I. Related Work

A. Malicious Office Macro Detection: Combined Features with Obfuscation and Suspicious Keywords

The objective of the study is to enhance the detection of malicious macros in Office documents by developing a machine learning-based model that combines obfuscation features and suspicious keywords. This approach aims to improve the accuracy and resilience of detecting malicious macros.The dataset consists of 2968 benign Office documents and 15,570 malicious Office documents, all of which are macro-based.Four machine learning models are used for detecting malicious macros: Random Forest (RF), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), and K-Nearest Neighbor (KNN). The study evaluates the performance of the models using precision, recall, accuracy, and F1-score. The Random Forest classifier with combined features (F1–F15) achieves the highest F1-score among all models. Incorporating obfuscation features and suspicious keywords enhances the model's resilience and detection performance. For example, the RF model achieves a precision of 0.953 on a set of new malicious samples, indicating the effectiveness of combining these features.

Xiang Chen, Wenbo Wang, Weitao Han, 7 November 2023

B. Macro Malware Detection using Machine Learning Techniques: A New Approach

The purpose of this paper is to study the behavior of attackers when creating malicious macros and their functioning. Moreover, it wants to demonstrate if detecting, analyzing, and using the most common methods of malware programming and obfuscation may facilitate the correct and automatic classification and distinction of documents containing legitimate macros, from those with malicious macros. The study gathered 1,671 Office documents in .doc, .docx, and .docm formats from various sources, including spam email accounts, public malware repositories (malwr.com, contagiodump), and general document repositories (P2P networks, search engines). Support Vector Machines (SVM) provided the best balance of precision, accuracy, and a low rate of false positives and negatives, outperforming even neural networks in the final validation phase.

Sergio De los Santos, José Torres, ElevenPaths, Telefónica Digital Cyber Security Unit, Madrid, Spain

C. Detection of Malicious VBA Macros Using Machine Learning Methods

The research aimed to develop a classifier for binary classification of macro-enabled documents, distinguishing between benign and malicious behaviors. Malicious samples were collected from malshare.com and verified using VirusTotal. Benign samples were sourced from VBA tutorial websites and Google searches for relevant documents. Feature Analysis: The study identified features that significantly contributed to the detection of malicious macros. These included: Average Variable Assignment Length: Malicious macros often declare abnormally long string variables. Count of Integer and String Variables: Malicious macros tended to define more integer and string variables. Macro Keywords: Certain keywords like AutoOpen and DocumentOpen were more prevalent in malicious macros. Consecutive Mathematical Operations: Malicious macros often employ anti-analysis techniques involving consecutive mathematical operations. The Random Forest classifier achieved the best performance with a TPR of 98.9875

Ed Aboud, Darragh O'Brien, Dublin City University

D. Detecting Unseen Malicious VBA Macros with NLP Techniques

The dataset includes samples with file extensions such as .doc, .docx, .xls, .xlsx, .ppt, and .pptx. The malicious samples were labeled as such by more than 50% antivirus vendors, while benign samples were unanimously identified as benign. The study employed several machine learning classifiers to detect malicious macros: SVM, RF, MLP. The classifiers were trained using feature vectors obtained from the VBA macros. The feature vectors were generated using both BoW and Doc2vec methods. The study found that Doc2vec was effective for this task and that linear classifiers worked well with Doc2vec-generated vectors. The experimental results demonstrated that the proposed method could detect 89% of new malware families, achieving a best F-measure of 0.93

Mamoru Mimura, Hiroya Miura1

E. Obfuscated VBA Macro Detection Using Machine Learning

Microsoft Office document files containing VBA macros, specifically .docm and .xlsm files, were collected via keyword searches and malware portals from 2016 to 2017. 2,537 files (773 benign, 1,764 malicious).VBA macros were extracted using oletools, an open-source Python package for analyzing MS Office files.SVM achieved 95.5

Sangwoo Kim, Seokmyung Hong, Jaesang Oh and Heejo Lee, Korea University Seoul, Republic of Korea

F. Automated Microsoft Office Macro Malware Detection Using Machine Learning

The data set used in the study consists of 40 malicious and 118 benign Microsoft Office files, all created in Office 2010. Malicious files were obtained from Payload Security and had a threat score of 85 or higher, indicating high maliciousness. Benign files were sourced from Contextures and contained a variety of macro functionalities such as ActiveX and database connections.The study employed the K-Nearest Neighbors (KNN) algorithm for classification. The KNN algorithm, combined with the TFIDF representation of p-codes and the feature selection methods, provided a viable approach to distinguishing between malicious and benign macro-containing Office files.

Ruth Bearden, Department of Computer Science, Kennesaw State University, Dan Chai-Tien Lo, Department of Computer Science, Kennesaw State University

G. A Feasibility Study on Evasion Attacks Against NLP-Based Macro Malware Detection Algorithms

Investigate the feasibility of evasion attacks against natural language processing (NLP)-based macro malware detection algorithms. The study focuses on evaluating the resilience of different feature extraction methods against evasion attacks.Genuine benign and malicious macros sourced from VirusTotal, samples initially uploaded, between 2016 and 2017, File extensions: doc, docs, xls, xlsx, ppt, or pptx, Malicious samples, flagged as malware by over half of 58 anti-virus programs, Benign samples: consistently identified as benign by all anti-virus programs, No sample duplication. LSA: Detection rate decreased to 27

H. Analysing corpus of office documents for macro-based attacks using Machine Learning

Macro-based malware attacks, using malicious code written in Visual Basic, are increasing in recent cyber-attacks. These macros can be obfuscated to evade antivirus detection. Current detection methods using machine learning often suffer from inadequate and unbalanced datasets. This paper proposes using the Word2Vec embedding technique, specifically an Obfuscated-Word2Vec model, for analyzing and detecting malicious Visual Basic macro code. The technique identifies obfuscated keywords and function names, classifying them as either obfuscated or benign. These classifications are then used to train models that accurately detect various types of malware, such as downloaders, droppers, and PowerShell exploits. Experimental results with a Random Forest classifier showed an 82.65

V Ravi, S.P. Gururaj, H.K. Vedamurthy, M.B. Nirmala

I. Using API Calls for Sequence-Pattern Feature Mining-Based Malware Detection

This paper presents an Artificial Neural Network (ANN)-based approach to malware detection focused on sequence-pattern feature mining. It targets polymorphic malware like Emotet and Trickbot, which evade detection by altering their file format. Unlike static characteristics, the behavioral patterns of malware, particularly OS API call sequences, remain consistent and can be monitored. The study builds ANN models using these behavioral features obtained via emulators, demonstrating that this method is promising for real-world applications. Experiments with Feed Forward Neural Networks show improved detection of polymorphic malware families. The paper discusses the limitations of current detection technologies, highlighting the effectiveness of behavioral analysis over static feature detection. This approach ensures better identification of new malware by analyzing the sequences of API calls, a key indicator of malicious behavior. The methodology and results showcase the potential of this ANN-based approach in enhancing malware detection.

Gheorghe Balan, Dragoș Teodor Gavriluț, and Henri Luchian
Faculty of Computer Science Iași, Bitdefender Labs, "Al.I. Cuza" University, Iași, Romania.

Faculty of Computer Science Iași, "Al. I. Cuza" University, Iași, Romania

J. Automated analysis of malicious Microsoft Office documents

Microsoft Office is a widely used suite for documents, spreadsheets, and presentations, making it a frequent target for malicious campaigns. Threat actors exploit its dynamic features, such as VBA macros and DDE, to launch attacks and penetrate systems. This study examines the tools and methods used by malware authors, presenting a taxonomy of these tools and exploring their tactics. A publicly shared dataset was created, combining benign and malicious documents with dynamic features to ensure fair and realistic analysis. Using an automated analysis pipeline, features are extracted to classify documents as benign or malicious with machine learning, achieving an F1 score above 0.98, outperforming current detection algorithms. The study highlights that malicious Office documents often act as droppers, downloading and executing a payload. This is the first academic work to use deobfuscation to reveal the actual behavior of these documents. The dataset includes features like VBA stomping and DDE, allowing for the detection of malware exploiting new vulnerabilities. The methodology and findings demonstrate the efficiency and potential of the proposed method, with the dataset available on Zenodo for further research.

Mamoru Mimura, Risa Yamamoto

K. File-level malware detection using byte streams

Developing a method for detecting malware in Microsoft Office files by analyzing byte streams using machine learning techniques. The dataset used in this study comprises actual Visual Basic for Applications (VBA) macros obtained from Virus Total (VT). These macros include both benign and malicious samples. The document files containing these VBA macros have various extensions, such as doc, docx, xls, xlsx, ppt, and pptx. The samples were uploaded to Virus Total between April 2015 and March 2017. The feature extraction process utilizes the Doc2vec model. The results of the study demonstrate that the proposed method effectively enhances the detection of malicious VBA macros. Several classifiers, including SVM, RF, MLP, and CNN,

Young-Seob Jeong, Medard Edmund Mswahili & Ah Reum Kang

L. DitDetector: Bimodal Learning based on Deceptive Image and Text for Macro Malware Detection

The objective of the study is to develop a robust model called DitDetector for detecting malicious documents. The model is designed to outperform existing machine learning (ML) models and commercial antivirus (AV) engines, particularly focusing on documents containing macros and other embedded malicious elements. MalDoc: Contains various types of malicious documents, 845 files for training, 93 for validation, and 124 for testing. XL4 Macro: Includes Excel 4.0 macro

files collected from January 1, 2022, to April 30, 2022, 1504 malicious files. RTInjection: Comprises documents with CVE-2017-0199 collected in the same period, 510 malicious files. DitDetector achieved an F1-score of 99.34

Jia Yan, Xiangkun Jia, Purui Sut,Institute of Software, Chinese Academy of Sciences, Beijing, China. Ming Wan, Lingyun Ying, Zhanyi Wang, QI-ANXIN Technology Group Inc. Beijing, China

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

This study addresses the challenge of detecting malicious VBA macros in Office documents using a novel approach combining P-Code analysis and machine learning. The research employs a dataset of 2015 malicious samples from VirusTotal and HatchingTriage, and 2014 benign samples from open-source platforms, verified using VirusTotal. The methodology involves:

Data processing: Extracting VBA source code and P-code from samples. **Feature definition:** Developing VBA-behavior-based and P-code-structure-based features. **Feature selection:** Using XGBoost and learning curve method to identify key features. **P-code analysis:** Applying TF-IDF and n-gram techniques for instruction sequence embedding. **Dimensionality reduction:** Utilizing UMAP to manage high-dimensional data. **Machine learning:** Implementing XGBoost for classification.

The approach combines static analysis of VBA source code with dynamic P-Code structural analysis, enhanced by NLP techniques. The model achieves 98.70

Candra Ahmadi, Jiann-Liang Chen, Yi-Cheng Lai

Comparison of Malware Detection Studies

Algorithms	Data Sets	Best Score
Malicious Office Macro Detection: Combined Features with Obfuscation and Suspicious Keywords		
RF, MLP, SVM, KNN	DS1: 2,939 benign, 13,734 malicious. DS2: 2,885, published after DS1 publication date (data sets available)	DS1: F1-score 0.997, DS2: RF with features 1-15 with precision of 0.953
Macro Malware Detection using Machine Learning Techniques: A New Approach		
SVM, DT, RF, NN	1,671 Office documents in .doc, .docx, and .docm	SVM: algorithm achieved the best score, correctly classifying 93%
Detection of Malicious VBA Macros Using Machine Learning Methods		
KNN, DT, RF, GNB	train - 200 benign 200 known malicious samples for training. test set - 528 malicious samples and 83 known benign samples	RF: 98.9875% (TPR), 1.07% (FPR)
Detecting Unseen Malicious VBA Macros with NLP Techniques		
SVM, RF, MLP.	.doc, .docx, .xls, .xlsx, .ppt, and .pptx files. The malicious samples were labeled as such by more than 50% of antivirus vendor	Doc2VEc with Linear classifiers: 0.93 (F-measure)
Obfuscated VBA Macro Detection Using Machine Learning		
SVM, RF, MLP, LDA, BNB	2,537 files (773 benign, 1,764 malicious)	MLP F2 score of 92%
Analysing corpus of office documents for macro-based attacks using Machine Learning		
RF	The proposed method utilizes datasets from public domains (Malware-bazaar, Malware-traffic-analysis, Malware-lake) and benign samples generated using Java Apache POI.	82.65%.
Automated analysis of malicious Microsoft Office documents		
RF, CVS, MLP	VirusTotal datasets and diverse techniques in Microsoft Office documents.	best score is Random Forest with F1-score 0.985
Using API Calls for Sequence-Pattern Feature Mining-Based Malware Detection		
FF-NN, NN	Emotet and Trickbot	99.41%
Automated Microsoft Office Macro Malware Detection Using Machine Learning		
KNN, combined with TFIDF	40 malicious and 118 benign	96.3%
File-level malware detection using byte streams		
SVM, RF, MLP, CNN	VBA macros from VirusTotal, with extensions such as doc, docx, xls, xlsx, ppt, and pptx, uploaded between April 2015 and March 2017.	Doc2Vec feature extraction enhances the detection of malicious VBA macros.
DitDetector: Bimodal Learning based on Deceptive Image and Text for Macro Malware Detection		
RFC, XGBoost, DitDetector	MalDoc: 845 training files, 93 validation files, 124 testing files. XL4 Macro: 1504 malicious files. RTInjection: 510 malicious files.	DitDetector achieved an F1-score of 99.34%, outperforming RFC (99.10%) and commercial AV engines like Kaspersky, Microsoft, and Symantec.
Enhancing Cybersecurity With P-Code Analysis and XG- Boost: A Novel Approach for Malicious VBA Macro Detection in Office Documents		
XGBoost	2015 malicious samples from VirusTotal and HatchingTriage, 2014 benign samples from open-source platforms (verified using VirusTotal)	98.70% accuracy

II. Results

A. Basic algorithms

In this task, we evaluated the performance of several classifiers (AdaBoost, Decision Tree, Gradient Boosting, KNN, MLP, Random Forest, and SVM) by varying the number of features from 10 to 3000. For each classifier, we measured the recall scores across different feature sets. The classifiers showed distinct behavior as the number of features increased. AdaBoost, Random Forest, and SVM consistently maintained high recall scores, even with a small number of features, reaching near-optimal performance early on. On the other hand, classifiers like Decision Tree and KNN exhibited more variation, improving recall as the number of features increased but leveling off as they approached the maximum number of features. The results indicate that while some classifiers benefit from additional features, others, like AdaBoost and SVM, achieve strong performance with a smaller feature set. The graph illustrates the recall scores as a function of the number of features for each classifier. Among the classifiers, Random Forest exhibited the highest performance during training, achieving near-perfect recall scores across all feature sets. It consistently outperformed other models, particularly when the number of features was small, demonstrating its robustness in feature selection. The model's recall quickly stabilized around the maximum score as the number of features increased, highlighting its efficiency and effectiveness in capturing relevant patterns in the data. This consistent high performance makes Random Forest a reliable choice for tasks requiring high recall.

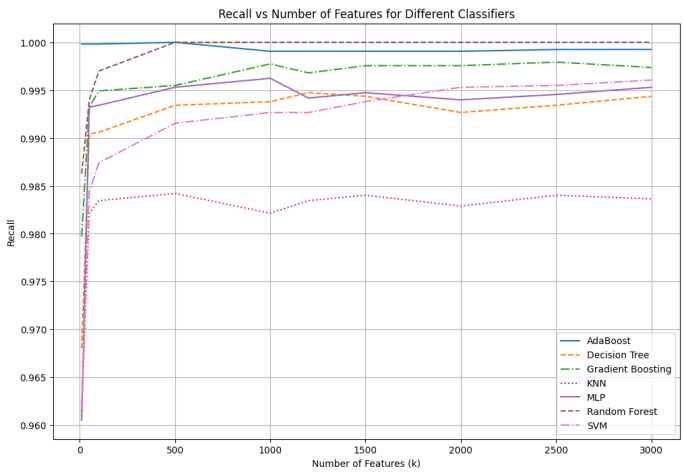

Fig. 1. Recall vs Number of Features for Different Classifiers

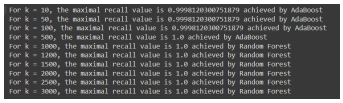

Fig. 2. Highest Recall Value for Each K-Features

[1]

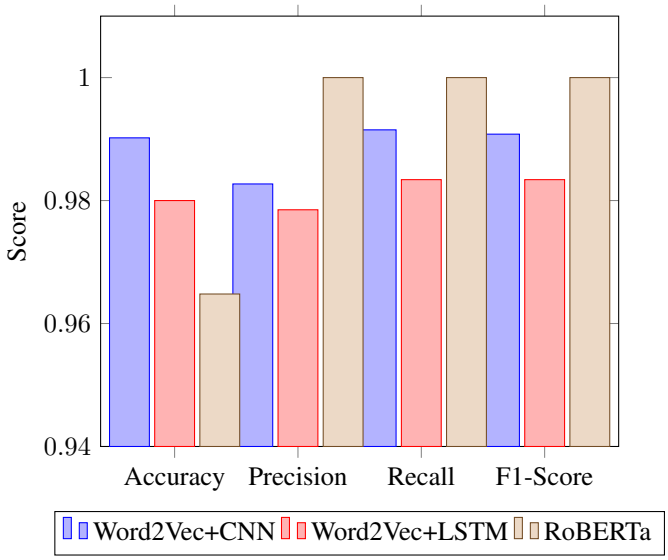

Fig. 3. Performance Comparison of Binary Text Classification Models

B. Advanced Algorithms

III. Summary and Comparison of Binary Text Classification Models

The three models (Word2Vec+CNN, Word2Vec+LSTM, and RoBERTa) show varying performance in binary text classification. Word2Vec+CNN achieves the highest test accuracy at 99.02%, followed closely by Word2Vec+LSTM at 98%. RoBERTa shows perfect precision and recall but lower test accuracy at 96.48%. CNN and LSTM models demonstrate gradual improvement over epochs, while RoBERTa maintains consistent performance. The CNN model exhibits the best overall balance of metrics, with high precision (0.9827) and recall (0.9915). LSTM shows good performance but slightly lower than CNN. RoBERTa, despite perfect precision and recall on the test set, has a lower overall accuracy, suggesting potential overfitting or dataset-specific behavior.

References

[1] Sangwoo Kim, Seokmyung Hong, Jaesang Oh, and Heejo Lee, *Obfuscated vba macro detection using machine learning*, 2018 48th annual ieee/ifip international conference on dependable systems and networks (dsn), IEEE, 2018, pp. 490–501.

IV. Exploratory Data Analysis (EDA)

A. Data Loading and Preprocessing

The initial step in our analysis involved loading the datasets comprising training, validation, and testing sets of VBA macro codes, which were extracted from Microsoft Office documents. These datasets were encoded using UTF-16LE to preserve the integrity of the textual data. To facilitate a more comprehensive analysis, we concatenated the three datasets into a single DataFrame. Furthermore, we converted the categorical labels—'white' (representing benign code) and 'mal' (representing malicious code)—into numeric values (1 and 0, respectively) to standardize the labels across the dataset.

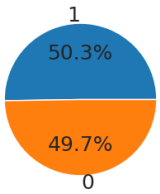

Fig. 4. Loading and standardizing VBA macro datasets with numeric labels for analysis.

B. Feature Extraction: Code Length and Word Count

To differentiate between benign and malicious VBA macros, we extracted fundamental features such as code length and word count from the VBA scripts. These features were then subjected to statistical analysis, revealing that, on average, malicious VBA macros are significantly longer and contain more words than their benign counterparts. To further elucidate these findings, we employed box plots and kernel density estimates, which visually confirmed that malicious scripts tend to exhibit greater complexity and verbosity compared to benign scripts.

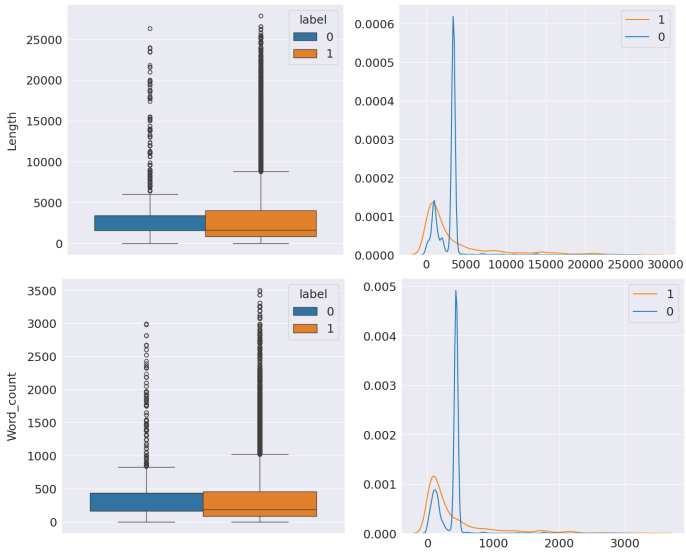

Fig. 5. Statistical analysis of VBA macros showing that malicious scripts are longer and more verbose than benign ones.

C. Word Frequency Analysis

In order to gain deeper insights into the linguistic characteristics of the VBA macros, we conducted a word frequency analysis. This involved tokenizing the VBA scripts into individual words, followed by the removal of punctuation and numeric characters. By identifying the most frequently occurring words in both benign and malicious samples, we observed distinct patterns: benign scripts predominantly contained generic programming structures, whereas malicious scripts were characterized by terms indicative of interactions with Excel worksheets and user prompts. These observations were visualized through bar plots, highlighting the divergence in word usage between the two classes.

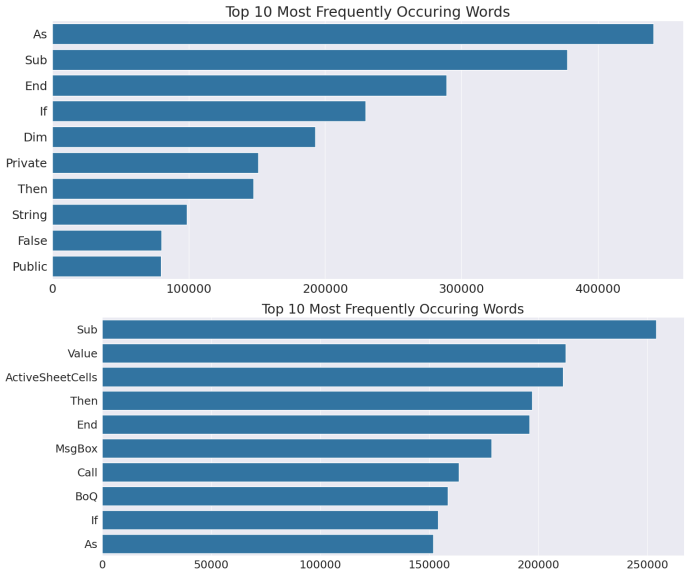

Fig. 6. Word frequency analysis revealing distinct linguistic patterns: benign scripts feature generic programming terms, while malicious scripts include terms related to Excel interactions and user prompts.

D. Word Cloud Visualization

To provide a visual comparison of word prominence within benign and malicious VBA scripts, we generated word clouds for each category. The word clouds revealed that both classes prominently featured common VBA keywords such as "Private Sub," "End Sub," "Option Explicit," and "Dim." However, subtle differences in the frequency and context of these and other words suggested potential distinguishing patterns between benign and malicious code samples.

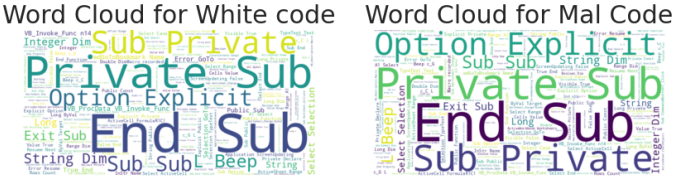

Fig. 7. Word clouds for benign and malicious VBA scripts showing common keywords and subtle frequency differences suggesting distinguishing patterns.

E. Analysis of Top N-Grams

We further extended our linguistic analysis by identifying and analyzing the top bigrams (two-word combinations) in the VBA scripts. The bigram analysis highlighted distinct patterns: benign scripts frequently contained standard coding structures such as "end sub" and "private sub," while malicious scripts were enriched with bigrams associated with active manipulations of Excel sheets and user prompts, such as "activesheet cells" and "value msgbox." The contrasting bigram usage between the two classes was visualized using bar charts, providing valuable insights into the operational characteristics of benign versus malicious VBA macros.

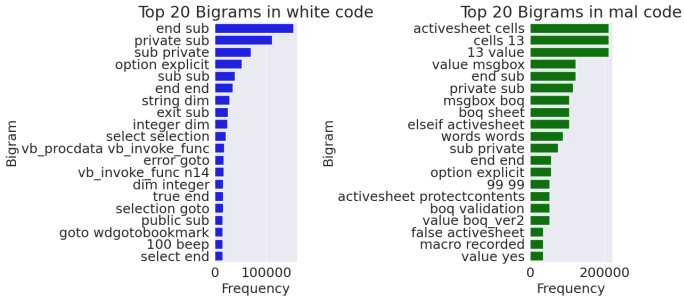

Fig. 8. Top bigrams analysis revealing distinct patterns: benign scripts feature standard coding structures, while malicious scripts include terms associated with Excel interactions and user prompts.

V. Implementations Comparison

A. Malicious Office Macro Detection: Combined Features withObfuscation and Suspicious Keywords

1) **Random Forest (RF):**
   - For F1-F14, ours implementation performs slightly better across all metrics.
   - For F15, the paper's implementation significantly outperforms ours.
   - For F1-F15, performance is comparable, with our implementation having a slight edge in some metrics.
2) **Multi-Layer Perceptron (MLP):**
   - For F1-F14, the paper's implementation generally performs better, especially in precision.
   - For F15, the paper's implementation significantly outperforms ours.
   - For F1-F15, the paper's implementation shows better performance across all metrics.
3) **Support Vector Machine (SVM):**
   - For F15, the paper's implementation significantly outperforms ours.
   - For F1-F15, the paper's implementation shows better performance across all metrics.
4) **K-Nearest Neighbors (KNN):**
   - For F15, the paper's implementation significantly outperforms ours.
   - For F1-F15, the paper's implementation shows better performance, but the gap is smaller compared to F15.

Performance Comparison: Paper vs. Ours Implementation

Model	Feature Selection	Dataset	FAR	Precision	Recall	Accuracy	F1-Score
RF	F1-F14	Original Paper	0.083	0.982	0.994	0.981	0.988
		Ours Datasets	0.014	0.986	0.998	0.992	0.992
	F15	Original Paper	0.012	0.995	0.995	0.993	0.996
		Ours Datasets	0.152	0.842	0.812	0.830	0.827
	F1-F15	Original Paper	0.015	0.997	0.997	0.994	0.997
		Ours Datasets	0.012	0.988	0.998	0.993	0.993
MLP	F1-F14	Original Paper	0.138	0.971	0.979	0.958	0.975
		Ours Datasets	0.052	0.950	0.974	0.961	0.962
	F15	Original Paper	0.013	0.996	0.995	0.994	0.996
		Ours Datasets	0.133	0.855	0.792	0.829	0.822
	F1-F15	Original Paper	0.035	0.993	0.994	0.989	0.993
		Ours Datasets	0.026	0.974	0.982	0.978	0.978
SVM	F15	Original Paper	0.022	0.995	0.987	0.986	0.991
		Ours Datasets	0.152	0.841	0.805	0.827	0.823
	F1-F15	Original Paper	0.028	0.994	0.992	0.988	0.993
		Ours Datasets	0.080	0.923	0.962	0.941	0.942
KNN	F15	Original Paper	0.020	0.996	0.992	0.990	0.994
		Ours Datasets	0.150	0.843	0.805	0.828	0.824
	F1-F15	Original Paper	0.028	0.992	0.992	0.988	0.993
		Ours Datasets	0.029	0.971	0.977	0.974	0.974

Fig. 9. Models performance comparison using different data sets with the same features

General observations:

1) Our implementation struggles significantly with the F15 feature set across all models.
2) The paper's implementation consistently performs well across all feature sets and models.
3) Our implementation is competitive for the F1-F14 and F1-F15 feature sets in some cases, particularly with the Random Forest model.

B. Macro Malware Detection using Machine Learning Techniques A New Approach

1) **Neural Networks:**
   - The paper shows significantly higher accuracy (0.99 vs 0.8635) and F1-score (0.961 vs 0.86).
   - Our implementation's AUC (0.8976) is relatively close to the paper's (0.95), indicating good discriminative ability despite lower accuracy.
2) **SVM:**
   - Our implementation's accuracy (0.8637) is slightly lower than the paper's (0.89).
   - The AUC scores differ more significantly (0.8627 vs 0.95), suggesting the paper's implementation might have better overall classification performance.
3) **Decision Tree:**
   - The academic paper shows higher accuracy (0.95 vs 0.8643) and F1-score (0.968 vs 0.86).
   - Our implementation has a lower AUC (0.8984 vs 0.91), which could indicate wrose ranking ability.
4) **Random Forest:**
   - Accuracy is higher in the academic paper (0.94 vs 0.8643).
   - F1-scores show a similar trend (0.961 vs 0.86).
   - AUC for the academic paper's Random Forest implementation is not provided, making a direct comparison impossible.

General Observations

5) **Consistency:** Our implementation shows very consistent performance across all algorithms (accuracy ∼0.864, F1-score 0.86), while the academic paper shows more variation between algorithms.
6) **Feature Set Differences:** As noted, Our implementation uses only VBA code features, while the academic paper likely includes document-level features. This difference in feature sets could explain much of the performance gap.
7) **Precision and Recall:** Our implementation shows balanced precision and recall for both classes (around 0.85-0.88), which is a positive aspect. The academic paper doesn't provide these details for a direct comparison.

While Our implementation shows lower overall performance metrics compared to the academic paper, it demonstrates consistent results across different algorithms. The performance gap is likely due to the difference in feature sets used. Your implementation's balanced precision and recall, along with respectable AUC scores, suggest a solid foundation for malicious macro detection using VBA code features alone. Further improvements could potentially be achieved by incorporating additional features or fine-tuning the algorithms.

TABLE I
COMPARISON OF MALICIOUS MACRO DETECTION ALGORITHMS: OUR IMPLEMENTATION VS. ACADEMIC PAPER

Algorithm	Accuracy		AUC		F1-Score	
	Ours	Paper	Ours	Paper	Ours	Paper
Neural Networks	0.8635	0.99	0.8976	0.95	0.86	0.961
SVM	0.8637	0.89	0.8627	0.95	0.86	0.927
Decision Tree	0.8643	0.95	0.8984	0.91	0.86	0.968
Random Forest	0.8643	0.94	0.8984	N/A	0.86	0.961

C. Detection of malicious VBA macros using Machine Learning methods

- K-Nearest Neighbors (KNeighbors):
  - Paper: TPR 97.527%, FPR 2.46%
  - Implementation: TPR 98.9264%, FPR 1.6917%

  The implementation shows higher TPR and lower FPR, suggesting better performance.
- Decision Tree:
  - Paper: TPR 98.225%, FPR 1.768%
  - Implementation: TPR 97.1181%, FPR 1.6353%

  The paper's results show slightly higher TPR, while the implementation has a marginally lower FPR.
- Random Forest:
  - Paper: TPR 98.9875%, FPR 1.07%
  - Implementation: TPR 98.9264%, FPR 3.3459%

  The paper's results show slightly better performance in both TPR and FPR.
- Gaussian Naive Bayes (GaussianNB):
  - Paper: TPR 97.02%, FPR 2.97%
  - Implementation: TPR 70.0697%, FPR 15.3195%

  The paper's results significantly outperform the implementation for this classifier. Execution times differ considerably, with the implementation being much faster for most algorithms.

TABLE II
COMPARISON OF MALICIOUS MACRO DETECTION ALGORITHMS: OUR IMPLEMENTATION VS. ACADEMIC PAPER

Algorithm	TPR (%)		FPR (%)	
	Ours	Paper	Ours	Paper
K-Nearest Neighbors	98.9264	97.527	1.6917	2.46
Decision Tree	97.1181	98.225	1.6353	1.768
Random Forest	98.9264	98.9875	3.3459	1.07
Gaussian Naive Bayes	70.0697	97.02	15.3195	2.97