A Needle is an Outlier in a Haystack: Hunting Malicious PyPI Packages with Code Clustering

基于代码聚类的PyPI软件仓库恶意软件包检测方法

梁文韬 凌祥 吴敬征 罗天悦 武延军

The 38th IEEE/ACM International Conference on Automated Software Engineering (ASE2023)

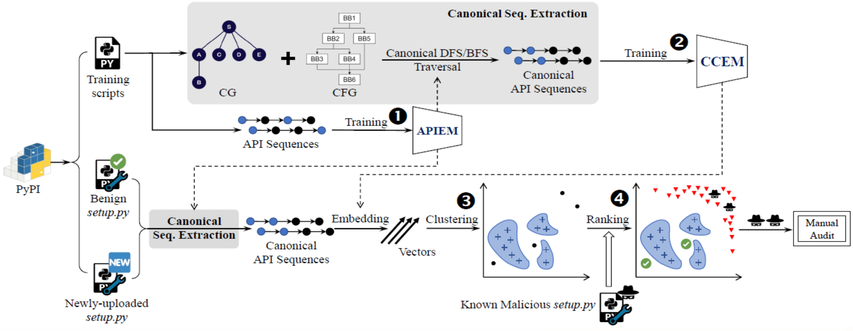
联系人：梁文韬，13146538991，liangwentao@iscas.ac.cn

Background

- The Python Package Index (PyPI), the official third-party software repository for Python, is becoming an increasingly vital resource for developers. Regrettably, the open nature of PyPI exposes end-users to substantial security risks stemming from malicious packages. Consequently, the timely and effective identification of malware within the vast number of newly-uploaded PyPI packages has emerged as a pressing concern.
- Existing detection methods are dependent on difficult-to-obtain explicit knowledge, such as taint sources, sinks, and malicious code patterns, rendering them susceptible to overlooking emergent malicious packages.

Methodology

We introduce a novel method called **MPHunter** (**M**alicious **P**ackages **H**unter) based on code clustering to detect malicious PyPI packages without requiring explicit kownledge. MPHunter is composed of four steps. First, it builds an API encoding model APIEM for canonical sequence extraction (❶). Second, it trains a code embedding model CCEM with the extracted API sequence (❷). Third, the installation scripts to be detected are encoded into vectors using CCEM, and then clustered in the vector space (❸). Finally, identified outliers are ranked to highlight the suspects for auditing (❹).



Evaluation

Table I: The Recall on Known Malicious Samples

Set No.	1	2	3	4	5	6	7	8	9	10	Total
Detected	10	10	9	9	8	9	8	10	10	9	92

- We also evaluated the recall rate of MPHunter using known malicious samples. As result, 92 of 100 malicious samples were ranked in the top ten by MPHunter (recall of 92%).
- We also conduct controlled experiments detecting malicious packages with only benign set or known malicious set, of which recall decline to 30% and 40% respectively.

Table II: The Time Cost of the Offline Stage

Offline Task	Preprocessing	Training	Embedding	Sum
Building APIEM	1,409s	320s	–	1,729s
Building CCEM	5,157s	572s	–	5,729s
Encoding Benign Samples	541s	–	1,101s	1,642s
Encoding Malicious Samples	191s	–	51s	242s
Total	**7,298s**	**892s**	**1,152s**	**9,342s**

Table III: The Time Cost of the Online Stage

Batch	#Pkgs	Preprocessing	Embedding	Merging	Clustering & Ranking	Sum
#1	4,435	178s	346s	155s	2.5s	**681.5s**
#3	1,278	218s	82s	35s	1.5s	**336.5s**
#6	533	42s	31s	14s	1.2s	**88.2s**
#7	737	48s	62s	26s	1.3s	**137.3s**
#10	974	243s	110s	47s	1.4s	**401.4s**
#13	1,392	251s	115s	45s	1.6s	**412.6s**
#16	1,334	353s	110s	49s	1.5s	**513.5s**
#17	3,226	834s	309s	131s	2.4s	**1276.4s**
#20	682	83s	59s	28s	1.5s	**171.5s**
#24	762	101s	66s	27s	1.2s	**195.2s**
#26	907	105s	93s	43s	1.5s	**242.5s**
#27	484	61s	37s	19s	1.3s	**118.3s**
#28	389	45s	39s	19s	1.3s	**104.3s**
#29	305	32s	25s	10s	1.0s	**68.0s**
#30	1,177	124s	80s	40s	1.4s	**245.4s**
Total	**18,615**	**2,718s**	**1,564s**	**688s**	**22.6s**	**4,992.6s**

- The time cost experiment shows that MPHunter is a light-weight tool which can even be deployed on a laptop.

Result

- In total, we successfully discovered 60 malicious packages in 15 batches (50%) out of the 30. All 60 suspects were confirmed to be truly malicious packages.
- All the 60 detected packages have been confirmed as real malware and removed by PyPI.
- Note that most of them (53 out of 60, 88%) rank in the top three, and 15 even rank first.
- To accurately evaluate the recall of MPHunter, we manually analyzed all installation scripts of the 15 batches and manual analysis showed that the recall rate of MPHunter for target batches is 100%
- Surprisingly, The state-of-art malware detection tool, MalOSS, cannot discover any of these malwares.

Conclusion

- We employed MPHunter to detect 30 batches of PyPI software packages, totaling 31,329 packages. From 15 of these batches, we detected 60 previously unknown malicious packages.
- The experiments also demonstrate that MPHunter's recall is perfect (100%) for detecting malicious setup.py in the target. For a detection method that does not require explicit knowledge, this result is very encouraging.
- Moreover, MPHunter has proven to be scalable, requiring only a few minutes to complete its analysis for most batches.

This paper is supported by Open Source Map