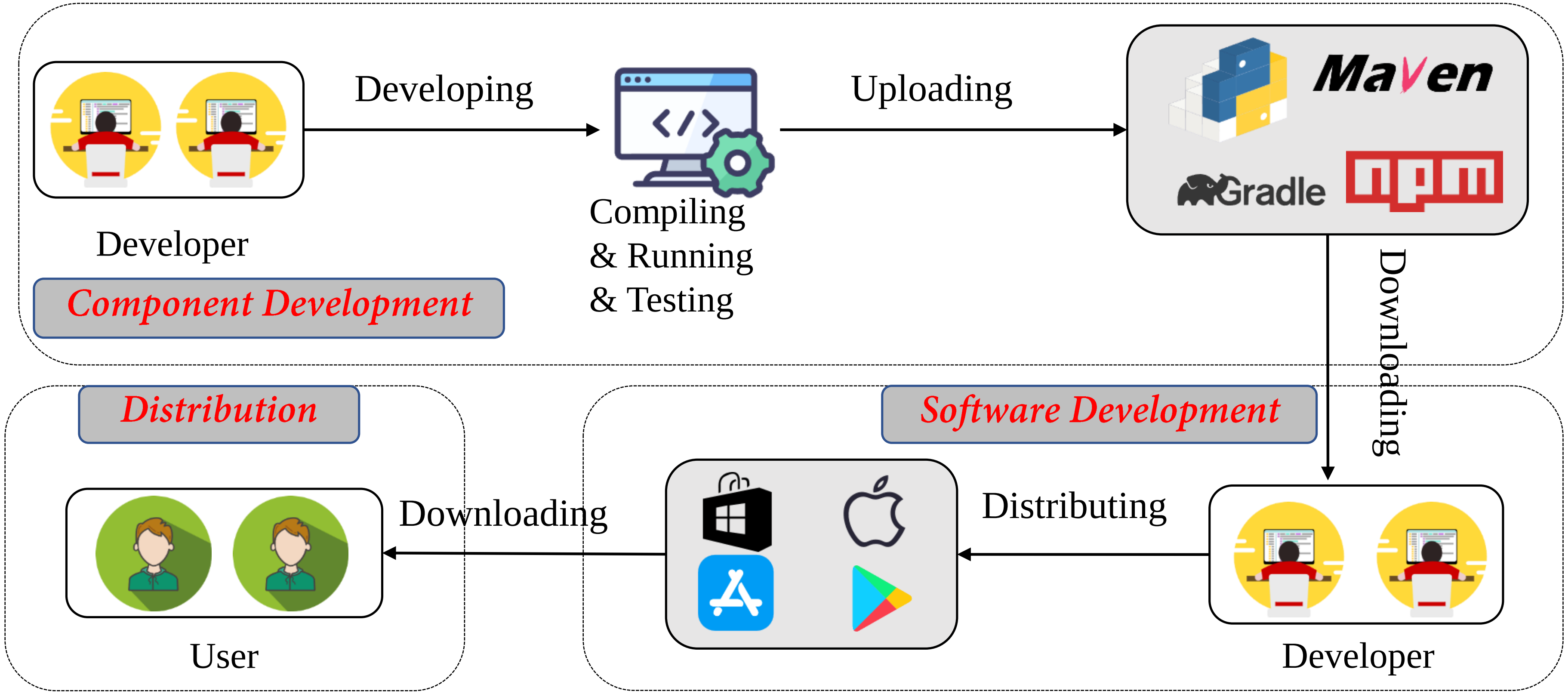
34th USENIX Security Symposium（USENIX Security 2025）

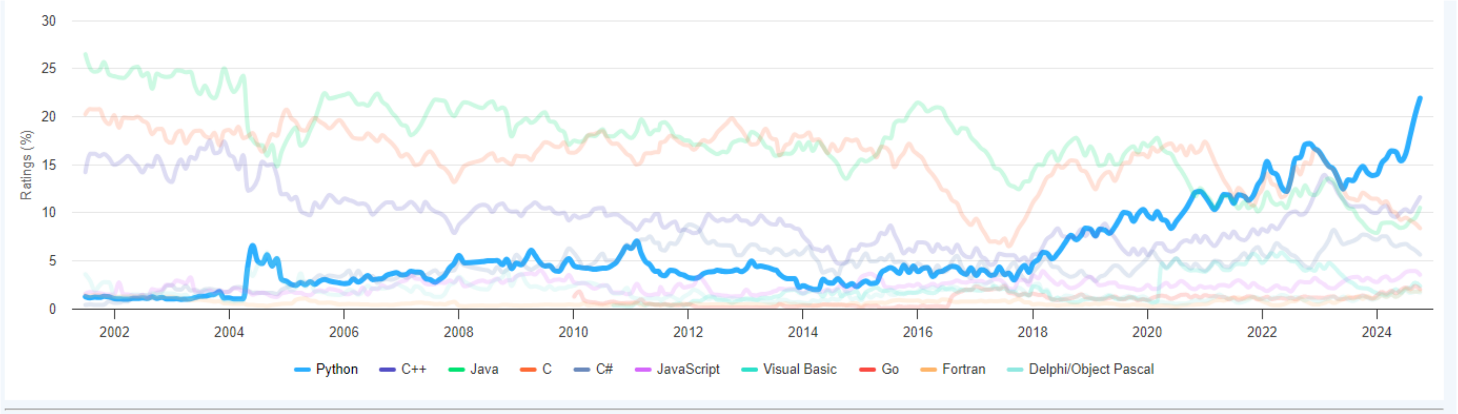
MalGuard: Towards Real-Time, Accurate, and Actionable Detection of Malicious Packages in PyPI Ecosystem

Xingan Gao[1], Xiaobing Sun[1,†], Sicong Cao[1,†], Kaifeng Huang[2], Di Wu[3], Xiaolei Liu[4], Xingwei Lin[5], Yang Xiang[6]

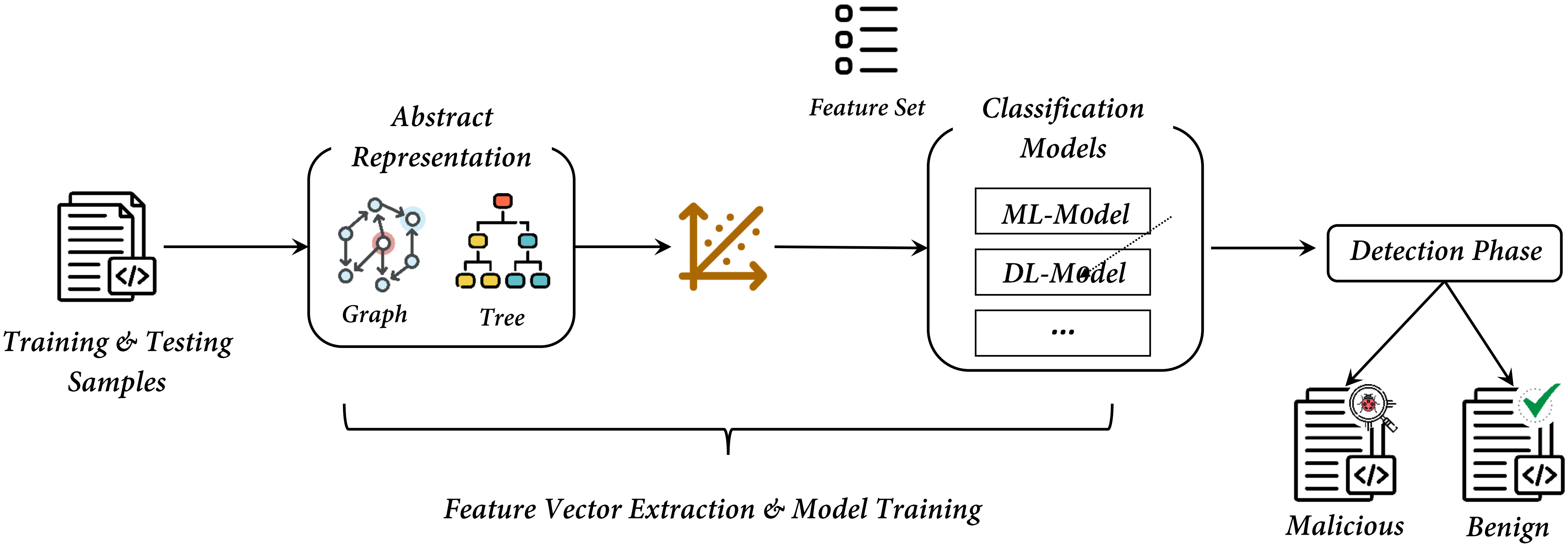
[1] Yangzhou University, [2] Tongji University, [3]University of Southern Queensland
[4]China Academy of Engineering Physics, [5]Zhejiang University, [6]Swinburne University of Technology

1.1 Open-Source Supply Chain

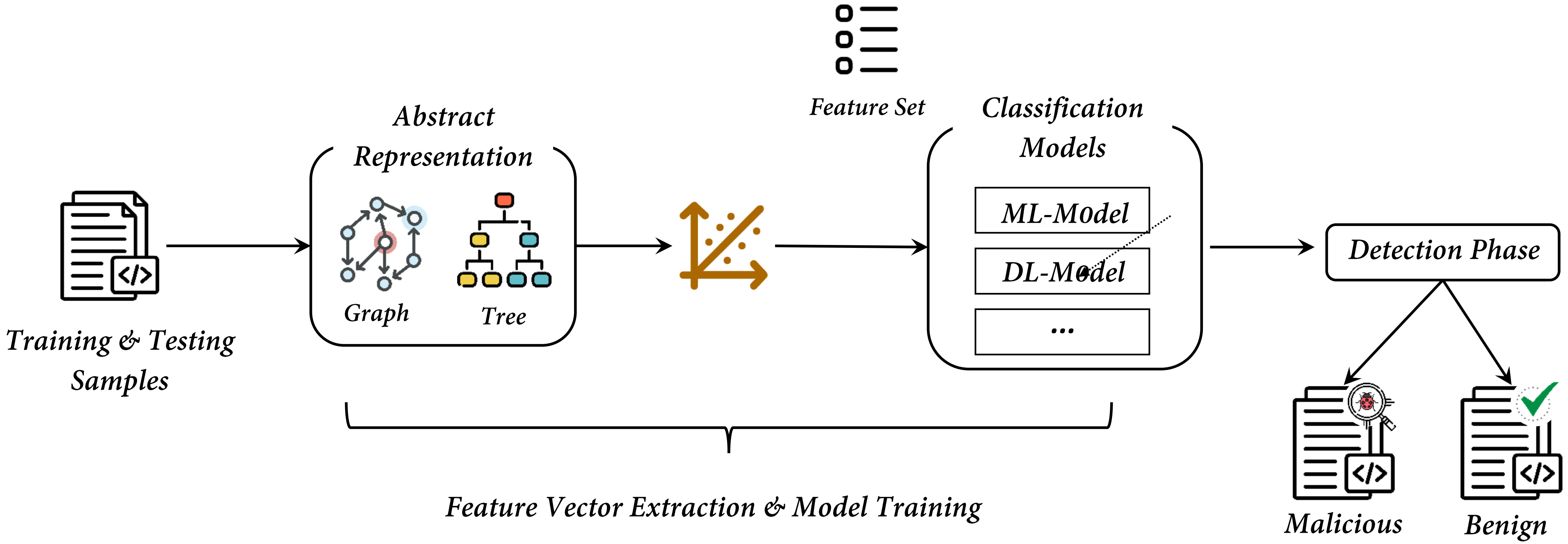




1.2 Traditional Feature Vector based Approaches

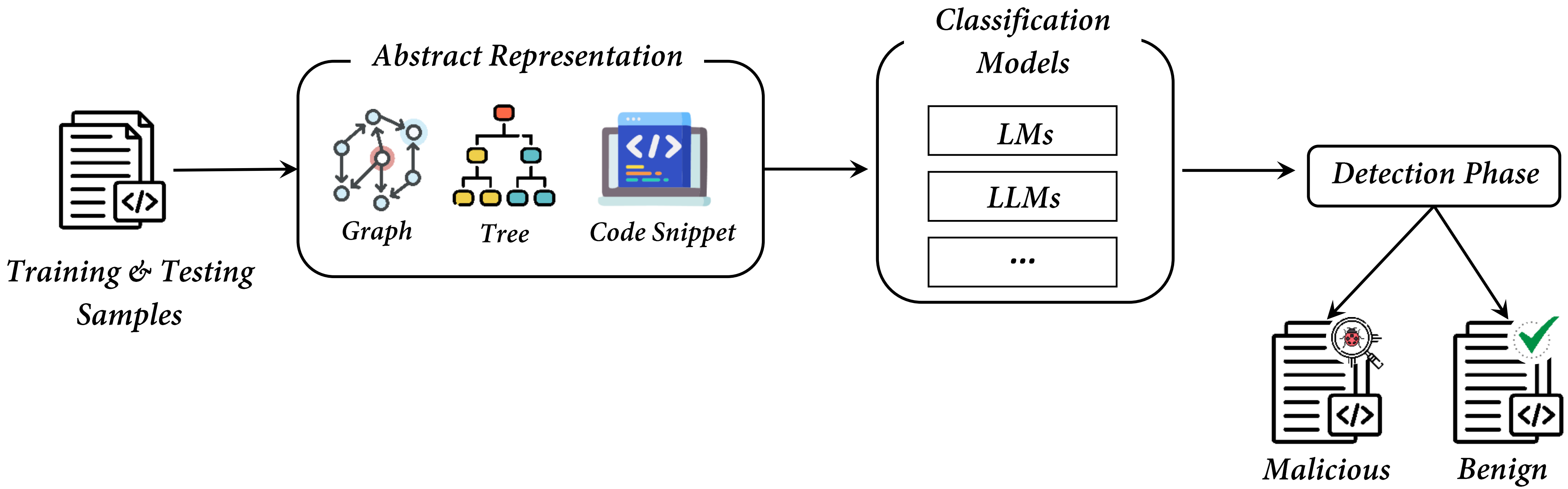


1.2 Traditional Feature Vector based Approaches

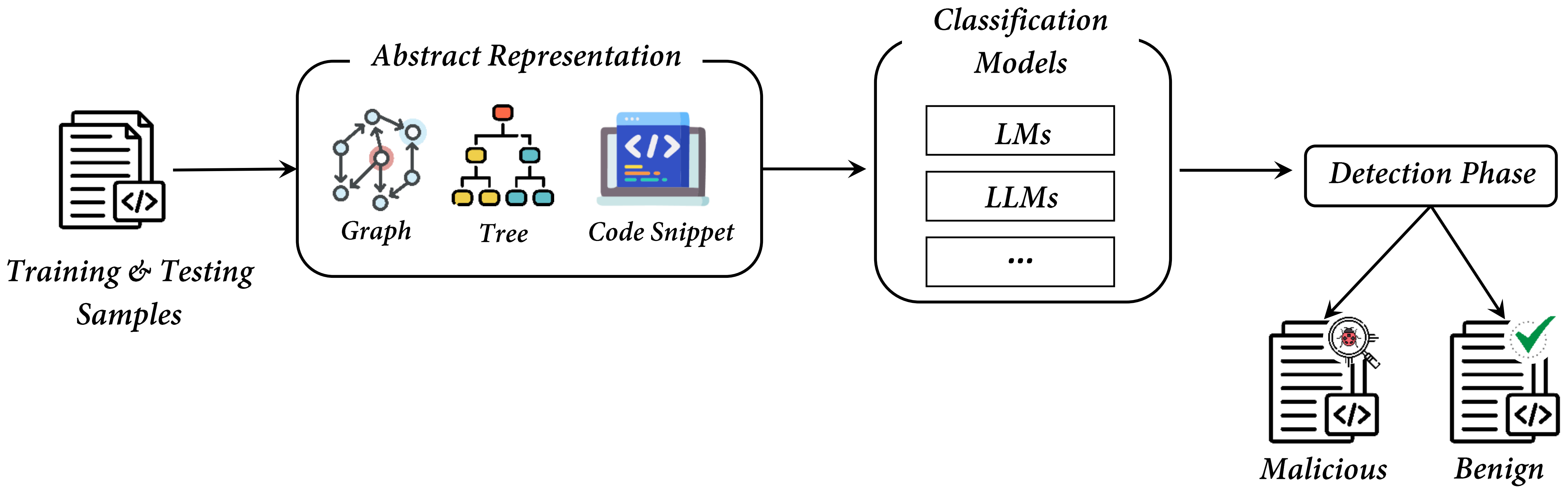


As time progresses, the dataset is continuously augmented with new malicious package samples, necessitating ongoing manual effort from security professionals to analyze their characteristics.

1.3 LLM based Approaches



1.3 LLM based Approaches



Iterative updates to LLMS and LMS are quite time-consuming and resource-intensive, and existing approaches lack the analysis of malicious packages

2 Empirical Study

Table 2: The categories of 132 different APIs in Feature Set.

Categories	API example
File-system access	os.mkdir() os.remove shutil.copy() write() ...
Process creation	subprocess.Popen multiprocessing.Process threading.Thread ...
Network access	socket.socket() requests request.urlopen() ...
Data encode & decode	base64.b64encode() base64.b64decode() ...
Package install	install.run() pip.main() ...
System access	os.getenv() os.getcwd() ...

Table 3: Effectiveness comparison of five different ML models and LLM-based approaches on the same dataset.

Group	Model	Precision (%)	Recall (%)	F1 score (%)	Time Consumption	
					Pre-process (s/package)	Train (s)
ML	NB	55.2	98.4	70.7	**0.8457**	0.19467
	XGBoost	98.1	98.4	98.2		4.79
	RF	98.5	98	98.2		1.0126
	SVM	89.2	94.7	91.9		0.097
	MLP	98.1	98.2	98.1		22.85157
PTM	EA4MP [37]	**99.1**	95.4	97.2	6.28	30,741.67
	CEREBRO [46]	98.6	85.7	91.7	12.489	2,439
LLM	GPT-3.5-turbo [30]	99.0	**99.3**	**99.1**	-	-

Table 5: Effectiveness comparison of different ML models and LLM-based approaches on newer samples by training an old dataset.

	XGBoost			RF			SVM			MLP			EA4MP		
Metrics (%)	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
2021&2022	88.2	80.3	84.1	97.1	82.0	88.9	88.6	80.3	84.2	95.3	80.6	87.3	94.7	90.7	92.7
2023	86.4	59.0	70.1	90.1	59.3	71.5	83.3	49.2	61.9	87.3	62.1	72.6	81.6	84.3	82.9
2024	81.5	53.4	64.5	72.6	52.1	60.7	75.4	51.0	60.8	79.6	57.1	66.5	72.7	70.5	71.6

3.1 API Call Graph Centrality Analysis

Closeness Centrality

$$C_C(v) = \frac{N-1}{\sum_{u \in V, u \neq v} d(v,u)}$$

Degree Centrality

$$C_D(v) = \frac{\deg(v)}{N-1}$$

Katz Centrality

$$C_K(v) = \alpha \sum_{u \in V} A_{vu} C_K(u) + \beta$$

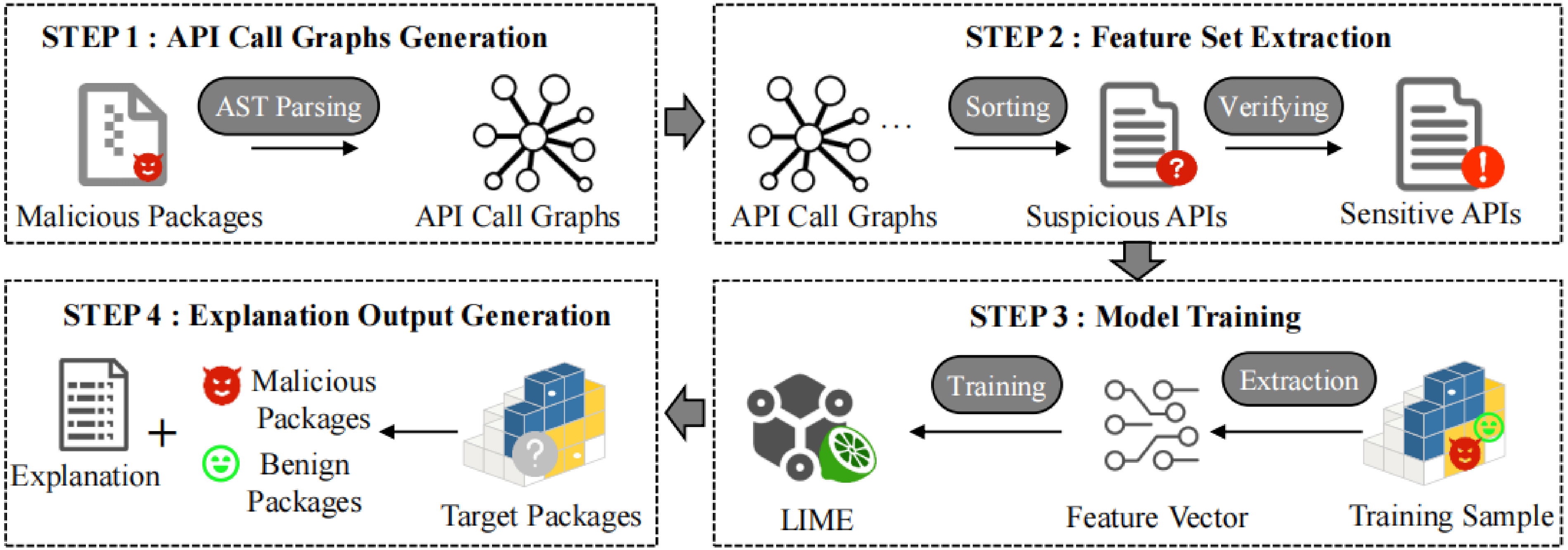
Harmonic Centrality

$$C_H(v) = \sum_{u \in V, u \neq v} \frac{1}{d(v,u)}$$

Table 6: The top 10 APIs calculated with different centrality in malicious&benign packages.

	Closeness	Degree	Harmonic	Katz
Malicious	setup exists subprocess.Popen open join range getattr map os.getenv install.run	setup exists subprocess.Popen join open range aetattr map exec os.getenv	join open decode getattr encode map exists os.getenv replace b64decode	setup exists subprocess.Popen open join install.run exec format os.getenv expanduser
Benign	open len setup print str isinstance int format list super	open len setup join print str isinstance int range format	len join str isinstance open int list print append super	setup open len join print str isinstance int range list

3.2 Overflow of Our Approach: MalGuard



Workflow of MalGuard

An ***API Call Graph* Centrality** and **LIME** based ***Malicious PyPI packages Detection Approach:***

- *API Call Graph Generation*
- *Sensitive API Extraction and Filter*
- *Malicious Package Detection*
- *Explanation Output Generation based on LIME*

3.3 *API Call Graph Generation and Sensitive API Extraction and Filter*

How to get rid of the feature sets that based on expert knowledge?

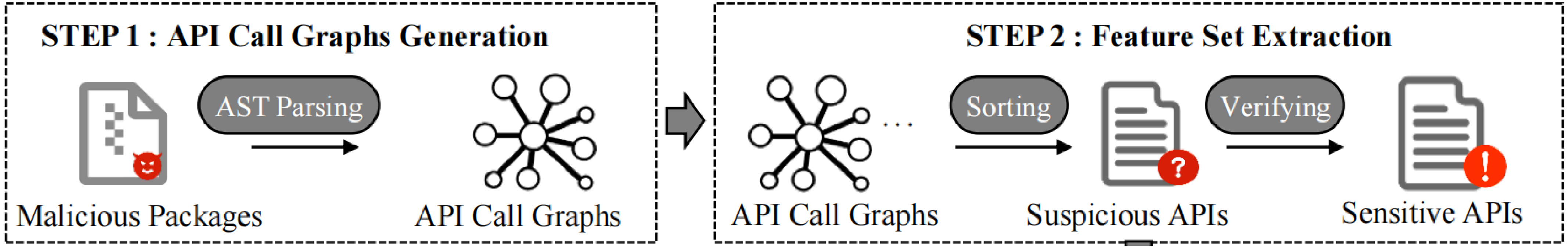
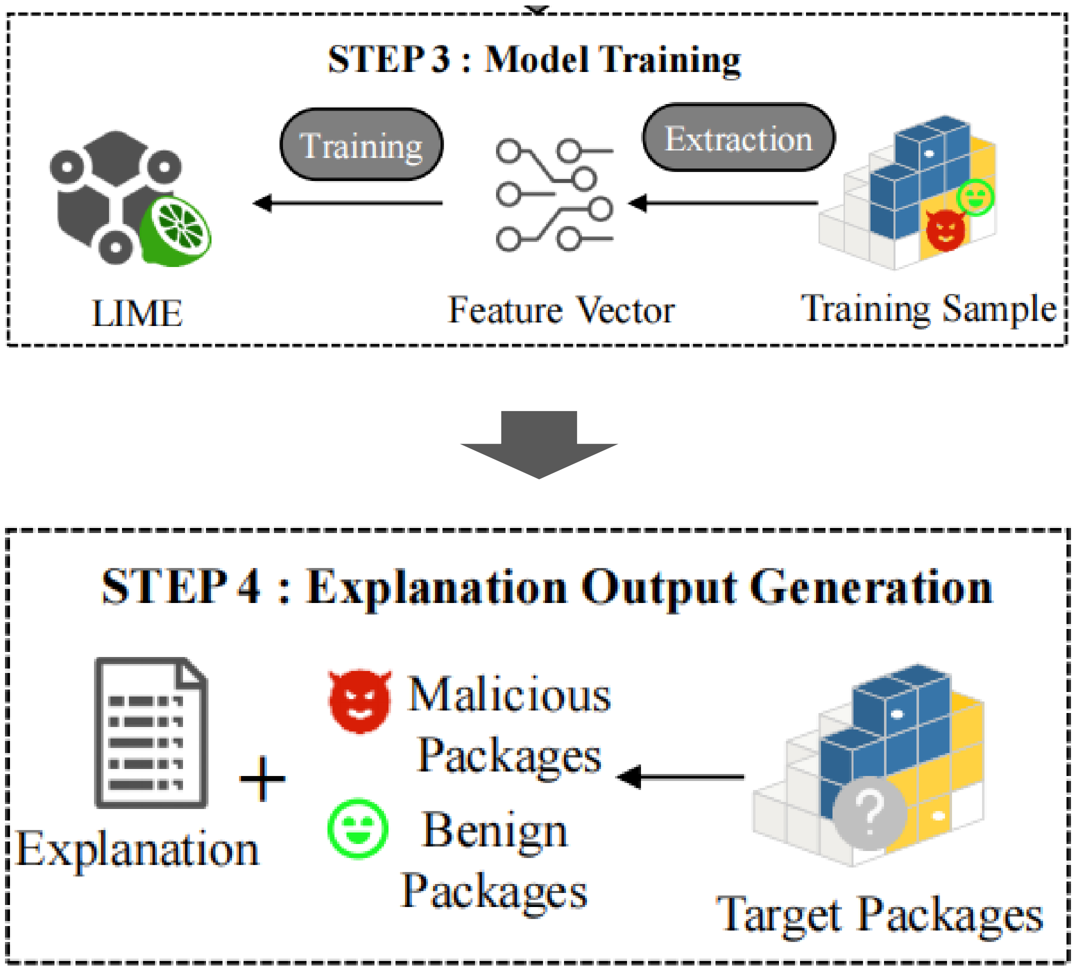


Table 7: The feature set dimension after pre-processing by general-purpose LLM.

Centrality	Closeness	Degree	Katz	Harmonic
Total Dimensions	265	255	294	135

3.4 *Malicious Package Detection and Explanation Output Generation based on LIME*



LIME Explanation for package *pandas-numpy-8.19.3*:

In file pandas-numpy-8.19.3/reqinstaller/__init__.py line 6, the package holder use the sensitive api:
[requests.get],
in function/global global,
which may be used for:
['Performing unauthorized data extraction from a remote server',
'Conducting SQL injection attacks',
"Gather sensitive information from the server's response data"]

In file pandas-numpy-8.19.3/reqinstaller/__init__.py line 11, the package holder use the sensitive api:
[subprocess.call],
in function/global global,
which may be used for:
['Execute harmful system commands or shell scripts']

In file pandas-numpy-8.19.3/setup.py line 7, the package holder use the sensitive api:
[setup],
in function/global global,
which may be used for:
['Potential for unauthorized access to sensitive data',
'Possibility of injecting malicious code or backdoors during the setup']

In file pandas-numpy-8.19.3/setup.py line 15, the package holder use the sensitive api:
[find_packages],
in function/global global,
which may be used for:
['Search for and gain unauthorized access to sensitive packages.']

In file pandas-numpy-8.19.3/setup.py line 38, the package holder use the sensitive api:
[base64.b64decode],
in function/global global,
which may be used for:
['Decoding base64-encoded strings.']

In file pandas-numpy-8.19.3/setup.py line 38, the package holder use the sensitive api:
[exec],
in function/global global,
which may be used for:
['Arbitrary code execution',
'Injection attacks']

Figure 2: The explanation output result of malicious package pandas-numpy-8.19.3

4.1 Experimental Setup

Dataset

Dataset	#Malicious	#Benign
Guo et al. [23]	9,148	-
Sun et al. [37]	516	-
Our work	-	10,000
Total	9,664	10,000

Baselines

- VIRUSTOTAL
- OSSGADGET
- BANDIT4MAL
- EA4MP
- CEREBRO
- GUARDDOG

Evaluation Metrics

- Precision
- Recall
- F1-Score

Experiments

1. Effectiveness Evaluation
2. Ablation Study
3. Explainability Evaluation
4. Hyperparameter Sensitivity Analysis
5. Robustness against Adversarial Attack
6. Practicality

4.2 Effectiveness Evaluation & Ablation Study

Table 8: Effectiveness comparison with the SOTA baselines.

Approach	Precision (%)	Recall (%)	F1 score (%)
VirusTotal [15]	95.2	80.6	87.3
OSSGadget [5]	74.8	85.0	79.6
Band4Mal [39]	84.8	96.7	90.4
Ea4mp [37]	99.1	95.4	97.2
Cerebro [46]	98.6	85.7	91.7
GuardDog [16]	95.6	82.6	88.6
MalGuard	**99.6**	**98.4**	**99.0**

Table 7: The feature set dimension after pre-processing by general-purpose LLM. From 500 dimension

Centrality	Closeness	Degree	Katz	Harmonic
Total Dimensions	265	255	294	135

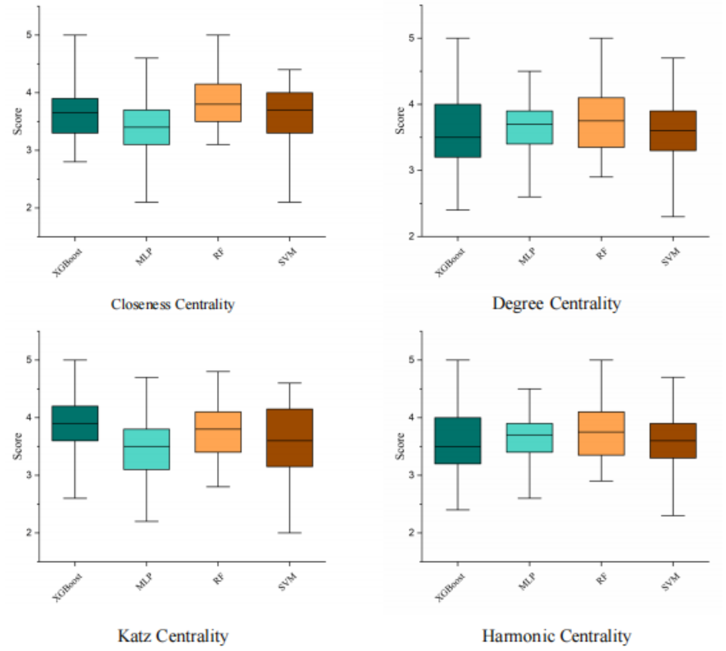
		with Feature Filtering				w/o Feature Filtering			
Metrics (%)		Closeness	Harmonic	Degree	Katz	Closeness	Harmonic	Degree	Katz
RF	Precision	99.4	92.5	99.3	99.6	99.9	99.9	99.9	94.9
	Recall	97.0	97.1	97.3	98.4	98.1	98.0	98.2	95.8
	F-1	98.2	94.8	98.3	99.0	99.0	99.0	99.1	95.3
XGBoost	Precision	99.4	99.2	92.5	99.3	99.2	99.3	99.2	93.0
	Recall	96.5	96.3	95.5	96.9	98.5	98.7	98.6	94.5
	F-1	97.9	97.7	94.0	98.1	98.8	99.0	98.9	93.7
SVM	Precision	97.9	87.1	97.6	97.9	82.8	86.6	72.6	71.8
	Recall	96.5	91.2	96.2	96.3	80.9	83.5	96.1	95.9
	F-1	97.2	89.1	96.9	97.1	81.8	85.0	82.7	82.1
MLP	Precision	98.5	92.0	98.2	98.4	99.0	99.1	98.3	89.8
	Recall	97.8	97.0	98.0	98.1	98.9	95.5	98.6	92.5
	F-1	98.1	94.4	98.1	98.2	99.0	97.3	98.4	91.1

The experimental results demonstrate that MalGuard achieves optimal effectiveness in terms of precision, recall, and F1 scores. The ablation study experimental results show that 1) Using API call graph centrality for automated feature extraction is effective. 2) Leveraging a general large language model can effectively help in filtering out irrelevant APIs from the feature set.

4.3 Explainability Evaluation

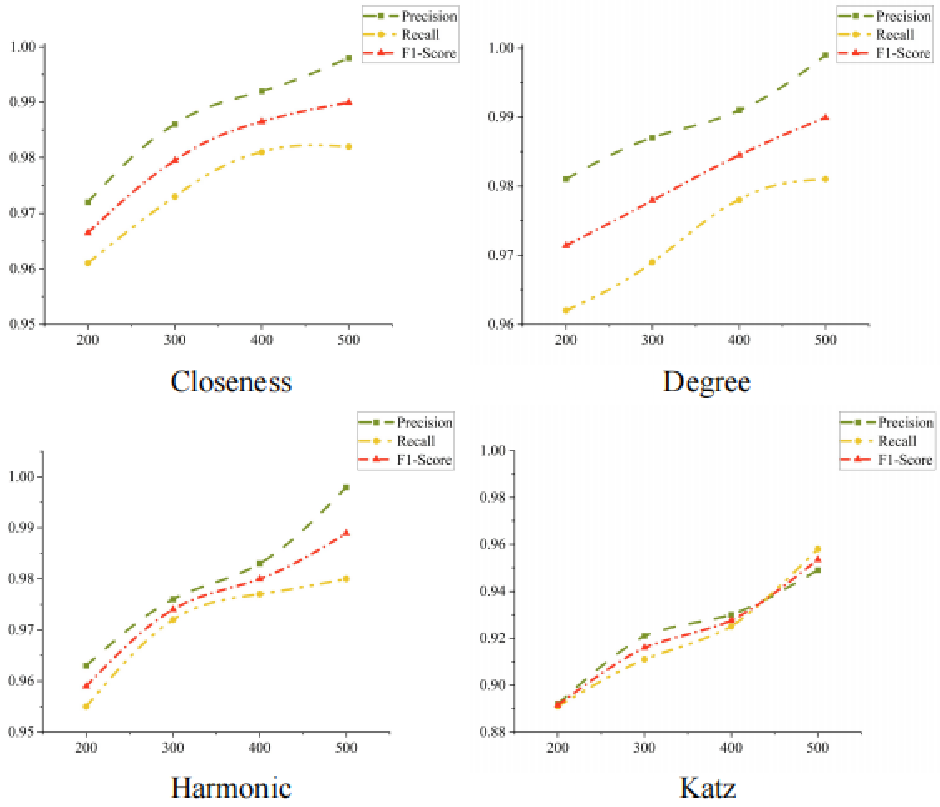
Table 10: Effectiveness of different ML Models in **Explanation Outputs Verification Dataset** (The Third Column shows the number of malicious packages that every model can detect and explain while the Fourth and Fifth Columns show the number of malicious packages that can be detected and accurately explained by more than 3 or 4 different models.).

Centrality	Model	Total detected	r>=3	r=4
Closeness	XGBoost	96	95	93
	RF	95		
	SVM	94		
	MLP	98		
Degree	XGBoost	97	96	90
	RF	96		
	SVM	91		
	MLP	97		
Katz	XGBoost	95	93	86
	RF	93		
	SVM	88		
	MLP	96		
Harmonic	XGBoost	95	92	79
	RF	93		
	SVM	82		
	MLP	95		



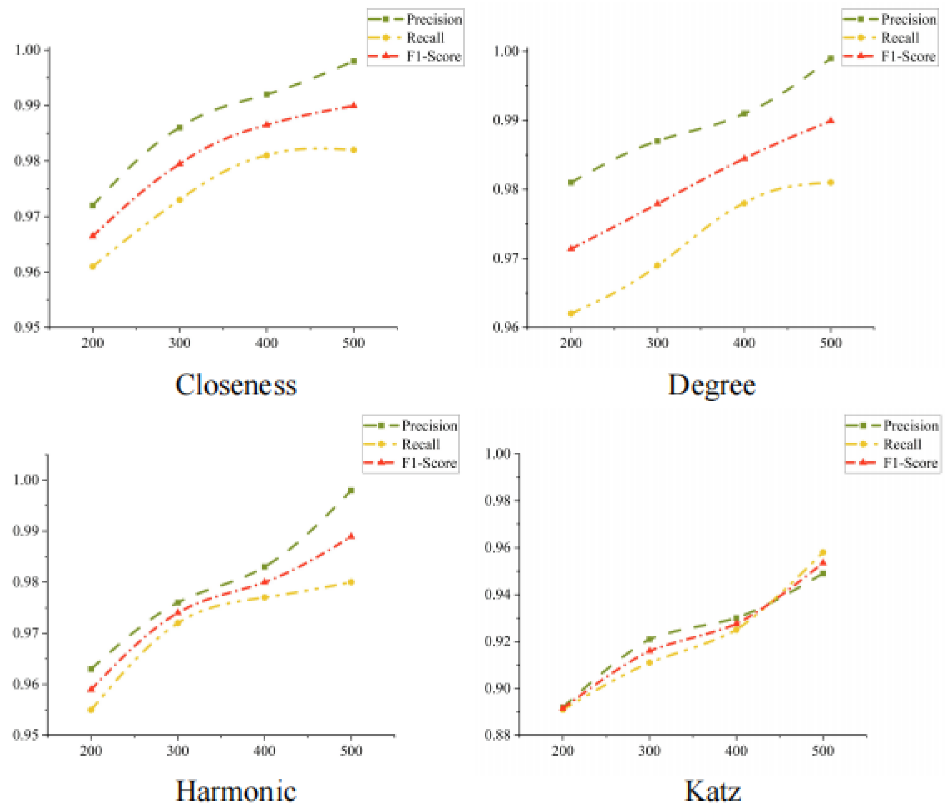
The experimental results demonstrate that the explainability content generated by MalGuard achieved an average score of 3.5 or higher, indicating that the explanation outputs are effective and useful for aiding in malicious behavior analysis.

4.3 Robustness against Adversarial Attack



The experimental results, show that as K increases, the model's effectiveness consistently improves across feature sets derived using four different centrality metrics. These findings suggest that setting K= 500 allows the feature set to capture the most comprehensive set of suspicious APIs.

4.4 Hyperparameter Sensitivity Analysis



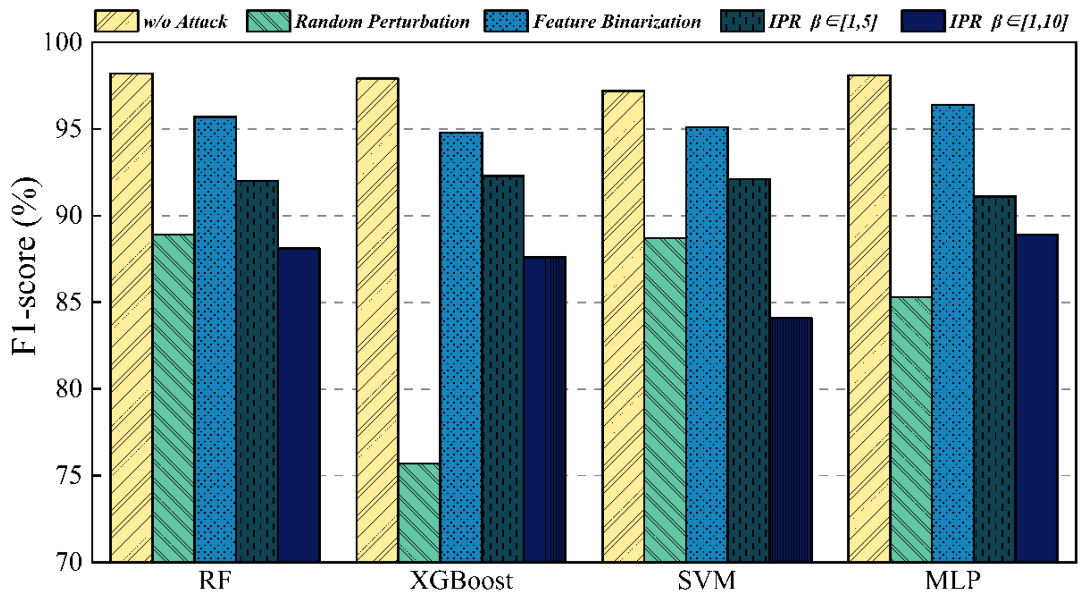
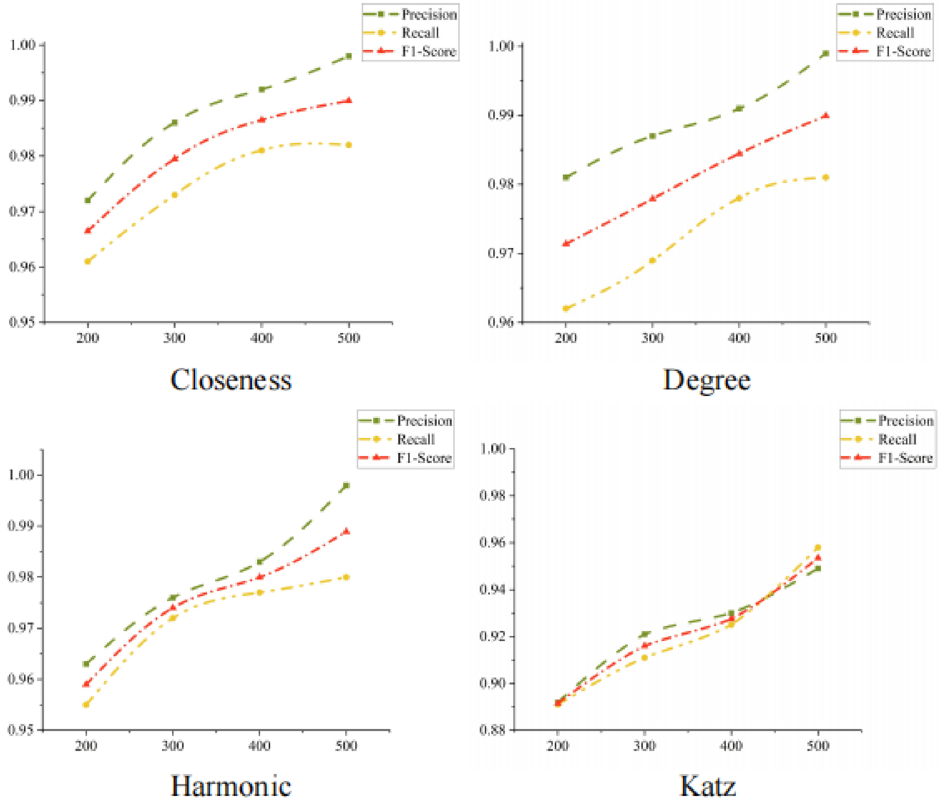


Figure 5: Model robustness against adversarial attacks.

These findings demonstrate that although MalGuard suffers some degradation under adversarial conditions, it maintains effectiveness at an acceptable level, highlighting its robustness against such attacks.

4.5 Practicality



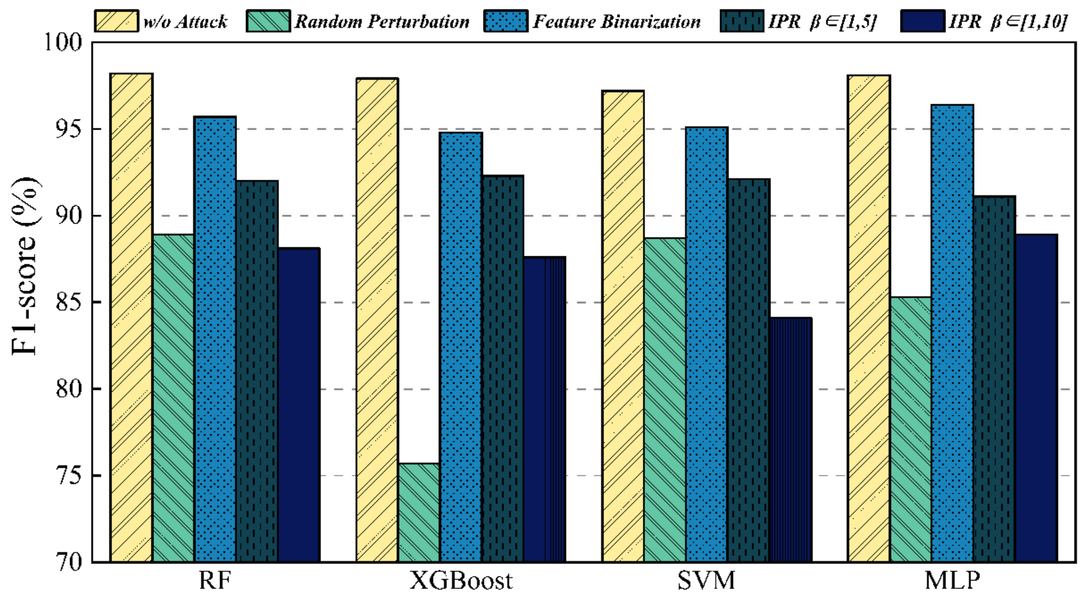


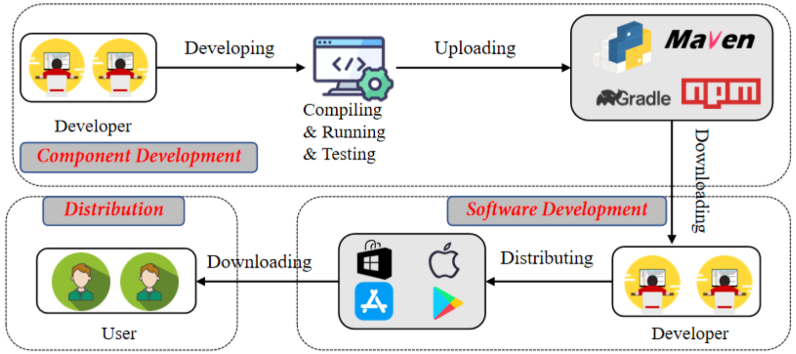
Figure 5: Model robustness against adversarial attacks.

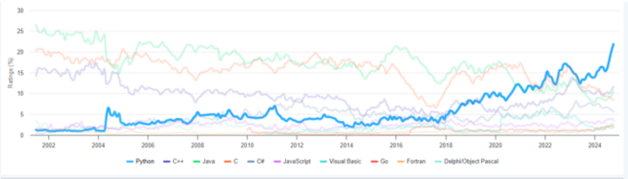
These findings demonstrate that although MalGuard suffers some degradation under adversarial conditions, it maintains effectiveness at an acceptable level, highlighting its robustness against such attacks.

In total, MalGuard discovered 144 suspicious packages. After manual review, 113 out of them were confirmed malicious. We reported these packages to the PyPI official. As of January 21, 2025, 109 of them have been removed.

5. Conclusion

1.1 Open-Source Supply Chain





2

2.1 Empirical Study

Table 2: The categories of 132 different APIs in Feature Set.

Categories	API example
File-system access	os.mkdir() os.remove shutil.copy() write() ...
Process creation	subprocess.Popen multiprocessing.Process threading.Thread ...
Network access	socket.socket() requests request.urlopen() ...
Data encode & decode	base64.b64encode() base64.b64decode() ...
Package install	install.run() pip.main() ...
System access	os.getenv() os.getcwd() ...

Table 3: Effectiveness comparison of five different ML models and LLM-based approaches on the same dataset.

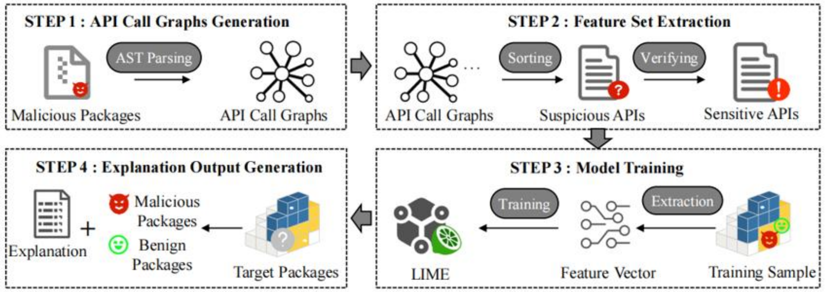
Group	Model	Precision (%)	Recall (%)	F1 score (%)	Time Consumption	
					Pre-process (s/package)	Train (s)
ML	NB	55.2	98.4	70.7	**0.8457**	0.19467
	XGBoost	98.1	98.4	98.2		4.79
	RF	98.5	98	98.2		1.0126
	SVM	89.2	94.7	91.9		0.097
	MLP	98.1	98.2	98.1		22.85157
PTM	EA4MP [37]	**99.1**	95.4	97.2	6.28	30,741.67
	CEREBRO [46]	98.6	85.7	91.7	12.489	2,439
LLM	GPT-3.5-turbo [30]	99.0	**99.3**	**99.1**	-	-

Table 5: Effectiveness comparison of different ML models and LLM-based approaches on newer samples by training an old dataset.

	XGBoost			RF			SVM			MLP			EA4MP		
Metrics (%)	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
2021&2022	88.2	80.3	84.1	97.1	82.0	88.9	88.6	80.3	84.2	95.3	80.6	87.3	94.7	90.7	92.7
2023	86.4	59.0	70.1	90.1	59.3	71.5	83.3	49.2	61.9	87.3	62.1	72.6	81.6	84.3	82.9
2024	81.5	53.4	64.5	72.6	52.1	60.7	75.4	51.0	60.8	79.6	57.1	66.5	72.7	70.5	71.6

7

3.1 Overflow of Our Approach: MalGuard



Workflow of MalGuard

An API Call Graph Centrality and LIME based Malicious PyPI packages Detection Approach:

- ☐ *API Call Graph Generation*
- ☐ *Sensitive API Extraction and Filter*
- ☐ *Malicious Package Detection*
- ☐ *Explanation Output Generation based on LIME*

9

4.2 Effectiveness Evaluation & Ablation Study

Table 8: Effectiveness comparison with the SOTA baselines.

Approach	Precision (%)	Recall (%)	F1 score (%)
VIRUSTOTAL [15]	95.2	80.6	87.3
OSSGADGET [5]	74.8	85.0	79.6
BAND4MAL [39]	84.8	96.7	90.4
EA4MP [37]	99.1	95.4	97.2
CEREBRO [46]	98.6	85.7	91.7
GUARDDOG [16]	95.6	82.6	88.6
MALGUARD	99.6	98.4	99.0

Table 7: The feature set dimension after pre-processing by general-purpose LLM. From 500 dimension

Centrality	Closeness	Degree	Katz	Harmonic
Total Dimensions	265	255	294	135

		with Feature Filtering				w/o Feature Filtering			
Metrics (%)		Closeness	Harmonic	Degree	Katz	Closeness	Harmonic	Degree	Katz
RF	Precision	99.4	92.5	99.3	99.6	99.9	99.9	99.9	94.9
	Recall	97.0	97.1	97.3	98.4	98.1	98.0	98.2	95.8
	F-1	98.2	94.8	98.3	99.0	99.0	99.0	99.1	95.3
XGBoost	Precision	99.4	99.2	92.5	99.3	99.2	99.3	99.2	93.0
	Recall	96.5	96.3	95.5	96.9	98.5	98.7	98.6	94.5
	F-1	97.9	97.7	94.0	98.1	98.8	99.0	98.9	93.7
SVM	Precision	97.9	87.1	97.6	97.9	82.8	86.6	72.6	71.8
	Recall	96.5	91.2	96.2	96.3	80.9	83.5	96.1	95.9
	F-1	97.2	89.1	96.9	97.1	81.8	85.0	82.7	82.1
MLP	Precision	98.5	92.0	98.2	98.4	99.0	99.1	98.3	89.8
	Recall	97.8	97.0	98.0	98.1	98.9	95.5	98.6	92.5
	F-1	98.1	94.4	98.1	98.2	99.0	97.3	98.4	91.1

The experimental results demonstrate that MalGuard achieves optimal effectiveness in terms of precision, recall, and F1 scores. The ablation study experimental results show that 1) Using API call graph centrality for automated feature extraction is effective. 2) Leveraging a general large language model can effectively help in filtering out irrelevant APIs from the feature set.

13

34th USENIX Security Symposium （USENIX Security 2025）

Thanks for listening！

MX120230566@stu.yzu.edu.cn

https://zenodo.org/records/15545824

揚州大學 YANGZHOU UNIVERSITY

浙江大学 ZHEJIANG UNIVERSITY

University of Southern Queensland Australia

同濟大學 TONGJI UNIVERSITY

中国工程物理研究院 CHINA ACADEMY OF ENGINEERING PHYSICS

SWINBURNE UNIVERSITY OF TECHNOLOGY