Journal of Korean Institute of Intelligent Systems
Vol. 28, No. 5, October 2018, pp. 464–469
http://dx.doi.org/10.5391/JKIIS.2018.28.5.464

ISSN(Print) 1976-9172
ISSN(Online) 2288-2324

복제방지 칩을 이용한 임베디드 기반 보안 제어기

Embedded Security Controller with Copy Protection Chip

박재호* · Raimarius Delgado* · 이천호** · 최병욱*†
Jaeho Park, Raimarius Delgado, Cheonho Lee and Byoung Wook Choi†

*서울과학기술대학교 전기정보공학과, **젝스컴퍼니 주식회사
*Dept. of Electrical and Information Engineering, Seoul National University of Science and Technology
**JECS Company Incorporated

요 약

본 논문에서는 임베디드 제어 장치에 있어서 악의적 프로그램으로부터 시스템을 보호하기 위하여 복제방지 칩을 이용한 보안제어기를 제안한다. 보안 제어 방법은 사용자 프로그램과 복제방지 칩이 동일한 암호화 방식을 사용하는지 비교하며, 프로그램에서 복제방지 칩으로 데이터를 송신하면 복제방지 칩에서는 수신된 데이터를 암호화하여 반환해준다. 이후 프로그램이 전송하기 전의 데이터를 암호화해서 칩으로부터 반환받은 데이터와 비교해서 일치할 경우에만 프로그램이 정상적으로 실행되고 일치하지 않을 경우나 이러한 과정을 수행하지 않을 경우 프로그램은 종료된다. 또한 복제 방지 칩을 강제로 변경하거나 제거하는 하드웨어 해킹을 방지하기 위해 부팅 초기 단계인 부트로더에서 복제방지 칩이 정상적으로 연결되어 있고 칩이 변경되지 않은 경우에만 부팅과정을 진행한다. 실험에서는 부팅 시 복제방지 칩을 제거할 경우 부팅이 되지 않고, 운영체제가 인증과정이 실패하거나 인증과정을 수행하지 않는 프로그램의 실행을 차단할 수 있다는 것을 보여 준다. 이러한 결과를 바탕으로 임베디드 시스템에서 복제방지 칩을 이용한 보안제어기를 사용해서 시스템을 보호할 수 있음을 검증하였다.

키워드 : 임베디드 보안, 복제방지 칩, 안드로이드, 시스템 보안

Abstract

In this paper, we propose a security controller using a copy protection chip to protect system from malicious programs in the embedded controller. The security control method compares whether user program and the copy protection chip use the same encryption method. When the program transmits data to the copy protection chip, the copy protection chip encrypts and returns the received data. After that, the program encrypts the data before transmission and compares it with the data returned from the chip. If the data encrypted by the program matches the data returned by the chip, the program is executed. However, if the data does not match or the authentication process is not performed, the program is terminated. In addition, to prevent hardware hacking that forcedly changes or removes the copy protection chip, the boot process proceeds only if the copy protection chip is normally connected at boot time and the chip is not changed. The experiment shows that booting with no copy protection chip is not possible, and the operating system can block the execution of the program which does not perform or fails authentication process. Based on these results, we verified that the system can be protected by using the security controller using the copy protection chip in the embedded system.

Key Words : Embedded Security, Copy Protection Chip, Android, System Security

Received: Apr. 17, 2018
Revised: Oct. 11, 2018
Accepted: Oct. 13, 2018
†Corresponding authors
bwchoi@seoultech.ac.kr

1. 서 론

임베디드 시스템은 다양한 제품에 탑재되어 그 제품 안에서 특정한 작업을 수행하도록 하는 시스템을 말한다. 가전제품, 공장자동화 시스템, 엘리베이터, 휴대폰 등 대부분 장비에 임베디드 시스템이 탑재되어 있다. 다양한 분야에서 사용되는 임베디드 시스템은 사용자의 일상생활과 매우 밀접한 관계에 있으며 사용자의 개인정보 및 기업정보와 같은 중요한 정보가 임베디드 시스템 내에 저장되는 실정이다[1-3].

악의적 프로그램에 의해 정보들이 외부로 유출되거나 의료기기, 공장자동화 시스템과 같은 실시간 임베디드 시스템이 적용된 경우 정해진 시간 내에 작업이 수행되어야 하며[4] 시스템의 오작동이 생길 경우 사용자의 안전에도 위협을 끼칠 우려가 있다. 따라서 이러한 임베디드 시스템 내에서 정보보호와 시스템의 정상적인 운영을 위해 시스템의 부담을 줄이면서 동시에 높은 수준의

본 연구는 중소기업청에서 지원하는 2018년도 산학연협력 기술개발사업 (NO. C0541301) 의 연구수행으로 인한 결과물임을 밝힙니다.

보안을 유지해야 한다. 따라서 이러한 상황에 대처할 수 있는 보안 프로그램의 개발이 필요한 상황이다 [5-6]. 최근 다양한 플랫폼을 대상으로 하는 악성코드가 지속적으로 증가하고 있는 추세이다. McAfree 사의 보고서에 따르면 매 분기 모바일, PC 등 다양한 플랫폼에서 탐지된 악성코드 들이 증가하고 있다[7].

임베디드 기반의 다양한 플랫폼에서 악성 프로그램들에 대한 위협이 증대되고 있다. 이러한 상황에서 제한된 성능 및 자원을 가지는 임베디드 시스템에서는 시스템 자원을 많이 사용하는 보안기법을 적용하기 어렵고 기존의 서버나 데스크톱 PC와는 용도가 다르기 때문에 보안 요구사항 또한 이에 맞추어 재구성하여야 한다 [6, 8].

기존의 임베디드 시스템의 보안 솔루션으로는 서명 검증을 이용한 악성코드 차단 방식[6], grsecurity를 사용한 리눅스 커널 패치를 통한 리눅스 프로세스가 접근하는 자원을 제한하는 방식, LSM(Linux Security Module)을 이용한 방식[11], 시스템 콜을 후킹해서 접근제한 방식 등 소프트웨어적인 방법이 대부분이다. 서명 검증을 이용한 악성코드 차단 방식의 경우 서버를 통한 인증방식을 사용하기 때문에 서버구축에 추가적인 비용이 발생하게 되고 시스템이 서버 네트워크에 연결되어있지 않으면 시스템을 보호하지 못하게 되는 단점이 존재하고, grsecurity를 사용하는 경우에는 오버헤드가 크기 때문에 임베디드 시스템에서 사용하기 어렵다는 단점이 있으며 [9] 소프트웨어적인 보안 방식은 역공학을 통한 악성코드 삽입 및 변경에 취약하기 때문에 여러 개의 보안 기법을 적용해야 하는데 이렇게 되면 오버헤드가 커지고 프로그램의 크기가 증가하는 단점이 있다[12].

이러한 문제를 해결하기 위해 본 논문에서는 임베디드 시스템에서 쉽게 적용할 수 있으며 오버헤드를 최소한으로 줄이고 기존의 소프트웨어적인 방법만으로 제안된 보안 솔루션을 보완하기 위하여 복제방지 칩을 활용하는 방법이다. 모든 프로그램은 복제방지 칩을 통해서 인증을 수행하는 과정이 포함된다. 프로그램에 인증과정이 포함되어있지 않거나 인증이 실패한다면 해당 프로그램의 실행을 중지시킨다. 또한 하드웨어 개조를 통한 복제방지 칩 제거와 변경을 통한 하드웨어 해킹을 막기 위해 부팅 시 부트로더에서 복제방지 칩의 유무를 검사하고, 인증과정을 수행하여 성공하는 경우에만 정상적 부팅과정이 진행된다.

본 논문에서는 Boundary Devices 사의 i.MX6Q 프로세서 기반 NITROGEN6X 보드와 안드로이드 운영체제 4.4.3(Kitkat)를 사용하였고 복제방지 칩으로는 Neowine 사의 ALPU-C 칩을 사용하여 보안제어기를 구현하였다.

실험결과 인증과정을 수행하지 않거나 인증에 실패한 경우에 프로그램이 실행되지 않는 것을 확인하였고, 복제방지 칩을 제거한 경우 시스템이 부팅되지 않음을 확인하였다. 최종적으로 본 논문에서 제시된 방법에 의하여 임베디드 기반 플랫폼을 보호할 수 있음을 확인하였다.

2. ALPU-C 복제방지 칩

보안제어기 구현에 사용한 ALPU-C 복제방지 칩은 Neowine 사의 암호화 알고리즘이 적용된 칩으로 소형, 저 전력으로 구동하기 때문에 임베디드 시스템에 적합하다.

ALPU-C 칩은 I²C 통신방식을 사용하여 프로세서와 통신하며 인증과정을 수행한다. 칩의 내부에는 암호화에 사용되는 key 값을 내장하고 있으며 이 key 값이 다르면 같은 ALPU-C 칩이라고 하더라도 인증과정에 실패한다. 인증과정을 수행하는 모드에는 Bypass Mode, Feedback Encryption Mode, Hash Generator Mode로 총 3가지 모드가 있다. 3가지 모드 모두 8바이트의 데이터 교환을 통한 인증과정을 수행하게 된다. 또한 암호화 알고리즘을 여러 번 반복해서 적용할 횟수를 변경할 수 있으며, 이 횟수를 랜덤 값으로 지정해줄 경우 암호화 알고리즘 분석이 더욱 어려워지기 때문에 보안성이 더욱 높아지게 된다. 모드 설정과 알고리즘 반복적용 횟수는 프로세서에서 ALPU-C 칩으로 인증요청을 전송할 때 1바이트 크기의 I²C Sub Address로 설정해서 전송한다. Sub Address의 각 필드의 의미는 그림 1과 같으며, EE(Encryption Enable) 필드에서는 암호화 허용 여부를 결정하고, FE(Feedback Enable) 필드는 Feedback Mode로 동작 여부를 결정하고, HE(Hash Enable) 필드는 Hash Generation Mode로 동작 여부를 결정하고, EW[2:0](Encryption Width) 필드는 암호화 알고리즘을 반복해서 적용할 횟수를 지정할 수 있다. FE, HE 두 필드가 모두 0일 경우 Bypass Mode로 동작하게 된다.

본 논문에서는 EE 필드만 1로 지정하고 나머지 필드는 0으로 지정하여 I²C Sub Address를 0x80으로 지정하여 Bypass Mode로 테스트하였다.

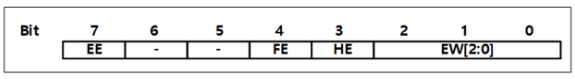


그림 1. I²C Sub 주소 설정

Fig. 1. I²C Sub Address Configuration

3. 보안제어기의 보안 유지 방법

3.1 부팅 제어

그림 2는 보안 제어기의 부팅 제어 과정을 나타낸다. 부팅 제어 과정은 U-BOOT 가 외부 RAM에 적재되고 실행될 때까지는 기존의 부팅과정과 동일하다. 그러나 본 방식에서는 U-BOOT가

하드웨어 초기화를 끝낸 후 커널을 로드하기 전에 시스템의 I²C bus를 확인해서 ALPU-C 칩의 유무를 검사하게 된다. ALPU-C 칩이 정상적으로 인식되어 있는 경우 ALPU-C 칩이 현재 시스템의 코드와 일치하는 검사하는 과정을 추가하여 칩의 변형을 확인한다. 이 과정에 실패할 경우 안드로이드 파일시스템으로 부팅이 진행되지 않을 뿐만 아니라 U-BOOT Command Line Interface으로도 진입하지 못하기 때문에 악의적 사용자로부터 ALPU-C 칩의 변경이나 제거와 같은 하드웨어 해킹으로부터 시스템을 안전하게 보호할 수 있다.

또한 이러한 부팅 제어 과정은 외부의 물리적 요인에 의해 ALPU-C 칩이 정상적으로 인식되지 않은 상태에서 모든 인증과정이 실패하게 된다. 즉, 어떠한 프로그램도 실행시키지 못하게 하는 상황이 발생할 수 있다. 따라서 부트로더에서 1차로 ALPU-C 칩을 검사하여 칩이 비정상적인 상태에서 부팅하지 못하도록 방지하며, 사용자에게 통보하여 조치를 취할 수 있도록 한다.

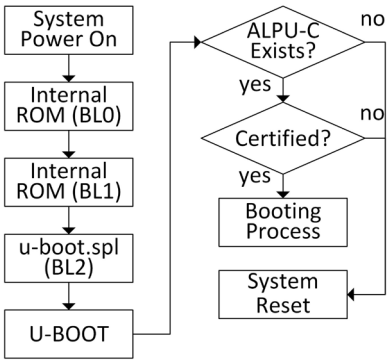


그림 2. 부팅 제어 과정

Fig. 2. boot control process

3.2 애플리케이션 실행 제어

그림 3 은 제안하는 보안 제어기의 애플리케이션 실행 제어 과정을 나타낸다.

애플리케이션 실행 제어는 운영체제에서 새로운 애플리케이션이 실행될 때 zygote에서 호출되는 fork 시스템 콜을 수행한다. 애플리케이션에 ALPU-C 칩과의 인증과정을 수행하는 라이브러리가 포함되어 있지 않으면 종료하게된다. 라이브러리를 포함하고 있다면 라이브러리에서는 먼저 ALPU-C 칩과의 인증과정을 수행한다. 성공 시 애플리케이션 본래의 기능이 실행되고, ALPU-C 칩이 정상적으로 인식되어 있지 않거나, ALPU-C의 key 값이 다를 경우 인증에 실패하여 애플리케이션을 종료시킨다.

이러한 제어를 통해 인증과정을 수행하지 않는 악의적 애플리케이션의 실행을 원천적으로 제한한다. ALPU-C 칩과의 인증과정을 통해 현재 하드웨어에 설치된 애플리케이션이 맞는지 검사하는 인증과정을 수행한다. 정상적인 애플리케이션에 역공학을 통하여 암호화하거나 악성코드를 삽입하여 다른 임베디드 시스템을 대상으로 유포된 애플리케이션의 경우에는 인증과정에서 ALPU-C 칩과 애플리케이션의 key 값이 일치하지 않는다. 따라서 인증 과정을 수행하는 라이브러리가 포함되어 있다고 해도 다른 시스템에서는 실행이 되지 않게 되어 보안을 유지할 수 있다.

애플리케이션이 실행될 때 한 번만 ALPU-C 칩과의 인증과정을 수행하도록 설계하였다. 따라서 기존의 소프트웨어적인 보안 기법에 비해 오버헤드가 적고 애플리케이션을 개발하는 입장에서는 인증을 수행하는 함수 한 줄만 추가하면 되기 때문에 새로운 애플리케이션을 개발할 때 쉽게 보안을 적용할 수 있다.

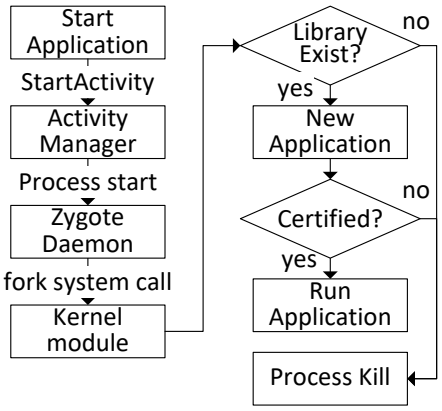


그림 3. 애플리케이션 실행 제어 과정

Fig. 3. application execution control process

4. 보안제어기 구현 및 실험

4.1 실험 환경

실험에 사용된 하드웨어 플랫폼은 i.MX6Q 프로세서와 1GB DDR3 RAM, 2MB Serial NOR FLASH, 3개의 I²C bus를 포함하는 Boundary Device NITROGEN6X 보드를 사용하였다. 복제방지 칩으로는 소형 저 전력 칩인 Neowine 사의 ALPU-C 칩을 사용하였다. 시스템 환경은 그림 4와 같다. 소프트웨어 플랫폼은 부트로더 U-BOOT 2015.07 버전, 커널 3.10.53 버전, 파일 시스템은 안드로이드 4.4.3(Kitkat) 버전을 사용하였다.

본 논문에서는 프로세서와 ALPU-C 칩의 인증과정이 올바르게 이루어졌는지 확인하기 위하여 ALPU-C 칩이 지원하는 3가지 동작 모드 중에서 인증과정 중에 데이터의 변화를 예측할 수 있는 Bypass Mode로 실험을 진행하였다.

Bypass Mode는 프로세서에서 ALPU-C 칩으로 8바이트의 데이터를 전송하면 ALPU-C 칩에서는 수신된 데이터를 1바이트 단위로 XOR 0x01 연산을 수행하여 결과를 프로세서로 반환하는 동작을 수행하는 모드이다.

그림 4. 실험환경
Fig. 4. Experimental environment

```
auto-detected panel 1280x720M@60
Display: hdmi:1280x720M@60 (1280x720)
In:    serial
Out:   serial
Err:   serial
Net:   Micrel ksz9021 at 6
FEC [PRIME], usb_ether
Error: usb_ether address not set.


i2c bus is 0, alpu not found

i2c bus is 1, alpu not found

i2c bus is 2, alpu found
ALPU WRITE :  11  22  33  44  55  66  77  ff
ALPU READ  :  10  23  32  45  54  67  76  fe
ALPU BYPASS Test success
Hit any key to stop autoboot:  0
=>
```

그림 5. 부팅 인증 성공
Fig. 5. Boot authentication successful

```
auto-detected panel 1280x720M@60
Display: hdmi:1280x720M@60 (1280x720)
In:    serial
Out:   serial
Err:   serial
Net:   Micrel ksz9021 at 6
FEC [PRIME], usb_ether
Error: usb_ether address not set.


i2c bus is 0, alpu not found

i2c bus is 1, alpu not found

i2c bus is 2, alpu not found

ALPU I2C WRITE ERROR

i2c bus is 0, alpu not found
```

그림 6. 부팅 인증 실패
Fig. 6. Boot authentication failure

4.2 부팅 제어 구현

U-BOOT 소스 코드의 main.c에서 U-BOOT의 command line으로 진행할지 부팅과정을 진행할지 결정하는 함수인 autoboot_command()를 호출하기 전에 ALPU-C 칩의 유무를 검사하고, 칩과의 Bypass Mode 인증을 수행하는 함수를 구현하였다.

칩의 유무를 검사하는 함수에서는 현재 U-BOOT에서 사용 가능한 모든 I²C bus를 확인하여 ALPU-C가 어떤 버스에 연결되어 있는지 확인한다. ALPU-C가 연결된 bus를 찾지 못할 경우에는 Bypass Mode 인증을 수행하는 함수를 실행하지 않고 ALPU-C 칩을 찾을 때까지 반복해서 I²C bus를 확인한다. ALPU-C 칩이 정상적으로 인식된 경우 칩과의 Bypass Mode 인증을 수행하는 함수가 실행된다.

Bypass Mode 인증을 수행하는 함수는 I²C 통신방식을 사용하여 Bypass Mode로 동작하도록 Sub Address를 설정하고 8바이트 데이터를 ALPU-C 칩으로 전송한다. ALPU-C 칩은 Bypass Mode 알고리즘을 적용하여 데이터를 프로세서로 반환한다. 프로세서에서는 ALPU-C 칩으로 전송하기 전의 데이터에 ALPU-C 칩에서 수행한 방식과 동일한 알고리즘을 적용하여 ALPU-C 칩으로부터 반환받은 데이터와 비교하는 인증과정을 수행한다.

인증과정이 성공적으로 진행되었을 경우에만 autoboot_command() 함수를 호출하여 이후의 부팅과정을 추가적으로 진행한다. 인증에 실패한 경우 부팅을 막기 위해 부트로더의 main.c 소스 코드를 수정하였다.

현재 실험환경에서는 I²C 버스가 총 3개가 존재한다. 따라서 부팅과정에서 3개의 I²C bus에서 ALPU-C 칩의 유무를 검사하였으며 그 결과 I²C Bus 2에 연결된 것을 확인하였다. ALPU-C칩으로 8바이트 데이터 "11 22 33 44 55 66 77 ff"를 ALPU-C 칩으로 전송하였다. ALPU-C 칩이 반환한 데이터가 전송한 데이터의 각 바이트에 XOR 0x01 연산을 수행한 결과인 "10 23 32 45 54 67 76 fe"인 것으로 보아 Bypass Mode에서 인증과정이 성공적으로 이루어진 것을 그림 5에서 확인할 수 있다. ALPU-C 칩을 찾는데 실패할 경우 ALPU-C 칩을 찾을 때까지 반복해서 I²C bus를 확인하는 것을 그림 6에서 확인하였다.

4.3 안드로이드 애플리케이션 실행 제어 구현

운영체제에서 보안을 유지하기 위해 새로운 애플리케이션이 실행되는 것을 감지하여 ALPU-C와의 인증과정을 수행하는지 검사한다. 시스템이 동작하고 있는 동안에는 계속 새로운 애플리케이션의 실행을 감지해야 하므로 커널 모듈로 구현하였다.

커널 모듈에서는 새로운 애플리케이션이 실행될 때 발생되는 fork 시스템 콜을 후킹 한다. 모듈이 커널에 올라가 있을 때, 새로운 애플리케이션이 실행되어 fork 시스템 콜이 호출되면 커널 모듈로 실행이 옮겨지게 된다. 커널 모듈에서는 수행하려 했던 본래의 fork 시스템 콜을 그대로 호출하여 실행되는 애플리케이션의 PID(Process ID)를 반환받는다. fork 시스템 콜이 호출된 원인이 사용자 애플리케이션 실행에 의한 것인지 확인하고 PID를 저장한다. 애플리케이션이 메모리에 적재가 완료돼서 실행준비가 되면 저장해둔 PID로 pid_task() 함수를 사용하여 커널이 프로세스를 관리하는 데 필요한 모든 정보를 가지고 있는 task_struct 구조체를 반환받는다. 이 구조체를 기반으로 그림 7의 과정에 따라 애플리케이션의 PID를 이용해서 프로세스의 메모리 맵에 포함된 파일 이름을 탐색한다. 메모리 맵에 인증과정을 수행하는 라이브러리를 찾게 된다. 탐색 결과 라이브러리가 존재할 경우 추가적으로 인증과정을 수행한다. 존재하지 않을 경우 애플리케이션은 커널 모듈에 의해 종료된다.

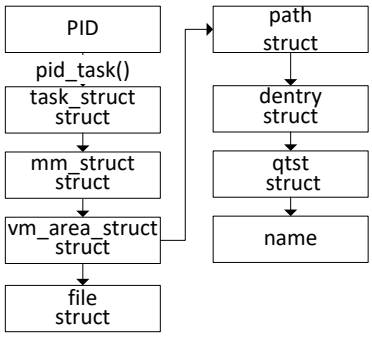


그림 7. 애플리케이션의 메모리맵에 포함된 파일 탐색 과정

Fig. 7. The process of searching for files contained in an application's memory map

그림 8. 애플리케이션 실행 실패

Fig. 8. Application execution failure

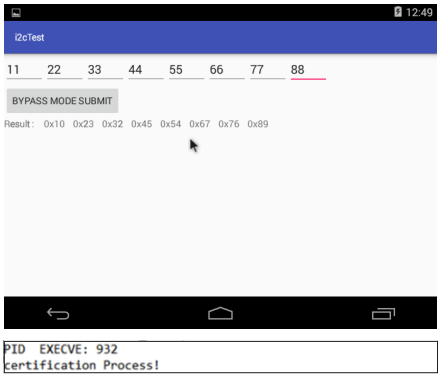


그림 9. 애플리케이션 실행 성공

Fig. 9. Application execution successful

애플리케이션의 인증과정에서는 ALPU-C 칩과 애플리케이션에 포함된 라이브러리의 key 값이 일치하여 인증 과정이 성공할 경우 애플리케이션이 정상적으로 실행된다. 실패할 경우 애플리케이션에 포함된 라이브러리에서 애플리케이션을 종료시킨다. 즉, 프로세스 메모리 맵에서 인증 라이브러리를 포함하고 인증과정에도 성공할 경우에만 애플리케이션이 정상적으로 동작하게 된다. 그림 8은 안드로이드 애플리케이션에 인증 라이브러리가 포함되어 있지 않은 기본적인 helloworld 애플리케이션을 수행했을 때 커널 모듈에서 helloworld 애플리케이션을 종료시킨 것을 확인할 수 있다. 애플리케이션이 종료가 되며 터미널에 인증과정을 수행하는 라이브러리가 포함되지 않은 프로세스의 PID와 프로세스 종료 사유가 표시되도록 하였다.

그림 9는 안드로이드 애플리케이션에서 인증과정을 수행하는 라이브러리를 포함하고 Bypass Mode 인증과정을 수행하도록 한 결과이다. 인증과정에서 "11 22 33 44 55 66 77 88"을 ALPU-C 칩으로 전송하였고 "10 23 32 45 54 67 76 89"를 반환받았다. 전송한 데이터에 XOR 0x01을 수행한 값이 예상 결과이므로 ALPU-C 칩에서 반환해준 결과가 예상 결과와 일치한 것을 확인할 수 있었다. 터미널에서는 인증과정을 모두 성공적으로 끝낸 프로세스의 PID와 성공한 결과를 확인할 수 있다.

5. 결론

본 논문에서는 임베디드 시스템에서 기존의 소프트웨어적인 보안 기법의 단점을 보완하기 위해 ALPU-C 복제방지 칩을 이용하여 하드웨어적인 방법을 사용하여 악의적 프로그램으로부터 시스템을 보호하기 위한 방법을 제안하였다. 실험 결과 하드웨어 보안 칩을 통하여 인증과정을 수행하지 않거나 실패할 경우 프로그램들이 수행되지 않는 것을 확인할 수 있었고, 보안 칩을 제거하거나 변경하는 하드웨어 해킹이 발생한 경우 부팅이 불가능한 것을 확인하였다. 따라서 이러한 방법이 기존의 소프트웨어 보안 방식의 단점을 보완하는 데 도움이 될 것으로 기대한다.

References

[1] Jong Hyuk Park. "A Study on Embedded Operating System Security Technology for Ubiquitous Computing." *Journal of Korea Multimedia Society*, 13.8 (2010.08): 1194-1201.

[2] Sung-Won Lee, Seung-Min Park, Kwee-Bo Sim. "Smart Door Lock Systems using encryption technology." *Journal of Korean Institute of Intelligent Systems*, 27.1 (2017.2): 65-71.

[3] Bomin Choi, Jong-Hwan Kong, Myung-Mook Han. "The Model of Network Packet Analysis based on Big Data." *Journal of Korean Institute of Intelligent Systems*, 23.5 (2013.10): 392-399.

[4] Gil-Jin Yang, Byoung-Wook Choi. "Joint Space Trajectory Planning on RTOS." *Journal of Korean Institute of Intelligent Systems*, 24.1 (2014.2): 52-57.

[5] Y.J. Jung, D.H Lim, Y.B Seo, J.M.Kim "The Trends of Embedded Operating System Security Technology." *Analysis of Electronic Communication Trends Volume 23*, Issue 1 February 2008

[6] JongSeok Lee, Ki Young Jung, Daniel Jung, Tae-Hyung Kim, Yuna Kim, Jong Kim. "Preventing ELF(Executable and Linking Format)-File-Infecting Malware using Signature Verification for Embedded Linux." *Journal of KIISE : Computing Practices and Letters*, 14.6 (2008.8): 589-593.

[7] McAfee Labs, "Threats Report", March 2018

[8] B.J. Kang, J.S. Han, E.G. Lim "Communications of the Korean Institute of Information Scientists and Engineers, 30.1 (2012.1): 44-53.

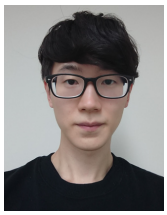
[9] Chan-Hee Lee, Yeong-Ung Park, Ji-Hyeog Lim, Hong-Geun Kim, Choong-Hyun Lee, Jaesoo Yang, Seong-Je Cho. "Access Control Mechanism Preventing Application Piracy on the Android Platform." *Journal of KIISE : Computing Practices and Letters*, 18.10 (2012.10): 692-700.

[10] Sung Soo Kim, Choong Seon Hong. "Prevention of personal Information leakage through system call hooking in Android Kernel." *Proceedings of the Korean Information Science Society*, (2014.6): 1015-1017.

[11] Morris, James, Stephen Smalley, and Greg Kroah-Hartman. "Linux security modules: General security support for the linux kernel." *USENIX Security Symposium*. 2002.

[12] Dongsoo Ha, Kanghyo Lee, Heekuck Oh. "Android Application Reverse Engineering Protection Techniques" *REVIEW OF KIISC*, 25.3 (2015.6): 19-28

저 자 소 개



박재호 (Park Jae Ho)

2018년 : 대전대학교 IT 전자공학과 공학사
2018년~현재 : 서울과학기술대학교 전기정보공학과 석사과정

관심분야 : Real-time Systems, Embedded Linux, Android, Embedded Security
Phone : ※ 개인정보 표시제한
E-mail : jaeho@seoultech.ac.kr

Raimarius Delgado

2014, 2016년 : 서울과학기술대학교 전기정보공학과 공학사, 공학석사
2016년~현재 : 서울과학기술대학교 전기정보공학과 박사과정

관심분야 : Real-time Systems, Embedded Linux, Real-time Streaming Protocol
Phone : ※ 개인정보 표시제한
E-mail : raim223@seoultech.ac.kr

이천호 (Lee Cheon ho)

1996년 : 서울과학기술대학교 기계설계학과 공학사
1996년~2006년 : 가산전자 외(기술지원, 마케팅)
2006년~2012년 : ㈜싱커스텍 부장(기술기획, 영업)
2013년~현재 : 산업용컴퓨터 및 임베디드 보드 전문 젝스컴퍼니(주) 대표이사

관심분야 : Industrial-PC, IOT, Andriod, Linux
Phone : ※ 개인정보 표시제한
E-mail : joel@jecs.co.kr

최병욱(Byoung Wook Choi)

1986년 : 한국항공대학교 항공전자공학공학사
1988년, 1992년 : 한국과학기술원 전기 · 전자공학 공학석사, 공학박사
1988년~2000년 : LG산전 중앙연구소, 책임연구원
2000년~2005년 : 선문대학교 제어계측공학과 부교수
2003년~2005년 : 임베디드웹 대표이사
2007년~2008년 : Nanyang Technological University, Senior Fellow
2005년~현재 : 서울과학기술대학교 전기정보공학과 교수

관심분야 : Real-time embedded systems design, Embedded Linux, Software Platform
Phone : ※ 개인정보 표시제한
E-mail : bwchoi@seoultech.ac.kr