2017 IEEE/ACM 39th IEEE International Conference on Software Engineering Companion

DroidBot: uma entrada de teste leve e guiada pela interface do usuário Gerador para Android

Yuanchun Li, Ziyue Yang, Yao Guo, Xiangqun Chen Laboratório Chave
de Tecnologias de Software de Alta Confiança (Ministério da Educação)
Escola de Engenharia Eletrônica e Ciência da Computação, Universidade de Pequim, Pequim, China E-mail: {yunchun.li,
yzydzyx, yaoguo, cherry}@pku.edu.cn

Resumo—Como muitas ferramentas automatizadas de geração de entrada de teste para Android precisam instrumentar o sistema ou o aplicativo, elas não podem ser usadas em alguns cenários, como testes de compatibilidade e análise de malware. Apresentamos o DroidBot, um gerador de entrada de teste guiado por interface do usuário leve, que é capaz de interagir com um aplicativo Android em praticamente qualquer dispositivo sem instrumentação. A principal técnica por trás do DroidBot é que ele pode gerar entradas de teste guiadas pela interface do usuário com base em um modelo de transição de estado gerado em tempo real e permitir que os usuários integrem suas próprias estratégias ou algoritmos. O DroidBot é leve, pois não requer instrumentação de aplicativos, portanto, não há necessidade de se preocupar com a inconsistência entre a versão testada e a versão original. É compatível com a maioria dos aplicativos Android e pode ser executado em quase todos os sistemas baseados em Android, incluindo sandboxes personalizadas e dispositivos de commodities. O Droidbot é lançado como uma ferramenta de código aberto no GitHub [1], e o vídeo de demonstração pode ser encontrado em https://youtu.be/3-aHG SazMY.

Palavras-chave-Android; análise dinâmica; testes automatizados; detecção de malware; testes de compatibilidade;

I. INTRODUÇÃO

Nos últimos anos, os aplicativos móveis (aplicativos em resumo) tiveram ampla adoção, com mais de dois milhões de aplicativos disponíveis para download no Google Play e na Apple App Store, enquanto bilhões de downloads foram acumulados.

Como existem muitos aplicativos e muitos dispositivos diferentes, automatizar o teste de aplicativos tornou-se uma importante direção de pesquisa. Em particular, uma grande quantidade de pesquisas tem sido focada em técnicas automatizadas de geração de entrada para aplicativos Android.
De acordo com uma pesquisa recente [2], a maioria das abordagens faz uso de instrumentação de aplicativos ou modificação do sistema para obter informações suficientes para orientar os testes.

No entanto, não é realista instrumentar um aplicativo ou o sistema em alguns cenários. Por exemplo, em testes de compatibilidade, um aplicativo deve ser testado "como está" em dispositivos comuns para descobrir qual dispositivo pode causar uma falha. Outro exemplo é a análise de malware. Como muitos aplicativos maliciosos são ofuscados, pode ser difícil, até mesmo não impossível, instrumentá-los.
Alguns aplicativos maliciosos também aplicam a detecção de sandbox, o que pode levar a comportamentos diferentes em dispositivos de teste instrumentados e dispositivos reais.

Este documento de demonstração apresenta o DroidBot, um gerador de entrada de teste guiado por interface do usuário leve para aplicativos Android. O princípio de design do DroidBot é oferecer suporte à geração de entrada de teste baseada em modelo com requisitos extras mínimos.

O DroidBot oferece geração de entrada guiada por interface do usuário com base em um modelo de transição de estado, que é gerado dinamicamente em tempo de execução. Isto em seguida, gera entradas de teste guiadas pela interface do usuário com base no modelo de transição. Por padrão, a entrada é gerada com uma estratégia de profundidade, que é eficaz na maioria dos casos. Os usuários também podem personalizar a estratégia de exploração escrevendo scripts ou integrar seus próprios algoritmos estendendo os módulos de geração de eventos, tornando o DroidBot uma ferramenta altamente extensível.

A principal razão pela qual o DroidBot é mais leve é que ele não requer conhecimento prévio de código inexplorado.
Ao contrário de muitos geradores existentes que dependem de análise estática e instrumentação para obter conhecimento de código inexplorado, o DroidBot apenas modela os estados explorados com base em um conjunto de utilitários de teste/depuração integrados do Android. Embora isso possa tornar o DroidBot mais difícil de acionar alguns estados específicos, a compensação permite que o DroidBot funcione com qualquer aplicativo (incluindo os aplicativos ofuscados/criptografados que não podem ser instrumentados) em quase qualquer dispositivo personalizado (a menos que o dispositivo remova intencionalmente o built-in módulos de teste/depuração do framework Android original, o que raramente ocorre.).

O DroidBot também oferece uma nova maneira de avaliar a eficácia das entradas de teste. As abordagens existentes usam principalmente o EMMA [3] em aplicativos de código aberto ou aplicativos de instrumentos para calcular a cobertura do teste. No entanto, para aplicativos anti-instrumentação (por exemplo, verificar a assinatura em tempo de execução ou criptografar o código), é difícil ou mesmo impossível obter sua cobertura de teste. O DroidBot é capaz de gerar o rastreamento da pilha de chamadas para cada entrada de teste, que contém os métodos do aplicativo e os métodos do sistema acionados pela entrada de teste. Podemos usar a pilha de chamadas como uma métrica aproximada para quantificar a eficácia das entradas de teste.

O código fonte do DroidBot está disponível no GitHub [1].

II. PROJETO DA FERRAMENTA

A arquitetura geral do DroidBot é mostrada na Figura 1.
Para testar um aplicativo em um dispositivo, o DroidBot requer que o dispositivo esteja conectado via ADB. O dispositivo pode ser um emulador, um dispositivo de commodities ou um sandbox personalizado, como TaintDroid [4] e DroidBox [5].

Apresentamos o módulo Adapter para fornecer uma abstração do dispositivo e do aplicativo em teste (AUT). Ele lida com questões técnicas de baixo nível, como compatibilidade com diferentes versões do Android e diferentes tamanhos de tela, manutenção da conexão com o dispositivo, envio de comandos para o dispositivo e processamento de saídas de comandos, etc.

O Adaptador também atua como uma ponte entre o ambiente de teste e o algoritmo de teste. Por um lado, monitora o estado

978-1-5386-1589-8/17 $ 31,00 © 2017 IEEE
DOI 10.1109/ICSE-C.2017.8

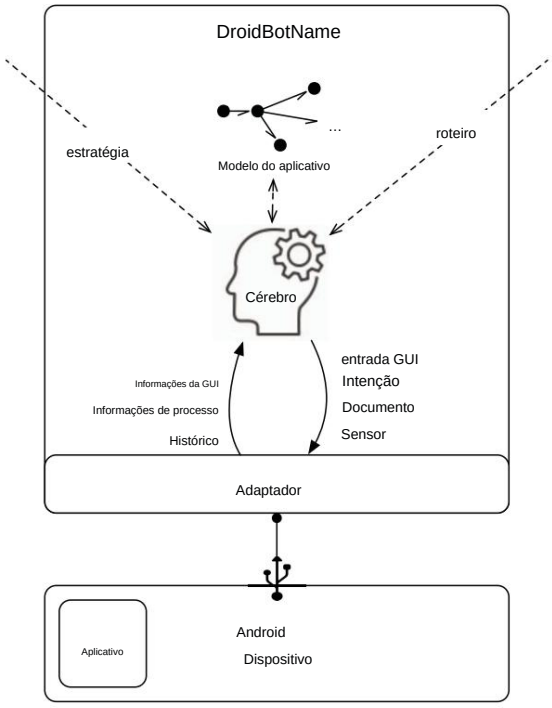


Fig. 1. Visão geral do DroidBot.

do dispositivo e AUT e converte as informações de estado para dados estruturados. Por outro lado, recebe as entradas de teste gerado pelo algoritmo e os traduz em comandos.
Com o Adapter, o DroidBot é capaz de fornecer um conjunto de APIs de alto nível fáceis de usar para que os usuários escrevam algoritmos, garantindo que os algoritmos funcionem em diferentes ambientes de teste.

O módulo Brain recebe informações do dispositivo e do aplicativo produzido pelo Adapter em tempo de execução, e envia testar as entradas para o Adaptador. A geração de entrada de teste é baseada em um gráfico de transição de estado construído em tempo real. Cada nó do gráfico representa um estado do dispositivo, enquanto a borda entre cada par de nós representa a entrada de teste que acionou o estado transição. O DroidBot integra um algoritmo de exploração em profundidade simples, mas eficaz, para gerar entradas de teste. Também permite usuários para integrar seus próprios algoritmos ou usar aplicativos específicos scripts para melhorar a estratégia de teste.

Esse design melhora a usabilidade do DroidBot. Tabela I mostra as comparações de usabilidade entre o DroidBot e outros ferramentas de geração de entrada de teste disponíveis ao público. Nós podemos ver isso O DroidBot requer tão poucos requisitos quanto o Monkey, enquanto fornecendo recursos muito mais extensíveis comparáveis a outros ferramentas que requerem instrumentação.

III. IMPLEMENTAÇÃO

A. Monitor leve e entrada

DroidBot busca informações do dispositivo/aplicativo do dispositivo e envia entradas de teste para o dispositivo através do ADB. Tanto o fases de monitoramento e entrada são leves porque são

TABELA I
COMPARAÇÃO DE USABILIDADE DE BLACK-BOX DISPONÍVEL PÚBLICAMENTE EXISTENTE GERADORES DE ENTRADA DE TESTE. OBSERVE QUE ALGUNS DADOS SÃO DE CHOUDHARY et al. [2].

Ferramenta	Instrumentação		Estratégia programável	
	Aplicativo do sistema			
Macaco [6]	ÿ ÿ Aleatório ÿ ÿ Modelo ÿ ÿ			ÿ
AndroidRipper [7]	Aleatório ÿ ÿ Modelo ÿ ÿ		Modelo	ÿ
DynoDroid [8]	Modelo			ÿ
SwiftHand [9]				ÿ
PUMA [10]				ÿ
DroidMate [11]				ÿ
DroidBot [1]				ÿ

principalmente baseado em utilitários de depuração/teste Android existentes, que estão disponíveis na maioria dos dispositivos Android.

As informações obtidas do dispositivo podem ser categorizadas em três conjuntos:

1) **Informações da GUI.** Para cada UI, o DroidBot registra o captura de tela e a árvore de hierarquia da interface do usuário despejada usando UI Automator (para versão do SDK superior a 16) ou Visualizador de hierarquia (para versões inferiores);
2) **Informações do processo.** DroidBot monitora o nível do sistema status do processo usando o comando ps e o nível do aplicativo status do processo usando a ferramenta dumpsys no Android.
3) **Registros.** Os logs incluem o rastreamento de método acionado por cada entrada de teste e os logs produzidos pelo aplicativo. Eles podem ser recuperado da ferramenta de criação de perfil do Android e do logcat.

Os tipos de entrada de teste suportados pelo DroidBot incluem Entradas de interface do usuário (como tocar, rolar etc.), intents (BOOT COMPLETED broadcast, etc.), documentos para upload (imagem, txt, etc.) e dados do sensor (sinal GPS etc.). Observe que a simulação do sensor é suportada apenas pela emulação.

DroidBot fornece uma lista de APIs fáceis de usar para buscar informações do dispositivo e enviar entradas para o dispositivo. Por exemplo, os desenvolvedores podem simplesmente chamar device.dump_views() para obter uma lista de visualizações de interface do usuário e chamar view.touch() para enviar uma entrada de toque para uma visualização.

B. Construção do modelo em tempo real

O DroidBot gera um modelo de AUT baseado nas informações monitoradas em tempo de execução. O modelo visa auxiliar na entrada algoritmos de geração para fazer melhores escolhas de entrada de teste.

A Figura 2 mostra um exemplo de modelo de transição de estado. Basicamente, o modelo é um grafo direcionado, no qual cada nó representa um estado do dispositivo, e cada aresta entre dois nós representa o evento de entrada de teste que acionou a transição de estado. Um nó de estado normalmente contém as informações da GUI e o informações do processo em execução e uma borda de evento contém as detalhes da entrada de teste e os métodos/logs acionados pelo entrada.

O grafo de transição de estado é construído em tempo real. DroidBotName mantém as informações do estado atual e monitora o estado muda após o envio de uma entrada de teste para o dispositivo. Uma vez o estado do dispositivo é alterado, ele adiciona a entrada de teste e o novo estado para o grafo, como uma nova aresta e um novo nó.

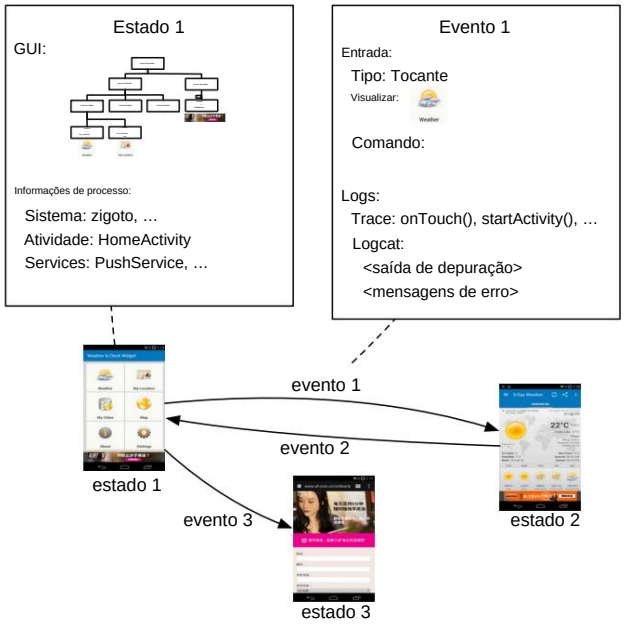


Fig. 2. Um exemplo de gráfico de transição de estado. Observe que os dados neste gráfico são simplificados para facilitar o entendimento.

O processo de construção do gráfico depende do algoritmo de comparação de estado subjacente. Atualmente, o DroidBot usa comparação baseada em conteúdo, onde dois estados com diferentes conteúdos de interface do usuário são considerados nós diferentes.

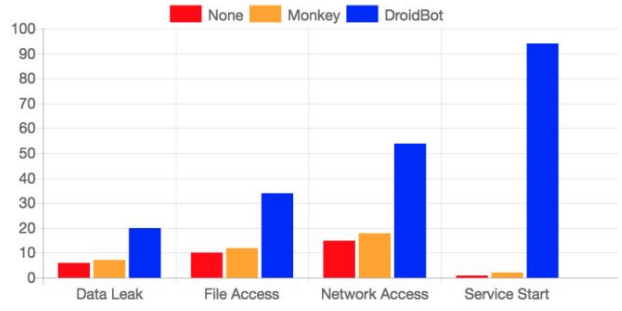
C. Quantificando a eficácia da entrada de teste Um problema

enfrentado por pesquisadores e testadores ao realizar testes de caixa preta é a dificuldade de avaliar a eficácia do teste, pois os métodos de cobertura de teste existentes exigem o código-fonte do AUT [3] ou precisam instrumentar o AUT [12].

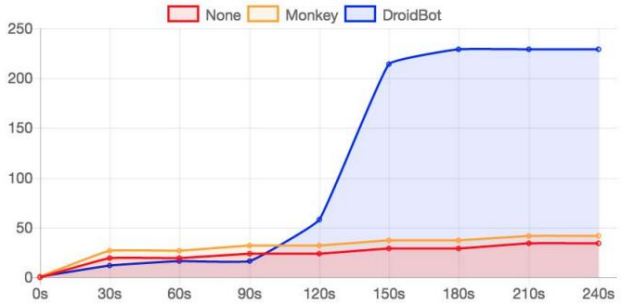
DroidBot integra dois métodos para quantificar o efeito do teste
tividade sem código fonte ou instrumentação:

- Rastreamento de métodos. O DroidBot é capaz de imprimir o rastreamento de método de cada entrada de teste usando a ferramenta de perfil oficial do Android. O rastreamento de método contém os métodos do aplicativo e os métodos do sistema acionados pela entrada de teste. Com o rastreamento do método, também podemos calcular a cobertura do método se o número total de métodos estiver disponível.
- Monitoramento de comportamento sensível. Para análise de malware, o número de comportamentos sensíveis acionados pode refletir a eficácia do teste. Por exemplo, DroidBot pode ser usado com DroidBox [5] para monitorar os comportamentos sensíveis acionados por cada entrada.

O mecanismo de rastreamento de método é melhor dimensionado, pois funciona com quase qualquer dispositivo e qualquer aplicativo, enquanto o mecanismo de monitoramento de comportamento sensível requer aplicativos executados em uma determinada sandbox. No entanto, o número de comportamentos sensíveis pode ser mais intuitivo na análise de malware. Ambos os métodos não podem fornecer um valor normalizado da eficácia exata de um caso de teste, mas podem fornecer estatísticas significativas ao comparar diferentes casos de teste no mesmo aplicativo.



(a) Número total de comportamentos sensíveis em quatro categorias.



(b) Velocidade de desencadear comportamentos sensíveis.

Fig. 3. Comparação da eficácia em acionar comportamentos sensíveis ao testar um malware com Monkey e DroidBot.

4. CENÁRIOS DE USO

A. Análise de Compatibilidade

Um dos cenários úteis do DroidBot é o teste de compatibilidade, que visa avaliar a exatidão e a robustez do aplicativo ao ser executado em diferentes dispositivos. O teste de compatibilidade deve ser realizado em muitos dispositivos comuns diferentes, portanto, a instrumentação do sistema não é realista. Enquanto isso, a instrumentação do aplicativo também pode ser indesejada porque o aplicativo instrumentado pode se comportar de maneira diferente do aplicativo original.

Com o DroidBot, um desenvolvedor pode testar seu aplicativo em diferentes dispositivos sem instrumentação, alcançando mais estados de interface do usuário em um tempo muito menor em comparação com o Monkey. Além disso, com o recurso de script fornecido pelo DroidBot, o desenvolvedor pode personalizar a entrada de teste a ser gerada.

B. Análise de malware

A análise de malware também é um cenário útil do DroidBot. Como muitos malwares criptografam seu código ou verificam sua assinatura antes de realizar ações maliciosas, pode ser impossível instrumentá-los ou garantir a consistência entre o aplicativo instrumentado e o aplicativo original.

O Monkey [6] é capaz de testar malware sem instrumentação, mas a estratégia aleatória do Monkey pode não ser eficiente em descobrir os comportamentos maliciosos. O DroidBot é tão fácil de usar quanto o Monkey, mas é melhor na exploração de aplicativos, pois usa uma estratégia baseada em modelo. Por exemplo, se um malware não apresentar comportamento malicioso até que o usuário clique em determinados botões, pode ser difícil para o gerador de entrada de teste aleatório encontrar os botões corretos, enquanto o gerador baseado em modelo tem a

informações sobre o AUT obtidas do dispositivo em tempo de execução, portanto, é mais fácil acionar os comportamentos confidenciais.

A Figura 3 mostra a comparação com o Monkey em um exemplo de prova de conceito do uso do DroidBot na análise de malware. Selecionamos um malware que criptografou seu código como o aplicativo em teste e usamos o DroidBox [5] como dispositivo de teste para monitorar os comportamentos confidenciais, como acessos a arquivos, acessos à rede, vazamentos de dados etc.

Usamos Monkey e DroidBot para gerar entradas de teste respectivamente. O resultado mostra que a quantidade de comportamentos sensíveis acionados pelo DroidBot é muito maior do que o Monkey, enquanto as entradas geradas pelo Monkey quase não acionaram nenhum comportamento extra sensível. Inspecionamos os processos de teste do Monkey e do DroidBot. O motivo da ineficácia do Monkey é que o aplicativo exige que os usuários toquem em dois botões em uma caixa de diálogo pop-up sucessivamente para entrar em um estado malicioso. O DroidBot encontrou os botões com sucesso e os tocou em cerca de 80 segundos, enquanto as entradas de teste aleatórias geradas pelo Monkey não conseguiram passar na caixa de diálogo pop-up.

V. TRABALHO RELACIONADO

A geração de entrada de teste para Android vem atraindo o interesse dos pesquisadores há muito tempo.

Monkey [6] é a ferramenta mais popular para realizar testes de caixa preta, e é a mais leve. No entanto, as entradas geradas pelo Monkey são completamente aleatórias, o que não é extensível e fácil de ser intencionalmente contornado. DynoDroid [8] também gera entrada aleatória, mas é mais inteligente na seleção de entradas de teste.

AndroidRipper [7], SwiftHand [9], A3E [13] e GUICC [14] são geradores de teste automatizados baseados em modelo, enquanto usam métodos diferentes para construir o modelo e gerar entrada com base no modelo. PUMA [10] é um framework de teste baseado em modelo, que é programável com PUMAScript.
SmartDroid [15] e Brahmastra [16] estão focados em testes direcionados que visam acionar certos pedaços de código.

Andlantis [17] é projetado para análise de malware. Ele é focado no gerenciamento de máquinas virtuais em larga escala e capaz de executar malware em vários emuladores ao mesmo tempo.

DroidMate [11] é uma abordagem semelhante ao DroidBot, pois também enfatiza a robustez e a estratégia extensível, mas ainda precisa de uma pequena instrumentação para permitir o monitoramento da API.

Comparado a essas ferramentas, o DroidBot é tão fácil de usar quanto a tecla Mon, enquanto fornece recursos muito avançados como a maioria das outras ferramentas, incluindo geração de entrada baseada em modelo e script extensível, etc.

VI. CONCLUSÃO

Esta demonstração apresenta o DroidBot, um gerador de entrada de teste leve para aplicativos Android. O DroidBot é capaz de testar um aplicativo Android em praticamente qualquer dispositivo com requisitos de ambiente menores. É fácil de usar porque, por um lado, é extensível com base em um conjunto de APIs de alto nível e um modelo de transição de estado construído em tempo real, por outro lado, fornece um conjunto de utilitários para avaliar a eficácia do teste.
Além das tarefas de teste regulares, o DroidBot também pode ser usado em cenários, incluindo testes de compatibilidade, análise de malware e outros casos em que a instrumentação é indesejada.

RECONHECIMENTO

Este trabalho é parcialmente apoiado pelo Programa Nacional de Pesquisa e Desenvolvimento Chave sob o Grant No.2016YFB1000105 e a Fundação Nacional de Ciências Naturais da China sob
Concessão nº 61421091.

REFERÊNCIAS

[1] honeynet, “Droidbot: A light test input generator for android,” https://github.com/honeynet/droidbot, 2016, acessado: 2016-11-10.
[2] SR Choudhary, A. Gorla e A. Orso, “Geração de entrada de teste automatizada para android: já chegamos? (e)”, em Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), ser. ASE '15. Washington, DC, EUA: IEEE Computer Society, 2015, pp. 429–440.
[3] V. Roubtsov, “Emma: uma ferramenta gratuita de cobertura de código java”, 2006.
[4] W. Enck, P. Gilbert, B.-G. Chun, LP Cox, J. Jung, P. McDaniel e AN Sheth, “Taintdroid: Um sistema de rastreamento de fluxo de informações para monitoramento de privacidade em tempo real em smartphones”, em Anais da 9ª Conferência USENIX sobre Projeto e Implementação de Sistemas Operacionais, ser. OSDI'10, 2010, pp. 393–407.
[5] A. Desnos e P. Lantz, “Droidbox: An android application sandbox for análise dinâmica”, 2011.
[6] A. Desenvolvedores, “macaco exercitador de interface do usuário/aplicativo”, 2012.
[7] D. Amalfitano, AR Fasolino, P. Tramontana, S. De Carmine e AM Memon, “Usando gui ripping para testes automatizados de aplicativos Android”, em Anais da 27ª Conferência Internacional IEEE/ACM sobre Engenharia de Software Automatizada, Ser. ASE 2012, 2012, pp. 258–261.
[8] A. Machiry, R. Tahiliani e M. Naik, “Dynodroid: Um sistema de geração de entrada para aplicativos Android”, em Anais da 9ª Reunião Conjunta de 2013 sobre Fundamentos de Engenharia de Software, ser. ESEC/FSE 2013, 2013, pp. 224–234.
[9] W. Choi, G. Necula e K. Sen, “Guided gui testing of android apps with minimal restart and approximating learning”, em Anais da Conferência Internacional ACM SIGPLAN de 2013 sobre Linguagens de Sistemas de Programação Orientada a Objetos & Aplicações, sr. OOPSLA '13, 2013, pp. 623-640.
[10] S. Hao, B. Liu, S. Nath, WG Halfond e R. Govindan, “Puma: Programmable ui-automation for large-scale dynamic analysis of mobile apps”, em Proceedings of the 12th Annual International Conference on Sistemas, Aplicativos e Serviços Móveis, ser. MobiSys '14, 2014, pp. 204–217.
[11] K. Jamrozik e A. Zeller, “Droidmate: Um gerador de teste robusto e extensível para android”, em Anais da Conferência Internacional sobre Engenharia e Sistemas de Software Móvel, ser. MOBILESoft '16, 2016, pp. 293–294. [12] ylimit, “androcov: medir a cobertura do teste sem código-fonte,” https://github.com/ylimit/androcov, 2016, acessado: 2016-11-10.
[13] T. Azim e I. Neamtiu, “Targeted and depth-first exploration for sistemáticos testes de aplicativos Android”, em Anais da Conferência Internacional ACM SIGPLAN 2013 sobre Linguagens de Sistemas de Programação Orientada a Objetos & Aplicações, sr. OOPSLA '13, 2013, pp. 641-660.
[14] Y.-M. Baek e D.-H. Bae, “Automated model-based android gui testing using multi-level gui comparação critérios”, em Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ser. ASE 2016, 2016, pp. 238–249.
[15] C. Zheng, S. Zhu, S. Dai, G. Gu, X. Gong, X. Han e W. Zou, “Droid inteligente: um sistema automático para revelar condições de gatilho baseadas em interface do usuário em aplicativos android, ” em Anais do Segundo Workshop da ACM sobre Segurança e Privacidade em Smartphones e Dispositivos Móveis, ser. SPSM '12, 2012, pp. 93-104.
[16] R. Bhoraskar, S. Han, J. Jeon, T. Azim, S. Chen, J. Jung, S. Nath, R. Wang e D. Wetherall, “Brahmastra: Driving apps to test the security of componentes de terceiros”, in Proceedings of the 23rd USENIX Conference on Security Symposium, ser. SEC'14, 2014, pp. 1021-1036.
[17] M. Bierma, E. Gustafson, J. Erickson, D. Fritz e YR Choe, “Andlantis: análise dinâmica android em grande escala”, pré-impressão arXiv arXiv:1410.7751, 2014.