Appendix of RepSEO

I. Feature Engineering

To detect abusive packages on software repositories, we implemented a classification method based on the motivation example shown in Figure 1. This method includes features from five aspects: structure, semantics, links, metadata, and historical behavior. The full features are listed in Table I.

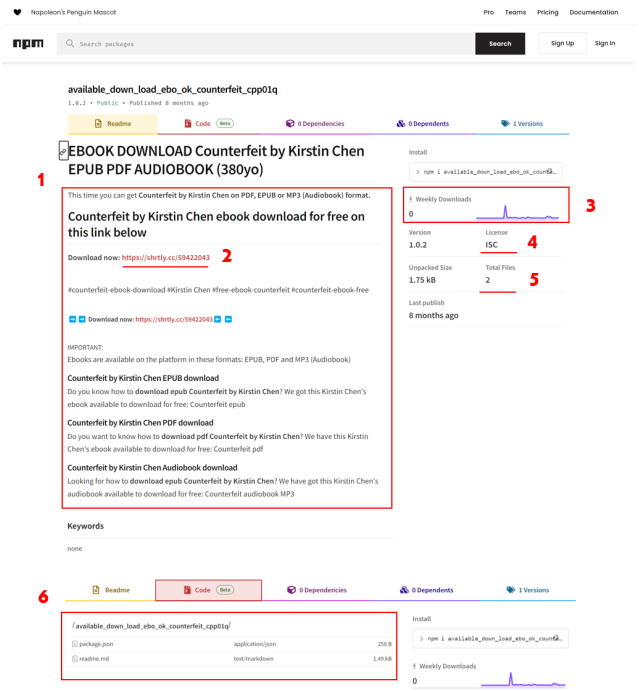

Figure 1: Motivation Example. 1) project introduction; 2) promotion link; 3) download count; 4) license; 5) total files; 6) files and directories.

We acknowledge that individual features might be bypassed. However, we intentionally designed our detection system to integrate 16 features from 5 perspectives, rather than a singular rule-based approach, to provide a more comprehensive and robust detection of RepSEO packages. More importantly, security research still leads to an iterative development between the detection technique and evasion tricks. The detection features can eventually be bypassed by the attackers. However, this feature selection strategy significantly increases the difficulty for attackers to bypass our system as a whole, thereby enhancing the cost and effort required on their part.

Structure features. Adhering to de facto standards in software engineering, developers typically organize their code into distinct folders and files, such as *src*, *frontend*, and *backend*, to enhance maintainability. Additionally, they often include detailed instructions, sometimes accompanied by code snippets, to assist other developers and users in further developing or integrating their packages. In contrast, blackhat SEO packages focus primarily on content that will be indexed by search engines. They typically lack the descriptive files that instruct users on how to integrate such packages, deviating significantly from conventional software development practices.

Table I: Feature set of the abusive package detection, which contains 16 features spanning five different categories.

Type	Feature	Data Type	Length
Structure	# of directories	int	1
	Presence of introduction	boolean	1
	Usage of HTML formatting	boolean	1
	Presence of code blocks	boolean	1
Semantic	Platform semantic distances	float	10
Link	Ratio of internal links	float	1
	Domain diversity of external links	int	1
	Ratio of short links	float	1
	Avg. rank of external domains	float	1
	Duplication of links	int	1
Metadata	Copyright license	boolean	1
	Official package	boolean	1
	Repository URL	boolean	1
	Homepage URL	boolean	1
	Domain rank of homepage URL	float	1
	# of Download	int	1
Historical	User historical behavior	float	25

Thus, we propose the following four features:

- *Number of directories*, which represents the complexity of the package in terms of directories.
- *Presence of project introduction*. SEO attackers mainly use project introduction to present their promotional content.
- *Usage of HTML formatting*. Traditional blackhat SEO techniques have primarily targeted webpages, while the malicious practices migrate to project introduction, remnants of HTML syntax may still be present in the content.
- *Presence of code blocks in the introduction*. Programmers tend to add code blocks providing detailed usage instructions, while the SEO packages not.

Semantic features. Due to the utilization of package introductions for promoting links and the inclusion of extensive text describing the purpose of these SEO packages, there is a significant difference in the overall semantic content between these packages and other regular packages on the platform. In contrast, other packages typically focus on describing their functionality and usage. Based on this insight, we use the

semantic distance between the platform and the package as its semantic features. If reducing the semantic distance to the platform, the promotional content of attackers may be drowned out by a large amount of normal content, making the true purpose more difficult for users to perceive.

The key step of our approach is to automatically build a semantic profile for each platform and each package, which is represented as a set of terms, then search for the inconsistency between them. For platform semantic representation, we automatically identify the terms from three different sources:

• *Wikipedia*: the Wikipedia pages for package management platforms provide a comprehensive summary of different platforms. For example, the Wikipedia page of npm tells its language ("JavaScript"), usage ("download and install specific modules"), command ("npm install"), etc. We ran a crawler to collect the wiki pages for three platforms.

• *Search result*: the search results for package management platforms provide the most relevant webpages to them, such as highly ranked packages and introductions to package management platforms. Therefore, we collect the search result pages of the three platforms on Bing and Google, and gather the content from the top 200 search results to construct the platform semantic profile.

• *Representative packages*: the most representative packages in software repositories are typically those that best align with the expectations of the platform and developers. The package introductions of these packages convey their characteristics and usage, making them more favored by developers. For our study, we select the top 30 packages on the platform and extract their package names, abstracts, and package introductions as the semantic information for the platform.

For package semantic representation, we extract their package names, abstracts, and package introductions as the semantic information for the package.

The text we collect in packages is multilingual, including Russian, French, Thai, Japanese, *etc.* We utilize Google Translate to convert non-English text into English. We utilized WordNinja [2] for word segmentation without spaces, and then removed stop words and special characters. Subsequently, part-of-speech tagging was performed, and some representative semantic features were selected, including the highest-frequency nouns and verbs. For both platform semantics and package semantics, we selected the 20 and 10 highest frequency terms, respectively. To compare the semantic distance, we then utilized a word-embedding tool, a pre-trained Word2Vec model [1], to convert the terms into vectors. Given the vectors of a term in the package and a platform keyword, we measure their semantic distance by calculating the cosine distance between the vectors. For each term in the package, we calculate its average distance to all the keywords of platform semantics. Finally, we compose the 10 distances of the 10 terms in the package as semantic features.

Link features. In software repositories, it is common for packages to utilize hyperlink techniques to incorporate various external resources, such as image rendering, video embedding, and online developing documents. Blackhat SEO packages also extensively use hyperlinks, but with a divergent intent. Besides referring to remote media resources, a critical component of their blackhat SEO strategy involves embedding target promotional links directly within their homepage. Interestingly, our preliminary research has revealed a significant number of blackhat SEO packages using shortened links for promotion, a practice rarely seen in legitimate packages.

Here we donate the number of links is the number of all external URLs except static resource. The external links refer to the links that reside beside the software repositories, i.e., GitHub, Travis, and Bitbucket. We also define the ranking of the domain as $rank\_score = 1 - rank/1000000$, here the rank is the rank of the domain's SLD [3]. We identify the shared short links according to a predefined list of short link domains [4].

Thus, we propose the following five features.

- *Ratio of internal links.* Genuine packages commonly use internal links to guide users to the projects' homepage, documentation, and demo pages. For attackers, adding internal links may make it difficult for victims to find the link that the attacker truly wants to promote.
- *Domain diversity of external links.* Generally, developers tend to leverage a limited number of domains to host their external resources, even outside the popular software repositories. For attackers, reducing domain diversity can cause attackers to only promote very limited links at once. Moreover, we found that attackers will include attractive images to entice victims, which also increases domain diversity. We define the diversity of domain as $1 - 1/n$, where n is the number of external domains.
- *Ratio of short links.* Legitimate packages often link to remote resources or documents with a stable duration. In contrast, most short links tend to expire within a short period, typically only a few days or weeks. If attackers want to bypass by using landing page links directly, it will expose the promotional links and make promotional link more easily identifiable and trackable by platforms, or raise victims' alertness.
- *Average ranking of external domains.* Genuine packages typically utilize reputable services such as documentation, customer support, donations, or online chatting, which usually have higher domain rankings. On the other hand, links promoted by blackhat SEO often originate from less popular domains that require promotion. For bypassing, raising the average ranking needs adding more high-rank links, which makes it difficult for users to find the link that the attacker truly wants to promote.
- *Duplication of links*. While genuine packages provide a variety of resources and thus link to different types of services, abusive packages may repetitively embed promotional links to draw the attention of search engines and potential victims.

Metadata features. To facilitate a more user-friendly approach to locating ideal packages, software repositories man-

date that each package declares its license or explicitly labels its corresponding repository or homepage URL. Furthermore, these repositories prominently indicate whether a repository is official. Another key differentiator lies in the download patterns: numerous users often clone or download repositories to build their own projects, a practice not typically associated with blackhat SEO packages. As a result, there is a significant variance in the download count between genuine and malicious packages. Consequently, we consider the following six features in our analysis:

- *Copyright license.* The presence of a copyright license in a package signifies defined usage terms and conditions, aiding users in selecting packages that meet their specific needs. For attackers, adding licenses will increase the chances of being noticed by programmers, leading to reports and potential removal from the software repository.
- *Official package*. packages recognized as "official" often indicate validation by the software repository's support team, showing a higher level of software quality and security. RepSEO packages are unlikely to be published under an official account. Should an attacker somehow gain official certification or impersonate an official account, it would attract heightened scrutiny.
- *Repository URL.* This serves as a direct link to the package's source code, enabling code review, collaboration, and contributions from the community.
- *Homepage URL.* A dedicated homepage for a package provides additional information, such as documentation, updates, and community support. Like the repository URL, a homepage is indicative of a more comprehensive and quality project.
- *Domain rank score of homepage URL and repository URL.* The reputation of a package repository can also be gauged by the domain rank score of its homepage and repository URLs, reflecting their credibility and reputation. The Repository URL and Homepage URL fields shown in the homepage are highly visible to search engines and users. Filling them with high-ranking links would confuse potential victims' attention and fail to mislead them.
- *Download Count*. This metric indicates the popularity and adoption level of a package. A higher download count typically reflects widespread use, with many users cloning or downloading the repository for their own projects. To bypass by inflating download counts, attackers expend significant effort, thereby increasing their costs. However, higher download numbers also increase the chances of being noticed by programmers/regulators, leading to reports and potential removal from the software repository.

Historical behavior features. Given the associated registration costs, normal developer accounts on package management platforms are less likely to engage in abusive practices. Conversely, an attacker may upload multiple abusive packages using a single account. Therefore, analyzing a user's historical behavior can serve as evidence of their abusive actions. To incorporate historical behavior into our classifier, we consider the feature vectors of the two packages preceding the current package, belonging to the same user. These feature vectors encompass structural features, semantic features, link-based features, and metadata features. Specifically, we calculate the average feature vector of the most recent package and the second most recent package. This average feature vector is then appended to the feature vector of the current package, forming the historical behavior feature vector.

$$V_n^H = \sum_{i=n-j}^{n-1} \alpha_{n-i} V_i^M \quad \text{where} \quad \sum_{i=1}^{j} \alpha_i = 1 \tag{1}$$

For attackers, maintaining a good historical behavior record will be limited to releasing only one RepSEO package per account and using a higher number of legitimate packages as cover. This significantly increases the demand for accounts, which is unacceptable for attackers who rely on releasing thousands of packages using a single account.

References

[1] Fasttext english word vectors. URL https://dl.fbaipublicfiles.com/fasttext/vectors-english/wiki-news-300d-1M.vec.zip.

[2] keredson. wordninja, 2023. https://github.com/keredson/wordninja.

[3] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the 24th Network and Distributed System Security Symposium (NDSS)*, 2019.

[4] Alexander Neumann, Johannes Barnickel, and Ulrike Meyer. Security and privacy implications of url shortening services, 2011. https://www.ieee-security.org/TC/W2SP/2011/papers/w2sp-urlShortening-slides.pdf.