JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 28 No. 2, pp. 1-7, February 2023
https://doi.org/10.9708/jksci.2023.28.02.001

A firmware base address search technique based on MIPS architecture using $gp register address value and page granularity

Seok-Joo Mun*, Young-Ho Sohn**

*Researcher, Institute of Industrial Technology, Yeungnam University, Gyeongbuk, Korea
**Professor, Dept. of Computer Engineering, Yeungnam University, Gyeongbuk, Korea

[Abstract]

In this paper, we propose a base address candidate selection method using the $gp register and page granularity as a way to build a static analysis environment for firmware based on MIPS architecture. As a way to shorten the base address search time, which is a disadvantage of the base address candidate selection method through inductive reasoning in existing studies, this study proposes a method to perform page-level search based on the $gp register in the existing base address candidate selection method as a reference point for search. Then, based on the proposed method, a base address search tool is implemented and a static analysis environment is constructed to prove the validity of the target tool. The results show that the proposed method is faster than the existing candidate selection method through inductive reasoning.

▸ **Key words**: MIPS, Base Address, Firmware, Firmware Analysis, IoT

[요 약]

본 논문에서는 MIPS 아키텍처 기반 펌웨어의 정적분석 환경을 구축하기 위한 방법으로, $gp 레지스터와 페이지 입상도를 활용한 베이스 주소 후보군 선정 방식을 제안한다. 해당 연구는 기존 연구의 귀납적 추론을 통한 베이스 주소 후보군 선정 방식의 단점인 베이스 주소 탐색 시간 단축을 위한 방법으로 기존 베이스 주소 후보군 선정방식 내 $gp 레지스터를 탐색의 기준점을 바탕으로 페이지 단위의 탐색을 수행하는 방법을 제시한다. 이후, 제시된 방법을 바탕으로 베이스 주소 탐색 도구를 구현 및 정적분석 환경구축을 통해 대상 도구의 타당성을 증명하고자 한다. 본 논문에서 제시된 방법은 기존 귀납적 추론을 통한 후보군 선정 방안보다 속도 면에서 더 우수함을 나타낸다.

▸ **주제어**: MIPS, 베이스 주소, 펌웨어, 펌웨어 분석, IoT

• First Author: Seok-Joo Mun, Corresponding Author: Young-Ho Sohn
*Seok-Joo Mun (hooohly@naver.com), Institute of Industrial Technology, Yeungnam University
**Young-Ho Sohn (ysohn@ynu.ac.kr), Dept. of Computer Engineering, Yeungnam University
• Received: 2023. 01. 26, Revised: 2023. 02. 22, Accepted: 2023. 02. 22.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

최근 저전력 무선통신 기술의 발달과 임베디드 아키텍처 기술의 발달로 다양한 IoT(Internet of Things) 기기들이 폭발적으로 증가하면서[1], 임베디드 기기들을 대상으로한 보안적인 위협 또한 빠르게 증가하고 있다. 이러한 임베디드 기기들은 기기 구동을 위한 특수한 바이너리인 펌웨어 파일을 통해 구동된다.

펌웨어는 대상 임베디드 기기의 모든 요소를 정상 구동시키기 위함을 그 목적으로 한다. 따라서 기기 내 기능 실행을 위한 바이너리들과 사용 데이터 등, 기기 구동을 위한 모든 핵심 요소들을 포함하고 있다. 공격자들은 이러한 요소들 중, 보안적으로 악용 가능한 취약 요소들을 활용하여 루트킷, 백도어 등, 임베디드 장비에 대한 공격을 꾸준히 시도하고 있다[2]. 이러한 공격자들로부터의 공격을 방지하기 위해서는 펌웨어에 대한 역분석(Reverse Engineering) 과정을 반드시 필요로 한다.

펌웨어 파일은 여러 종류의 바이너리 이미지가 합쳐진 패킹(Packing)된 형태를 지닌다. 따라서 분석대상 이미지 파일을 식별 및 분석하기 위해서는 패킹되어있는 펌웨어 내의 분석 대상 이미지 추출 과정이 선행되어야하며, 분석대상 이미지의 분석 및 추출을 위해서는 기존에 공개된 펌웨어 분석 및 이미지 추출도구[3]를 사용하여 펌웨어 내 이미지들을 분석 및 추출할 수 있다.

추출된 이미지에 대한 정적분석 환경구축을 위해서는 대상 이미지의 아키텍처 정보와 더불어 이미지 베이스 주소(Image Base Address) 정보를 필요로 한다.

이미지의 아키텍처 정보의 경우, 이전에 제시되었던 웨어 분석도구 사용을 통해 획득할 수 있으나, 이미지 베이스 주소의 경우 부트로더(Bootloader) 내에 존재하므로 추가적으로 부트로더 이미지를 식별하여 분석하는 과정을 필요로 한다. 따라서 이미지 베이스 주소에 대한 복원은 분석자의 경험에 근거한 추론에 의존하여 복원하는 경향이 있다. 그러나 이를 해결하고자하는 논문들은 과거에 존재하지 않았던 실정이다.

최근 이러한 어려움을 해결하고자 분석대상 이미지 내 절대주소를 활용하여 분석대상의 베이스 주소를 자동으로 선별하는 연구가 발표되었다[4].

본 논문은 기존의 문자열 절대주소 통계 및 문자열 참조를 기반 MIPS 펌웨어 베이스 주소 선별방안과 이를 개선한 학술연구[5] 내용들을 바탕으로, MIPS 아키텍처 기반의 펌웨어 이미지를 대상으로 한 베이스 주소 선별 도구를 제안한다.

본 논문의 구성은 2장에서는 해당 논문을 이해하기 위한 사전 지식, 연구의 배경 및 본 논문에 사용된 관련 연구에 대해 소개하고, 3장에서는 도구의 개발에 사용된 이론과 이미지 베이스 주소를 분석하는 도구의 동작원리에 대해 설명한다. 4장에서는 개발된 도구를 활용하여 실험한 결과를 바탕으로 해당 도구를 증명하며, 마지막 5장에서는 결론으로 끝을 맺을 것이다.

II. Preliminaries

1. Backgrounds

1.1 MIPS Architecture

MIPS 아키텍처는 파이프라이닝(Pipelinine) 효율을 극대화하기 위함을 목적으로 설계된 RISC(Reduced Instruction Set Computer) 방식의 아키텍처로, 세 가지 고정된 크기의 명령어 세트로 구성된다. 명령어 세트의 크기는 아키텍처의 1 word 크기에 고정되며, MIPS32 기준, 4-byte의 고정된 크기의 세 가지 타입의 명령어로 구성된다. 각 타입은 용도에 따라 R-Type, I-Type, J-Type으로 구분된다[6].

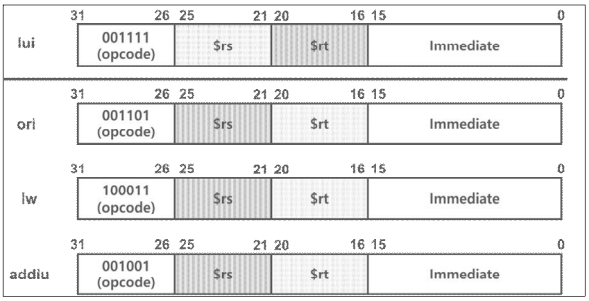

Fig. 1. I-Type instruction in MIPS architecture

Fig. 1.은 MIPS 아키텍처의 I-Type 명령어 구조를 나타낸 것이다[4]. I-Type의 경우, 주로 절대주소를 지정하는데 사용되는 명령어 타입으로, 32-bit 레지스터 내 직접 값을 삽입하기 위한 네 가지 명령어로 구성되어 있다. 대상 명령어는 상위 16-bit를 구성하기 위한 명령어 lui와 하위 16-bit를 삽입하기 위한 세 가지 명령어 lw, ori, addiu로 구성된다.

1.2 Image Base Address

이미지 베이스 주소는 프로그램의 로드 시, 절대 주소를 산출하기 위한 기준 주소로써, 펌웨어 이미지의 세그먼트(Segment)가 로드되는 메모리상의 첫 위치를 의미한다.

일반적으로 ELF(Executable and Likable Format), PE(Portable Executable) 등과 같은 실행 포맷의 경우, 헤더 내에 세그먼트의 베이스 주소에 대한 단서가 포함되어 있어 링커(Linker)가 헤더 내의 정보를 읽어 세그먼트를 메모리 내에 배치한다[7, 8].

임베디드 장비에 사용되는 펌웨어 파일의 경우, 부트로더(Bootloader)가 메모리 내에 펌웨어 이미지를 로드하는 역할을 한다. 따라서 부트로더를 분석하면, 메모리 상 이미지가 로드되는 위치인 베이스 주소를 획득할 수 있다. 그러나 이는 부트로더 분석을 위한 추가적인 자원을 소모한다.

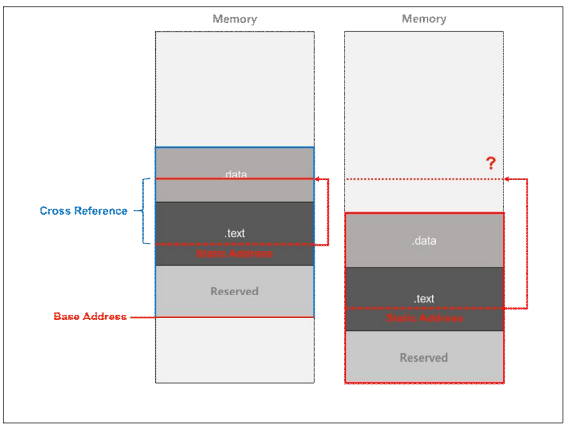

Fig. 2. Utilize of a base address in firmware

Fig. 2.는 펌웨어 베이스 주소를 적용하여 절대주소 영역에 접근하는 방법을 나타낸 것이다. 메모리상의 실제 위치에 접근하기 위해 펌웨어 베이스 주소를 기준으로 코드 내 오프셋을 가산하여 절대주소를 산출한다. 따라서 베이스 주소가 정확하게 설정되지 않는다면 정적분석 수행 시, 정상적인 참조관계가 구성되지 않아, 함수, 문자열 등의 요소에 대한 정상접근이 불가하다.

1.3 $gp register

$gp 레지스터는 전역 데이터에 대한 접근 시 기준이 되는 주소를 저장하는 용도로 사용되며, 상대주소를 대상 레지스터의 값에 가산하여 절대 주소를 산출하기 위해 사용된다. 이를 통해 절대주소로의 상대주소 접근을 가능하게 한다.

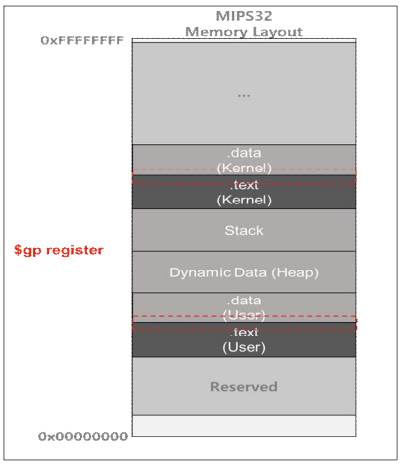

Fig. 3. location of $gp register in a firmware

Fig. 3.은 MIPS 펌웨어의 메모리 영역을 나타낸 것이다. $gp 레지스터의 값은 전역 데이터의 영역 일부에 존재하며, 하나의 가산 및 감산 명령어를 활용, 최대 64KB의 영역에 대한 빠른 접근을 지원한다.

$gp 레지스터를 사용한 데이터 접근 방식(gp-Relative Addressing)은 MIPS의 ECOFF(Extended Common Object File Format) 형식에서 지원한다[9].

2. Related works

베이스 주소를 추론하는 방법에 있어 가장 정확도가 높은 방법은 펌웨어의 부트로더 코드 분석을 통한 베이스 주소 획득 방법이다. 그러나 이러한 방법은 펌웨어 내 부트로더의 존재 여부에 따라 분석이 제한될 수 있으며, 베이스 주소를 찾아내기 위해 분석자의 과도한 분석 시간을 요구한다는 단점이 존재한다. 이러한 단점으로 인해 기존에는 주로 분석자들의 경험적 추론에 의거하여 베이스 주소를 획득하는 방법을 사용해왔다.

지난 2010년에 발표된 자료[10]에 따르면, 대상 RAW 바이너리 내에 존재하는 바이너리 재배치, 초기화, 점프 테이블, 문자열 테이블 요소들을 활용하여 절대주소 내의 주소 값에서 오프셋 내에 포함된 베이스 주소를 추론하는 방법을 제안하였다. 그러나 이러한 방법들은 분석자의 분석 경험에 의존하므로 베이스 주소 탐색 시간의 일관성을 보장할 수 없다.

ARM 기반의 펌웨어를 대상으로 한 베이스 주소 선정 자동화 연구의 경우, 펌웨어 내 절대주소를 가진 요소들을 활용하여 베이스 주소를 유추하는 방법들이 제시되었다. 그러나 해당 연구들에서 사용된 점프 테이블(Jump Table) 및 Literal Pool의 경우, MIPS 아키텍처 기반의 펌웨어 내에는 대상 요소들이 존재하지 않아, 해당 요소를 활용할

수 없다[11, 12].

최근 MIPS 기반의 펌웨어를 대상으로 한 베이스 주소 선별 자동화 방식의 경우, 귀납적 추론을 통한 문자열 매칭 베이스 주소 탐지 방법을 사용한다[4]. 본 방법은 MIPS 아키텍처 기반의 펌웨어를 대상으로 문자열 절대주소를 지정하는 세 가지 명령어 쌍, lui-ori, lui-lw, lui-addiu의 절대 주소를 수집하여 주소의 등장 빈도가 빈번한 주소 범위를 파악한다. 파악한 주소의 범위는 "베이스 주소 후보군"이라 명명하며, 문자열 주소에서 오프셋을 추출하기 위한 값으로 사용한다.

절대주소에 포함된 오프셋을 추출하기 위한 방법은 문자열의 주소에서 베이스 주소 후보군을 제외하는 방법을 사용한다. 베이스 주소 후보군은 1-byte씩 이동, 문자열의 절대 주소에서 베이스 주소를 제외한 뒤, 오프셋의 위치로 접근하여 대상 위치에 완성된 문자열이 존재하는지를 확인하며, 일치하는 문자열들의 개수가 가장 많은 주소를 최종 베이스 주소 후보군으로 선별한다. 그러나 1-byte 단위로의 베이스 주소 후보군 선별 방법은 불필요한 주소군까지 베이스 주소 검증을 수행함으로써, 베이스 주소 후보군 탐지에 과다한 시간이 소요된다는 단점이 존재한다.

III. The Proposed Scheme

3.1 A firmware base address searching method using $gp register

본 논문에서는 기존 문자열 절대주소 통계 및 문자열 참조를 기반 MIPS 펌웨어 베이스 주소 선별방안 연구의 문자열 기반 매칭 방법을 기반으로, $gp 레지스터의 주소 범위 내 페이지 입상도 값을 적용한 베이스 주소 선정방법을 제안한다.

```
ROM:00000184                # ---------------------------------------------------------
ROM:00000184 3C 1C 80 2A+       li      $gp, 0x802A3960  # Load Immediate
ROM:00000184 27 9C 39 60
ROM:0000018C 03 E0 00 08        jr      $ra              # Jump Register
ROM:00000190 00 00 00 00        nop
ROM:00000194                # ---------------------------------------------------------
ROM:00000194 3C 08 A0 00+       lw      $t0, 0xA0000000  # Load Word
ROM:00000194 8D 08 00 00
ROM:0000019C 03 E0 00 08        jr      $ra              # Jump Register
ROM:000001A0 00 00 00 00        nop
ROM:000001A4                # ---------------------------------------------------------
```

Fig. 4. Utilize of $gp register in a fimware

$gp 레지스터의 주소 값은 기존 논문의 문자열의 탐지 방식과 동일한 I-Type 명령어, lui-ori, lui-lw, lui-addiu의 세 가지 명령어 쌍에 대한 검색을 통해 대상 레지스터의 주소 값을 획득할 수 있다.

Fig. 4.와 같이 $gp 레지스터의 주소 값은 절대 주소 0x802A3960임을 확인할 수 있다. 대상의 주소는 절대주소 내의 요소를 하나의 가산 혹은 감산 명령어로 접근하기 위해 절대 주소의 형태를 가진다. 대상 주소는 펌웨어의 베이스 주소를 기반으로, 각 세그먼트들의 오프셋 위치가 합쳐진 형태로 이루어져 있다.

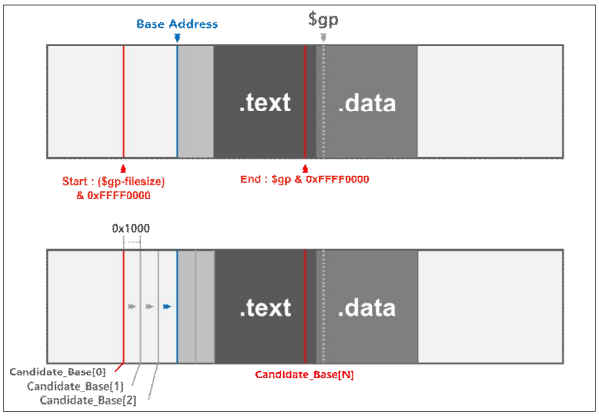

Fig. 5. Detection Flow Using $gp Register and Page Granularity

Fig. 5.는 $gp 레지스터와 페이지 입상도를 활용한 베이스 주소 탐색의 흐름을 나타낸 것이다. 검색의 범위는 펌웨어 전체 이미지의 크기를 해당 이미지에서 제외한 뒤, 세그먼트의 첫 위치부터 검색하기 위해, 하위 2-byte를 마스킹 한 주소에서부터, $gp 레지스터의 값에서 하위 2-byte를 마스킹 한 주소까지를 베이스 주소 후보군 탐색 범위로 사용한다[6]. 이를 통해 베이스 주소가 포함된 주소의 위치부터 현재 $gp 레지스터가 지정할 수 있는 세그먼트 위치까지가 주소검색의 범위가 된다.

베이스 주소 탐색은 페이지 입상도 단위로 수행한다. 이는 베이스 주소가 세그먼트 단위로 정렬되므로, 검색의 후보군의 검색 단위는 페이지 입상도 단위로 검색할 경우 빠른 탐색이 가능하기 때문이다. 페이지의 크기는 32-bit 기준, 접근 단위인 4KB(0x1000)가 된다[5].

3.2 Development of Image-Base address detection tool

이미지 베이스 주소 탐지 도구 개발은 Python 3.10.6을 사용하여 개발하였으며, 구동을 위해서는 해제된 펌웨어 내의 분석 이미지와 매칭할 문자열 개수를 필요로 한다.

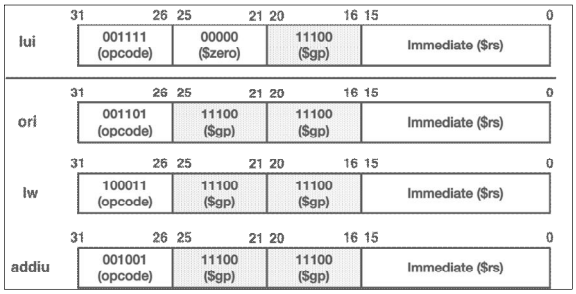

Fig. 6. $gp register instruction in MIPS architecture

Fig. 6.은 MIPS 아키텍처 기반 펌웨어 내에서 $gp 레지스터 주소 삽입 명령어 식별에 사용되는 명령어들을 나타낸 것이다. $gp 레지스터의 주소 삽입을 위해서는 lui $gp immediate와 짝이 되는 세 가지 명령어들을 식별한다.

명령어의 식별은 해당 명령어의 식별은 기존 논문의 문자열 검색 알고리즘과 동일하다[4].

$gp 레지스터의 식별을 위해 상위 2-byte 기준, 0x3C 0x1C를 기준으로 짝이 되는 세 가지 명령어 ori(0x37, 0x80), lw(0x8F, 0x80), addiu(0x27, 0x80)에 대해 명령어를 검색한다. 16-byte이내에 해당 명령어가 있다면, 이를 짝 명령어로 판단한다.

다음 Fig. 7.은 도구 동작의 흐름을 나타낸 것이다.

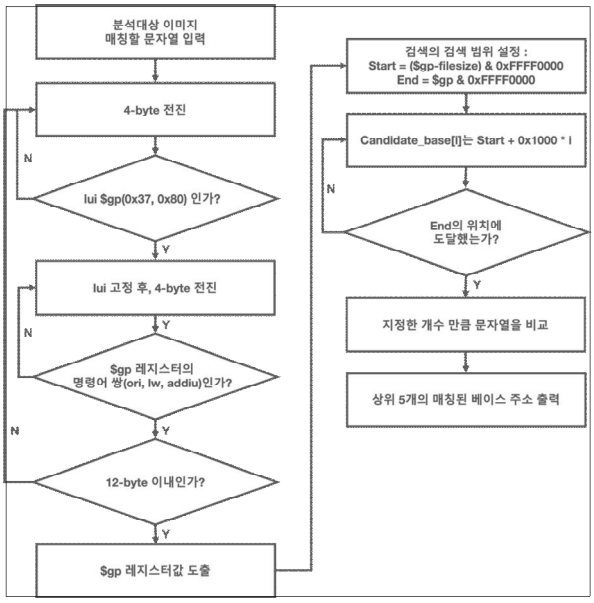

Fig. 7. Run flow of the detection tool

$gp 레지스터 주소 값 검색은 이미지의 가장 최상단에서부터 하단까지 4-byte 단위로 검색을 수행하며, 레지스터의 값이 발견될 경우, $gp 레지스터 검색을 종료하고 베이스 주소 후보군 선정을 위한 마스킹을 수행한다.

마스킹 된 범위 내에서의 베이스 주소 후보군 선별은 페이지 입상도 단위로 수행된다. 검색의 시작과 끝은 이전의 $gp 레지스터의 주소 값에서 분석대상 이미지의 크기를 제외하고, 하위 2-byte를 마스킹 한 값에서부터 $gp 레지스터의 하위 2-byte를 마스킹 한 값까지이다. 해당 주소는 베이스 주소 후보군 선정을 위한 범위가 되며, 마스킹 된 주소의 하위 주소부터 상위 주소까지 0x1000 바이트씩 가산하여 베이스 주소 후보군을 선별한다. 선별된 베이스 주소 후보군 중, 베이스 주소 선별 과정은 문자열의 절대 주소에서 후보군의 주소를 제외한 뒤, 해당 오프셋에 접근, 문자열의 완성여부를 확인하는 방법을 사용한다. 이는 기존 논문에서의 문자열 검증 매칭 알고리즘을 사용을 의미하며, 가장 문자열 매칭의 수가 많은 주소 값 5개를 베이스 주소로 선정한다.

IV. Evaluation

4.1 Base address not set state

$gp 레지스터의 주소 값을 사용한 베이스 주소 탐지 도구의 테스트를 위해서 T사의 스위치의 펌웨어를 선정하였으며, 정적 분석을 위해 IDA Pro를 사용하였다.

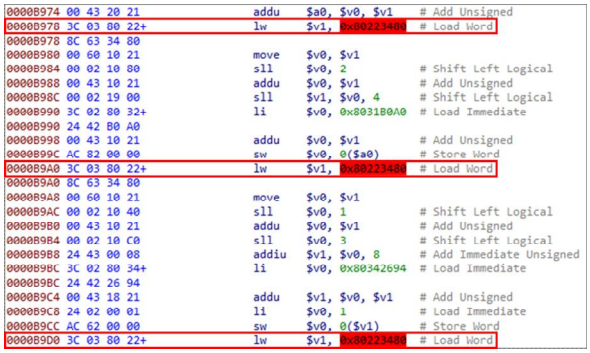

Fig. 8. Base Address not set statement

Fig. 8.은 분석 대상 펌웨어의 아키텍처 정보를 설정하고, 펌웨어 베이스 주소를 초기상태인 0x00000000으로 설정하여 정적분석 환경을 구축한 것이다. 이 경우, 정상적인 분석환경 내 정상적인 상호 참조가 구성되지 않아 절대 주소 영역에 접근할 수 없는 것을 확인할 수 있다.

4.2 Tool testing

$gp 레지스터의 주소 값을 사용한 베이스 주소 탐지 도구의 탐지 결과 Fig. 9.에 나타난 바와 같이 탐지된 베이스 주소는 0x80010000임을 확인할 수 있다.

```
❯ python3 ./main.py -t ./TARGET_FW/TPLINK_WR543G/20081013145812/_wr543gv2-en-up.bin.extracted/54B5 -fw ./TARGET_FW/TPLINK_WR543G/20081013145812/wr543gv2-en-up.bin -w 4 -od big
[#] Target image :  ./TARGET_FW/TPLINK_WR543G/20081013145812/_wr543gv2-en-up.bin.extracted/54B5
[#] Target Firmware Size : 1048196-bytes
[#] Finding $gp register value...
   → $gp : 0x802a3960
   → Range of candidate Base address : 0x80010000 ~ 0x802a0000

[#] Num of String :  13742
[#] Firmware Size : 1023-KB
[#] Num of Matching :  268
[#] Extract String :  51
[=] Matching rate
   → 0x80010000 : 911
   → 0x800b2000 : 338
   → 0x801b5000 : 319
   → 0x80079000 : 275
   → 0x80128000 : 223
[!] TIME RESULT : 10.179567098617554
```

Fig. 9. Results of running the implemented tool

획득한 베이스 주소는 정적분석 툴인 IDA Pro의 'Edit-Segment-Rebase program' 설정을 통해 대상 이미지의 시작점을 조절하여 확인할 수 있으며, Fig. 10.과 Fig. 11.은 구현한 도구를 통해 얻어낸 베이스 주소 삽입 시, 펌웨어 이미지 내 상호참조 구축을 나타낸 것이다.

대상의 상호참조가 정상적으로 구성됨을 확인할 수 있다.

```
CODE:8001B978 3C 03 80 22+              lw      $v1, dword_80223480  # Load Word
CODE:8001B978 8C 63 34 80
CODE:8001B980 00 60 10 21               move    $v0, $v1
CODE:8001B984 00 02 10 00               sll     $v0, 2          # Shift Left Logical
CODE:8001B988 00 43 10 21               addu    $v0, $v1        # Add Unsigned
CODE:8001B98C 00 02 19 00               sll     $v1, $v0, 4     # Shift Left Logical
CODE:8001B990 3C 02 80 32+              li      $v0, 0x803180A0  # Load Immediate
CODE:8001B990 24 42 80 A0
CODE:8001B998 00 43 10 21               addu    $v0, $v1        # Add Unsigned
CODE:8001B99C AC 82 00 00               sw      $v0, 0($a0)     # Store Word
CODE:8001B9A0 3C 03 80 22+              lw      $v1, dword_80223480  # Load Word
CODE:8001B9A0 8C 63 34 80
CODE:8001B9A8 00 60 10 21               move    $v0, $v1
CODE:8001B9AC 00 02 10 40               sll     $v0, 1          # Shift Left Logical
CODE:8001B9B0 00 43 10 21               addu    $v0, $v1        # Add Unsigned
CODE:8001B9B4 00 02 10 C0               sll     $v0, 3          # Shift Left Logical
CODE:8001B9B8 24 43 00 08               addiu   $v1, $v0, 8     # Add Immediate Unsigned
CODE:8001B9BC 3C 02 80 34+              li      $v0, 0x80342694  # Load Immediate
CODE:8001B9BC 24 42 26 94
CODE:8001B9C4 00 43 18 21               addu    $v1, $v0, $v1   # Add Unsigned
CODE:8001B9C8 24 02 00 01               li      $v0, 1          # Load Immediate
CODE:8001B9CC AC 62 00 00               sw      $v0, 0($v1)     # Store Word
CODE:8001B9D0 3C 03 80 22+              lw      $v1, dword_80223480  # Load Word
```

Fig. 10. Cross-reference recovery through base address revision

```
sub_80030034:

var_10= -0x10
var_C= -0xC
var_8= -8
var_s0=  0
var_s4=  4
var_s8=  8

addiu   $sp, -0x30       # Add Immediate Unsigned
sw      $ra, 0x20+var_s8($sp)  # Store Word
sw      $s1, 0x20+var_s4($sp)  # Store Word
sw      $s0, 0x20+var_s0($sp)  # Store Word
move    $s1, $a0
jal     sub_801C1784     # Jump And Link
move    $s0, $a1
move    $a3, $v0
move    $a2, $s0
move    $a1, $s1
li      $a0, aErrorDSErrnoD  # "Error =%d\t[%s] errno:%d\n"
sw      $zero, 0x20+var_10($sp)  # Store Word
sw      $zero, 0x20+var_C($sp)  # Store Word
jal     sub_801C679C     # Jump And Link
sw      $zero, 0x20+var_8($sp)  # Store Word
lw      $ra, 0x20+var_s8($sp)  # Load Word
move    $v0, $s1
lw      $s0, 0x20+var_s0($sp)  # Load Word
lw      $s1, 0x20+var_s4($sp)  # Load Word
jr      $ra              # Jump Register
addiu   $sp, 0x30        # Add Immediate Unsigned
 # End of function sub_80030034
```

Fig. 11. String matching to check for a base address match.

4.3 Result of comparison with previous research.

본 항목에서는 구현한 도구와의 효율성 비교를 위해 기존의 문자열 절대주소 통계 및 문자열 참조 기반 MIPS 펌웨어 베이스 주소 선별방안에서 제시한 내용을 바탕으로 도구를 구현하고, 제조사가 각기 다른 세 가지의 MIPS 펌웨어를 대상으로 베이스 주소를 선별하는 시간을 측정, 두 도구를 비교하는 실험을 진행하였다. 실험환경은 Mac M1 Pro 10 코어, 32GB 환경에서 실험을 진행하였으며, 구성언어는 Python 3.10.6을 사용하였다.

Table 1.은 $gp 레지스터 기반 페이지 입상도 단위 베이스 주소 탐색 도구와 기존 연구인 문자열 절대주소 통계 및 문자열 참조 기반 MIPS 펌웨어 베이스 주소 선별방안 바탕으로 구현한 귀납적 추론을 통한 문자열 매칭 베이스 주소 탐색 도구 간의 효율성을 비교한 것이다. 각 펌웨어는 이미지의 크기가 다른 각각 56.2KB, 2.66MB, 4.24MB 이며, 구현된 도구의 선별 주소 매칭 문자열의 개수는 10,000개를 사용하였다. 이때 기존 연구를 바탕으로 구현한 도구를 A, 본 연구에서 구현한 도구를 B라 한다.

Table 1. The execution result of tools

Model	Base Address	A	B
IPTIME N604R	0x8040000	6.25 min	0.15 sec
TP-LINK WR543GV2	0x80010000	296.33 min	9.89 sec
LINK DIR822	0x80000000	1261.23 min	4.78 sec

실험결과, 기존 연구의 경우 펌웨어의 크기에 비례하여 베이스 주소 선별에 소요되는 시간이 유의미하게 증가함을 알 수 있었다. 이는 다수의 베이스 주소 후보군 선별 실험을 통해 탐색의 범위를 이미지 내의 문자열을 최대한 많이 매칭시켜야만 정확도 높은 베이스 주소 후보군을 도출해낼 수 있다는 점으로 인해 분석대상 이미지 전체를 대상으로 문자열 수집을 수행하였기 때문이다. 기존 도구를 사용하여 베이스 주소를 선별하는데 걸리는 시간은 각각 6.25분, 296.33분, 1261.23분으로, 구현된 도구는 기존 도구의 실행시간 대비 평균 99.96% 속도향상이 있었음을 보인다. 이를 통해 기존 논문에서 제시한 방법에 대비하여 베이스 주소 선별에 걸리는 소요시간이 획기적으로 단축됨을 확인할 수 있다.

V. Conclusions

본 연구에서는 MIPS 아키텍처 기반 펌웨어를 대상으로 베이스 주소를 탐지하는 방법인 문자열 매칭을 통한 베이스 주소 탐지 자동화 기법의 후보군 선정 방식을 보완하기 위해 $gp 레지스터의 주소 값을 기반으로 한 페이지 입상도 단위의 펌웨어 베이스 주소 탐색 방법을 제안하였고, 도구를 개발 및 검증하였다. 본 논문에서 수행된 방법은 기존의 후보군 선정 방법에 비해 속도면에서 보다 우수함을 보였다. 이를 $gp 레지스터 값이 존재하는 펌웨어를 대상으로, 기존의 문자열 매칭을 통한 베이스 주소 탐지 방법에 적용시켜, 정확한 기준 주소 값을 획득할 수 있었으며, 이를 통해 펌웨어의 정적 분석 환경 구축을 수행할 수 있었다.

향후 다양한 아키텍처 기반의 임베디드 펌웨어를 대상으로 정적 분석 환경 구축을 위한 요소들을 판단하고, 이에 대한 추가적인 연구를 진행할 예정이다.

REFERENCES

[1] IoT Analytics, "State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally", https://iot-analytics.com/number-connected-iot-devices/, 2022

[2] SONICWALL, “Mid-Year Update: 2022 SonicWall Cyber Threat Report”, https://www.sonicwall.com/medialibrary/en/white-paper/mid-year-2022-cyber-threat-report.pdf, 2022

[3] Heffner C. Binwalk - Firmware Analysis Tool. https://github.com/ReFirmLabs/binwalk, 2022.

[4] Xiaodong Zhu, Yi Zhang, Liehui Jiang, Rui Chang, "Determining the base address of MIPS firmware based on absolute address statistics and string reference matching", Computers & Security, Volume 88, 2020. DOI: https://doi.org/10.1016/j.cose.2019.02.015

[5] S. Mun, D. Jang, "Automated extraction of MIPS firmware image base using page-granularity", The Korea Society of Computer and Information Winter Conference, p.5-6, Daejeon University, Korea, Vol 31, No 1., 2023.

[6] MIPS – MIPS Reference Guide. https://training.mips.com/basic_mips/PDF/Assemble _Language.pdf, 2018.

[7] ELF File Format – Executable and Linkable Format (ELF). https://www.cs.cmu.edu/afs/cs/academic/class/15213-f00/docs/elf.pdf

[8] PE File Format - PE Format https://learn.microsoft.com/en-us/windows/win32/debug/pe-format, 2022.

[9] MIPS ECOFF Object - machine-dependent things http://korea.gnu.org/manual/release/as/as-ko_8.html#SEC196

[10] Skochinsky, Igor. "Intro to embedded reverse engineering for PC reversers." REcon conference, Montreal, Canada. 2010.

[11] Kang, Ji-Hun and Ryu, Jae-cheol, “A Base Address Analysis Tool for Static Analysis of ARM Architecture-Based Binary,” Journal of the Korea Institute of Information Security & Cryptology, vol. 26, no. 5, pp. 1185-1189, Oct. 2016.

[12] R. Zhu, Y. Tan, Q. Zhang, Y. Li, and J. Zheng, “Determining image base of firmware for ARM devices by matching literal pools,” Digital Investigation, vol. 16. Elsevier BV, pp. 19-28, Mar-2016.

[13] CHEN, Daming D., et al. "Towards automated dynamic analysis for linux-based embedded firmware.", NDSS. p. 1.1-8.1. 2016. DOI: http://dx.doi.org/10.14722/ndss.2016.23415

Authors

Seok-Joo Mun received the B.S. degree in Computer Engineering from Yeungnam University, Korea, in 2016 and the M.S. degree in Information Security from KAIST, Korea in 2019.

Seok-Joo Mun joined the Research Institute of Industrial Technology at Yeungnam University, Gyeongbuk, Korea, in 2022. He is currently a researcher at the Institute of Industrial Technology, Yeungnam University. He is interested in system security, embedded systems, and kernel integrity.

Young-Ho Sohn received the B. Eng degree in Electronic Engineering from Kyungpook National University, Korea, in Feb. 1989, and the Ph.D in Electrical and Computer Eng. from Texas A&M University, USA, in May 2002.

He joined the Dept. of Computer Engineering at Yeungnam University since Mar. 2005. His research interests include IoT, Fiber Optic Networks, and Sensor System Securities.