IoT 보안 인증 제도 기반 홈 IoT 기기 애플리케이션의 취약점 분석 및 대응책 제시*

윤혜진*, 김은주*, 최지원*, 위한샘**, 이옥연*

*국민대학교 정보보안암호수학과(학부생, 교수)

**국민대학교 금융정보보안학과(대학원생)

Vulnerability Analysis of Home IoT Device Applications and Suggestion of Countermeasures Based on IoT Security Certification

Hyejin Yoon*, Eunju Kim*, Jiwon Choi*, Hansaem Wi**, Okyeon Yi*

*Dept. Information Security, Cryptology, Mathematics, Kookmin University(Undergraduate, Professor)

**Dept. Financial Information Security, Kookmin University(Graduate)

요 약

최근 IoT 기술의 발전과 편리성으로 인해 홈 IoT 기기의 수요가 늘면서 다양한 홈 IoT 기기가 보급되고 있다. 홈 IoT 기기는 가장 안전해야 할 사용자의 보금자리에 사용되므로 사용자의 개인정보와 자산을 필수적으로 보호해야 한다. 하지만 대부분의 홈 IoT 기기 애플리케이션의 보안 수준이 낮아 이를 겨냥한 해킹 범죄들이 증가하고 있는 추세이다. 본 연구에서는 홈 IoT 기기 애플리케이션에 대해 NOX 프로그램으로 추출한 APK를 MITMPROXY, SUPER툴을 이용해 각각의 취약성을 분석하여 발생 가능한 위협을 식별한다. 또한 KISA의 'IoT 보안 시험·인증 기준 해설서'에 따른 인증 기준 부합 여부를 평가하고 이에 따른 적절한 보안 대응책을 제시하여 인증 제도에 부합하는 안전한 홈 IoT 기기 애플리케이션(이하 앱.)의 증가를 기대한다.

I. 서론

사물인터넷(Internet of Things, IoT)은 유·무선 네트워크를 기반으로 모든 사물을 연결하여 사람과 사물, 사물과 사물 간의 통신을 제공하는 기술을 의미한다. IoT는 산업 분야에서 자동화 시스템을 구축함으로써 기업, 국방, 의료 분야 등에 편리성을 제공한다. 그뿐만 아니라, 건강 관련 서비스, 홈 네트워크 등에도 적용되어 사용자 삶의 질이 향상된다. 이러한 편리성으로 인해 최근 전 세계적으로 사용자들을 위한 IoT 적용 제품 및 서비스가 증가하고 가정 내에 적용되는 스마트 홈 IoT 기술도 발전하고 있다. 스마트 홈 IoT 기술은 리모컨, 도어락, 제어 센서 등 다양한 기기에 IoT 기술을 융합하여 원격으로 제어할 수 있어 사용자에게 편리성을 제공한다.

하지만 이러한 홈 IoT 기기를 활용해 최근 다양하고 고도화된 해킹 사례가 늘고 있다. 경찰청 통계에 따르면 2020년 9월까지의 누적 해킹 범죄는 2019년 대비 21.7% 증가하였다. 하지만 검거율은 17.6%로 감소하여 IoT 해킹 범죄가 점점 지능화되고 있음을 알 수 있다.[1]

따라서 본 연구는 특정 IoT 기기와 연동하는 앱 A를 실험 대상으로 다양한 방식의 취약점 분석을 진행하며 이에 대해 보안 인증 제도를 기반으로 한 보안 대응책을 제시한다.

* 이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2020-0-00085, 5G+ 기반 6G 이동통신 정보보안 기술 연구)

II. 취약점 분석

2.1 분석 앱 및 분석 도구 소개

앱 A는 로그인, 연동 가능한 IoT 기기 작동, 감사 등 다양한 기능들을 수행한다.

분석에 사용한 도구는 <표 1>과 같다.

Experiment Environment		
Device / OS	App Excution	· iPhone XS / ios 14
	Analysis Tool Excution	· Intel / Window
Analysis Tool	Network Analysis	· MITMPROXY
	APK file Analysis	· Nox App Player · Decompiler.com · SUPER

<표 1> 분석 도구

본 논문에서 연구를 통해 발견한 취약점은 영어 소문자(a, b)로 인덱스 화하여 설명한다.

2.2 MITMPROXY[2]를 이용한 취약점 분석

IoT 앱이 안전하게 데이터를 전달하는지를 살펴보기 위해 트래픽을 추적하여 MITM 공격이 가능하도록 도와주는 네트워크 분석 툴인 MITMPROXY를 이용하여 MITM(Man In The Middle) 공격을 수행하였다.

2.2.1 로그인 취약점

MITMPROXY 프로그램을 사용하여 앱 A를 iPhone XS에서 분석하였다. 앱 A에서 아이디와 비밀번호를 입력하는 과정을 MITMPROXY 프로그램으로 살펴보면, <그림 1>과 같이 아이디와 해싱된 비밀번호를 확인할 수 있다. <그림 1>은 JSON의 Request의 일부이다.

<그림 1> 로그인 과정에서의 사용자 아이디 및 비밀번호

위 <그림 1>을 보면 비밀번호가 해싱되어 있어 안전하다고 생각할 수 있다. 하지만 앱 A의 비밀번호 설정 특성상, 영문자와 숫자만을 이용하여 6-8자리의 비밀번호를 만들어내기 때문에 최대 62^8(약 2^{48})의 계산량으로 전수조사가 가능하다. 이는 KISA에서 권고하는 비밀번호 보안 강도[3]인 2^{112} 이상에 맞지 않고 취약점이 발견된 DES의 계산량인 2^{56}보다 적은 계산량으로 비밀번호를 알아낼 수 있다(취약점 a). 복호화를 하기 위해 앱 A를 설치해 APK를 추출한 후 JAVA 코드를 분석한 결과는 다음과 같다.

가. LoginActivity.java파일에서 비밀번호를 입력받은 후 UtilHandler.Sha256Encoding을 사용하여 저장한다.

나. Sha256Encoding함수는 비밀번호와 비밀번호 길이를 Sha256의 입력값으로 사용하여 출력값을 JSON 파일에 저장한다.

비밀번호의 Sha256Encoding 값을 알아내기 위해 전수조사를 수행하였다. <그림 2>는 실제 APK 파일 속 UtilHandle.java의 Sha256Encoding 함수이다. <그림 3>은 위의 함수를 이용하여 62^8(약 2^{48})가지 비밀번호의 해시값을 저장하는 테이블을 생성하는 코드 일부이다. <그림 3> 코드를 실행하여 만들어진 전수조사 테이블의 일부인 <그림 4>에서 <그림 1>의 해시값과 같은 값을 가지는 것을 볼 수 있다. 이때 <그림 4>에서 해시값과 함께 기록된 사용자의 비밀번호를 알아낼 수 있었다.

```
import java.security.MessageDigest;

public class UtilHandler {
    public static String Sha256Encoding(String str) {
        try {
            byte[] digest = MessageDigest.getInstance("SHA-256").digest((str + str.length()));
            StringBuffer stringBuffer = new StringBuffer();
```

<그림 2> UtilHandle.java의 Sha256Encoding 함수

```
for(n8 = 0; n8 < 62; n8++) {
    password.append(charArr[n1]);   password.append(charArr[n2]);   password.append(charArr[n3]);
    password.append(charArr[n4]);   password.append(charArr[n5]);   password.append(charArr[n6]);
    password.append(charArr[n7]);   password.append(charArr[n8]);
    sha = UtilHandler.Sha256Encoding(password.toString());
    password.append(" ");   password.append(sha);   password.append("\n");
```

<그림 3> 해시값 생성 전수조사 소스 코드 일부

```
18gkrqjr ab2c603b8c3c059ce63c7fe63fdb9234a2f59bd0a0994faca0642860a9e2abfb
18gkrqjs 97e9d637fb8589b1785cbf5f2efcb6f68f0c21bd8100406eec8d8d391449d27b
18gkrqjt 77c77b3e8ad79c139dad21521940f3d04eba6f3c1889eb9d7f5d1c5fe5ab2ed7
18gkrqju cf903c38f36ed1342cfb43fa6298704cdac300921da5b1d5e90733af48a66561
```

<그림 4> 해시값 테이블 일부

2.2.2 로그인 소스 코드 취약점

위의 2.2.1 연구에서 또 다른 앱 A의 로그인 과정 취약점을 발견할 수 있다. 위 연구에서 짧은 비밀번호의 취약점을 이용해 전수조사를 통해 충돌 쌍을 발견했다. 이때 전수조사가 성공할 수 있었던 배경은 APK로 추출한 JAVA 코드에서 비밀번호 해시 입력값과 함수를 알 수 있었기 때문이다. 해시 함수 이름과 입력값 정보가 난독화가 되지 않으면 공격자가 이를 가지고 공격이 가능하다(취약점 b).

2.3 SUPER[3]를 이용한 취약점 분석

SUPER(SUPER Android Analyzer)는 안드로이드 앱 취약점을 윈도우 등의 환경에서 APK 파일을 사용해 분석하는 툴이다.

앱 A의 APK 파일을 추출하여 SUPER를 이

용해 취약점을 분석해 본 결과, 총 1,738개의 소스 코드 취약점이 발견되었다. SUPER에서 제공하는 취약점 정보는 5가지의 취약 레벨로 나뉘어 제공되는데 앱 A의 경우 높은 취약성 순서대로 Critical 0건, High 3건, Medium 1건, Low 1,660건, Warnings 74건이 발견되었다. <표 2>는 SUPER를 통해 발견한 1,738개의 소스 코드 취약점을 유형 별로 묶은 총 23가지 종류의 취약점 내용을 설명한다.

High	Weak Algorithms	취약 알고리즘을 사용한다.
Medium	Allows Backup	adb을 사용해 애플리케이션 데이터 백업을 허용하여 앱의 개인 데이터를 PC로 가져올 수 있다.
Low	Generic Exception in Throws	메소드를 구체적으로 제시하지 않고 예외 처리를 한다.
	Generic Exception in catch	메소드를 구체적으로 제시하지 않고 예외 포착을 한다.
	Math Random method	충분히 랜덤하지 않은 함수를 사용한다.
	Sleep Method	sleep 메서드에 변수와 함께 인수로 사용되므로 앱이 무기한 중지될 수 있다.
	Unchecked output in Logs	민감한 정보를 기록한다.
	Unknown permission	응용프로그램이 자체 권한을 만들 수 있더라도 거부되어야 한다.
Warnings	Access fine location permission	사용자의 위치를 확인할 수 있다.
	Base64 Encode	안전하지 않은 데이터 인코딩 방식인 Base64를 사용한다.
	Base64 decode	안전하지 않은 데이터 디코딩 방식인 Base64를 사용한다.
	Call phone permission	사용자의 개입 없이 전화 번호를 호출할 수 있으므로 권한을 확인해야 한다.
	Certificate or Keystore disclosure	소스 코드의 분석을 통해 하드코딩된 인증서 또는 키 저장소가 노출될 수 있다.
	Exported activity	내보낸 활동이 발견되고 다른 사용자가 사용할 수 있다.
	Exported receiver	내보낸 수신자가 발견되고 다른 사용자가 사용할 수 있다.
	Exported service	내보낸 서비스가 발견되고 다른 사람이 사용할 수 있다.
	Get SIM OperatorName	응용 프로그램에서 장치 네트워크 운영자 이름을 기록하고 프로세스는 사용자 모르게 수행될 수 있다.
	IP Disclosure	소스 코드의 분해를 통해 개인 IP가 노출될 수 있다.
	Internet permission	네트워크 소켓을 만들고 사용자 지정 네트워크 프로토콜을 사용할 수 있다.
	Read external storage permission	SD 카드의 내용을 읽을 수 있다.
	Read phone state permission	장치의 전화 기능에 액세스할 수 있다.
	URL Disclosure	소스 코드의 분해를 통해 개인 URL이 노출될 수 있다.
	Write external storage permission	SD 카드에 쓸 수 있다.

<표 2> SUPER를 통해 발견한 취약점

<표 2>에서 High 레벨 취약점인 “Weak Algorithm”을 자세히 분석해본 결과, 앱 A가 “MD5”으로 계산한 메시지를 앱에서 사용한다는 점을 알게 되었다. MD5는 128비트 해시 함수로 충돌 쌍이 발견되는 취약점이 제시되어 사용하면 안 되는 해시 함수이다. 하지만 앱 A는 MD5를 사용하기 때문에 SUPER에 의해 High 레벨 취약점이 발견된 것이다.

III. IoT 보안 인증 제도[4] 기반 비교 분석 및 대응책 제시

현재 출시되는 IoT 제품들은 KISA에서 제공하는 IoT 보안 시험·인증 제도를 기준으로 보안 요구 사항에 부합하는지 시험을 시행한다.

KISA의 IoT 보안 시험·인증 기준 해설서는 IoT 보안 인증을 위한 보안 인증 수행 과정의 주요 절차 및 세부 활동 사항과 해당 기준 내용을 포함한다. IoT 제품(모듈 및 앱 포함)의 보안 요구사항을 기반으로 LITE, BASIC, STANDARD 3개의 등급으로 분류되며 본 연구에서는 앞서 발견한 IoT 앱의 취약점을 STANDARD 등급 보안 요구사항을 기반으로 비교·분석한다.

3.1 앱 A의 취약점 및 비교 분석

본 절은 2장의 연구 결과인 2.2절에서의 취약점 a, b와 2.3절에서 발견한 취약점 <표 2>를 IoT 보안 요구 사항에 따라 비교 분석한다.

a. 앱 A의 비밀번호가 전수조사할 수 있다는 것으로 AU1-8**에 부합하지 않는다.

b. 함수 이름, 인수로 함수의 기능을 쉽게 알 수 있는 것으로 PL1-3***에 부합하지 않는다.

SUPER로 발견한 취약점을 IoT 보안 요구사항에 비교·분석한 결과는 <표 3>과 같다.

보안 인증 기준	오류 이름
AU1-5 제품에서 사용되는 사용자 계정 및 권한에 대한 관리 기능을 제공해야 한다.	· Allows Backup · Unknown permission · Access fine location permission · Call phone permission · Internet permission · Read phone state permission
CR1-1 중요정보 전송 또는 저장 시 안전한 암호 알고리즘을 사용해야 한다.	· Weak Algorithm · Base64 Encoding · Base64 decoding
CR3-1 난수 생성 시 난수성이 검증된 알고리즘을 이용해야 한다.	· Math Random method

** 길이, 주기, 복잡성을 고려하여 안전한 비밀번호로 설정되도록 해야 한다.
*** 소스 코드 분석 방지를 위해 난독화를 적용해야 한다.

DP1-1 제품 간 전송되는 중요 정보는 암호화되어야 한다.	· Unchecked output in Logs · Exported activity · Exported receiver · Exported service
PL1-3 소스 코드 분석 방지를 위해 난독화를 적용해야 한다.	· Certificate or Keystore disclosure · IP Disclosure · URL Disclosure
PH1-1 외부 인터페이스는 활성화하되, 필요 시 접근통제 기능을 지원해야 한다.	· Read external storage permission · Read phone state permission · Write external storage permission

<표 3> 보안 인증 항목 기반 취약점 분류표

3.2 대응책

본 절에서는 3.1절에서 언급한 인증 제도의 항목에 따른 대응책을 제시한다.

가. AU1-5 : 제품에서 사용할 기능에 대한 최소한의 권한만 부여해야 한다.

나. AU1-8 : 개인정보보호위원회에서 제시한 안전한 비밀번호 작성 규칙[5]에 따르면 영문 대소문자, 숫자, 특수문자 중 2종류를 선택하면 최소 10자리 이상이어야 하고, 3종류를 선택할 땐 최소 8자리 이상이어야 한다.

다. CR1-1 : Base64는 이진 데이터를 ASCII 문자열로 변환하는 알고리즘으로 한 번 수행하는 것은 안전하지 않다. Base64 인코딩 후 안전한 해시 알고리즘으로 한 번 더 해싱해야 한다.

라. CR3-1 : FIPS 140-2 인증을 받은 암호 모듈의 난수 생성기와 특정 함수의 출력값이나 IV를 seed로 사용하여 난수를 생성해야 한다. 예를 들어, KISA에서 제공하는 난수 발생기 알고리즘[6]은 CTR-DRBG(SEED/ARIA)이다.

마. DP1-1 : 공격자가 중간에서 가로챌 수 있는 중요정보는 AES와 같은 안전한 암호 알고리즘으로 암호화되어야 한다.

바. PL1-3 : 변수와 함수 이름을 의미 없이 짓거나 코드의 흐름을 알 수 없게 해야 한다.

사. PH1-1 : 제품에서 접근할 인터페이스에 대한 최소한의 권한만 부여해야 한다.

위의 대응책 이외에도 소스 코드 구현 시 sleep과 같은 함수엔 변수를 인수로 사용해선 안 되며 예외 처리를 명확히 해야 한다.

IV. 결론

본 논문은 NOX 프로그램으로 추출한 APK를 MITMPROXY, SUPER 툴을 이용해 홈 IoT 기기 앱에 대한 취약점 분석하였다. 또한, 분석한 취약점이 보안 인증 기준에 부합하는지 평가하고 이를 바탕으로 인증 항목에 대한 대응책을 제시하였다. 분석한 결과, 앱 전송 데이터인 사용자 아이디와 비밀번호를 쉽게 알아낼 수 있었고 소스 코드의 취약점을 발견할 수 있었다. 2.2절에서 분량상 넣지 않았지만, 다른 2개의 홈 IoT 앱을 같은 과정으로 분석해 본 결과에서도 취약점 a가 발견되었다. 이로 알 수 있었던 점은 IoT 보안 요구 사항이 있음에도 불구하고, 이를 만족하는 IoT 앱이 많지 않다는 점이다.

3.2절에서 제시한 총 1,740개의 취약점에 대해 권한 부여를 제외한 암호 알고리즘, 난독화, 소스 코드 예외 처리 및 함수 인수의 설정을 수행한다면 취약점의 1,731개가 해결될 것으로 예상한다. 이는 약 99% 정도의 효과를 내는 것으로 기존의 IoT 기기 앱이 3.2의 대응책을 적용한 보안 인증 제도를 통해 안전한 홈 IoT 환경을 구성하는 데에 기여할 것이라 기대한다.

[참고문헌]

[1] 전유진. “늘어나는 IoT 기기, “해킹 공포 여전해””. 『CCTVNEWS』. 2021년 1월 8일. https://www.cctvnews.co.kr/news/articleView.html?idxno=217963.

[2] “mitmproxy”.『mitmproxy』. https://mitmproxy.org.

[3] “암호 알고리즘 및 키 길이 이용 안내서”.『KISA』. 2018년 12월

[4] “SUPER Android Analyzer”. 『SUPER Android Analyzer』. https://superanalyzer.rocks.

[5] “사물인터넷(IoT) 보안 시험·인증 제도 해설서”.『KISA』. 2019년 2월

[6] “안전한 비밀번호 사용 규칙”.『개인정보보호위원회』. https://www.privacy.go.kr/a3sc/per/chk/examInfoViewCQ4.do.

[7] “모바일환경에 적합한 암호알고리즘 소스 코드”.『KISA』. https://seed.kisa.or.kr/kisa/Board/23/detailView.do.