Beyond Typosquatting, An In-depth Look at Package Confusion

Shradha Neupane[1], Grant Holmes[2], Elizabeth Wyss[2], Drew Davidson[2], Lorenzo De Carli[3]

August 10, 2023

1. Worcester Polytechnic Institute
2. University of Kansas
3. University of Calgary

Package Confusion

- Presence of a package that can be confused with some other package.
- Has implications in the security of the ecosystem and applications

Example:
`Confusing package:` `mllearnlib`
`Original package:` `learnlib` and `mllearn`

Malicious Behavior: Downloads and executes 3rd party cryptominer through malicious dependency

Developers Under Attack – Leveraging Typosquatting for Crypto Mining

By Andrey Polkovnychenko and Ilya Khivrich | June 24, 2021
10 min read
SHARE:

Large-scale npm attack targets Azure developers with malicious packages

The JFrog Security Research team identified hundreds of malicious packages designed to steal PII in a large scale typosquatting attack

Sonatype Catches New PyPI Cryptomining Malware

June 21, 2021 By Ax Sharma
8 minute read time

Source: https://jfrog.com/blog/developers-under-attack-leveraging-typosquatting-for-crypto-mining/

Typosquatting and Confusion

- People have intuitive notion of how package confusion occurs, which is usually limited to typos [1].
- Limited understanding on how package confusion beyond typos

Goal:

Does package confusion beyond typo or lexical confusion exist, and can we detect it algorithmically?

Impact of Package Confusion

- Intentional confusion: Add maliciousness to the package uploaded that adversely affects the developer or application users
- Unintentional confusion: May degrade quality of projects introducing potentially unmaintained, vulnerable code to projects [2, 3]

[1] Matthew Taylor, Ruturaj Vaidya, Drew Davidson, Lorenzo De Carli, and Vaibhav Rastogi. Defending Against Package Typosquatting. In NSS, 2020
[2] Elizabeth Wyss, Lorenzo De Carli, and Drew Davidson. What the fork?: Finding hidden code clones in npm. In IEEE/ACM ICSE, 2022.
[3] Markus Zimmermann, Cristian-Alexandru Staicu, and Michael Pradel. Small World with High Risks: A Study of Security Threats in the npm Ecosystem. In USENIX Security, 2019.

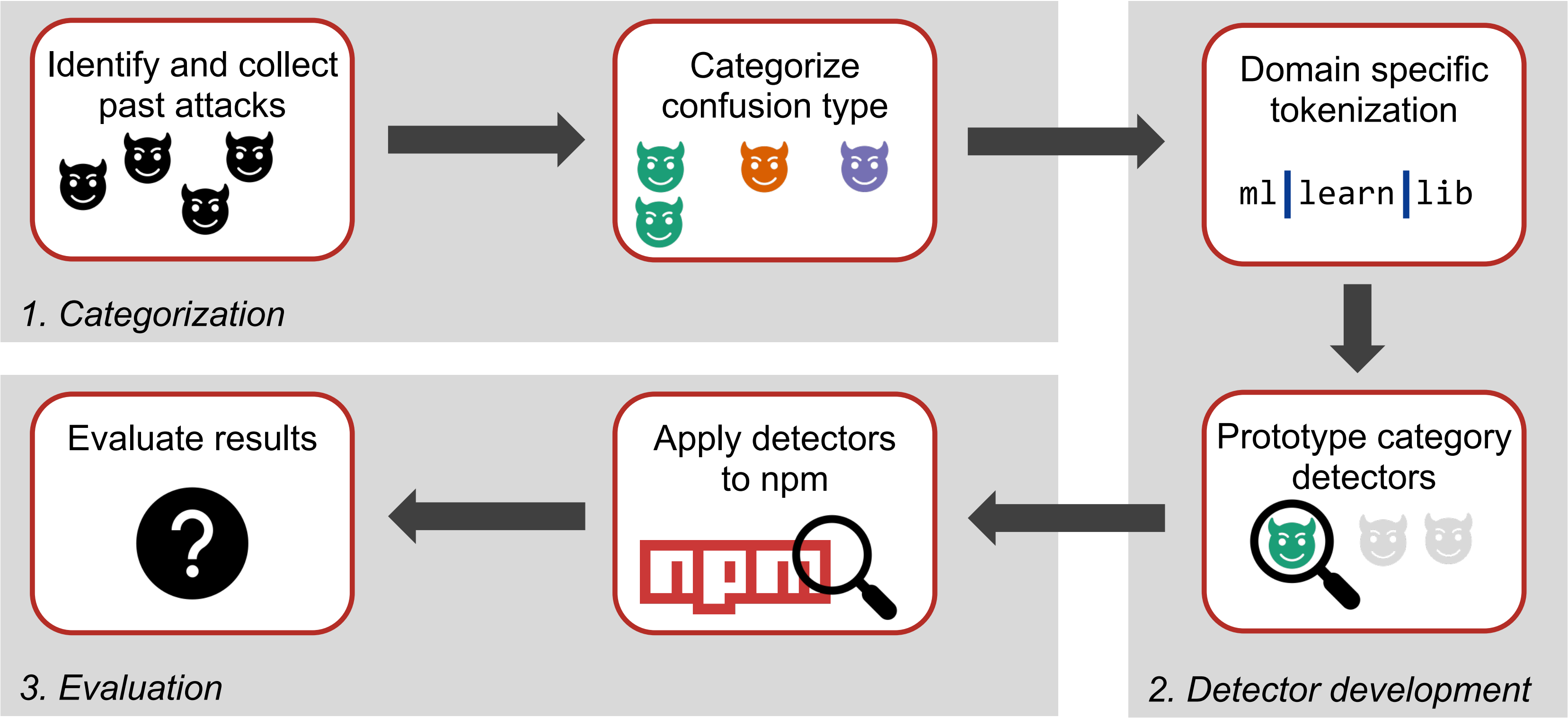
CONTRIBUTION

1. Package confusion occurs beyond typo squatting – we consider 13 categories of confusability

2. Find potentially confusing packages in the wild and evaluate effectiveness of detection rules

3. Evaluate the security impact of package confusion

Research Outline



Collecting Attacks Results

Results of Collecting Historical Data

1232

Distinct attacks / confusing packages uploaded

7

Campaigns with 10 or more packages uploaded

Distribution Across Ecosystems

723 npm

462 python Package Index

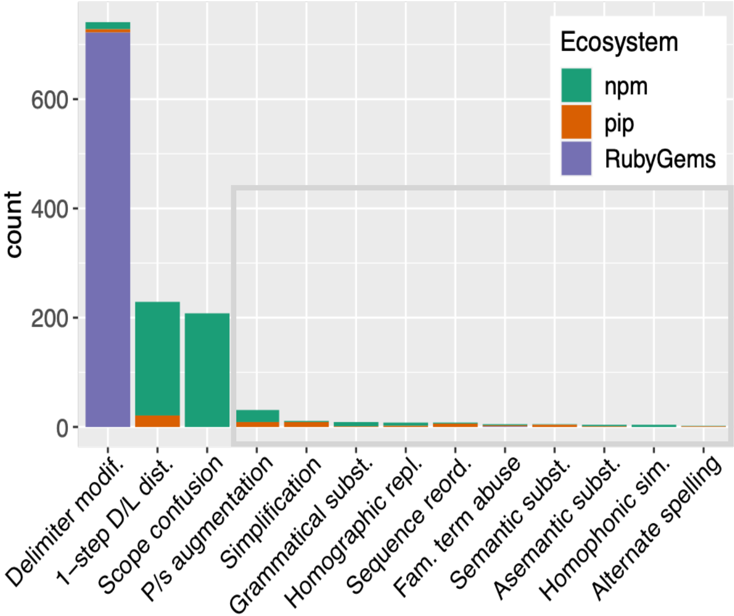
48 RubyGems

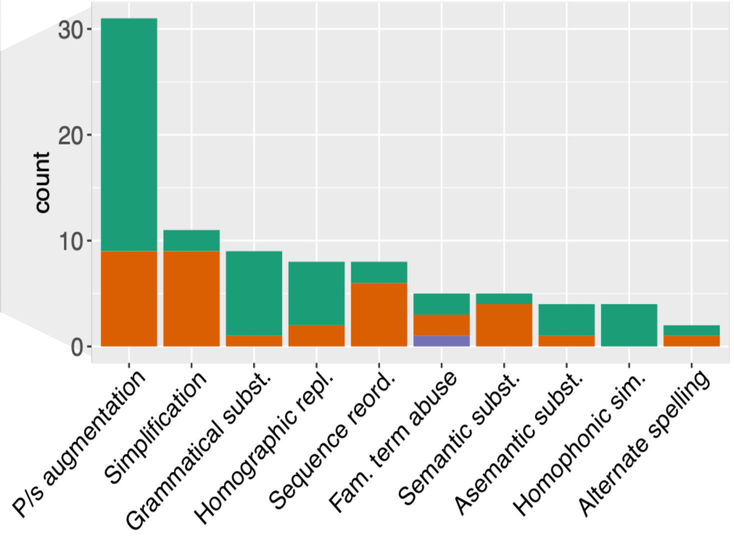
Confusability models and categorization

Thematic Analysis

After Round 4
1-step D/L distance
Alternate spelling
Asemantic substitution
Delimiter modification
Familiar term abuse
Grammatical substitution
Homographic replacement
Prefix/Suffix augmentation
Scope confusion
Semantic substitution
Sequence reordering
Simplification
Homophonic similarity

$\alpha = 0.96\ (0.94, 0.99)$





Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2):77–101, 2006.

Processing package names: Delimiter-less Tokenization

- Number of detectors need transformation of package name into sequence of tokens
- Package names consist of technical jargons, which do not have valid English words but assume valid connotation in technical language. (json, db, py, js etc)
- Built a delimiter-less tokenization algorithm using the npm package names.

Example:

Confuser package: `mllearnlib`
Breaking down the package into tokens: [**ml, learn, lib**]

Establised package: `mllearn`
Breaking down the package into tokens: [**ml, learn**]

Detection rules: Prefix/Suffix Augmentation as there is an addition of "`lib`" in the confuser package.

Performance of Detection Rules

Rule	Precision	Recall	F1
P/S augmentation	0.95	0.70	0.81
Sequence reordering	0.88	0.88	0.88
Delimiter modification	1	0.97	0.98
Grammatical subst.	0.88	0.88	0.88
Scope confusion	1	0.90	0.95
Semantic subst.	1	0.4	0.57
Asemantic subst.	0.75	0.75	0.75
Homophonic sim.	0.07	0.75	0.13
Simplification	0.58	0.64	0.61
Alternate spelling	1	1	1
Homographic repl.	0.5	0.88	0.64

Table: Performance of detection rules

Detection Rule Optimization

Created Initial prototype and optimized it on each round

Goal: Maximize the chances of identifying actually confusable packages, at the cost of missing some attacks.

Accounted for significantly imbalanced samples.

EVALUATION

RQ1: How many potential instances of package confusion exist in the npm ecosystem?

Methodology
Apply the detection rules to npm
Focus on: (popular, unpopular) package pairs

Popularity threshold
15,000 weekly downloads

Popular package: Established Original Packages

Unpopular packages: Confuser Packages

Total:
1, 727, 553 × 24871
Reduced analysis space from all $(1.7e6)^2$ npm package pairs

Results

Rule	#Instance
P/S augmentation	143864
Asemantic subst.	139160
Simplification	27743
Homophonic sim.	24735
Semantic subst.	9610
Delimiter modification	7183
Scope confusion	4247
Grammatical subst.	2461
Homographic repl.	2393
Sequence reord.	1734
Alternate spelling	21

Table: Matches in npm for each category

Results

- **~ 360,000** package pairs detected as confusing
- Analysis took **0.22ms/pair**
- **2799** pairs matching multiple categories
- *Homophonic similarity & Prefix/ suffix augmentation*, *Delimiter modification & Sequence reordering*, and *Delimiter modification & Grammatical substitution*

RQ2: How confusing are the identified matches?

Online survey of to perceive confusability of randomly selected package pairs.

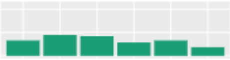
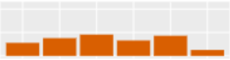
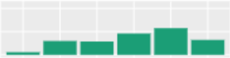
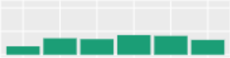
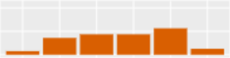
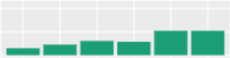
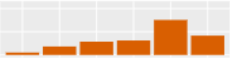
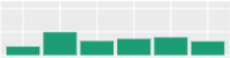
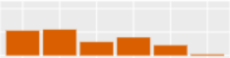
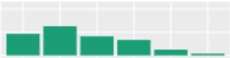
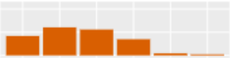
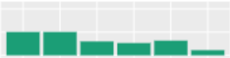
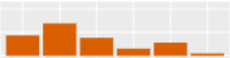
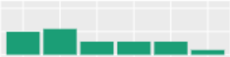
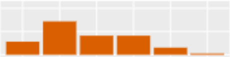
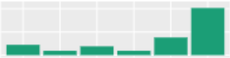
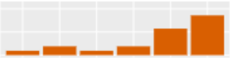
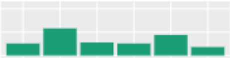
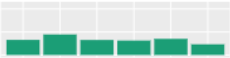
On a scale of 1 to 6, how likely are you to misremember or mistype the package in column V with package column P?

Sampling: 50 questions from a pool of 100 package pairs from each category + 100 control samples

Recruitment: Email recruiting and snowball sampling of student developers (Number of recruits: **64**)

Goal: Determine which rules can return reliable matches.

Results

Rule	Rating Distribution	Median Distribution	n samples	%(2+r≥ 4)	%(3r≥ 5)
P/s augmentation			79	44%	2.5%
Sequence reord.			58	79%	10%
Delimiter modif.			78	56%	7.7%
Grammatical subst.			77	74%	18%
Scope confusion			84	52%	4.8%
Semantic subst.			83	31%	0.0%
Asemantic subst.			86	21%	0.0%
Homophonic sim.			78	24%	3.8%
Simplification			78	29%	1.3%
Alternate spelling			21	81%	38%
Homographic repl.			62	39%	6.5%
Overall			**62**	**45%**	**6.1%**

Results

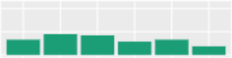
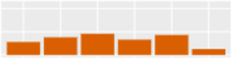
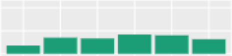
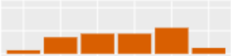
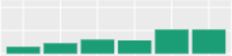
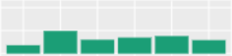
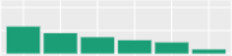
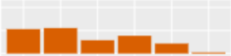
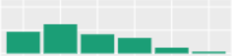
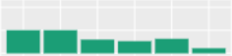
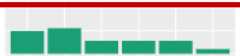
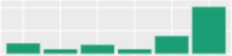
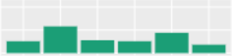
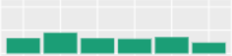
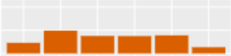
>10% with "highly confusing" criterion

>70% with "potentially confusing" criterion

Rule	Rating Distribution	Median Distribution	n samples	%(2+r≥ 4)	%(3r≥ 5)
P/s augmentation			79	44%	2.5%
Sequence reord.			58	79%	10%
Delimiter modif.			78	56%	7.7%
Grammatical subst.			77	74%	18%
Scope confusion			84	52%	4.8%
Semantic subst.			83	31%	0.0%
Asemantic subst.			86	21%	0.0%
Homophonic sim.			78	24%	3.8%
Simplification			78	29%	1.3%
Alternate spelling			21	81%	38%
Homographic repl.			62	39%	6.5%
Overall			**62**	**45%**	**6.1%**

Results

< 25% with "potentially confusing" criterion

Rule	Rating Distribution	Median Distribution	n samples	%(2+r≥ 4)	%(3r≥ 5)
P/s augmentation			79	44%	2.5%
Sequence reord.			58	79%	10%
Delimiter modif.			78	56%	7.7%
Grammatical subst.			77	74%	18%
Scope confusion			84	52%	4.8%
Semantic subst.			83	31%	0.0%
Asemantic subst.			86	21%	0.0%
Homophonic sim.			78	24%	3.8%
Simplification			78	29%	1.3%
Alternate spelling			21	81%	38%
Homographic repl.			62	39%	6.5%
Overall			**62**	**45%**	**6.1%**

RQ3: What is the security impact of identified confusing packages?

Goal:
Assess density of malicious packages amongst detected confusing packages

Problem:
No ground truth.

Solution:
Analysis of existing vulnerability database (lower bound)

Results:
Packages flagged by our rules are **3 times more** likely to be malicious than control.

Details:
Sample: Unique packages = 210,741, Malicious packages found: 168 (0.079%)
Control: Unique packages = 150,000, Malicious packages found: 39 (0.026%)

Malicious Behavior in Confusing Packages

Attack Category	#pkgs
Stealing	70
Backdoor	9
Sabotage	2
Cryptojacking	2
Virus	1
Maladvertising	2
PoC	1
Cryptotheft	33
Downloader	1
Confusion	2
Unknown	45

Table: Distribution of confusing packages according to malicious behavior

- We categorized the malicious behaviors in the flagged packages as per [1] Duan et al.
- Added 3 new categories: Crypto Theft, Downloader, Confusion
- Could not be verify malicious behavior in some due to removal of packages from ecosystems

[1] Ruian Duan, Omar Alrawi, Ranjita Pai Kasturi, Ryan Elder, Brendan Saltaformaggio, and Wenke Lee. Towards measuring supply chain attacks on package managers for interpreted languages. In IS NDSS, 2021.

Conclusions

- **Package confusion is a credible threat and our categorization helps to specify how the attacks may occur.**

- **Our categories provide a new dimension to package confusion beyond typosquatting**

- **Some detection rules may benefit from refinement, some may be usable as warning mechanism as is**

Thank You!

Shradha Neupane
sneupane@wpi.edu

Other Collaborators
Prof. Lorenzo De Carli - lorenzo.decarli@ucalgary.ca
Prof. Drew Davidson - drewdavidson@ku.edu
Elizabeth Wyss - elizabethwyss@ku.edu
Grant Holmes - g.holmes429@gmail.com

This work was supported in part by a gift from Google and the funding from the WPI CS department.