RoIFuzz: 강화된 로봇 보안 정책을 적용한 ROS IDL 퍼저 연구

박민건*, 유지현*, 윤주범**

*, **세종대학교 (대학원생, 교수)

RoIFuzz: Enhancing ROS IDL Fuzzer with Strengthened Robot System Policies

MinGeon Park*, Jihyeon Yu*, Joobeom Yun**

*, **Sejong University(Graduate student, Professor)

요 약

로봇 시스템의 복잡성과 자율성이 높아짐에 따라 보안 안정성 확보를 위한 효과적인 로봇 보안 기술의 수요가 증가하고 있다. 최근에는 선제적 보안 취약점 탐지 기술인 퍼징 기법을 로봇 시스템에 적용하려는 연구가 적극적으로 시도되고 있는데, 이 중 RoboFuzz는 ROS(Robot Operating System) 기반의 로봇 시스템을 위한 효과적인 퍼징 아키텍처를 제안했으나, ROS 기반의 상용 로봇에 초점을 맞추고 있어 ROS 자체 패키지에 대한 취약점 검사를 제한적으로 시행하여 보안 안정성 확보에 한계가 있다. 본 논문에서는 이러한 한계를 극복하고자 RoboFuzz의 아키텍처를 기반으로 ROS IDL(Interface Definition Language)에 대한 퍼징 정책을 강화하여 새로운 취약점을 발견할 수 있는 퍼징 시스템을 제안한다.

I. 서론

*로봇 기술의 발전으로 로봇 시스템의 복잡성과 자율성이 증가하면서 보안 위협 또한 급증하고 있다. 특히 인간과 밀접하게 상호작용하는 로봇 시스템의 증가로 보안 안정성 확보가 중요한 문제로 대두되고 있다. ROS1의 네트워크 취약점 사례와 이를 해결하기 위한 ROS2, SROS2 등의 패키지 단위의 취약점 탐지를 위한 노력이 있었으나 주로 네트워크 취약점에 초점을 맞춰서 발전하고 있다는 한계점이 있다.

최근 로봇 하드웨어 자체에 대한 보안을 사전에 탐지하려는 목적으로 선제적 취약점 탐지 기법인 퍼징을 로봇 시스템에 적용하는 시도가 활발히 이뤄지고 있다, 퍼징은 무작위 입력을 통해 알려지지 않는 취약점을 발견하는 기술로 오랜 역사를 자랑하지만, 로봇 시스템에 대한 퍼징 연구는 비교적 최근에 시작되었다. 로봇 시스템은 사이버-물리 시스템이라는 복합적인 특징을 가지고 있어서 기존의 퍼징 기법을 로봇 시스템에 이식하는 연구는 아직 많은 도전 과제가 남아 있다. 그중 RoboFuzz는 로봇 시스템의 특징을 고려한 효율적인 취약점 탐지 방법을 제시했다. 그러나 ROS 내부 패키지에 대한 취약점 탐지가 제한적이라는 한계가 있다. 특히 ROS IDL 패키지는 메시지 타입의 정의에 중요한 역할을 하지만 현재 충분한 정도의 보안 검사가 이뤄지지 않고 있다. 따라서 본 논문에서는 RoboFuzz의 아키텍처를 기반으로 ROS IDL의 보안 정책을 강화하여 ROS 내부 패키지 취약점을 탐지하는 새로운 로봇 퍼저인 ROIFuzz(Robot Interface Fuzzer)를 제안한다.

II.배경지식

2.1 ROS2 (Robot Operating System2)

* "본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터사업의 연구결과로 수행되었음"
(IITP-2024-RS-2024-00437494)

ROS[1]는 노드(node)라는 독립 프로세스로 구성된 분산 시스템을 기반으로 메시지(message)라는 정보의 단위를 토픽(topic), 서비스(service), 액션(action)이라는 다양한 형태로 전송한다. 이 전송을 위해 DDS(Data Distribution Service) 기반의 RTPS(Real-Time Publish-Subscribe) 프로토콜을 채택하여 노드 간의 발행자-구독자 관계를 기반으로 통신을 수행한다.

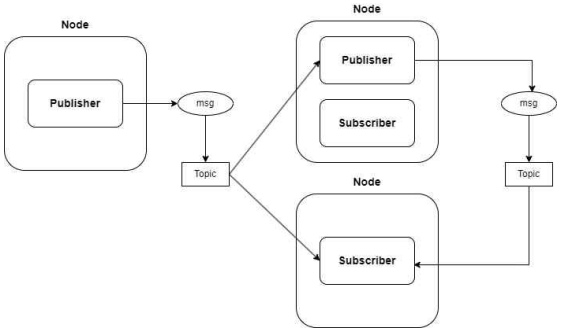

그림 1. Topic 통신방식의 node 예

모든 노드는 그림1처럼 발행자에 의해 정보(message)를 생성하고 RTPS의 발행자-구독자 관계를 이용하여 구독자에게 전송한다. 로봇 퍼징은 RTPS 프로토콜의 가장 기본적인 형태인 토픽 전송기법을 이용하여 노드의 메시지를 변환하면서 무작위 입력을 전송하는 방식으로 동작한다.

2.2 IDL(Interface Definition Language)

ROS IDL은 ROS2 시스템에서 노드 간의 통신에 사용되는 메시지, 서비스, 액션의 구조를 정의하는 언어이다. ROS1의 msg, service, action을 통합하여 정의하는 추상화된 문법을 제공하는 Interface Language로 node의 구조를 정의할 때 실질적인 자료형을 지정한다.

2.3 퍼징(Fuzzing)

퍼징(Fuzzing)은 무작위로 생성된 데이터를 프로그램에 입력하여 취약점을 발견하는 소프트웨어 테스트 기법이다. 예상치 못한 입력 값에 대한 프로그램의 반응을 확인하고 잠재적인 버그나 보안 취약점을 사전에 식별하여 소프트웨어의 안정성과 신뢰성 향상에 목적을 두고 있다.

로봇 시스템의 퍼징은 사이버-물리의 복합적인 시스템으로 기존의 퍼징과는 다르게 로봇 시스템의 분산 프로세스를 고려하여 노드에 전달되는 메시지를 변조하는 방식으로 퍼저를 구성해야 한다.

2.4 RoboFuzz

Algorithm 1 ROS IDL Correctness Oracle on RoboFuzz

1: **if** f_1에 $t \neq t'$인 t' 타입의 값이 할당됨 **then**
2: 할당 실패
3: **else if** f_1에 $\max(t_1)$보다 큰 값이 할당됨 **then**
4: 할당 실패
5: **else if** f_1에 $\min(t_1)$보다 작은 값이 할당됨 **then**
6: 할당 실패
7: **else if** f_2의 원소 중 하나에 $t_2 \neq t'$인 t' 타입의 값이 할당됨 **then**
8: 할당 실패
9: **else**
10: 할당 성공
11: **end if**

알고리즘 1. RoboFuzz의 IDL 오라클

RoboFuzz[2]는 ROS 기반 로봇 시스템을 위한 피드백 기반 퍼징 프레임워크이다. 로봇 시스템을 하나의 상태를 가진 노드로 정의하여 정확성 오라클을 기반으로 실행 상태를 검사하여 semantic feedback mechanism을 이용한 score를 산출해 효율적으로 정확성 버그(correctness bug)를 탐지하는 시스템이다. 주로 ROS 기반의 소프트웨어를 채택한 로봇 시스템(PX4, MI2, TSM)을 대상으로 구현되었으나, 다른 실험군에 대비해 IDL의 실험 오라클은 Algorithm 1처럼 간단한 보안 검사만 진행하고 있어 충분한 취약점을 탐지하기엔 어려움이 있다. 본 논문에서는 몇 가지 추가 검사를 진행하는 강화된 보안 정책을 적용한 RoIFuzz를 제안한다.

III. RoIFuzz

3.1 RoIFuzz 구현

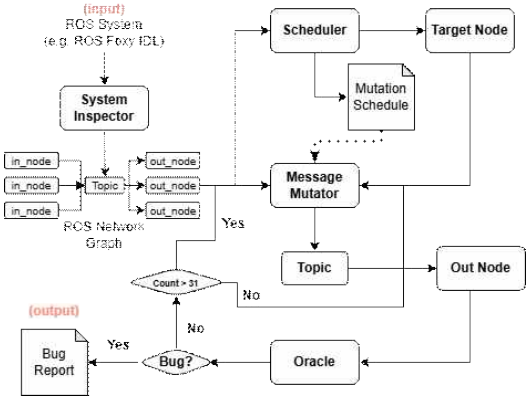

그림 2. RoIFuzz 아키텍처

RoboFuzz는 Algorithm 1의 불변 조건에 따라 타입의 일치, 최대·최소 준수, 고정 크기 배열의 타입 일치를 검사하고 그 외의 값은 할당 성공한다. 이 과정은 4가지 검사가 성공적으로 실패해서 IDL 타입 시스템이 예상대로 동작함을 검증하는 요소다. 그러나 타입 시스템의 복잡성과 로봇의 특수성을 고려하기에는 몇 가지 한계점이 존재하고 보안 안정성 강화를 위해 더욱 견고한 검증 모듈의 구성이 필요하다. 따라서 본 논문은 아래의 개선사항을 도입하여 ROS IDL의 보안 정책을 강화하고자 한다.

Algorithm 2 개선된 ROS IDL Correctness Oracle

```
function CHECK(config, msg_list, state_dict, feedback_list)
    errs ← 빈 리스트
    msg ← 테스트할 메시지 선택
    msg_name ← msg의 타입 이름
    topic_name ← "/idltest_" + msg_name + "_out"
    if topic_name이 state_dict에 없음 then
        errs에 "Topic {topic_name} is lost" 추가
    else
        msg_out_list ← state_dict[topic_name]
        if msg_out_list의 길이 ≠ 1 then
            "[-] multiple messages replayed by idl target" 출력
            프로그램 종료 (-1)
        end if
        (ts, msg_out) ← msg_out_list[0]
        if msg의 타입 ≠ msg_out의 타입 then
            errs에 "Message types do not match" 추가
        else if msg 또는 msg_out에 'data' 속성이 없음 then
            errs에 "Invalid message structure: 'data' attribute missing" 추가
        else
            if msg.data의 타입이 배열 then
                if msg.data의 길이 ≠ msg_out.data의 길이 then
                    errs에 "Sent and replayed array lengths do not match" 추
가
                else
                    for i ← 0 to msg.data의 길이 - 1 do
                        if msg.data[i]의 타입 ≠ msg_out.data[i]의 타입 then
                            errs에 "Array element types do not match at index
{i}" 추가
                        else if msg.data[i] ≠ msg_out.data[i] then
                            errs에 "Array element values do not match at index
{i}" 추가
                        end if
                    end for
                end if
            else
                if msg.data ≠ msg_out.data then
                    errs에 "Sent and replayed data do not match" 추가
                end if
            end if
        end if
    end if
    중복 제거된 errs를 err-report 파일에 기록
    return errs
end function
```

알고리즘 2. 강화된 정책의 IDL Oracle

첫째, 타입 일치 검사의 정교화를 위한 노드 간 일치 검사 추가. 단순 비교 연산 대신 메시지 타입에 대한 명시적 검사를 추가했다. 그리고 노드 간의 메시지 타입 불일치를 더 정확하게 탐지한다. 이를 통해 ROS IDL 시스템의 신뢰성을 검증할 수 있으며 복잡한 사용자 정의 타입의 데이터 검사에 효과적으로 대응할 수 있다.

둘째, 특수값과 범위 검사를 통한 경계 검사 강화. 기존의 불변식의 max와 min에 대한 경계 검사를 범위 단위 검사로 확장하여 더욱 정밀한 데이터 검증을 수행한다. 구체적으로 특수값 검증을 추가하고 입력이 범위 내에 있는지 확인한다. inf와 -inf는 ROS의 표준에 올바른 값이지만 입력 mutation이 너무 크거나 작으면 IDL 의해 임의로 변경되는 문제가 있어 노드 간의 불일치의 발생을 일으켰다. 따라서 inf와 -inf에 대한 검사를 따로 분리하고 기존의 큰 값은 에러로 변경하였다.

셋째, 배열 타입 불일치 검사. 기존 RoboFuzz에서도 다양한 배열 타입에 대한 검사가 적절히 이뤄지지 않아 다른 타입의 데이터를 저장할 수 있는 취약점을 발견한 바 있다. 이 취약점은 ROS foxy에서 가장 많이 발견되는 취약점으로 강화된 타입 검사를 적용하면 시스템의 견고성을 높이는데 기여할 수 있다. 기존의 if 검사를 isInstance 함수를 이용한 검사로 대체하고 각 타입을 ROS IDL 표준에 맞는 numpy에서 제공하는 정확도가 높은 타입으로 변경하여 각 요소의 타입에 정확히 할당이 이뤄졌는지 검증해볼 수 있다.

넷째, 데이터 구조의 유효성 검사. data 속성의 존재 여부를 확인하는 유효성 검사가 없어 사이클이 생략되거나 잘못된 메시지 구조로 인한 오류가 발생할 수 있는 여지가 있었다. 이걸 사전 방지하기 위해 데이터 구조에 대한 유효성을 검사한다. 일반적인 형태의 데이터 구조를 검사하고 추후 개선을 통해 복합적인 형태를 가진 데이터 구조 검사를 추가하는 것이 목표이다.

마지막으로 에러 메시지와 에러 리포트를 개선한다. 기존의 RoboFuzz는 에러를 서브 프레임과 함께 파일로 제공하는데, IDL 테스트의 경우 더미 메시지 기반으로 하나의 메시지만 가진 노드를 이용해 테스트를 진행한다. 따라서 하나의 검사 사이클에서 발생한 에러를 거시적으로 파악하기 어려운 문제가 있었다. 이를 중복제거와 err-report라는 파일에 취합하여 생성하는 방법으로 리포트를 생성하는 과정을 개선했다.

3.2 RoIFuzz 실험 및 평가

Table 1: RoIFuzz에 의해 발견된 ROS IDL 버그 목록

#	버그 설명
1	32/64비트 float 경계 미확인, 모든 float을 double로 취급
2	byte 타입 오처리, 내부적으로 문자열 리터럴로 취급
3	int 배열 요소 데이터 범위 검사 누락
4	float 배열 요소 데이터 범위 검사 누락
5	배열 요소 암시적 타입 변환으로 인한 데이터 변경
6	bool 배열 요소 타입 검사 누락
7	byte 배열 요소 타입 검사 누락
8	string 배열 요소 타입 검사 누락
9	메시지 구조 유효성 검사, 노드 생성 시 메시지 구조 검사
10	특수값 검사, -inf와 inf에 대한 특수값 검사를 추가하여 엄밀히 구분
11	BoundedDynArray 배열 요소 타입 검사 누락

RoIFuzz: Robot Interface Fuzzer

표 1. RoIFuzz ROS IDL 버그 목록

실험환경. 본 연구의 실험환경은 RoboFuzz 저자의 Docker Container를 기반으로 진행했다. Docker 환경은 Ubuntu 20.04, ROS2 Foxy, FastDDS, Python3로 구성되어 있다. 실험 대상은 ROS IDL 시스템이며, RoIFuzz를 이용하여 12시간 동안 취약점 탐지 테스트를 진행했다.

실험흐름. 그림 4의 RoIFuzz의 흐름처럼 IDL 시스템을 입력하면 System Insepctor를 통해 ROS Network Graph로 변환되어 list로 반환된다. 이 리스트는 스케줄러에 의해 대상 노드와 스케줄이 정해지고 Message Mutator에 의해 메시지를 생성한다. 생성된 메시지는 Oracle에 의해 Bug인지 검증되어 리포트로 반환한다. 설정한 count만큼 퍼저가 동작하면 새로운 노드를 무작위로 선택하여 시간이 끝날 때까지 반복한다.

취약점 분류	RoIFuzz	RoboFuzz
데이터 타입 처리	3	2
배열 요소 타입 검사 누락	6	6
메시지 구조 검증	1	0
특수 값 검사	1	0
합계	**11**	**8**

표 2. RoIFuzz와 RoboFuzz 취약점 항목 비교

실험결과 및 평가. RoIFuzz에 의해 표1과 같이 총 11개의 취약점을 발견했다. 표2는 발견된 취약점 목록으로 기존의 RoboFuzz에서 발견한 8개의 취약점과 추가로 발견한 3개의 취약점으로 구성된다. 특히 배열 타입 처리에 관련된 취약점이 다수 발견되었고 이는 ROS2의 메시지 처리 과정에서 배열 데이터 구조의 취약점이 높다는 것을 보여준다. 이 실험 결과는 ROS가 메시지를 처리하는 과정에서 보안 검사나 유효성 검증을 충분히 하지 않음을 알 수 있다. 따라서 ROS 내부 보안 강화를 위해 더욱 엄격한 보안 검증이 필요하다.

IV. 결론

본 논문에서는 강화된 로봇 정책을 통해 기존 RoboFuzz 불변식을 강화한 RoIFuzz를 제시하였다. 실험 중 11가지 Bug를 검출하여 로봇 시스템의 보안 안정성 강화에 이바지하였다. 그러나 IDL 검사는 로봇 시스템의 복잡성으로 인해 여전히 한계점이 명확하다는 문제점이 있어 이를 개선하기 위해 강화학습을 도입하여 복합 메시지를 생성하는 방안을 향후 연구과제로 두고 있다.

[참고문헌]

[1] M. Quigley et al., "ROS: an open-source Robot Operating System," in Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics, Kobe, Japan, May 2009.

[2] S. Kim and T. Kim, "RoboFuzz: Fuzzing Robotic Systems over Robot Operating System (ROS) for Finding Correctness Bugs," in Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software