Verificação Formal de Software para Internet das Coisas com Validação de Aplicações em Kotlin/Java: Mapeamento Sistemático Preliminar

Evelim B. Rocha[1], **Vandermi J. da Silva**[1], **Andrey Rodrigues**[1]

[1]Instituto de Ciências Exatas e Tecnologia
Universidade Federal do Amazonas (UFAM) – Itacoatiara – AM – Brasil

evelim.rocha@ufam.edu.br, vandermi@ufam.edu.br

andrey.rodrigues@ufam.edu.br

Resumo. *Esse projeto destina-se à utilização de técnicas de verificação formal em bytecodes Kotlin/Java para identificação de deadlocks, memory leak, arithmetic overflow e data race em aplicações IoT. Para isso, será utilizada a ferramenta de verificação ESBMC-Jimple em uma aplicação Android para detectar possíveis vulnerabilidades e tratá-las visando à segurança e melhoria do software.*

1. Introdução

O uso de dispositivos com poder computacional e acesso à Internet é cada vez mais comum no dia a dia [Rowland et al. 2015]. Com o avanço da tecnologia a Internet das coisas (do inglês "Internet of Things" ou IoT), vem colaborando para trazer conforto ao cotidiano das pessoas, por exemplo, na área industrial e tecnológica corroborando para o crescimento do uso desses dispositivos inteligentes interconectados [Moraes and Hayashi 2021]. Nos dias atuais, a quantidade de *software* em produtos embarcados tem aumentado exponencialmente de tal modo que a verificação formal desempenha um papel importantíssimo para assegurar a qualidade do produto.

Vários métodos permitem a verificação formal de como os aplicativos se comportam com comportamentos imprevistos, como: estouros aritméticos, bloqueios fatais e execuções de dados durante a fase de teste [SILVA et al. 2013]. Uma delas é o Bounded Model Checking (BMC), que é uma técnica de verificação formal que pode verificar erros de implementação em *software* [Biere et al. 2009]. A ideia básica da técnica BMC é verificar a negação de uma dada propriedade em uma dada profundidade. Uma força significativa do BMC é analisar apenas execuções de programas limitados, alcançando assim a decibilidade [Monteiro et al. 2022], buscando por violação durante a execução de um programa de computador. No entanto, ainda existe a necessidade de um maior desenvolvimento dessas ferramentas.

Alguns verificadores suportam verificação de *bytecode* Java (por exemplo, JBMC [Cordeiro et al. 2018] e JayHorn [Kahsai et al. 2019]). No entanto, possuem deficiência ao verificar *bytecodes* Kotlin devido as limitações de modelos e asserções ausentes. Para que a identificação mais precisa em Kotlin fosse possível, os autores apresentados em [Menezes et al. 2022] discorrem sobre uma nova ferramenta de verificação de *software* combinando o BMC — ESBMC, com sua extensão para verificação de programas em Kotlin, "Jimple"— e técnicas de *fuzzing* para detectar problemas como estouro

de memória e vulnerabilidades de simultaneidade em protocolos criptográficos de IoT e implementações simultâneas, a fim de identificar possíveis erros de *software*.

Dessa forma, esta pesquisa pretende retratar a verificação formal de aplicações Android focadas em IoT desenvolvidas em Kotlin/Java, a fim de identificar erros de segurança e memória como *deadlock*, *memory leak*, *arithmetic overflow* e *data race*. É esperado obter dados com base nas literaturas e verificações realizadas durante o período de estudos nas aplicações mencionadas para concluir a eficácia do verificador formal estudado, visando à melhoria do *software* em IoT codificado nas linguagens especificadas.

2. Objetivo

O principal objetivo de um mapeamento sistemático é prover uma visão geral de uma área de pesquisa, identificando a quantidade, o tipo de pesquisa e os resultados disponíveis dentro dessa área [Petersen et al. 2008]. O objetivo deste mapeamento sistemático está estruturado de acordo com o paradigma GQM (Goal- Question-Metric) definido na Figura 1.

Analisar as	publicações científicas
com o propósito de	mapear ferramentas de verificação formal
com relação a	linguagem Kotlin
do ponto de vista dos	pesquisadores
no contexto de	Internet das Coisas

Figura 1. Objetivo do MSL segundo paradigma GQM.

De acordo com o objetivo apresentado, as questões de pesquisa desse mapeamento sistemático estão apresentadas na Figura 2.

QP	Quais são as ferramentas de verificação formal que possuem compatibilidade com a linguagem *Kotlin*?
SQ1	As ferramentas de verificação formal para *Java* são compatíveis com *Kotlin*?
SQ2	Quais dessas ferramentas identificam problemas de segurança e memória como *deadlock, memory leak, data race* e *arithmetic overflow*?

Figura 2. Questão de pesquisa principal e secundárias do MSL.

3. Estratégia de busca dos estudos

A estratégia de busca incluiu a definição da string de busca a ser utilizada nas fontes de buscas selecionadas. As strings de busca são:

(”java programming language”**OR** ”kotlin programming language”) **AND**

(”formal verification kotlin”**OR** “formal verification java” **OR** ”formal verification java kotlin”).

4. Critérios para seleção dos estudos

Os critérios de seleção servem para definir se um estudo será incluído ou excluído do mapeamento sistemático. Neste mapeamento, esses critérios foram divididos em dois filtros. Os critérios de seleção do primeiro filtro estão apresentados na 3.

Critério	Descrição
CI.1	O título e/ou resumo da publicação descreve uma ferramenta de verificação formal para Kotlin ou Java.
CE.1	A publicação não atende ao critério de inclusão.

Figura 3. Critérios de inclusão (CI) e exclusão (CE) para o primeiro filtro.

5. Estratégia para extração dos dados

Após a seleção das publicações com aplicação da estratégia de seleção descrita anteriormente, os dados dessas publicações foram extraídos. Os dados extraídos neste mapeamento foram classificados em dados da publicação e dados específicos do contexto da publicação. Os dados específicos da publicação extraídos estão descritos na figura 4.

Código	Número que identifica a publicação extraída
Ano de Publicação	Ano em que foi publicado
Fonte	Veículo em que foi publicado
Título	Título da publicação
Autores	Os autores da publicação

Figura 4. Dados específicos da publicação.

Na figura 5 estão listados os dados específicos extraídos em relação à utilização da métrica de produtividade por parte do pesquisador na publicação científica. Os dados de contexto são importantes, pois as métricas definidas para um contexto normalmente não são aplicáveis a outros contextos.

6. Resultados Preliminares

Ao final do processo foram encontradas 408 publicações na biblioteca digital ACM, 299 publicações na biblioteca digital IEEE Xplore e 47 publicações na biblioteca digital Scopus, gerando um total de 754 publicações encontradas. Após a remoção das publicações duplicadas, o total de publicações selecionadas para filtragem ficou em 518. Dessas 518 publicações, 497 não atendiam ao critério de inclusão do primeiro filtro e, por isso, foram

Ferramentas de verificação	**A ferramenta de verificação formal utilizada na publicação**
Descrição	Descrição da ferramenta de verificação da publicação.
Avaliação Experimental	Se a publicação apresenta alguma avaliação experimental.
Métricas utilizadas	Caso tenha o item acima, mencionar quais métricas foram utilizadas.

Figura 5. Dados específicos do contexto da publicação.

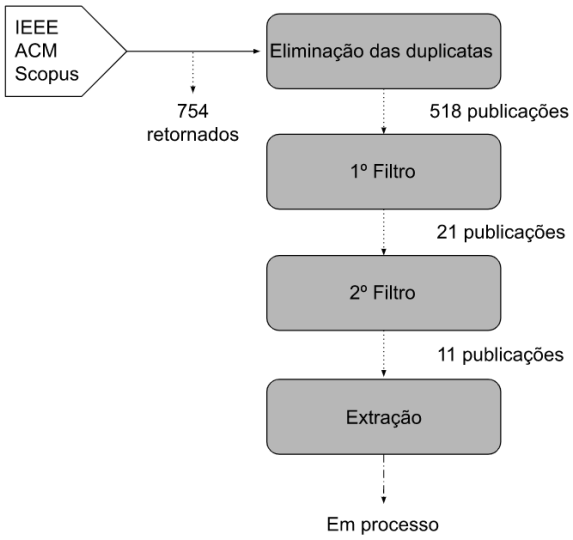

Figura 6. Resultado preliminar do processo de seleção

excluídas. As 21 publicações restantes foram lidas integralmente e, no final da seleção, somente 11 publicações atenderam aos critérios do segundo filtro (Figura 6).

O processo de extração ainda encontra-se em processo e em análise pelo orientador. Até o presente momento, com as métricas coletadas de artigos iniciais, os verificadores formais para Java, em sua maioria, apresentam implicitamente compatibilidade com Kotlin. No entanto, eles apresentam deficiências na verificação do Kotlin devido às limitações de modelos e asserções ausentes [Menezes et al. 2022], por exemplo, a inicialização do array no Kotlin depende de seu SDK para a implementação.

7. Contribuições esperadas

É esperado que essa pesquisa contribua com a área de verificação formal, dê visibilidade para esse nicho e a devida importância. Despertando a importância do desenvolvimento de ferramentas que suportam nativamente a linguagem Kotlin, assunto pouco explorado, e garantir a qualidade e relevância das evidências encontradas para que a pesquisa futura sobre o assunto seja baseada em informações confiáveis e bem fundamentadas. Também espera-se que este mapeamento sirva de suporte para a revisão sistemática do PIBIC que leva o mesmo nome desta pesquisa.

Referências

Biere, A., Cimatti, A., Clarke, E. M., Strichman, O., and Zhu, Y. (2009). Bounded model checking. *Handbook of satisfiability*, 185(99):457–481.

Cordeiro, L., Kesseli, P., Kroening, D., Schrammel, P., and Trtik, M. (2018). Jbmc: A bounded model checking tool for verifying java bytecode. In *International Conference on Computer Aided Verification*, pages 183–190. Springer.

Kahsai, T., Rümmer, P., and Schäf, M. (2019). Jayhorn: A java model checker: (competition contribution). In *Tools and Algorithms for the Construction and Analysis of Systems: 25 Years of TACAS: TOOLympics, Held as Part of ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part III 25*, pages 214–218. Springer.

Menezes, R., Moura, D., Cavalcante, H., de Freitas, R., and Cordeiro, L. C. (2022). Esbmc-jimple: verifying kotlin programs via jimple intermediate representation. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 777–780.

Monteiro, F. R., Gadelha, M. R., and Cordeiro, L. C. (2022). Model checking c++ programs. *Software Testing, Verification and Reliability*, 32(1):e1793.

Moraes, A. d. and Hayashi, V. (2021). *Segurança em IoT: Entendendo os Riscos e Ameaças em IoT. Alta Books*. Editora Alta Books.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, pages 1–10.

Rowland, C., Goodman, E., Charlier, M., Light, A., and Lui, A. (2015). *Designing connected products: UX for the consumer Internet of Things*. "O'Reilly Media, Inc.".

SILVA, V. J., CORDEIRO, L. C., and JÚNIOR, V. F. D. L. (2013). Verificação de aplicações ami usando java pathfinder.