Institution: University of Edinburgh

Unit of Assessment: 11

Title of case study: The novel testing tool DexFuzz enables a smooth transition to Google's Android Runtime (ART) and its continuous bug-free development

Period when the underpinning research was undertaken: 2003 – 2018

Details of staff conducting the underpinning research from the submitting unit:

Name(s):	Role(s) (e.g. job title):	Period(s) employed by submitting HEI:
Björn Franke	Reader	2003 – present
Hugh Leather	Reader	2009 – present
Michael O'Boyle	Professor	1997 – present
Tom Spink	Senior Researcher	2016 – present

Period when the claimed impact occurred: 2014 – 2020

Is this case study continued from a case study submitted in 2014? No

1. Summary of the impact

A research collaboration between the University of Edinburgh and ARM Ltd has resulted in a tool that utilises novel binary fuzz testing methodologies to scan for bugs in Android's virtual machine and enable them to be eradicated before they can cause disruptions in mobile phone apps. Named DexFuzz, the tool's development led to Google changing their specifications for the Android Runtime (ART), the system that underpins the running of each of the 2,500,000,000 Android phones in use worldwide. It has also been adopted as part of the Android Open Source Project, the development package used by all manufacturers of Android phones, who collectively make up 85% of the global smartphone market.

2. Underpinning research

Compiler testing is critical to ensuring that developed code is free of errors, and will generate machine code with exactly the behaviour the user expects from their provided application source code. Since 2003 Dr Björn Franke and Dr Hugh Leather of the University of Edinburgh (UoE)'s Compiler and Architecture Design group (CArD) have conducted research into optimising compiler technology [3.1, 3.3], while developing strong testing methodologies alongside this work [e.g. in 3.4, 3.5, 3.6]. Their research has resulted in a programme of novel approaches to compiler testing, exploring a range of techniques.

In 2013, together with PhD students Harry Wagstaff and Tom Spink (staff since 2016), Franke developed a testing methodology based on instruction set architecture branch coverage analysis [3.1], capable of generating a set of test cases with a broader coverage than existing approaches. Shortly after this, Franke and Leather with student Stephen Kyle, developed a testing methodology for Google's Android virtual machine, based on a binary fuzzing technique combined with differential testing [3.2]. This work was carried out with Dave Butcher and Stuart Monteith from ARM Ltd., who sponsored Kyle's PhD studentship. The compiler fuzzing methodology was later extended using deep learning, in award-winning research led by Leather [3.3].

Firstly, the team used a probabilistic fuzz testing technique to discover coding errors [3.2]. Fuzz testing involves inputting massive amounts of random data, some of which may trigger faulty behaviour, thus uncovering a bug in the system. They observed that traditional fuzz testing was ineffective for compilers operating on binary encoded input languages, e.g. most compiler intermediate representations (IR). Instead, they extended the previously used naïve binary fuzz testing approach with domain awareness by providing hints of the encoding structure of binary encoded IR instructions. This significantly improved the effectiveness of their compiler testing methodology, as trivially rejected input sequences would be reduced by several orders of magnitude, leading to more diverse input sequences that can discover compiler bugs substantially faster. Within 24 hours the team found over 30x more programs that hang (become stuck) than a generic fuzz testing tool did during the same time period [3.2, Fig. 4].

At the same time as this work (2013-2014), Google had begun to transition their Android virtual machine from its existing Just-In-Time compiler, Dalvik, to a new and improved Ahead-Of-Time version, Android Runtime (ART). The UoE team took the opportunity to apply the compiler fuzzing technique to the virtual machine, with the aim of efficiently detecting and eliminating errors that would disrupt Google's transition from Dalvik to ART.

The key idea was to feed structured randomised input sequences to both the old Dalvik and the new ART compilers simultaneously, and to observe the behaviour of the synthetic code snippets on both systems side-by-side. Dalvik was the baseline or "golden standard" for reference implementation, therefore any differences exposed directly pointed to errors in ART. The team identified around 30 distinct bugs, divided into ART implementation errors and incorrect specifications.

The culmination of the research was the tool DexFuzz: a novel piece of software which developers can run to automate the process of "fuzz testing" of ART [3.2]. DexFuzz has since proved central to Google's Android product development, and since the development of Android 5.0 has been applied to identify and eliminate bugs in ART before each new version is released. It is now shipped with the Google Android Open Source Project (AOSP) - the open-source repository for the Android source code (https://github.com/aosp/art/tree/master/tools/dexfuzz).

3. References to the research

3.1. Wagstaff, H., Spink, T., & Franke, B. (2014). Automated ISA branch coverage analysis and test case generation for retargetable instruction set simulators. In *Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2014 International Conference on* (pp. 1-10). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1145/2656106.2656113 **(11 citations; CASES 2014 acceptance rate 33%)**

3.2. Kyle, S., Leather, H., Franke, B., Butcher, D., & Monteith, S. (2015). Application of Domain-aware Binary Fuzzing to Aid Android Virtual Machine Testing. In *Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments* (pp. 121-132). New York, NY, USA: ACM. https://doi.org/10.1145/2731186.2731198 **(11 citations)**

3.3. Cummins, C., Petoumenos, P., Murray, A., & Leather, H. (2018). Compiler Fuzzing through Deep Learning. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp. 95-105). Amsterdam, Netherlands: ACM. https://doi.org/10.1145/3213846.3213848 **(Received Distinguished Paper award at ISSTA 2018; 48 citations; ISSTA 2018 acceptance rate 26%)**

3.4. Agakov, F., Bonilla, E., Cavazos, J., Franke, B., Fursin, G., O'Boyle, M. F. P., Thomson, J., Toussaint, M., & Williams, C. K. I. (2006). Using Machine Learning to Focus Iterative

Optimization. In Proceedings of the International Symposium on Code Generation and Optimization (pp. 295-305). (CGO '06). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/CGO.2006.37 **(448 citations; CGO '06 acceptance rate 30%)**

3.5. Dubach, C., Cavazos, J., Franke, B., Fursin, G., O'Boyle, M., & Temam, O. (2007). Fast Compiler Optimisation Evaluation Using Code-feature Based Performance Prediction. In CF '07 Proceedings of the 4th international conference on Computing Frontiers (pp. 131-142). ACM. https://doi.org/10.1145/1242531.1242553 **(98 citations; CF 2007 acceptance rate 37%)**

3.6. Leather, H., Bonilla, E., & O'Boyle, M. (2009). Automatic Feature Generation for Machine Learning Based Optimizing Compilation. In Code Generation and Optimization, 2009. CGO 2009. International Symposium on (pp. 81-91). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/CGO.2009.21 **(Received Test of Time Award at CGO'19; 148 citations; CGO 2009 acceptance rate 30%)**

Citation counts obtained from Google Scholar 2020-12-03.

Key research grants

European Commission: MILEPOST (035307, GBP294,432)
EPSRC: SUMMER (EP/P003915/1, GBP126,284)

4. Details of the impact

The DexFuzz tool has contributed significantly to the improvement of Android, one of Google's flagship products that currently powers 2,500,000,000 smartphones worldwide [5.1]. The research enabled Google to transition smoothly to an upgraded Virtual Machine (from Dalvik to ART), and has streamlined their bug-fixing engineering processes, once a costly part of Android's contribution to Google's business. It is also now part of Android's Open Source project, helping smartphone manufacturers create increasingly innovative devices.

Android generates a considerable portion of Google's income, accounting for 34% (USD31,000,000,000 [01-2016]) of the company's USD90,000,000,000 (02-2020) revenue in 2016 [5.2, 5.3]. Aside from its economic value, as a household name Android supplies Google with reputational capital. ART is a core Android component that underpins the running of every app installed on a given smartphone. It is consistently faster and more efficient than the previous virtual machine, Dalvik, having been observed to perform a benchmark test over three times faster than Dalvik, while using 20% less memory [5.4, para. 9].

Google stated that in response to University of Edinburgh (UoE) research they [text removed for publication] [5.5]. They elaborate:

> *[text removed for publication].* [5.5, para. 2]

Crucially DexFuzz's role has allowed Google to succeed in avoiding a poor outcome when rolling out ART. While the benchmark tests (above) demonstrate ART is a superior virtual machine, it would have had little value had apps created for Dalvik no longer been able to function on Android phones after the upgrade, or had the transition created bugs. The forum Android Central reports users experiencing bugs in pre-DexFuzz releases of ART, including in WhatsApp and Spotify, as well as user anxiety about the transition [5.6]. Thanks to implementing DexFuzz, disruption to Android users has been avoided.

Google have also made gains in productivity through the new tool, specifically through DexFuzz's ability to root out bugs at an early stage in the continuous ART development process, before software is signed off for shipping. Although it is not possible to quantify the exact cost savings, in 2014 Google released data on their generic bug-fixing costs, which was subsequently quoted by software testing company VectorCAST in a white paper on bug-fixing. "Bugs that resulted from incomplete testing had become one of the biggest barriers to Google's continued success", the paper states [5.7, para. 2]. Approximately 40% of software engineers' time was once dedicated to fixing bugs at Google, with the average cost of fixing a bug being USD1,500 (10-2014) [5.7, para. 3]. Most significantly, the Google figures show that the cost of fixing bugs becomes cheaper the earlier in the development process the bug is caught, varying between USD5 (10-2014) (unit test) and USD5000 (10-2014) (system test) [5.7, table 3]. By employing a tool that tests for bugs far in advance of software being finalised, resources are saved either in engineer time, which can be re-routed to other activity, or in the cost savings of employing fewer engineers. Google also note that the [text removed for publication]. [5.5, para. 2]

Google broadened the impact's reach by making DexFuzz an official part of the Android Open Source Project [5.8], [text removed for publication] [5.5, para. 2]. This means that any company who wishes to design and create an Android smartphone receives free access to DexFuzz for testing their design, and when Google wishes to perform changes or upgrades to Android DexFuzz is routinely employed to test for bugs in the transition. Google confirm:

> *[text removed for publication].* [5.5, para. 2]

Through AOSP DexFuzz now underpins the development and innovation of a considerable portion of the smartphone market. Google Android software powered 85% of smartphones in the world in 2020 [5.9], which includes devices by global smartphone brands such as Samsung, Motorola and LG. DexFuzz formed a crucial component in the transition to the smarter, faster and equally robust Ahead-of-Time ART virtual machine, which in turn allows developers to put their devices to more challenging and innovative uses.

5. Sources to corroborate the impact

5.1. Brandom, R. (2019, May 7). There are now 2.5 billion active Android devices. Retrieved May 12, 2020, from https://www.theverge.com/2019/5/7/18528297/google-io-2019-android-devices-play-store-total-number-statistic-keynote

5.2. Hall, G. (2016, January 22). Legal showdown between Google and Oracle reveals astonishing value of the Android OS. Retrieved March 12, 2020, from https://www.bizjournals.com/sanjose/news/2016/01/22/legal-showdown-between-google-and-oracle-reveals.html

5.3. Clement, J. (2020, February 05). Google: Annual revenue. Retrieved July 15, 2020, from https://www.statista.com/statistics/266206/googles-annual-global-revenue/

5.4. Snell, J. (2014, July 7). Android Runtime Performance Analysis: ART vs. Dalvik. Retrieved May 11, 2020, from https://blog.newrelic.com/technology/android-art-vs-dalvik/

5.5. Letter of corroboration from Google ART TechLead and compiler engineer

5.6. Windows Central - Dynamic. (2013, December 14). Retrieved July 5, 2020, from https://forums.androidcentral.com/google-nexus-5/343787-android-runtime-art.html

5.7. Vector Software. (2014, October 7). Quantifying The Cost of Fixing vs Preventing Bugs. Retrieved March 12, 2020, from https://assets.markallengroup.com/article-images/65147/Vector_PDF.pdf

5.8. Google Git. (2014, November 28). tools/dexfuzz - platform/art - Git at Google. Retrieved October 21, 2020, from

https://android.googlesource.com/platform/art/+/959ffdf65f280ee90b7944a8dd610564e7f99e69

5.9. International Data Corporation. (2020, September 14). Smartphone Market Share - OS. Retrieved October 21, 2020, from https://www.idc.com/promo/smartphone-market-share/os