Université de Namur
Faculté d'informatique
Année académique 2021–2022

Génération de tests unitaires pour simples programmes python

Ortegat Pierre

Promoteur : __________________________ (Signature pour approbation du dépôt - REE art. 40)
Xavier Devroey

Co-promoteur : Benoît Vanderose

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Chapitre 1

Remerciements

Chapitre 2

Résumé

Table des matières

Chapitre 3

Introduction

Chapitre 4

État de l'art

4.1 Méthodes de test

4.1.1 A la main

Le style le plus classique
Prends du temp
biais de confirmation
flackiness
=> techniques automatiques utiles, cqfd

4.1.2 Fuzzing

Quand ca a été inventé, sigification : entrée random dans les progs [13]
Ajdh : fort utilisé dans la sécu dès qu'il y a un user input [16] (fait partie du Microsoft Security Development Lifecycle [22])

Fuzzing en boite noire

Prendre tt le prob et donner à l'aveugle des inputs [13]
Dépend critiquement d'un set de seed valides à la base si on veut etre efficace
important aussi de limiter le bruit inutile et pas générer plein de shizer

Fuzzing grammatical ou en boite grise

greybox fuzzer : [40]
graybox grammar fuzzer : [41]
classic grammar fuzzer :
peach (intégré dans gitlab mtnt) [4]
spike [1]
sulley [3]
other grammar fuzzer : [35]
g fuzzing pour trouver des failles de sécu dans les browser : [20]
g fuzzing trouver bug complexes dans des compilateurs C [38]
g fuzzing pour trouver les bugs dans les proto réseaux [2]
apprentissage auto gramaire : [9]
tracer process pour créer gramaire automatiquement [21]

limité par la grammaire en elle même, plus gros défaut

Fuzzing en boite blanche et exécution symbolique

parser le prog, le faire tourner et tenter de résoudre les conditions pour toucher toutes les branches avec un solveur.

Plus efficace pour un covering complet et pour taper sur toutes les branches et chopper les bugs de meeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

dynamic execution testing : SAGE [19] (symbolic execution x86 level avec opti pour enorme stack traces [16])

qui étends le travail d'autre sur le génération de tests auto [16] [10] [17]

utilisé en prod partout, plus de 100 année machines dans les dents "largest computational usage ever for any Satisfiability-Modulo-Theories (SMT) solver" d'après les auteurs de z3 [28]

4.1.3 Property based

[12]

[30]

[29]

4.1.4 Fault injection

4.1.5 Utilisation conjointe

NOTE : all can be combined to try to be more efficient!! [16] => Hybrid fuzzing

plusieurs approches en meme temps : Portfolio approaches.

Chapitre 5

Développement

5.1 Sélection de stratégie de tests

5.1.1 Innovations

5.2 Compromis

5.2.1 Combinaisons de techniques

5.2.2 Complexité spatiale vs temporelle

5.3 Efficacité & limitations

5.3.1 Résolution des branches

Le problème d'arrêt

5.3.2 Complexité temporelle

5.3.3 Valeurs par défaut

5.4 Intégration

5.4.1 Interface unifiée

5.4.2 Gestion des erreurs

5.4.3 Intégration dans Inginious

Chapitre 6

Conclusion

Chapitre 7

Bibliographie

[1] Fuzzer Automation with SPIKE - Infosec Resources — resources.infosecinstitute.com. `https://resources.infosecinstitute.com/topic/fuzzer-automation-with-spike/`. [Accessed 18-Apr-2022].

[2] Github - aflnet/aflnet : Aflnet. `https://github.com/aflnet/aflnet`. [Accessed 18-Apr-2022].

[3] GitHub - OpenRCE/sulley : A pure-python fully automated and unattended fuzzing framework. — github.com. `https://github.com/OpenRCE/sulley`. [Accessed 18-Apr-2022].

[4] Integrating security into your DevOps Lifecycle — peachfuzzer.com. `http://www.peachfuzzer.com/`. [Accessed 18-Apr-2022].

[5] Andrea Arcuri, Gordon Fraser, and Juan Pablo Galeotti. Automated unit test generation for classes with environment dependencies. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*. ACM, September 2014.

[6] Thanassis Avgerinos, Alexandre Rebert, Sang Kil Cha, and David Brumley. Enhancing symbolic execution with veritesting. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, May 2014.

[7] D. Avresky, J. Arlat, J.-C. Laprie, and Y. Crouzet. Fault injection for formal testing of fault tolerance. *IEEE Transactions on Reliability*, 45(3) :443–455, 1996.

[8] Roberto Baldoni, Emilio Coppa, Daniele Cono D'elia, Camil Demetrescu, and Irene Finocchi. A survey of symbolic execution techniques. *ACM Computing Surveys*, 51(3) :1–39, May 2019.

[9] Osbert Bastani, Rahul Sharma, Alex Aiken, and Percy Liang. Synthesizing program input grammars. *ACM SIGPLAN Notices*, 52(6) :95–110, 2017.

[10] Cristian Cadar and Dawson Engler. Execution generated test cases : How to make systems code crash itself. In *International SPIN Workshop on Model Checking of Software*, pages 2–23. Springer, 2005.

[11] Cristian Cadar, Patrice Godefroid, Sarfraz Khurshid, Corina S. Păsăreanu, Koushik Sen, Nikolai Tillmann, and Willem Visser. Symbolic execution for software testing in practice. In *Proceedings of the 33rd International Conference on Software Engineering*. ACM, May 2011.

[12] George Fink and Matt Bishop. Property-based testing. *ACM SIGSOFT Software Engineering Notes*, 22(4) :74–80, July 1997.

[13] Justin E. Forrester and Barton P. Miller. An empirical study of the robustness of windows nt applications using random testing. In *Proceedings of the 4th Conference on USENIX Windows Systems Symposium - Volume 4*, WSS'00, page 6, USA, 2000. USENIX Association.

[14] Gordon Fraser and Andrea Arcuri. A large-scale evaluation of automated unit test generation using EvoSuite. *ACM Transactions on Software Engineering and Methodology*, 24(2) :1–42, December 2014.

[15] Gordon Fraser, Matt Staats, Phil McMinn, Andrea Arcuri, and Frank Padberg. Does automated unit test generation really help software testers ? a controlled empirical study. *ACM Transactions on Software Engineering and Methodology*, 24(4) :1–49, September 2015.

[16] Patrice Godefroid. Fuzzing. *Communications of the ACM*, 63(2) :70–76, January 2020.

[17] Patrice Godefroid, Nils Klarlund, and Koushik Sen. Dart : Directed automated random testing. In *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, pages 213–223, 2005.

[18] Patrice Godefroid, Michael Y. Levin, and David Molnar. SAGE. *Communications of the ACM*, 55(3) :40–44, March 2012.

[19] Patrice Godefroid, Michael Y Levin, David A Molnar, et al. Automated whitebox fuzz testing. In *NDSS*, volume 8, pages 151–166, 2008.

[20] Christian Holler, Kim Herzig, and Andreas Zeller. Fuzzing with code fragments. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 445–458, 2012.

[21] Matthias Hoschele and Andreas Zeller. Mining input grammars with autogram. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 31–34. IEEE, 2017.

[22] Michael Howard and Steve Lipner. *The security development lifecycle*, volume 8. Microsoft Press Redmond, 2006.

[23] James C. King. Symbolic execution and program testing. *Communications of the ACM*, 19(7) :385–394, July 1976.

[24] Andreas Leitner, Manuel Oriol, Andreas Zeller, Ilinca Ciupa, and Bertrand Meyer. Efficient unit test case minimization. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering - ASE '07*. ACM Press, 2007.

[25] Rupak Majumdar and Koushik Sen. Hybrid concolic testing. In *29th International Conference on Software Engineering (ICSE'07)*. IEEE, May 2007.

[26] Paul D. Marinescu and George Candea. Efficient testing of recovery code using fault injection. *ACM Transactions on Computer Systems*, 29(4) :1–38, December 2011.

[27] Shabnam Mirshokraie, Ali Mesbah, and Karthik Pattabiraman. Jseft : Automated javascript unit test generation. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, pages 1–10, 2015.

[28] Leonardo de Moura and Nikolaj Bjørner. Z3 : An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[29] Manolis Papadakis and Konstantinos Sagonas. A PropEr integration of types and function specifications with property-based testing. In *Proceedings of the 10th ACM SIGPLAN workshop on Erlang - Erlang '11*. ACM Press, 2011.

[30] Zoe Paraskevopoulou, Cătălin Hriţcu, Maxime Dénès, Leonidas Lampropoulos, and Benjamin C. Pierce. Foundational property-based testing. In *Interactive Theorem Proving*, pages 325–343. Springer International Publishing, 2015.

[31] Xiao Qu and Brian Robinson. A case study of concolic testing tools and their limitations. In *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE, September 2011.

[32] José Miguel Rojas, Gordon Fraser, and Andrea Arcuri. Automated unit test generation during software development : a controlled experiment and think-aloud observations. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ACM, July 2015.

[33] Z. Segall, D. Vrsalovic, D. Siewiorek, D. Ysskin, J. Kownacki, J. Barton, R. Dancey, A. Robinson, and T. Lin. FIAT – fault injection based automated testing environment. In *Twenty-Fifth International Symposium on Fault-Tolerant Computing, 1995, ' Highlights from Twenty-Five Years'*. IEEE.

[34] Koushik Sen. Concolic testing. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering - ASE '07*. ACM Press, 2007.

[35] Michael Sutton, Adam Greene, and Pedram Amini. *Fuzzing : brute force vulnerability discovery*. Pearson Education, 2007.

[36] Kunal Taneja and Tao Xie. Diffgen : Automated regression unit-test generation. In *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, pages 407–410, 2008.

[37] Xinyu Wang, Jun Sun, Zhenbang Chen, Peixin Zhang, Jingyi Wang, and Yun Lin. Towards optimal concolic testing. In *Proceedings of the 40th International Conference on Software Engineering*. ACM, May 2018.

[38] Xuejun Yang, Yang Chen, Eric Eide, and John Regehr. Finding and understanding bugs in c compilers. In *Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation*, pages 283–294, 2011.

[39] Andreas Zeller, Rahul Gopinath, Marcel Böhme, Gordon Fraser, and Christian Holler. *The Fuzzing Book*. CISPA Helmholtz Center for Information Security, 2021. Retrieved 2021-10-26 15 :30 :20+02 :00.

[40] Andreas Zeller, Rahul Gopinath, Marcel Böhme, Gordon Fraser, and Christian Holler. Greybox fuzzing. In *The Fuzzing Book*. CISPA Helmholtz Center for Information Security, 2022. Retrieved 2022-01-23 17 :13 :33+01 :00.

[41] Andreas Zeller, Rahul Gopinath, Marcel Böhme, Gordon Fraser, and Christian Holler. Greybox fuzzing with grammars. In *The Fuzzing Book*. CISPA

Helmholtz Center for Information Security, 2022. Retrieved 2022-01-11 09 :26 :47+01 :00.

[42] Hong Zhu, Patrick A. V. Hall, and John H. R. May. Software unit test coverage and adequacy. *ACM Computing Surveys*, 29(4) :366–427, December 1997.

Chapitre 8

Annexes