Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.1, Feb. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.1.31

체계적인 IoT 기기의 펌웨어 보안 분석 방법에 관한 연구*

김 예 준,[1†] 김 정 현,[1] 김 승 주[2‡]
[1,2]고려대학교 정보보호대학원 (대학원생, 교수)

A Study on Systematic Firmware Security Analysis Method for IoT Devices*

Yejun Kim,[1†] Jeonghyeon Gim,[1] Seungjoo Kim[2‡]
[1,2]CIST(Center for Information Security and Technologies)
School of Cybersecurity, Korea University (Graduate student, Professor)

요 약

IoT 기기는 네트워크 통신이 가능한 임베디드 기기를 의미한다. IoT 기기는 금융, 개인, 산업, 공공, 군 등과 같이 우리 주변의 다양한 분야에서 많이 사용되고 있기 때문에 공격이 발생할 경우 개인정보 유출과 같은 다양한 피해가 발생할 수 있다. IoT에 대한 취약점 분석은 IoT 기기와 상호작용 하는 스마트폰의 애플리케이션, 웹 사이트와 같은 소프트웨어 인터페이스 분석뿐만 아니라, IoT 기기의 주요 구성요소인 펌웨어에 대해서도 필수적으로 수행되어야 한다. 하지만 문제는 펌웨어의 추출 및 분석이 생각보다 쉽지 않으며, 보안팀 내 분석하는 사람의 전문성에 따라 같은 대상을 분석하더라도 결과물의 수준이 다를 수 있어 일정한 수준의 품질 관리가 쉽지 않다는 것이다. 따라서 본 논문에서 우리는 IoT 기기의 펌웨어에 대한 취약점 분석 프로세스를 정립하고 단계별로 사용 가능한 도구를 제시함으로써, IoT 보안성 분석에 있어 로드맵을 제시하고 일정한 수준의 품질 관리가 가능하였다. 우리는 다양한 상용 제조사들이 생산한 IoT 기기의 펌웨어 획득부터 분석까지의 과정을 제안하였으며, 이를 다양한 제조사의 드론 분석에 직접 적용해 봄으로써 그 타당성을 입증하였다.

ABSTRACT

IoT devices refer to embedded devices that can communicate with networks. Since there are various types of IoT devices and they are widely used around us, in the event of an attack, damages such as personal information leakage can occur depending on the type of device. While the security team analyzes IoT devices, they should target firmware as well as software interfaces since IoT devices are operated by both of them. However, the problem is that it is not easy to extract and analyze firmware and that it is not easy to manage product quality at a certain level even if the same target is analyzed according to the analyst's expertise within the security team. Therefore, in this paper, we intend to establish a vulnerability analysis process for the firmware of IoT devices and present available tools for each step. Besides, we organized the process from firmware acquisition to analysis of IoT devices produced by various commercial manufacturers, and we wanted to prove their validity by applying it directly to drone analysis by various manufacturers.

Keywords: IoT, Firmware, Firmware Analysis, Vulnerability Analysis

Received(09. 10. 2020), Modified(10. 26. 2020), Accepted(10. 27. 2020)

* 본 연구는 고려대 암호기술 특화연구센터(UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

† 주저자, v3locy@korea.ac.kr
‡ 교신저자, skim71@korea.ac.kr(Corresponding author)

I. 서 론

IoT 기기는 네트워크 통신이 가능한 드론, 라우터, IP 카메라, AI 스피커, 스마트 TV 등의 다양한 임베디드 기기로, 이는 네트워크로 연결되어 실시간으로 제어할 수 있다는 장점을 가진다. 네트워크에 연결된 IoT 기기의 수는 지속적으로 증가하여 2020년에는 2019년 대비 약 44억 개가 증가한 약 310억 개의 IoT 기기가 네트워크에 연결될 것으로 예상된다[1]. 또한 IoT 기기는 금융, 개인, 산업, 공공, 군과 같은 많은 분야에서 사용되고 있다. IoT 기기는 이처럼 우리 주변에서 많이 사용되고 있기 때문에 이에 대한 공격이 발생했을 경우 기기의 종류나 활용 분야에 따라 개인정보 유출, 금전적 손실 등 다양한 종류의 피해가 발생할 수 있다[2].

따라서 제조사에서는 IoT 기기에 대한 일정 수준 이상의 보안성을 확보할 필요성이 있다. 또한 보안팀은 IoT 기기의 보안성 증대를 위해 외부 침입자(공격자, 제 3자)의 입장에서 취약점을 분석해 문제점을 밝혀냄으로써, 조기에 적절한 보안 패치가 이루어질 수 있도록 해야 한다. 이때, IoT 기기는 펌웨어에 의해 작동하므로 IoT 기기에 대한 취약점 분석 과정에서는 IoT 기기와 상호작용 하는 스마트폰의 애플리케이션, 웹 사이트와 같은 소프트웨어 인터페이스뿐만 아니라, IoT 기기의 주요 구성요소인 펌웨어에 대한 분석도 필수적으로 수행되어야 한다. 하지만 펌웨어는 일반적인 소프트웨어와 다른 구조를 가지고 있으며 하드웨어에 의존적인 특징을 가지고 있기 때문에 펌웨어를 기기로부터 추출하고 분석하는 것이 생각보다 쉽지 않다. 또한 보안팀 내 분석하는 사람의 전문성에 따라 같은 대상을 분석하더라도 결과물의 수준이 다를 수 있어 IoT 기기에 대한 일정한 수준의 품질 관리가 쉽지 않다[3].

따라서 본 논문에서 우리는 IoT 기기의 펌웨어에 대한 취약점 분석 프로세스를 정립하고 각 단계에서 사용 가능한 도구를 제시함으로써, IoT 보안성 분석에 있어 로드맵을 제시하고 일정한 수준의 품질 관리가 가능토록 하고자 한다. 우리는 다양한 상용 제조사들이 생산한 IoT 기기의 펌웨어 획득부터 분석까지의 과정을 체계화시켰으며, 이를 다양한 제조사의 드론 분석에 직접 적용해 봄으로써 그 타당성을 입증한다. 본 논문의 결과는 향후 다양한 기기의 펌웨어 취약점 분석에 있어 적은 시간 내에 최선의 결과를 도출하는데 활용될 수 있을 것으로 생각된다.

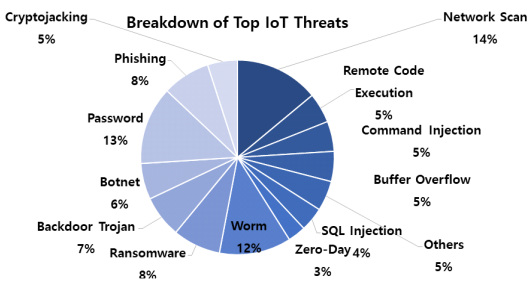


Fig. 1. Threats to IoT devices[2]

본 논문은 총 6장으로 구성되며 2장에서는 펌웨어 분석과 관련된 연구에 대해 서술하고, 3장에서는 펌웨어 획득 및 분석 프로세스를 제안한다. 4장에서는 제안한 프로세스에 대한 검증을 수행하고 5장에서 제조사를 위한 권장사항을 제시한다. 마지막으로 6장에서 본 논문에 대한 결론을 맺는다.

II. 관련 연구

본 장에서는 펌웨어 분석과 관련한 최근 연구 동향을 살펴본다. 우리는 IoT 기기를 분석하기 위한 프로세스를 도출하기 위하여 총 309건의 관련 문헌을 조사하였다. 그 중, IoT 기기를 분석하기 위한 상세한 방법론을 제안한 문헌은 총 2건으로 IoT 기기에 대한 리버싱 방법과 사용되는 펌웨어를 추출하고 분석하는 방법에 대하여 설명하고 있다. 하지만 각 분석 방법이 순서대로 표현되지 않고 독립적으로 기술되어 있기 때문에 취약점 분석 수행 시 로드맵으로써 참고하기 어렵다. 그렇기에 우리는 이러한 문제를 개선하기 위해 기존 연구를 참고하고 실험을 진행함으로써, 보안팀이 취약점 분석 시 쉽게 참고하고 용이하게 활용할 수 있는 프로세스를 도출하는 것을 목표로 한다.

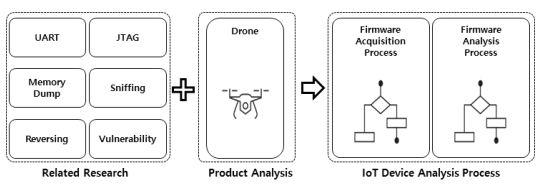


Fig. 2. Derivation of analysis methodology

2.1 연구 동향 파악을 위한 논문 수집 방법론

우리는 본 논문에서 다루고자 하는 주제와 관련된

연구 동향을 체계적으로 수집·분류하기 위하여 Chitu와 Kira의 "A Guide to Conducting a Systematic Literature Review of Information Systems Research"에서 제시한 방법론을 따랐다[4]. 해당 방법론에서는 체계적인 문헌 검토(SLR, Systematic Literature Review)를 수행하기 위한 8단계의 절차를 제공하며, 이는 (1)문헌 검토의 목적 정의 (2)프로토콜 정의 및 교육, (3)문헌 검색, (4)문헌 선별, (5)품질 평가, (6)분류 및 데이터 추출, (7)연구 내용 결합, (8)리뷰 작성의 단계로 구성되어 있다. 본 절에서는 해당 방법론에 따른 문헌 검토 수행 과정을 설명하며 (7), (8) 단계의 수행 과정은 2.2, 2.3, 2.4에서 다룬다.

1) 문헌 검토의 목적 정의: 산업 공정이나 전문 기술을 위한 것이 아닌 일반적인 사용자를 대상으로 개발된 IoT 기기에 대한 펌웨어의 취약점과 관련된 연구의 흐름을 분석한다.

2) 프로토콜 정의 및 교육: 문헌 검토과정에서 우리는 번호, 카테고리, 제목, 내용 요약, 인용 수, 발행 연도, 저자, 발행지 등의 정보를 효과적으로 입력할 수 있도록 표의 형식으로 구성하여 문헌의 내용을 문서화 할 수 있도록 하였다.

3) 문헌 검색: 우리는 전 단계에 이어, 문헌 검색을 실시하였다. 관련 문헌을 수집하기 위해 키워드를 기반으로 탐색하였으며 검색을 위한 기준은 [Table 1]과 같다.

4, 5) 문헌 선별 및 품질 평가: 문헌 검색 과정은 단순히 키워드에 의해 이루어졌으므로, 수집된 모든 문헌으로부터 본 연구와 관련 있는 데이터를 추출하는 것은 불가능하다. 따라서, 앞서 소개한 방법대로 문헌의 내용을 문서화하고, 내용을 정리하여 본 연구와 관련 있는 문헌을 선별하였다. 또한 최종 선별 단계를 통해 제외되지 않은 문헌에 대하여 품질 평가를 수행하였다. 품질 평가는 본 연구와 직접적으로 관련이 있다고 판단되어 선별된 문헌 중에서 우리가 분류할 카테고리 내에 포함될 수 있는 문헌을 최종적으로 선별하는 과정이다.

Table 1. Paper search criteria

Category	Condition
Databases	BlackHat, IEEE[5], NDSS[6], Usenix[7], ACM[8], Elsevier[9], Springer[10]
Search keyword	"IoT", "Firmware", "IoT" +"Firmware", "IoT" + "Firmware" + "Analysis", "IoT" + "Firmware" + "Hacking", "IoT" + "Firmware" + "Attack", "IoT" + "Firmware" + "Vulnerability"
Publication period	11 Years (2010 ~ 2020)
Review period	2020. 02. ~ 2020. 08.

- 최초 수집: 309개
- 1차 선별: 279개
- 2차 선별: 143개
- 최종 선별: 52개

6) 분류 및 데이터 추출: 수집한 문헌의 카테고리를 우리가 설정한 기준에 맞게 분류하고, 각 문헌에서 필요한 데이터를 추출한다. 문헌의 카테고리는 [Table 2]와 같다.

Table 2. Paper classification category

Category	Explanation
Analysis methodology	Literature including vulnerability analysis methodology for IoT firmware [11, 12]
Discovering new attacks (vulnerabilities) and countermeasures	Documents including new vulnerability analysis results and countermeasures for IoT firmware [13-40]
Tool development	A document proposing a tool that can perform analysis and attacks on IoT firmware [41-62]

2.2 분석 방법론

본 절에서는 기존에 수행된 IoT 기기의 펌웨어에 대한 취약점 분석 방법론에 관한 연구 동향에 대해 서술한다.

Jonas Zaddach 등은 2013년 "Embedded Devices Security and Firmware Reverse Engineering"를 통해 임베디드 장치의 구조와 펌웨어에 대한 분석 방법과 분석에 사용되는 도구에 대하여 설명하였다[11]. 또한 Omer Shwartz 등은

2018년 “Reverse Engineering IoT Devices: Effective Techniques and Methods”를 통해 IP 카메라, 베이비 모니터 등을 대상으로 IoT 기기의 디버깅 포트를 이용한 분석을 시도하였으며, 펌웨어 내부의 암호화 방식에 대한 분석 및 취약점 분석 방법을 설명하였다[12].

2.3 신규 공격(취약점) 발견 및 대응 방안

본 절에서는 기존에 수행된 IoT 펌웨어에 대한 분석중 신규 취약점을 찾거나 대응 방안을 제시한 연구에 관해 서술한다.

Andrei Costin 등은 2014년 “A Large-Scale Analysis of the Security of Embedded Firmwares”를 통해 32,000개의 펌웨어 이미지에 대해 취약점 분석을 수행한 결과를 설명하였다[18]. 이후 Yifei Xu 등은 2018년 “A Search-based Firmware Code Analysis Method for IoT Devices”에서 공개된 정보를 기반으로 펌웨어 코드 분석을 수행하였다[19]. 그리고 Grant Hernandez 등은 2019년 “Toward Automated Firmware Analysis in the IoT Era”를 통해 펌웨어에 대해 수행한 다양한 연구를 대상으로 사례 분석을 진행하고 향후 개선 방향을 제시하였다[13].

2.4 도구 개발

본 절에서는 기존에 수행된 IoT 펌웨어에 대한 분석 및 공격을 수행할 수 있는 도구를 제안한 연구에 관해 서술한다.

Jiongyi Chen 등은 2018년 “IOTFUZZER: Discovering Memory Corruptions in IoT Through App-based Fuzzing”를 통해 IoT 기기와 연결되는 모바일 애플리케이션에 대한 퍼징을 수행한 과정 및 결과를 설명하였다[41]. 이후 Yaowen Zheng 등은 2019년 “Firm-AFL: High-Throughput Greybox Fuzzing of IoT Firmware via Augmented Process Emulation”을 통해 에뮬레이터에서 펌웨어를 에뮬레이션 할 수 있도록 호환성 문제를 해결한 AFL 기반의 퍼저인 FIRM-AFL을 제안하였다[43]. 또한 Taegyu Kim 등은 같은 해에 “RVFuzzer: Finding Input Validation Bugs in Robotic Vehicles through Control-Guided Testing”를 통해 Robotic Vehicles(RVs)를 대상으로 시뮬레이션이 가능한 별도의 장치를 통해 실제 사용 환경과 흡사한 테스트베드를 제작하여 퍼징을 수행한 연구에 대해 설명하였다[44]. 그리고 Zhijie Gui 등은 2020년 “FIRMCORN: Vulnerability-Oriented Fuzzing of IoT Firmware via Optimized Virtual Execution”을 통해 가상 실행의 초기 환경과 실행 프로세스를 최적화하여 속도, 정확성, 안정성을 개선한 FIRMCORN을 제안하였다[45].

이러한 퍼저 외에도 펌웨어 자동분석 도구 개발에 대한 연구가 활발히 진행되고 있다. Jonas Zaddach 등은 2014년 “Avatar: A Framework to Support Dynamic Security Analysis of Embedded Systems Firmwares”를 통해 실제 하드웨어와 함께 에뮬레이터의 실행을 조정하여 임베디드 장치에 대한 동적 분석을 가능하게 하는 도구를 제시하였다[60]. 또한 Daming D. Chen 등은 2016년 ″Towards Automated Dynamic Analysis for Linux-based Embedded Firmware″를 통해 네트워크 연결 COTS 장치에서의 Linux 기반 펌웨어를 타겟으로 하는 자동 동적 분석 시스템인 FIRMADYNE을 제시하였다[62]. 그리고 Grant Hernandez 등은 2017년 ″FirmUSB: Vetting USB Device Firmware using Domain Informed Symbolic Execution“을 통해 USB 관련 펌웨어에 대한 분석 도구인 FirmUSB를 제안하였다[58]. 마지막으로, Vasaka Visoottiviseth 등은 2018년 ″Firmaster: Analysis Tool for Home Router Firmware“를 통해 라우터에 대한 펌웨어에 대해 동적 및 정적 분석을 수행할 수 있도록 하는 도구인 Firmaster를 제시하였다[57].

III. 펌웨어 분석 프로세스

앞서 설명한 바와 같이 기존의 IoT 펌웨어 분석과 관련된 논문에는 각 분석 방법이 독립적으로 기술되어 있기 때문에 보안팀이 취약점 분석을 수행할 때 로드맵으로써 참고하기 어렵다는 문제점이 존재한다. 따라서 본 장에서는 이러한 문제를 개선하기 위해 IoT 기기의 펌웨어를 획득하고 이를 분석할 수 있는 분석 프로세스를 제시하여 보안팀이 취약점 분석을 수행할 시 용이하게 활용할 수 있도록 하고자 한다.

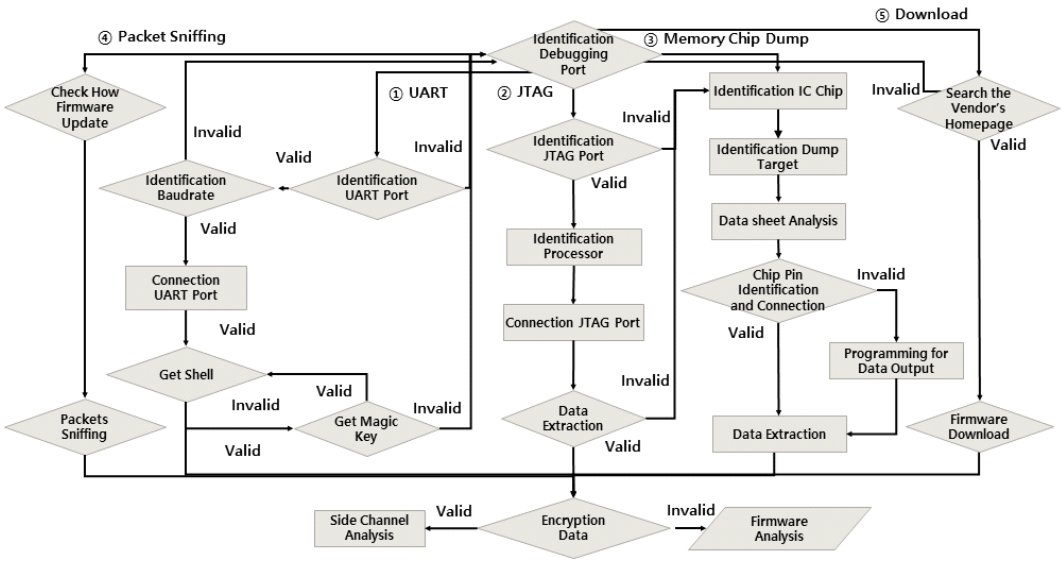


Fig. 3. Firmware acquisition process

3.1 펌웨어 획득 프로세스

보안팀이 분석 대상 기기에서 펌웨어를 획득하기 위해 필요한 프로세스는 〔Fig. 3〕과 같다.

3.1.1 펌웨어 다운로드 및 패킷 스니핑

보안팀이 IoT 기기의 펌웨어를 획득할 수 있는 가장 쉬운 방법은 제조사의 공식 홈페이지에 공개된 펌웨어를 다운로드하거나 업데이트 패킷을 스니핑하는 것이다. 해당 방법은 별도의 분석기술이 없어도 수행될 수 있기 때문에 가장 먼저 시도해야 하는 펌웨어 획득 방법이다.

1) 펌웨어 다운로드: IoT 기기의 제조사는 업데이트를 위해 기기에서 사용되는 펌웨어를 공식 홈페이지에 공개하여 사용자가 다운로드할 수 있도록 지원한다. 이 방법은 가장 쉽게 펌웨어를 획득하는 방법으로서, 드물지만 오픈소스를 이용한 펌웨어의 소스코드까지 획득할 수 있다. 하지만 제조사가 공개한 펌웨어를 직접 다운로드할 수 있다 하더라도, 펌웨어가 암호화돼있어 보안팀이 별도의 복호화를 수행해야 하는 경우도 있다.

2) 패킷 스니핑: 보안팀이 제조사를 통해 펌웨어 획득이 불가능할 경우 IoT 기기 내의 펌웨어 업데이트 기능을 이용해 펌웨어를 획득할 수 있다. IoT 기기에서는 별도의 기능을 통해 펌웨어를 업데이트하거나 기기를 모바일 애플리케이션에 연결했을 때 펌웨어의 버전을 확인하여 업데이트가 수행된다. 이때, 패킷 캡쳐 도구인 WireShark나 Fiddler 등을 이용해 펌웨어를 업데이트할 때 전송되는 패킷을 스니핑하여 펌웨어를 획득할 수 있다〔63〕.

이처럼 공식 제조사 홈페이지를 통해 펌웨어를 다운로드하거나 업데이트 패킷을 스니핑해 펌웨어를 획득하기 위한 세부 단계와 이때 사용되는 도구들은 〔Table.3〕과 같다.

Table 3. Download and sniffing steps

Detailed steps	Tools	Purpose
Checking the firmware update method	-	Confirm the firmware update method of the analysis target
Sniffing firmware update packets	Packet capture tool	Sniffing firmware update packets through packet capture and analysis tools
Browsing the manufacturer's homepage	-	Download the firmware published on the manufacturer's website
Searching for research material	-	Download the published firmware by browsing the research data on the analysis target

3.1.2 하드웨어 디버깅 포트 및 IC 칩 식별

보안팀은 디버깅 포트를 이용하여 IoT 기기를 분석할 수 있다. 이를 위해서는 별도의 장비를 사용해야 하며, 이 외에 IC 칩의 메모리를 덤프하여 펌웨어를 획득할 수도 있다[64].

1) 디버깅 포트 식별: 디버깅 포트는 IoT 기기에서 데이터를 전송하거나 개발자가 내부 펌웨어를 디버깅하기 위해 사용되며 UART(Universal Asynchronous Receiver Transmitter)[65], JTAG(Joint Test Action Group)[66, 67] 등이 존재한다. 보안팀은 디버깅 포트를 통해서 IoT 기기에서 사용하는 펌웨어의 정보를 식별하거나 펌웨어 추출, 명령어 쉘 획득, 동적 분석 등 다양한 기능을 사용할 수 있다.

UART 포트는 IoT 기기의 종류나 사용되는 프로세서의 종류와 관계없이 Vcc, Ground, Rx, Tx 4개의 핀으로 구성된다. 하지만 JTAG 포트는 사용되는 프로세서의 종류에 따라 핀의 개수와 구성요소가 다르다. 디버깅 포트는 [Fig. 4]와 같이 육안으로 쉽게 식별이 가능한 경우도 존재하지만 [Fig. 5]처럼 포트 식별을 방해하기 위해 숨겨져 있는 경우도 있으며, 이때는 숨겨진 포트를 찾기 위해 Jtagulator와 같은 별도의 분석 장비를 사용해야 한다.

JTAG 포트는 IoT 기기의 프로세서 종류에 따라 핀의 개수, 역할, 위치가 달라지며 이 때문에 사용할 수 있는 분석 장비의 종류가 달라진다. JTAG 포트는 매우 다양하게 구성되어 있기 때문에, 각 핀의 위치와 역할을 식별하기 위해서는 Jtagulator라는 도구와 의심되는 모든 핀들을 연결해 봄으로써 각 핀의 역할을 식별해야 한다.

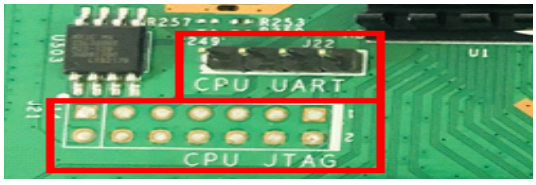


Fig. 4. Debugging port for IoT devices

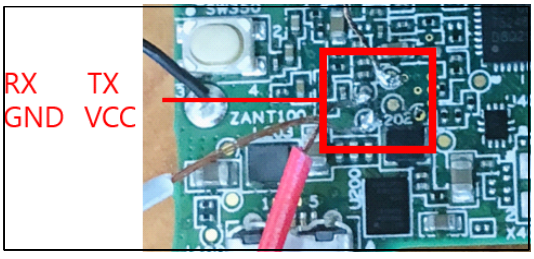


Fig. 5. Hidden UART port

2) IC 칩 식별: IoT 기기는 내부에서 펌웨어를 저장하고 실행하기 위해 SPI(Serial Parallel Interface) Flash Memory와 EEPROM 장치를 통해 직접 명령어를 로드한다. 해당 장치들은 일반적으로 25XX/26XX 시리즈의 SPI Flash Memory나 24XX 시리즈의 EEPROM 모듈과 결합되는 NAND Flash 등으로 구성된다[12]. 보안팀은 이러한 IC 칩을 식별하기 위해서 각인되어있는 모델명을 확인해야 한다.

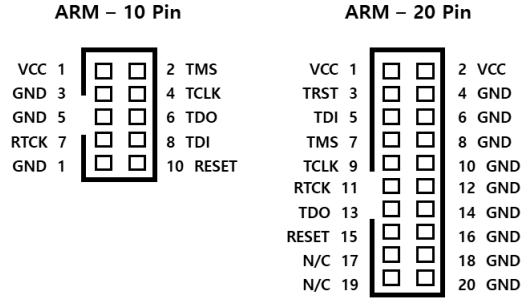


Fig. 6. Example of JTAG port type (ARM[68])

3.1.3 UART 포트를 이용한 분석 프로세스

UART 포트를 이용한 분석 프로세스에서는 UART 포트를 통해 디버그 메시지나 부팅 메시지 등을 획득하여 취약점 분석에 필요한 각종 정보를 획득할 수 있으며, 부트로더나 커맨드 쉘을 통해 펌웨어를 획득하거나 동적 분석을 수행할 수 있다.

1) 전송 속도 식별: 보안팀은 UART 포트를 식별한 뒤, 데이터 전송 속도를 식별해야 한다. 전송 속도를 식별하지 않고 분석 장비와 연결하는 경우 메시지 입출력이 정상적으로 이루어지지 않는다. IoT 기기에서 사용되는 전송 속도는 주로 9600, 57000, 115200 bit가 있으며 Jtagulator나 Logic Analyzer를 사용하여 정확한 전송 속도를 식별할 수 있다.

2) UART 포트 연결: 보안팀은 UART 포트를 USB-ISP와 같은 장비와 연결하여 Putty, Xshell 등의 터미널 프로그램을 통해 시리얼 통신을 수행할

Fig. 7. Identify baudrate using jtagulator

수 있다.

3) 쉘 권한 획득: 보안팀이 UART 포트를 사용하여 시리얼 통신에 성공할 경우 관리자 쉘, U-Boot 쉘 등을 획득하여 IoT 기기의 시스템에 접근할 수 있다. 하지만 분석 대상에 따라 부팅 메시지와 같이 일부 정보를 읽을 수만 있고 데이터 입력은 제한될 수 있다. 이때 보안팀은 내부 소프트웨어 분석을 통해 Magic Key를 추출하거나 특정한 조건을 만족함으로써 펌웨어 내부의 디버그 모드로 연결하거나 쉘을 획득할 수 있다. UART 포트를 이용한 상세 분석 프로세스와 사용되는 도구는 〔Table.4〕와 같다.

Table 4. UART analysis steps

Detailed steps	Tools	Purpose
Identifying the UART port	Jtagulator	Identifying the transmission rate by connecting a Jtagulator or Logic Analyzer to the identified UART port.
Identifying the Baudrate	Jtagulator, Logic Analyzer	Connect the Jtagulator or Logic Analyzer to the identified UART port to identify the transfer rate
Connecting the UART port	Jumper cable, UART to USB equipment	Connecting the identified UART port to a PC using equipment such as USB-ISP
	Terminal program	Displaying data acquired by connecting analysis equipment with a terminal program to the user screen
Obtaining Magic Key	Disassembler	Identifying the MagicKey required for shell connection Through an analysis of the firmware released
Obtaining shell privileges	Terminal program	Obtaining shell privileges through UART connection and Magic Key input

3.1.4 JTAG 포트를 이용한 분석 프로세스

JTAG 포트를 이용한 분석 프로세스에서는 JTAG 연결을 통해 IoT 기기에 대한 동적 디버깅, 펌웨어 수정, 획득, CPU 모니터링 등 하드웨어를 구성하는 모든 기능을 제어할 수 있다.

1) 프로세서 식별: 기기에서 사용되는 프로세서의 종류에 따라서 JTAG에 연결할 수 있는 분석 장비가 다르다. 그렇기 때문에 보안팀은 부팅 메시지, 데이터 시트, 제조사가 제공하는 기기 정보를 활용하여 장치에서 사용하는 프로세서를 식별해야 한다.

2) JTAG 포트 연결: 보안팀은 기기에서 사용되는 프로세서를 확인한 후 JTAG 포트와 분석 장비를 연결함으로써 분석 장비와 호환이 되는 소프트웨어를 활용하여 디버깅을 수행할 수 있다.

3) 데이터 추출 및 분석: 보안팀은 JTAG 포트를 이용한 디버깅으로 IoT 기기의 커널 분석과 동적 분석을 통해 사용되는 메모리의 변화를 확인할 수 있

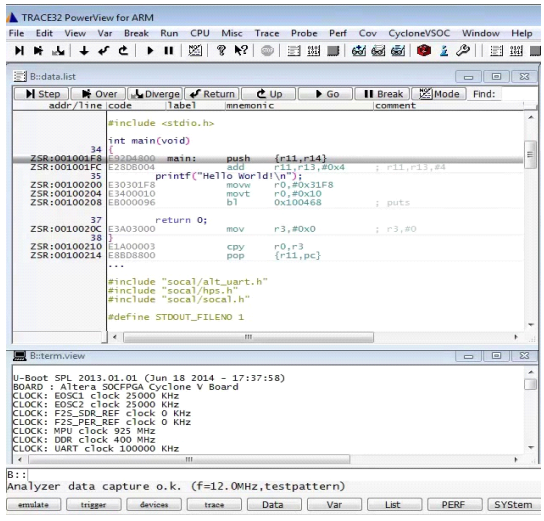


Fig. 8. Debugging with JTAG ports

다. 보안팀은 TRACE32와 같은 디버깅 장비와 소프트웨어를 사용하여 동적 디버깅을 수행할 수 있는데, 이때 ARM, MIPS[69]와 같이 프로세서마다 사용되는 어셈블리 명령어가 다르다. 그렇기 때문에 JTAG 포트를 이용한 분석 방법은 분석 대상에 대한 이해도와 디버깅 지식이 필요하다.

이처럼 JTAG 포트를 이용한 분석 방법은 기본적으로 디버깅 수행 경험과 어셈블리어에 대한 지식이 필요하다. 하지만 JTAG 포트를 이용한 분석은 펌웨어 획득은 물론 동적, 정적 분석이 모두 가능하여 폭넓은 분석 범위를 제공한다. JTAG 포트를 이용한 상세 분석 단계와 사용되는 도구는 [Table.5]와 같다.

Table 5. JTAG analysis steps

Detailed steps	Tools	Purpose
Identifying the JTAG port	Jtagulator	Identifying the JTAG port by connecting the pins of the board to the Jtagulator.
Identifying the processor	-	Identifying the processor used in the device by collecting information about the subject of analysis
Connecting to the JTAG port	Jumper Cable	Connecting to the identified JTAG port using jumper cables
Connecting debugging equipment	JTAG analysis equipment such as Trace32 and J-Link	Connecting the JTAG analysis equipment (Trace32, J-Link, etc.) suited the identified architecture to the instrument
Extracting data	Software, terminals for the analytical equipment used	Analyzing the device and extracting data through debugging using the connected equipment and suitable software

3.1.5 IC 칩 메모리 덤프

IC 칩 메모리 덤프를 통해 펌웨어를 획득하기 위해서는 IoT 기기의 하드웨어 구성요소인 Flash Memory나 EEPROM을 분석한 뒤, 전용 분석 장비를 사용해야 한다.

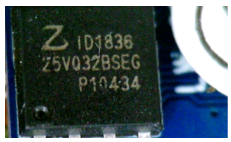
1) 덤프 대상 식별: 보안팀은 UART와 JTAG과 같은 디버깅 포트에 접근할 수 없을 경우 IoT 기기 내부 보드의 EEPROM이나 Flash Memory를 이용하여 펌웨어를 추출할 수 있다. 보안팀은 펌웨어 추출을 위해서 IoT 기기의 보드에 존재하는 IC 칩 중 EEPROM, Flash Memory의 모델을 식별하여 덤프 대상 칩을 선정해야 한다.

2) 데이터 시트 분석: 보안팀은 덤프 대상을 선정한 뒤, 제조사 홈페이지를 통해 데이터 시트를 확보할 수 있다. 이후 데이터 시트를 분석하여 IC 칩에 대한 성능과 종류, 각 핀의 역할, 사용되는 명령어, Data Flow Timing 등 여러 가지 정보를 획득할 수 있다. 또한 [Fig. 9]와 같이 각인된 모델명의 각 숫자, 문자가 의미하는 정보를 알 수 있는 경우가 있어 보안팀은 하나의 데이터 시트를 획득하여 같은 제조사의 다른 칩에 대한 정보도 유추할 수 있다.

3) 칩 핀 식별 및 장비 연결: 보안팀은 데이터 시트 분석을 통해 덤프 대상의 각 핀의 정보를 획득한 뒤 ROM Writer를 연결하여 펌웨어를 추출할 수 있다. ROM Writer는 각 제조사, IC 칩의 종류를 선택하는 것만으로도 내부 펌웨어를 추출, 수정, 삭제할 수 있는 도구이다.

4) 데이터 출력을 위한 프로그래밍: 보안팀은 데이터 시트를 분석하여 IC 칩에 대한 정보를 이용하여 Arduino나 Raspberry pi를 통해 프로그래밍을 수행하여 펌웨어를 추출할 수 있다. 하지만 데이터 시트를 분석하기 위해서는 하드웨어 및 전자회로에 대한 지식이 필요하며 데이터 시트를 획득하지 못한 경우에는 해당 프로세스를 수행할 수 없다.

온 칩 상태에서는 데이터 전송 과정에서 혼선이 발생하여 정확한 추출이 어려운 경우가 존재한다. 이때 보안팀은 열풍기, 납땜 키트를 사용하여 보드에서 덤프 대상을 칩 오프 한 뒤 데이터를 추출할 수 있다. IC 칩 덤프를 이용한 상세 분석 프로세스와 사용되는 도구는 [Table. 6]과 같다.



Flash Memory (ZB25VQ32BSEG)

ZB : Zbit (Manufacturer)
25 : SPI (Product Family)
VQ : 2.5V 4KB, Quad Mode (Series)
32 : 32Mbit (Device Density)

Fig. 9. Information of flash memory

Table 6. IC chip memory dump steps

Detailed steps	Tools	Purpose
Identifying IC chip	-	To identifying the type of each pin through the identification number or model name of the memory chip used in the PCB
Identifying the dump target	Magnifying Glass, microscope	Identifying dump targets such as EEPROM and FlashMemory through a magnifying glass or microscope
Acquiring a data sheet	-	Acquiring a data sheet mapped to the chip by searching the model name of the dump target
Identifying IC chip information	Data sheet	Identification of IC chip information using data sheets
Identifying instructions and Data Flow Timing[70]	Data sheet	Identifying data transmission/reception section by identifying Data Flow Timing used by command through data sheet
Identifying and connecting chip pins	Data sheet, identified IC chips	Identifying the role of each pin by comparing the pins of the identified IC chip with the data sheet
Programming for data output	Arduino, Raspberry Pi	Extracting internal data by performing programming directly from information obtained through data sheets
Using dedicated tools for data extraction	ROM Writer	Acquiring and modifying firmware by connecting a ROM writer to the identified IC chip.

3.2 펌웨어 분석 프로세스

보안팀은 위와 같이 펌웨어를 획득한 후 취약점이나 암호 모듈 식별과 같은 분석 목표를 설정한 뒤 분석을 수행해야 한다. 보안팀이 IoT 기기의 펌웨어를 분석하기 위해 필요한 프로세스는 [Fig. 10]과 같다.

펌웨어 분석 방법은 디스어셈블러나 디버거 등의 분석 도구를 사용하여 보안팀이 직접 분석을 수행하는 수동 분석과, 퍼저나 AEG(Automatic Exploit Generation)[71] 도구 등과 같이 목적에 맞게 개발된 자동화 도구를 사용하여 분석을 수행하는 자동분석으로 나눠진다.

1) 암호화 여부 식별: 보안팀은 펌웨어 획득 후 바이너리를 대상으로 디스어셈블러나 디버거를 통해 수동 분석을 수행할 수 있다. 펌웨어 분석에 있어서 가장 먼저 수행되어야 하는 프로세스는 펌웨어에 대한 암호화 및 인코딩 여부를 식별하여 분석할 수 있는지 확인하는 것이다. 보안팀은 Hex Edit 도구를 이용하여 매직 헤더, 블록 패턴 등을 확인하거나 Strings 명령어를 통해 펌웨어 내부에서 식별 가능한 문자열의 존재 여부를 확인함으로써 펌웨어의 암호화 및 인코딩 여부에 대한 식별을 간단하게 수행할 수 있다.

2) 파일시스템 분석 및 언패키징: 펌웨어는 파일 구조상 여러 개의 링크파일과 오브젝트 파일들이 하나로 패키징되어 동작하기 때문에 내부 파일들에 대한 구조분석을 수행한다. 펌웨어에 대한 구조분석은 주로 FMK(Firmware Mode Kit)[72]나 binwalk[73]를 사용하여 언패키징을 수행한다. 펌웨어는 IoT 기기에서 따라 ARM, MIPS 등 다양한 프로세서를 사용하기 때문에[74] 이를 분석하기 위해서는 언패키징 과정과 함께 펌웨어의 프로세서를 확인해야 한다. 이는 binwalk에서 Analysis 옵션(-A)을 사용하거나 Hex값의 블록 패턴을 확인하여 식별할 수 있다.

3) 분석 목표/대상 선정: 보안팀은 펌웨어에 대한 언패키징을 수행한 후 내부 바이너리 중 분석하고자 하는 대상을 선정해야 한다. 분석 대상은 펌웨어 내부의 모든 바이너리를 대상으로 선정하거나 취약할 것으로 의심되는 바이너리를 선정한다. 보안팀이 분석 대상을 선정하기 위한 예시로는 grep 명령어를 사용하여 취약한 함수(strcpy, sprintf, system 등)를 사용하는 바이너리를 찾는 방법이 있다. 또한 취약점이 발생할 수 있는 기능을 포함하는 바이너리(네트워크 취약점을 대상으로 한다면 httpd와 같은 네트워크 모듈)를 분석 대상으로 선정할 수 있다.

4) 디스어셈블러/ 디버거 연결: 보안팀은 분석 목표를 선정한 뒤 IDA Pro나 GDB(GNU Debugger)와 같은 디스어셈블러나 디버거를 통해 바이너리 분석을 수행할 수 있다. 하지만 프로세서의

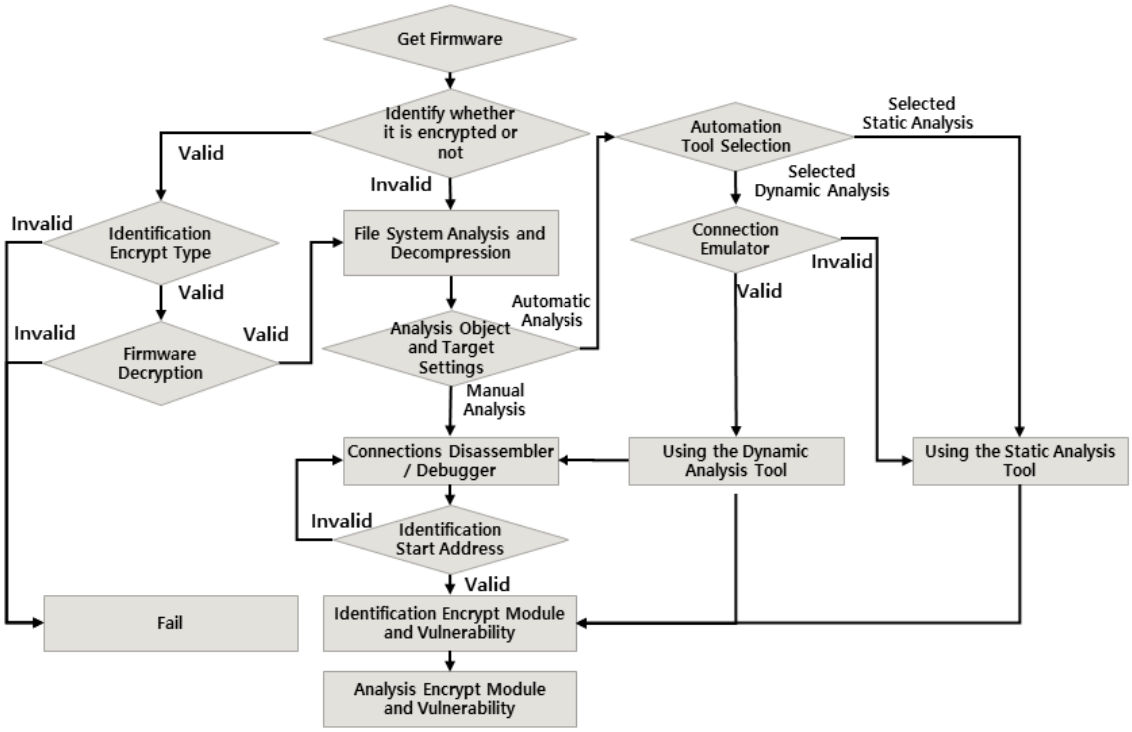


Fig. 10. Firmware analysis process

종류에 따라서 사용되는 어셈블리언어가 다르기 때문에 분석을 위해 각 프로세서에서 사용되는 어셈블리어에 대한 사전지식이 필요하다.

5) 시작 주소 식별: 보안팀은 IDA Pro와 같은 디스어셈블러를 사용한 정적 분석을 수행하기 위해 펌웨어 메모리의 시작 주소를 식별해야 한다. 메모리의 시작 주소는 바이너리 분석을 통해 레지스트리에 로드되는 메모리 주소를 확인하여 식별할 수 있고 부팅 메시지 등을 통해서 식별할 수 있다. 보안팀이 디버깅을 수행하기 전 시작 주소를 설정하지 않으면 〔Fig. 11〕과 같이 함수 블록을 구분하지 못해 분석하지 못하는 함수가 발생할 수 있다.

6) 동적 분석 도구 사용: 보안팀이 디버거를 활용하여 동적 분석을 수행하는 경우에는 QEMU(Quick Emulator)〔75〕나 Unicorn 엔진 등과 같은 가상 에뮬레이터를 사용하여 Cross compile 환경을 구축해야 한다. Cross compile 환경에서는 디버거를 사용하거나 해당 환경에 원격 디버깅을 사용하여 분석을 수행할 수 있다. 보안팀은 에뮬레이팅을 통해 동적 분석을 수행하여 메모리나 레지스터값의 변화를 확인하여 정적 분석보다 더 많은 정보를 획득할 수 있다. 또한 보안팀은 펌웨어에 대한 에뮬레이팅이 가능한 경우 퍼저나 AEG와 같은 자동 취약점 분석 도구를 사용하여 분석을 수행할 수 있으며, 분석 목적과 대상에 따라 다양한 자동화 도구를 선정하여 사용할 수 있다. 하지만 펌웨어는 특수한 하드웨어 구성요소에 의존적인 특징을 가지고 있기 때문에〔76〕 펌웨어의 에뮬레이팅 성공 가능성은 매우 낮다〔22〕.

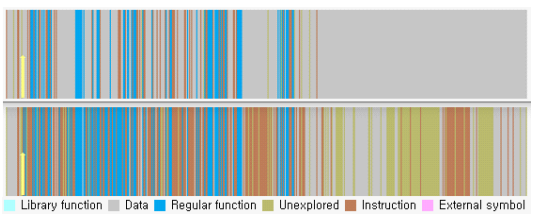


Fig. 11. Not setted start address(above), setted start address(below)

이처럼 획득한 펌웨어를 분석하기 위해 필요한 상세 분석 프로세스와 사용되는 도구는 〔Table. 7〕과 같다.

Table 7. Firmware analysis steps

Detailed steps	Tools	Purpose
Identifying whether or not to encrypt	binwalk	Identifying whether to apply encryption to the firmware by checking the file structure and entropy
	Hxd	Identifying whether to encrypt by checking the Hex value of the binary
	Strings	Identifying whether to be encrypted by checking the string inside the firmware
Identifying the encryption method	-	Identifying the encryption method applied to the firmware
	HxD	Checking the block pattern
Decrypting the firmware	Programming	Programming to decrypt encrypted firmware
Analyzing and decompressing filesystems	FMK	Repackaging firmware after decompression, modification
	binwalk	Acquiring important firmware information such as file structure, architecture, entropy, and file system
Setting analytics goals and targets	-	Setting a target for the result to be analyzed, and setting a target file corresponding to the analysis target through system commands
Identifying the architecture	binwalk	Identifying the architecture and format through file structure and file system analysis functions of binwakl
	HxD	Identifying the architecture and format through the firmware's Hex value pattern and magic number
Connecting the disassembler /debugger	IDA Pro	Converting the file to be analyzed into assembly language through IDA Pro
	GDB	Debugging using GDB for dynamic analysis
	QEMU, Unicorn	Emulating for dynamic analysis
Identifying the starting address	IDA Pro, GDB	Identifying the starting address using a disassembler, debugger, etc. to specify a starting point for analysis
Choosing an automation tool	-	Selecting an automation tool suitable for the characteristics of the analysis target
Using automatic analysis tools	Fuzzer	Analyzing the vulnerability automatically using a fuzzer
	AEG	Analyzing the vulnerability automatically using a AEG
Connecting to the emulator	QEMU, Unicorn	Configuring the environment for using dynamic analysis tools
Using static analysis tools	Disasse-mbler, Program-ming	Using static analysis tools for firmware by selecting and using an open source analysis tool through Github, etc.

IV. 분석 프로세스 검증

우리는 IoT 기기를 분석하기 위한 펌웨어 획득 및 분석 프로세스와 분석 과정에서 사용되는 도구를 정립하여 IoT 기기 분석에 대한 로드맵을 제시하였다. 또한 3장에서 정립한 프로세스를 상용 제품에 직접 적용하여 정립한 프로세스에 대한 검증을 수행하였다.

4.1 프로세스 검증

본 장에서는 3장에서 정립한 펌웨어 획득 및 분석 프로세스를 DJI, Syma, Fine 사의 상용 드론 4종에 직접 적용한 결과를 보여준다.

4.1.1 UART

드론의 공식 제조사 홈페이지를 통해 펌웨어 다운로드와 패킷 스니핑을 통한 펌웨어 획득이 불가능했기 때문에, 우리는 드론의 보드를 직접 분석함으로써 UART 포트를 식별하였다. UART 포트를 이용해 분석을 수행하기 전 Logic Analyzer를 사용하여

전송 속도를 식별하였다. 전송 속도는 Logic Analyzer를 통해 분석된 신호의 속도를 사용하여 식별할 수 있으며, 100us는 9600bit, 17us는 57600bit, 8.6us는 115200bit와 매핑된다. 분석 대상 드론의 전송 속도는 [Fig. 12]와 같이 8.5us로 115200bit를 사용한다.

전송 속도를 식별한 뒤 UART 분석 장비와 터미널 프로그램을 사용하여 드론의 UART 포트에 접근할 수 있었다. [Fig. 13]은 UART 포트를 이용한 U-Boot 쉘 획득 화면으로, 명령어를 입력할 수 있는 쉘 입력 콘솔이다.

Fig. 12. Identification baudrate

```
RamBoot: Start
Trying to boot from SPI
SF: Got idcode ef 70 17 00 00
use default flash ops...


U-Boot 2010.06 (Mar 11 2017 - 15:33:50)

DRAM:  16 MiB
SF: Got idcode ef 70 17 00 00
use default flash ops...
In:    serial
Out:   serial
Err:   serial
MMC:   FH_MMC: 0
MID:0x9c RBlock:512 WBlock:512 Chip:3831MB Name:"USD00"
SD:    Ver:2.0 High Capacity Speed:25000000Hz Bus Width:1bit
Net:   ENC28J60-MAC
reading rtthread.bin

** Unable to read "rtthread.bin" from mmc 0:1 **
Hit any key to stop autoboot:  0
U-Boot #
```

Fig. 13. Get U-Boot shell

4.1.2 IC 칩 덤프

우리는 U-Boot 쉘을 획득한 뒤 IC 칩 식별을 위해 드론의 보드를 분석한 결과 Winbond 사의 8-핀으로 구성된 SPI Flash Memory를 식별할 수 있었다.

우리는 온 칩 상태의 Flash Memory를 대상으로 덤프를 수행하기 위해 8-핀의 IC Test Clip을 사용해 CH341A라는 ROM Writer와 연결하여 펌

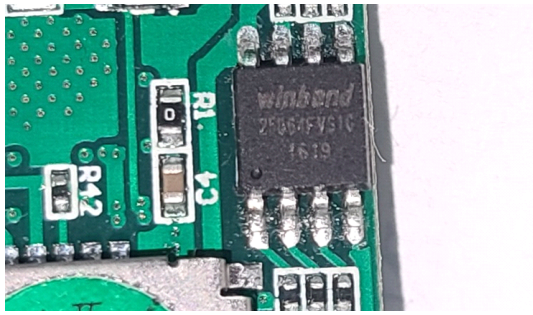


Fig. 14. Flash memory

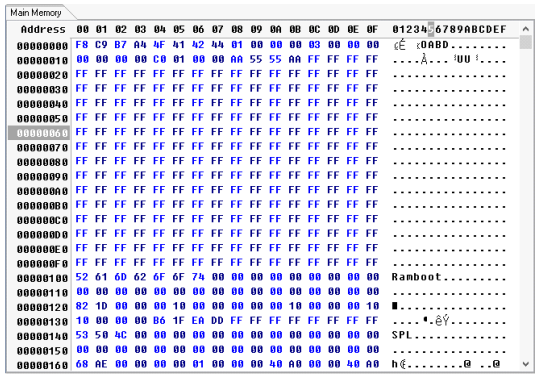


Fig. 15. Flash memory dump

웨어를 추출하였다.

4.1.3 펌웨어 분석

IC 칩 덤프를 통해 추출한 펌웨어는 ARM 프로세서를 사용하고 있었으며, binwalk 도구를 사용하여 파일시스템의 코드 블록 구조를 식별할 수 있다. 또한 우리는 UART 포트를 이용한 쉘에서 획득한 보드 정보를 통해 펌웨어 바이너리의 시작 주소를 식별하였다.

식별된 펌웨어의 프로세서와 시작 주소는 IDA Pro와 같은 디스어셈블러를 사용한 정적 분석에 사용된다. 우리는 펌웨어의 프로세서와 시작 주소를 디스어셈블러에 입력하여 펌웨어에서 사용되는 어셈블리어를 확인할 수 있었으며 CFG(Code Flow Graph)까지 확인할 수 있었다. 이후의 펌웨어 분석 과정은 일반적인 소프트웨어에 대한 코드 정적 분석

DECIMAL	HEXADECIMAL	DESCRIPTION
4372	0x1114	ARM instructions, function prologue

Fig. 16. Check firmware processor

```
U-Boot # bdinfo
arch_number = 0x0000270F
env_t       = 0x00000000
boot_params = 0xA0000100
DRAM bank   = 0x00000000
-> start    = 0xA0000000
-> size     = 0x01000000
ethaddr     = 10:20:30:40:50:60
ip_addr     = 10.81.81.81
baudrate    = 115200 bps
U-Boot #
```

Fig. 17. Get firmware start address

```
li      $gp, 0x3CAD8      # _init
addu    $gp, $t9
addiu   $sp, -0x20
sw      $gp, 0x20+var_10($sp)
sw      $ra, 0x20+var_4($sp)
sw      $gp, 0x20+var_8($sp)
bal     loc_40672C
nop
```

Fig. 18. Disassembly of firmware binary

과정과 동일하다.

Table 8. Results of each process experiment

Category	Detailed Steps	Drone A	Drone B	Drone C	Drone D
UART	Identifying the UART port	O	O	O	O
	Identifying the Baudrate	O	X	O	O
	Connecting the UART port	O	X	O	O
	Obtaining Magic Key	X	X	X	X
	Obtaining shell privileges	O	X	X	O
JTAG	Identifying the JTAG port	X	X	X	X
	Identifying the processor	O	O	X	O
	Connecting to the JTAG port	X	X	X	X
	Extracting data	X	X	X	X
IC Chip Memory Dump	Identifying IC chip	O	O	O	O
	Identifying the dump target	O	O	O	O
	Acquiring a data sheet	O	O	X	O
	Identifying IC chip information	O	O	X	O
	dentifying instructions and Data Flow Timing	O	O	X	O
	Identifying and connecting chip pins	O	O	X	O
	Programming for data output	X	X	X	X
	Using dedicated tools for data extraction	O	O	X	O

4.1.4 프로세스 검증 결과

상용 드론을 대상으로 프로세스를 검증한 결과 JTAG 포트는 식별하지 못했지만, UART 포트와 IC 칩을 식별하여 일부 기기에서는 명령어 쉘과 펌웨어를 획득하여 디스어셈블러를 통해 정적 분석을 수행할 수 있었다. 본 논문에서 제시한 프로세스의 세부 단계별 수행 결과는 〔Table. 8〕과 같다.

V. 제조사(Vendors)를 위한 권장 사항

본 장에서는 앞서 언급한 결과들을 기반으로, IoT 기기 제조사가 기기의 보안성을 향상시키기 위해 따를 수 있는 권장 사항을 설명한다.

1) 디버깅 포트 접근 제어: 제조사는 IoT 기기를 개발하는 과정에서 소프트웨어, 하드웨어 디버깅이나 Flash Memory 바이너리 다운로드를 효율적으로 수행하기 위해, 보드에 UART와 JTAG 포트를 만들어 놓고 필요할 때 이용한다. 하지만, 해당 포트들이 제대로 관리되거나 보호되지 않을 경우, 악의적인 사용자가 이를 통해 펌웨어를 추출하거나 내부 데이터 흐름을 관찰해 내부 정보를 탈취할 수 있다. 일부

Category	Detailed Steps	Drone A	Drone B	Drone C	Drone D
Firmware Download and Packet Sniffing	Checking the firmware update method	O	O	X	O
	Sniffing firmware update packets	X	X	X	X
	Browsing the manufacturer's homepage	O	O	X	O
	Searching for research material	X	X	X	O
Firmware Analysis	Identifying whether or not to encrypt	O	X	X	O
	Identifying the encryption method	X	X	X	X
	Decrypting the firmware	X	X	X	X
	Analyzing and decompressing filesystems	O	X	X	O
	Setting analytics goals and targets	O	X	X	O
	Identifying the architecture	O	O	X	O
	Connecting the disassembler/debugger	O	X	X	O
	Identifying the starting address	O	X	X	O
	Choosing an automation tool	X	X	X	X
	Connecting to the emulator	X	X	X	X
	Using automatic analysis tools	X	X	X	X
	Using static analysis tools	O	X	X	O

제조사에서는 디버깅 포트에 대한 읽기/쓰기 등의 접근 권한을 제어하거나 특수한 가공 과정을 거쳐 접근에 제한을 두거나, 포트를 완전히 제거하여 제품을 출시한다. 하지만, 아직 많은 제조사가 이에 대한 문제점을 인지하지 못하고 디버깅 포트를 노출한 상태로 IoT 기기를 출시하고 있다. 따라서, 우리는 제조사가 IoT 기기 개발 완료 이후 제품 생산 시 보드에 존재하는 디버깅 포트에 대한 접근을 제어하거나 보호하거나 이를 제거할 것을 권장한다.

2) 부트로더 보안 적용: 공격자는 디버깅 포트를 통해 IoT 기기의 부트로더에 접근한 뒤 명령어 쉘을 획득하고 명령어를 이용하여 기기의 상세 정보를 획득하거나 하드웨어에 대한 디버깅을 수행할 수 있다. 이를 방지하기 위해 우리는 IoT 기기의 부트로더에 패스워드를 설정하고 전기적 신호, 특정 입력 조건을 만족할 때만 접근할 수 있도록 할 것을 권장한다.

3) SSDLC(Secure Software Development Life Cycle)를 통한 개발[77]: SSDLC는 안전한 제품을 개발하기 위해 설계된 개발 생명 주기를 의미하며 제품을 설계 및 개발하고 테스팅하여 배포하는 전 과정에 대해 다루고 있다. IoT 기기 제조사는 제품 개발 시 SSDLC를 적용하여 결함이 상대적으로 적은 안전한 제품을 생산할 수 있다. 따라서, 우리는 제조사가 새로운 제품을 계획하기 이전에 사내 개발, 보안, 유지보수 관련 팀들이 독립적으로 수행하던 개발 과정을 SSDLC를 통해 체계적으로 통합할 것을 권장한다. 이를 통해 제조사는 안전한 제품을 생산함과 동시에 보안 사고의 발생으로 소모되는 비용을 감축할 수 있다.

VI. 결 론

IoT 기기는 금융, 개인, 산업, 공공, 군 등 점차 사용 범위가 넓어지고 있으며, 개체의 수 또한 지속적으로 증가하고 있다. 이러한 IoT 기기에 대한 공격이 발생할 경우 개인정보 유출, 금전적 손실 등 다양한 종류의 피해로 이어질 수 있기 때문에 보안팀은 IoT 기기 분석을 통해 남아있는 취약점을 밝혀내 조기에 보안 패치가 이루어질 수 있도록 해야 한다. 취약점 분석 시에는 IoT 기기와 상호작용 하는 애플리케이션, 웹 사이트 같은 소프트웨어 인터페이스를 비롯하여 펌웨어에 대한 분석 또한 필수적으로 수행되어야 한다. 하지만 IoT 기기 내의 펌웨어를 추출하고 이를 분석하는 과정이 생각보다 쉽지 않으며, 보안팀 내 분석하는 사람의 전문성에 따라 같은 대상을 분석하더라도 결과물의 수준이 차이가 발생할 수 있

어 일정한 수준의 품질 관리가 쉽지 않다는 것이다. 이를 개선하기 위해 IoT 기기에 대한 취약점 분석 시 참고할 로드맵이 제공될 필요성이 존재한다.

따라서 본 논문에서 우리는 IoT 기기의 펌웨어에 대한 취약점 분석 프로세스를 정립하고 단계별로 사용 가능한 도구를 제시하였다. 이를 위해 우리는 다양한 상용 제조사들의 IoT 기기에 대한 펌웨어 획득부터 취약점 분석 단계까지의 과정을 체계화하였다. 또한 이를 다양한 제조사의 드론을 대상으로 적용하여 분석함으로써 그 타당성을 입증하였다. 본 논문에서 도출한 결과는 향후 보안팀이 수행할 다양한 종류의 IoT 기기에 대한 취약점 분석 시 활용되어 적은 시간 내에 최선의 결과를 도출하는 데 기여할 수 있을 것으로 생각된다.

References

[1] Security Today, "The IoT Rundown For 2020: Stats, Risks, and Solutions." Security Today, 13 Jan 2020. https://securitytoday.com/Articles/2020/01/13/The-IoT-Rundown-for-2020.aspx?Page=2

[2] PALOALTO, [online] Available: https://unit42.paloaltonetworks.com/iot-threat-report-2020/

[3] Alshamrani, Adel, and Abdullah Bahattab. "A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model." International Journal of Computer Science Issues (IJCSI) 12.1 (2015): 106.

[4] Okoli, Chitu, and Kira Schabram. "A guide to conducting a systematic literature review of information systems research." (2010).

[5] IEEE. https://ieeexplore.ieee.org/

[6] NDSS. https://dblp.org/db/conf/ndss/index

[7] Usenix. https://www.usenix.org/

[8] ACM. https://dl.acm.org/

[9] NDSS. https://www.sciencedirect.com/

[10] Springer, https://www.springer.com/

[11] Zaddach, Jonas, and Andrei Costin. "Embedded devices security and firmware reverse engineering." Black-Hat USA (2013).

[12] Shwartz, Omer, et al. "Reverse engineering IoT devices: Effective techniques and methods." IEEE Internet of Things Journal 5.6 (2018): 4965-4976.

[13] Hernandez, Grant, et al. "Toward Automated Firmware Analysis in the IoT Era." IEEE Security & Privacy 17.5 (2019): 38-46.

[14] Schulz, Matthias, Daniel Wegemer, and Matthias Hollick. "The Nexmon firmware analysis and modification framework: Empowering researchers to enhance Wi-Fi devices." Computer Communications 129 (2018): 269-285.

[15] Fowze, Farhaan, et al. "ProXray: Protocol Model Learning and Guided Firmware Analysis." IEEE Transactions on Software Engineering (2019).

[16] Basnight, Zachry, et al. "Firmware modification attacks on programmable logic controllers." International Journal of Critical Infrastructure Protection 6.2 (2013): 76-84.

[17] Lee, Seoksu, and Eun-Sun Cho. "Toward Firmware-Type Analysis Using machine Learning Techniques." 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). Vol. 1. IEEE, 2018.

[18] Costin, Andrei, et al. "A large-scale analysis of the security of embedded firmwares." 23rd {USENIX} Security Symposium ({USENIX} Security 14). 2014.

[19] Xu, Yifei, et al. "A Search-based Firmware Code Analysis Method for IoT Devices." 2018 IEEE Conference on Communications and Network Security (CNS). IEEE, 2018.

[20] English, K. Virgil, Islam Obaidat, and Meera Sridhar. "Exploiting Memory Corruption Vulnerabilities in Connman for IoT Devices." 2019 49th Annual IEEE/IFIP International Conference on Dependable

Systems and Networks (DSN). IEEE, 2019.

[21] Cam, Nguyen Tan, et al. "Detect malware in android firmware based on distributed network environment." 2019 IEEE 19th International Conference on Communication Technology (ICCT). IEEE, 2019.

[22] Cheng, Kai, et al. "DTaint: detecting the taint-style vulnerability in embedded device firmware." 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2018.

[23] Liu, Muqing, et al. "Security analysis of vendor customized code in firmware of embedded device." International Conference on Security and Privacy in Communication Systems. Springer, Cham, 2016.

[24] Classen, Jiska, et al. "Anatomy of a vulnerable fitness tracking system: Dissecting the fitbit cloud, app, and firmware." Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 2.1 (2018): 1-24.

[25] Xie, Wei, et al. "Vulnerability detection in iot firmware: A survey." 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2017.

[26] Yao, Yao, et al. "Identifying Privilege Separation Vulnerabilities in IoT Firmware with Symbolic Execution." European Symposium on Research in Computer Security. Springer, Cham, 2019.

[27] Al-Alami, Haneen, Ali Hadi, and Hussein Al-Bahadili. "Vulnerability scanning of IoT devices in Jordan using Shodan." 2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS). IEEE, 2017.

[28] Krishnankutty, Deepak, et al. "Fiscal: Firmware identification using side-channel power analysis." 2017 IEEE 35th VLSI Test Symposium (VTS). IEEE, 2017.

[29] Hou, Jin-bing, Tong Li, and Cheng Chang. "Research for vulnerability detection of embedded system firmware." Procedia Computer Science 107 (2017): 814-818.

[30] Shirani, Paria, et al. "Binarm: Scalable and efficient detection of vulnerabilities in firmware images of intelligent electronic devices." International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, Cham, 2018.

[31] Cui, Ang, Michael Costello, and Salvatore Stolfo. "When firmware modifications attack: A case study of embedded exploitation." (2013).

[32] Mulliner, Collin, and Benjamin Michéle. "Read It Twice! A Mass-Storage-Based TOCTTOU Attack." WOOT. 2012.

[33] Miller, Charlie. "Battery firmware hacking." Black Hat USA (2011): 3-4.

[34] Hudson, Trammell, and Larry Rudolph. "Thunderstrike: EFI firmware bootkits for Apple MacBooks." Proceedings of the 8th ACM International Systems and Storage Conference. 2015.

[35] Papp, Dorottya, Zhendong Ma, and Levente Buttyan. "Embedded systems security: Threats, vulnerabilities, and attack taxonomy." 2015 13th Annual Conference on Privacy, Security and Trust (PST). IEEE, 2015.

[36] Choi, Byung-Chul, et al. "Secure firmware validation and update for consumer devices in home networking." IEEE Transactions on Consumer Electronics 62.1 (2016): 39-44.

[37] Konstantinou, Charalambos, and Michail Maniatakos. "Impact of firmware modification attacks on power systems field devices." 2015 IEEE International Conference on Smart Grid

Communications (SmartGridComm). IEEE, 2015.

[38] Liu, Jiajia, and Wen Sun. "Smart attacks against intelligent wearables in people-centric internet of things." IEEE Communications Magazine 54.12 (2016): 44-49.

[39] Ling, Zhen, et al. "Security vulnerabilities of internet of things: A case study of the smart plug system." IEEE Internet of Things Journal 4.6 (2017): 1899-1909.

[40] Shudrak, Maxim, and Vyacheslav Zolotarev. "The technique of dynamic binary analysis and its application in the information security sphere." Eurocon 2013. IEEE, 2013.

[41] Chen, Jiongyi, et al. "IoTFuzzer: Discovering Memory Corruptions in IoT Through App-based Fuzzing." NDSS. 2018.

[42] Manès, Valentin Jean Marie, et al. "The art, science, and engineering of fuzzing: A survey." IEEE Transactions on Software Engineering (2019).

[43] Zheng, Yaowen, et al. "FIRM-AFL: high-throughput greybox fuzzing of iot firmware via augmented process emulation." 28th {USENIX} Security Symposium ({USENIX} Security 19). 2019.

[44] Kim, Taegyu, et al. "RVFUZZER: finding input validation bugs in robotic vehicles through control-guided testing." 28th {USENIX} Security Symposium ({USENIX} Security 19). 2019.

[45] Gui, Zhijie, et al. "FIRMCORN: Vulnerability-Oriented Fuzzing of IoT Firmware via Optimized Virtual Execution." IEEE Access 8 (2020): 29826-29841.

[46] Yu, Bo, et al. "Poster: Fuzzing iot firmware via multi-stage message generation." Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019.

[47] Srivastava, Prashast, et al. "FirmFuzz: automated IoT firmware introspection and analysis." Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things. 2019.

[48] Shila, Devu Manikantan, Penghe Geng, and Teems Lovett. "I can detect you: Using intrusion checkers to resist malicious firmware attacks." 2016 IEEE Symposium on Technologies for Homeland Security (HST). IEEE, 2016.

[49] Li, Yanlin, Jonathan M. McCune, and Adrian Perrig. "VIPER: verifying the integrity of PERipherals' firmware." Proceedings of the 18th ACM conference on Computer and communications security. 2011.

[50] Eriksson, Jakob, Srikanth V. Krishnamurthy, and Michalis Faloutsos. "Truelink: A practical countermeasure to the wormhole attack in wireless networks." Proceedings of the 2006 IEEE International Conference on Network Protocols. IEEE, 2006.

[51] Cao, Fei, Qingbao Li, and Zhifeng Chen. "Vulnerability Model and Evaluation of the UEFI Platform Firmware Based on Improved Attack Graphs." 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS). IEEE, 2018.

[52] Sun, Pengfei, et al. "Hybrid Firmware Analysis for Known Mobile and IoT Security Vulnerabilities." 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2020.

[53] Shoshitaishvili, Yan, et al. "Firmalice-automatic detection of authentication bypass vulnerabilities in binary firmware." NDSS. 2015.

[54] Maskiewicz, Jacob, et al. "Mouse Trap: Exploiting Firmware Updates in {USB} Peripherals." 8th {USENIX} Workshop on

Offensive Technologies ({WOOT} 14). 2014.
〔55〕 David, Yaniv, Nimrod Partush, and Eran Yahav. "Firmup: Precise static detection of common vulnerabilities in firmware." ACM SIGPLAN Notices 53.2 (2018): 392-404.
〔56〕 Costin, Andrei, Apostolis Zarras, and Aurélien Francillon. "Automated dynamic firmware analysis at scale: a case study on embedded web interfaces." Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. 2016.
〔57〕 Visoottiviseth, Vasaka, et al. "Firmaster: Analysis Tool for Home Router Firmware." 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE). IEEE, 2018.
〔58〕 Hernandez, Grant, et al. "Firmusb: Vetting USB device firmware using domain informed symbolic execution." Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.
〔59〕 Davidson, Drew, et al. "{FIE} on firmware: Finding vulnerabilities in embedded systems using symbolic execution." 22nd {USENIX} Security Symposium ({USENIX} Security 13). 2013.
〔60〕 Zaddach, Jonas, et al. "AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares." NDSS. Vol. 14. 2014.
〔61〕 Thomas, Sam L., Flavio D. Garcia, and Tom Chothia. "HumIDIFy: a tool for hidden functionality detection in firmware." International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, Cham, 2017.
〔62〕 Chen, Daming D., et al. "Towards Automated Dynamic Analysis for Linux-based Embedded Firmware." NDSS. Vol. 16. 2016.
〔63〕 Okmianski, Anton, Mickael Graham, and Joshua B. Littlefield. "Method of identifying a home gateway using network traffic sniffing and apparatus employing the same." U.S. Patent No. 7,505,464. 17 Mar. 2009.
〔64〕 Chong, Hon Fong, and Danny Wee Kiat Ng. "Development of IoT device for traffic management system." 2016 IEEE Student Conference on Research and Development (SCOReD). IEEE, 2016
〔65〕 Lu, Chung-Ming. "Communication system for devices with UART interfaces." U.S. Patent No. 7,650,449. 19 Jan. 2010.
〔66〕 Zadigian, Timothy, Jonathan Stroud, and Michael Moriarty. "JTAG-based programming and debug." U.S. Patent No. 8,856,600. 7 Oct. 2014.
〔67〕 Rosenfeld, Kurt, and Ramesh Karri. "Attacks and Defenses for JTAG." IEEE Design & Test of Computers 27.1 (2010): 36-47.
〔68〕 Hwang, Joo-Young, et al. "Xen on ARM: System virtualization using Xen hypervisor for ARM-based secure mobile phones." 2008 5th IEEE Consumer Communications and Networking Conference. IEEE, 2008.
〔69〕 Ito, Masayuki, et al. "An 8640 MIPS SoC with independent power-off control of 8 CPUs and 8 RAMs by an automatic parallelizing compiler." 2008 IEEE International Solid-State Circuits Conference-Digest of Technical Papers. IEEE, 2008.
〔70〕 Pastrnak, Milan, et al. "Data-flow timing models of dynamic multimedia applications for multiprocessor systems." 4th IEEE International Workshop on System-on-chip for Real-time Applications. IEEE, 2004.
〔71〕 Avgerinos, Thanassis, et al. "AEG: Automatic exploit generation." (2011).
〔72〕 Firmware Mod Kit, 〔online〕 Available: https://github.com/rampageX/firmware-mod-kit/wiki.

〔73〕 Binwalk, https://github.com/ReFirmLabs/Binwalk.
〔74〕 Pa, Yin Minn Pa, et al. "IoTPOT: analysing the rise of IoT compromises." 9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15). 2015.
〔75〕 Bellard, Fabrice. "QEMU, a fast and portable dynamic translator." USENIX Annual Technical Conference, FREENIX Track. Vol. 41. 2005.
〔76〕 Zheng, Yaowen, et al. "FIRM-AFL: high-throughput greybox fuzzing of iot firmware via augmented process emulation." 28th {USENIX} Security Symposium ({USENIX} Security 19). 2019.
〔77〕 Microsoft, "Security Development Lifecycle - SDL Process Guidance Version 5.2", 2012

〈저 자 소 개〉

김 예 준 (Yejun Kim) 학생회원
2019년 2월: 순천향대학교 정보보호학과 학사
2019년 3월~현재: 고려대학교 정보보호대학원 석박사통합과정
〈관심분야〉 보안공학, 리버스 엔지니어링, 악성코드 분석

김 정 현 (Jeonghyeon Gim) 학생회원
2018년 2월: 순천향대학교 정보보호학과 학사
2020년 2월: 순천향대학교 정보보호학과 석사
2020년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 보안공학, 커넥티드 카 보안

김 승 주 (Seungjoo Kim) 종신회원
1994년~1999년: 성균관대학교 정보공학과(학사, 석사, 박사)
1998년~2004년: 한국인터넷진흥원(KISA) 팀장
2004년~2011년: 성균관대학교 정보통신공학부 부교수
2004년~현재: 한국정보보호학회 이사
2007년: 국가정보원장 국가사이버안전업무 유공자 표창
2010년: 방송통신위원회 정보통신망 침해사고 민관합동조사단 위원
2011년~현재: 고려대학교 사이버국방학과/정보보호대학원 정교수
2012년: 선관위 디도스 특별검사팀 자문위원
2014년~2015년: 육군사관학교 초빙교수
2014년~2016년: 다음카카오 프라이버시 정책 자문위원회 위원
2015년~현재: 방위사업청 방산기술보호 자문관
2016년~2018년: 개인정보분쟁조정위원회 위원
2016년~현재: 산업통상자원부 전략물자기술 자문위원
2016년~현재: 한국카카오뱅크 정보보호부문 자문교수
2017년~현재: 고려대학교 국방RMF연구센터(AR2C) 센터장
2018년~2020년: 4차산업혁명위원회 위원: 대통령직속 4차산업혁명위원회 위원
2018년~현재: 고신뢰 보안운영체제 연구센터(CHAOS) 센터장
2020년~현재: 합동참모본부 정책자문위원회 자문위원
〈관심분야〉 보안공학 및 보안내재화 방법론, 보안성 평가/인증, RMF A&A, 암호학 및 블록체인