Exposing the Hidden Layer: Software Repositories in the Service of SEO Manipulation

Mengying Wu[†*], Geng Hong[†*], Wuyuao Mai[†], Xinyi Wu[†], Lei Zhang[†], Yingyuan Pu[‡], Huajun Chai[‡], Lingyun Ying[‡], Haixin Duan[§¶] and Min Yang[†]
[†]Fudan University, China, wumy21@m.fudan.edu.cn, {ghong, maiwuyuao20, xinyiwu20, zxl, m_yang}@fudan.edu.cn
[‡]QI-ANXIN Technology Research Institute, China, {puyingyuan, chaihuajun, yinglingyun}@qianxin.com
[§]Tsinghua University, China, duanhx@tsinghua.edu.cn
[¶]Quancheng Laboratory, China

***Abstract*—Distinct from traditional malicious packages, this paper uncovers a novel attack vector named "blackhat Search Engine Optimization through REPositories (RepSEO)". In this approach, attackers carefully craft packages to manipulate search engine results, exploiting the credibility of software repositories to promote illicit websites.**

Our research presents a systematic analysis of the underground ecosystem of RepSEO, identifying key players such as account providers, advertisers, and publishers. We developed an effective detection tool, applied to a ten-year large-scale dataset of npm, Docker Hub, and NuGet software repositories. This investigation led to the startling discovery of 3,801,682 abusive packages, highlighting the widespread nature of this attack. Our study also delves into the supply chain tactics of these attacks, revealing strategies like the use of self-hosted email services for account registration, redirection methods to obscure landing pages, and rapid deployment techniques by aggressive attackers. Additionally, we explore the profit motives behind these attacks, identifying two primary types of advertisers: survey-based advertisers and malware distribution advertisers. We reported npm, NuGet, and Docker Hub about the RepSEO packages and the related supply chain vulnerabilities of Google, and received their acknowledgments. Software repositories have started removing the abusive packages as of this paper's submission. We also open-source our code and data to facilitate future research.

***Index Terms*—software repository, blackhat SEO, supply chain vulnerability.**

I. Introduction

Software repositories such as npm [1], NuGet [2], and Docker Hub [3] have gained immense popularity among developers worldwide. They provide developers with a convenient way to share, publish, and install software packages and libraries. Through these software repositories, developers can easily access millions of open-source projects and third-party components, accelerating the software development process and promoting code reuse. With the widely used software repositories, researchers have demonstrated severe security vulnerabilities, such as malicious packages [4, 5], dependency confusion [6, 7], and compromised name attacks [8, 9].

Blackhat SEO through the software repositories. While prior research has uncovered security issues in software repositories, it predominantly centers on the malicious packages polluting victims through *malicious code behavior*, *i.e.,* injecting malicious code into the downstream projects. In this paper, we reveal a novel attack: blackhat **S**earch **E**ngine **O**ptimization through the **REP**ositories (*RepSEO*). Blackhat SEO involves aggressive tactics violating search engine guidelines to manipulate algorithms and unfairly boost website rankings in search results. Since search engines can quickly label hacked SEO websites [10], attackers target software repositories. These software repositories are favored by search engines for their high-quality content, leading to higher domain ranks and increased visibility for the embedded links. Meanwhile, the open-source community lacks sufficient and effective auditing measures that entice attackers to publish SEO content on software repositories, compared to well-known user-generated content platforms. For example, 92% of Twitter spam accounts were suspended within three days [11].

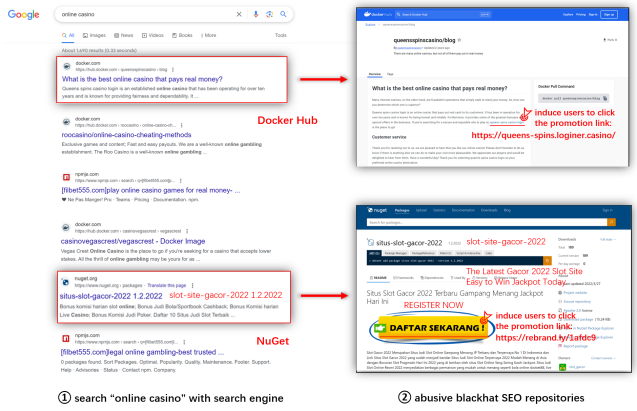

Fig. 1. Motivation example. A search of "online casino" will lead to webpages with high page ranking from software repositories such as Docker Hub and NuGet, which contain introductions and links to the casino websites.

As shown in Fig. 1, a Google search query for "online casino" might return results featuring recommendations from Docker Hub, showcasing items such as the "best online casino that pays real money". Upon clicking this link, users are directed to a page with a Docker Hub project homepage. An illusive and conspicuous call-to-action button is present on the homepage, urging visitors to click on and then redirect to the casino website. This illustrates these software repositories have been manipulated to facilitate the RepSEO attack.

*These authors contributed equally to this work.

In RepSEO, attackers exploit the open-source community's reputation by crafting and publishing enormous packages with promotional content. [1]

Breakdown of RepSEO. To the best of our knowledge, this paper represents the first systematic study to uncover the underground ecosystem of blackhat SEO through software repositories. Our research comprehensively demonstrates how attackers have successfully published abusive packages on these software repositories and identify the various parties involved in the RepSEO supply chain. We have identified three critical participants in the creation of these abusive packages: the account provider, the advertisers, and the publishers (Section II-A). To thoroughly evaluate the severity of these attacks, we created a detection tool with features across five aspects to identify RepSEO packages, with an average precision and recall of 98.27% and 98.24%.

Our dataset, encompassing 17,087,643 packages from 2011/01/07 to 2024/03/31, revealed 3,801,682 RepSEO packages, over 22.25% of the total. Docker Hub had the most, 2,725,573, followed by npm with 929,614, and NuGet with 146,495 (Section III). The longitudinal study observed that software repositories have initiated responses to this abuse, albeit after a significant delay. For instance, npm began to notice the upsurge in abusive package uploads four months after the initial surge, implementing remedial actions in July 2023. However, there were still 2,812,618 RepSEO packages hosted on these software repositories before we reported.

Supply chain of RepSEO. In the RepSEO supply chain (Section IV), various parties exhibit unique characteristics. *Account Providers* create numerous email addresses using self-hosted services for registering repository accounts, aiding RepSEO execution. *Advertisers* employ redirection techniques to easily configure and conceal promotion links. Interestingly, beyond public URL shortening services, we found a new "underground shorten links service" for servicing landing pages. Besides, attackers also exploited Google's redirect service to bypass spam detection. *Publishers* can rapidly activate tens of thousands of accounts on software repositories; for example, a publisher activated 62,190 accounts in only five days.

Profit logic. Our research further explores the profit mechanisms behind RepSEO (Section V). Our detailed case study reveals traditional cybercrimes are using the new attack vector. We have identified two main types of advertisers: survey-based aggressive advertisers and malware distribution advertisers. The former use online surveys to lure victims into revealing personal information, then entangle them in different scams. The latter advertisers spread Trojan malware, often linked with regularly updated Command and Control (C&C) servers.

Mitigations and disclosure. To assist open-source community in combating RepSEO, we offer more than just an analysis of the abuse. We suggest mitigation strategies for different roles and supply chains, including examining accounts with non-reputable email providers, monitoring accounts that upload packages quickly, and using detection methods for RepSEO packages. We have reported our findings and supply chain vulnerabilities to npm, NuGet, Docker Hub, and Google. These entities have acknowledged our reports and begun removing the identified RepSEO packages and links (Section VI).

[1] "Package" in this paper includes npm package, NuGet package, and Docker image.

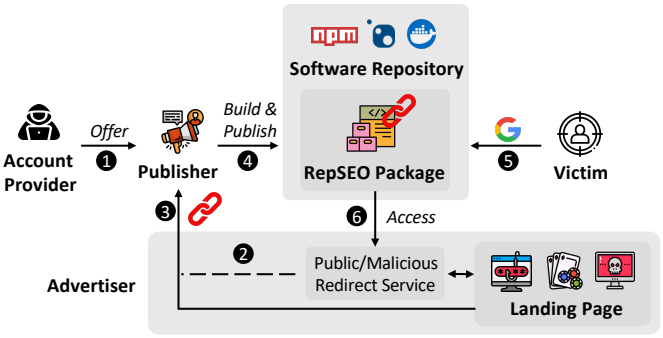

Fig. 2. Ecosystem of RepSEO. Advertisers may use redirect services to wrap landing pages. Publishers build and publish abusive packages involving promotion links, attracting victims to visit the homepage and access the landing pages.

Contributions. This paper makes the following contributions:

- This is the first systematic study to identify and report on a novel attack vector, termed "blackhat Search Engine Optimization through REPositories (RepSEO)", in which attackers carefully craft packages to manipulate search engine results.
- We developed a detection tool and made a comprehensive measurement study over a ten-year dataset, uncovering 3,801,682 RepSEO packages, representing 22.25% of the total packages.
- Our study provides detailed insights into the supply chain tactics of the attacks, uncovering strategies used by account providers, advertisers, and publishers.
- We also examine the profit-making strategies of the attackers, identifying two predominant types: survey-based advertisers and malware distribution advertisers.
- We release our code and data at [12] to help future research.

II. Methodology

In this section, we initially delineate the key roles involved in RepSEO and illustrate its operational workflows. Subsequently, we describe the detection tool and supply chain analysis method.

A. Overview

In this study, we delve into attackers' operational pipeline for promoting illicit operations, such as unlicensed online pharmacies and online survey scams, through open-source software repositories like npm and NuGet.

Threat model. We assume attackers can easily create or acquire numerous accounts within software repositories without breaching network security. They abuse the software repositories to publish packages embedded with promotional content. The attackers can tailor this content, altering both the descriptive information and the code itself.

Supply chain. We pinpointed three key participants in the construction of these packages:

• *Account Provider:* Software repositories require the users to log in before uploading packages. Account providers register numerous accounts by exploiting legitimate email services or setting up dedicated email systems, facilitating attackers in publishing abusive packages.

• *Advertiser:* They generate promotion links, which are designed to entice users and lead to potential security or privacy threats, *e.g.,* ransomware attacks or data breaches. Then, they handle the links to the publishers.

• *Publisher:* Publishers disseminate packages containing promotion links from advertisers, using accounts provided by account providers. They create deceptive packages to trick search engines and entice users into clicking on these links.

Workflows. The RepSEO attack starts when a publisher acquires numerous accounts from an account provider (❶). Advertisers create landing pages for profit schemes like scams or malware distribution and may use redirect services to evade detection (❷). They provide promotion links to the publisher (❸), who embeds these links in packages and publishes them on repositories like npm and NuGet (❹). Once these stages are set, the attack unfolds: victims searching for SEO-targeted keywords find RepSEO packages in top search results (❺). Clicking these links redirects them to landing pages, posing security risks such as scams and malware installation (❻).

B. RepSEO Package Detection

Detecting RepSEO is challenging due to the lack of existing methodologies. Traditional techniques focus on malicious packages compromising operating systems, such as ransomware, spyware, or cryptojacking [13, 14, 9, 15]. However, RepSEO packages typically do not interact with sensitive system APIs, which these methods rely on. Additionally, RepSEO packages are spread across popular software repositories, which are designed for different programming languages and purposes, requiring distinct approaches to model abusive behavior. Developing a universal and robust detection methodology is essential to ensure comprehensive protection.

Our insight is based on the understanding that search engines predominantly index the descriptive content of webpages. Consequently, RepSEO attackers strategically manipulate this descriptive content of the packages to promote their targeted links. Search engines, aiming to balance the trade-off between cost and information collection, prioritize indexing the homepage, such as project introduction and tags, while overlooking other content. As a result, RepSEO packages often prominently display illicit promotional content distinct from other packages on their homepage. Their usage of links, especially short and unpopular links, is also distinct for highlighting of promotion links. Furthermore, the RepSEO package introduction may remain in HTML syntax when migrated from traditional SEO webpages. These distinguishing characteristics help us to construct an effective detection method.

Architecture. Fig. 3 illustrates the process of our methodology. Overall, we first collect a broad spectrum of package information (*e.g.,* package files, metadata, and maintainer). We then built a classification method to discover abusive packages in software repositories, which includes features from five aspects: structure, semantics, links, metadata, and historical behavior. Table I lists the full features we used, and we leave the details of feature selection and definition in [12].

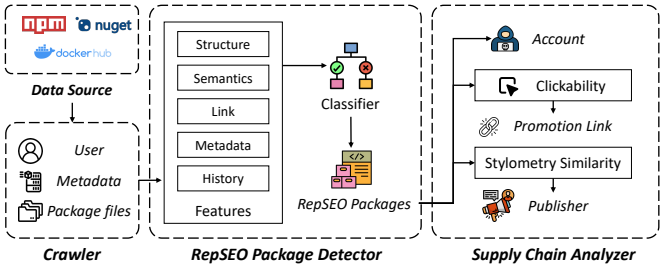

Fig. 3. Methodology overview. We begin by collecting packages on software repositories, then detect abusive packages using a learning approach. Next, we extract the abusive accounts, identify promotion links by clickability, and detect the publishers by measuring stylometry similarity of accounts. The details of feature engineering and feature selection can be found in [12].

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

Using these features, we trained a Random Forest model to classify the abusive packages over the ground truth dataset. Random Forest is an ensemble method combining multiple decision trees, each trained on a random subset of data and features, and makes predictions by majority voting. This random selection helps prevent overfitting, making it suitable for our dataset.

C. Supply Chain Analysis

Upon identifying RepSEO packages, a critical question emerges: how are such abusive packages developed, and who contributes to their creation and publishing? To understand the supply chain of RepSEO, we conduct an in-depth analysis of the three key participants.

1) Account Provider: During registration on software repositories, attackers usually use disposable or free email addresses. Identifying commonly used providers helps software repositories enhance account registration security, and prompt free email services to increase registration thresholds. We extract the maintainer emails in metadata from repositories and use a public email provider list [16] to distinguish public and private email providers.

2) Promotion Link: Identifying embedded promotion links in RepSEO packages poses a significant challenge. Simply finding hyperlinks in package descriptions is inadequate, as these packages often contain links for various purposes.

Our approach to distinguishing promotional links is based on their design to capture user attention and encourage clicks. Unlike manual copy-paste links, promotional links are typically made clickable to facilitate easier user interaction. Thus, we classify clickable links as potential promotional links.

To identify these links, we analyze clickable patterns in software repositories. For instance, on certain platforms (*e.g.,* Docker Hub), where content is rendered as HTML, we identify clickable elements using HTML standards [17] and use regular expressions to search for them systematically.

To monitor post-clicking promotion link activities, we use Puppeteer [18] and CDPSession [19] to capture all network requests associated with each promotional link and reconstruct the redirection sequence by request timestamps.

3) Publisher: Publishers create RepSEO packages by embedding promotional links and deceptive descriptive content. Understanding publishers is vital for both evading detection and enhancing preventive measures by software repositories.

Our analysis observes that publishers employ distinct writing styles and semantic mutation strategies to evade repository detection when creating multiple advertisement copies, with identifiable styles based on specific linguistic preferences.

Stylometry analysis is widely used in identifying the same entity behind different accounts [20, 21, 22]. We use the same method in [20] and customize the text feature by using the hidden state of bert [23] and the vocabulary richness feature by the ratio of unique words to the total number of words in the text. To identify similarities in posting patterns among different accounts, we normalize and average the feature vectors of packages for each account, as the writing style feature vector for that account. We then cluster the account vectors using HDBSCAN [24] to identify publishers.

D. Implementation

To assess our detection tool's effectiveness, we gathered data from three representative software repositories: the JavaScript package manager (npm), the .NET platform NuGet, and the image shipping and deploying platform Docker Hub.

Groundtruth dataset. For benign samples, we acquire the top 9,000 download packages on each platform, and verify 1,000 benign packages with 0 downloads to avoid misclassifying low-popularity packages. As for the malicious RepSEO dataset, we construct it with an unbiased and diverse approach by using representative keywords. We extracted keywords from a previous NuGet report [25], which identified abusive NuGet packages using ad-hoc rules, by term frequency, and incorporated common illicit terms from previous SEO work [26]. This resulted in 52 search keywords covering 10 SEO categories mentioned in [27, 28, 29, 30]. For each keyword, we crawled the top 250 search results in software repositories, manually annotating and randomly selecting 10,000 samples to construct our malicious dataset. Three experts jointly reviewed 100 randomly selected packages (1% of the total) from each of the three software repositories, establishing a consistent standard based on whether packages direct users to external sites without executable code. They then spent around 90 hours independently annotating the rest and measured inter-rater reliability using Krippendorff's alpha, achieving a high score of 0.97, indicating strong agreement [31].

Feature selection. In pursuit of accurately identifying abusive packages on each platform, we tailor our experimental setup to align with the distinct characteristics of each platform.

- *Structural features.* For npm, we scan the rendered `Readme.md/README.md` file, which serves as the package introduction. For NuGet, we scan the `description` field of the `nuspec` file and the `Readme.md` file for as introduction. Some packages only have a `nuspec` file without `Readme.md`. Within Docker Hub, data is derived solely from the web, including short and full descriptions, leaving the total number of files unknown.
- *Metadata features.* For npm and NuGet, we extract their metadata from their files, including license, repository URL, and homepage URL. For Docker Hub, we check if they are official packages and use their download numbers.
- *Historical behavioral features.* We consider the historical feature as the feature vectors of the two packages preceding the current package, belonging to the same user. We retrieve the uploader details of npm packages from the namespace field of the metadata, obtaining the feature vectors of the two prior packages from the same uploader as the current package. For NuGet and Docker Hub, we retrieve the uploader details of NuGet packages from the `authors` field of the metadata.

Training. Then, we train a Random Forest model with *n_estimator=100*, employing a 4:1 split for training and testing sets. Subsequently, we refine the iterative training set by implementing a hard negative case mining method.

E. Evaluation

According to the optimal model, we assess the effectiveness of our abusive package mining approach on the ground truth dataset through 5-fold cross-validation. Our method yielded an average precision and recall of 98.27% and 98.24%.

To verify real-world performance, we compiled a comprehensive dataset with three parts: 1) 500 randomly chosen RepSEO samples, 2) 500 randomly chosen benign samples, and 3) 500 newly uploaded benign packages with zero downloads for potential popularity bias. For three software repositories, we collected 4,500 samples as our real-world dataset.

We find six false positives and 45 false negatives in 4,500 selected cases. Specifically, the six FPs occur because both

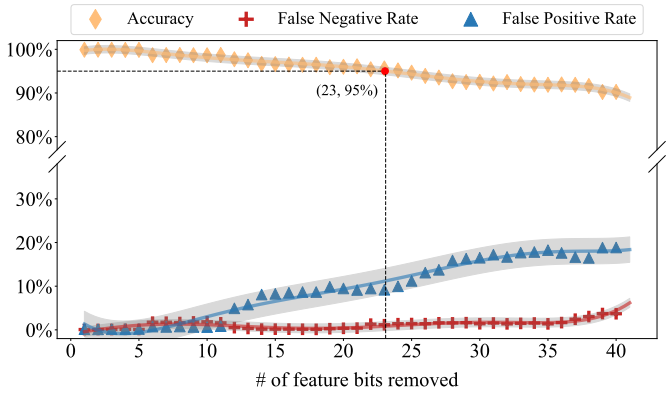

Fig. 4. Drop-off in RepSEO classifier performance when removing the most important feature gradually and re-training.

the textual content and links in their descriptive content are inadequate. Also, we found no FPs in the new packages. As for the FNs, six cases involve malicious software downloads, guiding users to download and install Android applications, with semantics closely resembling legitimate packages. Additionally, 10 promotion links were erroneously integrated with repo links, turning them into benign links and failing to reach the promotion destination, and 29 FNs resulted from insufficient text in the descriptions. Overall, our approach attains an average precision and recall of 99.67% and 97.59% in 4,500 manually labeled samples.

To ensure our method does not overly rely on a limited subset of powerful features, we studied the decay of model performance as we iteratively removed the most important features. We determine the importance of each feature by iteratively removing it, retraining, and evaluating performance. Fig. 4 shows the decrease in accuracy and increase in FPR and FNR as features are removed. Even after removing 23 top feature bits, our classifier still achieves 95% accuracy.

To assess the validity of treating clickable links as promotional, we randomly select 20 RepSEO packages in each repository. By manually labeling 264 links from them, we achieved 100% precision and 94.31% recall, with all false negatives being plain text links without clickable tags.

III. Breakdown of RepSEO

In this section, we conduct the first large-scale measurement study, to gain a comprehensive understanding of blackhat SEO practices in software repositories.

Dataset. We collected a substantial dataset using specialized crawlers for npm and NuGet, following a methodology inspired by prior work [32]. The method involves daily monitoring for updates in package index files, followed by acquiring packages and their metadata. For Docker images, we implemented a customized web crawler using the Docker Hub API described in [33]. Our dataset, covering a wide range of versions as shown in Table II, spans from 2011/01/07 to 2024/03/31 and totals approximately 52.6 terabytes.

TABLE II
Overview of abusive packages detected across three software repositories, with data collection concluding in 2024/03.

Software Repository	Start Time	# P[1]	# RP[2]	% RP[3]
npm [1]	2013/01	3,990,172	929,614	23.30%
NuGet [2]	2011/01	660,194	146,495	22.19%
Docker Hub [3]	2013/04	12,437,277	2,725,573	21.91%
Total	-	17,087,643	3,801,682	22.25%

[1] Number of packages.
[2] Number of RepSEO packages.
[3] Ratio of RepSEO packages.

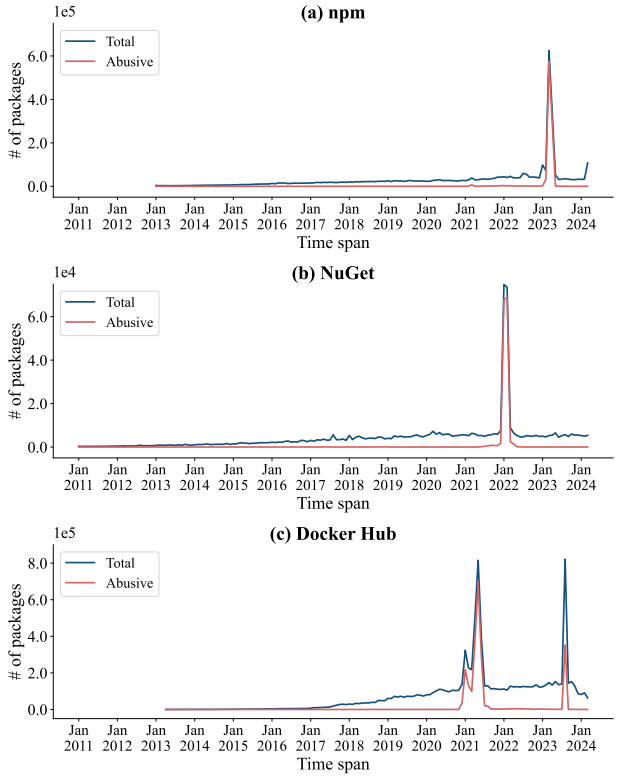

Fig. 5. Evolution of RepSEO packages.

Landscape. Our detection approach applied to 17,087,643 packages, and identified 3,801,682 RepSEO packages on npm, NuGet, and Docker Hub, showing that 22.25% packages are abusive, revealing widespread abuse. Table II shows the distribution of RepSEO packages across the three software repositories. Docker Hub has the highest number (2,725,573) of abusive packages, followed by npm and NuGet. This prevalence strains the storage resources of software repositories and their mirrors, amplifying RepSEO's damage. Even if the official software repository removes packages, mirrors may not follow suit [32], allowing RepSEO's impact to persist.

Longitudinal study. Based on the version history of packages, we analyze the timeline of this abuse in Fig. 5. We locate the package creation time according to the metadata offered by the software repositories. Specifically, we use the "time" field in npm, "published" field in NuGet, and "date_registered" field

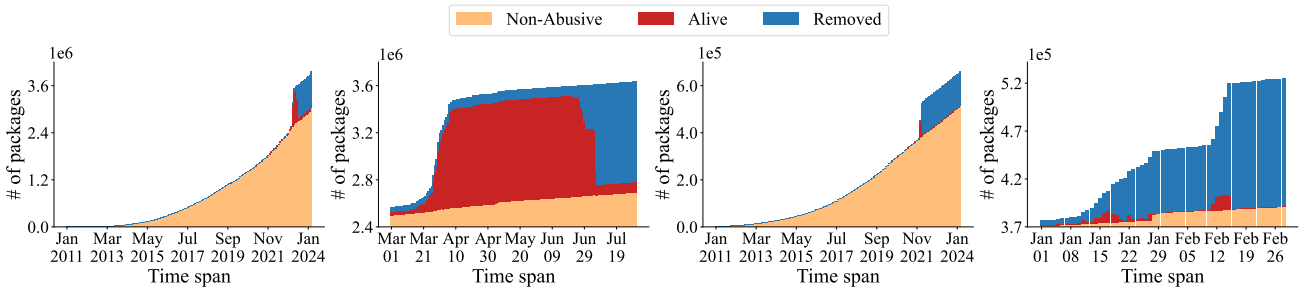

Fig. 6. Lifespan evolution of RepSEO packages on npm and NuGet.

in Docker Hub. The earliest RepSEO package was in 2016/11 for NuGet, 2017/11 for npm, and 2017/04 for Docker Hub.

Interestingly, our study reveals significant spikes in RepSEO package uploads across various software repositories. From 2020/12 to 2021/08, there was a significant rise in abusive Docker images, peaking in 2021/05 with 693,434 images, constituting 85.02% of the uploads that month. Another peak occurred in 2023/08, with 352,496 RepSEO images. A similar trend was observed in npm and NuGet, with 92.07% of npm packages uploaded in 2023/03 and 90.76% of NuGet packages uploaded in 2022/01 being abusive.

Finding I: *All spikes in the total number of packages in software repositories (2023/03 of npm, 2022/01-2022/02 of NuGet, 2021/01-2021/05 and 2023/08 of Docker Hub) have coincided with increases in RepSEO.*

Lifespan. Facing RepSEO attacks, concerns arise regarding software repositories' action and attack duration. We introduce package lifespan as a metric, capturing the time between package upload and software repository removal.

To identify the removal time of packages, we leverage the metadata API provided by the software repositories. For npm packages, we check the "unpublished" field and extract its timestamp as the removal time. For NuGet, we examine the catalog of each package. If the "listed" field is false, we record its "lastEdited" time as the removal time. We then calculate each package's lifespan by subtracting the upload time from the removal time. The maximum lifespan of RepSEO package is 2,210 days for npm and 1,775 days for NuGet.

We then measure the change in software repository response time (Fig. 6). For example, npm noticed the large-scale uploading of abusive packages 4 months after the event occurred, starting remedial actions in 2023/07. Notably, npm has improved its response time, reducing the lifespan of abusive packages from 73 months initially to the current 2.5 hours, indicating effective measures.

Unfortunately, all abusive images were still present on Docker Hub before we reported, with an average lifespan of 952.58 days. Even if Docker Hub claim they have implemented an image security mechanism, they does not take the RepSEO into account.

TABLE III
TOP FIVE ABUSIVE ACCOUNTS BY THE NUMBER OF PACKAGES.

Software Repository	Username	# of Packages
npm	linuxmooerktjmt	80,019
	cikvoroooeiiei	47,391
	jfeifieikkjdkejde334	43,093
	vreopppproo23	36,441
	npmpublishq	34,361
NuGet	nDFE	10,730
	GAMES	2,601
	David	2,127
	Allen	1,623
	Williams	1,385
Docker Hub	scofuterag1988	2,262
	ralicalfa1982	1,190
	fasisasdai1987	1,187
	skiptentida1988	1,187
	steadolunet1989	1,186

IV. Understanding the Supply Chain of RepSEO

This section details the architecture of RepSEO supply chain stakeholders: accounts, promotion links, and publishers.

A. Accounts

The cornerstone of RepSEO is the creation of malicious accounts specifically for mass uploading promotion content. By retrieving account information from metadata, we found 30,981, 84,851, and 140,114 unique abusive accounts in npm, NuGet, and Docker Hub, respectively. Table III shows the users with the most RepSEO packages.

Account usage strategy. We found that account registration cost significantly impacts abusive package utilization, as shown in Fig. 7. 89.12% accounts in NuGet and 90.93% in Docker Hub only upload one package before abandonment, while only 37.62% npm accounts choose this strategy. In npm, we observed a radical strategy involving a small number of accounts uploading many packages—up to 80,019—without concern for attracting attention. Specifically, 54.17% of npm abusive packages are uploaded by just 1.12% of accounts.

The difference in account usage strategies may be attributed to npm's implementation of Multi-Factor Authentication (MFA) [34] for login authentication in 2022. Since npm's abuse peak occurred with MFA while Docker Hub and NuGet

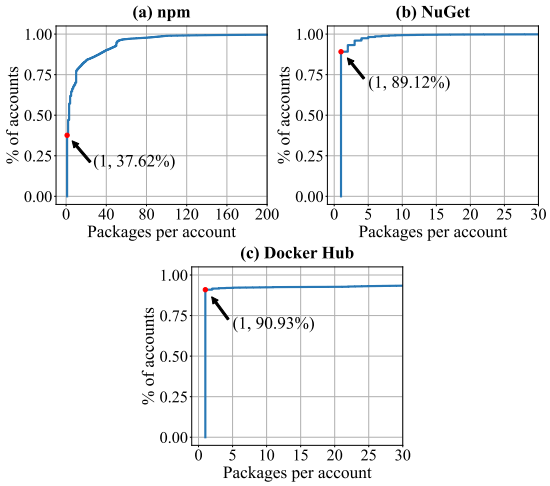

Fig. 7. CDF of packages per account. We found that 89.12% account in (b) and 90.93% in (c) only uploaded one package, while in (a) is 37.62%.

did not, acquiring and utilizing bulk accounts may have been more challenging compared to Docker Hub and NuGet.

Email provider. Account registration cost is linked to the feasibility of automating email account creation. Registration methods differ across platforms, including Microsoft and GitHub accounts, but commonly require email verification. We focus on npm's email service providers, as only npm can access uploader email accounts.

Attackers used 53 free public email services, with Gmail being the most abused, resulting in 4,843 accounts and 475,079 abusive packages. Anonymous registration services like *proton.me* were also abused for greater concealment.

Attackers also registered accounts by 734 private email providers. *pay-exchange.tech* and *pdflivres.com* are prominent, with 4,395 and 2,299 accounts, contributing to 84,071 and 57,786 abusive packages, comprising about 15.26% of total npm abuses. We recommend software repositories implement stricter auditing strategies for private email registrations.

Finding II: *Attackers abuse public email service and deploy self-hosted email services to register numerous accounts on software repositories.*

Hibernated accounts. Additionally, we discovered that a group of hibernated accounts resumes RepSEO activities after a certain period of inactivity. 3,247 accounts resume publishing RepSEO images after over two years in Docker Hub, with each account uploading more than 100 RepSEO images. Furthermore, in NuGet, we have identified four accounts named David (first package in 2015), Allen (2015), Nelson (2017), and James (2020), who became active again after years of hibernation. Their packages uploaded initially were legitimate, but after several years, they started uploading RepSEO packages in 2021. This suggests the possibility of account theft or illicit trading. As npm forced MFA login authentication [34], we did not encounter similar situations within npm.

TABLE IV
TOP 10 UNDERGROUND URL SHORTENING SERVICES.

Domain	# of Links	Ratio of Dataset
picfs.com	166,694	6.27%
fancli.com	158,748	5.97%
blltly.com	156,843	5.90%
tlniurl.com	155,275	5.84%
bytlly.com	155,014	5.83%
tiurll.com	154,444	5.81%
geags.com	152,920	5.75%
imgfil.com	152,226	5.73%
urluss.com	24,544	0.92%
urlca.com	24,520	0.92%

Finding III: *Attackers reactive hibernated accounts, originally used for legitimate packages, to upload RepSEO packages, indicating possible account theft or illicit trading.*

B. Promotion Links

The publisher embeds promotional links in abusive packages as the attack payload to lure clicks. Here we investigate the infrastructure of 9,924,230 promotion links found in RepSEO packages, including the destinations, evasion techniques, and the presence of malicious parties. In these links, 35.63% were inaccessible. Among the accessible, 38.87% led directly to landing pages, and 61.13% involved redirections.

We investigated if redirection links were attacker-created or involved third parties. Self-hosted services usually redirect to a single domain, while third-party services point to multiple destinations. We found 2,761,642 promotion links led to multiple destinations, with only 17.90% (1,143,428 links) pointing to a single SLD.

To identify services that potentially (ab)use links redirecting through multiple destinations, we first filter out the public short links based on a public URL shortener domain list [35]. Then, if the destination domain is contained within the promotion links and the destination domain ranks within the Tranco [36] top 500k, we classify it private redirect service. All remaining links are categorized as "others".

Public URL shortening services. Of the 3,905,070 promotion links involving redirections, 103,871 (57.17%, 41.58%, and 0.82% from *t.co*, *bit.ly*, and *tinyurl.com*, respectively) were short links, distributed across 41,982 npm packages, 16 NuGet packages, and 59,000 Docker images. Notably, *bit.ly* claims to have a spam link filtering system, but is still heavily exploited, indicating the insufficiency of its spam filtering system.

Finding IV: *The security mechanism of public URL shortening services is insufficient to avoid attackers abusing such service on RepSEO.*

Underground URL shortening services. Among the remaining links, we found that 1,582,670 links (59.55%) from 211 domains were flagged as malicious by at least one engine on VirusTotal [37]. After manually inspecting the malicious domains, we found they were URL shortening services. Some

of them have a similar domain name *(typosquatting)* with a public reputable URL shortening service, *i.e., blltly.com* v.s. *bitly.com*. We discovered 32 of the domains that provide underground URL shortening services, as shown in Table IV.

Additionally, we observed that underground shorteners provide more evasion techniques along with URL redirection. For instance, *tiurll.com* dynamically generates unique links for each user. Moreover, *tinybit.cc* uses JavaScript for dynamic page loading to hide landing pages. These methods allow attackers greater control over the redirection process.

Finding V: *Attackers increasingly rely on underground URL shortening services equipped with evasion techniques, like dynamic redirection, to configure their landing pages.*

Private redirect services. In the remaining links, we have also identified a new form of abuse targeting private redirect services. 4,681 unique links abuse redirect services from 23 domains like *www.google.com*, *cse.google.bg*, *www.folkd.com*. These services, like the Google Search example *https://www.google.com/url?q=https%3A%2F%2Fbltlly.com%2F2vw1VP*, track clicks and sources for ad campaign analysis, usually store the redirect link in the parameters of the service API. When users click on these links, they are redirected to the target URL *https://bltlly.com/2vw1VP*, which leads to a malware download page. Notably, eight Google domains are heavily abused, generating 4,658 unique links. This tactic allows promotional links to appear as credible Google domains, evading domain-based security measures.

Finding VI: *Attackers abuse private redirect service of reputable companies, i.e., Google, to redirect RepSEO landing pages.*

Intermediate redirections. We tracked traffic from promotion links to landing pages and found that 54.48% links used multiple redirections to hide their true destination, with the highest number reaching 17. The 301 redirection is consistently employed across 17 intermediate domains, such as *sadisflix.bet*, *coflix.loan*, and *filmoflix.host*, maintaining an unchanged path. Additionally, 807,637 links were first redirected to Cloudflare before reaching the landing page, indicating its use to hide the true origin or improve security and performance. Some links also generate a new landing page link with each visit for human verification, even if the final page remains the same.

Finding VII: *Attackers employ multi-layer redirects (up to 17), along with human verification and firewalls to ensure that only human visitors can access landing pages.*

C. Publishers

As creators of abusive packages, publishers utilize their expertise and resources to maliciously embed promotion links to attract user clicks and interactions with the links. We identified 226 publishers across software repositories: 205 in npm, 17 in NuGet, and five in Docker Hub. The number of promotion links integrated by publishers varies, the most aggressive publishers embed 21.5 unique links per package on average, comparing 2.2 links per package to others.

Publishing capability. Publishers exhibit a strong capacity to publish numerous packages, with 36 publishers releasing over a thousand each. The largest published 1,020,637 packages in five months (2021/01/15 to 2021/06/03), accounting for 8.21% of all packages on Docker Hub, while another released 390,390 packages in one month (2023/03/25 to 2023/04/21) on npm. The number of accounts activated in a short period also reflects a publisher's ability. Using the release time of the first package as the activation time, we identified 13 publishers that activated over 100 accounts each. For example, on NuGet, one publisher activated 62,190 accounts in five days (2022/02/11 to 2022/02/15), and on npm, another publisher activated 3,814 accounts on 2023/04/21.

Finding VIII: *Aggressive publishers can quickly deploy attacks, activating up to 62,190 accounts in just five days.*

Package names. The RepSEO package names are designed to explicitly indicate the purpose of the content within the package. For instance, an package named "down_load_ebook_sundhaftes_geheimnis_by_sawyer_bennett_dzlrs" informs users that they can download an ebook titled "Sundhaftes Geheimnis" written by Sawyer Bennett. To optimize RepSEO package generation, creating a substantial quantity of package names is crucial. We witness three primary mutation methods for name generation: 1) employing variations that maintain semantic meaning, like "down_load" and "dowload" for "download"; 2) appending unique random strings, like "dzlrs", to extend a name; and 3) substituting and rearranging similar words.

Finding IX: *The attacks frequently mutate package metadata to bypass repository potential detection and improve the coverage of search engine queries.*

V. Case Study

In this section, we demonstrate two aggressive RepSEO campaigns, which together account for 58.51% of abusive packages. These cases aim to uncover the underlying technical architecture and profit logic driving such campaigns.

A. Survey-based Aggressive Advertisers

We identified an aggressive survey-based [38] scam advertiser campaign on Docker Hub. These advertisers direct users to click on fake download links for e-books or software, leading them to survey platforms offering monetary rewards, where users may be coerced into divulging personal information.

Profit logic. Fig. 8 illustrates the advertiser's promotion process, encompassing four stages. ❶ Upon clicking the promotion link, users reach a human verification page (*e.g., verif.za.com*), confirming an operational PayPal account. If absent, users are prompted to register for a platform account. ❷ At the Register stage (*e.g., app.rewardflux.com*), users are

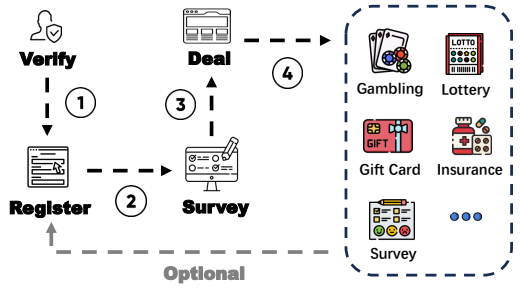

Fig. 8. The process of survey-based advertiser promotion.

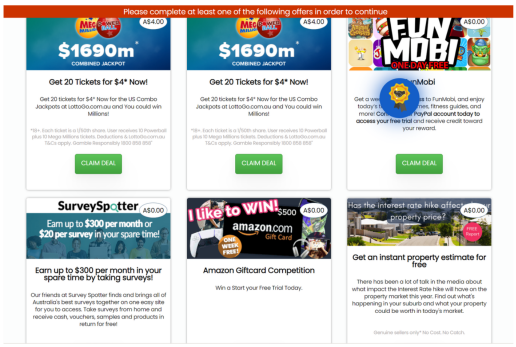

Fig. 9. The task distribution portal page of a survey-based advertiser.

required to disclose their financial status, including their housing situation. Also, users be asked to provide their name, email address, and address, which may lead to sensitive privacy leakage. ❸ In the Survey stage, users are asked enticing questions like "Are you interested in online gambling?" to encourage questionnaire completion for a "gift card". ❹ In the Deal stage, users enter a task distribution portal (Fig. 9) where they must complete at least one task, including online gambling, gift card acquisition, and health insurance purchase, all requiring card account information. These portals also link to other survey-based platforms. However, even if users complete the sub-task, they will not receive the "gift card", but divulge their personal information.

Advertiser patterns. This campaign includes 1,334,307 abusive packages, spanning from 2021/01/15 to 2021/06/18. Before 2021/06/03, 971 accounts were using ebooks to attract victims, after that, the topic changed to software downloads, and another 272 accounts with similar usernames are activated. Each account uploads about 628 packages, with all the links leading to the survey platform.

Finding X: *RepSEO attackers distribute survey scams, tricking users into providing sensitive information by promising gift cards, and lead to other scams.*

B. Malware Distribution Advertisers

We also detect massive abusive packages utilized for malware distribution on software repositories. These packages disguise themselves as convenient file download channels instead of directly providing identifiable malware [39].

Profit logic. The malware distribution process is concise: users, clicking promotional links, first encounter a short link redirecting them to a human verification interface. Malware is concealed within zip files named after the user's interest, like Tamil-Dubbed-Movies-Free-Download-In-720p-English.zip. After passing verification, users are directed to a download instruction page with the unzip password. Such password is a social engineering trick to downgrade user awareness. The downloaded file, identified as a Trojan virus, establishes a botnet by altering repository keys and disabling system security tools, allowing malicious software downloads from the C&C server.

Advertiser patterns. This campaign includes 890,201 abusive packages, released by 5,395 accounts from 2018/01/06 to 2023/08/22. Two peaks were observed during this period: in three days after 2020/12/11, 2,291 accounts were created and released 587,947 abusive packages; and in one day of 2021/04/17, 1,959 accounts were created to release 150,028 packages.

For this campaign, the advertiser uploads multiple packages simultaneously with varying attractive titles, to lure users to download diverse kinds of files, such as mp3, movie, and APK. Victims, after clicking RepSEO webpages, are all guided towards the download of the Trojan virus called *file_1234* with password 1234.

The advertising strategy has evolved. Two years ago, abusive packages used private short links and static HTML "Download" buttons, with some failing due to rendering errors. This year, packages use improved rendering and *google.com* for redirecting promotional links to bypass mitigation measures. Short links in abusive packages enable attackers to easily switch domains, extending their operations. For example, domains changed from *psfmi.com* two years ago to *ytomb.com* in 2023/08 and *ooppnm.com* in 2023/11. Given the prevalent nature of RepSEO, attackers successful on one platform often attempt cross-platform attacks, as seen in a malware campaign affecting both npm and NuGet.

Finding XI: *Attackers post download links to e-books or cracked software in software repositories, redirecting users to malware downloads.*

VI. Discussion

In this section, we discuss the security implications, then propose countermeasures against the emerging threat and discuss ethical considerations and responsible disclosure.

A. Security Implication

This paper aims to increase awareness of RepSEO within the open-source community. Despite their few downloads and dependencies, RepSEO packages significantly impact software engineering, especially in open-source ecosystems that are critical to modern development. Their high volume strains storage resources and they can remain in downstream mirrors

even after upstream removals. While RepSEO packages do not directly compromise software functionality, they present notable challenges to reputation and user trust in the software ecosystem. More critically, 22.25% of these packages are noise, complicating data collection and analysis for researchers. Before our report, limited knowledge of RepSEO has led to 2.7 million RepSEO packages persisting on Docker Hub. We also witnessed RepSEO in many software repositories and its downstream mirrors, such as PyPI, replit, and MyGet, showing their unawareness.

Our work builds on prior blogs and technical reports [40, 41, 42] related to RepSEO, which fall under the category of grey literature (GL). As noted by Garousi et al. [43], GL serves as a valuable source of knowledge, particularly in software engineering, where practitioners often share insights and experiences outside of peer-reviewed publications. While these prior works provided isolated observations on RepSEO incidents, they lacked systematic methodologies for identifying and analyzing such packages and primarily focused on phishing incidents. By synthesizing and extending this foundational knowledge, we fill this gap by proposing a universal and robust RepSEO detection methodology that can easily adapt to other software repositories and programming languages, helping them detect RepSEO packages.

Our comprehensive analysis of the underground activities within the software supply chain has yielded invaluable insights, which provide valuable context for mitigation strategies in Section VI-B. Notably, we identified private email providers and underground shortening services as pivotal elements in the spread of RepSEO. These findings highlight the growing threat to open-source communities, which are increasingly targeted by cybercriminals. Addressing this threat is crucial to maintaining reputation and trust in these platforms. In response, we have open-sourced our detection methods to help mitigate the RepSEO threat.

B. Mitigation

Countermeasures. Based on our understanding of RepSEO, we propose several countermeasures to mitigate this emerging and underestimated threat. Specifically, (1) software repositories should strengthen their account management practices, especially checking accounts registered with non-reputable or unknown email providers with high occurrence. (2) In addition to checking for malicious code, the content of uploaded packages should be examined for suspicious links. Short links are uncommon in software repositories, thus requiring increased vigilance. Frequently appearing low-ranking domains across different packages should also be scrutinized. (3) Monitor accounts that quickly upload many packages or appear simultaneously, as they may be linked to the same publisher. Learning their patterns and promptly removing newly uploaded similar packages and accounts is crucial. (4) Software repositories can employ our detection method for RepSEO packages, which can be easily adopted with minimal effort.

Adapt for other platforms. Our approach uses multiple features to detect RepSEO packages, allowing software repositories to map these features to their own attributes. Due to resource limitations, we conducted detections only on three platforms, but our method can also be applied to others, such as PyPI, GitHub, and Bitbucket. Using PyPI as an example, we can leverage the "Project Description" field for extracting structural, semantic, and link features. The "Metadata.License" and "Project Links.Homepage" fields can be metadata features. Notably, our feature engineering primarily focuses on the promotional project introduction, and the absence of specific metadata features would not hinder our method's usability.

C. Limitations

Although our work has uncovered valuable insights into the RepSEO attacks, it has limitations. We only analyzed three software repositories due to limited computing resources. With more resources, we could extend the experiments to other software repositories, potentially detecting more RepSEO samples. While our detection method is robust against the current generation of RepSEO, attackers may eventually find ways to bypass it. However, we significantly increase the difficulty for them to do so, raising the cost and effort required. For instance, reducing the semantic distance to the software repository prevents user attraction. Also, maintaining a good historical record is expensive for attackers managing thousands of RepSEO packages in one account.

We also acknowledge that not all promotional links were successfully located under our assumptions. However, since manual evaluation has confirmed that we have identified most (94.31%) promotional links, we believe that our study has provided comprehensive insights into the strategies behind these links, *i.e.,* utilization of underground URL shorteners and redirection methods.

Another limitation is our inability to directly assess the monetization of the attacks due to a lack of publicly available datasets with specific pricing details, restricting our financial analysis of repository exploitation.

D. Ethics and Disclosure

Ethic concerns. We place a strong emphasis on ethical considerations. To evaluate security issues within software repositories, we maintain continuous monitoring and assessment of the status of packages in targeted repositories. Notably, all our experiments were conducted using legitimate methods. The information we gathered through crawling is publicly available and can be obtained using authorized approaches. During the crawling process, as we crawled the packages based on the changes in the package index, we ensured that the number of requests made to the software repositories was reasonable and did not impose excessive strain on their infrastructure.

Disclosure. Since the discovery of blackhat SEO abuse through software repositories, we have been in active communication with the software repositories affected. So far, we have reported the abusive packages we found to npm, NuGet, and Docker Hub. By now, all three repositories have responded to our report and are offlining related packages.

VII. Related Work

Blackhat SEO. To enhance efficacy in countering blackhat SEO, multiple studies have been conducted to gain insight into these practices. Attackers employ a range of tactics to execute blackhat SEO, including manipulating specific site elements and exploiting search engine mechanisms. Attackers utilized search redirection attacks to promote illicit drugs [44, 45] and used iframe masquerading techniques to promote counterfeit luxury brands [10], which make it easier to advertise on pre-existing portal pages. Attackers also exploit autocomplete manipulation of search queries to generate promotion words [46].

The scope of blackhat SEO has expanded beyond search engines in recent years, including cloud-based long-tail SEO spam [47], tweets on the illegal sale of controlled substances on social media [48], exploitation in Google Maps location search [49], extended by Wang et al. [30] to detect illegal drug promotion listings on local search services. From a threat modeling standpoint, Lin et al. [50] proposed the MAWSEO method, automating covert blackhat SEO by modifying wiki articles to promote illicit businesses. In contrast, we focus on the escalating blackhat SEO ecosystem across various software repositories, aiming to identify and summarize abusive behaviors and understand attackers' scaling patterns.

In terms of detection techniques, Liao et al. [26] analyzed infected websites on selected sTLDs via search result snippets' semantics. [29] developed a jargon normalization algorithm to track endorsed websites. Besides, SCDS [51] detects blackhat SEO deception trends through semantic obfuscation by assessing linguistic semantic diversity within websites. Compared to these techniques based on web search results, our paper focuses on the single package of the repositories, which is highly structured. We synthesize a variety of features within the packages from different dimensions, including but not limited to semantics and links, to enhance the effectiveness of this detection method.

Threats on software repositories. Reusable software packages are a critical component of modern software development. Security issues within the software supply chain have received widespread attention. Numerous empirical studies have been conducted to investigate related ecosystems [13, 52, 14, 8, 15], addressing specific forms of attacks such as typosquatting [9] and vulnerable package dependencies [6, 7]. Zimmermann et al. [52] developed indicators to assess npm's attack risks, examining package dependencies, maintainers, and known vulnerabilities. Analyzing the metadata of 1.63 million JavaScript npm packages, Zahan et al. [53] identified signals of security vulnerabilities, including installation scripts, maintainers with expired email domains, and inactive packages. Gu et al. [32] investigated potential vulnerabilities in six popular software registry ecosystems and identified twelve potential attack vectors. To address potential malicious npm packages, Sejfia and Schäfer [4] have implemented a lightweight machine learning-based technology called Amalfi. Previous research focused on direct malicious behavior, overlooking abusive blackhat SEO through software repositories due to subtle indicators of intent. Our study is the first to systematically analyze this new threat, advancing software supply chain security research.

VIII. Conclusion

In this paper, we have a comprehensive analysis of blackhat SEO through software repositories, named RepSEO, a novel and emerging threat in the landscape of software supply chains. Our study unveiled the extensive abuse of popular platforms such as npm, NuGet, and Docker Hub, where attackers exploit the open-source community's credibility to manipulate search engine results through SEO abusive packages. This investigation led to the discovery of 3,801,682 RepSEO packages, highlighting the widespread of this attack. We identified the critical roles of account providers, advertisers, and publishers in the ecosystem. Our study provides actionable insights for these software repositories to enhance their security mechanism and mitigate such abuses. We have informed npm, NuGet, and Docker Hub about RepSEO packages and reported the associated supply chain vulnerabilities to Google. They have acknowledged this and have begun removing the abusive packages and links.

Acknowledgment

We would like to thank Ruimin Wang for providing technical and data assistance for this paper. We also thank the anonymous reviewers for their insightful comments that helped improve the quality of the paper. This work was supported in part by National Natural Science Foundation of China (62302101, 62102093). Min Yang is the corresponding author, and a faculty of Shanghai Institute of Intelligent Electronics & Systems, and Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China.

References

[1] I. npm. (2023) Normally pleasant mixture — npm — home. *https://npmjs.com*.

[2] Microsoft. (2023) Nuget gallery — home. *https://www.nuget.org/*.

[3] I. Docker. (2023) Docker hub container image library. *https://hub.docker.com*.

[4] A. Sejfia and M. Schäfer, "Practical automated detection of malicious npm packages," in *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*. ACM, 2022, pp. 1681–1692.

[5] N. Boucher and R. Anderson, "Trojan source: Invisible vulnerabilities," in *32nd USENIX Security Symposium (USENIX Security 23)*, Anaheim, CA, Aug. 2023, pp. 6507–6524.

[6] J. C. Davis, E. R. Williamson, and D. Lee, "A sense of time for JavaScript and node.js: First-Class timeouts as a cure for event handler poisoning," in *27th USENIX Security Symposium (USENIX Security 18)*, Baltimore, MD, Aug. 2018, pp. 343–359.

[7] C. Staicu, M. Pradel, and B. Livshits, "SYNODE: understanding and automatically preventing injection attacks

on NODE.JS," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, 2018.

[8] D. L. Vu, I. Pashchenko, F. Massacci, H. Plate, and A. Sabetta, "Towards using source code repositories to identify software supply chain attacks," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. ACM, 2020, pp. 2093–2095.

[9] R. Duan, O. Alrawi, R. P. Kasturi, R. Elder, B. Saltaformaggio, and W. Lee, "Towards measuring supply chain attacks on package managers for interpreted languages," in *Proceedings 2021 Network and Distributed System Security Symposium*, 2021.

[10] D. Y. Wang, M. Der, M. Karami, L. Saul, D. McCoy, S. Savage, and G. M. Voelker, "Search + seizure: The effectiveness of interventions on seo campaigns," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, New York, NY, USA, 2014, p. 359–372.

[11] K. Thomas, C. Grier, D. Song, and V. Paxson, "Suspended accounts in retrospect: an analysis of twitter spam," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011, pp. 243–258.

[12] Anonymous. (2024) Repseo-artifacts. *https://anonymous.4open.science/r/RepSEO_Classifer-FDB5*.

[13] A. Decan, T. Mens, and E. Constantinou, "On the impact of security vulnerabilities in the npm package dependency network," in *Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018*, A. Zaidman, Y. Kamei, and E. Hill, Eds. ACM, 2018, pp. 181–191.

[14] A. Decan, T. Mens, and P. Grosjean, "An empirical comparison of dependency network evolution in seven software packaging ecosystems," *Empir. Softw. Eng.*, vol. 24, no. 1, pp. 381–416, 2019.

[15] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "Sok: Taxonomy of attacks on open-source software supply chains," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 1509–1526.

[16] ammarshah. (2023) A list of all email provider domains. *https://gist.github.com/ammarshah/f5c2624d767f91a7cbdc4e54db8dd0bf#file-all_email_provider_domains-txt*.

[17] W3C. (2023) Html standard. *https://www.w3.org/TR/2011/WD-html5-20110405/*.

[18] Puppeteer. (2020) Puppeteer. Https://pptr.dev/.

[19] cyrus and. (2022) chrome-remote-interface. Https://github.com/cyrus-and/chrome-remote-interface.

[20] Y. Zhang, Y. Fan, W. Song, S. Hou, Y. Ye, X. Li, L. Zhao, C. Shi, J. Wang, and Q. Xiong, "Your style your identity: Leveraging Writing and Photography Styles for Drug Trafficker Identification in Darknet Markets over Attributed Heterogeneous Information Network," in *The World Wide Web Conference*, May 2019, pp. 3448–3454.

[21] S. Afroz, A. C. Islam, A. Stolerman, R. Greenstadt, and D. McCoy, "Doppelgänger Finder: Taking Stylometry to the Underground," in *2014 IEEE Symposium on Security and Privacy*, May 2014, pp. 212–226.

[22] P. Chairunnanda, N. Pham, and U. Hengartner, "Privacy: Gone with the typing! identifying web users by their typing patterns," in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, 2011, pp. 974–980.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[24] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.

[25] Jossef. (2023) illustria checkmarx phishing campaign package list. *https://gist.github.com/jossef/1c1152368ff6210340644f44afec7e8e*.

[26] X. Liao, K. Yuan, X. Wang, Z. Pei, H. Yang, J. Chen, H. Duan, K. Du, E. Alowaisheq, S. Alrwais, L. Xing, and R. Beyah, "Seeking nonsense, looking for trouble: Efficient promotional-infection detection through semantic inconsistency search," in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 707–723.

[27] M. Joslin, N. Li, S. Hao, M. Xue, and H. Zhu, "Measuring and Analyzing Search Engine Poisoning of Linguistic Collisions," in *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019, pp. 1311–1325.

[28] L. Lu, R. Perdisci, and W. Lee, "Surf: detecting and measuring search poisoning," in *Proceedings of the 18th ACM conference on Computer and communications security*, New York, NY, USA, Oct. 2011, pp. 467–476.

[29] R. Yang, X. Wang, C. Chi, D. Wang, J. He, S. Pang, and W. C. Lau, "Scalable detection of promotional website defacements in black hat SEO campaigns," in *30th USENIX Security Symposium (USENIX Security 21)*, Aug. 2021, pp. 3703–3720.

[30] P. Wang, Z. Lin, X. Liao, and X. Wang, "Demystifying local business search poisoning for illicit drug promotion," in *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*. The Internet Society, 2022.

[31] A. F. Hayes and K. Krippendorff, "Answering the call for a standard reliability measure for coding data," *Communication methods and measures*, vol. 1, no. 1, pp. 77–89, 2007.

[32] Y. Gu, L. Ying, Y. Pu, X. Hu, H. Chai, R. Wang, X. Gao, and H. Duan, "Investigating package related security threats in software registries," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 1578–1595.

[33] pedrorijo91. (2019) Api to get top docker hub images. *https://stackoverflow.com/questions/38070798/where-is-the-new-docker-hub-api-documentation*.

[34] GitHub. (2023) All npm accounts are now enrolled in login verification. *https://github.blog/changelog/2022-03-01-all-npm-accounts-are-now-enrolled-in-login-verification/*.

[35] A. Neumann, J. Barnickel, and U. Meyer. (2011) Security and privacy implications of url shortening services. *https://www.ieee-security.org/TC/W2SP/2011/papers/w2sp-urlShortening-slides.pdf*.

[36] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings of the 24th Network and Distributed System Security Symposium (NDSS)*, 2019.

[37] VirusTotal. (2020) Virustotal. *https://www.virustotal.com/*.

[38] A. Kharraz, W. Robertson, and E. Kirda, "Surveylance: Automatically detecting online survey scams," in *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P)*, 2018, pp. 70–86.

[39] P. Vadrevu and R. Perdisci, "What You See is NOT What You Get: Discovering and Tracking Social Engineering Attack Campaigns," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, Oct. 2019, pp. 308–321.

[40] J. Harush. (2022) How 140k nuget, npm, and pypi packages were used to spread phishing links. *https://checkmarx.com/blog/how-140k-nuget-npm-and-pypi-packages-were-used-to-spread-phishing-links/*.

[41] Y. Gelb. (2023) How npm packages were used to spread phishing links. *https://checkmarx.com/blog/how-npm-packages-were-used-to-spread-phishing-links/*.

[42] L. Valentić. (2023) Operation brainleeches: Malicious npm packages fuel supply chain and phishing attacks. *https://www.reversinglabs.com/blog/operation-brainleeches-malicious-npm-packages-fuel-supply-chain-and-phishing-attacks*.

[43] V. Garousi, M. Felderer, M. V. Mäntylä, and A. Rainer, *Benefitting from the Grey Literature in Software Engineering Research*. Cham: Springer International Publishing, 2020, pp. 385–413. [Online]. Available: *https://doi.org/10.1007/978-3-030-32489-6_14*

[44] N. Leontiadis, T. Moore, and N. Christin, "Measuring and analyzing Search-Redirection attacks in the illicit online prescription drug trade," in *20th USENIX Security Symposium (USENIX Security 11)*, San Francisco, CA, Aug. 2011.

[45] N. Leontiadis, T. Moore, and Christin, "A nearly four-year longitudinal study of search-engine poisoning," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, p. 930–941.

[46] P. Wang, X. Mi, X. Liao, X. Wang, K. Yuan, F. Qian, and R. A. Beyah, "Game of missuggestions: Semantic analysis of search-autocomplete manipulations," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, 2018.

[47] X. Liao, C. Liu, D. McCoy, E. Shi, S. Hao, and R. Beyah, "Characterizing long-tail seo spam on cloud web hosting services," in *Proceedings of the 25th International Conference on World Wide Web*, Republic and Canton of Geneva, CHE, 2016, p. 321–332.

[48] T. K. Mackey, J. Kalyanam, T. Katsuki, and G. Lanckriet, "Twitter-based detection of illegal online sale of prescription opioid," *American Journal of Public Health*, vol. 107, no. 12, pp. 1910–1915, 2017, pMID: 29048960.

[49] D. Y. Huang, D. Grundman, K. Thomas, A. Kumar, E. Bursztein, K. Levchenko, and A. C. Snoeren, "Pinning down abuse on google maps," in *Proceedings of the 26th International Conference on World Wide Web*. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, p. 1471–1479.

[50] Z. Lin, Z. Li, X. Liao, X. Wang, and X. Liu, "Mawseo: Adversarial wiki search poisoning for illicit online promotion," in *2024 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2024, pp. 49–49.

[51] H. Yang, K. Du, Y. Zhang, S. Hao, H. Wang, J. Zhang, and H. Duan, "Mingling of clear and muddy water: Understanding and detecting semantic confusion in blackhat seo," in *Computer Security – ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2021, p. 263–284.

[52] M. Zimmermann, C.-A. Staicu, C. Tenny, and M. Pradel, "Small world with high risks: A study of security threats in the npm ecosystem," in *28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA, Aug. 2019, pp. 995–1010.

[53] N. Zahan, T. Zimmermann, P. Godefroid, B. Murphy, C. S. Maddila, and L. A. Williams, "What are weak links in the npm supply chain?" in *44th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2022, Pittsburgh, PA, USA, May 22-24, 2022*. IEEE, 2022, pp. 331–340.