BTS-Validator: identificando Aplicações Potencialmente Prejudiciais embarcadas no Android por meio de relatórios

Lucas Sousa[1], **Joao Nascimento**[1], **Roberto Souza**[1], **Joao Aragao**[1], **Thiago Menezes**[2], **Ewerton Andrade**[1,3]

[1]Sidia Instituto de Ciência e Tecnologia – SIDIA, Porto Velho / RO – Brasil

[2]Sidia Instituto de Ciência e Tecnologia – SIDIA, Manaus / AM – Brasil

[3]Universidade Federal de Rondônia – UNIR, Porto Velho / RO – Brasil

{lucas.sousa,joao.nascimento,roberto.souza,joao.aragao}@sidia.com

{thiago.menezes,ewerton.andrade}@sidia.com

Resumo. *A Google exige uma licença para que dispositivos móveis inteligentes equipados com Android e seus serviços móveis básicos sejam comercializados. Como esse Processo de Aprovação pode levar semanas e custa um valor significativo, é comum que sejam implementados e executados testes preliminares. Neste sentido, este trabalho apresenta o BTS-Validator, uma ferramenta que implementa uma alternativa para o teste de identificação de Aplicações Potencialmente Prejudiciais por meio da extração de dados de Relatórios emitidos pela Google. Sendo que, com esta nova abordagem, foi possível reduzir o tempo de execução de dezenas de horas para apenas alguns minutos.*

Abstract. *Google imposes a license to vendors market Android-powered mobile devices with Google Mobile Services. Since this process might take weeks and cost a large amount of money, it is typical to implement and execute preliminary tests. In this context, this study introduces the BTS-Validator, a tool that uses data extracted from reports provided by Google to construct an alternative to the Potentially Harmful Applications detection test. With this new approach, the execution time was reduced from several hours to a few minutes.*

1. Introdução

O Android é um sistema operacional disponibilizado e mantido pela Google sob licença de código aberto. Além de disponibilizar e manter o Android, a Google possui uma série de aplicativos e serviços proprietários que são executados em dispositivos equipados com seu sistema operacional. Essa variedade de aplicativos e serviços é conhecida como *Google Mobile Services* (GMS) [Android 2023c]. Sendo que no GMS estão incluídos serviços como: YouTube, Gmail, Google Play Store, entre outros. Como a maioria dos usuários Android espera que esses aplicativos básicos do GMS venham pré-instalados no dispositivo e funcionem sem falhas [Arthur and Gibbs 2014], torna-se imperativo considerá-los no desenvolvimento de um novo dispositivo baseado no Android [Doshi 2023].

Em vista disso, embora não cobre pelo sistema operacional Android, a Google exige uma licença dos fabricantes de dispositivos para que eles possam comercializar aparelhos com o GMS incorporado. Complementarmente, a Google impede que aplicativos e serviços do GMS sejam posteriormente instalados pelos usuários em dispositivos

Android não licenciados. Sendo que o valor cobrado por cada uma dessas licenças pode variar de US$ 40.000,00 a US$ 75.000,00 [Arthur and Gibbs 2014].

Para garantir que os aplicativos e serviços do GMS funcionem corretamente, a Google realiza uma série de testes conhecida como *Google Approval Process* (Processo de Aprovação da Google). Mais especificamente, são realizados testes de compatibilidade, drivers, hardware, segurança, e diversas outras validações [Stone 2019]. Tudo para garantir que os softwares embarcados pelos fabricantes sejam confiáveis e compatíveis com a versão de código aberto do Android, bem como estejam rodando sem falhas e travamentos. Todavia, por existir uma fila de execução e ser um conjunto com testes complexos e numerosos, o *Google Approval* pode levar semanas [Doshi 2023].

Desta forma, para evitar o desperdício de tempo e recursos financeiros, os fabricantes de *smartphones* Android realizam testes preliminares para diminuir as chances de falhas nesse processo. Alguns desses testes preliminares são disponibilizados pela Google [Android 2023b, Android 2023a]. Porém, outros são implementados por terceiros ou não possuem implementações disponíveis na literatura. Contudo, eles podem contribuir significativamente para o mapeamento de falhas ainda no processo de desenvolvimento dentro da indústria. Diminuindo, assim, o tempo de desenvolvimento do software que será embarcado em um *smartphone* Android, além de economizar recursos financeiros no pagamento de execuções do *Google Approval* que resultem em falhas.

O *Build Test Suite* – BTS (Conjunto de Teste de Compilação) [Stone 2019] é um exemplo de teste do Processo de Aprovação que não possui implementação oficial e/ou descrição detalhada disponibilizada pela Google. Esse teste de segurança tem por objetivo analisar todos os binários e bibliotecas pré-carregados na distribuição do Android que está sendo testada, para verificar a existência de Aplicações Potencialmente Prejudiciais – PHA (*Potentially Harmful Application*) [Stone 2019, Gamba 2022].

Embora não exista implementação oficial nem uma especificação detalhada sobre o que é realizado no BTS, é possível encontrar implementações e trabalhos acadêmicos que discutam uma versão alternativa para execução de uma verificação preliminar do BTS [Sutter and Tellenbach 2023, Zheng et al. 2014]. Usualmente, essas alternativas se baseiam em análises estáticas e dinâmicas para inspecionar o comportamento das aplicações para determinar se ocorrerá falha durante o processo oficial do BTS [Sutter and Tellenbach 2023, Zheng et al. 2014]. Todavia, não existe nenhuma garantia de aprovação no BTS caso essas verificações alternativas não encontrem falhas.

Apesar de funcionais, é importante destacar que as execuções dessas versões alternativas podem durar muitas horas (ou até dias), devido à grande quantidade de informações que análises estáticas e dinâmicas precisam processar [Sutter and Tellenbach 2023, Zheng et al. 2014, Gamba 2022]. Assim, em contraponto a esta abordagem e buscando diminuir o tempo necessário para execução desse teste preliminar alternativo, este trabalho propõe o "BTS-Validator", uma ferramenta que implementa um teste preliminar de segurança para o BTS, utilizando uma estratégia diferente: a busca binária em uma base de conhecimento ordenada.

2. Materiais e métodos

Foram utilizados os seguintes softwares e ferramentas: a linguagem de programação Python; o MongoDB para persistência dos dados e disponibilização de uma base de

dados ordenados; os frameworks web Flask e React para desenvolvimento do sistema; e a plataforma Docker para compartimentalização dos serviços e componentes. Em relação ao software da solução, destaca-se que foi utilizada uma arquitetura baseada em microsserviços e uma rotina de desenvolvimento baseada em métodos ágeis. O método científico utilizado neste trabalho foi o da pesquisa aplicada, baseada em hipótese-dedução, com a utilização de referenciais tecnológicos e científicos para definição do problema, especificação da hipótese de solução e sua avaliação [Wazlawick 2009]. Dentre as métricas utilizadas, destaca-se que foram consideradas a acurácia e usabilidade da solução proposta. Assim como foi comparado o seu desempenho, frente as demais soluções.

3. Fluxo para lançamento de um dispositivo aprovado pela Google

A Figura 1 ilustra o fluxo generalizado para licenciamento de dispositivos Android que venham equipados com o GMS, onde depreende-se que [Doshi 2023]:

Início: o ciclo de desenvolvimento do produto começa com a idealização do dispositivo, onde o fabricante projeta e especifica suas características e detalhes técnicos.

Modifica Android: após todas características serem definidas, elas são comparadas com as recomendações da Google [Android 2023a]. Caso todas as especificações sejam compatíveis, os desenvolvedores começam a modificar a versão base do Android para atender as necessidades estabelecidas pelo fabricante e implementar as funcionalidades desejadas. É nessa etapa que são incorporadas as aplicações e bibliotecas que não fazem parte da versão de código aberto do Android (software de terceiros). Para controlar o desenvolvimento dessas modificações e a incorporação das aplicações e bibliotecas de terceiros, é comum que as empresas utilizem sistemas proprietários.

Realiza testes preliminares: conforme discutido anteriormente, solicitar o Processo de Aprovação da Google pode custar muito tempo e dinheiro. Por isso, os fabricantes executam testes preliminares. Assim, muitos recursos são economizados e falhas são diagnosticadas precocemente. Na prática, são executados todos os testes que a Google disponibiliza publicamente e todos outros testes complementares que o fabricante julgar necessário para avaliar o software que será embarcado. Embora seja fortemente recomendado que o time de desenvolvimento siga as recomendações dos testes preliminares, algumas empresas permitem que seus desenvolvedores ignorem essas recomendações e mesmo com falhas/alertas submetam seu Android modificado ao processo de aprovação da Google.

Solicita aprovação da Google: ao superar todas as etapas internas e o novo dispositivo estar pronto e com o Android modificado portado para seu hardware, finalmente a fabricante pode solicitar formalmente que o Processo de Aprovação da Google seja realizado.

Relatório: caso todos os testes executem sem encontrar falhas, a Google gera um relatório com todos os detalhes do hardware e do software embarcado. Caso contrário, a Google gera um relatório especificando os problemas encontrados e envia de volta para a fabricante realizar as correções nos componentes que apresentaram alertas ou falhas. Mais especificamente, em ambos os relatórios é possível encontrar informações como: código identificador da submissão; apelido da submissão; detalhes do hardware; países onde o dispositivo será comercializado; nome do desenvolvedor responsável pela modificação do Android; entre outras informações que caracterizem o dispositivo e o software encaminhados para licenciamento. Já no relatório de reprovação é possível encontrar informações complementares, como: nome do teste que apresentou falha/alerta; tipo de falha/alerta; nome e versão do componente que apresentou falha/alerta; descrição da falha/alerta; código identificador da falha/alerta (assinatura); prazo para correção da

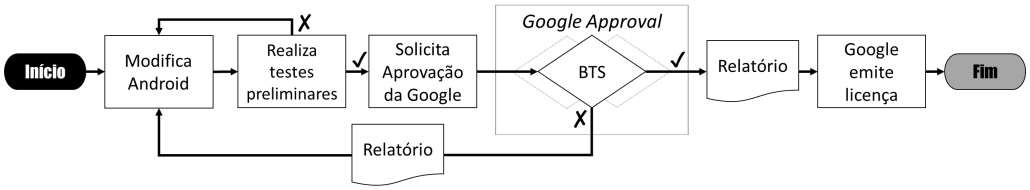

Figura 1. Fluxo para licenciamento de um dispositivo Android.

falha/alerta (até quando será aceita); comentários e outras informações adicionais. Tudo para facilitar que a indústria refatore o código da maneira mais assertiva e adequada.
Google emite a licença: assim que o dispositivo passa pelo Processo de Aprovação, sua assinatura será registrada na Google e poderá ser vista posteriormente no Play Store.
Fim: ao final, o produto está licenciado e pode ser comercializado com o GMS.

4. Descrição da solução

Utilizando como base as informações descritas na Seção 3, é possível notar que o fluxo para lançamento de um dispositivo aprovado pela Google gera diversos dados que podem ser utilizados para aprimorar a modificação do Android. Sobretudo os relatórios emitidos pela Google, pois além de conterem informações que identificam o software e o hardware que está sendo testado, também trazem informações valiosas sobre falhas e alertas.

Desta forma, recorrendo a essas informações contidas nos relatórios e utilizando os softwares e métodos descritos na Seção 2, foi desenvolvido um teste de segurança que (I) utiliza técnicas de extração de dados para coletar e padronizar as informações referentes as Aplicações Potencialmente Prejudiciais (PHA) e demais falhas e alertas do BTS; (II) integra essas informações aos demais sistemas já utilizados durante a modificação do Android; (III) implementa um novo teste preliminar que pode ser incorporado ao fluxo para lançamento de um dispositivo aprovado pela Google.

Mais especificamente, utilizou-se expressões regulares e técnicas de raspagem de dados para extrair as informações de interesse que são disponibilizadas nesses relatórios. Sendo que neste primeiro protótipo foram extraídas informações como: nome do teste que apresentou falha/alerta; tipo de falha/alerta; nome e versão do componente que apresentou falha/alerta; descrição da falha/alerta; código identificador da falha/alerta (assinatura); e prazo para correção da falha/alerta (até quando será aceita). Vale destacar que após extraídas essas informações são padronizadas e alimentam um banco de dados que armazena suas chaves primárias de busca de forma ordenada, além de criar índices para facilitar a busca por meio de cadeia de caracteres.

Em seguida, é possível consultar os sistemas internos da empresa para complementar as informações dos relatórios e, assim, criar uma base mais robusta. Essas consultas podem fornecer informações como: atualizações do componente que apresentou falha/alerta; componentes alternativos que desempenham a mesma função; modificações que corrijam a falha/alerta; entre diversas outras informações.

Não obstante, é importante salientar que esse processo de coleta e complementação dos dados por meio de outras consultas deve ser realizado de forma periódica. Buscando um intervalo de tempo que não seja demasiadamente esparso (evitar que as informações fiquem defasadas), mas também não ocorra a todo momento (evitar

a sobrecarga do sistema). Sendo que nesta primeira versão da ferramenta ficou definido que estas atualizações devem ocorrer a cada 6 (seis) horas, pois neste intervalo um determinado fabricante receberá uma boa quantidade de informações em seus relatório e não irá sobrecarregar seu parque tecnológico com as rotinas de atualização do BTS-Validator. Todavia, esse intervalo pode ser ajustado de acordo com as necessidades específicas.

Além disso, vale destacar que foi desenvolvido um sistema web que funciona como interface da ferramenta. Nele, o desenvolvedor pode submeter a versão preliminar do software que será embarcado em determinado dispositivo. Assim, o BTS-Validator irá verificar se o software submetido a análise contém algum componente que já apresentou falha ou alerta em algum processo de aprovação da Google anteriormente realizado.

5. Resultados e discussões

Como toda a ferramenta foi desenvolvida de forma modular e baseada em microsserviços, foi possível adaptar o fluxo genérico descrito na Seção 3 para receber o teste preliminar de BTS proposto neste trabalho. Na prática, o novo fluxo foi adaptado para que, logo após a etapa de modificação do Android, ocorram as verificações do BTS-Validator antes de qualquer outro teste preliminar. A Figura 2 fornece uma visão geral deste novo fluxo.

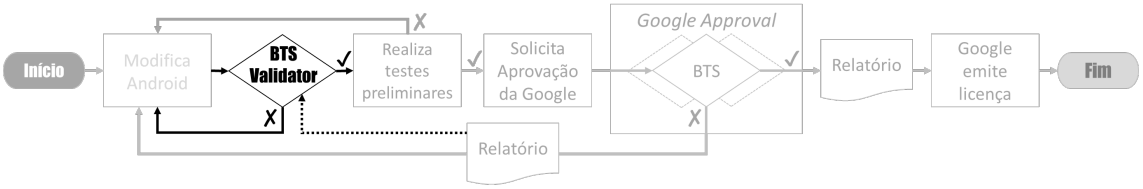

Figura 2. Inclusão do "BTS-Validator" no fluxo para licenciamento.

Isto porque, quando comparado com os demais testes preliminares necessários para evitar falhas no *Google Approval*, verificou-se que o BTS-Validator é rápido em termos de tempo de execução, além de ser efetivo no diagnóstico de erros. Ademais, quando comparado com outros testes preliminares de BTS presentes na literatura [Sutter and Tellenbach 2023, Zheng et al. 2014], o BTS-Validator mostrou um melhor tempo de execução. Isso porque essas outras soluções se baseiam em análises estáticas e dinâmicas para inspecionar o comportamento das aplicações. Sendo que análises estáticas apresentam tempos de execução de ordem linear, enquanto análises dinâmicas podem apresentar tempos de execução exponenciais. Já o BTS-Validator, por realizar buscas binárias em uma base ordenada, apresenta tempos de execução de ordem logarítmica.

Na prática, após 3 meses sendo utilizado em nossa instituição, o tempo de execução do teste preliminar do BTS reduziu de uma média de 72 horas para aproximadamente 5 minutos. Valendo frisar que agora é utilizado o BTS-Validator, mas anteriormente era utilizada uma solução proprietária que realizava análise estática e dinâmica no software que será submetido ao processo de aprovação da Google.

6. Considerações finais

Neste trabalho foi apresentado o BTS-Validator, uma ferramenta que fornece suporte ao desenvolvimento seguro de distribuições Android modificadas. De forma geral, observou-se que a estratégia de se utilizar dados históricos ao invés de análises estáticas e dinâmicas

resultou em um teste preliminar do BTS com um tempo de execução significativamente menor. Além disso, o BTS-Validator demonstrou ser versátil ao incorporar conhecimento sobre novas Aplicações Potencialmente Prejudiciais (PHA) e demais falhas e alertas do BTS sem a necessidade de grandes adaptações ou processos de treinamento.

Um outro resultado colateral dessa abordagem é que empresas de grande porte (e que realizam muitos Processos de Aprovação) possuirão uma base de dados maior. Assim, terão acesso a mais informações validadas pela Google sobre PHAs, falhas e alertas do BTS. Logo, poderão implementar uma instância do BTS-Validator ainda mais efetiva.

Ademais, acreditamos que seja possível desenvolver novas ferramentas aplicando esta estratégia nos demais testes necessários para o licenciamento do novo dispositivo (*e.g.*, CTS, CTS-V, VTS, CAT, STS, entre outros); uma vez que os Relatórios do Processo de Aprovação da Google também apresentam informações desses testes. Além disso, prospectamos realizar novos estudos para (I) comparar a taxa de sucesso no Processo de Aprovação da Google antes e após a utilização do BTS-Validator; (II) expandir os experimentos de eficiência dos principais testes preliminares de BTS; (III) testar novas escolhas de parâmetros e otimizações de código para avaliar os seus impactos.

7. Agradecimentos

Este artigo é o resultado do projeto de PD&I Mobile Cybersecurity, realizado pelo Sidia Instituto de Ciência e Tecnologia em parceria com a Samsung Eletrônica da Amazônia Ltda., usando recursos da Lei Federal nº 8.387/1991, estando sua divulgação e publicidade em conformidade com o previsto no artigo 39º do Decreto nº 10.521/2020.

Referências

Android (2023a). Android Compatibility Definition Document. `https://source.android.com/docs/compatibility/cdd`. Acesso em: 2023-08-01.

Android (2023b). Compatibility Test Suite. `https://source.android.com/docs/compatibility/cts`. Acesso em: 2023-08-01.

Android (2023c). Google Mobile Services. `https://www.android.com/gms/`.

Arthur, C. and Gibbs, S. (2014). TheGuardian: The hidden costs of building an Android device. `https://www.theguardian.com/technology/2014/jan/23/how-google-controls-androids-open-source`. Acesso em: 2023-08-01.

Doshi, S. (2023). How to Obtain Google's GMS Certification for Latest Android Devices? `https://www.einfochips.com/blog/how-to-obtain-googles-gms-license-for-android-devices/`. Acesso em: 2023-08-01.

Gamba, J. (2022). *On Privacy in the Android Supply Chain*. PhD thesis, Carlos III, Spain.

Stone, M. (2019). Securing the System: A Deep Dive into Reversing Android Pre-Installed Apps. `https://www.blackhat.com/us-19/briefings/schedule/`.

Sutter, T. and Tellenbach, B. (2023). FirmwareDroid: Towards Automated Static Analysis of Pre-Installed Android Apps. In *2023 IEEE/ACM 10th MOBILESoft*, pages 12–22.

Wazlawick, R. (2009). *Metodologia de pesquisa para ciência da computação*. Elsevier.

Zheng, M., Sun, M., and Lui, J. C. (2014). DroidRay: A Security Evaluation System for Customized Android Firmwares. In *ASIA CCS '14*, page 471–482, New York, USA.