[1]. M. Zalewski, *American fuzzy lop,* <https://lcamtuf.coredump.cx/afl/>

[2]. Google: honggfuzz (2018). <https://github.com/google/honggfuzz>

[3]. Serebryany K. OSS-Fuzz-Google's continuous fuzzing service for open source software[J]. 2017.

[4]. Gan S, Zhang C, Qin X, et al. Collafl: Path sensitive fuzzing[C]//2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018: 679-696.

[5]. C. Lyu, S. Ji, C. Zhang, Y. Li, W. H. Lee, Y. Song, R. Beyah(2019). MOPT: Optimized mutation scheduling for fuzzers. In *28th USENIX Security Symposium. (USENIX Security 19)* , 1949-1966.

[6]. Böhme M, Pham V T, Roychoudhury A(2017). Coverage-based greybox fuzzing as markov chain. IEEE Transactions on Software Engineering, 45(5): 489-506.

[7]. Baldoni R, Coppa E, D’elia D C, et al. A survey of symbolic execution techniques[J]. ACM Computing Surveys (CSUR), 2018, 51(3): 1-39.

[8]. Majumdar, Rupak , and K. Sen . "Hybrid Concolic Testing." International Conference on Software Engineering IEEE, 2007.

[9]. Xie Xiaofei, Xie Xiaofei, Li Xiaohong, et al. Hybrid testing method based on symbolic execution and fuzzy testing [J].  Journal of Software, 2019, Vol.30Issue(10):3071-3089.

[10]. Stephens N, Grosen J, Salls C, Dutcher A, Vigna G(2016, February). Driller: Augmenting Fuzzing Through Selective Symbolic Execution. NDSS, 16(2016): 1-16.

[11]. Chess B, McGraw G. Static analysis for security[J]. IEEE security & privacy, 2004, 2(6): 76-79.

[12].King J C. Symbolic execution and program testing[J]. Communications of the ACM, 1976, 19(7): 385-394.

[13]. Miller B P, Koski D, Lee C P, et al. Fuzz revisited: A re-examination of the reliability of UNIX utilities and services[R]. University of Wisconsin-Madison Department of Computer Sciences, 1995.

[14]. Peng H, Shoshitaishvili Y, Payer M. T-Fuzz: fuzzing by program transformation[C]//2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018: 697-710.

[15]. You W, Liu X, Ma S, Perry D, Zhang X, Liang B(2019, May). SLF: fuzzing without valid seed inputs. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019: 712-723.

[16]. Pak B S. Hybrid fuzz testing: Discovering software bugs via fuzzing and symbolic execution[J]. School of Computer Science Carnegie Mellon University, 2012.

[17]. QSYM Yun I, Lee S, Xu M, et al. {QSYM}: A practical concolic execution engine tailored for hybrid fuzzing[C]//27th {USENIX} Security Symposium ({USENIX} Security 18). 2018: 745-761.

[18] Chen Y, Li P, Xu J, et al. SAVIOR: towards bug-driven hybrid testing[C]//2020 IEEE Symposium on Security and Privacy (SP). IEEE, 2020: 1580-1596.

[19]. Cadar C, Dunbar D, Engler D R. Klee: unassisted and automatic generation of high-coverage tests for complex systems programs[C]//OSDI. 2008, 8: 209-224.

[20]. PANGOLIN: Incremental Hybrid Fuzzing withPolyhedral Path AbstractionHeqing Huang†, Peisen Yao†, Rongxin Wu‡, Qingkai Shi†, Charles Zhang††The Hong Kong University of Science and Technology, China‡Xiamen University, China