Classification of Software Vulnerability Artifacts Using Public Internet Data

Leonardo Ambrus de Lima*, Estevao Rabello Ussler*, Miguel Angelo Santos Bicudo*,
Daniel Sadoc Menasche*, Anton Kocheturov†, Gaurav Srivastava†
*Institute of Computing, Federal University of Rio de Janeiro (UFRJ), †Siemens AG, Princeton, NJ, USA

***Abstract*—Artifacts associated with vulnerabilities, such as patches, exploits, and scanners, provide valuable insights in the context of network security. In particular, the network protocols used by scanners to identify vulnerabilities offer clues about their exploitation mechanisms and associated risks. In this work, we analyze network-related vulnerabilities using data from the NomiSec repository, with a special focus on scanners. For example, we observe that some artifacts indicate that exploitation occurs via HTTP, while others require direct socket interactions. Additionally, we perform clustering and visualization of these artifacts, identifying relationships between different categories. We find that certain artifact (directly taken from GitHub) are associated with the exploitation of network devices, such as firewalls, while others focus on protocol-specific vulnerabilities, such as SSL/TLS. These findings contribute to a better understanding of the vulnerability ecosystem and the improvement of mitigation strategies based on data automatically and periodically collected from GitHub.**

***Index Terms*—Software vulnerabilities, security artifacts, vulnerability classification, cybersecurity analytics.**

I. Introduction

The growing complexity and connectivity of software systems have significantly increased security vulnerabilities, posing substantial challenges to safeguarding systems and sensitive data [1]–[4]. This work proposes a comprehensive approach to classify, filter, and annotate artifacts related to software vulnerabilities. These artifacts include *patches* and *exploits*, as well as proof-of-concept codes (PoCs) and checkers, with particular emphasis on the latter, providing a broad overview of existing threats [5].

The source of artifacts chosen for this work, NomiSec, is a repository whose primary purpose is the automated collection of Proofs of Concept (PoCs) available on GitHub. This collection is carried out by searching all GitHub repositories that mention vulnerability identifiers, known as *Common Vulnerabilities and Exposures* (CVEs). The collected data is then organized into JSON files[1], where, in cases of multiple repositories referencing the same CVE, all are stored in the same JSON file. These files contain not only links to relevant repositories but also detailed descriptions, creation dates, updates, and inclusion dates in NomiSec, as well as information about the users responsible for the postings. They also include data related to repository visibility, associated discussions, and other relevant metrics. This systematic and comprehensive approach of NomiSec offers a valuable source of information for the information security community, enabling more detailed and well-founded analyses of software vulnerabilities and their possible implications.

In this context, our main contributions are:

Evaluation of the scope and relevance of sources like NomiSec: We identified that sources like NomiSec cover a significant portion of vulnerability data available on GitHub, but there is a *trade-off* between publication delays and data quality (Section III).

Identification of the nature and quality of data: We observed that some data published on NomiSec refer to artifacts that do not pose a threat, such as checkers. The latter, in turn, provide valuable information about the mechanisms that can be used to exploit vulnerabilities, such as HTTP versus explicit *sockets* (Sections IV and V).

Representation and visualization of data: We propose a methodology to represent and visualize artifact data related to vulnerabilities present in GitHub repositories, indicating that some of the identified clusters refer to the exploitation of network devices, such as firewalls, while others focus on the exploitation of specific protocols, such as SSL/TLS (Section VI).

Research hypotheses: Our work is guided by two main hypotheses. **Hypothesis 1** states that most of the artifacts classified as checkers are associated with network access vulnerabilities, indicating a prioritization of risks with broader attack surfaces. **Hypothesis 2** suggests that the reputation of a GitHub user influences the likelihood of their artifact being cited by curated databases like NVD or InTheWild, for example.

Takeaway message of the paper: Our study shows that classifying artifacts enables a differentiation of vulnerabilities with higher potential of exploitation and suggests that multiple data sources must be combined for effective risk assessment.

The remainder of this article is organized as follows. Section II presents related work. In Section III, we discuss the relationship between sources, indicating their complementarity. Section IV reports observations on artifacts, including proof-of-concept (PoC) and vulnerability checkers. Motivated by the relevance of the latter, we consider additional analyses of them in Section V, followed by their clustering and visualization in Section VI, and Section VII concludes.

[1]https://bit.ly/JSON_example

II. Related Work

Public repositories, such as GitHub, are rich sources for obtaining *exploits*, vulnerability scanners/checkers [6], and *patches* [7]. However, the quality and reliability of these artifacts vary considerably, requiring careful analysis.

In [3], [8] the authors analyze user reputation and stars in GitHub repositories. In [3], in particular, the authors examine the relationship between GitHub repository popularity and the presence of malicious artifacts, indicating that the number of stars is an unreliable indicator of artifact legitimacy. Our study builds upon these works by showing that user reputation, e.g., measured by number of stars, is correlated with the number of citations to a repository from different sources, serving in our dataset as a proxy for the maturity of a repository, as detailed in Section III-A.

Other studies explore techniques for classifying and clustering vulnerability artifacts. In [5], an approach is proposed to categorize vulnerabilities based on exploitation patterns, using machine learning to identify emerging trends. In this study, we apply a similar method by using clustering techniques to classify artifacts collected from the NomiSec repository. By focusing on the classification of artifacts as opposed to the classification of vulnerabilities, our work complements the vast literature on the latter.

The relationship between scanners and the network protocols used to detect vulnerabilities has also been investigated. In [9], the authors analyze how different protocols impact the way vulnerabilities are exploited and mitigated. Our research reinforces this perspective by demonstrating that the nature of the protocol influences both the exploitation and the effectiveness of detection tools.

Finally, initiatives such as EPSS (*Exploit Prediction Scoring System*) [10] estimate the likelihood of a vulnerability being exploited, based on public artifacts. However, there is no transparency regarding the semi-automatic differentiation between *checkers* and *exploits*. Our work highlights the importance of this distinction and proposes a methodology to classify them based on the structure of repositories and how they interact with vulnerabilities. This way, our work can serve as input to models such as EPSS.

In summary, we complement the existing literature by providing an analysis of the relationship between vulnerability artifacts and their respective network protocols, as well as presenting a methodology for the automatic categorization and visualization of artifact data collected from public sources.

III. Relationship Between Sources: Complementary or Replaceable?

There is a balance between characteristics of different sources of vulnerability data. NomiSec[2] is powered by bots that scan all of GitHub for proof-of-concept information. The NVD,[3] on the other hand, contains only curated information about vulnerabilities. The response time of the former for publishing new vulnerability information is naturally much shorter than the latter. In between, we have InTheWild,[4] a platform that, like NVD, also contains curated information about vulnerabilities, but with a much shorter update response time than NVD.

Inthewild.io is a source for artifacts related to software vulnerabilities. The central proposal of Inthewild.io is to catalog artifacts as well as to provide information about real security incidents that occurred "in the wild," that is, in production environments. This is achieved through the collection of data from various sources, including news feeds, security forums, and cybersecurity company reports. InTheWild uses 10 data sources, with GitHub being one of them. On the other hand, GitHub is the only source used by NomiSec.

Data until 2024. From the inception of InTheWild until January 2024, a total of 5,489 GitHub links were identified within its dataset, alongside links from other sources such as Apple, AwesomePoc, Chrome, CISA, Google, Metasploit, Microsoft, and the NVD. In comparison, 11,627 GitHub links were recorded in the NomiSec repository during the same period. This significant discrepancy led to a deeper investigation to analyze the intersection between the link sets from InTheWild and NomiSec.

It was observed that all 5489 GitHub links present in InTheWild were also in NomiSec. This finding led to the conclusion that the two repositories are complementary: InTheWild is more comprehensive due to the inclusion of more sources beyond GitHub, allowing for better detailing of collected vulnerabilities, while NomiSec offers an exclusive set of GitHub links. It is important to note that not all these links, which are exclusively present in NomiSec, are necessarily PoCs. Therefore, further investigation was necessary to assess their relevance, as we consider in the rest of this article.

Data from 2024. In 2024, the trend described above partially continued. In particular, NomiSec continued to publish references to many artifacts not cited by InTheWild. However, in the opposite direction, there were also instances of artifacts cited by InTheWild but not listed in NomiSec. Specifically, we found for CVE-2024-⋆ vulnerabilities a total of 719 common links between InTheWild and NomiSec. The number of links exclusive to NomiSec and InTheWild was 1,333 and 925, respectively.

Still focusing on CVE-2024-⋆ vulnerabilities, we analyzed GitHub users appearing in links from NomiSec, InTheWild, and NVD. Figure 1 presents a Venn diagram of users found in NomiSec, InTheWild, and NVD, based on data collected on January 9, 2025. It is interesting to highlight the complementary nature of the three databases: a substantial number of users contributed exclusively to one of the databases. However, not all contributions have the same quality.

Users present in NomiSec, but never cited by InTheWild and NVD, tend to raise more questions about their reputation. In particular, of the 1,277 users referenced by NomiSec, 652 (109+126+417) appeared either in NVD, InTheWild, or both.

[2]https://github.com/nomi-sec/PoC-in-GitHub

[3]https://nvd.nist.gov/

[4]https://inthewild.io/

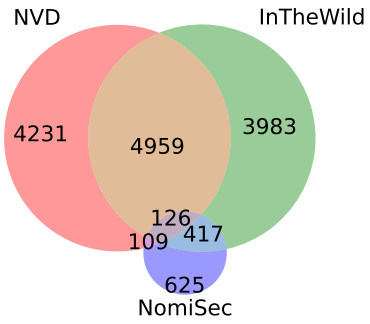

Fig. 1. Venn diagram of users in NomiSec, InTheWild, and NVD.

TABLE I
CVEs, ARTIFACTS, AND CHECKERS (THE TABLE ENTRIES CONTAIN CLICKABLE HYPERLINKS WITH DETAILS ABOUT EACH CELL)

		All years	
	CVE-2024-⋆	All CVEs	Checkers
CVEs analyzed	1,128	6,214	196
Unique GitHub links, i.e., repositories, (these are Nomisec unique artifacts)	2,263	13,727	441
Users referenced in NomiSec	1,367	7,975	363
also appearing in NVD or InTheWild	652	-	-
CVEs with more than one artifact	238	1,581	66
Valid checkers identified	67	442	
Empty or README-only repositories	0	19	
Checkers of vulns with attack vector 'local'	0	15	
CVEs linked to multiple clusters	2	12	

This suggests that these users have high credibility, considering that the data from NVD and InTheWild are curated. The remaining 625 users, who do not appear in either NVD or InTheWild, require a more detailed analysis to assess whether the artifacts they shared are functional or even malicious. Next, we present additional details about user reputation.

A. User Reputation

What is the relationship between user and repository reputation and the maturity of the artifacts they share? To answer this question, we analyzed GitHub users referenced from NomiSec. In particular, we assume that users cited only by NomiSec, and not yet cited by NVD or InTheWild, produce less mature artifacts than those who have also been cited by an alternative source. Given this hypothesis, we assessed whether users who are in the intersection of the three sets from the Venn diagram in Figure 1 can be distinguished by characteristics such as the number of stars awarded to them. Depending on the classification power of such characteristics, we can use them as partial indicators of the priority that should be allocated to the artifacts produced by these users.

The graphs in Figure 2 show the cumulative distribution (CDF) of reputation metrics (stars, followers, public repositories, and forks) for users who contributed to vulnerability repositories cited by NomiSec in 2024 CVEs. These users were classified into four groups according to their presence in NVD and InTheWild. To avoid distortions, three users with an exceptionally high number of repositories were removed when generating Figure 2(d): User gmh5225 (23,934 repositories), user CrackerCat (18,728 repositories), and user killvxk (11,600 repositories). The fourth highest, ahlfors, has 1,475 repositories. Users cited only by NomiSec exhibit lower metrics in the upper percentiles, indicating less visibility and impact. On the other hand, users mentioned in both NVD and InTheWild show significantly higher values across all metrics, suggesting greater reputation and influence. The number of stars and followers reflects recognition and exposure, while the quantity of public repositories and forks indicates higher productivity and relevance in the community.

Takeaway message. User reputation and contribution history are key factors in evaluating the reliability of artifacts. The predominance of highly visible users in the NVD and InTheWild sets suggests that these curated sources prioritize well-established contributors. Conversely, the presence of lesser-known users exclusively in NomiSec highlights the need for careful vetting to distinguish legitimate artifacts from scams or low-quality code, a challenge explored in the following sections.

Actionable implications. Automated sources like NomiSec play a crucial role in quickly identifying relevant artifacts on GitHub. However, to ensure reliability, these sources should be complemented by reputation-based scoring mechanisms or additional validation methods to assess artifact quality and legitimacy, e.g., using user reputatioon as indicated in this section.

TABLE II
COMPARISON OF ARTIFACTS FOUND IN NOMISEC AND GITHUB

CVE	NomiSec	GitHub
CVE-2021-44228	396	451
CVE-2021-4034	159	174
CVE-2019-0708	118	130
CVE-2021-41773	116	123
CVE-2022-0847	91	98
CVE-2018-6574	82	91
CVE-2022-30190	78	86

B. NomiSec Coverage in Relation to the Entire GitHub: Relevance and Delay

Next, we present additional details about the vulnerabilities with artifacts in NomiSec. In particular, we aim to evaluate the consistency and completeness of the data among the different sources considered, remembering that our main source, NomiSec, is a subset of GitHub.

Among the vulnerabilities cited by NomiSec, we found that a total of 1,498 vulnerabilities, each identified by a CVE, contain more than one artifact, ranging from 2 to 396 artifacts per vulnerability. Driven by this finding, we decided to conduct a search on GitHub for vulnerabilities with a significant number of artifacts. During this exercise, we identified that many of the links found on GitHub were not present in the NomiSec repository.

To provide a clearer view of this discrepancy, we present in Table II the CVEs with the highest number of links. We report the number of repository links found by NomiSec and the number present on GitHub.

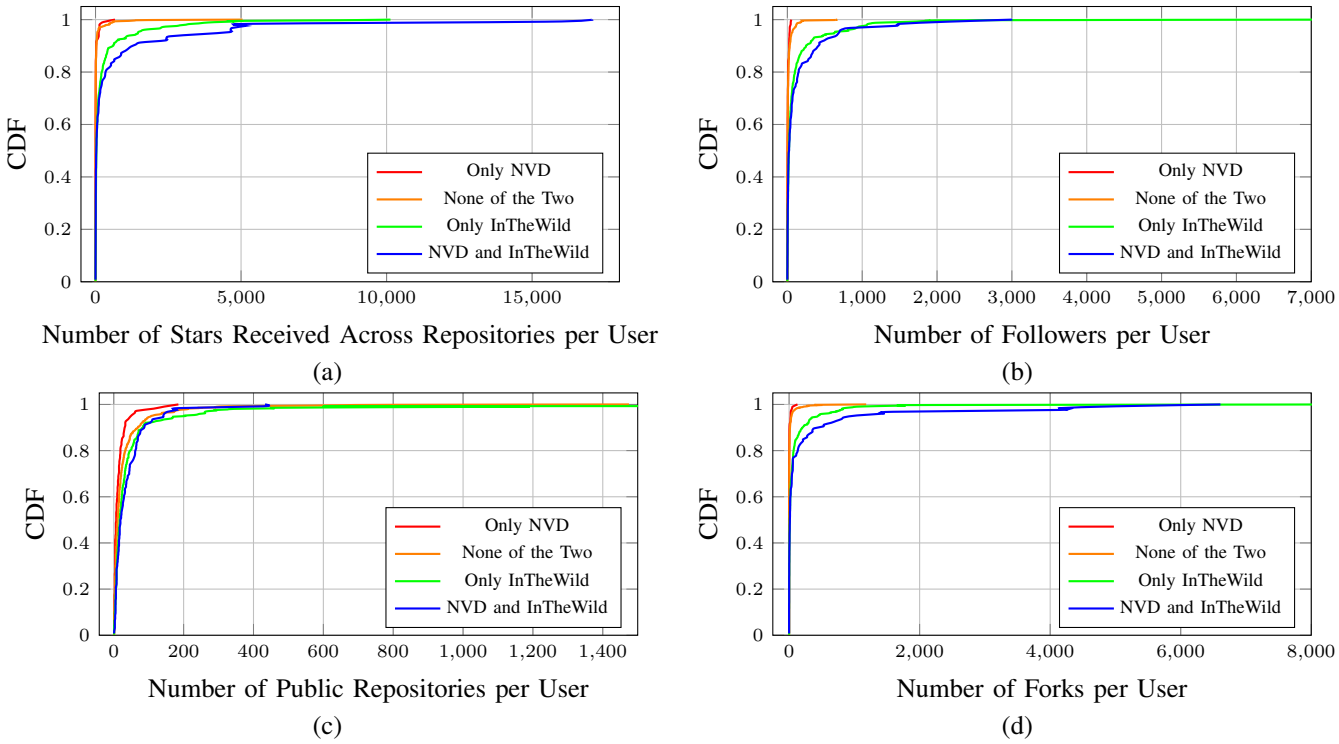

Fig. 2. Cumulative distribution functions (CDF) of user attributes.

Table II raises questions about the reasons why NomiSec does not achieve full coverage of GitHub, considering that this would theoretically be its fundamental premise. To explore this issue, we formulated some hypotheses. NomiSec may apply a filter to select only certain links, discarding invalid or malicious content. This is reinforced by the possibility that users report *issues* in the repository itself, leading to the removal of certain links (see Section V). Additionally, updates occur every 6 hours, which may cause delays in including new links, although the lag is small. Another possibility is that NomiSec faces limitations when using the GitHub API to collect links, preventing the complete retrieval of repositories related to CVEs. The above hypotheses provide a starting point for further investigations into the limitations and challenges faced by NomiSec in collecting and updating GitHub data. For the purposes of this work, considering the complexity of analyzing NomiSec in its entirety and the fact that it represents only a fraction of GitHub, we decided to focus on checkers. This choice is justified by the relevance of these artifacts in vulnerability assessment and the impacts their categorization can have on risk interpretation, as will be detailed in the following sections.

Takeaway message. Even automated sources like NomiSec still provide a limited perspective towards GitHub. To increase coverage, one needs to consider multilpe sources.

Actionable implications. Users relying on NomiSec or any other automated source for vulnerability tracking should be aware that 1) it provides a substantial amount of checkers for vulnerabilities, i.e., a number of the artifacts are not PoC, 2) it must be supplemented with broader GitHub searches to avoid missing critical artifacts.

IV. Identifying Checkers

When analyzing the artifacts shared on NomiSec, we conducted a detailed manual analysis using keywords such as "checker" and other related terms. This search revealed the presence of 451 artifacts that are supposedly vulnerability checkers. Each artifact identified as potentially related to *checkers* underwent a careful analysis, in which we examined the associated repositories and code to determine their nature and function.

The results of this analysis were recorded in a detailed report, published and made publicly available on GitHub.[5] In total, **442** artifacts were classified as positive, meaning exclusively checkers, while 9 were categorized as negative, encompassing proof-of-concept tools combined with checkers or just proof-of-concept tools.

A. Why Study Checkers?

Checkers are useful to scan systems against vulnerabilities. Below, we highlight additional reasons why studying *checkers* is relevant:

- **Risk analysis (EPSS):** Differentiating between *checkers* and other tools is essential to avoid misinterpreting risks. For instance, the existence of a *checker* does not necessarily imply the availability of a functional *exploit*; the *checker* might merely be a verification tool. This distinction is crucial for accurate risk assessment. However,

[5]https:\bit.ly/tagging_nomisec

tools like EPSS [6], [10] do not disclose the process used to distinguish between *checkers* and other artifacts.

- **Vulnerability mitigation via reverse engineering:** Understanding how *checkers* work allows the proposal of countermeasures that do not necessarily depend on *patches*. For example, identifying specific functionalities checked by a *checker* might enable disabling them as a preventive measure.[6]
- **Understanding vulnerabilities:** Analyzing how network protocols are used by *checkers* and how they relate to vulnerability categories (such as those described in CWE) provides valuable clues about the real-world exploitation potential of vulnerabilities. This understanding can guide more effective mitigation and prevention approaches.

B. Attack Vector: Local vs Adjacent vs Network

Our study analyzed a total of **451** checker repositories (each corresponding to a GitHub link), covering **198** distinct CVEs. As described above, certain CVEs have multiple artifacts. For example, CVE-2021-44228 (Log4Shell) corresponds to 40 checkers.

Among the various pieces of information provided by the NVD about vulnerabilities, through CVSS, we focused on the attack vector of each vulnerability. Our goal is to better understand the individual characteristics of each vulnerability and identify patterns in the vulnerabilities for which users developed checkers. The vast majority of CVEs with checkers—specifically, 182—have the Attack Vector (AV) classified as *Network*, and another 2 as *Adjacent*. We observed that, for these 184 vulnerabilities, the checkers use the network to identify them remotely. For 14 vulnerabilities, on the other hand, we found that local access to the vulnerability is necessary to exploit it, meaning it is not possible to exploit the vulnerability via the network unless remote shell access to the vulnerable device is obtained.

Takeaway message. Checker developers tend to prioritize vulnerabilities with broader impact, particularly those exploitable via the network. By analyzing the protocols used by checkers—typically simpler than full Proof-of-Concept (PoC) exploits—we can gain additional insights into the vulnerabilities themselves, as explored in the next section.

Actionable implications. Vulnerability scanning tools can integrate checkers from NomiSec to enhance their detection capabilities, with most checks being executable remotely. Automating this process could improve security assessments by leveraging detection methods in a timely manner.

C. Vulnerability Checker Packages and Network Protocols

We identified that some checkers are organized and distributed in packages. One of the packages we analyzed was the repository from the user 0xn0ne - weblogicScanner [7]. This package functions similarly to antivirus tools; however, instead of looking for viruses, such packages are designed to identify vulnerabilities in systems.

[6]For a concrete example, see https:\bit.ly/CVE_2023_36884

[7]https:\bit.ly/0xn0ne_weblogicserver

TABLE III
CLASSIFICATION OF CVES BY NETWORK PROTOCOL USED

Protocol	Associated CVEs
HTTP	CVE-2014-4210, CVE-2017-3506, CVE-2018-2894, CVE-2019-2618, CVE-2019-2725, CVE-2019-2729, CVE-2019-2888, CVE-2020-14750, CVE-2020-14882, CVE-2020-14883
T3	CVE-2016-0638, CVE-2017-3248, CVE-2016-3510, CVE-2017-10271, CVE-2018-2628, CVE-2018-2893, CVE-2018-3191, CVE-2018-3245, CVE-2018-3252, CVE-2019-2890, CVE-2020-2555, CVE-2020-2883
IIOP	CVE-2020-2551

A total of 23 vulnerabilities are checked by the package mentioned above. All vulnerabilities verified in the package are of the *remote code execution (RCE)* type, meaning they allow remote code execution on vulnerable systems. The vulnerabilities can be divided by the network protocol they use for verification. To determine the protocol, we identified a one-to-one relationship between the protocol cited by the NVD in the vulnerability description (i.e., HTTP, T3, or IIOP) and the protocol used by the *checker* to remotely verify its presence in a system.

- **Detection Method via HTTP:** Ten of these vulnerabilities use the *utils* library and the HTTP protocol. These tools attempt to access specific endpoints on vulnerable servers by sending HTTP requests to check for the presence of flaws.
- **Detection Method via T3:** The other twelve vulnerabilities use *sockets* for direct TCP/IP communication, employing the T3 protocol. This proprietary protocol is used by Oracle WebLogic Server for communications between servers and between client-server.
- **Detection Method via IIOP:** One *checker* uses the IIOP (Internet Inter-ORB Protocol) to detect the presence of the CVE-2020-2551 vulnerability. The method consists of sending a message encoded in the GIOP (General Inter-ORB Protocol) format, which is interpreted by servers using Oracle WebLogic. Upon receiving a response containing the GIOP signature, the scanner identifies potentially vulnerable servers.

Takeaway message. Although vulnerabilities are presented in isolation in databases like the NVD, in practice, much can be learned by analyzing similar vulnerabilities. In this section, in particular, we indicated that 23 vulnerabilities affecting Oracle products can be classified based on the network protocol used to exploit them, using data from *checkers* collected from NomiSec and cross-referenced with the NVD.

Actionable implications. Checkers can serve as a valuable resource for enhancing deep packet inspection (DPI) filters by offering detailed insights into how vulnerabilities manifest at the network protocol level. By incorporating protocol-specific detection methods from checkers, security tools can more effectively identify and mitigate threats in real time. This approach enables proactive defense mechanisms that go beyond static vulnerability databases, improving network

security monitoring and response capabilities.

V. NomiSec: The Good, The Bad and The Ugly

In this section, we present case examples where NomiSec has aided in risk analysis associated with vulnerabilities by providing information not available in the NVD (Section V-A), others where NomiSec may have more drastic consequences (Section V-B), and finally those where NomiSec may hinder if not carefully evaluated (Section V-C).

A. Positive: DNS Zone Transfer (CVE-1999-0532)

In this section, we illustrate an example of a critical vulnerability for which the NVD has virtually no information. This is CVE-1999-0532. This vulnerability addresses a flaw in DNS servers that allows zone transfer, exposing the network's DNS records. Despite being frequently used as an initial stage in attacks, this vulnerability is not well documented in the NVD.[8] However, its high Exploit Prediction Scoring System (EPSS) [10], currently over 97%, reflects its practical relevance and ease of exploitation, making it a strategic tool for attackers. NomiSec provides artifacts for this vulnerability.[9] Other data sources, such as Metasploit and Sploitus,[10] also offer artifacts.

Takeaway message. NomiSec stands out for the speed with which artifacts are collected, serving as a valuable source complementary to the NVD.

Actionable implications. Security analysts should leverage multiple sources, incluing NomiSec, Metasploit, and Sploitus, to supplement NVD data, particularly for under-documented but actively exploited vulnerabilities.

B. Negative: Scams and Malicious Code

Since NomiSec is itself a GitHub repository, it allows the creation of *issues*, which are discussion topics about the repository itself. Among the *issues* present in NomiSec, there are concerns raised about scams and malicious code in the links collected by NomiSec. An example is this *issue* [11] where the title points to the existence of malicious code: *"FAKE POC IN YOUR REPO SPREADING MALWARE"*, accompanied by details about the malware in its description.

It is worth noting that security researchers often use so-called *Honey PoCs* to understand hacker behavior. Some of these artifacts resemble scams as they collect data from their users.[12] An example of a *Honey PoC* is in the *issue* titled *CVE-2020-1350: there are scam links*, which points to the repository ZephrFish/CVE-2020-1350 [13]. This is not a scam but rather a *Honey PoC*.[14] To avoid misinterpretations, this particular repository was renamed to *CVE-2020-1350_HoneyPoC* to clarify its purpose, but not all repositories undergo such adjustments. For the purposes of this work, we removed the Honey PoCs that we were able to identify.

Takeaway message. While NomiSec is a useful repository of vulnerability artifacts, it also collects potentially harmful content, including fake PoCs, malware, and deceptive repositories. Misinterpretations, such as mistaking Honey PoCs for scams, highlight the need for caution when using these resources.

Actionable implications. Security teams should vet artifacts before use, verifying legitimacy through user reports, repository histories, and external validation. Automated filtering and tagging systems could help distinguish legitimate PoCs from deceptive or malicious artifacts, reducing the risk of misinformation and malware exposure.

C. Problematic: Empty Repositories or Only README

During our research, we identified a total of 19 repositories considered ineffective for the analysis of *checkers*. Specifically, we observed that 15 repositories contained only a README file, with no code or related functionality; 3 repositories were completely empty; and 1 repository was deleted by the owner before a more in-depth analysis could be conducted. These repositories were cataloged and documented in our GitHub repository for future reference and consultation: Tagging Nomisec [15] . Although such repositories may seem harmless, they are problematic, given that tools like EPSS[16] use the number of artifacts available for a particular vulnerability to determine its risk. Our analyses indicate that it is essential to carefully evaluate artifacts. Clearly, the examples presented in this section are extreme cases of ineffective artifacts. However, we foresee that there are numerous others that do not work or would require effort to become functional.

Takeaway message. A number of artifacts indexed by NomiSec lack functional code, potentially skewing vulnerability risk assessments. Since tools like EPSS consider the number of available artifacts when estimating risk, the presence of non-functional repositories can distort security metrics.

Actionable implications. Researchers and security practitioners should manually inspect or develop automated methods to filter out ineffective artifacts.

VI. Representing, Organizing, Visualizing and Interpreting Artifacts

How can we extract structured and useful information from artifacts available on platforms like NomiSec? While NomiSec can quickly provide pointers to numerous artifacts related to vulnerabilities, the collection process is automated via *bots*. In NomiSec's case, these *bots* simply distribute artifacts into folders based on the year the corresponding vulnerabilities were disclosed. Given the number of vulnerabilities published annually, it becomes imperative to adopt more efficient methods for organizing and classifying this data. To this end, in this section, we explore the use of clustering and visualization techniques.

[8] According to the NVD, it presents a severity of 0 as measured by CVSS, which is likely an error [11].

[9] https:\bit.ly/CVE_1999_0532_2 and https:\bit.ly/CVE_1999_0532_1

[10] See https:\bit.ly/sploitus

[11] https:\bit.ly/Nomisec_issues

[12] Canary Token: https://github.com/ZephrFish/AutoHoneyPoC

[13] https://github.com/ZephrFish/CVE-2020-1350

[14] The list of *issues* can be found at: https:\bit.ly/All_nomisec_issues

[15] https:\bit.ly/tagging_nomisec

[16] https://www.first.org/epss/

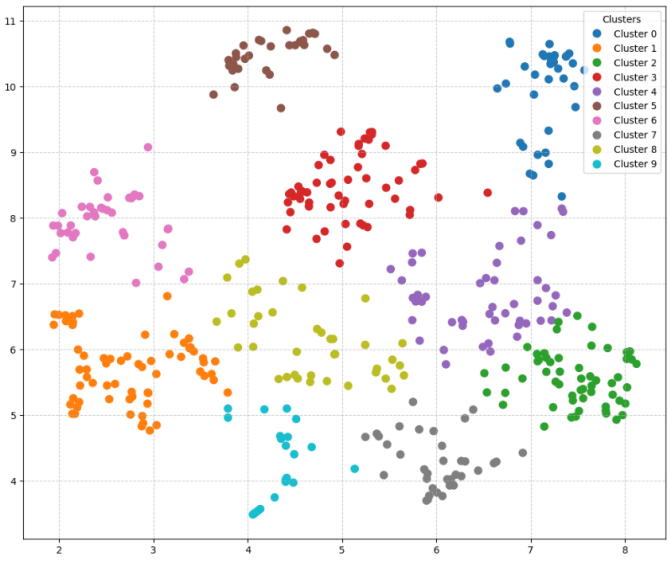

Fig. 3. Clusters of artifacts.

A. Methodology for Representing, Organizing, and Visualizing Artifacts

Initially, JSON files obtained from NomiSec containing information about the artifacts are parsed, and relevant text from each document is extracted. Specifically, for each artifact available on NomiSec, two fields are important for our analysis: 'html_url' and 'description', which contain the link to the artifact's GitHub repository and its corresponding description.

We then proceed with the representation, clustering, and visualization of the data. **Representation:** The extracted texts are transformed into numerical vectors using Term Frequency-Inverse Document Frequency (TF-IDF). During this step, common English *stopwords* are removed, following standard practices. **Clustering:** The resulting TF-IDF vectors are then processed using the K-Means algorithm to cluster the documents. In our study, we considered ten clusters. **Visualization:** To visualize the clustering results, we used UMAP (Uniform Manifold Approximation and Projection). UMAP maps the TF-IDF vectors into a 2D space, preserving the proximity relationships and local structures of the data (Figure 3).

B. Clusters, Artifacts, and Vulnerabilities

Table IV summarizes some of our observations about the identified clusters and can be analyzed in conjunction with Figure 3. After clustering the artifacts, we noted that out of the **198** CVEs considered, only **12** appeared in more than one cluster. In particular, CVE-2021-44228 (Log4Shell) was the only one that appeared in more than two clusters. It is present in clusters 2, 4, and 8, and is explicitly presented in Table IV as representative for the first two. The three clusters it belongs to are adjacent in Figure 3, suggesting possible similarities among the artifacts they contain.

Clusters 3 and 7 in Table IV include artifacts from F5 BIG-IP (CVE-2020-5902 and CVE-2022-1388). Although they affect the same product, these artifacts were grouped separately because they address distinct vectors: CVE-2022-1388 in cluster 7 relates to digital identity (*Missing Authentication for Critical Function*), while CVE-2020-5902 in cluster 3 is associated with path traversal. This separation reflects differences in the nature of the attacks, as illustrated in Figure 3.

The remaining **186** CVEs appear in only one cluster each. This observation helps us interpret the clusters by leveraging the properties of the respective vulnerabilities, as detailed below.

C. Data Interpretation: Manual Inspection and ChatGPT

To interpret the artifact clusters, we used manual inspection combined with ChatGPT. The manual inspection involved removing empty artifacts, those containing only README files, and those no longer available (see Section V-C). For example, in cluster 0, of the 37 artifacts found, one was empty (see the first row of Table IV). Next, we used ChatGPT to select a notable vulnerability in each cluster and identify related vulnerabilities within the same cluster. Specifically, we cross-referenced information from NomiSec (artifacts) with the NVD (CVEs) and uploaded the resulting table to ChatGPT. In this table, each row corresponded to an artifact, while the columns included the cluster identifier (ranging from one to ten), the artifact html_url, its description, the CVSS vector, and the list of products associated with the CVEs. We then issued the following prompt: `For each cluster, produce observations about its artifacts and identify a notable CVE`. Finally, we conducted an additional manual inspection step to verify the data generated by ChatGPT and refine the final observations. The results are presented in Table IV. **Intra- and inter-group similarities of artifacts.** Figure 3, when contrasted with Table IV, indicates that nearby clusters correspond to categories of vulnerabilities with similar characteristics. For example, cluster 0 (web applications) and cluster 4 (web and enterprise services) appear relatively close, reflecting the relationship between vulnerabilities exploited in web servers and corporate frameworks. Cluster 9 (security protocols), on the other hand, appears more isolated, highlighting vulnerabilities that impact cryptography and authentication differently from other groups. **Artifact packages.** As discussed in Section IV-C, some artifacts are provided in packages. Seven of the CVEs mentioned in the package discussed in Section IV-C are part of cluster 0 (web applications). The other CVEs in the package were not identified by NomiSec and therefore do not appear in the clusters in Table IV.

Attack vector: local vs. remote. Of the 14 vulnerabilities that require local access to the machine to be exploited(see Section IV-B), we noted that clusters 6 and 7 contain 5 and 4 of them, respectively. CVE-2022-0492, for example, corresponds to a flaw in control groups (cgroups) in the Linux Kernel, allowing container escape. It aligns with cluster 6 (cloud and virtualization) but was included in cluster 7 (digital identity) due to its violation of identity premises by enabling privilege

TABLE IV
ARTIFACTS CLUSTERS (LIMITED TO *checkers*). THE MAJORITY OF THE CORRESPONDING VULNERABILITIES ALLOW REMOTE CODE EXECUTION (RCE).

Cluster (Artifacts)			Notable CVE (Artifacts)	Observations
0. Web Applications	36	37	CVE-2018-7600 (Drupalgeddon 2) (2)	Includes flaws in platforms like SAP CRM (CVE-2018-2380) and WebLogic (CVE-2014-4210, CVE-2016-0638, CVE-2017-3248, CVE-2018-2628, CVE-2019-2618, CVE-2020-{2551, 14882}), see Section IV-C, as well as Cross-Site Scripting (XSS) attacks.
1. Remote Access and Firewalls	37	65	CVE-2024-6387 (OpenSSH Regression) (11)	Attacks on services exposed to the internet, including Fortinet FortiOS (CVE-2024-21762) and Palo Alto PAN-OS (CVE-2024-3400), allowing RCE and information disclosure via logs.
2. Supply Chain	41	57	CVE-2021-44228 (Log4Shell) (38)	Flaws in widely used libraries, enabling large-scale attacks. Includes Apache HTTP Server (CVE-2021-41773, code execution via path traversal) and OpenSSL (CVE-2022-3602, buffer overflow). Some of these vulnerabilities can be exploited for privilege escalation.
3. Network Infrastructure Exploitation	49	56	CVE-2020-0796 (SMBGhost) (11)	Attacks on network protocols and critical infrastructure, such as F5 BIG-IP (CVE-2020-5902, path traversal in firewalls) and Cisco ASA (CVE-2020-3452, traversal in VPNs). Includes information disclosure vectors, exposing network details for secondary attacks.
4. Web and Enterprise Services	49	52	CVE-2021-44228 (Log4Shell) (2)	Exploitation of corporate servers, such as Microsoft Exchange (CVE-2021-34473, ProxyShell RCE) and vSphere Client (CVE-2021-21985, remote code execution in virtualization environments).
5. Remote Desktop and Corporate	26	31	CVE-2019-0708 (BlueKeep) (11)	Attacks targeting RDP (CVE-2019-0708, remote execution without authentication) and corporate servers like Citrix ADC (CVE-2019-19781, directory traversal allowing remote execution). Includes privilege escalation flaws.
6. Cloud and Virtualization	34	39	CVE-2023-3519 (Citrix NetScaler) (2)	Exploitation of cloud and virtualization services. Affects Confluence Data Center (CVE-2023-22515, authentication bypass and RCE) and Oracle WebLogic (CVE-2023-21839, malicious deserialization).
7. Digital Identity	32	35	CVE-2022-22965 (Spring4Shell) (3)	Attacks on APIs and digital identity, including F5 BIG-IP (CVE-2022-1388, authentication bypass and RCE) and Zyxel USG (CVE-2022-30525, code execution via firewall APIs). Some of these flaws are exploited in XSS attacks.
8. Legacy Software and ICS	37	39	CVE-2009-0473 (Rockwell Automation ControlLogix) (1)	Exploitation of old vulnerabilities still found in corporate environments. Examples include Windows SMB (CVE-2017-0143, EternalBlue exploit) and Joomla! (CVE-2015-8562, remote code execution via PHP injection).
9. Security Protocols	20	22	CVE-2014-6271 (Shellshock) (4)	Flaws in fundamental protocols, allowing authentication bypass, such as OpenSSL (CVE-2014-0160, Heartbleed, memory leak) and TLS (CVE-2015-0204, FREAK attack).

escalation and bypassing namespace isolation. Other vulnerabilities requiring local access for exploitation and verification, such as CVE-2016-5195 (Dirty COW) and CVE-2022-0847 (Dirty Pipe), are also in cluster 7.

Takeaway message. The combination of manual inspection and ChatGPT facilitated the interpretation of artifact clusters. The proximity of certain clusters suggests similarities between exploitation categories, while the classification of attack vectors and identification of network protocols used by artifacts provide insights that differentiate remotely exploitable vulnerabilities from those requiring local access, enhancing risk analysis when combined with information from CVSS, EPSS [10], and other tools.

Actionable implications. The implications of our methodology for semi-automatically organizing artifacts are threefold:

- **Enhanced Threat Intelligence (TI):** Security researchers can leverage clustering methods to categorize artifacts systematically, aiding in the identification of similar vulnerabilities and exploitation patterns.
- **Improved vulnerability management and patch management:** Organizations can refine their vulnerability prioritization by distinguishing remotely exploitable threats from locally constrained ones, supplementing CVSS and EPSS scores.
- **Automation and scalability:** Integrating AI-based curation with manual inspection ensures better filtering and classification of artifacts, reducing noise and improving reliability for security teams.

VII. CONCLUSION AND FUTURE WORK

Artifacts related to vulnerabilities are essential for understanding and mitigating them. However, these artifacts emerge organically on GitHub, and organizing them is increasingly challenging. In this paper, we analyzed the availability and quality of data provided by NomiSec, explored its intersection with InTheWild, and focused on vulnerability checker artifacts (*checkers*). The categorization of artifacts allowed, for example, the differentiation of legitimate repositories from false positives, facilitating data interpretation and improving risk analysis for network vulnerabilities.

For future work, we intend to expand the automatic categorization of artifacts, enhancing our ability to systematically group and interpret the growing variety of vulnerability-related tools. We also propose investigating the impact of checkers on vulnerability prioritization using metrics such as EPSS and CVSS, aiming to better distinguish checkers from real proof-of-concept exploits. Another promising direction involves analyzing the latency between the publication of vulnerabilities and the appearance of related artifacts in the NomiSec database, as well as studying the evolution of artifact maturity over time. This temporal analysis can help assess the coverage and relevance of datasets used in vulnerability research. Additionally, we plan to expand the quantitative evaluation of our clustering approach, explore automated methods for artifact verification to improve scalability, and deepen our analysis of potentially malicious artifacts by developing more concrete detection strategies. Finally, we aim to strengthen the link between artifact classification and vulnerability metrics, which could further refine risk assessment and prioritization strategies.

REFERENCES

[1] L. Miranda *et al.*, "On the flow of software security advisories," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1305–1320, 2021.

[2] C. Figueiredo *et al.*, "A statistical relational learning approach towards products, software vulnerabilities and exploits," *IEEE Transactions on Network and Service Management*, 2023.
[3] S. E. Yadmani, R. The, and O. Gadyatskaya, "Beyond the surface: Investigating malicious CVE proof of concept exploits on github," *arXiv preprint arXiv:2210.08374*, 2022.
[4] M. O. F. Rokon, R. Islam, A. Darki, E. E. Papalexakis, and M. Faloutsos, "SourceFinder: Finding malware Source-Code from publicly available repositories in GitHub," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 149–163.
[5] S.-S. Yoon, D.-Y. Kim, G.-G. Kim, and I.-C. Euom, "Vulnerability assessment based on real world exploitability for prioritizing patch applications," in *CSNet*. IEEE, 2023, pp. 62–66.
[6] O. Suciu, C. Nelson, Z. Lyu, T. Bao, and T. Dumitraș, "Expected exploitability: Predicting the development of functional vulnerability exploits," in *USENIX Security*, 2022, pp. 377–394.
[7] X. Wang, S. Wang, P. Feng, K. Sun, and S. Jajodia, "Patchdb: A large-scale security patch dataset," in *DSN*. IEEE, 2021, pp. 149–160.
[8] S. Koch, D. Klein, and M. Johns, "The fault in our stars: An analysis of github stars as an importance metric for web source code," in *Measurements, Attacks, and Defenses for the Web (MADWeb)*, 2024.
[9] L. Miranda, C. Figueiredo, D. S. Menasché, and A. Kocheturov, "Patch or Exploit? NVD Assisted Classification of Vulnerability-Related GitHub Pages," in *International Symposium on Cyber Security, Cryptology, and Machine Learning*. Springer, 2023, pp. 511–522.
[10] J. Jacobs, S. Romanosky, B. Edwards, I. Adjerid, and M. Roytman, "Exploit prediction scoring system (EPSS)," *Digital Threats: Research and Practice*, vol. 2, no. 3, pp. 1–17, 2021.
[11] A. Anwar, A. Abusnaina *et al.*, "Cleaning the NVD: comprehensive quality assessment, improvements, and analyses," *IEEE Trans. Dependable and Secure Computing*, vol. 19, no. 6, pp. 4255–4269, 2021.
[12] Claroty, "ISA/IEC 62443-3-3 Paper – Claroty Tool," 2025, accessed: 2025-02-24. [Online]. Available: https://web-assets.claroty.com/claroty-isa-iec-62443-3-3-paper.pdf
[13] Cisco Systems, "ISA/IEC 62443-3-3 White Paper: RAT – Router Audit Tool," 2025, accessed: 2025-02-24. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/security/isaiec-62443-3-3-wp.html

APPENDIX A
TERMINOLOGY

In this appendix, we introduce the main terms used throughout the rest of the work.

- **JavaScript Object Notation (JSON):** A lightweight and structured data format commonly used for data interchange, especially in web applications and APIs. In the context of this work, the analyzed JSON file represents a list of GitHub repositories, each described by a set of metadata such as the repository name, owner, description, URL, and activity statistics. An example can be seen here
- **Exploit:** Malicious code that exploits a flaw or vulnerability related to the software or hardware of an electronic product.
- **Scanner or checker:** A program that analyzes a system or electronic device to identify specific vulnerabilities.
- **Proof of Concept (PoC):** Code or exploit developed to demonstrate that a vulnerability in a system or software can be practically exploited. It is not necessarily a real attack but serves to prove that the flaw exists and could be leveraged by attackers.
- **CVE:** A unique identifier used to catalog and track known security vulnerabilities in software and systems.
- **GitHub repository:** A user's directory of files on GitHub, which can be public or private, allowing collaboration among developers.
- **Description field:** A field created by each repository owner that is referenced by NomiSec at the time the repository is created.
- **Reputation (stars):** The number of stars attributed to a user refers to the sum of the stars received across all their public repositories.
- **EPSS (Exploit Prediction Scoring System):** A statistical model that predicts the likelihood of a known vulnerability being actively exploited by attackers within a given timeframe.
- **Clustering:** An unsupervised machine learning technique used to organize a dataset into groups (or clusters) with similar characteristics.
- **CWE (Common Weakness Enumeration):** A public catalog of types of software and hardware weaknesses that can lead to security vulnerabilities.
- **CVSS (Common Vulnerability Scoring System):** An open standard used to assess the severity of security vulnerabilities in software and hardware systems.
- **Fork:** A copy of a repository created in another user's account, allowing modifications to a project's code without affecting the original repository.
- **Attack Vector (AV):** A CVSS metric from the NVD indicating how a vulnerability can be exploited. Types include: Network (N) – exploitable remotely; Adjacent (A) – requires access to a shared local network; Local (L) – requires local or authenticated access to the system. Physical (P) is introduced in CVSS v4.0 for vulnerabilities requiring direct physical interaction.

APPENDIX B
TOWARDS AN OPEN AUDITING TOOL

There are a number of solutions to audit systems following pre-established standards, such as IEC-62443 [12], [13]. However, to the best of our knowledge, there is no public open tool that continuously updates the set of checkers and relates eventual alerts against those standards. The challenge consists in mapping non-structured data (from GitHub) against structured standards (such as IEC-62443). We envision that the findings presented in this work constitute a first step towards that goal.

APPENDIX C
COMPARISON BETWEEN OUR WORK AND "PATCH OR EXPLOIT? NVD-ASSISTED CLASSIFICATION OF VULNERABILITY-RELATED GITHUB PAGES"

While this work focuses on distinguishing between patches and exploits using interpretable machine learning models and NVD-labeled GitHub pages, our work takes a different direction by concentrating on vulnerability-related artifacts such as scanners and checkers. Rather than identifying whether a GitHub page represents a patch or an exploit, we analyze how certain tools interact with network protocols to uncover and characterize vulnerabilities. Our approach emphasizes the role of these artifacts in the detection and verification of

vulnerabilities, rather than in their remediation (patches) or demonstration (exploits). Additionally, instead of relying on the NVD as the primary source of ground truth, we leverage the NomiSec repository to study a broader set of real-world artifacts, enabling us to extract insights about protocol-level behavior and network-targeted exploitation techniques.