

PREDIÇÃO DE MUDANÇAS DE SOFTWARE COM IMPACTO EM SEGURANÇA

Arthur Velho Cristiano, Paulo Roberto Farah.

INTRODUÇÃO

A crescente complexidade dos sistemas de software torna a identificação de vulnerabilidades uma tarefa relevante para a engenharia de software. Inspeções manuais apresentam custo elevado e dificuldade de escala, enquanto técnicas automatizadas podem auxiliar na identificação de métodos com maior probabilidade de vulnerabilidade. A análise em nível de método permite localizar com maior precisão os componentes críticos e reduzir o esforço de revisão.

Métricas estruturais, como complexidade, acoplamento e coesão, podem indicar componentes mais propensos a vulnerabilidades (CHOWDHURY; ZULKERNINE, 2010, Farah et al., 2023). Sultana et al. (2021) utilizaram essas métricas com Support Vector Machine (SVM) e Logistic Regression (LR), obtendo resultados promissores em projetos Java. Paralelamente, Large Language Models (LLMs) permitem classificar código diretamente, inclusive em configuração zero-shot, embora estudos apontem limitações relacionadas a falsos positivos, prompts e contexto disponível (CHENG et al., 2024; ULLAH et al., 2023).

Este trabalho replica o estudo de Sultana et al. (2021) nas versões 9, 10 e 11 do Apache Tomcat e o estende com a avaliação do Gemini 3.1 Pro. Busca-se verificar a consistência dos resultados da pesquisa original, avaliar o LLM na predição de vulnerabilidades em nível de método e comparar as abordagens, considerando suas diferenças experimentais.

DESENVOLVIMENTO

A pesquisa foi estruturada como replicação e extensão da metodologia de Sultana et al., com três questões: verificar a consistência da replicação; avaliar a efetividade de LLMs na predição de vulnerabilidades; e comparar o LLM com modelos tradicionais baseados em métricas. O estudo concentrou-se no Apache Tomcat 9, 10 e 11, devido à disponibilidade do código-fonte e dos boletins de segurança. Apache CXF e Stanford SecuriBench foram excluídos por dificuldades de reprodução do ambiente original.

A identificação das vulnerabilidades foi realizada a partir dos boletins oficiais de segurança do Apache Tomcat (Apache Software Foundation, 2026). Para cada boletim, foram consultados o identificador CVE, as versões afetadas e o commit correspondente à correção. Os commits foram então inspecionados manualmente, comparando-se o código anterior e posterior à correção para identificar as classes e os métodos modificados especificamente para tratar a vulnerabilidade reportada. Os métodos alterados no código anterior à correção foram associados à vulnerabilidade e classificados como vulneráveis, enquanto as versões corrigidas desses mesmos métodos foram identificadas como referência de código corrigido. Para os modelos de aprendizado de máquina, 1ANO: 2026 métodos da versão mais recente mantida de cada ramo do Tomcat foram considerados não vulneráveis. Assim, a rotulagem foi estabelecida pela relação entre os boletins de segurança, os commits de correção e os métodos efetivamente modificados.

Cada método foi representado por métricas e classe binária (1 vulnerável; 0 não vulnerável). Foram utilizados SVM e LR, substituindo-se o WEKA original por Pandas e Scikit-learn. Devido ao desbalanceamento, aplicou-se RandomUnderSampler seguido de SMOTE exclusivamente nos dados de treinamento de cada fold da validação cruzada de 10 folds. A avaliação utilizou Precision, Recall, F1-Score e ROC AUC.

Para a extensão com LLM, utilizou-se Gemini 3.1 Pro em configuração zero-shot e temperatura zero. O prompt continha instrução de análise de segurança, código do método e solicitação de classificação binária. Inicialmente foram usados métodos não vulneráveis arbitrários como negativos, mas a estratégia gerou muitos falsos positivos. Por isso, foram adotadas as versões corrigidas dos próprios métodos vulneráveis, formando pares pré-fix e pós-fix e solicitando a verificação específica da CWE tratada pelo commit.

Foram analisados 116 pares no Tomcat 9, 117 no Tomcat 10 e 64 no Tomcat 11. Como o LLM forneceu somente respostas binárias, sem probabilidades ou escores de confiança, não foi possível construir ROC convencional; utilizou-se Balanced Accuracy. Assim, SVM e LR foram avaliados por validação cruzada e escores contínuos, enquanto o LLM foi avaliado por classificação zero-shot de pares pré-fix e pós-fix. Essa diferença limita comparações diretas entre métricas dependentes de limiares.

RESULTADOS

Os resultados, foram motivados pelas perguntas de pesquisa apresentadas abaixo.

1. RQ1: Em que medida os resultados do estudo de replicação são consistentes com os apresentados por Sultana et al. no estudo original?

O SVM apresentou Precision, Recall, F1-Score e ROC AUC de 0,8617, 0,8566, 0,8592 e 0,9255 no Tomcat 9; 0,8819, 0,8301, 0,8552 e 0,9263 no Tomcat 10; e 0,8798, 0,8216, 0,8497 e 0,9297 no Tomcat 11. A LR apresentou F1-Scores de 0,7946, 0,8236 e 0,8235, respectivamente, ficando abaixo do SVM. Os resultados são consistentes com Sultana et al. (2021), cujos F1-Scores variaram de 0,76 a 0,89 em diferentes projetos e versões. A estabilidade do SVM também foi observada entre as versões: o F1-Score caiu de 0,8592 para 0,8497, enquanto o ROC AUC aumentou de 0,9255 para 0,9297. Sem testes estatísticos, essa diferença não permite conclusões sobre significância.

2. RQ2: Quão eficazes são os Large Language Models (LLMs) na predição de vulnerabilidades de software em nível de método?

O Gemini 3.1 Pro apresentou, respectivamente nos Tomcats 9, 10 e 11, Precision de 0,9278, 0,9481 e 0,9091; Recall de 0,9574, 0,7526 e 0,9091; F1-Score de 0,9424, 0,8391 e 0,9091; e Balanced Accuracy de 0,8196, 0,7763 e 0,8545. A Precision elevada indica baixa ocorrência de falsos positivos entre as classificações positivas, mas o Recall variou consideravelmente, sobretudo no Tomcat 10. Nos métodos corrigidos, o modelo classificou incorretamente 7 de 22 casos no Tomcat 9, 4 de 20 no Tomcat 10 e 4 de 20 no Tomcat 11 como ainda vulneráveis. Entre os vulneráveis, classificou incorretamente 4 de 94 no Tomcat 9, 24 de 97 no Tomcat 10 e 4 de 44 no Tomcat 11 como não contendo mais a CWE. Os resultados indicam potencial do LLM, mas também sensibilidade às características das vulnerabilidades e ao contexto disponível.

RQ3: Como o desempenho da predição de vulnerabilidades dos Large Language Models (LLMs) se compara com a abordagem de aprendizado de máquina proposta por Sultana et al. para predição de vulnerabilidades em nível de método?

O LLM obteve a maior Precision nas três versões, variando de 0,9091 a 0,9481, enquanto SVM variou de 0,8617 a 0,8819 e LR de 0,8606 a 0,8768. Em Recall, o LLM foi superior no Tomcat 9 e 11, mas inferior ao SVM no Tomcat 10. Em F1-Score, o LLM foi superior nos Tomcats 9 e 11, enquanto o SVM obteve o melhor resultado no Tomcat 10. Portanto, não é adequado afirmar superioridade global de uma abordagem. O LLM apresentou vantagem em Precision e maiores F1Scores em duas versões, enquanto o SVM demonstrou maior estabilidade. Os resultados sugerem complementaridade e justificam investigar estratégias híbridas, nas quais modelos tradicionais realizem uma filtragem inicial e o LLM

faça uma análise posterior mais detalhada. A comparação, contudo, deve ser interpretada de forma indicativa devido aos protocolos distintos e à ausência de testes estatísticos.

CONSIDERAÇÕES FINAIS

O trabalho replicou e estendeu o estudo de Sultana et al. nas versões 9, 10 e 11 do Apache Tomcat, avaliando SVM e LR com métricas de software e Gemini 3.1 Pro em zero-shot. A replicação confirmou a utilidade das métricas para apoiar a identificação de métodos vulneráveis, com SVM superior à LR em todas as versões, F1-Scores acima de 0,84 e ROC AUC acima de 0,92.

O Gemini 3.1 Pro apresentou a maior Precision nas três versões e os maiores F1-Scores em duas delas, mas seu Recall variou, principalmente no Tomcat 10. Assim, Machine Learning e LLMs apresentam características complementares. Como trabalhos futuros, destacam-se a obtenção de escores de confiança calibrados, a investigação de few-shot e fine-tuning, a ampliação para outros projetos e linguagens e a realização de testes estatísticos

Palavras-chave: vulnerabilidades, métricas, segurança, machine learning, LLM

REFERÊNCIAS BIBLIOGRÁFICAS

APACHE SOFTWARE FOUNDATION. Apache Tomcat 11 vulnerabilities. 2026. Disponível em: <https://tomcat.apache.org/security-11.html>. Acesso em: 17 ago. 2026.

CHENG, Xin et al. Large language model for vulnerability detection: emerging results and future directions. In: ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), 2024.

CHOWDHURY, Istehad; ZULKERNINE, Mohammad. Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities. In: ACM SYMPOSIUM ON APPLIED COMPUTING. Proceedings [...]. p. 1140–1141, 2010.

FARAH, Paulo Roberto; SILVA, Rogério; VERGILIO, Silvia. An assessment of machine learning algorithms and models for prediction of change-prone Java methods. In: PROCEEDINGS OF THE XXXVII BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, 2023, p. 322–331. New York: Association for Computing Machinery, 2023. DOI: 10.1145/3613372.3613395.

SULTANA, Kazi Zakia; ANU, Vaibhav; CHONG, Tai Yin. Using software metrics for predicting vulnerable classes in Java and Python based systems. Journal of Software: Evolution and Process, 2024.

ULLAH, Saad; HAN, Mingji; PUJAR, Saurabh; PEARCE, Hammond; COSKUN, Ayse; STRINGHINI, Gianluca. LLMs cannot reliably identify and reason about security vulnerabilities (yet?): a comprehensive evaluation, framework, and benchmarks. arXiv preprint, arXiv:2312.12575, 2023.