TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Alachkar, K., Gaastra, D., Gadyatskaya, O., Barbaro, E., Van Eeten, M., & Zhauniarovich, Y. (2026). VSMEx: A Collection Tool and a Dataset of Malicious VS Code Extensions: Data/Toolset Paper. In *CODASPY 2026 - Proceedings of the 16th ACM Conference on Data and Application Security and Privacy* (pp. 138-144). Association for Computing Machinery (ACM). https://doi.org/10.1145/3800506.3803487

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

VSMEx: A Collection Tool and a Dataset of Malicious VS Code Extensions

Data/Toolset Paper

Kotaiba Alachkar
k.alachkar@tudelft.nl
TU Delft
Delft, The Netherlands

Dirk Gaastra
dirk@gcs.engineering
Independent Researcher
The Netherlands

Olga Gadyatskaya
o.gadyatskaya@liacs.leidenuniv.nl
Leiden University
Leiden, The Netherlands

Eduardo Barbaro
e.wildebarbaro@tudelft.nl
TU Delft
Delft, The Netherlands

Michel van Eeten
m.j.g.vaneeten@tudelft.nl
TU Delft
Delft, The Netherlands

Yury Zhauniarovich
y.zhauniarovich@tudelft.nl
TU Delft
Delft, The Netherlands

Abstract

Visual Studio Code (VS Code) is one of the most widely used code editors, and its extension ecosystem has increasingly become a target for software supply chain attacks. Developing and validating effective detection techniques in this area requires ground-truth data with both benign and malicious samples. While benign samples are easy to obtain, no publicly available or continuously updated dataset exists for malicious VS Code extensions. To address this gap, we built VSMEx, a continuously updated dataset of malicious VS Code extensions derived from Microsoft's official malicious and removed lists. Over a deployment period of more than three months, VSMEx successfully captured 214 extensions, demonstrating the viability of our approach. In this work, we present an initial analysis of the resulting dataset and share the associated metadata and the list of flagged or removed extensions collected during this period. In addition, we provide controlled access to the dataset itself, facilitating further research and contributing to the security of the VS Code ecosystem. Note that VSMEx continues to operate, making the resulting dataset well suited for training and validation in continuous machine learning settings.

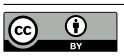
CCS Concepts

• **Security and privacy → Malware and its mitigation**; **Software and application security**; • **Software and its engineering → Integrated and visual development environments**.

Keywords

VS Code; Malicious Extensions; Dataset; Collection Tool

ACM Reference Format:
Kotaiba Alachkar, Dirk Gaastra, Olga Gadyatskaya, Eduardo Barbaro, Michel van Eeten, and Yury Zhauniarovich. 2026. VSMEx: A Collection Tool and a Dataset of Malicious VS Code Extensions: Data/Toolset Paper. In *Proceedings of the Sixteenth ACM Conference on Data and Application Security and Privacy (CODASPY '26), June 23–25, 2026, Frankfurt am Main, Germany*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3800506.3803487



This work is licensed under a Creative Commons Attribution 4.0 International License.
CODASPY '26, Frankfurt am Main, Germany
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2562-3/2026/06
https://doi.org/10.1145/3800506.3803487

1 Introduction

Developers and their devices have become the key targets in Software Supply Chain (SSC) attacks [27]. In early 2025, the Bybit crypto exchange suffered a major security breach in which attackers exploited a compromised developer workstation, inserting malicious code into the wallet software and siphoning off nearly $1.5 billion from the exchange's cold wallets [36]. A more recent attack involved a self-replicating worm called "Shai-Hulud" that scanned developer and Continuous Integration (CI) environments for exposed credentials and exfiltrated tokens to modify and publish infected package versions to the npm registry [5]. Finally, a recent variant of the XCSSET macOS malware has employed a novel propagation strategy that involves searching for and infecting Xcode projects on the compromised device [2]. All these high-profile attacks underscore the need to secure developers' infrastructure.

Nowadays, developers' primary work tools are either Integrated Development Environments (IDEs), such as IntelliJ IDEA [14], or advanced code editors like Visual Studio Code (VS Code) [23]. Among the latter, VS Code stands out: Microsoft reports tens of millions of developers adopting it [30], and the 2025 Stack Overflow Developer Survey [32] shows that 75.9% of developers used it regularly over the past year. A key factor contributing to this popularity is VS Code's high degree of extensibility: with more than 100,000 available extensions, developers can customize this editor to meet almost any requirement [30]. However, extensions have also started being employed for malicious purposes. For example, in 2023, Check Point uncovered a malicious extension able to steal secrets and establish remote connections [1]. In June 2024, a group of researchers demonstrated practical risks by "infecting" over one hundred organizations through a trojanized copy of a popular extension [35]. Their broader analysis of the VS Code Marketplace further revealed 1,283 extensions containing known malicious code, 8,161 communicating with hardcoded IP addresses, and 1,452 executing unknown binaries. These findings highlight the scale and urgency of the problem and motivate the need for automated techniques capable of detecting malicious extensions at scale.

Over the past decade, machine learning (ML)-based approaches have proven their efficacy for detecting malicious software, offering accuracy, scalability, and speed [4, 17, 9, 25]. However, building and validating effective ML models requires ground-truth data: verified benign and malicious samples [15, 10]. While benign extensions are relatively easy to obtain, for example, by taking extensions only from reputable publishers, collecting a diverse and representative set of malicious samples is far more challenging. This lack of reliable ground truth hinders research progress and the advancement of detection methods.

In this paper, we introduce *VSMEx*, a tool that maintains a continuously updated ground-truth dataset of malicious VS Code extensions. To achieve this, the tool periodically crawls the entire Marketplace to maintain a local cache of all extension packages and metadata, and continuously monitors two Microsoft-maintained lists that report flagged or removed extensions. By cross-checking these lists against the local cache, VSMEx identifies extensions that have disappeared from the Marketplace, retrieves their package files and metadata from the cache, and preserves them to support the development and evaluation of detection techniques. We publicly share the lists of extensions and the collected metadata, while restricting the access to the corresponding extension packages themselves only to verified academic researchers.

Thus, the contributions of this paper are the following:

- **Tool sharing:** We release the full source code of VSMEx[1] – our tool used to continuously collect the dataset of malicious VS Code extensions.
- **Data sharing:** We share the lists and metadata of the malicious extensions collected by VSMEx. We share the extensions[2] themselves with verified academic researchers.
- **Dataset characterization:** We perform an initial analysis of the dataset by examining the collected extension packages and their associated metadata.

It is important to note that the dataset analyzed in this paper consists of a fixed snapshot of 214 extensions collected between September 3 and December 23, 2025, while *VSMEx* continues collecting and preserving newly flagged or removed extensions.

2 Background

VS Code is an open-source, lightweight, and cross-platform code editor developed by Microsoft that provides rich, IDE-like functionality through extensions. Its extension model allows extensions to run on any platform supported by VS Code and covers a broad range of types, including themes, debugging tools, and language-specific features such as syntax highlighting and linting.

To distribute extensions, Microsoft operates Visual Studio Marketplace, the official repository where publishers can upload and share their VS Code extensions. Although alternatives exist – such as the Open VSX Registry [8] – Microsoft Marketplace remains the default and most widely used channel. Before publishing an extension, third-party developers need to register a Microsoft account and create a publisher profile. Once set up, they can package and upload their extensions.

[1]https://github.com/kalachkar/vsmex
[2]https://github.com/kalachkar/vsmex-dataset

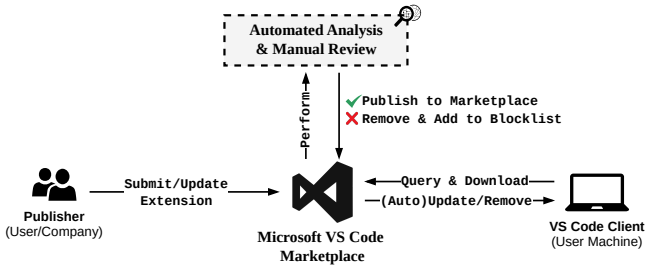


Figure 1: Simplified lifecycle of a VS Code extension from submission to removal.

During the publication process, an extension is vetted before appearing in the Marketplace to detect malicious or policy-violating extensions [21, 13]. New or updated extension submissions first undergo automated analysis, including malware scanning, sandbox testing, and name-squatting detection. Only extensions that pass this vetting process are published. Extensions identified as malicious at this stage are prevented from appearing in the Marketplace and are added to a blocklist. Releasing updates follows the same process as the initial publication.

Once an extension is installed, VS Code periodically checks for updates and, by default, installs them automatically. If an already published extension is later found to be malicious, whether through user reports, ongoing analysis, or later review, it is also removed from the Marketplace and added to a blocklist. This prevents further installation from the Marketplace and triggers automatic uninstallation from VS Code clients. This two-stage process ensures that malicious extensions are contained both before and after publication. Once removed, an extension is no longer available for download. The simplified publication lifecycle is shown in Figure 1.

3 VSMEx

In this section, we present the collection tool and describe its design and operation. We also detail the schema of the resulting dataset, outlining the metadata collected for each extension.

3.1 Collection Tool Design and Modus Operandi

The Marketplace documentation indicates [21] that if a malicious extension is identified, it is removed from the Marketplace, added to a blocklist, and automatically uninstalled from users' VS Code installations. This implies that VS Code should somehow check which extensions are marked as malicious and perform their automatic removal. Since we were unable to find documentation describing the details of this process, we conducted an empirical investigation to understand how VS Code detects malicious extensions in practice. Specifically, we carried out a close inspection of all API calls performed by VS Code and identified a JSON endpoint[3] that exposes a list of malicious extensions (hereafter *malicious list*) alongside other metadata, such as search recommendations and deprecated entries. In addition, we found a public log of removed extensions referenced in an official Microsoft blog post [13], hosted in Microsoft's GitHub

[3]https://main.vscode-cdn.net/extensions/marketplace.json

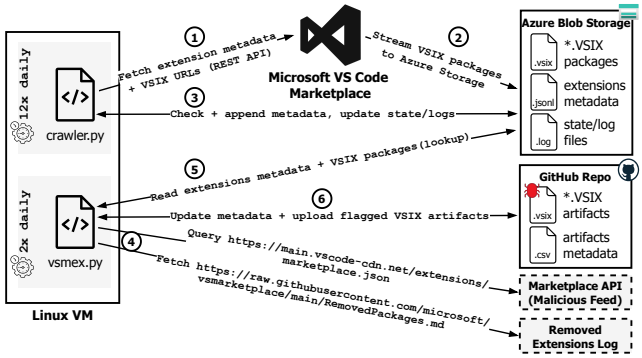


Figure 2: Modus operandi of the automated collection tool.

repository and maintained by Microsoft personnel[4] (hereafter *removed list*), which includes removal dates and reasons.

These two lists represent distinct parts of Microsoft's enforcement pipeline: the malicious list is consumed by the VS Code client to trigger automatic removal, while the removed list is maintained manually by Microsoft personnel and includes broader policy violations. Although both lists relate to extension takedowns, they differ in scope and only partially overlap. Notably, the two lists do not subsume one another; some extensions appear only in the malicious list, while others appear only in the removed list, and Microsoft provides no public explanation for this mismatch.

Building on these findings, we developed *VSMEx*, a Python-based tool that collects and maintains an up-to-date dataset of extensions flagged as malicious or removed. The tool runs in the cloud on Microsoft Azure on a Linux VM (Ubuntu 24.04.3 LTS) and depends only on the `requests` [26], `azure-core` [18], and `azure-storage-blob` [19] Python packages. VSMEx relies on Azure Blob Storage[5] (ABS) to store data, and uses the GitHub API[6] to publish automatically identified extensions and their metadata to a repository. We selected cloud storage to ensure that our cache can elastically scale up as more extension versions are accumulated. We chose GitHub to simplify the sharing process. It is important to note that the codebase itself is platform-agnostic and can be run outside Azure; however, the current implementation uses ABS as the backend and the GitHub API for publishing metadata. While the storage account and target GitHub repository are configurable, switching to different storage backends (e.g., AWS S3) or publishing destinations would require code changes. Figure 2 shows the VSMEx architecture and its overall workflow.

The tool operates in two main phases: *collection* and *selection*. During the collection phase, the tool's *crawling module* (or *the crawler* – `crawler.py`) gathers metadata about all new extensions from the VS Code Marketplace and downloads the corresponding VSIX files. During the selection phase, the tool's *selection module* (or *the selector* – `vsmex.py`) filters this corpus and retains only extensions that appear in the removed and malicious lists, producing the final dataset. Both modules share a common configuration file (`config.py`) that stores runtime parameters such as API endpoints, storage settings, and scheduling intervals.

Specifically, during step ① of the collection phase, the crawler queries the official VS Code Marketplace REST API to retrieve metadata and the corresponding VSIX[7] package download URLs. During the initial run, we captured all extensions available on the Marketplace. In subsequent runs, the crawler queries the Marketplace API with results sorted by the last update date, requesting only newly published or updated extensions, thus minimizing API calls and data transfer. At step ②, the crawler downloads the packages that are not yet present in our cache and stores them directly in ABS. At step ③, it updates a master metadata file (also stored in ABS) by appending new or modified entries and maintaining state and logs for incremental and fault-tolerant operation. These steps are necessary because once an extension appears in the malicious or removed lists, it is no longer available for download from the Marketplace. Therefore, we maintain a local replica of the Marketplace, which enables us to retrospectively retrieve both metadata and VSIX packages of malicious extensions. This approach is common in software ecosystem research [3], where local copies are kept to ensure continuity in case of content removal.

The selection phase builds upon this backend and proceeds through the remaining steps. At step ④, the selection module (`vsmex.py`) queries two sources, the *malicious* (`marketplace.json`) and the *removed* (`RemovedPackages.md`) lists, and identifies new extensions to add to our dataset. At step ⑤, for each newly identified extension, the selector checks whether the corresponding VSIX package exists in our cache. If available, the selector retrieves the extension's VSIX file and its associated properties from the master metadata file, extracts additional relevant features, and creates a revised metadata entry. Finally, at step ⑥, it appends one entry per captured version to the dataset description file (`vsmex_metadata.csv`) and updates the flagged extensions file (`msft_vscode_flagged_extensions.csv`) that consolidates information from both malicious and removed lists (see Section 3.2 for the detailed description of the entries). These updated files together with the retrieved VSIX packages are committed to the VSMEx dataset repositories on GitHub using the GitHub API.

The crawler runs twelve times per day (every two hours) to capture activity across global time zones and to provide sufficient coverage to detect newly published or updated extensions before removal, without introducing unnecessary API load. Crawling more frequently improves coverage but increases API load, while longer intervals reduce detail and risk missing extensions that are quickly removed. We aimed for a balance between stability and completeness by choosing moderate collection intervals, prioritizing consistent long-term operation. This makes the system reliable and sustainable but might introduce slight delays between Marketplace changes and their appearance in the dataset. The selection module runs twice per day, matching the typical update frequency we observed for the removed and malicious lists.

The tool relies entirely on public endpoints used by the VS Code client to fetch and update extensions, as well as the public `RemovedPackages.md` file hosted in Microsoft's GitHub repository.

[4] https://raw.githubusercontent.com/microsoft/vsmarketplace/main/RemovedPackages.md

[5] https://azure.microsoft.com/en-us/products/storage/blobs

[6] https://docs.github.com/en/rest

[7] The container file format for VS Code extensions.

If Microsoft were to restrict or revoke access to these sources, continuous collection would be affected. However, both interfaces are essential for the Marketplace's operation and transparency, making such a change unlikely. The Marketplace API has remained structurally stable since at least 2021 [31, 12, 33, 34]. Any major change would require coordinated updates to the VS Code client itself, making sudden removal unlikely. Should the structure or location of these lists change, minor tool adjustments would be sufficient to restore compatibility. Importantly, the crawler continuously maintains a cached replica of extension packages and metadata; even if Microsoft modifies its public lists, previously collected artifacts are retained and remain accessible retrospectively. Note that extensions removed very shortly after publication may still be missed if they disappear between two consecutive crawl runs; while increasing the crawl frequency significantly reduced the miss rate in practice, fast removals cannot be fully eliminated.

3.2 Dataset Description Schema

Data is collected and organized according to the schema defined in Table 2 (see Appendix A), which outlines the retained metadata fields and their descriptions, ensuring that the dataset remains consistent and easy to maintain as new records are added.

There are two fields that are particularly relevant for machine learning research: *classification label* and *flags*. Each extension is associated with a classification label (`msft_classification_type`) defined by Microsoft, i.e., the reason why it has been flagged or removed. These labels provide insight into the intent and nature of the detected violation, such as whether it involved malicious code, deceptive behavior, or policy abuse. They are helpful for researchers who aim to develop ML-based classifiers targeting specific categories of unwanted extensions. For entries originating from the *malicious list*, Microsoft does not specify a classification label, so we assign the label *Malicious* by default. Table 3 in Appendix A lists all classification labels with their descriptions.

The *flags* field represents each extension's publication and visibility state as reported by the Marketplace [22]. The values of this field are helpful for understanding changes in an extension's publication state and availability and for distinguishing between unpublished, removed, or temporarily hidden extensions. Table 4 in Appendix A lists all flag values and provides their descriptions.

4 Empirical Results

We ran VSMEx continuously for more than three months, from September 3 to December 23, 2025. In the remainder of this section, we report our findings obtained over this observation period.

4.1 Analysis of the Malicious and Removed Lists

We obtained the snapshots of both the malicious and removed lists as of December 23, 2025, and analyzed their contents, cleaning the information where necessary. On that date, the malicious list contained 252 entries; however, upon closer inspection, only 249 of them were unique. Specifically, the extensions with identifiers `gitkraken-dev.gitlens-pro`, `Glenn-marks1990.live-sass-compiler`, and `platformio-dev.platform-io` appeared twice. The removed list contained information on 402 unique extensions. We observed that the "Type" column (indicating the reason for removal) included two categories – "Typo Squatting" and "Typo-squatting" – that refer to the same cause. We merged these categories into a single class.

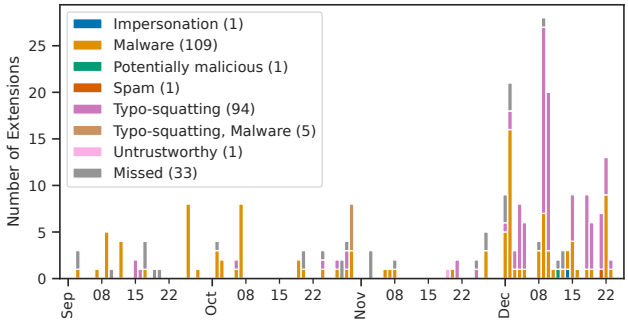


Figure 3: Number of extensions captured/missed by VSMEx (the numbers in the parentheses represent the total number of extensions belonging to the corresponding category).

Across both lists, there are 576 unique extensions, of which 75 appear in both. Among these overlapping extensions, 72 are labeled as "Malware" and 3 as "Typo-squatting" in the removed list. Microsoft does not provide further detail on the criteria governing inclusion of extensions in each list; however, it is clear that the removed list records Marketplace takedowns for a broader set of policy reasons, while the malicious list identifies the extensions that should be additionally blocked in VS Code, forcing the uninstallation of existing instances and preventing future installations.

The removed list contains 245 extensions that were taken down during the VSMEx collection period. Unfortunately, the malicious list does not report when a particular extension was removed; however, we store the date when a particular entry appeared in it (see `checked_date` field).

4.2 Collection Tool Evaluation

Effectiveness. During the collection period, VSMEx successfully gathered 214 extensions: 212 of them appear in the removed list and 2 in the malicious list[8]. Note that 14 extensions appear in both lists; however, we prioritize information from the removed list; therefore, the corresponding extensions in our dataset are labeled as originating from it. Thus, during the collection period, VSMEx missed 33 (13.47 %) extensions from the removed list. Figure 3 illustrates the number of captured (broken down by removal reason) and missed extensions per day, with the numbers in parentheses indicating the total number of extensions for each category. As shown in the figure, the majority of extensions were removed due to either malicious functionality or typo-squatting. Interestingly, starting from early December 2025, we observe a clear increase in the number of malicious extensions appearing in the removed list. This trend suggests a surge in the detection or reporting of malicious extension activity, potentially reflecting increased attacker activity, intensified enforcement efforts, or improved detection mechanisms within the Marketplace.

[8]Note that information for these 2 extensions is also present in the removed list; however, their removal dates precede the start of our collection period.

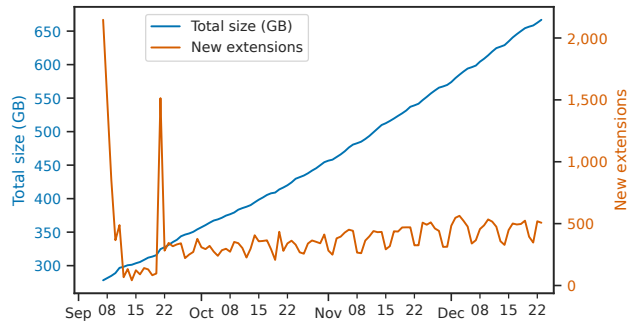


Figure 4: New extensions per day and the dataset total size.

The figure also indicates that most misses occurred during the first months, when we were tuning the execution frequency of the VSMEx crawling module. Initially, we ran VSMEx once per day, then increased the frequency to six per day, and eventually to twelve times per day (every two hours). It was raised after we observed instances in which extensions appeared in the malicious or removed lists but were absent from our captured instances.

Efficiency. Before starting the regular collection operation runs, we took a full snapshot of all extensions in the Marketplace on September 3. This snapshot contained 99,566 extensions with a total size of 268.62 GB. From September 7, we started regular extension updates in the cache, capturing only newly published extensions or those whose version number changed to keep the process efficient and optimize storage use. Over the collection period, the cache grew to include 140,934 extensions totaling 666.89 GB, containing nearly all extension versions collected over this period. Figure 4 shows the total cache size (blue line) and the number of new extensions collected per day (red line) after starting regular updates.

After the tool configuration stabilized at the end of September 2025, VSMEx captures approximately 400 new extensions per day (see orange line in Figure 4). The time series exhibits a clear weekly pattern and an overall upward trend, indicating that the VS Code Marketplace continues to grow and is not yet saturated.

4.3 Dataset Characterization

The complete dataset comprises 214 extensions, occupying 993.61 MB. Of these, 2 come from the malicious list. Of the remaining 212, which appear in the removed list, 109 are labeled as `Malware`, 94 as `Typo-squatting`, and 5 fall into both categories. The remaining 4 extensions comprise one each from the `Impersonation`, `Potentially malicious`, `Spam`, and `Untrustworthy` categories.

These extensions were published by 159 distinct publishers, indicating that some accounts were used to release multiple malicious extensions. For example, two publishers each published 7 such extensions. This observation suggests that a publisher's prior association with malicious extensions could serve as a useful signal for identifying additional suspicious candidates.

The earliest extension in the VSMEx dataset was published on May 31, 2017, while the most recent 13 were released on December 22, 2025 – one day before we concluded the dataset collection. Table 1 reports other Marketplace statistics, offering two noteworthy

Table 1: Malicious Extension Statistics

Parameter	# of N/A	Min	Average	Median	Max
Installation Count	56	1.00	51,080.07	32.00	2,920,720.00
Rating	167	1.00	4.85	5.00	5.00
Rating Count	167	1.00	1.72	1.00	14.00
Package Size	0	0.01	4.64	0.59	124.45

observations. First, the installation counts are relatively high (average: 51,080.07, maximum: 2,920,720), suggesting that adversaries may artificially inflate download numbers. Note that Marketplace statistics are collected at the time an extension is added to our cache rather than when it is removed from the Marketplace.[9] Therefore, these values represent a lower bound. This collection strategy also explains the substantial number of N/A values. Second, the majority of extensions have very high ratings (average value is 4.85), although the number of raters is remarkably low: the maximum is only 14, and 167 extensions (see "# of N/A") have no ratings at all. In our opinion, rating manipulation is more difficult to automate, leading the adversaries to occasionally rate their own extensions manually. This also indicates that the ratio between the number of ratings and installations could serve as a feature for ML.

The two most common extension versions in our dataset are `1.0.0`, observed in 35 extensions, and `0.0.1`, observed in 30 extensions. The next most frequent version, `1.0.2`, appears considerably less often, in only 4 extensions. The prevalence of versions `0.0.1` and `1.0.0` can be attributed to Marketplace tooling defaults and common developer practices. Specifically, VS Code's extension generator defaults to version `0.0.1` for new projects. Conversely, to appear mature and trustworthy, developers of malicious extensions may choose to publish their initial release directly as `1.0.0`. Given the typically short-lived nature of malicious extensions, this behavior results in the prevalence of these two versions in our dataset.

We also analyzed the target VS Code versions specified by the extensions (see the `engines_vscode` field in Table 2). In total, we identified 75 distinct target version patterns. Most commonly, extensions in our dataset targeted VS Code versions above 1.75 (`'^1.75.0'`) in 27 cases, above 1.70 (`'^1.70.0'`) in 19 cases, and above 1.85 (`'^1.85.0'`) in 10 cases. The next most frequent target, `'^1.105.0'`, appeared in 9 extensions, focusing on relatively recent VS Code versions released after October 9, 2025. However, given that most developers regularly update their VS Code installations, targeting recent versions is a reasonable choice. This observation is further supported by the fact that the wildcard pattern `'*'`, which targets all VS Code versions and could be expected in malicious extensions to maximize exposure, appears in only 1 extension.

Regarding the flags, 186 (87 %) of the extensions were marked as `validated` and `public`, meaning that they were approved by the Marketplace and were made publicly available. Another 21 (10 %) had an additional `unpublished` flag, which indicates that the extension was withdrawn either by Microsoft or the publisher.Another 2 `unpublished` extensions were also `locked`, meaning Microsoft suspended or restricted the extensions or the associated publishers' accounts. Finally, 3 extensions included the `preview` and 2 had the

[9] Obtaining the most recent data would require frequent crawling of the Marketplace, imposing unnecessary load on it; therefore, we refrain from doing so.

`trial` flag in addition to the `validated` and `public` flags. These flags reflect publisher-defined designations: `preview` is set via the `preview` field, while `trial` is set via the `pricing` field.

5 Related Work

To our knowledge, there is no tool that systematically collects malicious VS Code extensions and publishes them. Few initiatives in other ecosystems have pursued similar goals. Academic efforts like the Backstabbers Knife Collection (BKC) dataset [24] provide curated sets of malicious open-source packages, but the VSIX subset is limited to 30 entries. MalOSS [7] also identifies malicious packages across open-source ecosystems, offering partial datasets. Additionally, some commercial platforms [28, 16, 29] crawl the Marketplace to scan and classify extensions, albeit in proprietary ways.

We drew inspiration from previous work such as AndroZoo [3], which continuously collects Android applications for malware research. Other notable efforts include MalwareBench [37], the Android cryptojacking dataset [6], and the PyPI-focused study by Guo et al. [11], which released curated datasets of malicious artifacts, though these are snapshot-based rather than continuously updated. Building on these ideas, we focus on the VS Code ecosystem and present the first continuously updated dataset of extensions flagged or removed by Microsoft and made available to researchers.

6 Conclusion

In this paper, we introduced a fully automated system for collecting malicious and removed VS Code extensions from Microsoft-maintained public sources. The system continuously tracks Marketplace changes, maintains a local cache of extension packages and metadata, and cross-references this cache against Microsoft's malicious and removed extension lists. When a match is found, the extension is added to a dataset that aids security research and longitudinal analysis within the VS Code ecosystem.

While *VSMEx* is designed to operate continuously, the analysis presented in this paper is based on a fixed snapshot of 214 extensions collected between September 3 and December 23, 2025. We release the full source code of the tool and publicly share the lists and metadata of the collected extensions, while restricting access to the extension packages themselves to verified academic researchers.

References

[1] Ori Abramovsky. 2023. VSCode security: malicious extensions detected- more than 45,000 downloads- PII exposed, and backdoors enabled. https://blog.checkpoint.com/securing-the-cloud/malicious-vscode-extensions-with-more-than-45k-downloads-steal-pii-and-enable-backdoors/. Check Point, (May 2023).

[2] Lawrence Abrams. 2025. Microsoft warns of new xcsset macos malware variant targeting xcode devs. https://www.bleepingcomputer.com/news/security/microsoft-warns-of-new-xcsset-macos-malware-variant-targeting-xcode-devs/. BleepingComputer, (Sept. 2025).

[3] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. Androzoo: collecting millions of android apps for the research community. In *Proc. of MSR*. ACM, 468–471. doi:10.1145/2901739.2903508.

[4] Tristan Bilot, Nour El Madhoun, Khaldoun Al Agha, and Anis Zouaoui. 2024. A survey on malware detection with graph representation learning. *ACM Computing Surveys*, 56, 11, 1–36.

[5] CISA. 2025. Widespread supply chain compromise impacting npm ecosystem. https://www.cisa.gov/news-events/alerts/2025/09/23/widespread-supply-chain-compromise-impacting-npm-ecosystem. U.S. Department of Homeland Security, (Sept. 2025).

[6] Stanislav Dashevskyi, Yury Zhauniarovich, Olga Gadyatskaya, Aleksandr Pilgun, and Hamza Ouhssain. 2020. Dissecting android cryptocurrency miners. In *Proc. of CODASPY*, 191–202. doi:10.1145/3374664.3375724.

[7] Ruian Duan, Omar Alrawi, Ranjita Pai Kasturi, Ryan Elder, Brendan Saltaformaggio, and Wenke Lee. 2021. Towards measuring supply chain attacks on package managers for interpreted languages. In *Proc. of NDSS*. (Feb. 2021).

[8] Eclipse Foundation. 2025. Open vsx registry v0.26.0: extensions for vs code compatible editors. https://open-vsx.org/. (2025).

[9] Matthew G Gaber, Mohiuddin Ahmed, and Helge Janicke. 2024. Malware detection with artificial intelligence: A systematic literature review. *ACM Computing Surveys*, 56, 6, 1–33.

[10] Alejandro Guerra-Manzanares, Hayretdin Bahsi, and Sven Nõmm. 2021. Kronodroid: Time-based hybrid-featured dataset for effective Android malware detection and characterization. *Computers & Security*, 110, 102399.

[11] Wenbo Guo, Zhengzi Xu, Chengwei Liu, Cheng Huang, Yong Fang, and Yang Liu. 2024. An empirical study of malicious code in pypi ecosystem. In *Proc. of ASE*. IEEE Press, 166–177. doi:10.1109/ASE56229.2023.00135.

[12] Jossef Harush Kadouri. 2023. Vs code marketplace api: list all extensions and download artifacts. GitHub Gist. (2023). https://gist.github.com/jossef/8d7681ac0c7fd28e93147aa5044bc129.

[13] Sean Iyer. 2025. Security and trust in visual studio marketplace. https://developer.microsoft.com/blog/security-and-trust-in-visual-studio-marketplace. (2025).

[14] Jetbrains. 2025. Intellij idea. https://www.jetbrains.com/idea/. (2025).

[15] Robert J Joyce, Gideon Miller, Phil Roth, Richard Zak, Elliott Zaresky-Williams, Hyrum Anderson, Edward Raff, and James Holt. 2025. EMBER2024 - A benchmark dataset for holistic evaluation of malware classifiers. In *Proc. of KDD*, 5516–5526.

[16] Koi Security. 2025. Koidex. https://dex.koi.security/. (2025).

[17] Xiang Ling et al. 2023. Adversarial attacks against Windows PE malware detection: a survey of the state-of-the-art. *Computers & Security*, 128, 103134.

[18] Microsoft. 2025. Azure-core, version 1.35.1, released september 12, 2025. https://pypi.org/project/azure-core/1.35.1/. (2025).

[19] Microsoft. 2025. Azure-storage-blob, version 12.26.0, released july 16, 2025. https://pypi.org/project/azure-storage-blob/12.26.0/. (2025).

[20] Microsoft. 2025. Extension manifest reference. https://code.visualstudio.com/api/references/extension-manifest. (2025).

[21] Microsoft. 2025. Extension runtime security. https://code.visualstudio.com/docs/configure/extensions/extension-runtime-security. (2025).

[22] Microsoft. 2026. PublishedExtensionFlags enum. (2026). https://learn.microsoft.com/en-us/javascript/api/azure-devops-extension-api/publishedextensionflags.

[23] Microsoft. 2025. Visual studio code. https://code.visualstudio.com/. (2025).

[24] Marc Ohm, Henrik Plate, Arnold Sykosch, and Michael Meier. 2020. Backstabber's knife collection: a review of open source software supply chain attacks. In *Proc. of DIMVA*. Springer.

[25] Junyang Qiu, Jun Zhang, Wei Luo, Lei Pan, Surya Nepal, and Yang Xiang. 2020. A survey of Android malware detection with deep neural models. *ACM Computing Surveys (CSUR)*, 53, 6, 1–36.

[26] Reitz, Kenneth and contributors. 2025. Requests, version 2.32.5, released august 18, 2025. https://pypi.org/project/requests/. (2025).

[27] ReversingLabs. 2025. The 2025 software supply chain security report. https://www.reversinglabs.com/sscs-report. ReversingLabs, (2025).

[28] ReversingLabs Threat Research. 2025. Spectra assure community. https://secure.software/. (2025).

[29] Seraphic Security. 2025. Browsertotal. https://browsertotal.com/. (2025).

[30] Amanda Silver. 2025. Celebrating 50 million developers: the journey of Visual Studio and Visual Studio Code. https://devblogs.microsoft.com/blog/celebrating-50-million-developers-the-journey-of-visual-studio-and-visual-studio-code. Microsoft, (May 2025).

[31] ssbarnea. 2021. Document marketplace api. GitHub Issue. (2021). https://github.com/microsoft/vsmarketplace/issues/238.

[32] Stack Overflow. 2025. Stack Overflow Developer Survey 2025. https://survey.stackoverflow.co/2025. (2025).

[33] Stack Overflow Contributors. 2022. How to programmatically retrieve the list of visual studio code extensions on the marketplace? Stack Overflow. (2022). https://stackoverflow.com/questions/71742538.

[34] Stack Overflow Contributors. 2021. How to use contains in visual studio code extensions api? Stack Overflow. (2021). https://stackoverflow.com/questions/66009233.

[35] Bill Toulas. 2024. Malicious vscode extensions with millions of installs discovered. https://www.bleepingcomputer.com/news/security/malicious-vscode-extensions-with-millions-of-installs-discovered/. (June 2024).

[36] Colson Wilhoit and Terrance DeJesus. 2025. Bit ByBit - emulation of the DPRK's largest cryptocurrency heist. https://www.elastic.co/security-labs/bit-bybit. Elastic Security Labs, (May 2025).

[37] Nusrat Zahan, Philipp Burckhardt, Mikola Lysenko, Feross Aboukhadijeh, and Laurie Williams. 2024. MalwareBench: malware samples are not enough. In *Proc. of MSR* (MSR '24). ACM, 728–732. doi:10.1145/3643991.3644883.

A Dataset Schema

Table 2: Metadata Fields Description

Field	Description
Fields in `vsmex_metadata.csv`	
captured_date	Date of collection by the tool
source	Feed source (*malicious_list* or *removed_list*)
msft_classification_type	Microsoft classification type
extension_identifier	Unique identifier of the extension
publisher_name	Extension publisher name
version	Extension version number
artifact	VSIX package filename
sha256	SHA-256 hash of the VSIX package
size_mb	VSIX package size (MB)
published_date	Original publication date
last_updated_date	Date of the last update
verified_publisher	Publisher verification status (*true/false*)
installation_count	Number of installs at collection time
average_rating	Average user rating at collection time
rating_count	Total number of user ratings
categories[§]	Categories defined in the extension manifest
repository_url	Source code repository URL (if available)
flags	Marketplace flags (*validated*, *public*, etc.)
engines_vscode	VS Code version requirement from `package.json`
exists_in_dataset	Provenance of the extension package in the dataset
Fields in `msft_vscode_flagged_extensions.csv`	
source	Feed source (*malicious_list* or *removed_list*)
checked_date	Date when the feed was checked
extension_identifier	Unique identifier of the extension
msft_classification_type	Microsoft classification type
msft_removed_date	Removal date reported by Microsoft
captured	Whether the extension was captured (*yes/no*)
version_count	Number of captured versions
latest_version	Latest captured version number
capture_date	Date when the extension was captured

[§] *Category definitions from the VS Code Extension Manifest [20]*

Table 3: Taxonomy of the `msft_classification_type` Field

Label	Description
Malicious	Default label assigned to extensions from the *malicious list* where Microsoft does not specify a classification.
Copyright violation	Uses copyrighted or trademarked content without proper authorization.
Impersonation	Mimics another publisher or extension to deceive users.
Potentially malicious	Contains suspicious or obfuscated code that resembles malicious behavior.
Malware	Intentionally designed to infiltrate or harm user systems.
Spam	Deceptive or repetitive content intended to mislead or annoy users.
Typo-squatting	Mimics a popular extension's name to exploit search confusion.
Untrustworthy	Indicates publisher actions that undermine Marketplace integrity or trust.

Table 4: Taxonomy of the `flags` Field

Flag	Description
validated	Passed Marketplace validation and policy checks.
public	Visible and installable by users through the Marketplace.
unpublished	Withdrawn from the Marketplace by the publisher or Microsoft.
locked	Further updates and acquisitions are not allowed by Microsoft.
trial	The extension is marked as a trial offering by the publisher.
preview	The extension is marked as a preview release by the publisher.

Open Science Statement

In the spirit of open science, we release the full source code of VSMEx together with the metadata of the malicious extensions collected by the tool. The source code, the collection scripts, and the consolidated list of flagged extensions are publicly available at the *vsmex* repository,[10] covering the 214 extensions analyzed in this paper as well as extensions continuously collected beyond the observation period. These artifacts enable other researchers to inspect and validate the collection pipeline and to reproduce the analysis results reported in this paper. The detailed per-extension metadata and the VSIX packages themselves are maintained in a separate private repository (*vsmex-dataset*[11]), restricted to verified academic researchers with institutional affiliation. Access can be requested by contacting the authors via institutional email.

Ethics Statement

This work collects a dataset of malicious VS Code extensions based solely on Microsoft's publicly available malicious and removed extension lists. The study does not involve interaction with end users, developer accounts, or private Marketplace data, and no personal or identifying information is collected. We do not independently label extensions as malicious; instead, we rely exclusively on Microsoft's classifications.

Because the dataset includes malicious code, we carefully consider the risk of misuse. We believe that sharing such datasets with approved researchers provides greater benefit than harm by enabling the development and evaluation of detection techniques. This position is consistent with prior work that has shared malicious extension datasets under controlled conditions, such as the Backstabbers Knife Collection (BKC) [24] or AndroZoo [3].

[10] https://github.com/kalachkar/vsmex
[11] https://github.com/kalachkar/vsmex-dataset