Classificação de artefatos de vulnerabilidades de software usando dados públicos da Internet

Leonardo Ambrus de Lima, Estevao Rabello Ussler, Miguel Angelo Santos Bicudo, Daniel Sadoc Menasche

[1]Instituto de Computação – Universidade Federal do Rio de Janeiro (UFRJ)

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 groups 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.*

Resumo. *Artefatos associados a vulnerabilidades, como* patches*,* exploits *e* scanners*, podem fornecer informações valiosas no contexto da segurança em redes. Em particular, os protocolos de rede utilizados pelos scanners para identificar vulnerabilidades oferecem pistas sobre o funcionamento dessas falhas e os riscos associados. Neste trabalho, utilizamos dados do repositório NomiSec para analisar vulnerabilidades relacionadas a redes, com um foco em scanners. Observamos, por exemplo, que alguns artefatos indicam que a exploração ocorre via HTTP, enquanto outros exigem o uso direto de sockets. Além disso, realizamos uma clusterização e visualização dos artefatos, identificando relações entre categorias. Notamos que certos grupos de artefatos estão associados à exploração de dispositivos de rede, como firewalls, enquanto outros se concentram na exploração de protocolos específicos, como SSL/TLS. Esses achados contribuem para uma melhor compreensão do ecossistema de vulnerabilidades e para o aprimoramento de estratégias de mitigação, baseadas em dados coletados automaticamente e periodicamente a partir do GitHub.*

1. Introdução

A crescente complexidade e interconexão de sistemas de software têm levado a um aumento significativo nas vulnerabilidades de segurança, representando desafios substanciais para a proteção de sistemas e dados sensíveis [Ponce et al. 2022, Miranda et al. 2021, Figueiredo et al. 2023, Yadmani et al. 2022, Rokon et al. 2020]. Este trabalho propõe uma abordagem abrangente para classificar, filtrar e anotar artefatos relacionados a vulnerabilidades de software. Os artefatos abrangem *patches* e *exploits*, bem como códigos

de testes de conceito (PoCs) e verificadores (*checkers*), sendo que damos uma especial ênfase a estes últimos, fornecendo uma visão abrangente das ameaças existentes [Yoon et al. 2023].

A fonte de artefatos escolhida para este trabalho, o NomiSec, consiste num repositório cujo propósito central é a coleta automatizada de Provas de Conceito (PoCs) presentes no GitHub. Essa coleta é realizada por meio da busca em todos os repositórios do GitHub que mencionam identificadores de vulnerabilidades, conhecidos como *Common Vulnerabilities and Exposures* (CVEs). Os dados coletados são então organizados em arquivos JSON, onde, no caso de múltiplos repositórios referenciando uma mesma CVE, todos são armazenados no mesmo arquivo JSON. Esses arquivos não apenas contêm os links para os repositórios relevantes, mas também descrições detalhadas, datas de criação, atualização e inclusão no NomiSec, além de informações sobre os usuários responsáveis pela postagem. Esses arquivos também englobam dados relacionados à visibilidade dos repositórios, discussões associadas e outras métricas pertinentes. Essa abordagem sistemática e abrangente do NomiSec oferece uma valiosa fonte de informações para a comunidade de segurança da informação, possibilitando análises mais detalhadas e embasadas sobre vulnerabilidades de software e suas possíveis implicações.

Neste contexto, nossas principais contribuições são:

Avaliação da abrangência e relevância de fontes como NomiSec: identificamos que fontes como o NomiSec abrangem uma parcela significativa dos dados sobre vulnerabilidades presentes no GitHub, mas que existe um *tradeoff* entre atrasos de publicação e qualidade dos dados (Seção 3).

Identificação da natureza e qualidade dos dados: percebemos que alguns dados divulgados no NomiSec referem-se a artefatos que não representam uma ameaça, como verificadores (*checkers*). Estes últimos, por sua vez, fornecem informações valiosas sobre os mecanismos que podem ser usados para explorar vulnerabilidades, como HTTP versus *sockets* explícitos (Seções 4).

Representação e visualização dos dados: propomos uma metodologia para representar e visualizar os dados de artefatos relacionados a vulnerabilidades presentes em repositórios no GitHub, indicando que alguns dos clusters encontrados referem-se à exploração de dispositivos de rede, como firewalls, enquanto outros se concentram na exploração de protocolos específicos, como SSL/TLS (Seção 5).

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta trabalhos relacionados. Na Seção 3 discutimos a relação entre fontes, indicando sua complementariedade. A Seção 4 reporta observações sobre os artefatos, englobando provas-de-conceito (PoC) e verificadores de vulnerabilidades (*checkers*). Motivados pela relevância destes últimos, consideramos sua clusterização, na Seção 5, e a Seção 6 conclui. Uma versão estendida deste artigo, com apêndices adicionais, encontra-se em [Ambrus de Lima et al. 2025].

2. Trabalhos relacionados

Repositórios públicos, como o GitHub, são fontes ricas para a obtenção de *exploits*, verificadores de vulnerabilidades (*scanners/checkers*) [Suciu et al. 2022] e *patches* [Wang et al. 2021]. No entanto, a qualidade e a confiabilidade desses artefatos va-

riam consideravelmente, exigindo análises criteriosas.

Em [Yadmani et al. 2022], os autores analisam a relação entre a popularidade de repositórios no GitHub e a presença de artefatos maliciosos, concluindo que o número de estrelas não é um indicador confiável para avaliar a legitimidade de um artefato. Nossa análise estende esse estudo ao considerar a reputação dos usuários e sua relação com o número de citações aos usuários advindas de diferentes fontes, encontrando correlação entre ambas (Seção 3.1).

Outros trabalhos exploram técnicas de classificação e agrupamento de vulnerabilidades. Em [Yoon et al. 2023], é proposta uma abordagem para categorizar vulnerabilidades com base em padrões de exploração, utilizando aprendizado de máquina para identificar tendências emergentes. Neste estudo, utilizamos um método semelhante ao aplicar técnicas de *clustering* para classificar artefatos coletados no repositório NomiSec.

A relação entre scanners e os protocolos de rede utilizados para detecção de vulnerabilidades também tem sido investigada. Em [Miranda et al. 2023], os autores analisam como diferentes protocolos impactam a forma como vulnerabilidades são exploradas e mitigadas. Nossa pesquisa reforça essa perspectiva ao demonstrar que a natureza do protocolo influencia tanto a exploração quanto a eficácia das ferramentas de detecção.

Por fim, iniciativas como o EPSS (*Exploit Prediction Scoring System*) [Jacobs et al. 2021] estimam a probabilidade de uma vulnerabilidade ser explorada, com base em artefatos públicos. No entanto, não há transparência sobre a diferenciação entre *checkers* e *exploits*. Nosso trabalho destaca a importância dessa distinção e propõe uma metodologia para classificá-los com base na estrutura dos repositórios e na forma como interagem com vulnerabilidades.

Dessa forma, este trabalho complementa a literatura existente ao fornecer uma análise sobre a relação entre artefatos de vulnerabilidades e seus respectivos protocolos de rede, além de apresentar uma metodologia para categorização automática e visualização de dados coletados em fontes públicas.

3. Relação entre fontes: complementares ou substituíveis?

Existe um balanço entre características de diferentes fontes de dados sobre vulnerabilidades. O NomiSec[1] é alimentado por robôs, que varrem todo o GitHub por informações sobre provas-de-conceito. Já o NVD,[2] por outro lado, contém apenas informações curadas sobre vulnerabilidades. O tempo de resposta do primeiro, para publicação de novas informações sobre vulnerabilidades, naturalmente, é muito menor que o segundo. No meio termo, temos o InTheWild,[3] uma plataforma que como NVD também contém informações curadas sobre vulnerabilidades, mas que por outro lado tem um tempo de resposta para atualizações bem menor que o NVD.

A fonte Inthewild.io é uma plataforma notável que se destaca como uma fonte confiável para artefatos relacionados a vulnerabilidades de software. A proposta central do Inthewild.io é catalogar e disponibilizar informações sobre incidentes de segurança reais que ocorreram "em campo", ou seja, em ambientes de produção. Isso é alcançado por

[1] `https://github.com/nomi-sec/PoC-in-GitHub`
[2] `https://nvd.nist.gov/`
[3] `https://inthewild.io/`

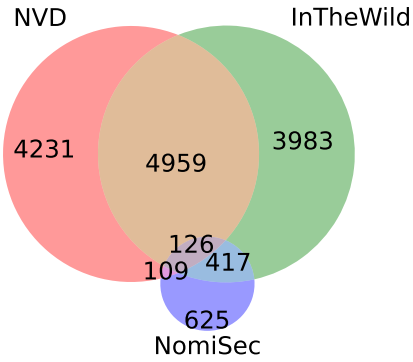

Figura 1. Diagrama de Venn dos usuários no NomiSec, InTheWild e NVD.

meio da coleta de dados de várias fontes, incluindo feeds de notícias, fóruns de segurança e relatórios de empresas de segurança cibernética. Esses dados são meticulosamente organizados e apresentados em uma interface acessível, permitindo que os pesquisadores e profissionais de segurança da informação obtenham insights valiosos sobre as ameaças atuais e emergentes.

Constatou-se que o InTheWild faz uso de 10 fontes de dados, sendo o GitHub uma delas. Por outro lado, o GitHub é a única fonte usada pelo NomiSec.

Dados até 2024. Desde a criação do InTheWild até janeiro de 2024, foram identificados 5489 repositórios do GitHub presentes no InTheWild, em comparação com os 11627 links do GitHub encontrados no NomiSec no mesmo período. Essa discrepância suscitou uma investigação mais aprofundada para determinar a interseção desses conjuntos de links entre o InTheWild e o NomiSec.

Observou-se que todos os 5489 links do GitHub presentes no InTheWild também estavam no NomiSec. Entretanto, chamou atenção o fato de o NomiSec possuir 6138 links adicionais que não estavam presentes no InTheWild. Essa constatação levou à conclusão de que os dois repositórios são complementares: o InTheWild é mais abrangente devido à inclusão de mais fontes além do GitHub, o que permite detalhar melhor as vulnerabilidades recolhidas, enquanto o NomiSec oferece um conjunto exclusivo de links.

É importante ressaltar que nem todos esses links, que estão presentes exclusivamente no NomiSec, são necessariamente PoCs. Portanto, uma investigação mais aprofundada se fez necessária para avaliar sua relevância, conforme consideramos no restante deste artigo.

Dados de 2024. Em 2024, a tendência acima descrita parcialmente se manteve. Em particular, o NomiSec continuou publicando referências para muitos artefatos que não foram citados pelo InTheWild. Entretanto, no sentido contrário, também passaram a ser observadas instâncias de artefatos citados pelo InTheWild mas que não constam no NomiSec. Em particular, encontramos para vulnerabilidades do tipo CVE-2024-$\star$ um total de 719 links comuns ao InTheWild e NomiSec. O número de links exclusivos do NomiSec e do InTheWild foi de 1,333 e 925, respectivamente.

Ainda focando nas vulnerabilidades do tipo CVE-2024-$\star$, analisamos os usuários do GitHub que aparecem nos links do NomiSec, InTheWild e NVD. A Figura 1 ilustra um diagrama de Venn dos usuários presentes no NomiSec, InTheWild e NVD, com dados

colhidos em 9 de janeiro de 2025. É interessante destacar o caráter complementar das três bases de dados: um número substancial de usuários contribuiu de forma exclusiva para uma das bases. Entretanto, nem todas as contribuições possuem a mesma qualidade.

Os usuários presentes no NomiSec, mas que nunca foram citados pelo InTheWild e NVD, tendem a suscitar mais dúvidas sobre sua reputação. Em particular, dos 1,277 usuários referenciados pelo NomiSec, 652 (109+126+417) deles apareceram ou no NVD ou no InTheWild, ou em ambos. Isto sugere que estes usuários têm alta credibilidade, tendo em vista que os dados do NVD e InTheWild são curados. Os demais 625 usuários, que não aparecem nem no NVD nem no InTheWild, requerem uma análise mais criteriosa para se avaliar se os artefatos por eles compartilhados são funcionais ou, ainda, maliciosos. A seguir, apresentamos detalhes adicionais sobre a reputação dos usuários.

3.1. Reputação dos usuários

Qual a relação entre a reputação dos usuários e repositórios e a maturidade dos artefatos por eles compartilhados? Para responder tal pergunta, analisamos os usuários do GitHub que são referenciados a partir do NomiSec. Em particular, assumimos que os usuários que só foram citados pelo NomiSec, e que ainda não foram citados pelo NVD e InTheWild, produzem artefatos menos maduros do que aqueles que já foram também citados por uma fonte alternativa. Dada esta hipótese, avaliamos se os usuários que encontram-se na interseção dos três conjuntos do diagrama de Venn da Figura 1 podem ser distinguidos por meio de características como número de estrelas a eles conferidas. É importante esclarecer que, no contexto deste estudo, um usuário do GitHub corresponde ao dono de um conjunto de repositórios. Assim, o número de estrelas atribuído a um usuário refere-se à soma das estrelas recebidas por todos os seus repositórios públicos. Como exemplo, podemos considerar o usuário Orange-418, que possui repositórios como AgentDVR-5.1.6.0-File-Upload-and-Remote-Code Execution[4]. Em função do poder classificatório de tais características, podemos usá-las como indicadoras parciais por fornecerem informações sobre a prioridade que deve ser atribuída aos artefatos produzidos pelos respectivos usuários. Essa priorização é especialmente útil para orientar estratégias de aplicação de patches, garantindo que vulnerabilidades com artefatos mais maduros ou críticos sejam corrigidas primeiro.

Os gráficos da Figura 2 mostram a distribuição acumulada (CDF) de métricas de reputação (estrelas, seguidores, repositórios públicos e forks) dos usuários que contribuíram para repositórios de vulnerabilidades citados pelo NomiSec em CVEs de 2024. Esses usuários foram classificados em quatro grupos, conforme sua presença no NVD e InTheWild. Ao gerar a Figura 2(d), para evitar distorções, removemos três usuários com números de repositórios muito acima da média: gmh5225[5] (23.934), CrackerCat[6] (18.728) e killvxk[7] (11.600). O quarto maior, ahlfors, possui 1.475 repositórios.

Os usuários citados apenas pelo NomiSec exibem métricas inferiores nos percentis superiores, indicando menor visibilidade e impacto. Já os usuários mencionados no

[4]repositórios do usuário Orange-418 citados: `https://github.com/Orange-418/AgentDVR-5.1.6.0-File-Upload-and-Remote-Code-Execution` e `https://github.com/Orange-418/CVE-2024-22515-File-Upload-Vulnerability`

[5]Usuário gmh5225: `https://github.com/gmh5225`

[6]Usuário CrackerCat: `https://github.com/CrackerCat`

[7]Uruário killvxk: `https://github.com/killvxk`

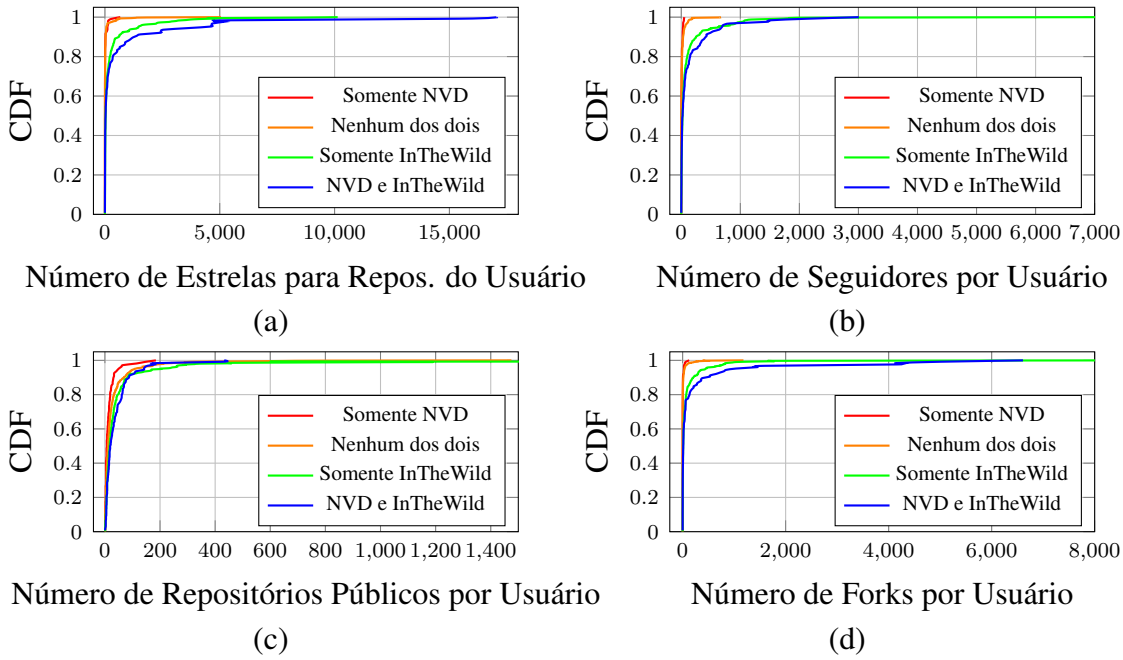

Figura 2. Distribuições acumuladas (CDF) dos atributos de usuários.

NVD e InTheWild apresentam valores significativamente mais altos em todas as métricas, sugerindo maior reputação e influência. O número de estrelas e seguidores reflete reconhecimento e exposição, enquanto a quantidade de repositórios públicos e forks indica maior produtividade e relevância na comunidade [He et al. 2024].

Mensagem principal. Os resultados indicam que a reputação e o histórico de contribuição dos usuários são fatores-chave para avaliar a confiabilidade dos artefatos. A predominância de usuários com alta visibilidade nos conjuntos NVD e InTheWild sugere que essas fontes curadas são mais seletivas na indexação de repositórios relevantes. Por outro lado, a presença de usuários pouco conhecidos exclusivamente no NomiSec reforça a necessidade de uma análise criteriosa para distinguir artefatos legítimos de *scams* ou códigos de baixa qualidade, conforme discutido nas próximas seções.

3.2. Abrangência do NomiSec com relação a todo GitHub: relevância e atraso

A seguir, apresentamos detalhes adicionais sobre as vulnerabilidades com artefatos no NomiSec. Em particular, visamos avaliar a consistência e a completude dos dados entre as diferentes fontes consideradas, lembrando que nossa principal fonte (NomiSec) é subconjunto de outra (GitHub).

Dentre as vulnerabilidades citadas pelo NomiSec, descobrimos que um total de 1.227 vulnerabilidades, cada uma delas identificada por um CVE, contém mais de um artefato, variando entre 2 e 396 artefatos por vulnerabilidade. Impulsionados por essa constatação, decidimos conduzir uma busca no GitHub por vulnerabilidades com um número significativo de artefatos. Durante tal exercício, identificamos que muitos dos links encontrados no GitHub não estavam presentes no repositório NomiSec.

Para fornecer uma visão mais clara dessa discrepância, apresentamos na Figura 3 um gráfico com as CVEs com maior número de links. As barras claras indicam o número de links para repositórios encontrados pelo NomiSec, enquanto as barras escuras indicam

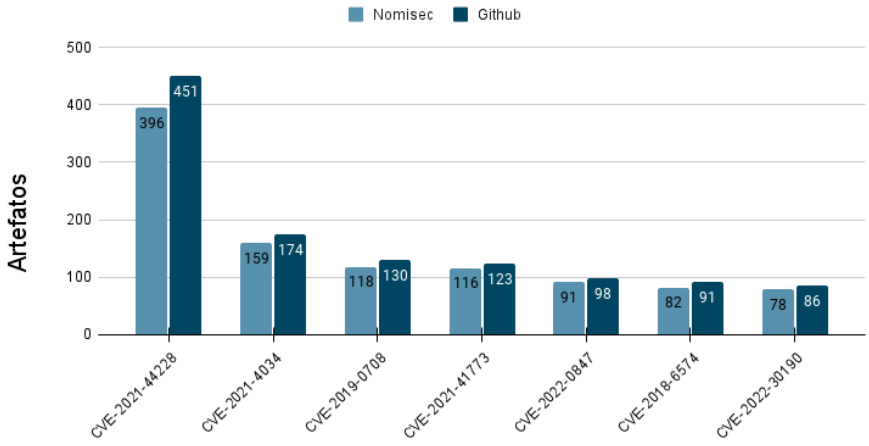

Figura 3. Abrangência do NomiSec: NomiSec abrange parcelas uma fração significativa dos artefatos do GitHub, mas não todos.

o número de links de repositórios presentes no GitHub.

A Figura 3 levanta questionamentos sobre as razões pelas quais o NomiSec não atinge uma cobertura completa do GitHub. Para explorar essa questão, formulamos algumas hipóteses. O NomiSec pode aplicar um filtro para selecionar apenas alguns links, descartando conteúdos inválidos ou maliciosos. Isso é reforçado pela possibilidade de que usuários reportem problemas via *issues* no próprio repositório, levando à remoção de certos links [Ambrus de Lima et al. 2025]. Além disso, as atualizações ocorrem a cada 6 horas, o que pode causar atrasos na inclusão de novos links, embora a defasagem seja pequena. Outra possibilidade é que o NomiSec enfrente limitações ao usar a API do GitHub para coletar links, impedindo a obtenção completa dos repositórios relacionados às CVEs.

Para os fins deste trabalho, considerando a complexidade de analisar o NomiSec em sua totalidade e o fato de que ele representa apenas uma fração do GitHub, decidimos focar nos verificadores (*checkers*). Essa escolha se justifica pela relevância desses artefatos na avaliação de vulnerabilidades e pelos impactos que sua categorização pode ter na interpretação de risco, como será detalhado nas seções a seguir.

Mensagem principal. Mesmo fontes automatizadas como o NomiSec ainda oferecem uma perspectiva limitada do GitHub. Para aumentar a cobertura, é necessário considerar múltiplas fontes.

Implicações práticas. Usuários que dependem do NomiSec ou de qualquer outra fonte automatizada para rastreamento de vulnerabilidades devem estar cientes de que:

1. ele fornece uma quantidade significativa de verificadores de vulnerabilidades, ou seja, muitos dos artefatos não são PoCs;
2. deve ser complementado com buscas mais amplas no GitHub para evitar a perda de artefatos críticos.

4. Identificando verificadores (*checkers*)

Ao analisar os artefatos compartilhados no NomiSec, conduzimos uma análise manual detalhada, empregando termos-chave como “checker” e outras palavras relacionadas. Essa

busca revelou a presença de 451 artefatos que supostamente são verificadores de vulnerabilidades. Cada artefato identificado como potencialmente relacionado a *checkers* foi submetido a uma análise cuidadosa, em que examinamos os repositórios e os códigos associados para determinar sua natureza e função.

Os resultados desta análise foram registrados em um relatório detalhado, publicado e disponibilizado publicamente no GitHub.[8] No total, **442** artefatos foram classificados como positivos, ou seja, exclusivamente verificadores, enquanto 9 foram categorizados como negativos, englobando somente provas de conceito.

4.1. Por que estudar *checkers*?

A seguir, destacamos alguns motivos pelos quais o estudo de *checkers* é relevante:

- **Análise de risco (EPSS):** Diferenciar entre *checkers* e outras ferramentas é essencial para evitar a interpretação errônea de risco. Por exemplo, não se pode assumir que a existência de um *checker* implica a disponibilidade de um *exploit* funcional, quando na realidade o *checker* pode ser apenas uma ferramenta de verificação. Essa distinção é fundamental para uma avaliação de risco precisa. Entretanto, ferramentas como EPSS [Suciu et al. 2022, Jacobs et al. 2021] não publicam o processo usado para distinguir entre *checkers* e demais artefatos.
- **Mitigação de vulnerabilidades via engenharia reversa:** Compreender o funcionamento dos *checkers* possibilita a proposição de contramedidas que não necessariamente dependam de *patches*. Por exemplo, ao identificar funcionalidades específicas verificadas por um *checker*, pode-se desabilitá-las como uma medida preventiva.[9]
- **Entendimento de vulnerabilidades:** Analisar como protocolos de rede são utilizados pelos *checkers* e como esses se relacionam com as categorias de vulnerabilidades (como as descritas no CWE) fornece pistas valiosas sobre o potencial de exploração de vulnerabilidades no mundo real. Essa compreensão pode orientar abordagens mais eficazes para mitigação e prevenção.

4.2. Vetor de acesso (*access vector*): Local vs Adjacent vs Network

Nosso estudo analisou um total de **451** repositórios no GitHub contendo verificadores de vulnerabilidades. Esses repositórios cobrem **198** CVEs distintas, já que algumas vulnerabilidades possuem mais de um verificador disponível. Por exemplo, a vulnerabilidade CVE-2021-44228 (Log4Shell) está associada a 40 verificadores diferentes. Cada verificador pode ser considerado um artefato relacionado à identificação ou exploração da vulnerabilidade.

Dentre as várias informações dispobilizadas no NVD sobre as vulnerabilidades, por meio do CVSS, focamos a seguir no vetor de acesso (*access vector*) de cada vulnerabilidade. Nosso objetivo é compreender melhor as características individuais de cada vulnerabilidade e identificar padrões nas vulnerabilidades para as quais os usuários desenvolveram verificadores. A grande maioria das CVEs com verificadores, a saber, 182 delas, possui o vetor de acesso classificado como *Network*, e outras 2 como *Adjacent*.

[8] `https://github.com/leoambrus/Tagging-NomiSec`

[9] Para um exemplo concreto, vide `https://github.com/tarraschk/CVE-2023-36884-Checker/blob/main/mitigate-cve-2023-36884.ps1`

Tabela 1. Classificação das CVEs por protocolo de rede utilizado

Protocolo	CVE Associadas
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

Percebemos que, para essas 184 vulnerabilidades, os verificadores fazem uso da rede para identificá-las remotamente. Para 14 vulnerabilidades, por outro lado, verificamos que é necessário acesso local à vulnerabilidade para explorá-la, ou seja, não é possível explorar a vulnerabilidade via rede, a não ser que se consiga um *shell* remoto para o dispositivo vulnerável.

Mensagem principal. Desenvolvedores de verificadores tendem a priorizar vulnerabilidades com impacto mais amplo, especialmente aquelas exploráveis via rede. Ao analisar os protocolos utilizados pelos verificadores — geralmente mais simples do que exploits completos de Prova de Conceito (PoC) — podemos obter insights adicionais sobre as próprias vulnerabilidades, como explorado na próxima seção.

Implicações práticas. Ferramentas de varredura de vulnerabilidades podem integrar verificadores do NomiSec para aprimorar suas capacidades de detecção, sendo que a maioria das verificações pode ser executada remotamente. A automação desse processo poderia melhorar as avaliações de segurança ao aproveitar métodos de detecção de maneira oportuna.

4.3. Pacotes de verificadores de vulnerabilidades e protocolos de rede

Identificamos que alguns verificadores (*checkers*) são organizados e distribuídos em pacotes. Um dos pacotes que analisamos foi [10]. Esse pacote funciona de maneira semelhante a ferramentas antivírus; entretanto, em vez de procurar por vírus, tais pacotes são projetados para identificar vulnerabilidades em sistemas.

Um total de 23 vulnerabilidades são verificadas pelo pacote acima mencionado (vide Tabela 1). Todas as vulnerabilidades verificadas no pacote são do tipo *remote code execution (RCE)*, i.e., permitem a execução remota de código em sistemas vulneráveis. As vulnerabilidades podem ser divididas pelo protocolo de rede que utilizam para fazer as verificações. Para determinar o protocolo, identificamos uma relação de um-para-um entre o protocolo citado pelo NVD na descrição da vulnerabilidade (HTTP, T3 ou IIOP) e o protocolo usado pelo *checker* para verificar, remotamente, a presença da mesma em um sistema.

- **Método de detecção via HTTP:** Dez dessas vulnerabilidades utilizam a biblioteca *utils* e o protocolo HTTP. Essas ferramentas tentam acessar endpoints específicos em servidores vulneráveis, enviando requisições HTTP para verificar a presença das falhas.

[10] Vide `https://github.com/0xn0ne/weblogicScanner/blob/master/ws.py`

- **Método de detecção via T3:** As outras doze vulnerabilidades utilizam *sockets* para comunicação direta via TCP/IP, empregando o protocolo T3. Este protocolo proprietário é usado pelo Oracle WebLogic Server para comunicações entre servidores e entre cliente-servidor.
- **Método de detecção via IIOP:** Um *checker* utiliza o protocolo IIOP (Internet Inter-ORB Protocol) para detectar a presença da vulnerabilidade CVE-2020-2551. O método consiste no envio de uma mensagem codificada no formato GIOP (General Inter-ORB Protocol), que é interpretada por servidores que utilizam o Oracle WebLogic. Ao receber uma resposta contendo a assinatura GIOP, o scanner identifica servidores potencialmente vulneráveis.

Mensagem principal. Embora as vulnerabilidades sejam apresentadas isoladamente em bancos de dados como o NVD, na prática é possível aprender muito analisando vulnerabilidades semelhantes. Nesta seção, identificamos 23 vulnerabilidades que afetam produtos da Oracle e demonstramos como elas podem ser classificadas com base no protocolo de rede utilizado para exploração. Essa classificação é possível graças ao cruzamento dos dados coletados no NomiSec com o NVD, que fornece informações detalhadas, incluindo métricas do CVSS. Esses dados permitem agrupar as vulnerabilidades conforme os protocolos explorados, facilitando a análise de seu impacto e risco.

Implicações práticas. Verificadores podem servir como um recurso valioso para aprimorar filtros de inspeção profunda de pacotes (DPI), fornecendo insights detalhados sobre como as vulnerabilidades se manifestam no nível do protocolo de rede. Ao incorporar métodos de detecção específicos para cada protocolo a partir dos verificadores, as ferramentas de segurança podem identificar e mitigar ameaças de forma mais eficaz em tempo real. Essa abordagem possibilita mecanismos de defesa proativos que vão além das bases de vulnerabilidades estáticas, aprimorando a monitoração da segurança da rede e a capacidade de resposta a incidentes.

5. Representando, Organizando, Visualizando e Interpretando Artefatos

Como extrair informações estruturadas e úteis a partir dos artefatos disponibilizados em plataformas como NomiSec? Embora o NomiSec potencialmente ofereça rapidamente ponteiros para inúmeros artefatos relacionados a vulnerabilidades, o processo de coleta de tais artefatos é automatizado via *bots*. No caso do NomiSec, os *bots* simplesmente distribuem os artefatos em pastas de acordo com o ano de divulgação das vulnerabilidades correspondentes. Dada a quantidade de vulnerabilidades publicadas anualmente, torna-se imperativo adotar métodos mais eficientes de organização e classificação dos dados. Para tal, nesta seção exploramos a utilização da técnica de clusterização e visualização.

5.1. Metodologia para Representar, Organizar e Visualizar Artefatos

Inicialmente, os arquivos obtidos a partir do NomiSec contendo informações sobre os artefatos são percorridos e o texto relevante de cada documento é extraído. Em particular, para cada artefato disponibilizado no NomiSec, dois campos importantes para nossas análises são 'html_url' e 'description', contendo o ponteiro para o repositório GitHub do artefato e a respectiva descrição.

Seguimos então com a representação, clusterização e visaulização dos dados.
Representação: Os textos extraídos são transformados em vetores numéricos usando

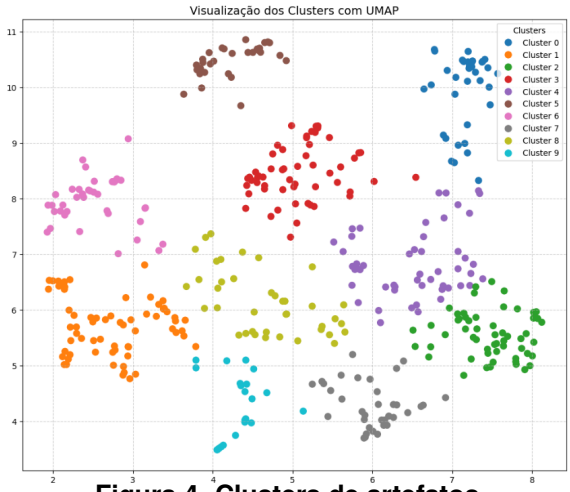

Figura 4. Clusters de artefatos.

o produto da frequência de termos e inverso da frequência nos documentos (TF-IDF). Durante essa etapa, seguindo a prática comum, as *stopwords* em inglês são removidas. **Clusterização:** Os vetores TF-IDF resultantes são então submetidos ao algoritmo K-Means para realizar a clusterização dos documentos. Em particular, no nosso estudo consideramos dez clusters.

Visualização: Para visualizar os resultados da clusterização usamos o UMAP (Uniform Manifold Approximation and Projection). O UMAP mapeia os vetores TF-IDF para um espaço 2D, preservando as relações de proximidade e as estruturas locais dos dados (Figura 4).

5.2. Clusters, Artefatos e Vulnerabilidades

A Tabela 2 resume algumas de nossas observações sobre os clusters encontrados e pode ser analisada em conjunto com a Figura 4. Após clusterizar os artefatos, notamos que das **198** CVEs consideradas, apenas **12** fizeram parte de mais mais de um cluster. Em particular, a CVE-2021-44228 (Log4Shell) foi a única que constou em mais de dois clusters. Ela consta nos cluster 2, 4 e 8, sendo explicitamente apresentada na Tabela 2 como representativa para os dois primeiros. Os três clusters ao qual ela pertence são adjacentes na Figura 4, sugerindo possível similaridades entre os artefatos neles contidos.

Os clusters 3 e 7 da Tabela 2 incluem artefatos do F5 BIG-IP (CVE-2020-5902 e CVE-2022-1388). Embora afetem o mesmo produto, tais artefatos foram agrupados separadamente, pois abordam vetores distintos: a CVE-2022-1388, no cluster 7, trata de identidade digital (*Missing Authentication for Critical Function*), enquanto a CVE-2020-5902, no cluster 3, está associada à travessia de caminho (*Path Traversal*). A separação reflete diferenças na natureza dos ataques, como ilustrado na Figura 4.

As demais **186** CVEs constam em apenas um cluster cada. Esta observação nos ajuda já que para interpretar os clusters podemos fazer uso das propriedades das respectivas vulnerabilidades, conforme detalhamos a seguir.

5.3. Rumo à Interpretação dos Dados: Inspeção Manual e ChatGPT

Para interpretar os clusters de artefatos fizemos uso de inspeção manual em conjunto com ChatGPT. A inspeção manual envolveu a remoção de artefatos vazios, que conti-

Tabela 2. Clusters de artefatos (restritos a *checkers*). A vasta maioria das vulnerabilidades correspondentes permite execução remota de código (RCE).

Cluster (Artefatos)			CVE Notável (Artefatos)	Observações
0. Aplicações web	36	37	CVE-2018-7600 (Drupalgeddon 2) (2)	Inclui falhas em plataformas como SAP CRM (CVE-2018-2380) e WebLogic (CVE-2014-4210, CVE-2016-0638, CVE-2017-3248, CVE-2018-2628, CVE-2019-2618, CVE-2020-{2551, 14882}), vide Seção 4.3, além de ataques de Cross-Site Scripting (XSS).
1. Acesso remoto e firewalls	37	65	CVE-2024-6387 (OpenSSH Regression) (11)	Ataques a serviços expostos à internet, incluindo Fortinet FortiOS (CVE-2024-21762) e Palo Alto PAN-OS (CVE-2024-3400), permitindo RCE e divulgação de informação via logs.
2. Cadeia de suprimentos	41	57	CVE-2021-44228 (Log4Shell) (38)	Falhas em bibliotecas amplamente usadas, permitindo ataques em escala. Inclui Apache HTTP Server (CVE-2021-41773, execução de código via path traversal) e OpenSSL (CVE-2022-3602, buffer overflow). Algumas dessas vulnerabilidades podem ser exploradas para escalação de privilégio.
3. Exploração de infra de rede	49	56	CVE-2020-0796 (SMBGhost) (11)	Ataques a protocolos de rede e infraestrutura crítica, como F5 BIG-IP (CVE-2020-5902, 'path traversal' em firewalls) e Cisco ASA (CVE-2020-3452, 'traversal' em VPNs). Inclui vetores de divulgação de informação, expondo detalhes da rede para ataques secundários.
4. Serviços web e empresariais	49	52	CVE-2021-44228 (Log4Shell) (2)	Exploração de servidores corporativos, como Microsoft Exchange (CVE-2021-34473, ProxyShell RCE) e vSphere Client (CVE-2021-21985, execução remota de código em ambientes de virtualização).
5. Desktop remoto e corporativo	26	31	CVE-2019-0708 (BlueKeep) (11)	Ataques direcionados a RDP (CVE-2019-0708, execução remota sem autenticação) e servidores corporativos como Citrix ADC (CVE-2019-19781, directory traversal permitindo execução remota). Inclui falhas de escalonamento de privilégios.
6. Nuvem e virtualização	34	39	CVE-2023-3519 (Citrix NetScaler) (2)	Exploração de serviços em cloud e virtualização. Afeta Confluence Data Center (CVE-2023-22515, bypass de autenticação e RCE) e Oracle WebLogic (CVE-2023-21839, deserialização maliciosa).
7. Identidade digital	32	35	CVE-2022-22965 (Spring4Shell) (3)	Ataques a APIs e identidade digital, incluindo F5 BIG-IP (CVE-2022-1388, bypass de autenticação e RCE) e Zyxel USG (CVE-2022-30525, execução de código via firewall APIs). Algumas dessas falhas são exploradas em ataques XSS.
8. Softwares legados e ICS	37	39	CVE-2009-0473 (Rockwell Automation ControlLogix) (1)	Exploração de vulnerabilidades antigas ainda encontradas em ambientes corporativos. Exemplos incluem Windows SMB (CVE-2017-0143, EternalBlue exploit) e Joomla! (CVE-2015-8562, execução remota de código via injeção de PHP).
9. Protocolos de segurança	20	22	CVE-2014-6271 (Shellshock) (4)	Falhas em protocolos fundamentais, permitindo bypass de autenticação, como OpenSSL (CVE-2014-0160, Heartbleed, vazamento de memória) e TLS (CVE-2015-0204, ataque FREAK).

nham apenas README, e que não existiam mais (vide Seção **??**). No caso do cluster 0, por exemplo, dos 37 artefatos encontrados, um deles estava vazio (vide primeira linha da Tabela 2). Em seguida, solicitamos ao ChatGPT que escolhesse uma vulnerabilidade notável em cada cluster, e que em seguida apontasse por vulnerabilidades relacionadas encontradas no mesmo cluster. Finalmente, uma etapa adicional de inspeção manual envolveu a verificação detalhada dos dados gerados pelo ChatGPT e a curadoria dos dados resultantes. O resultado é apresentado na Tabela 2.

Similaridades intra e inter grupos de artefatos. A Figura 4, quando contrastada com a Tabela 2, indica que clusters próximos correspondem a categorias de vulnerabilidades com características semelhantes. Por exemplo, o cluster 0 (aplicações web) e o cluster 4 (serviços web e empresariais) aparecem relativamente próximos, refletindo a relação entre vulnerabilidades exploradas em servidores web e frameworks corporativos. Já o clus-

ter 9 (protocolos de segurança) aparece mais isolado, destacando vulnerabilidades que impactam criptografia e autenticação de forma distinta dos outros grupos. A Tabela 2 fornece detalhes sobre cada cluster, listando as vulnerabilidades mais notáveis e descrevendo seus impactos específicos. A predominância de vulnerabilidades de execução remota de código (RCE) nos clusters reforça a criticidade dos artefatos analisados, corroborando o fato de que o NomiSec e outras fontes de dados complementam informações oferecidas por plataformas mais estabelecidas, como o NVD.

Pacotes de artefatos. Conforme discutido na Seção 4.3, alguns artefatos são disponibilizados em pacotes. Sete das CVEs citadas no pacote discutido na Seção 4.3 fazem parte do cluster 0 (aplicações web). As outras CVEs que fazem parte do pacote não foram identificadas pelo NomiSec, e por isso não aparecem nos clusters da Tabela 2.

Vetor de acesso: local versus remoto. Das 14 vulnerabilidades que precisam de acesso local à maquina para serem exploradas (vide Seção 4.2), notamos que os clusters 6 e 7 contém 5 e 4 delas, respectivamente. A CVE-2022-0492, por exemplo, corresponde a uma falha no controle de cgroups no Linux Kernel, permitindo um escape de *container* ('Container Escape'). Ela tem afinidade com o cluster 6 (nuvem e virtualização), mas foi incluída no cluster 7 (identidade digital), por violar premissas de identidade ao admitir escalada de privilégios, contornando o isolamento dos 'namespaces'. As vulnerabilidades CVE-2016-5195 (Dirty COW) e CVE-2022-0847 (Dirty Pipe) também permitem escalada de privilégios, precisam de acesso local à máquina para serem exploradas e verificadas, e estão no cluster 7. Como outros exemplos de vulnerabilidades cujo vetor de acesso é local, a CVE-2021-1675 (PrintNightmare) e a CVE-2021-4034 (PwnKit - pkexec vulnerability) afetam o Windows, e estão no cluster 4 (serviços web e empresariais).

Mensagem principal. A combinação de inspeção manual e ChatGPT facilitou a interpretação dos clusters de artefatos. A proximidade de certos clusters sugere semelhanças entre categorias de exploração, enquanto a classificação dos vetores de acesso e a identificação dos protocolos de rede utilizados pelos artefatos fornecem insights que diferenciam vulnerabilidades exploráveis remotamente daquelas que requerem acesso local, aprimorando a análise de risco quando combinada com informações do CVSS, EPSS [Jacobs et al. 2021] e outras ferramentas.

Implicações práticas. As implicações de nossa metodologia para a organização semi-automática de artefatos são três:

- **Aprimoramento da Inteligência de Ameaças (TI):** Pesquisadores de segurança podem utilizar métodos de clustering para categorizar artefatos de forma sistemática, auxiliando na identificação de vulnerabilidades similares e padrões de exploração.
- **Melhoria na gestão e correção de vulnerabilidades:** Organizações podem refinar a priorização de vulnerabilidades ao distinguir ameaças exploráveis remotamente daquelas restritas ao ambiente local, complementando as pontuações do CVSS e EPSS.
- **Automação e escalabilidade:** A integração de curadoria baseada em IA com inspeção manual garante uma melhor filtragem e classificação de artefatos, reduzindo ruídos e aumentando a confiabilidade para equipes de segurança.

6. Conclusão e Trabalhos Futuros

Artefatos sobre vulnerabilidades são fundamentais para compreender e mitigar as mesmas. Entretanto, os artefatos surgem de forma orgância no GitHub, e organizá-los é desafiador. Neste artigo analisamos a disponibilidade e a qualidade dos dados fornecidos pelo NomiSec, explorando sua interseção com o InTheWild e focando em artefatos verificadores de vulnerabilidades (*checkers*). A categorização dos artefatos permitiu, por exemplo, diferenciar repositórios legítimos de falsos positivos, facilitando a interpretação dos dados e aprimorando a análise de risco para vulnerabilidades de rede. Como trabalhos futuros, pretendemos investigar o impacto dos verificadores na priorização de vulnerabilidades, e analisar a latência e a evolução da maturidade dos artefatos no NomiSec.

Agradecimentos. Este trabalho foi financiado em parte pela CAPES, CNPq e FAPERJ (E-26/201.376/2021 e E-26/201.308/2024).

Referências

Ambrus de Lima, L., Ussler, E. R., Bicudo, M. A. S., and Menasche, D. S. (2025). Classificação de artefatos de vulnerabilidades de software usando dados públicos da internet. Relatório técnico: `https://bit.ly/Relat_tec_SBRC`.

Figueiredo, C. et al. (2023). A statistical relational learning approach towards products, software vulnerabilities and exploits. *IEEE Trans. Network and Service Management*.

He, H. et al. (2024). 4.5 Million (Suspected) Fake Stars in GitHub: A Growing Spiral of Popularity Contests, Scams, and Malware. *arXiv:2412.13459*.

Jacobs, J., Romanosky, S., Edwards, B., Adjerid, I., and Roytman, M. (2021). Exploit prediction scoring system (EPSS). *Digital Threats: Research and Practice*, 2(3):1–17.

Miranda, L. et al. (2021). On the flow of software security advisories. *IEEE Transactions on Network and Service Management*, 18(2):1305–1320.

Miranda, L., Figueiredo, C., Menasché, D. S., and Kocheturov, A. (2023). Patch or exploit? NVD assisted classification of vulnerability-related github pages. *International Symposium on Cyber Security, Cryptology, and Machine Learning*.

Ponce, L. M. S. et al. (2022). Caracterização escalável de vulnerabilidades de segurança: um estudo de caso na internet brasileira. *SBRC*.

Rokon, M. O. F., Islam, R., et al. (2020). SourceFinder: Finding malware source-code from publicly available repositories in GitHub. *RAID*, pages 149–163.

Suciu, O. et al. (2022). Expected exploitability: Predicting the development of functional vulnerability exploits. In *USENIX Security*, pages 377–394.

Wang, X. et al. (2021). PatchDB: A large-scale security patch dataset. In *IEEE/IFIP Conf. Dependable Systems and Networks (DSN)*, pages 149–160. IEEE.

Yadmani, S. E., The, R., and Gadyatskaya, O. (2022). Beyond the surface: Investigating malicious CVE proof of concept exploits on github. *arXiv preprint arXiv:2210.08374*.

Yoon, S.-S. et al. (2023). Vulnerability assessment based on real world exploitability for prioritizing patch applications. In *CSNet*, pages 62–66. IEEE.