE), to manage document storage. In an evolutionary leap in 2007, Microsoft introduced a novel file format, built upon XML and ZIP compression, named Office Open XML (OOXML). This initiative was strategically designed to significantly enhance document storage efficiency, ensure greater security measures, and promote interoperability across a wide range of software applications. In recognition of Microsoft's dedicated efforts towards achieving these goals, the International Organization for Standardization (ISO) formally acknowledged OOXML as a global XML standard, designating it ISO/IEC 29500 in 2008, thereby endorsing its utility in fostering a secure and efficient document management ecosystem [4].

In their comprehensive study, Ali et al. [9] explore the nuanced benefits of the Office Open XML (OOXML) format in terms of data storage and retrieval efficiencies. The researchers underscored OOXML's capability to markedly diminish file sizes through its use of ZIP compression, alongside facilitating more streamlined data recovery processes owing to its XML-based framework. This shift in document format standards from Object Linking and Embedding (OLE) to OOXML represents a significant evolution. For example, the transition saw Word documents move from the ''.doc'' file extension to''.docx'' within the OOXML schema, and Excel spreadsheets from ''.xls'' to ''.xlsx''. Lee et al. [10] corroborate these findings, noting that OOXML's detailed specification of file extensions, coupled with its structured

approach to the storage of document components such as paragraphs, tables, and images in a ZIP archive, substantially bolsters security.

While the introduction of Office Open XML (OOXML) significantly advanced the restructuring and storage mechanisms of Microsoft Office documents, enhancing flexibility and interoperability, it is not without its criticisms and operational challenges. For instance, May et al. have articulated concerns regarding OOXML's backward compatibility with legacy versions of Microsoft Office, suggesting that despite the format's innovative approach, it encounters challenges in achieving universal applicability [11]. Transitioning from the Compound File Binary Format (CFBF) to OOXML had minimal impact on functional capabilities. Notably, several extensions are compatible with both formats and support VBA macros. Specifically, the Word document ''docm'' extension retains the XML format for VBA macro codes. In contrast, macros are inexecutable in documents saved with the ''docx'' extension, unlike in the Object Linking and Embedding (OLE) format files with the ''doc'' extension, which do not impose any limitations on VBA macro functionalities [12].

B. VBA MACROS

VBA macros are integral to the Microsoft Office suite, including Word, Excel, and PowerPoint, providing a powerful programming environment [13]. These macros significantly alleviate the repetitive nature of tasks by introducing an automated approach to operations. With Microsoft Office's widespread adoption as a leading software suite globally, VBA macros are extensively utilized across diverse sectors, notably in business and finance. They also offer an interactive platform with the Windows Application Programming Interface (API), enabling developers to access and manipulate core operating system and hardware resources. Such resources include files, the Windows Registry, network connections, and input devices like keyboards. An example of VBA macro usage is presented in Fig. 1, illustrating its application in streamlining complex tasks.

```
Sub RunPowershell()

Set wShell = CreateObject("WScript.Shell")
Set wShellResult = wShell.Exec("powershell (whoami)")

MsgBox wShellResult.StdOut.ReadAll

End Sub

Sub Autoexec()
    Call RunPowershell
End Sub
```

FIGURE 1. Sample code - execution of runpowershell via VBA macro.

During its execution phase, VBA code is initially compiled into an intermediate instructional format known as P-code [14]. Figure 2 illustrates P-code extracted from a sample file, which is subsequently interpreted by a specific Windows virtual machine and integrated into the file's structural framework, akin to an opcode's functionality. Should VBA be executed again, provided there is compatibility between the VBA file version and its host, the compilation phase is skipped, enabling direct execution of the P-code. This mechanism has been leveraged by attackers to circumvent signature-based endpoint defenses. They embed malicious code within the P-code while presenting benign VBA source code, effectively masking their true intentions. This technique, identified as ''VBA Stomping,'' is thoroughly analyzed and acknowledged within the MITRE ATT&CK framework [15]. Figure 3 presents the sequence of operations characteristic of a VBA stomping attack, illustrating the sophisticated methodologies employed to exploit the dual nature of VBA code execution.

```
Line #0:
    FuncDefn (Private Sub HWkknK(AZExHptj As String, NFtovv As
String, Right As String, uTUBY As String))
Line #1:
    LitStr 0x000B "G-yJOFaEUCg"
    LitStr 0x0013 "DRDhes)]PUuLP^@Q*jk"
    Add
    LitStr 0x000A "qxJuTqYz*D"
    Add
    St UCase
Line #2:
    LitStr 0x000B "FMfGdWR)CTq"
    LitDI2 0x0005
    ArgsLd larpkGY 0x0002
    St xJmTr
```

FIGURE 2. Exemplar P-code dump originating from a sample file.

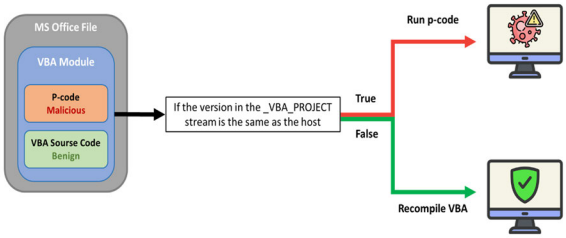

FIGURE 3. Procedural flowchart illustrating a VBA stomping attack.

C. MALICIOUS MACRO THREATS

The advent of VBA, notable for its power and ease of use, has led to a significant increase in incidents involving macro viruses. The first macro virus of note, dubbed Concept, emerged in 1995, targeting Microsoft Word and pioneering a series of increasingly sophisticated macro viruses [16]. Subsequent notable malware threats, including Emotet [17], BlackEnergy [18], and Locky [19], have exploited Microsoft Office documents by embedding malicious VBA scripts to facilitate system breaches. These incidents underscore the dual nature of VBA as both a tool for enhancing productivity and a vector for cybersecurity threats, highlighting the evolving landscape of digital security challenges.

According to the Kaspersky Security Bulletin 2021, the exploitation of Microsoft Office vulnerabilities has seen a slight decrease; however, a significant 49.75% of exploits are still linked to Microsoft Office. This indicates a continuing trend where vulnerabilities, including those involving malicious macros, remain a primary target for cyber adversaries [20]. Gutfleisch et al.'s 2021 research conducted a series of experiments to examine the likelihood of conventional users activating malicious macros received through external communications [21]. The study found that an alarming 63.9% of participants inadvertently activated at least one macro, highlighting the considerable risk posed by the distribution of malicious macros via email. Furthermore, this phenomenon emphasizes the widespread deficiency in user awareness concerning the dangers of enabling macros.

D. INVESTIGATING MALEVOLENT VBA MACROS

The growing threats posed by malicious VBA macros have attracted significant attention, leading to various research efforts and the development of tools dedicated to their detection. These methods are generally divided into three main categories: static analysis, dynamic analysis, and machine learning-based approaches.

1) STATIC ANALYSIS

Static analysis is a critical method in the cybersecurity domain, involving the examination of malicious software without executing it. This approach focuses on analyzing static attributes such as hash values, strings, and source codes to identify malware and understand its mechanisms. In response to the growing threat posed by VBA macro malware, several static analysis tools have been developed. A prominent example is Oletools [22], which offers comprehensive capabilities for the analysis, forensics, and debugging of Office-based malware. Its submodule, Olevba, specifically targets VBA macro analysis, evaluating potential maliciousness through heuristic methods and signatures, and facilitating the extraction of source code. Due to its modular design, Oletools can be extended for use with other applications. Additional noteworthy tools that rely on Oletools include ViperMonkey [23] and Vba2Graph [24]. Despite their capabilities, these tools face limitations against tactics such as VBA stomping, underscoring the need for innovative approaches beyond conventional static analysis methods.

2) DYNAMIC ANALYSIS

In contrast to static analysis, dynamic analysis involves executing VBA macro malware to monitor its behavior, including file changes, process initiation, registry alterations, and API calls. This approach requires a secure, isolated environment to prevent damage from malicious VBA macro execution, typically utilizing sandboxes such as Cuckoo [25] for analysis. These settings integrate advanced dynamic analysis tools within virtual environments, facilitating detailed malware inspection. However, dynamic analysis faces significant challenges, such as handling large volumes of data, navigating complex analysis procedures, and overcoming malware's anti-analysis techniques. These techniques, which may include detecting the analysis environment or executing file self-deletion, can significantly impede analysis efforts.

3) MACHINE LEARNING BASED APPROACHES

In their 2021 study, Mohandas et al. introduced a novel malware detection technique based on machine learning, utilizing opcode frequency to distinguish between benign and malicious executable files, demonstrating the Random Forest Classifier's superior accuracy in identifying potential malware without execution, thus enhancing malware detection capabilities without risking system integrity [26]. Upon evaluating three classification models, the Random Forest algorithm emerged as the most accurate, underscoring the effectiveness of hybrid analysis in detecting malicious documents. Similarly, M. Mimura et al. introduced a novel technique tailored for the detection of macro malware within imbalanced datasets [27]. This method utilizes two advanced language models, Doc2vec and Latent Semantic Indexing (LSI), to extract significant features from the VBA source code. When these features were analyzed using four different classifiers, the combination of Doc2vec and Support Vector Machine (SVM) was found to offer superior detection capabilities, further evidencing the potential of innovative approaches in combating macro malware threats.

In their groundbreaking research, Rana et al. developed a novel technique for detecting malicious macros utilizing P-code analysis, leveraging Term Frequency-Inverse Document Frequency (TF-IDF) to construct feature vectors from 2-gram P-code instruction sequences characteristic of individual macros. By employing machine learning models such as Random Forests and Decision Trees, this methodology demonstrated a notable increase in accuracy for identifying malicious macros based on P-code characteristics. This approach not only provides a refined method for the early detection of macro-based malware but also enhances the existing arsenal of tools for cyber defense by incorporating P-code analytical techniques [28].

In 2022, Fran Casino et al. embarked on pioneering research that leveraged the visual aspects of cyber attacks for the identification of malicious Microsoft Office documents [29]. Their approach harnessed the power of perceptual hashing and Optical Character Recognition (OCR) to detect and analyze embedded images designed to coerce victims into enabling macros or taking other malicious actions. By constructing a comprehensive database of malicious image datasets, their technique not only enhances the accuracy of malicious document detection but also illuminates the intricate web of similarities across various malware campaigns. This methodology, utilizing both static and dynamic analysis tools, underscores the evolving landscape of cyber threat intelligence and the critical role of visual cues in bolstering cybersecurity defenses.

In 2022, Jia Yan et al. introduced DitDetector, an ML-based dual-learning model designed to identify malicious macros through innovative analysis of P-codes. This model ingeniously combines TF-IDF for feature vector generation from 2-gram P-code instruction sequences and employs Random Forests and Decision Trees as its classification models. The integration of perceptual hashing and OCR for the detection of deceptive visual elements and textual content, respectively, alongside a novel application of the visual encoder MobileNetV3 and text encoder TextCNN, culminates in a binary classifier. DitDetector excels in mapping images and deceptive textual captures into a shared space, thereby significantly enhancing the accuracy of malicious macro detection through the adept utilization of P-code analysis [30].

In 2022, V. Ravi et al. introduced a groundbreaking VBA analysis detection system that leverages word embedding techniques, specifically employing Word2Vec embeddings to analyze the structure and content of macro codes within Microsoft Office documents [31]. This system generates embeddings for various elements such as keywords, function names, and strings found within VBA code, categorizing them as either benign or obfuscated. Subsequently, a random forest classifier is utilized to determine the malware family of the analyzed file. This novel approach underscores the effectiveness of Natural Language Processing (NLP) techniques in the analysis of VBA source code, demonstrating significant potential for enhancing malware detection capabilities.

III. METHODS

Figure 4 depicts the comprehensive system architecture, which is systematically divided into five distinct segments: Data Collection, where relevant data is gathered; Data Processing, involving the refinement and preparation of collected data; Feature Definition, where key characteristics are identified and defined; Detection Model, the stage at which the detection algorithms are applied; and Malicious Prediction, where potential threats are predicted based on the analysis. This structured approach facilitates an organized progression from data acquisition to threat prediction. The algorithms used in this research are a combination of existing techniques and modifications to optimize the detection of VBA macro malware.

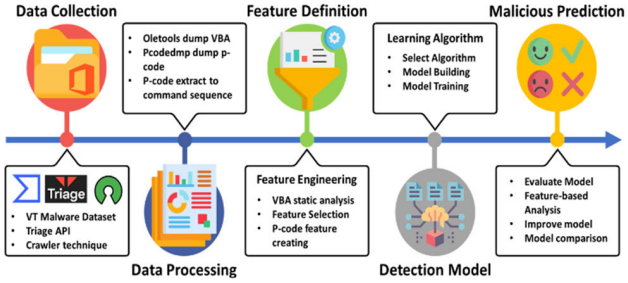

FIGURE 4. Overview of the system.

In the Data Collection phase, our methodology strategically sourced malicious samples from the comprehensive malware databases provided by VirusTotal and Triage. In parallel, benign samples were carefully collected from a wide range of Open-Source platforms, the specifics of which will be elaborated on in subsequent discussions. This dual-source approach ensures a robust dataset, critical for the nuanced differentiation between malicious and benign software behaviors, thereby enhancing the accuracy and reliability of our cybersecurity analysis.

During the Data Processing phase, our procedure involves extracting VBA or P-code from both benign and malicious documents. For VBA source code, we utilize the tool Olevba to dissect the code and construct features based on VBA behaviors. In the case of P-code, we extract instructions using the pcodedmp tool, organizing them into coherent instruction sequences. These sequences are then transformed through TF-IDF (Term Frequency-Inverse Document Frequency) analysis, employing an n-gram representation to enhance the textual analysis. Subsequently, we apply the UMAP (Uniform Manifold Approximation and Projection) algorithm for efficient dimensionality reduction. This meticulous process enables us to formulate distinct, structure-based features for P-code, significantly enriching our malware detection capabilities.

In the Feature Definition stage, we meticulously extract 30 critical features from the VBA-behavior-based dataset using advanced feature selection techniques, ensuring a focused approach to malware detection. For P-code-structure-based features, our exploration involved various combinations, ultimately identifying the integration of TF-IDF with a 3-gram representation of the P-code instruction sequence, coupled with the application of UMAP for dimensionality reduction to 50 dimensions, as the most effective strategy. Additionally, we introduced self-organized features to this set, enriching our dataset with a total of 55 P-code-structure-based features. This comprehensive approach culminated in the assimilation of 85 distinct features into our training model, setting a solid foundation for sophisticated malware analysis and detection.

In the context of enhancing model training methodologies, Chen and Guestrin [32] deployed the XGBoost algorithm, renowned for its efficacy in handling diverse feature combinations. The training phase meticulously employs a validation dataset, serving as a crucial mechanism to prevent model overfitting and to evaluate the model's predictive performance rigorously. This refined model training approach, leveraging the XGBoost algorithm, empowers the detection system to accurately discern the potential threats posed by Office documents embedded with VBA macros, showcasing a significant advancement in macro malware detection techniques [32].

A. DATA COLLECTION

VirusTotal is distinguished by its comprehensive malware analysis capabilities, aggregating malware samples from worldwide incidents. It achieves a broad detection spectrum

by integrating a variety of antivirus software vendors, alongside offering a robust API that significantly aids in the identification of malicious files, thus cementing its role as an indispensable resource in cybersecurity research. Concurrently, HatchingTriage, launched by Hatching International B.V. in the Netherlands in 2018, emerges as a pivotal online sandbox service. It grants researchers access to an extensive repository of malicious samples, with features allowing for the tailored searching of samples based on specific criteria through its search operator. This functionality enhances the efficiency of retrieving targeted samples, streamlining research efforts in the field.

The dataset for this study is primarily sourced from the extensive malware database of VirusTotal, incorporating office file samples with malicious VBA macros collected between 2017 and 2022, totaling 1844 samples. Additionally, 170 malicious samples were obtained from the HatchingTriage database, resulting in a comprehensive dataset of 2015 malicious samples. In contrast, benign samples were meticulously curated from a variety of open-source platforms, including GitHub and GitLab, as well as specialized platforms such as Contexture [33] and Sitestory [34]. The collection process involved the deployment of crawler technology to identify and gather open-source office documents featuring VBA macros. To ensure their non-malicious nature, these documents underwent a verification process using VirusTotal, ultimately compiling 2014 benign samples for analysis. This dual-faceted approach, combining both malicious and benign samples, establishes a robust foundation for examining the intricacies of macro-based malware in office files. Malicious samples were obtained from malware databases such as VirusTotal and Triage, while benign samples were collected from open-source platforms like GitHub [23], [24] and GitLab [33], [34], and verified using VirusTotal.

B. DATA PROCESSING

Data processing plays a crucial role in the experimental frameworks of cybersecurity, directly influencing the quality and relevance of the outcomes. In our approach, we meticulously extract VBA and P-code from both benign and malicious samples to lay the groundwork for thorough analysis and feature generation. This extraction allows us to differentiate between VBA-behavior-based features, which are derived from the behavioral patterns of the VBA code, and P-code-structure-based features, which originate from the structural composition of the P-code. An illustrative pseudo-code, presented in Figure 5, details this process, highlighting our systematic approach to dissecting and analyzing the code for the generation of insightful features critical to identifying cybersecurity threats effectively [35].

Figure 5 presents a pseudo-code outlining the data processing methodology applied to VBA source code within Microsoft Office files. The algorithm initiates by creating a dictionary to store features extracted from both malicious and benign VBA macro samples. The VBA code is then analyzed using the tool 'olevba', and keywords are categorized by type—with a particular focus on those deemed suspicious. These suspicious keywords are tallied and added to the feature dictionary, while other keyword types are enumerated separately. Finally, the algorithm consolidates these features into the feature dictionary, which is returned as the output, laying the groundwork for subsequent malware detection and analysis. The determination of whether kw_type is suspicious is based on heuristics and predefined signatures within analysis tools like olevba. The basic classifications used include categories such as AutoExec, Suspicious, and Indicators of Compromise (IOC).

```
Algorithm 1: Data Processing of VBA Source Code
Input: Malicious and benign samples of Office files with VBA macros
Output: Generate static analysis features from VBA source code
1: initialization;
2: feature_dict = A dictionary to store extracted features for All Malicious samples and
Benign samples do
3:    result = VBA code analysis by olevba;
4:    other_type = A dictionary to store other kw_type number cnt;
5:    for kw_type, keyword, description in result do
6:       if kw_type == Suspicious then
7:          append 1 to feature_dict[keyword] list;
8:       else
9:          other_type[kw_type] += 1;
10:      end if
11:   end for
12:   for key, value in other_type do
13:      append value to feature_dict[key] list;
14:   end for
15: end for
16: return feature_dict;
```

FIGURE 5. Pseudo-code illustrating data processing within VBA source code.

1) VBA SOURCE CODE EXTRACTION AND ANALYSIS

The initial phase of our research involves extracting VBA source code from Office documents to identify features that reveal behavioral patterns intrinsic to the code. This process utilizes the Oletools suite, particularly the Olevba submodule, to conduct a thorough analysis, resulting in a comprehensive feature set. Within this analysis, Olevba classifies keywords into distinct categories, including AutoExec, Suspicious, and Indicators of Compromise (IOC), with precision in assigning each keyword based on established criteria. This categorization is pivotal in differentiating between benign and potentially harmful VBA code, enhancing the effectiveness of cybersecurity detection mechanisms.

In the preliminary phase of data processing, the VBA source code is extracted from Office document samples to identify behavior-based features that are crucial for discerning potential threats. The analysis, facilitated by the Olevba submodule of Oletools, categorizes keywords into three

distinct types—AutoExec, Suspicious, and IOC—assigning each keyword to the appropriate category based on predefined criteria. This detailed categorization is essential for constructing a feature set that accurately reflects the behavioral patterns inherent to the VBA code, which is vital for the subsequent phases of cybersecurity threat analysis and detection.

2) P-CODE STRUCTURE AND FEATURE EXTRACTION

Figure 6, presumably detailed in your document, is likely to illustrate a pseudo-code algorithm that details the extraction and processing of P-code from Office document samples. The algorithm utilizes the tool pcodedmp to extract P-code, which represents the instruction sequences vital for identifying P-code-structure-based features. A filtering mechanism is then employed to eliminate any duplicate instruction sequences, enhancing the precision and efficiency of the resultant feature set. By doing so, the algorithm ensures the subsequent model training is accurate and not confounded by redundancy in instruction sequences, which is essential for developing a robust machine learning model for malware detection. This systematic approach underscores the importance of data quality in model training within the domain of cybersecurity.

Figure 6 depicts an algorithm for processing P-code to extract instruction sequence features from Office files containing VBA macros. The algorithm initiates by creating dictionaries for instruction types and resultant features. It then iterates over both malicious and benign samples, using pcodedmp to perform a source dump of the P-code. For each instruction in the P-code source dump, the instruction is appended to a list, compiling a sequence of P-code instructions. Additionally, the algorithm calculates other self-defined features, which are then combined with the P-code sequence in the resultant features dictionary, indexed by a sample hash. This systematic approach ensures that each sample is represented by a comprehensive feature set, merging P-code instruction sequences with other relevant features, thus enriching the dataset for subsequent machine learning model training.

```
Algorithm 2: Data Processing of P-code
Input: Malicious and benign samples of Office files with VBA macros
Output: Generate p-code instruction sequence features from P-code source dump
1: initialization;
2: instruction_type = A dictionary storing the types of P-code instructions;
3: result_features = A dictionary to store all generated features;
4: for all Malicious samples and Benign samples do
5:     pcode_dump = P-code source dump by pcodedmp;
6:     pcode_seq = A list to store P-code instruction sequence;
7:     for all instructions in P-code source dump do
8:         pcode_seq.append(instruction);
9:     end for
10:    other_features = Calculate result of other self-defined features;
11:    result_features[sample_hash] = pcode_seq + other_features;
12: end for
13: return result_features;
```

FIGURE 6. Pseudo-code detailing the data processing of P-code.

C. FEATURE DEFINITION

1) VBA-BEHAVIOR-BASED FEATURES

Feature selection is a critical process that significantly enhances the operational efficiency of machine learning models by reducing the complexity of VBA-behavior-based features. In this research, embedded feature selection techniques were employed within the XGBoost framework to identify the most informative features, with the learning curve method assisting in ascertaining the ideal feature count [36]. As a result, a set of 30 key features emerged as the most representative for VBA-behavior-based detection, as detailed in Table 1. This table provides an essential enumeration of these salient features, with their definitions, reflecting the robustness of the feature extraction method. Particularly, it includes four features that quantify category frequencies based on keyword types, emphasizing the precision of our feature engineering approach [37].

2) P-CODE-STRUCTURE-BASED FEATURES

During the data processing phase, the efficient extraction and transformation of p-code into a sequence of instructions leverage the Term Frequency-Inverse Document Frequency (TF-IDF) technique. This technique is employed to modify the n-gram representation of the p-code instruction sequence, thus generating an embedding vector [38], [39]. TF-IDF stands as a significant method for representing text features, emphasizing the importance of terms through the calculation of the term frequency (TF) and inverse document frequency (IDF) for each term. Applying n-grams to p-code instruction sequences produces a variety of instruction combinations. Each is considered a distinct term for the TF-IDF score computation, enabling a thorough encapsulation of crucial features within the p-code instruction sequences. This facilitates detailed analyses of the relationships between instructions and their contexts, enhancing our understanding of the data's intrinsic structure

TABLE 1. VBA-behavior-based features.

Feature	Name	Feature	Name
F1	Hex Strings	F16	Binary
F2	ShowWindow	F17	Run
F3	CreateObject	F18	open
F4	CreateTextFile	F19	ShellExecute
F5	StrReverse	F20	Lib
F6	IOC	F21	CodeModule
F7	ActiveWorkbook.SaveAs	F22	WScript.Shell
F8	URLDownloadToFileA	F23	SavetoFile
F9	CallByName	F24	write
F10	RegRead	F25	vbNormalFocus
F11	powershell	F26	Kill
F12	AutoExec	F27	put
F13	Shell	F28	User-Agent
F14	VBProject	F29	Chr
F15	Environ	F30	Base64 Strings

The expansion of 'n' in n-gram models is directly linked to an increase in the dimensionality of the resulting TF-IDF embeddings, which highlights the complexity and richness of the data representation. For instance, the representation of p-code instruction sequences through a 3-gram model can lead to embeddings with thousands of dimensions [39]. Such increased dimensionality, while rich in information, introduces several challenges in model learning, including longer training times, higher computational demands, and an elevated risk of overfitting. To address these issues, dimensionality reduction techniques become crucial. Our study leverages Uniform Manifold Approximation and Projection (UMAP) to effectively manage this complexity, demonstrating the balance between preserving informative features and maintaining model efficiency.

Uniform Manifold Approximation and Projection (UMAP) excels at retaining both local and global data structures, significantly outperforming traditional techniques like PCA and t-SNE in capturing complex nonlinear relationships within datasets [40]. Unlike PCA, which only conserves linear correlations, and t-SNE, primarily suited for data visualization without efficient dimensionality reduction, UMAP leverages stochastic gradient descent to minimize cross-entropy. This approach enables UMAP to adeptly handle large datasets by efficiently reducing dimensions, making it particularly pertinent to our study.

• F31∼80 (P-CODE INSTRUCTION EMBEDDING)

The intricacies of P-code instruction embedding are enhanced through the application of n-grams, meticulously transforming the instruction sequence to refine cybersecurity analysis [41]. This process amalgamates multiple instructions into a singular conceptual unit, calculates their TF-IDF scores, and generates distinct embeddings for each sequence. However, the inherent augmentation of embedding dimensionality through n-grams poses challenges for model training efficiency. To address this, we leverage Uniform Manifold Approximation and Projection (UMAP), a robust non-linear dimensionality reduction technique, to streamline the data [42]. Specifically, a 3-gram representation of the instruction sequence is crafted into embeddings, subsequently reduced to a manageable dimensionality of 50 via UMAP, illustrating its effectiveness in processing and analyzing cybersecurity data efficiently.

TABLE 2. Encompassing P-code-structure-based features.

Feature	Name
F31~F80	P-code#1 ~ P-code#50
F81	Information Entropy
F82	Comment Count
F83	Comment Ratio
F84	Function Count
F85	Function Ratio

Table 2 presents a detailed enumeration of P-code-structure-based features, central to the exploration of program characteristics within a cybersecurity context. Features F31 through F80 represent a comprehensive range of P-code instructions, labeled P-code#1 to P-code#50, enabling a granular analysis of code structure and behavior. Additional metrics, including Information Entropy (F81), offer insights into the randomness and complexity of the code, while Comment Count (F82) and Comment Ratio (F83) provide quantitative measures of code documentation. Furthermore, Function Count (F84) and Function Ratio (F85) assess the functional composition of the code, offering a nuanced view of its structural complexity. These features collectively facilitate a multifaceted analysis of programmatic elements, contributing significantly to advancements in cybersecurity research and applications.

• F81 (INFORMATION ENTROPY)

Information entropy is pivotal in assessing the randomness or uncertainty within a dataset, offering valuable insights across diverse analytical contexts. This analysis predominantly utilizes Shannon entropy, as described by Hsu et al. [43], where entropy is computed by aggregating the probabilities of discrete events with their natural logarithms, reflecting the 'information' contained within potential outcomes of a variable. Furthermore, Chao and Li [44] extend this concept through K-nearest neighbor (KNN) distance entropy, enhancing data quality assessment in remote sensing image classification. Such methodologies not only quantify information uncertainty but also enable the discrimination of data based on its inherent informational value, making them indispensable in cybersecurity and remote sensing applications.

• F83 (COMMENT RATIO)

The Comment Ratio, herein defined as the proportion of comment instructions to the total instructions within a P-code source dump, serves a critical function by enabling the scrutiny of the true proportion of comments from a P-code perspective [4]. Notably, code comments are instrumental in bolstering the readability, understandability, and, implicitly, maintainability of the code [45]. In the context of benign samples, the prevalence of explanatory comments, typically inserted by developers to elucidate code functionality and logic, is anticipated. In stark contrast, malicious samples frequently exhibit a conspicuous paucity of comments. The rationale underpinning this lies in attackers intentionally eschewing explanations, a tactical maneuver designed to obfuscate analysis and augment the complexity for researchers endeavoring to discern their objectives [46].

• F85 (FUNCTION RATIO)

The Function Ratio, delineated as the ratio of function instructions to the collective quantity of instructions within the P-code, affords a mechanism for scrutinizing the veritable scope of function declarations from a P-code viewpoint [31]. As system development advances and vulnerability patches are judiciously applied, the intricacy of malicious software correspondingly amplifies to adeptly achieve aggressive objectives [47]. Developers might necessitate the incorporation of auxiliary functions to amplify malware ferocity or to dissever program flow to circumvent antivirus software [48]. Hence, a rigorous analysis of the proportion of function

instructions can fortify the precision of malware detection mechanisms.

D. DETECTION MODEL

Ensemble Learning emerges as a fundamental machine learning paradigm, accentuating the enhancement of prediction accuracy and stability by coalescing the predictive insights of multiple learners [49]. This strategy predominantly bifurcates into two distinct methodologies, namely Bagging and Boosting, each encapsulating its unique operational mechanics [50]. While Bagging amalgamates multiple weak learners and consolidates results through averaging or voting to fortify the model against the variance, Boosting navigates through an iterative framework, consecutively harnessing the predictive inaccuracies of antecedent weak learners to steer and refine successive learning processes [49]. The XGBoost algorithm, herein utilized, stands as an exemplar of a boosting algorithm, iteratively maneuvering through a succession of weak decision trees aimed at minimizing the objective function and thereby, engendering optimal predictive outcomes [51]. The intrinsic objective function of the XGBoost model bears regularization, furnishing a formidable shield against overfitting while ensuring model generalization across varied datasets [37]. Simultaneously, the infusion of decision trees into the model not only augments its interpretability but also confers a clear understanding of the predictive process, thereby facilitating feature selection and nurturing transparent model interpretation. The clarity in decision-making, bestowed by decision trees, makes the XGBoost algorithm especially beneficial in contexts demanding a transparent elucidation of model decision pathways and feature significance, a condition frequently stipulated in cybersecurity realms [36]

IV. PERFORMANCE ANALYSIS AND DISCUSSION

A. ENVIRONMENT AND PARAMETER SETTINGS

The experiment was conducted using a system configured with an Intel Core i7-8700 CPU at 3.20GHz, 32GB DDR4 RAM, and a 2TB HDD. The system operated on Windows with Python 3.10 as the programming language. The machine learning training process was developed using Scikit-learn, employing the XGBoost algorithm for detecting samples. For handling dimensionality in embeddings composed of TF-IDF values from sequences of p-code instructions, the UMAP technique was applied in the feature definition phase.

The dataset was compiled from various sources: malicious samples from the VirusTotal and Triage malware repositories, benign samples from GitHub, and GitLab, as well as self-compiled open-source files. Following a filtering procedure, the total dataset comprised 4429 samples. Of these, 4030 samples were subjected to an 80-20 split for training and validation purposes, respectively. To assess the trained model's performance on unseen data, an independent test set of 399 samples, including both malicious and benign files, was utilized. Details regarding the distribution of the dataset are provided in Table 3.

TABLE 3. Allocation of datasets.

	Quantity	Data Sources	Proportion
Training Dataset	3224	VirusTotal Triage GitHub, GitLab Other Open Sources	80%
Validation Dataset	806		20%
Test Dataset (Unseen Dataset)	399	Triage GitHub	100%

For this investigation, the XGBoost algorithm was selected, leveraging a gbtree booster to construct its non-linear model architecture. The algorithm was initially configured with a maximum tree depth of 6 and a learning rate of 0.5, with the number of iterations set at 100. Parameter optimization was achieved through a grid search strategy, utilizing Logloss as the evaluation criterion to assess XGBoost's performance variation across iterations.

B. VBA-BEHAVIOR-BASED MODEL

The experiment prioritized a VBA-behavior-based model, as evidenced by the performance and confusion metrics detailed in Table 4 and Fig. 7, respectively. This approach marks a significant investigation into malware detection via VBA-behavior characteristics, affirming the model's efficacy with a notable accuracy rate of 0.946. Although this outcome suggests the model's robustness, it also highlights the necessity of further scrutinizing the recall metric, recorded at 0.92. Such an analysis is imperative for enhancing the precision of detecting genuinely malicious files, and identifying a critical path for model improvement.

To ensure the robustness of our methodological framework, the model's performance was rigorously evaluated against key metrics including accuracy, F1-Score, Log Loss, and Area Under the Curve (AUC), leveraging a statistically significant sample size. The model, powered by the XGBoost algorithm, was chosen for its proven efficacy in managing imbalanced and complex datasets. Crucially, an inductive-statistical analysis establishes a solid link between VBA-behavior-based features, extracted using Olevba, and the model's high-performance metrics. This connection supports the observed high accuracy and an AUC of 0.950, highlighting the model's effectiveness and potential areas for further optimization.

The model's high accuracy not only meets the predefined performance criteria but also underscores a critical link between the method of VBA-behavior-based feature extraction and its predictive accuracy. Furthermore, the implementation of the XGBoost model, renowned for its adept management of classification challenges and for enhancing the transparency of decision-making processes, likely plays a

key role in surpassing existing models in malware detection efficiency.

However, a recall metric of 0.92 subtly raises concerns; despite the model's commendable accuracy, the presence of false negatives warrants further scrutiny. This observation beckons an inquiry into the effectiveness of the proposed solution and whether additional refinements could improve the recall. Moreover, while the solution exhibits notable proficiency in the current context, assessing its applicability across diverse and more intricate scenarios is essential. This introduces a nuanced caution against overgeneralizing the results to various contexts without additional validation.

In summary, the VBA-behavior-based model demonstrates a notable enhancement in malware detection capabilities, as corroborated by experimental outcomes, achieving an accuracy of 0.946 (refer to Table 4) and further detailed through the confusion metrics (see Fig. 7). Despite these accomplishments, a focused examination of the recall metric reveals an area ripe for development. The model, although significantly precise, reveals an imperative for ongoing enhancement—especially in reducing false negatives—to forge a detection approach that is not only more accurate but also universally adaptable across varying scenarios. This insight underscores the necessity for the model to undergo continuous refinement, aiming to achieve an optimal balance of accuracy and generalizability in the dynamic landscape of malware detection.

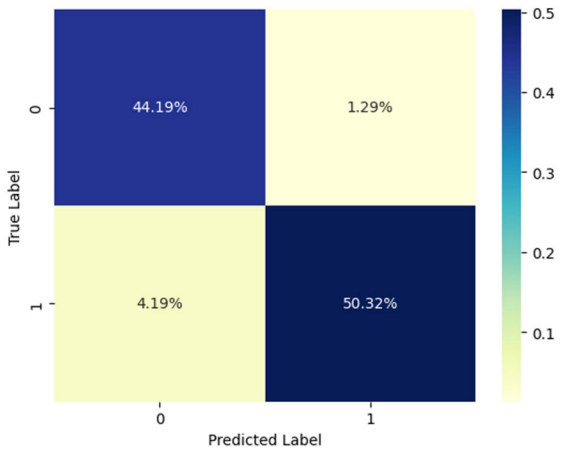

FIGURE 7. Confusion metrics of the VBA-behavior-based model.

TABLE 4. VBA-behavior-based model performance metrics.

Accuracy	F1-Score	Log Loss	AUC
0.946	0.949	0.159	0.950

Consequently, the core assertion that VBA-behavior-based features, precisely extracted via Olevba, foster a malware detection model of high accuracy is affirmed. However, this finding also accentuates the continuous requirement for iterative enhancements and the expansion of experimental conditions. Such steps are crucial to not only reaffirm the model's effectiveness but also to augment its overall performance and applicability in diverse scenarios, ensuring its sustained relevance and potency in the field of malware detection.

C. P-CODE-STRUCTURE-BASED MODEL

The empirical venture into the evaluation of the P-code-structure-based model, furnished by features F31 to F85 and elucidated through differing n-gram representations, stakes a compelling claim towards effective malware detection, particularly through adept utilization of the TF-IDF and UMAP algorithms (Table 5). Significantly, the optimal harmonic between a 3-gram representation and a dimensional reduction to 50 via UMAP not only aligns with the purported efficacy in the experimental model but also concretizes the claimed performance criteria through exhibited maximization in both accuracy (0.983) and F1-score (0.983).

TABLE 5. Evaluation of P-code embedding with different parameter combinations.

N-grams	Target Dimension	Accuracy	F1-Score
1-gram	30	0.976	0.977
	50	0.973	0.974
	100	0.971	0.972
2-gram	30	0.980	0.981
	50	0.980	0.981
	100	0.973	0.975
3-gram	30	0.980	0.981
	50	0.983	0.983
	100	0.972	0.973

Anchoring the experimental framework, a meticulous variation in n-gram representations and target dimensions was deployed, ensuring a focused and exhaustive exploration of the performance criteria within the tested parameters, safeguarding the methodological rigor and instilling a robust statistical basis for inference. The resultant outcomes, decisively indicative of a non-linear relationship between P-code instruction sequences, are potently underpinned by the experimental results and ensuing statistical analyses, thereby substantiating the initial claims with quantitative evidential support.

Moreover, the comparative analysis among various dimensionality reduction methods, encompassing PCA, KPCA, and UMAP (Table 6), articulates a clear preeminence of UMAP in enhancing model performance, thereby aligning precisely with the conceptual anticipation of its ability to discern and leverage the non-linear relationship between P-code instruction sequences. The divergences in performance metrics across varied algorithms substantiate that the enriched performance is inherently driven by the proposed solution, underscoring a definitive correlation between the strategic utilization of UMAP and optimized model efficacy.

A pivotal elucidation within the context of the proposed solution is the resounding affirmation of the conceptual hypothesis that the judicious selection and engineering of

features, particularly through strategic n-gram representation and dimensionality reduction, substantively influences model performance. The discernable efficacy of a 3-gram representation, conjoined with UMAP-reduced 50 dimensions, in not only optimizing accuracy but also concurrently minimizing log loss (Table 7), illustrates a tangible confirmation that the refined feature extraction and management technique propels the efficacy of the P-code-structure-based model.

In conclusion, the P-code-structure-based model, scrutinized through stringent experimental parameters and substantiated through the provided tables (Tables 5, 6 and 7), affirmatively validates the seminal claim that judicious feature selection, extraction, and management, particularly exploiting the synergetic relationship between n-gram representations and UMAP dimensionality reduction, significantly enhance model performance in detecting malicious VBA macros within Office documents.

TABLE 6. Evaluation of using different dimensionality reduction methods.

Dimensionality Reduction Method	Accuracy	F1-Score
PCA	0.975	0.977
KPCA	0.975	0.977
UMAP	0.982	0.983

TABLE 7. P-code-structure-based model performance metrics.

Accuracy	F1-Score	Log Loss	AUC
0.982	0.983	0.053	0.980

As the evidence posits, the clear conjunction between the meticulous management of P-code-structure-based features and the model's elevated accuracy and minimized log loss is not merely correlative but causative, driven by the thoughtful application of theoretically grounded feature management techniques. Nonetheless, a prudent perspective mandates an ongoing exploration into further optimizing these features, ensuring that as malware evolutionarily adapts, the model perennially refines to counteract emergent threats with sustained and enhanced efficacy.

D. ALL FEATURES

The synergistic amalgamation of the 85 meticulously curated features, encompassing VBA-behavior-based and P-code-structure-based types, fabricates an exemplary instance of optimized cybersecurity model performance, exhibited by an impressive accuracy rate of 0.988 and a commendably minimized log loss of 0.049 (Table 8). The quintessential correlation between this combined feature approach and the accrued enhancement in model proficiency, visibly surpassing the performances of the models utilizing isolated feature types, substantiates the claim that a holistic feature incorporation strategy perpetuates nuanced, multifaceted threat detection capabilities.

TABLE 8. All feature model performance metrics.

Accuracy	F1-Score	Log Loss	AUC
0.988	0.988	0.049	0.990

Rigorously ensured through a statistically sound and methodologically stringent experimental framework, the resulting performance metrics not only validate the model's operational methodology but also affirmatively advocate for the efficacious integration of diverse feature types. The corresponding confusion metrics (Fig. 8) fortify this assertion, illustrating the model's deft proficiency in concurrently maximizing true positive rates while diligently mitigating false negatives.

Intriguingly, the model's feature importance analysis, articulated through the XGBoost algorithm (Fig. 9), unveils Hex Strings (F1) as the pinnacle feature, inherently driving model decisions with remarkable potency. Simultaneously, the presence of P-code instruction embedding, VBA-behavior-based features (such as AutoExec (F12) and IOC (F6)), and P-code-structure-based features (e.g., information entropy, function_cnt (F84), and ratio_function (F85)) within the upper echelons of feature importance underscores a balanced influence of both feature types in propelling model accuracy and precision.

While the results manifestly exhibit the efficacy of the combined features, it is essential to affirm that the detected potency of features like Hex Strings (F1) and AutoExec (F12) substantively emanates from the integrative model approach, illustrating that an encompassing feature strategy not only augments the model's diagnostic capabilities but also broadens its operational scope and applicability across varied cybersecurity contexts.

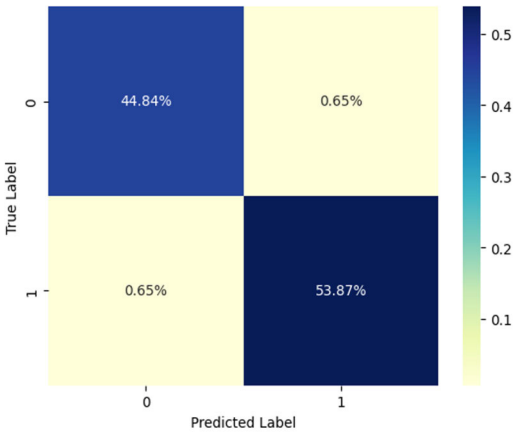

FIGURE 8. Confusion metrics of all feature model.

Conclusively, the results derived from the integrative model, employing all selected features, substantively validate the overarching claim of enhanced performance, optimizing accuracy and minimizing log loss through a diverse and comprehensive feature inclusion strategy. In light of these

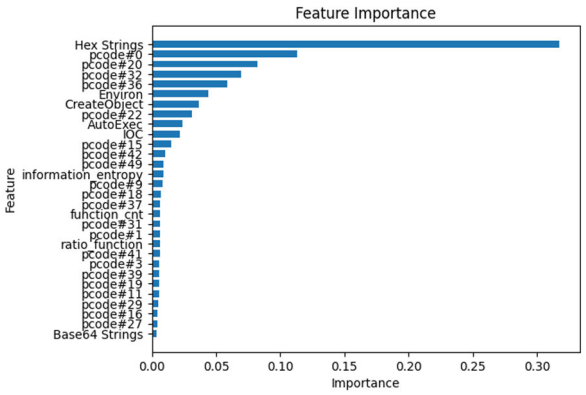

FIGURE 9. Confusion metrics of VBA-behavior-based mode.

findings, the implications reverberate beyond mere model performance and delve into the translational applicability of this approach within real-world cybersecurity contexts, where threat vectors are invariably complex and multivariate.

While this model exhibits exemplary performance, a foresighted perspective necessitates the continuous evolution and iterative refinement of this model, perpetually aligning with the dynamically evolving landscape of cyber threats and vulnerabilities. Moreover, a meticulous examination of the potential constraints and limitations of the employed algorithms, coupled with an exploratory foray into alternate or emergent algorithms and methodologies, will perpetuate the sustained relevance and efficacy of the model in confronting future cyber threats.

Upon meticulous evaluation and comparison with existing methodologies, particularly the Structural Feature Extraction Methodology (SFEM) as introduced by Cohen et al [52], our research delineates a significant advancement in the detection of malicious VBA macros within Office documents. By integrating P-code analysis with XGBoost, an innovative machine learning algorithm, our approach not only achieves an unparalleled accuracy rate of 98.8% but also markedly reduces the log loss to 0.049. This exemplifies a substantial improvement over the SFEM approach, which, while pioneering in employing machine learning for malicious document detection, does not leverage the nuanced insights provided by P-code analysis. Our method's robust performance underscores the efficacy of combining VBA-behavior-based and P-code-structure-based features, presenting a comprehensive and dynamic solution to cybersecurity threats. This integrative feature strategy not only enhances the model's diagnostic capabilities but also broadens its applicability across various cybersecurity contexts, making a pivotal contribution to safeguarding digital infrastructures against increasingly sophisticated cyber threats.

E. SUMMARY

The research meticulously implemented a cohesive experimental framework to unravel the nuanced capabilities of VBA-behavior-based and P-code-structure-based features in distinguishing benign from malicious file samples. Table 9 enumerates a decisive summarization of the experimental outcomes, unveiling a nuanced hierarchy of model performances under diverse feature-inclusion scenarios. Specifically, a discernible disparity emerges when comparing models trained exclusively on VBA-behavior and P-code-structure features, delineating the latter's superior potency in discerning malicious code structures.

TABLE 9. Performance summary for feature combination.

Testing Features	Accuracy	F1-Score	Precision	Recall
VBA-behavior	94.51%	94.83%	97.50%	92.30%
P-code-structure	98.17%	98.31%	98.61%	98.02%
All Features	98.70%	98.81%	98.81%	98.81%

Delving into the individualized performances, models solely trained on VBA-behavior-based features manifested a commendable, albeit suboptimal, capability in deciphering malicious entities, with an accuracy of 94.51% and recall of 92.30%. Conversely, the P-code-structure-based model substantively augmented both accuracy and recall to 98.17% and 98.02%, respectively. The intrinsic variances in these performances are emblematic of the respective feature types' capabilities in navigating the complex and obfuscated realms of malicious coding structures and behaviors.

The salient elevation of model performance to an accuracy of 98.70% and an equivalent precision and recall of 98.81%, upon the integration of both feature categories, stands testament to the theoretical assertion that a multifaceted, integrative feature approach is instrumental in optimizing the model's predictive capacities. The integrated model not only amalgamates the respective strengths and mitigates the inherent limitations of the individual feature categories but also synergistically enhances the model's comprehensive threat detection capabilities.

Moreover, the integrated model proficiently navigates the conundrum of maintaining a harmonized balance between precision and recall, thereby ensuring that its elevated accuracy does not inadvertently sacrifice either parameter. Such a balance is imperative, particularly in cybersecurity contexts, where both false negatives and false positives carry substantive risks and implications.

The achieved results, while exemplary, open avenues for further research and exploration into evolving threat vectors and increasingly sophisticated malicious code structures. The model's verified efficacy in detecting malicious VBA macro within office documents affirms its application within this specified context; however, its translational applicability to emerging and variant malicious entities warrants further exploration and validation.

Moreover, an in-depth exploration into alternative or supplementary feature types, diversified machine-learning algorithms, and varied dimensionality reduction techniques may unveil further potential enhancements or optimizations to the model. Additionally, as cyber threats perennially evolve, ensuring the model's sustained adaptability and relevance will necessitate its continuous alignment with, and validation against, emerging threat vectors and cyber-attack methodologies.

The validated efficacy of the integrated feature model, as substantiated through rigorous experimental validations, exemplifies a promising stride toward enhanced cyber-threat detection and mitigation, providing a robust foundation upon which future research and practical applications can confidently be built.

Upon comparing the performance metrics of our research with those detailed in the work of Huneault-Leblanc and Talhi on P-Code based classification to detect malicious VBA macros, a significant advancement in precision, accuracy, and recall can be observed in our approach. Our methodology, which utilizes an integrated feature set comprising both VBA-behavior-based and P-code-structure-based attributes, has proven to be more efficacious in identifying malicious VBA macros within Office documents, achieving an accuracy of 98.70% and a recall rate of 98.81%. This surpasses the high accuracy mark set by Huneault-Leblanc and Talhi's method [8], which achieved an accuracy of 98.8% but did not specify precision and recall values for direct comparison.

The distinction in performance is primarily attributed to our comprehensive analysis that incorporates a more diversified set of features, including both behavior and structural aspects of the code, thereby enhancing the model's ability to discern between benign and malicious documents with greater precision. Moreover, the integration of P-code-structure features has been instrumental in navigating the complexities of obfuscated malicious code, offering a robust framework that is not solely reliant on VBA behavior analysis but extends to examine the underpinning code structure for a more thorough evaluation.

This advancement is not only a testament to the effectiveness of incorporating a multifaceted feature approach but also underscores the potential for real-world application. The elevated performance metrics of our model affirm its capability to significantly contribute to the cybersecurity domain, offering a more reliable solution for detecting sophisticated malicious threats embedded within Office documents. The implications of this research extend beyond academic inquiry, presenting a viable tool for cybersecurity practitioners to enhance their defense mechanisms against the evolving landscape of digital threats.

In conclusion, the comparative analysis underscores the superiority of our integrated feature model in detecting malicious VBA macros, marking a significant step forward in the realm of cybersecurity threat detection. This progression fosters a deeper understanding of malicious code detection methodologies and paves the way for future research and development in this critical field.

In the realm of real-world applications, the implications of this research are far-reaching. The elevated detection accuracy and balanced precision-recall achieved by the integrated model promise significant advancements in cybersecurity measures for organizations. By embedding this model into existing security frameworks, companies can better protect sensitive information and prevent potentially devastating cyber-attacks. Future integration into endpoint protection platforms could offer real-time detection capabilities, significantly reducing the window of opportunity for attackers. This research lays a solid foundation for the next generation of cybersecurity tools, emphasizing the need for continuous adaptation and enhancement in response to the ever-evolving cyber threat landscape.

V. CONCLUSION

This study has presented an advanced detection system for identifying malicious VBA macros in Office documents. By combining VBA source code and P-code analyses with sophisticated feature extraction and machine learning techniques, we have significantly enhanced detection accuracy and reliability.

Our proposed solution leverages both VBA-behavior-based and P-code-structure-based features, utilizing the XGBoost algorithm. This dual approach enables a thorough examination of code behavior and structure, leading to superior detection capabilities. Additionally, the integration of NLP techniques and UMAP for dimensionality reduction refines the feature set, creating a robust and efficient model.

The results confirm the exceptional performance of our approach, achieving an impressive accuracy of 98.70% and an F1-score of 98.81%. These outcomes underscore the model's effectiveness in accurately distinguishing malicious macros, demonstrating its potential as a powerful cybersecurity tool. This work not only marks a significant advancement in malware detection but also lays the groundwork for future research to further enhance and adapt the model to counter evolving cyber threats.

REFERENCES

[1] X. Zhou, W. Liang, K. Yan, W. Li, K. I. Wang, J. Ma, and Q. Jin, "Edge-enabled two-stage scheduling based on deep reinforcement learning for Internet of Everything," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3295–3304, Feb. 2023, doi: 10.1109/JIOT.2022.3179231.

[2] S. B. Goyal, A. S. Rajawat, R. K. Solanki, M. A. M. Zaaba, and Z. A. Long, "Integrating AI with cyber security for smart industry 4.0 application," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Lalitpur, Nepal, Apr. 2023, pp. 1223–1232, doi: 10.1109/ICICT57646.2023.10134374.

[3] D. Q. Tai, P. Gallus, and P. Frantis, "Macro malware development issuses," in *Proc. Int. Conf. Mil. Technol. (ICMT)*, Brno, Brno, Czech Republic, May 2023, pp. 1–6, doi: 10.1109/ICMT58149.2023.10171257.

[4] V. Koutsokostas, N. Lykousas, T. Apostolopoulos, G. Orazi, A. Ghosal, F. Casino, M. Conti, and C. Patsakis, "Invoice #31415 attached: Automated analysis of malicious Microsoft office documents," *Comput. Secur.*, vol. 114, Mar. 2022, Art. no. 102582, doi: 10.1016/j.cose.2021.102582.

[5] Netskope. *Netskope Threat Labs Stats for January 2024*. Accessed: Feb. 25, 2024. [Online]. Available: https://www.netskope.com/blog/netskope-threat-labs-stats-for-january-2024

[6] VirusTotal. (Jul. 2023). *VirusTotal Malware Trends Report: Emerging Formats and Delivery Techniques*. Accessed: Nov. 11, 2023. [Online]. Available: https://assets.virustotal.com/reports/2023emerging.pdf

[7] R. Khan, N. Kumar, A. Handa, and S. K. Shukla, "Malware detection in word documents using machine learning," in *Advances in Cyber Security*, vol. 1347. Berlin, Germany: Springer, 2021, pp. 325–339, doi: 10.1007/978-981-33-6835-4_22.

[8] S. HUNEAULT-LEBLANC and C. TALHI, "P-code based classification to detect malicious VBA macro," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Montreal, QC, Canada, Oct. 2020, pp. 1–6, doi: 10.1109/ISNCC49221.2020.9297272.

[9] H. Lee and H.-W. Lee, "Hidden message detection in MS-word file by analyzing abnormal file structure," in *Proc. Int. Conf. Green Human Inf. Technol. (ICGHIT)*, Hanoi, Vietnam, Feb. 2020, pp. 54–57, doi: 10.1109/ICGHIT49656.2020.00021.

[10] N. U. A. Ali, W. Iqbal, and H. Afzal, "Carving of the OOXML document from volatile memory using unsupervised learning techniques," *J. Inf. Secur. Appl.*, vol. 65, Mar. 2022, Art. no. 103096, doi: 10.1016/j.jisa.2021.103096.

[11] M. J. May and E. Laron, "Combating ransomware using content analysis and complex file events," in *Proc. 10th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Canary Islands, Spain, Jun. 2019, pp. 1–5, doi: 10.1109/NTMS.2019.8763851.

[12] Microsoft. *Open XML Formats and File Name Extensions*. Accessed: Feb. 3, 2023. [Online]. Available: https://support.microsoft.com/en-us/office/open-xml-formats-and-file-name-extensions-5200d93c-3449-4380-8e11-31ef14555b18

[13] Microsoft. *VBA Language Specification*. Accessed: Feb. 17, 2023. [Online]. Available: https://learn.microsoft.com/en-us/openspecs/microsoft_general_purpose_programming_languages/ms-vbal/d5418146-0bd2-45eb-9c7a-fd9502722c74?redirectedfrom=MSDN

[14] M. Mimura, "An improved method of detecting macro malware on an imbalanced dataset," *IEEE Access*, vol. 8, pp. 204709–204717, 2020, doi: 10.1109/ACCESS.2020.3037330.

[15] MITER ATT&CK. *T1564.007 Hide Artifacts: VBA Stomping*. Accessed: Apr. 17, 2023. [Online]. Available: https://attack.mitre.org/techniques/T1564/007/

[16] The Virus Encyclopedia. *Concept Virus*. Accessed: May 1, 2023. [Online]. Available: http://virus.wikidot.com/concept

[17] MITER ATT&CK. *S0367 Software: Emotet*. Accessed: Apr. 17, 2023. [Online]. Available: https://attack.mitre.org/software/S0367/

[18] MITER ATT&CK. (2023). *S0089 Software: BlackEnergy*. Accessed: Apr. 17, 2023. [Online]. Available: https://attack.mitre.org/software/S0089/

[19] M. Ryan, "Ransomware case studies," in *Ransomware Revolution: The Rise of a Prodigious Cyber Threat, Advances in Information Security*, vol. 85. Cham, Switzerland: Springer, 2021, pp. 1–20, doi: 10.1007/978-3-030-66583-8_5.

[20] Kaspersky. *Security Bulletin 2021 Statistics*. Accessed: Mar. 28, 2023. [Online]. Available: https://go.kaspersky.com/rs/802-IJN-240/images/KSB_statistics_2021_eng.pdf

[21] M. Gutfleisch, J. H. Klemmer, N. Busch, Y. Acar, M. A. Sasse, and S. Fahl, "How does usable security (Not) end up in software products? Results from a qualitative interview study," 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2022, pp. 893–910, doi: 10.1109/SP46214.2022.9833756.

[22] P. Lagadec. *Oletools—Python Tools to Analyze OLE and MS Office Files*. Accessed: May 3, 2023. [Online]. Available: https://www.decalage.info/python/oletools

[23] P. Lagadec. *ViperMonkey—VBA Emulation Engine*. Accessed: May 3, 2023. [Online]. Available: https://github.com/decalage2/ViperMonkey

[24] MalwareCantFly. *Vba2Graph—Generates the VBA Call Graph*. Accessed: May 3, 2023. [Online]. Available: https://github.com/MalwareCantFly/Vba2Graph

[25] Cuckoo. *What is Cuckoo?* Accessed: Apr. 15, 2023. [Online]. Available: https://cuckoo.sh/docs/introduction/what.html

[26] P. Mohandas, S. K. Santhosh Kumar, S. P. Kulyadi, M. J. S. Raman, V. S. Vasan, and B. Venkataswami, "Detection of malware using machine learning based on operation code frequency," in *Proc. IEEE Int. Conf. Ind. 4.0, Artif. Intell., Commun. Technol. (IAICT)*, Bandung, Indonesia, Jul. 2021, pp. 214–220, doi: 10.1109/IAICT52856.2021.9532521.

[27] M. Mimura and R. Ito, "Applying NLP techniques to malware detection in a practical environment," *Int. J. Inf. Secur.*, vol. 21, no. 2, pp. 279–291, Apr. 2022, doi: 10.1007/s10207-021-00553-8.

[28] M. U. Rana, O. Ellahi, M. Alam, J. L. Webber, A. Mehbodniya, and S. Khan, "Offensive security: Cyber threat intelligence enrichment with counterintelligence and counterattack," *IEEE Access*, vol. 10, pp. 108760–108774, 2022, doi: 10.1109/ACCESS.2022.3213644.

[29] F. Casino, N. Totosis, T. Apostolopoulos, N. Lykousas, and C. Patsakis, "Analysis and correlation of visual evidence in campaigns of malicious office documents," *Digit. Threats, Res. Pract.*, vol. 4, no. 2, pp. 1–19, Jun. 2023, doi: 10.1145/3513025.

[30] J. Yan, M. Wan, X. Jia, L. Ying, P. Su, and Z. Wang, "DitDetector: Bimodal learning based on deceptive image and text for macro malware detection," in *Proc. Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Austin, TX, USA, 2022, pp. 1–13, doi: 10.1145/3564625.3567982.

[31] V. Ravi, S. P. Gururaj, H. K. Vedamurthy, and M. B. Nirmala, "Analysing corpus of office documents for macro-based attacks using machine learning," *Global Transitions Proc.*, vol. 3, no. 1, pp. 20–24, Jun. 2022, doi: 10.1016/j.gltp.2022.04.004.

[32] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[33] Contextures. *Contextures Excel Resources*. Accessed: Jan. 22, 2024. [Online]. Available: https://www.contextures.com/index.html

[34] Sitestory. *Excel VBA Macros*. Accessed: Jan. 8, 2024. [Online]. Available: https://sitestory.dk/excel_vba/downloads.htm

[35] I. H. Sarker, "Machine learning for intelligent data analysis and automation in cybersecurity: Current and future prospects," *Ann. Data Sci.*, vol. 10, no. 6, pp. 1473–1498, Dec. 2023, doi: 10.1007/s40745-022-00444-2.

[36] S. Vitel, M. Lupascu, D. T. Gavrilut, and H. Luchian, "Short-versus long-term performance of detection models for obfuscated MSOffice-embedded malware," *Int. J. Inf. Secur.*, vol. 23, no. 1, pp. 271–297, Feb. 2024, doi: 10.1007/s10207-023-00736-5.

[37] E. De Angelis, A. Pellegrini, and M. Proietti, "Automatic extraction of behavioral features for test program similarity analysis," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Wuhan, China, Oct. 2021, pp. 129–136, doi: 10.1109/ISSREW53611.2021.00054.

[38] B. A. S. Al-Rimy, M. A. Maarof, M. Alazab, F. Alsolami, S. Z. M. Shaid, F. A. Ghaleb, T. Al-Hadhrami, and A. M. Ali, "A pseudo feedback-based annotated TF-IDF technique for dynamic crypto-ransomware pre-encryption boundary delineation and features extraction," *IEEE Access*, vol. 8, pp. 140586–140598, 2020, doi: 10.1109/ACCESS.2020.3012674.

[39] H. J. Kim, S. E. Hong, and K. J. Cha, "seq2vec: Analyzing sequential data using multi-rank embedding vectors," *Electron. Commerce Res. Appl.*, vol. 43, Sep. 2020, Art. no. 101003, doi: 10.1016/j.elerap.2020.101003.

[40] X. Wang, J. Zhu, Z. Xu, K. Ren, X. Liu, and F. Wang, "Local nonlinear dimensionality reduction via preserving the geometric structure of data," *Pattern Recognit.*, vol. 143, Nov. 2023, Art. no. 109663, doi: 10.1016/j.patcog.2023.109663.

[41] B. Zhang, W. Xiao, X. Xiao, A. K. Sangaiah, W. Zhang, and J. Zhang, "Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes," *Future Gener. Comput. Syst.*, vol. 110, pp. 708–720, Sep. 2020, doi: 10.1016/j.future.2019.09.025.

[42] A. Vieth, A. Vilanova, B. Lelieveldt, E. Eisemann, and T. Höllt, "Incorporating texture information into dimensionality reduction for high-dimensional images," in *Proc. IEEE 15th Pacific Visualizat. Symp. (PacificVis)*, Apr. 2022, pp. 11–20, doi: 10.1109/PacificVis53943.2022.00010.

[43] C.-M. Hsu, C.-C. Yang, H.-H. Cheng, P. E. Setiasabda, and J.-S. Leu, "Enhancing file entropy analysis to improve machine learning detection rate of ransomware," *IEEE Access*, vol. 9, pp. 138345–138351, 2021, doi: 10.1109/ACCESS.2021.3114148.

[44] X. Chao and Y. Li, "Semisupervised few-shot remote sensing image classification based on KNN distance entropy," *IEEE J. Sel. Topics Appl. Earth Obser. Remote Sens.*, vol. 15, pp. 8798–8805, 2022, doi: 10.1109/JSTARS.2022.3213749.

[45] M. Nejati, M. Alfadel, and S. McIntosh, "Code review of build system specifications: Prevalence, purposes, patterns, and perceptions," in *Proc. IEEE/ACM 45th Int. Conf. Softw. Eng. (ICSE)*, May 2023, pp. 1213–1224, doi: 10.1109/ICSE48619.2023.00108.

[46] N. Z. Gorment, A. Selamat, and O. Krejcar, "Obfuscated malware detection: Impacts on detection methods," in *Recent Challenges in Intelligent Information and Database Systems* (Communications in Computer and Information Science), vol. 1863, N. T. Nguyen, Eds. Cham, Switzerland: Springer, 2023, pp. 55–66, doi: 10.1007/978-3-031-42430-4_5.
[47] S. Rani, K. Tripathi, Y. Arora, and A. Kumar, "Analysis of anomaly detection of malware using KNN," in *Proc. 2nd Int. Conf. Innov. Practices Technol. Manage. (ICIPTM)*, vol. 2, Gautam Buddha Nagar, India, Feb. 2022, pp. 774–779, doi: 10.1109/ICIPTM54933.2022.9754044.
[48] N. Ruaro, F. Pagani, S. Ortolani, C. Kruegel, and G. Vigna, "SYMBEXCEL: Automated analysis and understanding of malicious excel 4.0 macros," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2022, pp. 1066–1081, doi: 10.1109/SP46214.2022.9833765.
[49] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99129–99149, 2022, doi: 10.1109/ACCESS.2022.3207287.
[50] M. Shah, K. Gandhi, K. A. Patel, H. Kantawala, R. Patel, and A. Kothari, "Theoretical evaluation of ensemble machine learning techniques," in *Proc. 5th Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Tirunelveli, India, Jan. 2023, pp. 829–837, doi: 10.1109/ICSSIT55814.2023.10061139.
[51] I. L. Cherif and A. Kortebi, "On using eXtreme gradient boosting (XGBoost) machine learning algorithm for home network traffic classification," in *Proc. Wireless Days (WD)*, Manchester, U.K., Apr. 2019, pp. 1–6, doi: 10.1109/WD.2019.8734193.
[52] A. Cohen, N. Nissim, L. Rokach, and Y. Elovici, "SFEM: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods," *Exp. Syst. Appl.*, vol. 63, pp. 324–343, Nov. 2016, doi: 10.1016/j.eswa.2016.07.010.

CANDRA AHMADI is currently pursuing the Ph.D. degree with the National Taiwan University of Science and Technology (NTUST), Taiwan. His current research interests include the Industrial Internet of Things (IIoT), 6G technologies, cyber security, artifical intelligence, machine learning, and deep learning.

JIANN-LIANG CHEN (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from National Taiwan University, in 1989. After serving as a Professor and the Vice Dean with the Department of Computer Science and Information Engineering, National Dong Hwa University. In 2009, he joined the Department of Electrical Engineering, National Taiwan University of Science and Technology. His research interests include cellular mobility management, cybersecurity, personal communication systems, the Internet of Things, and AI applications.

YI-CHENG LAI was born in Taipei, Taiwan, in 1998. He received the B.S. degree, in 2021. He is currently pursuing the M.S. degree in electrical engineering with the National Taiwan University of Science and Technology, Taipei. His main research interests include cyber security, vulnerability research, and defense.