SBSeg 2026 Armação dos Búzios - RJ
VEJA OS SELOS DE ARTEFATOS

Extensões ao DefectDojo para Priorização de Vulnerabilidades

Francisco Aragão[1], Gabriel Pains de Oliveira Cardoso[1], Iago Silva Rios[1]
Leonardo Oliveira[1], Matheus Gimpel[1], Pedro Meireles Almeida[1], Italo Cunha[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, 31270-010, Brasil

{franciscoaragao, gabriel.cardoso, iago.rios, leonardooliveira,
matheusgimpel, pedromeireles, cunha}@dcc.ufmg.br

Abstract. *Security teams routinely face more scanner findings than they can investigate or remediate. We present a context-aware, personalized vulnerability prioritization system that combines analyst feedback with metadata about vulnerabilities, services, and detection mechanisms. We extend DefectDojo to present metadata from multiple sources about vulnerabilities imported in the system, better informing vulnerability assessment. Beyond assisting analysts, the metadata allows us to compute features for a model for prioritizing vulnerabilities, which we evaluate on vulnerabilities identified by OpenVAS on a large university network. Our results show that the proposed model quickly learns to prioritize vulnerabilities, promising a new tool to steer analyst effort.*

Resumo. *Equipes de segurança rotineiramente se deparam com mais vulnerabilidades do que conseguem investigar. Apresentamos um sistema de priorização de vulnerabilidades personalizado e sensível ao contexto que combina o* feedback *de analistas com metadados ricos. Estendemos a interface do DefectDojo para incluir metadados de diferentes fontes relacionados às vulnerabilidades carregadas no sistema, informando o processo de análise de vulnerabilidades. Além de auxiliar analistas, os metadados alimentam um modelo de priorização de vulnerabilidades, que avaliamos em vulnerabilidades identificadas pelo OpenVAS em uma rede universitária. Nossos resultados mostram que o sistema proposto aprende rapidamente a priorizar vulnerabilidades, prometendo uma nova ferramenta para direcionar o esforço dos analistas.*

1. Introdução

A comunidade de segurança reporta, verifica e cataloga novas vulnerabilidades continuamente. Ferramentas de monitoramento proprietárias e de código aberto implementam testes para identificar um número crescente de vulnerabilidades. Devido à complexidade, dinamicidade e alto nível de integração de sistemas computacionais modernos, a execução destas ferramentas geralmente identifica uma grande quantidade de vulnerabilidades, frequentemente de severidade crítica ou alta. A alta taxa de vulnerabilidades identificadas frequentemente sobrecarrega times de desenvolvedores, operadores e mesmo analistas de segurança [Tariq et al. 2025, Hassan et al. 2019], inviabilizando que entidades consigam investigar vulnerabilidades conhecidas em tempo hábil, muito menos implementar ou aplicar contramedidas. Esta situação é agravada quando entidades não conseguem alocar recursos humanos para a resolução de vulnerabilidades. Mesmo quando conseguem,

alocar os recursos limitados existentes de forma eficiente é desafiador face à quantidade de vulnerabilidades.

Instituições de qualquer tamanho, mesmo pequenas, frequentemente precisam lidar com uma quantidade de vulnerabilidades que excede a capacidade da equipe de operação. Para agravar esse cenário, diversos trabalhos mostram que varreduras realizadas por ferramentas de segurança apresentam concentração de vulnerabilidades críticas ou severas [Shimizu and Hashimoto 2026], dificultando o direcionamento de esforços de mitigação e reforçando a necessidade de mecanismos efetivos de priorização.

Propomos uma nova abordagem para a priorização de vulnerabilidades que considera o contexto da instituição onde as vulnerabilidades foram encontradas, realizando a priorização personalizada para cada instituição e analista, em vez de uma priorização genérica global. Nossa proposta é motivada por três observações. Primeiro, vulnerabilidades distintas têm impacto variado dependendo da entidade afetada. Por exemplo, uma empresa de análise de crédito pode priorizar vulnerabilidades que permitem vazamento de dados e afetam a privacidade de seus clientes, enquanto uma empresa de distribuição de conteúdo pode priorizar vulnerabilidades que afetam o desempenho ou a disponibilidade do serviço. Segundo, diferentes equipes de funcionários podem solucionar vulnerabilidades de formas distintas e com diferentes graus de eficiência. Por exemplo, um time de desenvolvedores pode melhorar a segurança atualizando a pilha de software para versões mais recentes, enquanto um time de administradores de sistemas terá maior facilidade em mitigar vulnerabilidades configurando firewalls ou políticas de acesso (ACLs). Por fim, o contexto externo à vulnerabilidade pode (des)motivar esforços imediatos de remediação. Por exemplo, vulnerabilidades recentes, de fato exploradas, com provas de conceito ou presentes em sistemas críticos devem ser tratadas com prioridade, dada a elevada chance de serem alvo de ataques e danos potenciais [Jacobs et al. 2020].

Neste trabalho propomos um modelo de inteligência artificial para priorização de vulnerabilidades encontradas por ferramentas de varredura. O modelo de priorização construído considera análises realizadas por uma equipe de segurança para inferir quais outras vulnerabilidades os analistas consideram mais importantes, permitindo o direcionamento de esforços para as vulnerabilidades de maior risco para a instituição. Nosso modelo utiliza características de vulnerabilidades capturadas de diversas bases de dados públicas, permitindo capturar propriedades dos sistemas impactados e dados envolvidos.

Integramos nossa ferramenta com o DefectDojo, sistema de gerência de vulnerabilidades de código aberto com suporte a dezenas de ferramentas de segurança. A principal funcionalidade dessa integração é uma interface para priorização de vulnerabilidades por analistas de segurança que apresenta metadados sobre as vulnerabilidades para instruir a investigação e priorização. As priorizações dos analistas são posteriormente utilizadas para treinar o modelo para priorização personalizada de vulnerabilidades.

Nossos resultados mostram que, em geral, a precisão da priorização do modelo de priorização melhora rapidamente com dezenas de análises de um analista, mas que a precisão do modelo depende significativamente do perfil do analista.

As ferramentas e os sistemas desenvolvidos estão disponíveis com a mesma licença código-aberto do DefectDojo [Aragão et al. 2025],[1] que podem contribuir com entida-

[1]Parte das nossas contribuições foram integradas ao repositório do DefectDojo. As extensões descritas

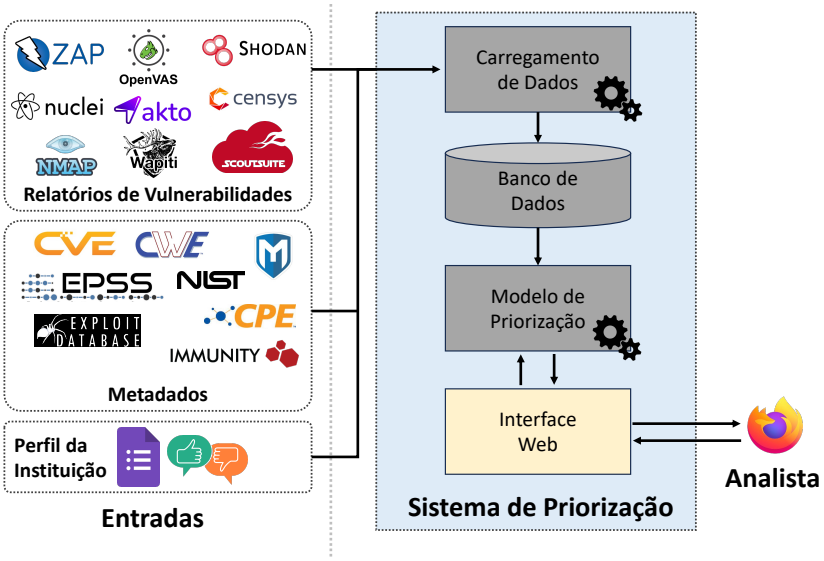

Figura 1. Visão geral da ferramenta desenvolvida.

des e equipes de segurança no direcionamento de esforços, para conseguirem mitigar um número maior de vulnerabilidades mais relevantes, obtendo a máxima melhoria de segurança dentro da capacidade operacional disponível.

2. Extensões ao DefectDojo

A Figura 1 mostra uma visão geral da ferramenta desenvolvida neste trabalho. O modelo de priorização desenvolvido utiliza como entrada relatórios de vulnerabilidade de ferramentas de monitoramento, metadados de diversas fontes, e informações sobre o perfil de analistas de segurança. Durante a utilização do sistema, analistas podem prover *feedback* manualmente priorizando vulnerabilidades através de uma interface Web; esta informação é utilizada para treinar e atualizar um modelo de priorização de vulnerabilidades, cuja saída pode ser utilizada pelo analista para direcionar seus esforços.

2.1. Requisitos

O desenvolvimento da nossa ferramenta é norteado pelos requisitos funcionais de (i) facilitar a análise de vulnerabilidades por analistas através da integração de metadados e (ii) permitir a priorização de vulnerabilidades com um modelo de aprendizado. Requisitos não funcionais considerados incluem (i) facilidade de instalação; (ii) não realizar nenhuma modificação no banco de dados do DefectDojo para permitir reversão para permitir desinstalação; e (iii) manutenabilidade, em particular através da integração de parte das modificações na versão *upstream* do DefectDojo.

2.2. Metadados, Agrupamento e Priorização de Vulnerabilidades no DefectDojo

Implementamos um sistema seguindo a arquitetura geral mostrada na Figura 1 integrando o modelo de priorização desenvolvido (Seção 3) ao DefectDojo, uma ferramenta de código

neste documento foram submetidas para inclusão, mas não foram aceitas. Parte das extensões e funcionalidades apresentadas nesta ferramenta estão disponíveis na versão proprietária (Pro) do DefectDojo, disponível mediante assinatura do serviço.

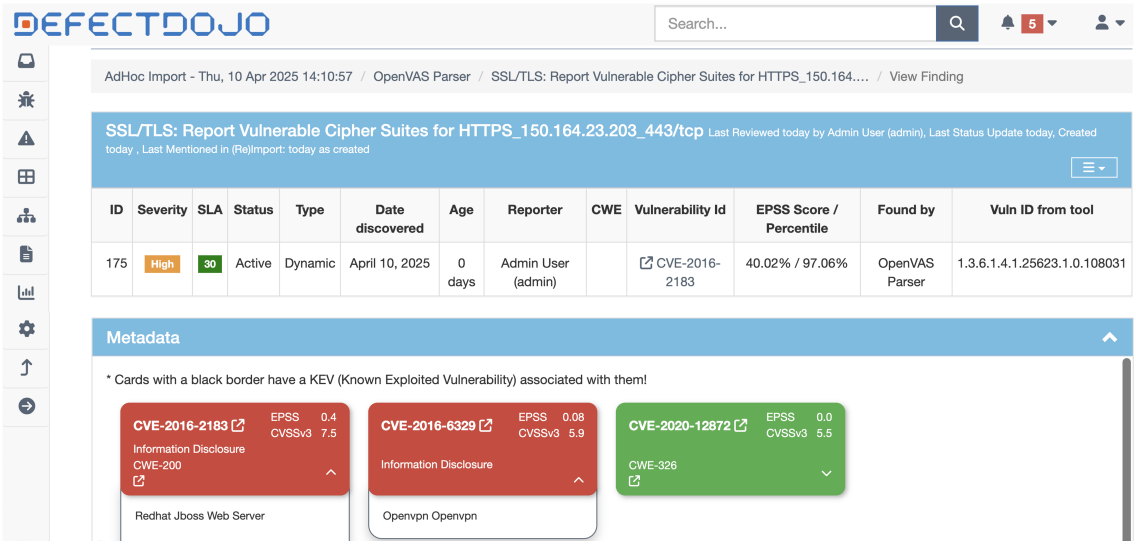

Figura 2. Modificações na interface do DefectDojo para apresentar metadados adicionais sobre vulnerabilidades para analistas de segurança.

aberto para gerência de vulnerabilidades. O DefectDojo tem integrações com dezenas de ferramentas, mas neste trabalho utilizamos primariamente a integração do DefectDojo com o OpenVAS, um sistema de varredura de vulnerabilidades. Utilizamos o OpenVAS para buscar vulnerabilidades em uma rede universitária, que utilizamos para análises e avaliação do modelo proposto.

A primeira funcionalidade que adicionamos ao DefectDojo foi a capacidade de carregar metadados sobre vulnerabilidades. Mais especificamente, nossas extensões fazem download das bases do CVE (que inclui informações sobre CPE e CVSS), CWE, EPSS e KEV. A Figura 2 ilustra as modificações que fizemos na interface do DefectDojo para mostrar essas informações para os usuários.

A segunda funcionalidade que adicionamos ao DefectDojo foi a capacidade de agrupar vulnerabilidades similares. Esta funcionalidade facilita a gerência de vulnerabilidades similares, e permite a analistas de segurança obterem uma visão geral mais precisa das vulnerabilidades em sua rede. Por exemplo, em uma varredura realizada em uma rede universitária, um subconjunto de 1000 vulnerabilidades foram agrupadas em apenas 39 grupos distintos, alguns com centenas de vulnerabilidades.

Por fim, estendemos a interface do DefectDojo para permitir que analistas atribuam um risco a uma vulnerabilidade. O risco atribuído por analistas é armazenado em um banco de dados e posteriormente utilizado para treinar modelos de priorização. Após treinados, os modelos de priorização podem ser utilizados para priorizar automaticamente outras vulnerabilidades que ainda não foram analisadas pelo analista. A Figura 3 mostra uma captura de tela do *drop-down* para atribuição de criticidade.

Além disso, notamos que nossas modificações ao DefectDojo são retrocompatíveis com a versão original: operadores podem fazer upgrade e downgrade entre nossa versão estendida e a versão oficial do DefectDojo sem perda de informações, necessidade de reconfiguração da plataforma, ou modificação do banco de dados da ferramenta. Esta funcionalidade foi um requisito específico da implementação para facilitar a avaliação

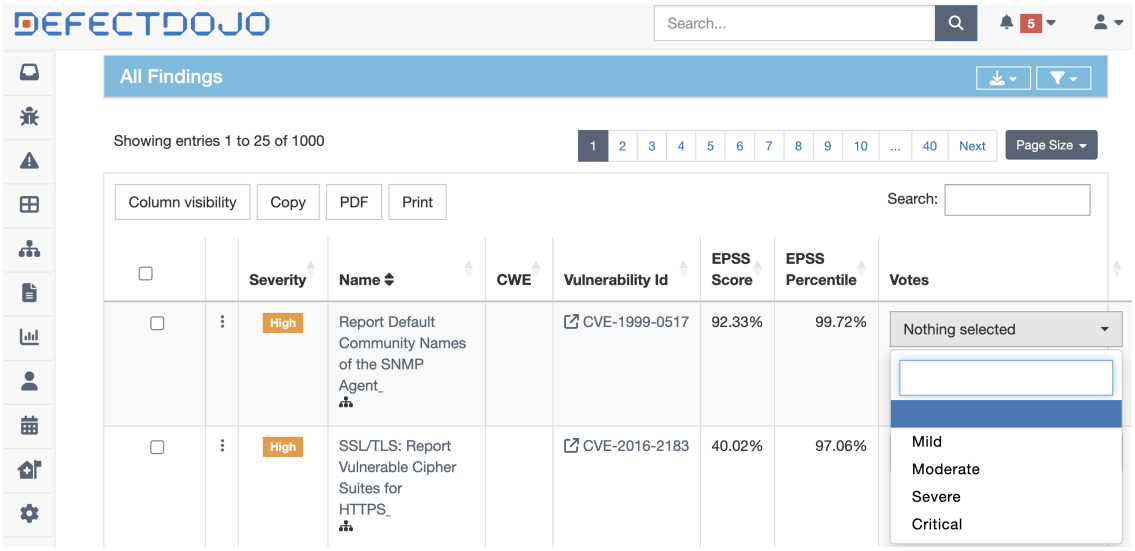

Figura 3. Modificações na interface do DefectDojo para permitir atribuição de risco a uma vulnerabilidade e posterior treino do modelo de priorização.

das extensões implementadas.

3. Priorização de Vulnerabilidades

A priorização de vulnerabilidades é um desafio central em processos de gestão de riscos de segurança, especialmente diante do grande volume de vulnerabilidades identificadas por ferramentas de varredura. Nesta seção, apresentamos a abordagem desenvolvida para priorizar vulnerabilidades de forma eficiente e adaptada ao contexto institucional.

3.1. *Features* para Priorização de Vulnerabilidades

Para instruir a priorização de vulnerabilidades, extraímos diversas propriedades (*features*) das próprias vulnerabilidades, bem como do contexto da instituição e sobre o analista. Agregamos metadados de diversas fontes aos dados dos sistemas de varredura. Abaixo descrevemos as *features* que calculamos para treinar o modelo de priorização. Indicamos as *features* categóricas com um "c", as *features* numéricas com um "n" e as *features* capturadas por um vetor de variáveis com "v":

Informações sobre a gravidade da vulnerabilidade. Número de CVEs (n); número de CVEs com vetor de ataque pela rede (n); número de CVEs considerados críticos (n); número de CVEs considerados graves (n); máximo, soma, média e mediana dos escores CVSS e EPSS para o conjunto de CVEs (v); presença no KEV (binário); existência de exploit no ExploitDB (binário); mecanismo de detecção de vulnerabilidade (c).

Informações sobre o serviço. Sistema operacional (c); fabricante do dispositivo (c); número de nomes de domínio conhecidos associados ao dispositivo (n); classificação de dinamicidade do endereço IP (c) [de Oliveira Cardoso et al. 2024]. O sistema operacional e fabricante podem ser extraídos da base do CPE ou do *script* de detecção da vulnerabilidade, mas são frequentemente desconhecidos.

Informações sobre o analista. Identificador do usuário (c), formação (c), anos de experiência (n), departamento ou setor de atuação (c), que podem ser coletados no momento de cadastro do analista.

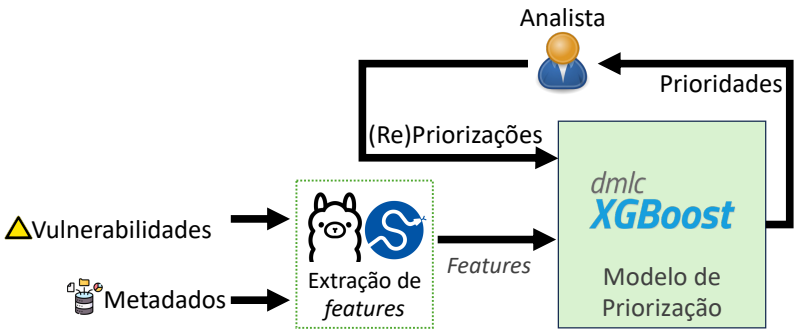

Figura 4. Visão geral do processo de treino e utilização do modelo de priorização de vulnerabilidades.

3.2. Treino do Modelo de Priorização

Consideramos o problema de priorização de vulnerabilidades como um problema de regressão. Construímos conjuntos de dados para treino onde extraímos *features* de vulnerabilidades (Seção 3.1) e associamos estas *features* a uma avaliação de gravidade feita por um analista, no contexto da instituição onde a vulnerabilidade foi encontrada. Avaliamos soluções onde analistas podem atribuir um risco a vulnerabilidades em uma escala numérica de 1 a 10 ou através da classificação de uma vulnerabilidade como leve, moderada, severa ou crítica. Para as atribuições de risco por classificação, convertemos cada classe em um valor numérico para transformar o problema de priorização em um problema de regressão. Em particular, convertemos classificações leve, moderada, severa ou crítica para os valores 1, 2, 3 e 4, respectivamente, preservando a ordem entre as classes, mas sem fazer ajustes de peso. Diferentes implantações podem fazer ajustes nos valores; p.ex., para reduzir a prioridade de vulnerabilidades classificadas como leves e aumentar a prioridade de vulnerabilidades classificadas como severas, as classes poderiam ser convertidas para os valores 1, 4, 8 e 10.

Optamos por utilizar o XGBoost [Wang et al. 2019] para realizar as predições de prioridade, mas ele pode ser substituído por outras soluções. O XGBoost é amplamente utilizado, indicado para problemas de regressão, suporta variáveis numéricas e categóricas, é resistente a *overfitting* devido à regularização incorporada, mantém alto desempenho mesmo com conjuntos de dados relativamente pequenos ou desbalanceados, permite interpretação do modelo através da importância das *features* na priorização, e sua eficiência computacional possibilita atualizações frequentes do modelo à medida que avaliações são realizadas por analistas.

A Figura 4 mostra a visão geral do treino e uso do modelo de priorização. O modelo de priorização é treinado utilizando as priorizações realizadas por analistas, mais as *features* calculadas das vulnerabilidades e metadados. As *features* são calculadas com diferentes níveis de complexidade (desde simples derivações numéricas utilizando a biblioteca pandas do Python até modelos de linguagem executando localmente via llamacpp). Após treinado, o modelo é utilizado para priorizar outras vulnerabilidades no sistema que ainda não foram analisadas por analistas. Há um ciclo entre as priorizações do modelo e análise dos analistas, o que permite ao analista focar seus esforços nas vulnerabilidades mais prioritárias e refinar o modelo através da repriorização, p.ex., no caso de o modelo ter superestimado ou subestimado a prioridade de uma vulnerabilidade.

Realizamos uma avaliação do modelo desenvolvido, que omitimos deste documento por restrições de espaço e tópico. Nossa avaliação utilizou vulnerabilidades reais de uma rede universitária e vulnerabilidades de redes brasileiras sondadas pelo Shodan, e centenas de análises de vulnerabilidades de sistemas em rede pelos autores e participantes do programa Hackers do Bem. Nossos resultados mostram que, em geral, a precisão do modelo de priorização melhora rapidamente com dezenas de análises de um analista, e que a precisão do modelo depende significativamente do perfil do analista.

3.3. Limitações

Uma limitação da solução proposta é a necessidade de análise de vulnerabilidades pelo sistema por analistas de segurança. Nossas análises demonstram que os modelos ficam mais precisos com o aumento de análises na base de treino. Notamos ainda que a análise das vulnerabilidades é uma atividade intrínseca já realizada continuamente por analistas de segurança, então a utilização da nossa solução não traz sobrecarga operacional além dos poucos cliques para informar ao sistema a gravidade de uma vulnerabilidade após sua análise. Por fim, uma análise em maior escala, com vulnerabilidades em diferentes contextos e com analistas mais experientes pode permitir refinamento da abordagem proposta.

4. Trabalhos Relacionados

Metadados para cibersegurança. A comunidade de cibersegurança mantém diversas fontes de metadados como os catálogos de vulnerabilidades, fraquezas e produtos conhecidos (CVE, CWE e CPE, respectivamente), os índices de severidade CVSS e EPSS; e bases de vulnerabilidades conhecidas como o ExploitDB.

Neste trabalho construímos bases para instruir a análise de vulnerabilidades por analistas ou para o treinamento de modelos de predição e inferência. Esta frente é complementar a vários outros trabalhos que também propõem a extração de metadados de fontes alternativas como código-fonte, registros de invasões e vazamento de dados [Ponta et al. 2018, Khanfir et al. 2022, Zadeh et al. 2023]. O treino e avaliação de modelos de classificação e inferência no contexto de cibersegurança, p.ex., para priorização de vulnerabilidades, frequentemente utiliza metadados para capturar características (*features*) úteis [Jimenez et al. 2019, Croft et al. 2023], com os quais nossos mecanismos contribuem.

IA para cibersegurança. Vários trabalhos aplicam aprendizado de máquina e IA em cibersegurança, com interesse recente em soluções assistidas por grandes modelos de linguagem [Jaffal et al. 2025, Rashid et al. 2026, Zhou et al. 2024]. Mais relacionados ao nosso trabalho são estudos sobre qualidade de dados, seleção de *features* e gerência de vulnerabilidades [Ahsan et al. 2021, Croft et al. 2023, Shimizu and Hashimoto 2026]. Neste contexto, nosso trabalho complementa uma ferramenta existente integrando metadados ricos para auxiliar a análise e priorização de vulnerabilidades.

Nosso trabalho tem objetivo distinto, mas não menos importante. Em geral, enriquecimento de dados com uma maior quantidade de características (*features*) permite o treino de modelos de inteligência artificial mais sofisticados, porém, a seleção de características é importante para evitar o crescimento da dimensionalidade e complexidade

computacional do modelo. Estudos na literatura já demonstraram que é possível reduzir significativamente a quantidade de características utilizadas em modelos preditivos no contexto de cibersegurança ao mesmo tempo em que a precisão do modelo é mantida ou até mesmo melhorada [Ahsan et al. 2021].

5. Demonstração da Ferramenta

Durante o evento iremos demonstrar uma versão operacional da ferramenta desenvolvida em um computador local. Os participantes do SBSeg poderão usar a ferramenta e interagir com as funcionalidades desenvolvidas. Também iremos executar o DefectDojo com a mesma base de dados, mas sem as extensões desenvolvidas, permitindo que os participantes possam comparar diretamente as diferenças entre as versões. Dentre as funcionalidades disponíveis estão a navegação pelas vulnerabilidades, com todos os metadados associados, e as funcionalidades adicionadas para priorização de vulnerabilidades, que pode ser feita manualmente pelo analista ou automaticamente pelo modelo de priorização.

Durante a demonstração iremos apresentar não só as funcionalidades, mas os esforços que fizemos para facilitar a instalação (e desinstalação) das nossas modificações, desde o passo inicial de obtenção dos metadados adicionais suportados pelas nossas extensões, até o treino e execução do modelo de priorização de vulnerabilidades.

Iremos demonstrar o sistema de priorização utilizando diferentes perfis no DefectDojo. Iremos criar e pré-popular perfis com priorizações distintas para demonstrar como o modelo se ajusta ao perfil do analista. Participantes do SBSeg poderão autenticar-se com um perfil diferente único. Como a atualização do modelo XGBoost e priorização de vulnerabilidades é eficiente, um participante interessado pode analisar algumas vulnerabilidades para ver como o modelo se ajustaria ao seu perfil.

Um vídeo demonstrando a utilização das extensões desenvolvidas está disponível no Youtube: https://tinyurl.com/dojoprio.

Além da ferramenta, também podemos discutir a avaliação que fizemos do modelo, que não foi incluída neste documento por restrições de espaço e tópico. Em particular podemos discutir os desafios em conseguir dados sobre vulnerabilidades para análise e treino dos modelos de priorização, diferentes metadados que construímos especificamente para informar a análise de vulnerabilidades e complementar o conjunto de *features* utilizadas pelo modelo de priorização, e diferenças de precisão do modelo em função de propriedades das vulnerabilidades e perfil dos analistas.

6. Conclusão

Neste trabalho desenvolvemos uma ferramenta para auxiliar em um dos desafios mais críticos da cibersegurança moderna: a priorização de vulnerabilidades em ambientes organizacionais complexos. Criamos modelos de aprendizado de máquina baseados em XGBoost que consideram não apenas as características técnicas das vulnerabilidades, mas também o perfil dos analistas. Integramos dados de múltiplas fontes públicas (CVE, EPSS, KEV, ExploitDB, entre outras) para criar um conjunto abrangente de características que instruem a análise dos analistas e o treino do modelo de priorização. Nossas soluções foram integradas ao DefectDojo, uma ferramenta de código-aberto com integrações a centenas de ferramentas de segurança. Os artefatos desenvolvidos são de acesso aberto às comunidades de pesquisa e de analistas de segurança.

Agradecimentos

O presente trabalho foi realizado com o apoio do Programa Hackers do Bem no Domínio Cibernético, executado pela RNP e SENAI, coordenado pela Softex e financiado pelo MCTI com recursos oriundos da Lei das TICs (Lei 8.248 de 23 de outubro de 1991). Ítalo Cunha e alunos do DCC/UFMG também são auxiliados pela CAPES, CNPq, FAPEMIG e FAPESP.

Referências

Ahsan, M., Gomes, R., Chowdhury, M. M., and Nygard, K. E. (2021). Enhancing Machine Learning Prediction in Cybersecurity Using Dynamic Feature Selector. *Journal of Cybersecurity and Privacy*, 1(1):199–218.

Aragão, F., de Oliveira Cardoso, G. P., Rios, I., Oliveira, L., Gimpel, M., Almeida, P., and Cunha, I. (2025). Extensões ao DefectDojo para Priorização de Vulnerabilidades. https://git.rnp.br/gt-crivo/django-defectdojo/-/blob/crivo-rnp/readme-docs/GT-CRIVO.md.

Croft, R., Babar, M. A., and Kholoosi, M. M. (2023). Data Quality for Software Vulnerability Datasets. In *Proc. IEEE/ACM Intl. Conf. on Software Engineering (ICSE)*.

de Oliveira Cardoso, G. P., Oliveira, L. B., and Cunha, I. (2024). Identificação de Endereços IP Dinâmicos com Dados Públicos. In *Anais SBSeg*.

Hassan, W. U., Guo, S., Li, D., Chen, Z., Jee, K., Li, Z., and Bates, A. (2019). NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

Jacobs, J., Romanosky, S., Adjerid, I., and Baker, W. (2020). Improving Vulnerability Remediation Through Better Exploit Prediction. *Journal of Cybersecurity*.

Jaffal, N. O., Alkhanafseh, M. Y., and Mohaisen, D. (2025). Large Language Models in Cybersecurity: Applications, Vulnerabilities, and Defense Techniques. *AI*.

Jimenez, M., Rwemalika, R., Papadakis, M., Sarro, F., Le Traon, Y., and Harman, M. (2019). The Importance of Accounting for Real-World Labelling When Predicting Software Vulnerabilities. In *Proc. ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.

Khanfir, A., Jimenez, M., Papadakis, M., and Traon, Y. L. (2022). CodeBERT-nt: Code Naturalness via CodeBERT. In *2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)*.

Ponta, S. E., Plate, H., and Sabetta, A. (2018). Beyond Metadata: Code-Centric and Usage-Based Analysis of Known Vulnerabilities in Open-Source Software. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*.

Rashid, M. B., Hossain, M. S. J., Khan, M. I., Tahora, S., Siddika, A., Prakash, M. I., Yeasmin, S., and Shahriar, H. (2026). A Survey on Large Language Models in Software Security: Opportunities and Threats. *Computers*.

Shimizu, N. and Hashimoto, M. (2026). Vulnerability Management Chaining: An Integrated Framework for Efficient Cybersecurity Risk Prioritization. *IEEE Access*, 14:31407–31424.

Tariq, S., Chhetri, M. B., Nepal, S., and Paris, C. (2025). Alert Fatigue in Security Operations Centres: Research Challenges and Opportunities. *ACM Computing Surveys*.

Wang, P., Zhou, Y., Sun, B., and Zhang, W. (2019). Intelligent Prediction of Vulnerability Severity Level Based on Text Mining and XGBboost. In *Intl. Conf. on Advanced Computational Intelligence*.

Zadeh, A., Lavine, B., Zolbanin, H., and Hopkins, D. (2023). A Cybersecurity Risk Quantification and Classification Framework for Informed Risk Mitigation Decisions. *Decision Analytics Journal*, 9:100328.

Zhou, X., Zhang, T., and Lo, D. (2024). Large Language Model for Vulnerability Detection: Emerging Results and Future Directions. In *IEEE/ACM International Conference on Software Engineering (ICSE)*.