به نام خدا

1. 2012, Automated Concolic Testing of Smartphone Apps

در این مقاله روشی برای تست برنامه های موبایل ارائه شده که تاکید بر تولید event ها با روش cocolic است. منظور از event اعمالی مثل لمس صفحه نمایش است. در این مقاله سعی شده که به صورت خودکار این eventها تولید شود. هدف هم تست GUI است. روش ارائه شده هم فقط تاکید بر tap events دارد و eventهای دیگر بررسی نشده است. همچنین در کارهای آینده بروی کارایی سیستم تاکید شده است. ابزار ارائه شده هم برای برنامه‌های اندرویدی است.

1. Dynodroid: An input generation system for Android apps, 2013

در این مقاله علاوه بر UI events به system events (مثل ورود SMS جدید) هم توجه شده است. از روش “observe-select-execute” ایsjthnش شده استبر برای برنامه‌های اندرویدی است.ده است.اکید بر تولید برای تولید eventها بدون تغییر دادن برنامه استفاده شده است. روش ارائه شده ترکیب روش خودکار و دستی است. در کاراهای مرتبط این مقاله بیان شده که در کل برای برنامه های موبایل 3 رویه وجود دارد: الف) Fuzz testing این رویه برای تست UI استفاده می‌شود که معروف ترین کار انجام شده ابزار MONKEY است. ب) Systematic testing: استفاده از روش symbolic execution برای اجرای مسیرهای مختلف در برنامه ج) Model-Based testing: برای تست برنامه‌های همراه با GUI است. معروف‌ترین فریم‌ورک هم GUITAR است. یا به وسیله کاربر یا به صورت خودکار مدل UI برنامه تولید می‌شود. Dynodroid از جمله ابزارهای خودکار است. در قسمت محدودیت‌ها آمده است که این ابزار نسبت به MONKEY کند تر است ولی پوشش بهتری دارد. این ابزار برنامه‌ها را ایزوله می‌کند و رابطه بین آنها را در نظر نمی‌گیرد و فقط ورژن 2.3.4 اندروید را پشتیبانی می‌کند.

1. Caiipa: Automated Large-scale Mobile App Testing through Contextual Fuzzing, 2014

در این مقاله تاکید بر stress test برنامه‌های ویندوزی است. (کارایی برنامه در شرایطی مثل کمبود حافظه و ...) این طور که فهمیدم برای تست این موارد از روش فازینگ برای تولید ورودی‌هایی که باعث کمبود حافظه و ... شود استفاده می‌شود ولی روش concolic استفاده نشده!

1. 2016, IntelliDroid A Targeted Input Generator for the Dynamic Analysis of Android Malware

در این مقاله ابزاری معرفی شده که بدافزارهای موجود در برنامه های اندرویدی کشف می شوند. در این مقاله بیان شده که روش concolic هم هنوز کارا نیست و برای کارایی بهتر روش targeted analysis ارائه شده که با استفاده از تحلیل ایستا و دادن اطلاعات به تحلیلگر پویا ( که باید این تحلیل پویا و رابطه آن با روش concolic بررسی شود) سعی می کند ورودی های کم تر ولی کاراتر تولید کند. در این مورد آمده است که اطلاعاتی در مورد یک سری از APIهای خاص که بدافزارها معمولا از آنها استفاده می کنند و همچنین بررسی کدهایی که از reflection و dynamic class loading استفاده می کنند، به تحلیلگر پویا داده می شود تا محدوده بررسی خود را هدفمند کند و ورودی های بهتری تولید کند. این ابزار با کمک این تحلیلگرها سعی در تولید event-chains و ورودی‌های مرتبط با UI دارد.

در ادامه در قسمت limitations ضعف‌های این ابزار آمده است:

مشکل اول در تولید call graph هست که ممکن است دقیق نباشد و فراخوانی بعضی از APIها را کشف نکند این مشکل رایج میان همه ابزارهاست(این می تواند یک open problem خوب باشد.)

ضعف دیگر این است که این ابزار در مورد متغییرهای درون heap و رابطه بین آنها را بررسی می کند و در مورد داده‌های درون فایل یا content providers و بررسی رابطه آنها و جریان این داده‌ها شکست می خورد. (open problem)

مشکل دیگر در رابطه با constraint solverهاست که ورودی هایی که تولید می کنند در بعضی اوقات از نظر عملی قابل اجرا نیستند(دقیقا این موضوع متوجه نشدم باید بررسی بیشتری شود)

ضعف دیگر این است که ابزار حاضر در مورد constant reflection targetها در تولید call graph کار می کند در حالی که بدافزارهای جدید سعی می کنند رفتار خود را با رمزنگاری، توابع درهم‌ساز یا reflection پنهان کنند که این ابزار توانایی پردازش این دسته را ندارد.

علاوه بر آن توسعه دهندگان ممکن است رفتار بدخواه خود را در packaged app یا native code پنهان کنند که این ابزار از آنها حمایت نمی کند (دقیقا متوجه نشدم)

(به طور خلاصه به نظر این مقاله به عنوان مقاله اصلی که باید روی آن تمرکز کرد مناسب است و open problemهای ذکر شده هم مناسب هستند.)