P R O J E C T D O C U M E N T

Test Plan / Test Cases Design Document

Project Name	Fuzz Testing을 통한 위성 SW 분석

05 조

202002473 김승혁
201902733 이정윤
202002699 조민기

지도교수: 이성호 교수님 (서명)

Table of Contents

1. Introduction

1.1. 연구 질문/ 가설

본 연구는 다음과 같은 연구 질문에 답하고자 한다:

- RQ1.
  fprime 위성 비행 소프트웨어 환경에서 다양한 퍼징 도구를 활용한 결함 탐지 활동이 기존의 테스팅 방식에 비해 결함 탐지율에 어떠한 영향을 미치는가?
- RQ2.
  퍼징 도구의 유형(예: Coverage-Guided, Generation-Based)이 fprime의 컴포넌트 기반 아키텍처와 통신 방식(fpp)에 따라 결함 탐지 성능에서 어떤 구체적인 차이를 보일 것인가?

본 연구는 다음과 같은 가설을 설정할 수 있다:

- H1.
  fprime 환경에서 제안하는 퍼징 전략을 활용한 결함 탐지 활동이 기존의 테스팅 방식보다 결함 탐지율을 유의미하게 향상시킬 것이다.
- H2.
  Generation-Based 퍼징 도구는 fprime의 컴포넌트 간 통신 방식(fpp)을 고려할 때, 더 높은 상태 탐색 능력과 결함 탐지 성능을 보일 것이다.

2.Test Plan

1. 배경과 목적
1.1 배경
본 연구는 fprime 위성 소프트웨어에 다양한 퍼징 도구를 적용하여, 기존 테스트 방식 대비 퍼징의 효과성을 실험적으로 분석하고, 퍼저 유형 간의 차이를 구조적으로 비교하고자 한다. 이를 통해 fprime 아키텍처에 적합한 퍼저 전략을 제안하는 것이 목적이다.
2. 테스트 상세
2.1 독립/ 종속 변수 정의
- 독립 변수 - 테스트 방식: 기존 수동 테스트, 퍼저(BooFuzz, AFL++, libFuzzer) 적용 - 퍼저 유형: Generation-Based (BooFuzz), Mutation-Based (AFL++), Coverage-Guided (libFuzzer) - 종속 변수 - 결함 탐지 수: 발견된 충돌, 오류, 로그 기반 비정상 동작 수 - 코드 커버리지: 퍼징 동안 도달한 코드 라인 비율 - 수행 시간: 동일 조건에서 수행된 퍼징 시간 - 자원 사용량: CPU, 메모리 사용량
2.2 실험 대상/ 환경
- 대상 소프트웨어: NASA의 fprime open-source satellite flight software - 모듈: CmdDispatcher, Health, ActiveLogger - 퍼저 도구: BooFuzz, AFL++, libFuzzer - 실행 환경: Ubuntu 20.04 기반 Docker container(4-core CPU, 8GB RAM) - 입력 데이터: fprime 명령어 또는 시퀀스 포맷 기반 입력
3. 테스트 관리
3.1 실험 절차 요약
1. Fprime 대상 모듈을 별도 타겟 바이너리 또는 모듈 단위로 분리 2. 각 퍼저별 테스트 환경 구성(별도 harness 구성 및 입력 corpus 설정) 3. 동일 자원 조건에서 각 fuzzer를 6시간 이상 수행(Docker compose를 통한 병렬 컨테이너 실행) 4. 결함 탐지 수, 커버리지, 자원 사용량 등을 기록 5. 퍼저간 성능을 정량 비교
3.2 측정 지표 및 도구
- 정량 평가 지표, 정성 평가 지표, 사용 도구 등을 명시

- 정량 지표
  - Crash 수: 퍼저 내 크래시 로그 분석
  - 코드 커버리지: llvm-cov, gcov 등의 도구 활용
  - 경로 다양성: 퍼저 내부 상태
  - 자원 사용량: docker stats 명령어 도구 사용
- 정성 지표
  - 로그 분석: 수작업
  - 적용 난이도: 퍼저별 사전 설정 난이도 평가

3.Test Cases

1. 테스트 케이스

1.1 테스트 케이스 명세

Id	대상(모듈)	실험 조건	테스트 데이터	평가지표	예상 결과
TC-H1-1	CmdDispatcher	기존 테스트 코드 실행	정상 명령어 시퀀스	결함 수, 커버리지	낮은 탐지율
TC-H2-1	CmdDispatcher	BooFuzz	유효한 명령어 시퀀스	충돌 수, 커버리지, 경로 다양성	경로 다양성 높음
TC-H2-2	CmdDispatcher	AFL++	기존 시드 시퀀스 파일의 변형본	충돌 수, 커버리지, 경로 다양성	경로 다양성 낮음
TC-H2-3	CmdDispatcher	libFuzzer	무작위 바이트 스트림	충돌 수, 커버리지, 경로 다양성	커버리지 높으나 의미 해석 부족
TC-H2-4	CmdDispatcher	BooFuzz, AFL++, libFuzzer	유효한 CmdDispatcher 제어 패킷	충돌 수, 커버리지, 경로 다양성	도구별 탐지 차이 관찰
TC-H2-5	Health	BooFuzz, AFL++, libFuzzer	유효한 Health 제어 패킷	충돌 수, 커버리지, 경로 다양성	도구별 탐지 차이 관찰
TC-H2-6	ActiveLogger	BooFuzz, AFL++, libFuzzer	유효한 ActiveLogger 제어 패킷	충돌 수, 커버리지, 경로 다양성	도구별 탐지 차이 관찰

1.2 검증 기준(metirc)

- 평가 기준 표로 제시(로직 설명)

지표	정의	측정 방법	단위
충돌 수	퍼징 중 발견된 고유한 충돌 수	퍼저 내 크래시 로그	건
코드 커버리지	실행된 소스 코드의 라인/분기 비율	gcov, llvm-cov 도구 활용	%
경로 다양성	퍼저가 탐색한 고유 실행 경로 수	퍼저 내부 경로 해시, 커버리지 map	건
자원 사용량	평균 CPU 및 메모리 사용량	docker stats	%, MB

4.AI 도구 활용 정보

사용 도구	*ChatGPT*
사용 목적	*문장 흐름 정리 및 내용 정정*
프롬프트	● *문서를 살펴보고, 틀린 내용이 있는지 알려줘*
반영 위치	*1. 테스트 케이스 목록*
수작업 수정	*있음(논리 보강, 사례 교체 등)*