정 보 보 호 학 회 지
제 31 권 제5호, 2021. 10

시뮬레이션 환경에서 오픈소스 드론 제어권 획득 취약점 분석

김 숙 영*, 이 대 현*, 서 형 민**, 문 종 섭***

요 약

오픈소스 드론 펌웨어는 많은 민간 회사의 드론 OS의 기반이 되는 등 그 활용처가 매우 넓으며, 오픈소스 드론 커뮤니티가 활성화됨에 따라 드론 펌웨어에 대한 일반인들의 접근성 또한 높아졌다. 반면, 오픈소스 드론의 뛰어난 접근성에 반비례하여 취약점에 대한 분석이 미비하다. 이에 본 논문에서는 오픈소스 드론의 동적 분석을 위해 SITL과 HITL 시뮬레이션 환경을 구축하였으며, 구축된 환경을 활용하여 운용상 발생 가능한 제어권 획득 취약점을 연구하였다. 그 결과 드론의 펌웨어를 수정하여 다중 접속 및 제어권 획득을 성공하였으며, 모뎀으로 통신하는 드론의 경우 펌웨어 수정 없이도 다중 접속 및 제어권 탈취가 가능함을 확인하였다.

Ⅰ. 서 론

오픈소스 드론 커뮤니티가 활성화되고 발전함에 따라, 세계 각국의 유명 드론 업체에서도 오픈소스 기반의 드론 OS를 개발하여 출시할 정도로 오픈소스 드론 펌웨어의 영향이 높아졌다[1]. 반면, 오픈소스의 특성상 보안에 취약하고, 검증되지 않은 오픈소스는 해커와 같은 외부 공격에 취약점을 노출시킬 수 있다[2].

따라서 보안 사고를 방지하기 위해서는 보안 검증이 필수이나, 공간적, 비용적 제약으로 인해 실제 환경에서 오픈소스 드론의 취약점을 분석하기가 어렵다. 이에 본 논문에서는 드론이 실제 환경과 흡사한 동작을 수행할 수 있는 시뮬레이션 환경을 구축하여 오픈소스 드론의 제어권 획득 취약점을 분석하였다. 제어권 획득 취약점이란 정상사용자가 운용 중인 드론의 제어권을 획득하여 공격자의 의도대로 드론 기체를 동작시키는 취약점이다. 본 논문에서는 정상사용자와 동시에 드론에 접속하는 다중 접속을 통한 제어권 획득과 정상사용자와 드론의 통신을 해제하고 공격자만 드론에 접속하여 단독제어하는 제어권 탈취 형태의 취약점을 분석하였다. 해당 취약점은 공격 대상 드론의 데이터를 획득하거나 기체를 무력화하는데 그치지 않고, 비행 중인 드론의 제어권을 완전히 획득 또는 탈취할 수 있다. 또한, 사실상의 표준인 MAVLink 프로토콜[11]을 사용하는 드론에 대해 공격이 가능하므로 일반적인 적용이 가능하다[5].

본 논문의 구성은 다음과 같다. 2장에서는 오픈소스 드론에 대한 취약점을 분석한 이전 연구들에 대해 간단히 소개하고 3장에서는 시뮬레이션 환경의 구성 요소와 본 논문에서 사용하는 시뮬레이터에 대해 설명하였다. 4장에서는 소프트웨어로만 구성된 software in the loop(SITL) 시뮬레이션[3]와 소프트웨어 및 하드웨어로 구성된 hardware in the loop(HITL) 시뮬레이션[4] 환경를 구축한 후, 해당 환경에서 오픈소스 드론의 제어권 획득 가능성을 실험하여 다중 접속 및 단일 접속을 통한 제어권 획득이 가능함을 확인하였다. 5장에서는 본 연구의 결론을 제시한다.

이 연구는 ETRI부설연구소의 위탁연구과제[2020-040]로 수행한 연구결과입니다.
* 고려대학교 정보보호대학원 (대학원생, sykim1223@korea.ac.kr; 대학원생, msms1009@korea.ac.kr)
** ETRI 부설연구소 (연구원, sylar@nsr.re.kr)
*** 고려대학교 정보보호대학원 (교수, jsmoon@korea.ac.kr)

Ⅱ. 관련 연구

2.1. 오픈소스 드론 취약점 분석

Kwon[5]은 드론 통신에 사용되는 MAVLink 프로토콜[6]의 암호화되지 않은 메시지를 이용하여 드론을 비활성화하는 공격 방법을 제시했다. ICMP 플러딩 공격을 수행하여 명령 패킷 수신을 지연시키고, 프로토콜의 취약성을 이용하여 악성 패킷을 삽입하여 드론을 비활성 상태에 진입하게 만드는 방법을 제안하였다. Kwon이 제시한 방법은 MAVLink 프로토콜로 통신하는 모든 드론에 적용 가능하며 공격 거리의 제약이 없다. 그러나 드론에 비활성 상태에서 자동으로 RTH (Return To Home) 기능이 활성화되면 저장된 홈 위치로 되돌아가게 되어 공격이 무효화 된다. 반면 본 논문에서 제안한 방법은 드론의 상태를 활성 상태로 유지하고 제어권을 획득하여 희생자로부터 기체를 탈취할 수 있다.

Rodday[7]는 중간자 공격을 수행하여 드론을 비활성화하는 실험을 수행하였다. 보안 설정이 약한 드론의 네트워크에 참여하여 드론의 통신 패킷을 수집하고, 드론의 비행 소프트웨어를 리버스 엔지니어링하여 만든 새로운 패킷을 전송함으로써 드론을 비활성화하였다. 수 킬로미터 밖에서도 공격을 수행할 수 있으나, 와이파이 통신을 사용하는 드론에만 유효하다는 한계점이 있다. 반면, MAVLink 프로토콜은 대다수의 드론에서 지원하기 때문에[13], 본 논문에서 제안한 방법은 보다 일반적으로 적용가능하다.

Arteaga[8]는 GPS 스푸핑을 수행하여 대상 드론을 사용자가 의도하지 않은 위치로 이동시키는 방법을 제안하였다. 가짜 GPS 좌표를 학습하여 대상 드론에게 전송하고 이를 위성이 보낸 진짜 신호로 착각하게 만듦으로써 공격자가 의도한 위치로 드론을 이동시키는데 성공하였다. 비행 중인 드론을 물리적으로 탈취할 수 있으나 이를 위해서는 특정 위치의 GPS 좌표를 사전에 학습하여야 하기에 공격의 적용이 어렵다는 단점이 있다. 반면, 본 논문에서 제어권을 획득하는 방법의 경우 사전 작업이 없이 비행 중인 드론에 공격을 수행하여 기체를 탈취할 수 있다.

Lee[9]는 오픈소스인 SikRadio를 이용하는 드론에 대해 telemetry hijacking 수행 절차와 드론 운용 시 발생가능한 공격 시나리오를 제안하였다. 오픈소스 SikRadio 펌웨어를 변조하여 통신 인증을 우회하고 MAVLink 쉘을 장악하는 방법을 보였으며, MAVLink 쉘에서 악성코드를 실행하여 드론을 무력화시키거나 홈 좌표를 변조하는 방법을 제시하였다. NetID 정보를 사전에 알고 있지 않아도 공격이 적용가능하지만, 사용자가 드론을 조종기로 제어하는 경우 변조한 홈 좌표로 자동 복귀하지 않는다는 한계가 있다. 반면, 본 논문에서 제안하는 방법을 활용하면 사용자의 드론 제어권을 완전히 박탈시키고 드론의 제어권을 완전하게 획득할 수 있다.

Ⅲ. 시뮬레이션 환경 구성 요소

본 절에서는 논문에서 구축한 시뮬레이션 환경을 구성하는 요소들을 설명한다. 먼저 1절에서 취약점 분석 대상 오픈소스 드론을 소개하고, 이후 2절에서 오픈소스 드론을 제어하는 지상 제어 시스템(Ground Control Station, GCS)에 대해 기술하였다. 그리고 3절에서 본 논문이 활용하는 시뮬레이터를 설명한다.

3.1. 오픈소스 드론

3.1.1. 비행제어 펌웨어(PX4)

PX4는 3DR, 인텔 등 많은 민간 회사로부터 지원을 받는 Dronecode 프로젝트[10]의 일부이며 현재 가장 널리 사용되고 있는 오픈소스 드론 비행제어 소프트웨어이다. 이에 본 연구에서는 프로젝트 규모와 드론 시장에서의 영향력 등을 고려하여, 오픈소스 드론 펌웨어 중 Dronecode 프로젝트 하위의 PX4를 연구대상으로 선정하였다. 해당 프로젝트는 취리히 연방 공과대학교(ETH Zurich) 출신의 로렌츠 마이어(Lorenz Meier)가 중심이 되어 진행 중인 자동항법 시스템으로, 학계와 여러 커뮤니티에 표준화된 자동조종 장치를 제공하는 것을 목표로 하고 있다.

PX4는 크게 저장소, 드라이버, 외부 연결, 비행 제어 모듈로 구성되어 있으며, 각 모듈들은 uORB(Micro Object Request Broker) 메시지 버스를 통해 통신한다. 각 블록에 대한 설명은 아래 표 1.과 같다.

PX4는 시뮬레이터 또는 GCS와 통신을 수립하고 이들로부터 제어 명령을 전달받는다. 따라서 PX4와

〔표 1〕 각 PX4 블록의 기능

Block	Descriptions
Storage	Storing the drone's internal status information, parameter values, and log information.
Drivers	Hardware control such as cameras, GPS, RC signals, IMUs and other sensors.
External Connectivity	Responsible for mavlink protocol communication and Fast RTPS protocol communication for video transmission
Flight Control	In charge of flight control by applications such as commander, mc_pos_control, and ekf2
Message Bus	Responsible for communication between each module.

연결을 수립하고 타 사용자의 연결을 해지하면 드론의 제어권을 획득할 수 있게 된다. 이에 본 PX4의 통신 수립 방식상의 문제를 이용하여 드론의 제어권을 획득하기 위부 연결 부분을 중점적으로 분석하였다.

3.1.2. MAVLink 통신

MavLink는 소형 무인 장치들 및 자체내의 다른 내부 컴포넌트와 통신하기 위해 설계된 소형 비행체 통신 라이브러리(Micro Air Vehicle Communication Protocol)이다. 2009년 처음 배포되었고, Dronecode의 하위 프로젝트로 운영 중이다. MAVLink는 경량화된 메시지 프로토콜로, 프로토콜의 메시지는 XML 파일로 정의되며, 한 패킷에 최소 8Byte에서 최대 263Byte를 한 번에 전송가능하다.

또한, 다수의 GCS 소프트웨어가 MAVLink 프로토콜을 지원하고 있어, 별도의 설정 없이도 드론과 연결이 가능하다[12]. MAVLink 프로토콜은 제한된 통신 대역을 가지는 애플리케이션에 최적화되어 있으며, 제한된 RAM과 플래시 메모리의 제약이 있는 시스템에 최적화되어 있다.

3.1.3. Pixhawk 비행 제어 보드

Pixhawk2[13]는 비행 제어 보드 하드웨어로서, Dronecode의 하위 프로젝트인 PX4와, Ardupilot 프로젝트의 기반이 된다. Pixhawk2는 오픈소스 특성상 개발자들에 의해 수많은 파라미터들이 정의되어, 헬기나 드론 이외에도 RC카, 보트 등 다양한 플랫폼에 적용가능하다. 본 연구에서는 HITL 시뮬레이션을 위해 비행 제어 보드로는 pixhawk2를 사용하고, 시뮬레이터로는 jMAVSim을 사용하였다. pixhawk2는 자동 비행 기능과 멀티스레딩을 이용한 개발 환경을 제공하며, 센서 또는 UART와 같은 주변 장치를 자동으로 탐지한다는 특징이 있다. 또한 추가적인 입출력 메모리 및 기타 기능을 갖춘 단일 보드로 통합되어 있다.

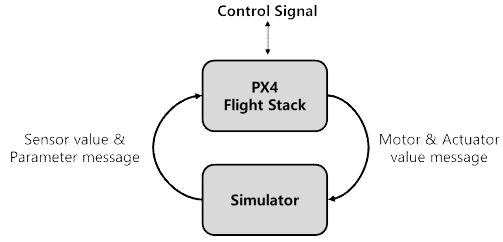
3.2. 지상 제어 시스템 (Ground Control Station)

지상 제어 시스템(Ground Control Station, GCS)은 드론을 운용하기 위해 필요한 시스템이다. 드론과 마찬가지로 Mission Planner[14], QGroundControl[15], APM Planner2[16] 등 다양한 오픈소스 지상 제어 소프트웨어가 존재하며, 공통적으로 MAVLink 프로토콜을 사용하므로 어떤 지상 제어 소프트웨어를 사용하더라도 동일한 기능을 사용할 수 있다[12]. 본 논문에서는 오픈소스 지상 제어 시스템 중, PX4와 마찬가지로 DroneCode의 하위 프로젝트로 속한 QGroundControl을 연구에 활용하였다.

3.3. 시뮬레이터

본 논문에서 구축한 드론 시뮬레이션 환경은 크게 드론 펌웨어(PX4), GCS, 시뮬레이터로 구성된다. 그림. 1.은 시뮬레이터와 PX4 사이의 메시지 흐름을 나타낸 그림이다.

그림. 1.에서 사용자가 GCS로 PX4에 조종 신호를 명령하면, PX4로부터 생성된 모터 및 액추에이터 값 메시지가 시뮬레이터로 전달된다. 시뮬레이터는 수신한 값에 대해 생성된 센서 값 및 파라미터 메시지를 PX4로 전달한다. 이때, 시뮬레이터와 PX4는



〔그림 1〕 시뮬레이터와 PX4 간 인터페이스

MAVLink API를 사용하여 통신한다. 시뮬레이션 시 생성된 내부 펌웨어의 파라미터 값과 비행 중 저장된 로그로 비행 분석이 가능하다.

시뮬레이션 환경은 크게 SITL(Software In The Loop)과 HITL(Hareware In The Loop) 환경으로 나눌 수 있다. SITL 시뮬레이션은 드론 펌웨어, GCS, 시뮬레이터 등 소프트웨어만으로 구성되며, HITL 시뮬레이션 환경은 SITL 시뮬레이션의 구성요소에 pixhawk 비행 제어 컴퓨터를 추가하여 구성된다. HITL 시뮬레이션의 경우 pixhawk 비행 제어 컴퓨터 내부에서 센서가 동작하고, 센서값을 계산하여 시뮬레이터에 전달하여 보다 실제 환경과 유사한 실험이 가능하다.

본 논문에서 실험에 사용될 pixhawk 비행제어 컴퓨터는 Gazebo[17] 시뮬레이터 및 jMAVSim[18] 시뮬레이터와 연동이 가능하다. jMAVSim은 Java기반의 시뮬레이션으로 운영체제에 독립적이고 매우 간단하여 변경이 용이하다. Gazebo는 jMAVSim에 비해 사용법이 복잡하지만 프로펠러의 공역학적 동작 등을 표현할 수 있다[19]. 본 논문에서는 드론의 취약점 분석을 위한 펌웨어 변형에 중점을 두었기에 jMAVSim을 시뮬레이터로 이용하였다. jMAVSim은 많은 대다수 드론의 비행형태인 쿼드콥터 드론을 지원하며, 소스코드를 제공하기에 필요한 기능을 추가하기에 용이하다.

Ⅳ. 시뮬레이션 환경에서 제어권 획득 취약점 분석

4.1. 시뮬레이션 구축 환경그림의 사용 예

시뮬레이션은 표 2.와 같은 가상 환경에서 대상 펌에어 및 하드웨어를 이용하여 구축하였다.

〔표 2〕 분석 환경 및 대상

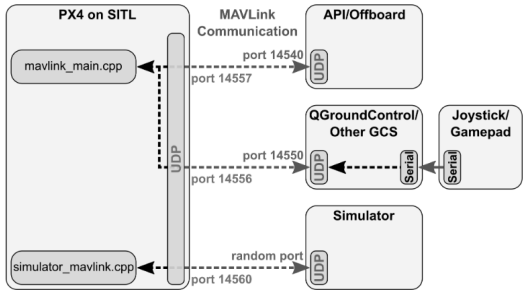
Environ-ment	VMware Ubuntu 18.04 LTS 64 bit
Target	PX4 firmware version 1.8.2
Board	Pixhawk2

4.2. SITL 환경에서의 분석

본 절에서는 오픈소스를 활용하여 SITL 시뮬레이션 환경을 구축하고 다중 접속을 통한 제어권 획득 실험을 수행한다. SITL 시뮬레이터는 실물 하드웨어 없이 드론을 시뮬레이션할 수 있는 환경을 제공한다. SITL 시뮬레이션 환경은 기본적으로 시뮬레이션 환경에서 동작할 드론 펌웨어인 PX4, 드론 기체의 동작을 시뮬레이션하는 시뮬레이터, 시뮬레이션 환경에서 드론의 비행을 제어하고 로그를 확인할 수 있는 GCS로 이루어진다.

4.2.1. 시뮬레이션 환경

그림. 2.는 SITL 시뮬레이션 환경의 연결 구조를 도식화한 그림이다. 시뮬레이션 환경에서 핵심 구성요소인 QGroundControl과 PX4, 시뮬레이터가 서로 UDP통신을 하고 있으며, PX4는 고정 포트 번호를 사용하고 있다. UDP는 통신 수립 시 인증절차가 없으며, 이에 PX4측에 새로운 UDP 포트를 오픈하여 또 다른 GCS를 연결함으로써 다중 접속 여부에 대한 실험을 진행하였다.



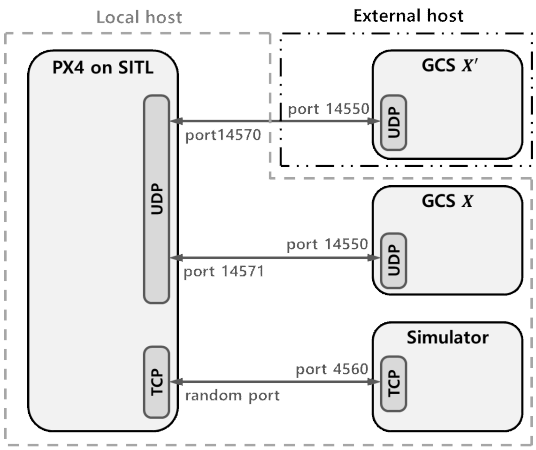
〔그림 2〕 SITL 시뮬레이션 환경

4.2.2. 다중 접속을 통한 제어권 획득 실험

본 절에서는 SITL 시뮬레이션 환경에서 로컬의 QGroundControl과 외부 호스트의 QGroundControl이 동시에 로컬 내의 PX4 시스템으로 다중 접속하여 각각의 GCS에서 드론을 제어할 수 있는지 실험한다.

우선 공식 github 사이트[20]에서 다운로드한 PX4 소스코드 중 통신과 관련된 /src/Firmware/ROMFS/PX4fmu_common/initd-posix/rcS 시스템 초기화 스크립트 파일에 외부 호스트의 GCS X'가 접속할 포트를 선언한다. 로컬에서 SITL 시뮬레이션을 실행하면 로그를 통해 PX4에 2개의 GCS가 연결될 포트가 활

성화된 것을 확인할 수 있다. 이후 로컬 호스트 및 외부 호스트에서 GCS X'를 실행시키면 동시 접속이 가능하며, 각각의 GCS에서 PX4로 명령이 전달 가능하였다. 각각의 GCS에서 내린 명령이 마치 하나의 GCS에서 명령을 내린 것처럼 동작하는 것을 로컬 시뮬레이터에서 확인할 수 있다. 또한, 다중 접속으로 인한 충돌은 발생하지 않았다. 그림. 3.은 다중 접속 시 SITL 시뮬레이션 환경을 도식화한 것이다. 그림에서 로컬 호스트의 GCS X와 PX4가 UDP로 통신하고 있으며, PX4의 새로운 UDP 포트를 통해 외부 호스트의 GCS X'이 다중 접속을 수행하고 있다.



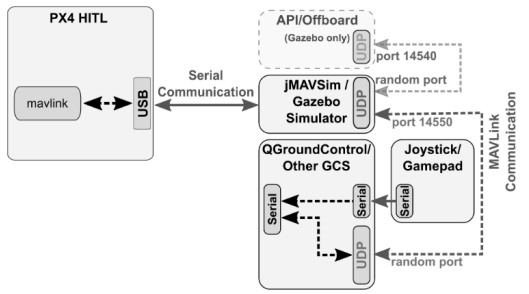
〔그림 3〕 SITL 시뮬레이션 환경에서 다중 접속을 통한 제어권 획득 시 구성

4.3. Pixhawk를 활용한 HITL 환경에서의 분석

본 절에서는 오픈소스들을 활용하여 HITL 시뮬레이션 환경 구축하고 와이파이 환경에서의 다중 접속을 통한 제어권 획득과 단일 접속을 통한 제어권 탈취 실험을 수행한다. HITL 시뮬레이터는 PX4, pixhawk2 보드, 시뮬레이터와 GCS로 구성한다.

4.3.1. 시뮬레이션 환경

그림. 4.는 HITL 시뮬레이션 환경의 연결 구조를 도식화한 것으로, jMAVSim 시뮬레이터는 컨트롤러와 UDP 통신을 하며 PX4 시스템과 컨트롤러 사이의 게이트웨이 역할을 수행한다. 즉, jMAVSim이



〔그림 4〕 HITL 시뮬레이션 환경

QGroundControl로부터 수신한 명령을 PX4 시스템으로 전달하고, PX4 시스템으로부터 수신한 응답을 다시 QGroundControl로 전달한다. 이에 jMAVSim 측에 새로운 UDP 포트를 오픈하여 또 다른 GCS를 연결함으로써 다중 접속 및 제어권 획득이 가능할 것이라 추측하였다.

4.3.2. 다중 접속을 통한 제어권 획득 실험

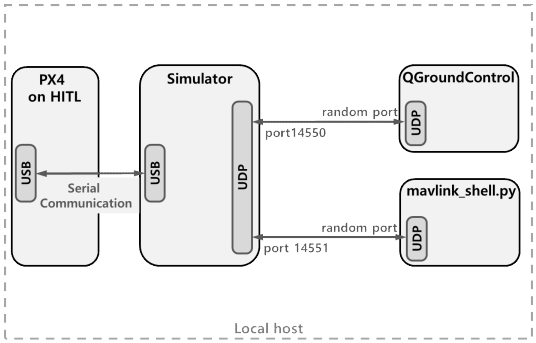
본 절에서는 HITL 시뮬레이션 환경에서 하나의 드론 하드웨어에 2개의 GCS를 다중 접속하여 각각의 GCS에서 드론을 제어할 수 있는지 실험한다. 본 연구의 실험 환경에서는 QGroundControl 프로그램을 두 개 이상 실행시킬 수 없었기 때문에 GCS 역할을 할 수 있는 mavlink_shell.py 코드를 활용하였다. mavlink_shell.py는 NuttX 운영체제 기반 시스템의 nsh 콘솔 접근 코드로, PX4에 접속하여 해당 펌웨어 및 펌웨어가 탑재된 드론을 제어할 수 있다. 해당 코드는 PX4 오픈소스 프로젝트 내에 기본으로 포함되어 있다.

우선 jMAVSim 시뮬레이터의 통신과 관련된 /src/Firmware/Tools/jMAVSim/src/me/drton/jmavsim/Simulator.java 파일을 수정하여 mavlink_shell.py가 접속할 UDP 포트를 추가한다.

mavlink_shell.py를 실행시켜 시뮬레이터에 다중 접속했을 때의 HITL 환경은 그림. 5.와 같다.

그림. 5.에서 시뮬레이터와 QGroundControl이 UDP로 통신하고 있으며, 시뮬레이터의 새로운 UDP 포트를 통해 mavlink_shell.py이 다중 접속하고 있다.

QGroundControl과 mavlink_shell.py가 하나의 펌웨어에 다중 접속하였을 때, 드론에서 발생시킨 현재 위치 또는 고도 값 등을 각각의 GCS에서 확인할 수



〔그림 5〕 HITL 시뮬레이션 환경에서 다중 접속을 통한 제어권 획득 시 구성

있으며 각각의 GCS를 통해 드론을 제어할 수 있음을 확인하였다.

4.4. 모뎀으로 통신하는 HITL 환경에서의 분석

본 절에서는 오픈소스 드론과 GCS가 RFD 900x 모뎀[22]으로 통신하는 HITL 시뮬레이션 환경에서 다중 접속을 통한 제어권 획득 및 단일 접속을 통한 제어권 탈취 실험을 수행한다.

4.4.1. 시뮬레이션 환경

본 절에서 구축한 HITL 시뮬레이션 환경의 구조는 그림. 4.와 같으며 실험에는 RFD 900x 모뎀을 사용하였다. 모뎀의 펌웨어는 RFD SiK 3.07 on RFD900X R1.3d을 사용하였으며 해당 펌웨어는 PX4에서 사용하는 MAVLink 프로토콜 통신 방식이 구현되어 있다. RFD 900x 모뎀은 ‘plug-n-play’ 방식으로 대부분의 pixhawk 시리즈와 호환이 가능하다[22].

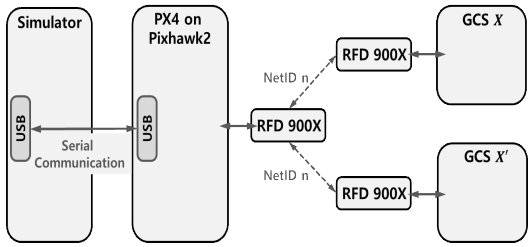
본 논문에서는 해당 모뎀의 Mavlink 프로토콜을 활성화하고 모뎀 간 원활한 통신을 위해 모든 모뎀의 Tx Power를 증가시켜 실험을 진행하였다.

4.4.2. 다중 접속을 통한 제어권 획득 실험

본 절에서는 하나의 드론에 모뎀으로 통신하는 2개의 GCS가 연결되는 다중 접속을 통한 제어권 획득 실험을 수행한다.

본 실험에서 활용하는 RFD 900x 모뎀의 펌웨어는 기본적으로 다중 접속을 허용하지 않는다. 또한 RFD 900시리즈는 16bits의 NetID를 통해 동일한 주파수를 사용하는 통신 신호 중 연결 대상을 식별하고 동일한 NetID를 가진 모뎀 사이에 통신이 수립한다. 이때 본 실험에서는 통신에 사용되는 NetID를 사전에 알고있다고 가정한다.

실험을 위해 시뮬레이터가 동작하는 컴퓨터와 정상 GCS X 및 다중 접속 GCS X'를 실행할 컴퓨터 3대에 각각의 모뎀을 연결하고 GCS X와 GCS X'를 실행시키면 그림. 6.과 같이 다중 접속이 가능하다.



〔그림 6〕 모뎀으로 통신하는 HITL 시뮬레이션 환경에서 다중 접속을 통한 제어권 획득 시 구성

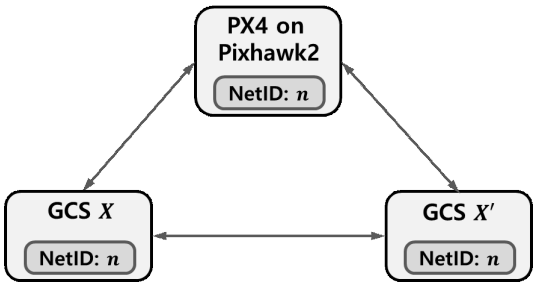
그림. 6.과 같이 다중 접속하였을 때, GCS X와 GCS X'에서 전송한 명령이 모뎀을 통해 pixhawk2에 정상적으로 전달되어 동작하는 것을 시뮬레이터에서 확인할 수 있다. 이를 통해 모뎀 및 PX4의 기본 설정을 변경하지 않아도 다중 접속을 통한 제어권 획득이 가능함을 확인하였다.

4.4.3. 접속 해제를 통한 제어권 탈취 실험

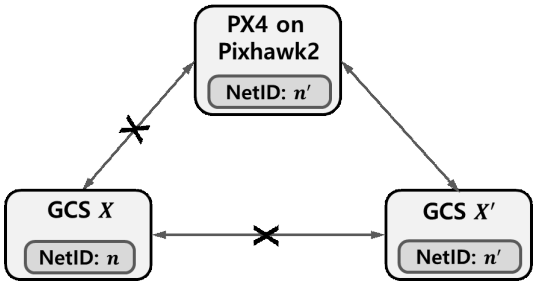
본 절에서는 2개의 GCS를 pixhawk2에 다중 접속한 후, pixhawk2의 NetID를 변경하여 정상사용자의 접속을 해제하고 제어권을 탈취하는 실험을 수행한다.

제어권 탈취를 수행하기에 앞서 그림. 7.과 같이 모든 통신 구성 요소들은 동일한 NetID를 가지며, pixhawk2와 GCS X가 정상적인 통신을 하고 GCS X'는 다중 접속을 수행하고 있다고 가정한다.

다중 접속된 상태에서 정상사용자의 GCS X의 연결을 해지시키면 GCS X'가 단독으로 드론을 제어하는 제어권 탈취를 성공할 수 있다. 이를 위해 실험에



〔그림 7〕 모뎀으로 통신하는 HITL 환경에서 다중 접속 중 연결 구조



〔그림 8〕 접속 해제를 통한 제어권 탈취 시 연결 구조

서는 RFD900Tools[22]을 이용한다. 해당 툴을 이용하면 RFD900Tools과 USB로 직접 연결된 Local 모뎀뿐만 아니라 통신 중인 원격 모뎀의 NetID까지도 변경할 수 있다.

따라서 그림. 7의 상태에서 RFD900Tools를 이용하여 GCS X'와 pixhawk2의 NetID를 변경하면 그림. 8.과 같이 정상사용자의 GCS X의 연결이 끊어지고 GCS X'만 pixhawk2와 연결되어 단독으로 드론을 제어할 수 있게 된다.

Ⅴ. 결 론

본 논문에서는 각 시뮬레이션 환경에서 제어권 획득 취약점을 찾음으로써 다중 접속을 성공하였다. 특히, 실물 pixhawk2와 RFD900x 모뎀을 이용하여 실제와 유사한 환경을 구축하였을 때, 정상사용자와 드론의 연결을 해제하여 제어권 탈취가 가능했다. 통신 대상의 NetID를 사전에 알아야 한다는 한계점이 있지만, 펌웨어의 수정 없이 대상 드론의 제어권을 완전히 탈취할 수 있고 MAVLink 프로토콜을 사용하는 모든 드론에 대해 적용이 가능하다.

본 논문에서 제시한 방법들을 통해 오픈소스 드론 펌웨어의 취약점을 발견할 수 있었지만, 시뮬레이션 환경에서 동작 중인 오픈소스 드론 펌웨어 내부를 분석하지 못하여 문제가 발생한 코드 또는 기능을 구체화하지 못하는 한계점을 가지고 있다. 본 논문에서 제안한 오픈소스 드론 분석 연구의 한계점은 원격으로 동적 분석이 가능한 디버거와 시뮬레이터를 결합한 분석 환경을 구축함으로써 개선 가능할 것으로 예상된다.

참 고 문 헌

[1] Yeong-cheol Choe, Hyo-seong An, “Drones Current and Technology Development Trends and Prospects.“ The world of Electrical Engineers, 64(12), pp. 20-25, Dec. 2015

[2] 강태임, 최창빈, 김가연, 이태현, 이경호, 조세나, 장영수, ”공객 오픈소스의 보안 취약성에 대한 학부생의 인식 조사 연구“, 한국정보처리학회, pp. 131-132, 2018

[3] SITL simulation environment, https://docs.px4.io/master/en/simulation/

[4] HITL simulation environment, https://docs.px4.io/master/en/development/development.html

[5] Young-min Kwon, Jae-min Yu, Byeong-moon Cho, Yong-soon Eun, Kyung-joon Park, “Empirical Analysis of MAVLink Protocol Vulnerability for Attacking Unmanned Aerial Vehicles.” IEEE Access, vol. 6, pp. 43203-43212, 2018

[6] Mavlink, https://mavlink.io/en/

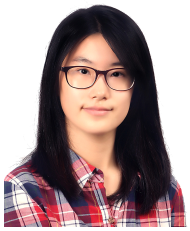
[7] N. M. Rodday, R. d. O. Schmidt and A. Pras, "Exploring security vulnerabilities of unmanned aerial vehicles," NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, pp. 993-994, 2016

[8] S. P. Arteaga, L. A. M. Hernanez, G. S. Perez, A. L. S. Orozco and L. J. G. Villalba, “Analysis of the GPS Spoofing Vulnerability in the Drone 3DR Solo.” IEEE Access, vol. 7, pp. 51782-51789, 2019

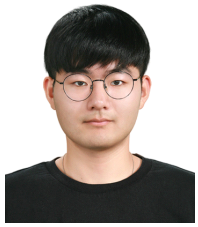
[9] 이우진, 서경덕, 채병민, “오픈소스 활용 드론에 대한 보안 위협과 Telemetry Hijacking을 이용한 군

용 드론 공격 시나리오 연구", 한국융합보안학회, 20(4). pp. 103-112, 2020
[10] Drone Code, https://www.dronecode.org/
[11] P. E. Ross, "Open-source drones for fun and profit." IEEE spectrum, 51(3), pp. 54-59, 2014
[12] 배명진, 김성일, "항공 드론 지상 제어 시스템 기술 동향", 한국멀티미디어학회, 20(1_2), pp. 22-28, 2016
[13] Pixhawk2, https://docs.PX4.io/master/en/flight_controller/pixhawk-2.html
[14] Mission planner, https://ardupilot.org/planner/
[15] QGroundControl, http://qgroundcontrol.com/
[16] APM Planner2, https://ardupilot.org/planner2/
[17] Gazebo, https://gazebosim.org/
[18] S. Moon, Y. Choi, and H. Gong, "Development of HLIS System based on Pixhawk for swarm flight." Proceeding of the 2015 KSAS Fall Conference, pp. 1048-1051, 2015
[19] 문성태, 최연주, 김도윤, 공현철, "픽스호크 기반의 소형 드론 시험 환경 구축", 한국항공우주학회 학술발표회 초록집, pp. 728-730, Apr, 2016
[20] PX4 github, https://github.com/PX4/PX4-Autopilot.git
[21] RFD900x, https://docs.PX4.io/master/en/telemetry/rfd900_telemetry.html
[22] RFD900Tools, https://files.rfdesign.com.au/tools/

〈저 자 소 개〉



김 숙 영 (Sukyoung Kim)
정회원
2018년 2월 : 상명대학교 컴퓨터과학과 학사
2021년 2월 : 고려대학교 정보보호대학원 정보보호학과 석사
<관심분야> 사물인터넷, 정보보호



이 대 현 (Daehyeon Lee)
학생회원
2019년 2월 : 고려대학교 전자 및 정보공학과 학사
2021년 2월 : 고려대학교 정보보호대학원 정보보호학과 석사
2021년 3월~현재 : 고려대학교 정보보호대학원 정보보호학과 박사과정
<관심분야> 전자공학, 인공지능, 빅테이터, 사물인터넷, 정보보호

서 형 민 (Hyeongmin Seo)
정회원
2013년 2월 : 광운대학교 컴퓨터소프트웨어학과 학사
2016년 2월 : 고려대학교 정보보호대학원 정보보호학과 석사
2016년 3월~현재 : ETRI 부설연구소 연구원
<관심분야> 디지털 포렌식, 취약점 분석, 정보보호

문 종 섭 (Jongsub Moon)
종신회원
1981년 2월 : 서울대학교 계산통계학과 학사
1983년 2월 : 서울대학교 계산통계학과 석사
1991년 2월 : Illinois Institute of Technology 전산학과 박사
1993년 3월~현재 : 고려대학교 전자 및 정보공학부 교수
2001년 2월~현재 : 고려대학교 정보보호대학원 겸임교수
<관심분야> 정보보호, 운영체제, 침입탐지