Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.5, Oct. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.5.1185

ARM 아키텍처 기반 바이너리 정적 분석을 위한 기준 주소 분석 도구*

강 지 훈,† 류 재 철‡
충남대학교 컴퓨터공학과

A Base Address Analysis Tool for Static Analysis of ARM Architecture-Based Binary*

Ji-Hun Kang,† Jae-Cheol Ryou‡
Department of Computer Engineering, Chungnam National University

요 약

현대 사회에서는 임베디드 장비의 수가 급증하고 있다. 그러나 급증하는 임베디드 장비와 동시에 악용 가능한 취약점과 백도어 등이 계속해서 발견되고 있어 이에 대한 분석의 필요성이 끊임없이 제기되고 있는 실정이다. 이에 따라 본 논문에서는 임베디드 장비 펌웨어의 정적 분석 환경 구축을 위해 필요한 기준 주소 정보를 추출하는 도구를 개발하고, 이를 사용하여 정적 분석 환경을 구축함으로써 펌웨어 내부 문자열의 파싱과 참조를 가능하게 하고, 증가된 함수식별 개수 등을 통해 도구의 타당성을 증명한다.

ABSTRACT

In modern society, the number of embedded devices has been increasing. However, embedded devices is growing, and the backdoor and vulnerabilities are found continously. It is necessary for this analysis. In this paper, we developed a tool to extract the base address information for the static analysis environment built of the embedded device's firmware. By using this tool, we built the environment for static analysis. As a result, this point enables us to parse the strings and to check the reference. Also, through the increased number of functions, we proved the validity of the tool.

Keywords: Firmware Analysis, ARM Base Address, Binary Static Analysis

I. 서 론

현대 사회는 사물인터넷(Internet of Things) 시대를 맞이하여 임베디드 장비의 수가 기하급수적으로 증가하고 있다. 이러한 임베디드 장비는 펌웨어에 의해 구동되는데, 최근 주니퍼사의 기업용 방화벽 장비에서 백도어[1]가 발견되는 등 임베디드 장비 펌웨어 내부에서 악용 가능한 취약점/백도어가 계속해서 발견됨으로 인해 이에 대한 분석의 필요성이 대두되고 있는 실정이다. 임베디드 장비의 펌웨어를 정적 분석하기 위해서는 펌웨어 내부에 패킹(Packing)되어 있는 바이너리를 추출해야 하며, 추출된 바이너리의 분석을 위한 아키텍처 정보와 기준 주소(Base Address) 정보가 필요하다. 본 논문에서는 디스어셈블링(Disassembling) 도구를 사용하여 ARM 아키텍처 기반 펌웨어에 대한 정적 분석 환경을 구축할 시 필요한 기준 주소 정보를 펌웨어의 바이너리를

Received(08. 18. 2016), Accepted(09. 30. 2016)
* 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업의 연구결과로 수행되었음 (IITP-2016-H8501-16-1008)
* 본 논문은 2016년도 하계학술대회에 발표한 우수논문을 개선 및 확장한 것임
† 주저자, ttpwer@nate.com
‡ 교신저자, jcryou@home.cnu.ac.kr(Corresponding author)

바탕으로 추출하는 도구를 개발하였다.

본 논문의 구성은 다음과 같다. 2장에서 연구 배경에 대해 소개하고 3장에서 도구 개발에 사용된 기준 주소 분석 기법과 도구의 동작 원리에 대해 설명한다. 4장에서는 개발된 도구의 테스트 결과를 바탕으로 펌웨어의 정적 분석환경을 구축하여 타당성을 증명하고 마지막으로 5장에서 결론을 맺는다.

II. 연구 배경

2.1 펌웨어 바이너리 추출

펌웨어란 하드웨어의 제어와 사용자와의 상호작용을 수행하는 소프트웨어를 의미한다. 펌웨어는 커널과 파일시스템 등으로 구성되어 있지만 구체적으로 정의된 표준이 존재하지 않아 세부적으로 살펴보면 제조사, 모델별로 상이한 구조를 가지고 있다. 이렇게 펌웨어 별로 각기 다른 구조를 가지고 있음에도 불구하고 대부분의 펌웨어들은 임베디드 장비의 제한적인 자원과 역 공학 방지 등의 이유로 내부에 바이너리가 패킹된 형태로 존재한다. 따라서 펌웨어를 분석하기 위해서는 펌웨어 내부에 압축된 형태로 존재하는 바이너리를 언패킹(Unpacking)하는 과정이 우선되어야 한다[2]. 압축된 바이너리를 언패킹 하는 루틴은 일반적으로 부트로더에 존재한다. 이러한 부트로더는 공유기, 프린터와 같은 비교적 규모가 작은 임베디드 장비의 경우 펌웨어 내부에 포함되어 있으나 스위치, 라우터와 같이 규모가 큰 장비의 경우 펌웨어와 부트로더가 분리되어 있는 형태로 존재한다. 그러므로 분석 가능한 형태의 바이너리를 추출하기 위해서는 부트로더 분석을 통해 언패킹 루틴을 식별하고, 이를 재연하여 바이너리를 추출해야 한다.

2.2 펌웨어 정적 분석환경 구축

추출된 바이너리의 정적 분석환경을 구축하기 위해서는 펌웨어가 실행되는 아키텍처 정보와 실제 동작하는 기준 주소 정보가 필요하다. 추출된 바이너리가 ELF(Executable and Linkable Format) 파일 형식을 가지고 있을 경우 헤더에 아키텍처 정보와 기준주소 정보를 포함하고 있으므로 디스어셈블링 도구를 사용하여 바로 분석이 가능하지만 바이너리의 파일 형식이 ELF 파일과 같이 파일 헤더를 가지고 있는 형식이 아닌 순수 바이너리 형태일 경우 분석을 통해 아키텍처 정보와 기준주소 정보를 설정해 줘야 한다. 이러한 정보를 알지 못하는 상태에서 분석환경을 구축하는 것은 매우 어려운 일이고, 이러한 정보 없이 분석환경을 구축한다 해도 분석된 결과의 정확성을 보장할 수 없다.

2.3 아키텍처 정보

일반적으로 임베디드 장비에는 ARM, MIPS, PowerPC 등과 같은 RISC(Reduced Instruction Set Computer) 방식으로 설계된 CPU 아키텍처가 주로 사용된다. 바이너리를 분석할 시 이러한 CPU 아키텍처 정보를 정확하게 식별하지 못할 경우 분석 자체를 수행하지 못할 가능성이 있으며, CPU 아키텍처에 대한 정확한 정보 없이 임의의 아키텍처로 분석환경을 구축하게 될 경우 디스어셈블링 도구의 false positive로 인한 잘못된 분석결과를 도출 할 위험 또한 존재한다. 이러한 아키텍처 정보는 CPU에 따라 특정한 명령어 집합(Instruction Set)의 코드패턴이 존재하므로 바이너리의 Hex값을 통해 식별이 가능하다[3]. 각 CPU 아키텍처의 코드패턴에 익숙하지 않을 경우 펌웨어 분석도구인 Binwalk의 '-A'옵션을 통해서 명령어 집합을 식별할 수 있다.

2.4 기준 주소 정보

기준 주소는 프로그램 실행 시 주소 계산의 기준이 되는 주소로써, 상대 주소를 가산하여 절대 주소를 산출하는데 사용된다. 프로그램은 주기억 장치의 일정한 메모리 영역을 점유하지만 물리적인 주소 범위는 바뀔 수 있으므로 통상 상대주소로 작성되며, 이에 기준주소를 더하여 주기억 장치 상의 물리적 주소가 결정된다. 따라서 펌웨어에서 추출된 바이너리에 대한 정적 분석 수행 시 실제 바이너리가 동작하는 기준 주소를 정확하게 설정하지 못한다면 문자열, 함수 등의 참조관계가 올바르게 구성되지 않아 실제 바이너리의 동작과 일치하지 않는 분석결과를 얻을 가능성이 있다. 이렇듯 바이너리 파일의 정적 분석을 위해서는 기준 주소 정보가 필수적이지만 현재 공개된 기준 주소 분석 도구는 존재하지 않는다.

III. 기준 주소 분석 기법 및 도구개발

3.1 기준 주소 분석 기법

본 논문에서 제안하는 기준 주소 분석 기법은 ARM 아키텍처 기반 바이너리의 점프 테이블을 바탕으로 기준 주소 정보를 추출 하였다. 점프 테이블은 Fig. 1.과 같이 절대 주소 값을 가지고 있다. LDRLS 명령어에 의해 점프할 주소가 결정되는데, 이 때 점프 테이블의 마지막 offset+1에 해당하는 0x1520 영역이 실제 바이너리 파일이 기준 주소를 기반으로 동작할 시 첫 번째 점프 테이블의 주소(0x171320)에 해당한다. 그러므로 첫 번째 점프 테이블의 주소 값(0x171320)과 점프테이블의 마지막 offset+1(0x1520)의 차이 값을 계산한 0x16FE00이 해당 바이너리 파일의 기준 주소가 된다.

```
ROM:000014F0    STR             LR, [SP,#var_4]!
ROM:000014F4    CMP             R1, #7  ; switch 8 cases
ROM:000014F8    LDRLS           PC, [PC,R1,LSL#2] ; switch jump
ROM:000014FC    B               locret_155C ; jumptable 000014F8 default case
ROM:000014FC ; ---------------------------------------------------------------
ROM:00001500    DCD 0x171320            ; jump table for switch statement
ROM:00001500    DCD 0x171328
ROM:00001500    DCD 0x171330
ROM:00001500    DCD 0x171338
ROM:00001500    DCD 0x171340
ROM:00001500    DCD 0x171348
ROM:00001500    DCD 0x171350
ROM:00001500    DCD 0x171358
ROM:00001520 ; ---------------------------------------------------------------
ROM:00001520    LDR             LR, [SP+4+var_4],#4
ROM:00001524    B               sub_12C0
```

Fig. 1. Jump table of ARM

3.2 기준 주소 분석 도구개발

Fig. 2.는 기준 주소 분석 도구의 플로우 차트에 해당한다. 위 도구는 파이썬 언어를 사용하여 개발하였고, 분석대상 펌웨어의 바이너리를 입력 받아 동작한다. ARM의 바이너리를 16진수로 변환할 경우 LDRLS 명령어에 해당하는 2바이트 0x9F 0x97을 바이너리 내부에서 검색한다. LDRLS 명령어가 발견될 경우 LDRLS 명령어 앞에 CMP 명령어에 해당하는 1바이트 기계어 코드 0xE3이 존재하는지 확인한다. 존재한다면 LDRLS 명령어 다음 offset+5에 위치하는 점프 테이블의 처음 4바이트 주소 값을 저장한 후 실제 첫 번째 점프테이블의 실행 영역에 해당하는 점프테이블의 마지막 offset+1의 값과 sub연산을 통해 기준 주소를 계산하고 이를 출력한다.

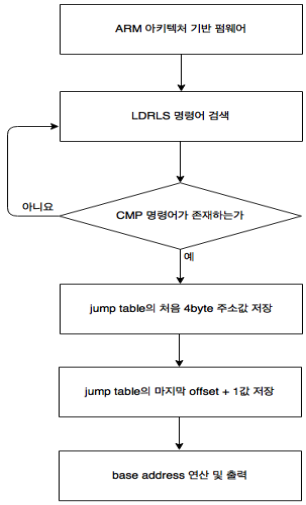

Fig. 2. Flowchart

IV. 기준 주소 분석 도구 테스트

4.1 기준 주소 미 설정

기준 주소 분석 도구의 테스트를 위해 H사의 프린터 펌웨어를 분석 대상 펌웨어로 선정 하였다. Fig. 3.은 펌웨어 내부에서 추출한 바이너리 파일을 IDA Pro를 사용하여 아키텍처 정보 설정 뒤 기준 주소를 0x0(default)으로 설정하고 분석 환경을 구축한 화면이다. 이 경우 메시지 창을 통해 알 수 있듯이 문자열에 대한 참조가 되지 않아 정상적인 분석이 불가능하며, 좌측 상단의 분석 그래프 또한 연속

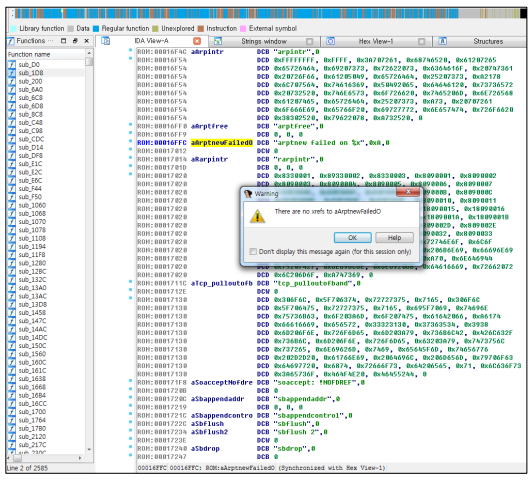

Fig. 3. Analysis environment that do not set the base address

되지 않고 깨져있는 것을 확인 할 수 있다. 바이너리 내부 함수는 총 2585개가 식별되었다.

4.2 기준 주소 분석 도구 테스트

기준 주소 분석 도구를 사용하여 분석 대상 바이너리를 분석한 결과 Fig. 4.와 같이 기준 주소가 0x80002000인 것을 확인 할 수 있다. 위 기준 주소 분석 도구는 32bit ARM 모드로 컴파일 된 바이너리를 대상으로 동작하며, ARM 명령어를 16bit로 압축하여 사용되는 Thumb 모드로 컴파일 된 바이너리를 대상으로 사용 불가능하다.

획득한 기준 주소 정보는 IDA Pro의 설정 창을 통해 Fig. 5.와 같이 ROM start address로 설정할 수 있다. 기준 주소를 설정할 경우 Fig. 3.에서는 참조 되지 않던 문자열이 Fig. 6.에서와 같이 참조되는 것을 확인 할 수 있다. 또한 파싱(Parsing)되지 않던 문자열이 파싱되어 식별 가능한 문자열 형태로 변화 하였으며, 좌측 상단의 분석 그래프가 전보다 연속된 형태로 이루어진 것을 확인 할 수 있다. 그리고 식별된 함수의 개수가 2585개에서 3497개로 약 900개 증가 하였다.

```
C:₩Users₩jihun₩Desktop>python firmware_analysis.py unpack
************************************************

  - Architecture : ARM
  - Endianness   : Little Endian
  - Base address : 0x80002000L

   Set the ROM start address! 0x80002000L

************************************************
```

Fig. 4. Base address analysis tool

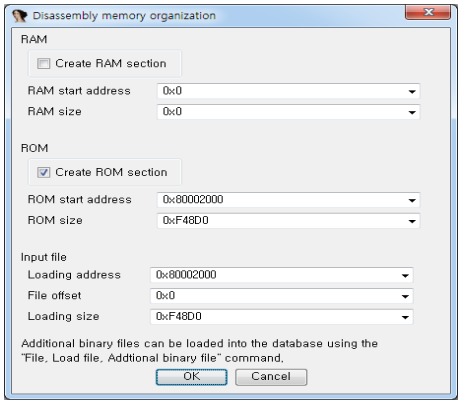

Fig. 5. Base address setting window

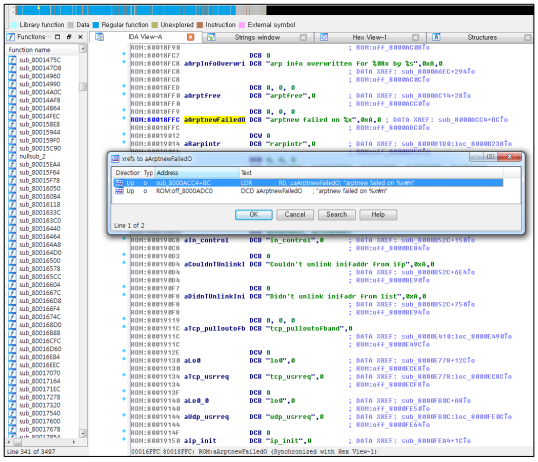

Fig. 6. Analysis environment that do set the base address

V. 결 론

본 논문에서는 ARM 아키텍처 기반 바이너리에서의 기준 주소 분석 도구를 개발하였다. 또한 개발된 도구를 바이너리에 적용시켜 정확한 기준 주소 값을 획득할 수 있었고, 이를 통해 펌웨어의 정적 분석 환경 구축이 가능하였다. 향후 다양한 임베디드 장비의 펌웨어에 대한 분석을 진행하며 취약점/백도어 탐지에 대해 연구할 계획이다.

References

[1] ScreenOS Authentication Backdoor, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-7755

[2] A. Costin, J. Zaddach, A. Francillon and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares," In Proc. 23rd USENIX Security Symposium, pp. 95-110, Aug. 2014.

[3] J. Zaddach and A. Costin. Embedded Devices Security and Firmware Reverse Engineering. Black-Hat USA, 2013.

〈저 자 소 개〉

강 지 훈 (Ji-hun Kang) 학생회원
2015년 2월: 한밭대학교 멀티미디어공학과 졸업
2015년 3월~현재: 충남대학교 컴퓨터공학과 석사과정
〈관심분야〉 펌웨어 분석, 시스템 보안, 역 공학

류 재 철 (Jae-cheol Ryou) 종신회원
1985년 2월: 한양대학교 산업공학과 졸업
1988년 5월: Iowa State University 전산학 석사
1990년 12월: Northwestern University 전산학 박사
1991년 2월~현재: 충남대학교 컴퓨터공학과 교수
〈관심분야〉 정보보호, 네트워크 보안, 암호학, 보안 프로토콜