사물인터넷 소형 스마트 홈·가전 보안 가이드 [기업용]

2016. 12.

미래창조과학부
Ministry of Science, ICT and Future Planning

KISA 한국인터넷진흥원
KOREA INTERNET & SECURITY AGENCY

목 차

제1장
개요

제1장 개요

1. 필요성 및 목적

정보통신기술의 급속한 발전으로 사람, 사물, 공간, 데이터 등이 연결되는 초연결사회가 도래하게 되면서, 사물인터넷이 주요 이슈로 부각되었다. 사물인터넷이란 사람과 사람들이 살고 있는 공간, 그리고 그 안에 있는 사물들이 인터넷에 서로 연결된 것을 의미한다. 이는 정보를 생성, 수집, 공유, 가공하여 다양한 서비스를 가능하게 해준다.

이러한 사물인터넷은 기업 생산성과 효율성 제고, 각종 사회적 이슈 해결 및 인간 개인의 삶의 질을 향상시켜 줄 것이다. 하지만 기존의 사이버 공간에서만 발생했던 보안 위협들도 우리 생활과 밀접하게 연결되어 있는 현실세계로 들어옴으로써, 직접적으로 영향을 미치는 부정적 측면도 존재하게 되었다.

특히 사물인터넷 환경 중에서도, 최근 보급이 확산되고 있는 스마트 홈 분야는 가정 내 스마트 디바이스들을 유·무선 네트워크로 연결하여 자동화, 원격 제어, 에너지 관리 등을 통해 이용자에게 편의성을 제공하고 있다. 이러한 스마트 홈·가전이 존재하는 한 개인의 프라이버시 문제와 직접적인 연관이 있을 수밖에 없다. 스마트 홈에서 수집되는 데이터는 사람과 사물간의 데이터 교환을 기반으로 하고 있어 개인정보(이름, 생년월일, 전화번호, 주소 등), 개인 영상 정보 등 사생활에 밀접한 정보가 포함되므로 프라이버시 침해 위험성이 높다. 또한 스마트 도어락 해킹을 통해 무단침입을 하거나 스마트 홈 컨트롤러를 장악하여 타인의 스마트 홈·가전 조작, 전기 요금 과다 청구 등 파급도가 큰 사고가 발생할 수도 있다. 이러한 스마트 홈 침해 사고는 금전적, 물리적, 정신적인 피해를 유발하므로 무엇보다 사전 예방하는 것이 중요하다.

침해 사고를 미리 예방하기 위해서는 이용자의 보안 인식 제고도 중요하지만, 스마트 홈 제조사에서 예방, 조치해야하는 부분이 존재한다. 스마트 홈·가전을 납품하기 전 사전 점검 및 사후 대응책을 마련하여 보안 내재화를 실천해야 한다.

이에 본 가이드에서는 IP 카메라, 홈 컨트롤러, 도어락 위주의 소형 스마트 홈·가전에 대한 보안 위협과 이를 효과적으로 예방하기 위한 보안 항목을 제공한다. 이를 통해 사고 예방 및 피해를 최소화하고자 하며, 향후 대형 스마트 홈·가전에 대한 가이드도 배포될 예정이다.

2. 가이드 목적 및 구성

목적	- 스마트 홈·가전 보안 위협 실태 - 스마트 홈·가전 보안 내재화
대상	- 스마트 홈·가전 제조사용
범위	- 소형 스마트 홈·가전
구성	- [1장] 개요 1.1 필요성 및 목적 1.2 가이드 목적 및 구성 - [2장] 스마트 홈·가전 보안 위협 2.1 웹 인터페이스, 인증/허가 보안 위협 2.2 네트워크 서비스 보안 위협 2.3 모바일 앱 보안 위협 2.4 펌웨어/업데이트/물리적 보안 위협 - [3장] 스마트 홈·가전 보안 대응 방안 3.1 웹 인터페이스 보안 3.2 인증/허가 보안 3.3 네트워크 서비스 보안 3.4 모바일 앱 보안 3.5 펌웨어 보안 3.6 업데이트 보안 3.7 물리적 보안 - [4장] 스마트 홈·가전 보안 대응 방안 해설서

제2장

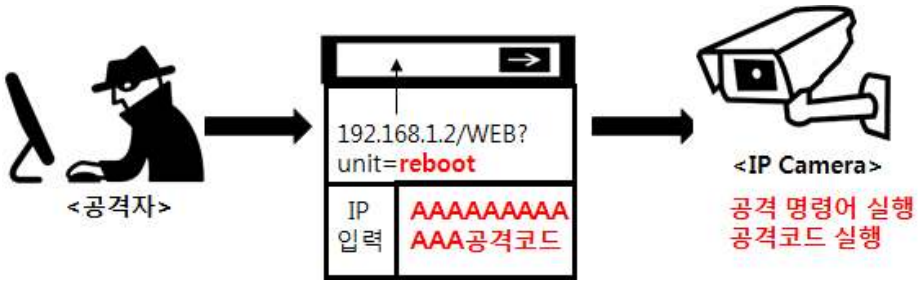
스마트 홈·가전 보안 위협

제2장 스마트 홈·가전 보안 위협

■ 웹 인터페이스, 인증/허가 보안 위협

최근 IoT 기기 대상으로 미라이(Mirai)를 이용한 대규모 DDoS 공격이 발생하여 1200여개 미국 주요기관 사이트를 마비시키는 사건이 발생했다. 실제로 약 12만대 IoT 기기(IP 카메라, NAS 등)가 공격에 악용됐다. 미라이 악성코드는 공장 출하 시 설정된 기본 ID/PW를 방치한 IoT 기기를 대상으로 기본 계정을 삽입하여 관리자 권한을 획득, 감염시키는 방식을 사용했다.

또한 인터넷에 연결된 IP 카메라가 전 세계로 생중계 되는 사례가 있었다. IP 카메라 관리페이지 접속 시 접근 가능한 사용자를 확인하지 않거나, 기본 ID/PW를 이용해 로그인이 가능한 점을 악용한 침해사고 사례였다. 이처럼 보안에 취약한 IoT 기기를 대상으로 하는 악성코드 감염시도가 증가하고 있어 이에 대한 보안대책 마련이 시급하다.



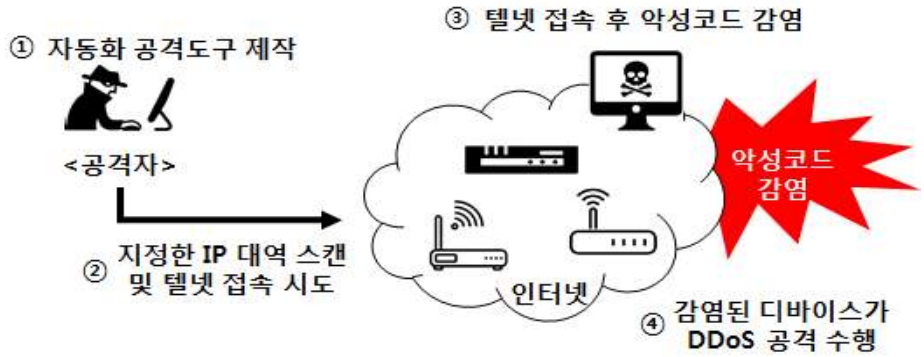
<그림 2-1> 명령 실행, 오버플로우 취약점을 악용한 공격 사례

명령실행, 버퍼오버플로우(BoF) 취약점은 파라미터 등의 공격 가능한 인자 값 검증 미흡으로 인해 발생하는 것이다. 최근 KISA 신규 취약점 신고포상제로 가장 많이 접수되고 있는 IoT 취약점이다. 구체적으로 사용자 입력 값, URL 파라미터 데이터를 확인하지 않고 시스템 권한으로 실행하거나, 데이터 길이 값을 확인하지 않아서 발생한다. 공격자는 이를 악용하여 원하는 시스템 명령, 임의코드 실행을 수행할 수 있다. 실제로 이러한 취약점을 악용하여 국내 ISP 망에 연결된 IoT 기기를 마비시킨 사례가 존재한다. 이러한 침해사고를 예방하기 위해서는 제품 출시 전 취약점 점검을 통한 사전 대응이 필수적이다.

다음으로 IP 카메라와 유·무선 공유기에서 많이 발생한 사례이다. 개발자는 편의성을 위해 시스템 명령입력이 가능한 디버깅 페이지 및 유지보수용 계정을 따로 만들어놓고 제거하지 않은 채 사용자에게 배포한 경우다. 대다수의 공격자는 일차적으로 펌웨어 분석을

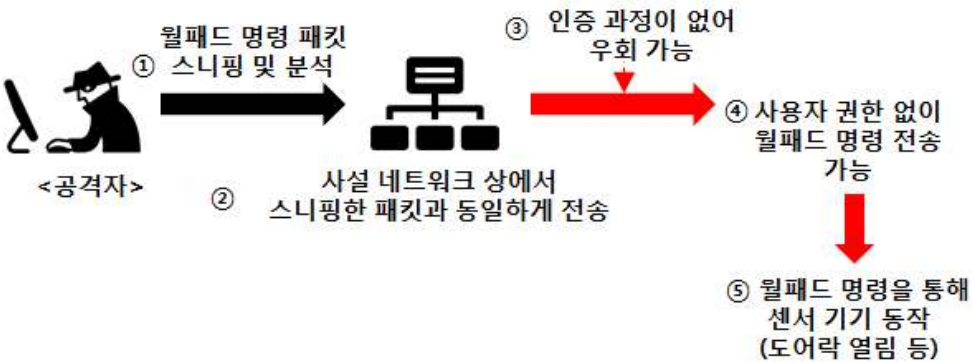
통해 취약점을 찾는다. 권한 없이 접근 가능한 유지보수용 페이지 및 하드 코딩된 계정 등이 있는 경우 펌웨어 소스코드를 통해 손쉽게 IoT 기기 관리자 권한을 장악할 수 있다. 그러므로 반드시 초기 개발 단계에서 유지보수용 계정 등을 삭제 한 뒤 사용자들에게 배포하는 것이 중요하다.

■ 네트워크 서비스 보안 위협



<그림 2-2> 불필요한 포트 사용 취약점을 악용한 공격 사례

취약한 IoT 기기가 악성코드에 감염, DDoS 공격을 수행하여 인터넷 서비스가 약 2시간가량 지연된 사례가 있다. 공격자는 개방된 텔넷 포트를 통해 기본 ID/PW로 로그인하여 관리자 권한을 획득했다. 취약한 포트와 기본 ID/PW 설정으로 방치된 점을 악용한 사례라고 볼 수 있다.



<그림 2-3> 제어 패킷 인증 취약점을 악용한 공격 사례

다음으로 IoT 게이트웨이가 IoT 센서로 명령을 내릴 때 암호화를 하지 않은 상태로 전송하는 경우이다. 패킷이 암호화가 안 되어 있을 경우, 공격자는 사용자가 아파트 내

IoT 센서로 명령을 내릴 때, 스니핑을 통해 명령 패킷을 획득하여 어떤 방식으로 명령을 전달하는지 알 수 있다. 특히 홈 컨트롤러를 공용으로 쓰고 있는 아파트는 대게 IP 등을 순서대로 지정해주기 때문에 원하는 집 내부 화면, 도어락 설정 등을 자유롭게 지정하여 명령할 수 있다. 실제로 IoT 기기 제어 패킷이 암호화가 안 되어 있는 경우가 많다. 이는 도어락, 가스 벨브 잠금 등 파급도 큰 사고가 일어날 수 있기 때문에 제어 패킷 송신 시 사용자에게 패킷 알람을 보내는 등 공격자의 비정상적인 패킷이 사전에 차단되도록 예방해야 한다.

■ 모바일 앱 보안 위협

모바일 앱은 주로 IoT 기기와 함께 사용되며, IoT 센서 명령을 조작하는 용도로 사용된다. 안드로이드 앱의 경우 디버깅이 가능하기 때문에 실제 코드가 노출되기 쉽다. 난독화가 안 되어 있는 코드는 실제 앱이 어떤 방식과 알고리즘으로 돌아가는지 상세히 알려주는 역할을 한다. 이를 통해 공격자가 권한 우회 방법 등 취약점을 찾는 기반을 만들어주므로 제품 출시 전 난독화를 하는 것이 무엇보다 중요하다.

■ 펌웨어 업데이트, 물리적 보안 위협

<그림 2-4> 펌웨어 취약점을 악용한 공격 사례

최근 하드웨어를 통한 공격 방법도 늘어나고 있다. 공격자는 주요정보를 획득하기 위해 1차적으로 UART 포트에 접속한다. UART로 접속하면 쉘을 획득하거나 관리자 PW 등 중요 정보 수집이 가능하다. 그러므로 하드웨어 부분에 직접 접속 할 경우, 중요정보가 노출되지 않도록 사전 조치하는 것이 필요하다.

최근 발견된 쇼다운 프록시(SSHowDowN Proxy)와 국내 침해사고를 발생시킨 취약점 Bash 취약점의 공통점은 오픈 라이브러리라는 점이다. 특히 쇼다운 프록시(SSHowDowN

Proxy)는 기존 Open SSH 취약점과 IoT 기기 기본 ID/PW를 사용하는 점을 결합한 것으로 파급도가 매우 광범위했다. 지금도 IP 카메라나 위성 안테나, 공유기, NAS 등 여러 IoT 기기가 과거 Open SSH로 적용 되어있어 공격에 무방비로 노출 되어 있다. 오픈 라이브러리는 무료로 누구나 사용 할 수 있기 때문에 많이 사용하고 있으며, 공격자들은 오래된 버전의 오픈소스 취약점을 악용한다. 그러므로 업데이트를 통해 최신 버전의 오픈소스를 사용해야 한다.

그리고 사용자가 IoT 펌웨어 업데이트 파일을 받을 때 파일이 정상파일인지 확인하는 무결성 검증을 안 할 경우, 조작된 업데이트 파일을 받게 되면 바로 악성코드에 감염된다. 여기서 무결성 검증이란, 서버가 업데이트 파일의 고유한 값(해쉬 등)을 키로 암호화하여 전송하고 사용자 펌웨어는 암호화한 값을 해당키로 해독하여 실제 받은 파일의 고유한 값과 동일한지 비교하는 절차이다. 이러한 검증 절차가 부재할 경우, 사용자단에서는 서버에서 정상적으로 보낸 파일인지 확인 하지 않고 의심 없이 업데이트를 수행한다. 그러므로 업데이트 시 무결성 검증을 수행해야한다.

IoT 기기 악성 코드는 위와 같은 취약점을 통해 공격함으로써 기기 내부파일 시스템에 악성코드를 설치한다. 한 번 설치된 악성 코드는 전원 꺼진 이후에도 삭제되지 않고 영구적으로 저장되어 있다가 켜질 때 재실행된다. 악성 코드에 감염됐는지 실시간 탐지하는 백신이 없는 상황이므로 파일을 실행하기 전에 무결성 검사를 통해 악성 코드 파일인지 탐지하여 실행되지 않도록 차단하는 것 또한 필요하다.

IoT가 보급화 되어 감에 따라 보안 위협 또한 이처럼 광범위해지고 있으며, 이에 따른 사전 대응이 무엇보다 필요한 실정이다.

제3장

스마트 홈·가전 보안 가이드

제3장 스마트 홈 · 가전 보안 가이드

본 장에서는 사물인터넷 기반의 스마트 홈 · 가전 시스템 보안 대응 방안을 제시한다. 이를 위하여 웹 인터페이스, 인증/허가, 네트워크, 모바일, 펌웨어, 업데이트, 물리적 보안으로 분류하고 해당 항목에 대하여 발생할 수 있는 보안 위협 및 보안 취약점을 분석하였으며, 이를 기반으로 보안 취약점에 대한 세분화된 보안 대응 방안을 제시하였다.

구분	구현항목
웹 인터페이스 보안	- 관리자 페이지에서 사용자가 5회 로그인 실패할 경우, 캡차(CAPTCHA) 등의 보안 기능을 수행한다. **(대안)** 로그인을 5회 이상 실패할 경우, 사용자 입력을 3~5분 동안 중지시키는 등의 계정 잠금 메커니즘을 적용한다. - XSS, SQL Injection 취약점 등에 대한 보안 대응책을 수립한다.
인증/허가 보안	- 관리자 페이지 접속 시 최초 ID와 비밀번호의 경우 제품마다 다르게 한다. **(대안)** 비밀번호 복잡도를 만족하는 최초 비밀번호 설정 시에만 디바이스 사용이 가능하도록 한다. - 모든 비밀번호(최초, 변경 모두 해당)는 영문, 숫자, 특수문자를 포함하여 8자 이상으로 설정하여야 한다. - 사용자 관리 페이지 이외에 유지 보수 등을 위한 디버깅 페이지(숨김 페이지 등) 및 계정을 별도로 구현하지 않아야 한다. - 모든 관리자 페이지는 인증 후에 접근할 수 있도록 세션 인증 등을 구현한다. - 세션 인증 구현 시 순차적으로 증가하는 값이나 단순 ID 값 등 예측 가능한 세션 ID 값을 사용하지 않는다. - 디바이스는 게이트웨이를 인증하고 게이트웨이는 디바이스를 인증하는 등의 상호 인증(이중 인증)을 수행해야 한다. (게이트웨이/서버, 게이트웨이/모바일앱 등) - (도어락) 디바이스를 제어하는 패킷이 탐지될 경우 이용자에게 알림을 제공한다.

<table>
<tr><td>네트워크
서비스
보안</td><td>- 시큐어코딩(URL 파라미터, 사용자 입력 값에 대한 길이 및 데이터 확인 등)을 통해 보안 취약점(버퍼오버플로우, 시스템 명령 취약점 등)을 최소화하여야 한다.

- 불필요한 외부 접속 포트나 Telnet, FTP, UPnP, SNMP 등의 서비스를 기본 비활성화 하여야 한다.

- 네트워크 패킷에서 중요 정보(계정 로그인, IoT 기기 제어 명령 시 등)가 보이지 않도록 암호화하여야 한다.
※ 중요정보 암호화 시 HTTPS(SSL/TLS), CoAP(DTLS) 등의 프로토콜을 이용하도록 한다.
※ (IP 카메라) 영상 암호 송출 시 SRTP, IPsec, SSL 등의 경량 암호 프로토콜을 이용하도록 한다.
<table><tr><th colspan="2">SSL/TLS 사용 시 권고 사항</th></tr><tr><td>암호 알고리즘</td><td>AES128 및 256(CBC모드), LEA(128bit, 경량 암호 알고리즘), HIGHT(64bit 경량 암호 알고리즘)</td></tr><tr><td>해쉬 함수</td><td>SHA-256</td></tr></table></td></tr>
<tr><td>모바일 앱
보안</td><td>- 앱에 역분석 방지 기능(난독화 등)을 적용한다. (Proguard, Dexguard 등)

- 무선 네트워크 연결 시 인증 정보가 노출되지 않도록 한다.</td></tr>
<tr><td>펌웨어
보안</td><td>- (IP 카메라) 개인영상정보를 DB에 별도로 저장, 보관하는 경우에는 이를 암호화하여야 한다.

- 중요 정보(테스트용 계정, 개인키 등)를 하드 코딩하지 않도록 한다.
※ 비밀번호 등이 소스코드에 포함되지 않도록 확인하고, 백업파일 또는 개발 샘플 파일 등을 모두 삭제했는지 검토 필요
※ busybox등에 불필요한 명령어가 포함되지 않도록 한다.

- 패스워드 저장 시 일방향 해쉬로 저장하여야 한다. (SHA-256 이상)

- 커널이나 프로그램 컴파일 시 옵션에 각종 보안 기능이 포함되도록 빌드해야 한다. (ASLR, DEP, NX 등).

- (도어락, 홈 컨트롤러) 기기별 MAC 주소 인증 방식을 사용하여, ARP 스푸핑을 최소화하도록 한다.</td></tr>
</table>

업데이트 보안	- 취약점 발견 시 업데이트를 할 수 있는 기능을 제공해야 한다. ※ 펌웨어 다운그레이드는 허용하지 않는 것을 권고 - 취약한 오픈소스 라이브러리를 사용하지 않도록 하며 업데이트 시 보안 패치된 라이브러리가 존재할 경우 업데이트 하도록 한다. - 펌웨어 업데이트가 발생하는 경우 사용자가 인지할 수 있는 방안을 강구하여야 한다. **(대안)** 팝업 등을 통해 실시간 알려주기 어려운 경우, 제조사 및 서비스 홈페이지에 보안 업데이트 사항을 게시한다. - 펌웨어 업데이트, 실행 시 파일 고유 해시 값을 비교하여 변조 여부에 대한 무결성 검증을 실시할 수 있도록 한다. (중요 파일의 무결성 검사를 파일 실행 전 또는 부팅 시점에 수행하여야 한다.) ※ 검증 시 해시 알고리즘을 이용하도록 한다.(SIMON, SPECK, LEA 등) - 무결성 검증에 실패한 경우 다운로드 받거나 복사한 파일은 삭제시킨다.
물리적 보안	- USB 포트 등 외부 포트를 사용하지 않도록 설정하는 기능을 제공하여야 한다. - 디버그 포트(UART, JTAG)를 통해 중요 정보가 노출되지 않도록 한다.

제4장
보안 가이드 항목 해설서

제4장 보안 가이드 항목 해설서

■ 웹 인터페이스

- 관리자 페이지에서 사용자가 5회 로그인 실패할 경우, 캡차(CAPTCHA) 등의 보안 기능을 수행한다.

사물인터넷 기기는 인가된 이용자만이 관리자 페이지 등을 통해 기기를 제어할 수 있어야 하며, 이에 허가받지 않은 이용자의 접속을 방지하기 위해 캡차를 활용한다. 캡차를 적용하기 어려운 경우, 사용자가 5회 이상 로그인에 실패했을 때 입력을 3~5분간 중지시키는 등의 계정 잠금 메커니즘을 적용한다.

- XSS, SQL Injection 취약점에 대한 보안 대응책을 수립해야한다.

스마트 홈 · 가전 제품에 시큐어 코딩을 적용하지 않았을 경우 입력 데이터 검증 및 표현, 보안 기능, 시간 및 상태, 에러 처리, 코드오류, 캡슐화, API 오용 등에서 보안 취약점이 발생할 수 있으며 공유기, IP 카메라, 앱 등의 XSS 취약점은 사용자 정보를 모으기 위한 1차 공격으로 이루어지는 경우가 있어 필터링 적용이 필요하다.

※ 시큐어 코딩 세부 사항은 **행정안전부의 시큐어 코딩 가이드라인** 참고
(http://www.moi.go.kr/frt/bbs/type001/commonSelectBoardArticle.do;jsessionid=EnceiH21S3zShV33v0xqbGAFytDYkM1t8eCYDyl2Z83qsixSO9vXcm6rerOhaElm.mopwas51_servlet_engine1?bbsId=BBSMSTR_000000000045&nttId=34430)

■ 인증/허가 보안

- 관리자 페이지 접속 시 최초 ID와 비밀번호의 경우 제품마다 다르게 한다.

- 모든 비밀번호(최초, 변경 모두 해당)는 영문, 숫자, 특수문자를 포함하여 8자 이상으로 설정해야 한다.

제품에 대한 관리 및 제어를 위해 이용자에게 관리자 페이지를 웹 인터페이스를 통해 제공되고 있으며, 허가받지 않은 이용자가 관리자 페이지에 접근하지 못하게 해야 한다.

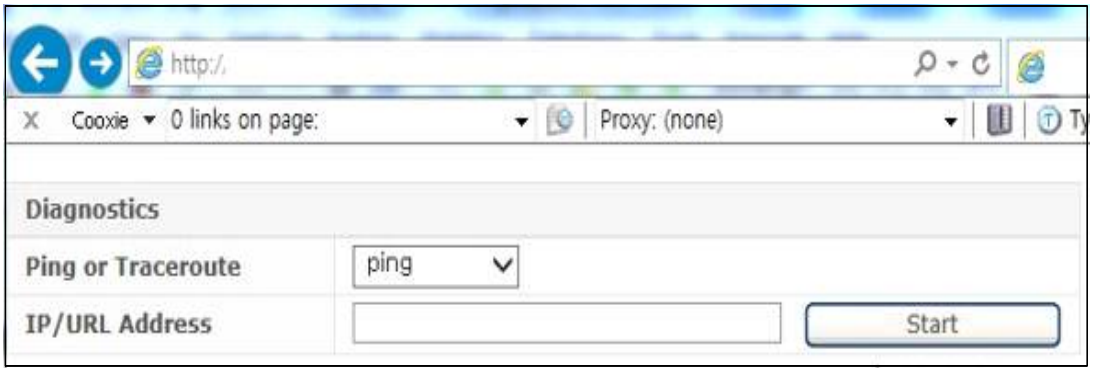
이에 악의적인 목적을 가진 비인가자가 관리자 페이지에 접근할 수 없도록 기기별 비밀번호는 다르게 하고, 추측이 어려운 비밀번호로 설정 하게끔 해야 한다. 따라서 제조사에서는 기기에 각각의 비밀번호를 명시하고, 사용자가 비밀번호 설정/변경 시 영문, 숫자, 특수문자를 포함하여 8자 이상으로 설정하여 충분한 보안 수준을 유지할 수 있도록 해야 한다. 각각 기기별 비밀번호 명시가 어려울 경우, 사용자가 최초 접속 시 복잡도를 만족한 비밀번호를 설정하도록 한다. <표 4-1>은 복잡도가 높은 패스워드를 설정하는 방법 예시이다.

<표 4-1> 복잡도가 높은 패스워드 설정 방법 예시

예측이 어려운 비밀번호	
1.	영문자(대・소문자), 숫자, 특수 문자들을 혼합한 구성 예시) ‘10H+20Min’, ’!!Can&9it’ 등과 같은 구성
2.	특수 문자 활용 시, 비밀번호 시작 혹은 끝 부분이 아닌 위치에 삽입하여 설정 예시) ’Security1!’ 이 아니라 ‘Securi2t&&y’ 와 같은 형태로 설정
3.	영문자(대・소문자)를 구분할 수 있을 경우, 대・소문자를 혼합하여 설정 예시) ’gksrnrdlsxjsptwjdqh’ -> ’gkSrnSdlSxJspTwjDqH’ 등

- 사용자 관리 페이지 이외에 유지 보수 등을 위한 페이지(숨김 페이지 등) 및 계정을 별도로 구현하지 않아야 한다.

유지보수 등을 위해 숨김 페이지 또는 계정을 별도로 구현하는 경우가 있다. 주로 공격자들은 펌웨어 분석을 통해 공격하므로 쉘 명령 실행 페이지 및 임의의 슈퍼 계정을 구현하지 말아야 한다. 또한 관리자 권한 인증 없이 관리자 권한의 페이지에 접속 가능한지 사전 검증해야 한다.



<그림 4-1> 유지보수용 숨김 페이지

- 모든 관리자 페이지는 인증 후에 접근할 수 있도록 세션 인증 등을 구현한다. - 세션 인증 구현 시 순차적으로 증가하는 값이거나 단순 ID 값 등 예측 가능한 세션 ID 값을 사용하지 않도록 한다.

세션 인증에 대한 침해 사고를 방지하기 위한 보안 대응 방안은 다음과 같다.

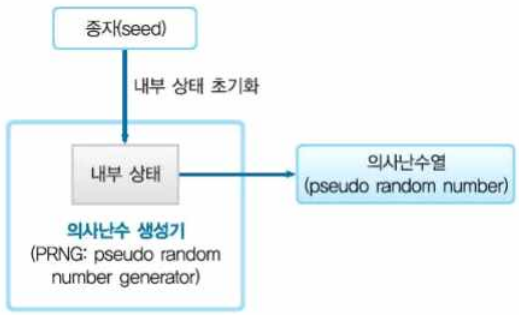
① 세션 인증 구현 시 예측 가능한 세션 ID 값을 사용하지 않는다.
② 세션 인증 구현 시 Hidden Field, Client Side Cookie 기반의 인증 기법이 아닌 서버세션 ID 인증을 사용한다.
③ 세션 인증 시 주기적인 세션 값을 변화 시켜야 하며 일정 시간이 지난 후에 폐기한다.

세션 ID는 큰 랜덤 풀로부터 생성되어야 하며, 32비트 이상의 값을 가져야 한다. 가짜 난수와 같은 동적 데이터는 전수 공격을 어렵게 할 수 있다.

그러나 길이가 길고 복잡한 항목으로 구성된 세션ID가 만들어져도 공격자가 충분한 시간과 자원이 있다면 이를 분석해내는 것이 가능하다. 예측하기 힘든 세션 ID를 생성하는 목적은 수많은 대역폭과 처리 자원을 가지고 있는 공격자가 하나의 유효한 세션ID를 추측하는데 최대한 오랜 시간이 걸리게 해서 쉽게 추측하지 못하게 하는 것이다.

이를 위하여 단순 조합 보다는 상용 웹서버나 웹 애플리케이션 플랫폼에서 제공하는 세션ID를 사용하거나, 하드웨어로 구성된 의사난수 생성기를 통해 세션을 구성하여야 한다. <표 4-2>은 의사난수 생성기의 구조이다.

<표 4-2> 의사난수 생성기 구조

구조	설명
	- 내부 상태 = 메모리값 - 내부 상태에 의해 다음에 생성할 의사난수가 정해지므로 내부 상태가 공격자에게 알려지지 않게 하여야 한다. - 외부에서 주어진 종자(seed)는 의사난수 생성기의 내부상태를 초기화한다.

또한 특정 기간의 비활동 또는 정해진 시간 이후의 상태 정보와 세션 ID를 무효화하여야 하며, 동시 로그인을 제한하여야 한다. 세션 만료에 대한 ASP, JSP 적용 방안은 다음과 같다.

- ASP

접속자 별로 세션을 생성하여 사용자의 정보를 각각 저장할 수 있는 Session 오브젝트를 사용하여 타임아웃 기능을 구현한다.

Session 오브젝트는 페이지의 접근을 허가하거나 금지할 때 또는 사용자 별로 정보를 저장할 때 많이 사용하며 접속자의 브라우저에서 쿠키기능을 지원해야 Session 오브젝트의 사용이 가능하다. 다음과 같은 설정이 적용될 경우 사용자가 로그아웃할 때 세션은 바로 삭제되며, 로그아웃하지 않고 10분 동안 웹 서버로의 요청이 없을 경우에도 세션은 없어지게 된다.

예) Session.Timeout = 10

<표 4-3>는 ASP의 세션 관련 항목이다.

<표 4-3> ASP의 세션 관련 항목

구분		설 명
Property	SessionID	사용자마다 갖게 되는 고유한 세션값
	Timeout	세션이 유지되는 시간
Method	Abandon	강제로 세션을 소멸시키는 함수
Event	Onstart	각각의 사용자가 처음 방문할 때 발생
	Onend	사용자의 세션이 끝나는 시점에 발생

- JSP

세션타임아웃기능을 구현하는 방법은 session.getLastAccessedTime() 를 이용하여 세션의 마지막 접근 시간으로부터 일정 시간 이내에 다시 세션에 접근하지 않은 경우 자동으로 세션을 종료하도록 구현한다.

세션의 타임아웃은 두 가지 방법으로 설정할 수 있다.

(1) web.xml 파일에서 <session-config> 태그를 사용하여 타임아웃을 지정하는 방법
(2) session 기본 객체가 제공하는 setMaxInactiveInterval() 메소드를 사용
<% session.setMaxInactiveInterval(600); %>

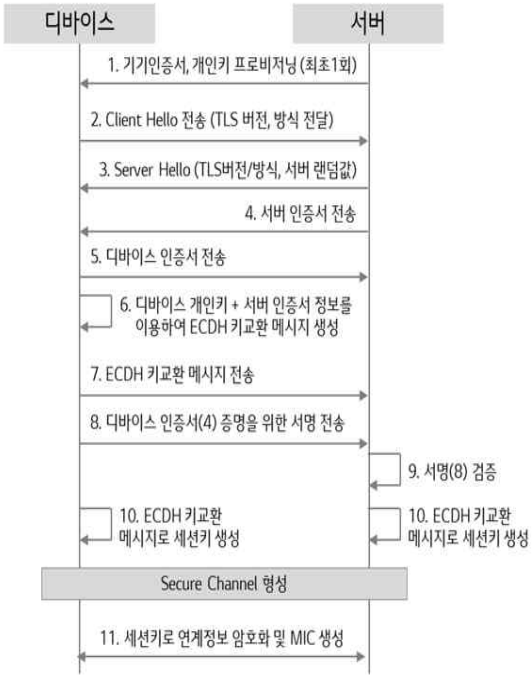
※ 세션 인증 및 관리에 대한 세부적인 사항은 **한국인터넷진흥원의 홈페이지 취약점 진단·제거 가이드** 참고 (https://www.kisa.or.kr/public/laws/laws3_View.jsp?mode=view&p_No=259&b_No=259&d_No=49&ST=T&SV=)

- 디바이스는 게이트웨이를 인증하고 게이트웨이는 디바이스를 인증하는 등 상호 인증(이중 인증)을 수행해야 한다. (게이트웨이/서버, 게이트웨이/모바일앱 등)

디바이스가 서버를 인증하는 로직은 구현됐으나 서버가 디바이스를 인증하는 로직이 없는 경우에는 서버에 신뢰되지 않은 공격자의 디바이스가 인증을 시도하여 성공할 수 있다. 인증은 상호간에 수행되어야 하며 다음의 고려 사항이 반영되어야 한다.

(1) 상호 인증을 시작하기 전에 초기키(PSK: Pre-Shared Key)가 안전하게 주입 및 관리되어야 한다.
(2) 상호 인증을 위한 적절한 매커니즘을 설계 및 구현해야 한다. 비대칭키 방식의 개인키를 이용한 인증 방식을 권장한다.
(3) 인증 후 발행된 세션키는 주기적으로 갱신될 수 있어야 한다. 또한, 디바이스에 키를 저장할 경우 소프트웨어 또는 하드웨어 상황을 고려하여 최대한 안전한 방식으로 저장해야 한다. (키 관리)

상호 인증을 위한 매커니즘은 비대칭키 방식의 개인키를 이용한 방식을 권장한다. 실제 적용할 수 있는 방식은 매우 다양하므로 하드웨어와 소프트웨어의 구성에 따라 적절히 선택해야 한다.



<그림 4-2> 비대칭키 기반 상호 인증

- 디바이스를 제어하는 패킷이 탐지될 경우 이용자에게 알림을 제공한다.

스마트 도어락은 허가된 이용자만이 이용 및 관리하여야 한다. 사전에 IP 및 맥 등을 인증 받고, 인증 받지 않은 공격자가 위·변조를 통해 정상적이지 않은 패킷을 전송할 경우, 이를 탐지(IP 및 MAC 대조)하여 이용자에게 알림을 제공하거나, 모든 명령 패킷 전송 시 알림을 전달하는 방식을 사용해야 한다. <그림 4-3>은 스마트 도어락의 개폐 알림 예시이다.

<그림 4-3> 스마트 도어락 개폐 알림 예시

■ 네트워크 서비스 보안

- 시큐어코딩(URL 파라미터, 사용자 입력 값에 대한 길이 및 데이터 확인 등)을 통해 보안 취약점(버퍼오버플로우, 시스템 명령 취약점 등)을 최소화하여야 한다.

버퍼오버플로우, 시스템 명령 취약점은 침해사고 뿐만 아니라 신고포상제를 통해 가장 많이 조치되고 있는 취약점 중 하나이다. 파라미터, 사용자 입력 값의 길이, 데이터 값을 필터링 없이(사용자 입력 값에 대한 검증 누락 또는 부적절한 검증) 받아서 시스템 함수 인자로 바로 대입되거나, PC 값을 변조하는 등의 문제가 많이 발생한다.

이에 DirBuster, Cross Fuzz 등의 퍼징 툴을 이용하여 취약점에 대한 사전 점검을 진행하는 것이 필요하며, 설계 및 개발 단계에서 시큐어 코딩을 적용하여야 한다. <표 4-4>은 최근 주로 발생하는 명령어 삽입, 버퍼 오버플로우 등에 대한 시큐어코딩 예시이다.

<표 4-4> 입력데이터 검증 및 표현에 대한 보안 약점에 대한 대응 방안 예시(C)

보안 약점	운영체제 명령어 삽입
설명	외부 입력이 시스템 명령어 실행 인수로 적절한 처리 없이 사용되어 발생하는 보안 약점이다. 일반적으로 명령 인수나 스트림 입력 등 외부 입력을 사용하여 시스템 명령어를 생성하는 프로그램에서 발생하며, 이러한 경우 외부 입력 문자열에 대한 적절한 처리를 해주지 않을 시 공격자가 원하는 명령어 실행이 가능하게 된다.
안전한 코딩 기법	- 외부 입력이 직접 또는 문자열 복사를 통하여 함수에 직접 전달되는 것은 위험하다. 미리 적절한 후보 명령어 리스트를 만들고 선택하게 하거나, 위험한 문자열의 존재 여부를 검사하는 과정을 수행하여야 한다.

보안 약점	버퍼 오버플로우
설명	할당되는 버퍼의 한계치를 넘는 경우 발생하는 보안 약점
안전한 코딩 기법	- 프로그램이 버퍼가 저장할 수 있는 것보다 많은 데이터를 입력하지 않는다. - 프로그램이 버퍼 경계 밖의 메모리 영역을 참조하지 않는다. - 프로그램이 사용할 메모리를 적절하게 계산하여 로직에서 에러를 발생시키지 않도록 한다. - 입력에 대해서 경계 검사(Bounds Checking)를 한다. - strcpy()와 같이 버퍼 오버플로우에 취약한 함수를 사용하지 않는다.

※ 시큐어 코딩에 대한 세부적인 사항은 **행정안전부의 시큐어 코딩 가이드라인** 참고 (http://www.moi.go.kr/frt/bbs/type001/commonSelectBoardArticle.do;jsessionid=EnceiH21S3zShV33v0xqbGAFytDYkM1t8eCYDyl2Z83qsixSO9vXcm6rerOhaEIm.mopwas51_servlet_engine1?bbsId=BBSMSTR_000000000045&nttId=34430)

- 불필요한 외부 접속 포트나 Telnet, FTP, UPnP, SNMP 등의 서비스 비활성화를 기본 값으로 하여야 한다.

사물인터넷 기기는 IoT의 특성상 유·무선 네트워크로 연결되어 있어 다양한 침투 경로를 통한 보안 위협을 지니고 있다, 이에 다양한 서비스가 제공될수록 해당 서비스에 존재하는 보안 위협이 공유 될 수 있으며 이를 위한 대응 방안은 다음과 같다.

① 불필요한 외부 접속 포트(Telnet, FTP, UPnP, SNMP) 등의 서비스 비활성화를 기본 값으로 설정한다.
② 외부 접속 포트를 사용할 경우 비밀번호 설정, 접근 IP 제한 등의 추가적인 보안 조치를 수행하도록 한다.

- 네트워크 패킷에서 중요 정보(계정 로그인, IoT 기기 제어 명령 시 등)가 보이지 않도록 암호화해야 한다.

※ 중요정보 암호화 시 HTTPS(SSL/TLS), CoAP(DTLS) 등의 프로토콜을 이용하도록 한다.
※ (IP 카메라) 영상 암호 송출 시 SRTP, IPsec, SSL 등의 암호 프로토콜을 이용하도록 한다.

SSL/TLS 사용 시 권고 사항	
암호 알고리즘	AES128 및 256(CBC모드), 3DES, LEA(경량 암호 알고리즘)
해쉬 함수	SHA-256

사물인터넷 기반에서 네트워크 간의 보안을 확보하기 위해서는 연결되어 있는 디바이스가 서로 신뢰할 수 있는 디바이스인지 확인할 수 있어야 하며, 디바이스간의 송·수신 데이터가 제3자에 의해 도청되는 것을 방지해야 한다.

따라서 연결되어 있는 디바이스가 신뢰할 수 있는 것을 증명하기 위해 전자서명이 포함된 인증서를 사용하여, SSL/TLS 등의 프로토콜을 통해 송·수신 데이터를 암호화하여야 한다. 또한 영상 암호화의 경우 경량 암호 프로토콜인 SRTP, IPsec, SSL 등을 이용하여 전송하는 것을 권장한다.

■ 모바일 앱 보안

- 컴파일러 옵션을 활용하여 난독화 등의 메모리 보호 기법을 적용하여야 한다.

모바일 앱의 경우 공격자가 메모리 해킹을 통해 관리자 권한 탈취가 가능한 데이터를 수집할 수 있다. 이에 스마트 홈·가전 제품 개발 시, 컴파일러 옵션을 활용하여 난독화 등의 메모리 보호 기법을 적용해야 한다.

현재 안드로이드 운영체제에서는 ProGuard, Dex Guard, Android Env 등을 이용하여 난독화 할 수 있다. ProGuard는 안드로이드 SDK를 통해 난독화를 지원하며, 변수 명 변경 등의 간단한 난독화 기능을 무료로 이용할 수 있다. Dex Guard는 ProGuard의 유료 버전으로, ProGuard 대비 난독화 보안성이 향상된 특징을 가지고 있다. Android Env는 안드로이드 상용 난독화 도구로써, 단순 함수 명 변경 뿐 아니라 프로그램 플로우 셔플 등의 난독화를 지원한다. 이 중 무료로 제공되는 ProGuard를 통한 난독화는 build.gradle 파일에 소스 코드를 삽입하여 적용할 수 있다. <그림 4-4>은 ProGuard를 통한 난독화 적용 소스 코드 예시이다.

※ ProGuard를 이용한 난독화에 대한 세부적인 사항은 **안드로이드 개발자 사이트의 이용자 가이드** 참고 (https://developer.android.com/studio/build/shrink-code.html#configuring)

```
buildTypes {

    release {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
```

<그림 4-4> ProGuard를 통한 난독화 적용 소스 코드(JAVA)

컴파일러의 난독화 옵션을 적용할 시, 공격자가 기능을 파악하지 못하도록 함수 명, 배열 명 등이 의미 없는 문자열로 변환된다. <표 4-5>는 컴파일러 난독화 옵션 적용 전·후의 예시이다.

<표 4-5> 컴파일러의 난독화 옵션 적용 예시(JAVA)

컴파일러의 난독화 옵션 적용 예시(전)

```
public Car(TrackPosition pos, Color drawColor,{
        this.pos_ = pos;
        this.drawColor_ = drawColor;
        this.eraseColor_ = eraseColor;
        this.gasPedal_ = gasPedal;
        int[] xs = new int[4];
        int[] ys = new int[4];
        this.poly_ = new Polygon(xs, ys, 4);
}
```

컴파일러의 난독화 옵션 적용 예시(후)

```
public a(h paramh, Color paramColor1, Color paramColor2{
        this.a = paramh;
        this.b = paramColor1;
        this.c = paramColor2;
        this.e = paramb;
        int[] arrayOfInt1 = new int[4];
        int[] arrayOfInt2 = new int[4];
        this.d = new Polygon(arrayOfInt1, arrayOfInt2, 4);
}
```

※ 모바일 앱 보안성 검증에 대한 세부적인 사항은 **행정안전부의 모바일 대국민 전자정부서비스 앱 가이드라인** 참고
(https://www.kisa.or.kr/jsp/common/down.jsp?folder=uploadfile&filename=%EB%AA%A8%EB%B0%94%EC%9D%BC_%EB%8C%80%EA%B5%AD%EB%AF%BC_%EC%A0%84%EC%9E%90%EC%A0%95%EB%B6%80%EC%84%9C%EB%B9%84%EC%8A%A4_%EC%95%B1_%EA%B0%80%EC%9D%B4%EB%93%9C%EB%9D%BC%EC%9D%B8_v4.pdf)

- 무선 네트워크 연결 시 인증 정보가 노출되지 않도록 한다.

무선 네트워크 연결 시, 공격자가 같은 망에 있는 경우(카페 등) 로그인 또는 중요사항 변경 시 개인 정보 등이 노출되어 있는 경우 스니핑을 통해 획득할 수 있으므로 이를 사전 방지해야 한다.

■ 펌웨어 보안

- 개인영상정보를 DB에 별도로 저장, 보관하는 경우에는 이를 암호화하여야 한다.

IP 카메라의 경우 기능의 특성상 개인영상정보 데이터 탈취가 가장 많이 발생되고 있다. 이에 영상정보 데이터를 수집 및 저장할 경우 데이터 암호화를 통해 사생활 침해, 영상 데이터 노출 등을 방지해야 한다.

따라서 개인영상정보를 DB 등에 집적하여 별도로 저장, 보관하는 경우에는 이를 암호화하여 악의적인 목적을 가진 공격자로부터 보호해야 한다. 이를 통해 의도치 않게 영상정보가 유출되더라도 암호화 된 데이터이기 때문에 개인정보를 보호할 수 있다.

※ 영상 정보 보호에 대한 세부적인 사항은 **CCTV 개인영상정보보호 가이드라인 해설서** 참고
(http://www.kisa.or.kr/jsp/common/downloadAction.jsp?bno=18&dno=29&fseq=1)

- 중요 정보(테스트용 계정, 개인키 등)를 하드 코딩하지 않도록 한다. ※ 비밀번호 등이 소스코드에 포함되지 않도록 확인하고, 백업파일 또는 개발 샘플 파일 등을 모두 삭제했는지 검토 필요 ※ busybox등에 불필요한 명령어가 포함되지 않도록 한다.

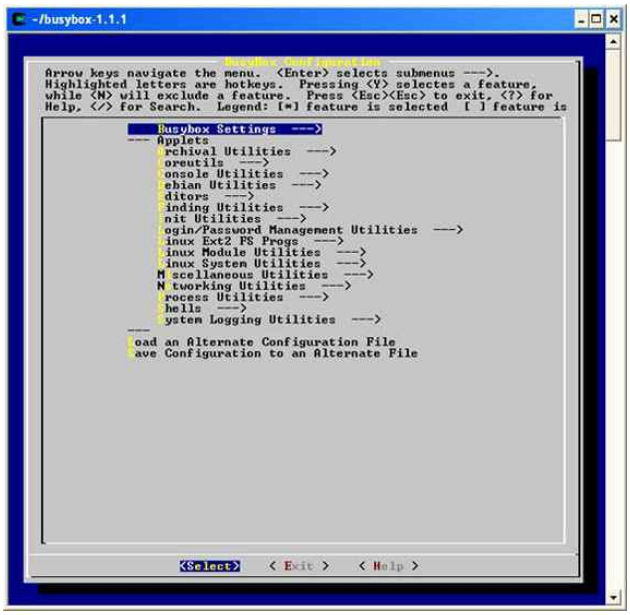
취약점 중 가장 많이 펌웨어에서 발견되는 것 중 하나가 하드코딩 된 비밀번호 등이다. 테스트 용으로 사용하거나, 관리자용으로 만들어 놓은 후 그대로 배포하는 경우가 많다. 공격자들은 펌웨어 분석을 통해 취약점을 파악하여 쉽게 관리자 권한을 획득할 수 있으므로 소스코드에 중요정보가 들어갔는지 배포 전 주의해야한다.

추가로 디바이스 운영체제가 리눅스인 경우 다양한 명령어가 busybox에 패키징되어 있다. 명령 실행 취약점 등이 있다면 공격자는 원격에서 이러한 명령어를 이용하여 악성코드를 주입하거나 원하는 기능을 실행하여 기기의 실행파일과 속성 값을 변경할 수 있다. 다음은 배시 취약점(shellshock)을 이용하여 무선랜으로 busybox 명령어를 이용한 공격자의 실행을 예로 든 것이다.

- 텔넷 오픈: header_value () {:;}; /telnetd && ps
- 무선랜 설정 변경: header_value () {:;}; echo "wl -i $WLAN1 set_pmk 1234" >> /usr/local/wifi/connect_wpa2_tkip.sh

이 외에도 명령어를 조합하여 원격 접속, 관리자 권한 상승, 무결성 침해, 디도스 공격 등이 가능하므로, 의도하지 않은 명령어가 실행되지 않도록 불필요한 busybox 명령어를 삭제하는 것이 바람직하다. busybox menuconfig 를 통해 최소한 다음과 같이 악용될 수 있는 명령어는 삭제하여 빌드를 해야 하며, 가능하다면 모든 명령어를 삭제한 후 필요한 것만 추가하는 것을 권장한다.

① nc - 원격 접속에 악용될 수 있음
② wget - 외부의 악성코드를 기기로 다운받을 수 있음
③ telnetd - 텔넷 데몬이 기동되어 원격 백도어로 악용될 수 있음
④ ftpd - 파일 전송



<그림 4-5> busybox 컴파일 설정

- 커널이나 프로그램 컴파일 시 옵션에 각종 보안 기능이 포함되도록 빌드 해야 한다. (ASLR, DEP, NX 등)

버퍼오버플로우 등의 취약점을 발견한 공격자는 펌웨어 주소를 기반으로 익스플로잇 코드를 작성하게 되는데 이 경우, ASLR/DEP 등의 보호기능을 추가하여 빌드할 경우, 주소값이 실행시마다 변경되고, 임의코드 실행 권한 영역이 제한되어 있어 공격에 성공하기 매우 어렵게 되므로 추가적인 공격을 막기 위해 보안 기능을 수립해야 한다.

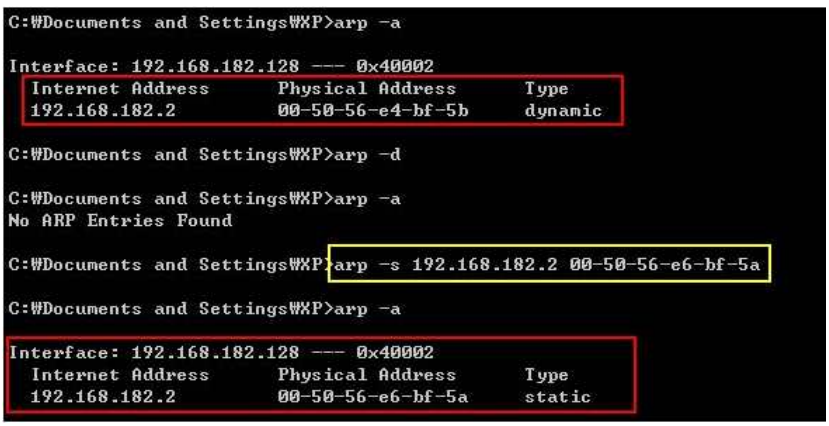
- 기기별 MAC 주소 인증 방식 등을 사용하여, ARP 스푸핑을 최소화하도록 한다.

홈 컨트롤러는 공격자가 ARP 스푸핑 공격 등을 통해 네트워크상의 패킷을 도청할 수 있는 보안 취약점을 가지고 있으며, 이는 피해자의 개인정보 유출, 사생활 침해 등으로 이어질 수 있다. 따라서 기업은 이용자가 이러한 공격에 노출되지 않도록 해야 한다. ARP 스푸핑, 리플라이어택 방지를 위한 대응 방안은 다음과 같다.

① 지속적인 ARP 응답 발생 시 이용자에게 알림을 제공한다.
② 정적인 ARP 테이블 관리를 수행하여야 한다.
③ 패킷 필터링을 적용한다.
④ 데이터 통신 시 타임 스탬프, 순서 번호 등을 이용한다.
⑤ 각 디바이스 별 MAC 주소를 기반으로 한 인증 방식을 제공하여야 한다.

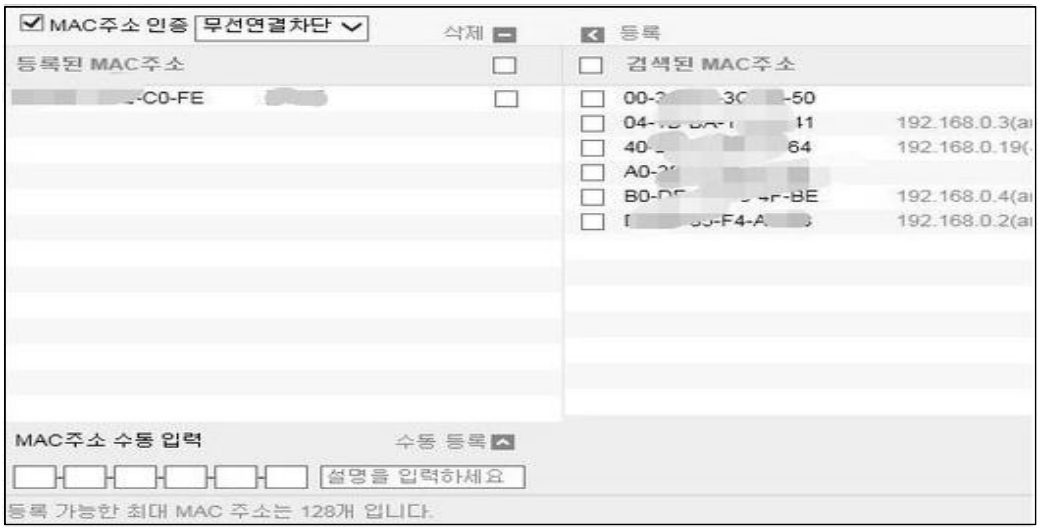
ARP 스푸핑 공격은 공격 시, 지속적인 ARP 응답이 발생하는 특징을 가지고 있으므로 이를 탐지하여 이용자에게 알림을 제공할 수 있다. 또한 정적인 ARP 테이블 관리를 통한 인증방식, ARP Request가 없는 ARP Response를 수신하면 알람을 주는 등의 패킷 필터링을 통해 ARP 스푸핑을 방지할 수 있다. 이 공격 또한 우회할 수 있기 때문에 명령 제어 패킷 송신을 할 경우 사용자에게 알람을 주어 2차 확인을 할 수 있도록 제공하는 방법도 권장한다.

제조사는 이러한 보안 설정을 이용자가 쉽게 이용할 수 있도록 설명 내용을 해당 페이지에 명시하거나 별도 매뉴얼로 제공하는 것을 권고한다. <그림 4-6>은 ARP 스푸핑을 방지하기 위하여 MAC 주소를 정적 주소로 변환하는 예시이다.



<그림 4-6> 정적 MAC 주소 설정 예시

<그림 4-7>는 스마트 도어락의 MAC 주소 기반 인증 설정하는 화면이다.



<그림 4-7> 스마트 도어락의 MAC 주소 기반 인증 예시

이와 같이 디바이스 별로 부여되어 있는 MAC 주소 인증 방식을 통해 정상적이지 않은 기기를 차단하도록 해당 기능을 제공해야 한다.

■ 업데이트 보안

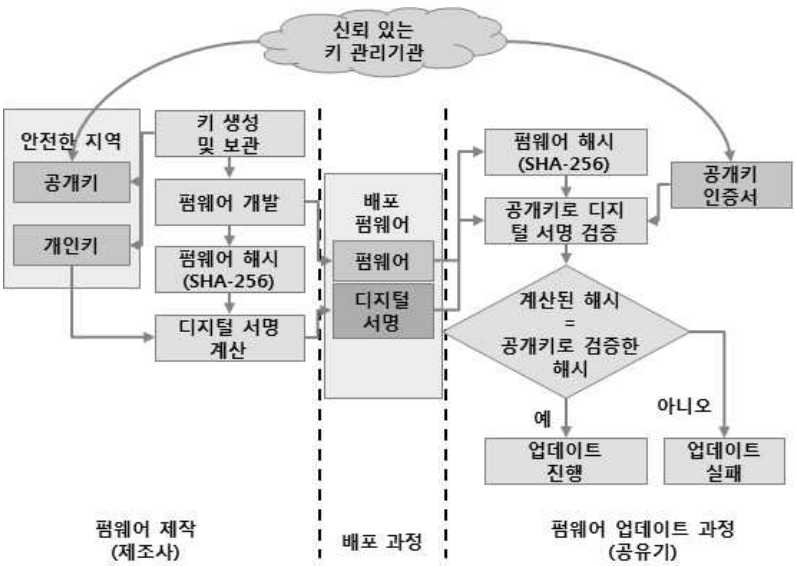
- 취약점 발견 시 업데이트를 할 수 있는 기능을 제공해야한다.
  ※ 펌웨어 다운그레이드는 허용하지 않는 것을 권고
- 취약한 오픈소스 라이브러리를 사용하지 않도록 하며 업데이트 시 보안 패치된 라이브러리가 존재할 경우 업데이트 하도록 한다.
- 펌웨어 업데이트가 발생하는 경우 사용자가 인지할 수 있는 방안을 강구하여야 한다.
  (대안) 팝업 등을 통해 실시간 알려주기 어려운 경우, 제조사 및 서비스 홈페이지에 보안 업데이트 사항을 게시한다.

지속적으로 스마트 홈·가전 보안 위협은 높아지고 있고 새로운 취약점이 발견되고 있다. 이에 대응하기 위해 취약점 발견 시 업데이트 할 수 있는 기능을 제공해야 한다. 디바이스는 펌웨어 업데이트를 통해 이용자 편의성 및 보안성 등을 향상시킬 수 있다. 그러나 펌웨어 업데이트를 제공하는 경우에도 일반 이용자의 경우 펌웨어 업데이트에 대한 필요성을 체감하지 못하고, 별도의 알림이 제공되지 않을 경우 존재 유무조차 모른 채 사용하는 경우가 많다.

따라서 펌웨어를 업데이트 패치를 배포할 경우 업데이트 알림 등을 통해 이용자가 인지할 수 있는 방안을 제시하여야 한다. 팝업 등으로 디바이스를 통해 실시간 알려주기 어려운 경우, 제조사 및 서비스 홈페이지에 보안 업데이트 사항을 게시하여 추가적으로 인지할 수 있도록 해야 한다.

- 펌웨어 업데이트, 실행 시 파일 고유 해시 값을 비교하여 변조 여부에 대한 무결성 검증을 실시할 수 있도록 한다.
  ※ 검증 시 해시 알고리즘을 이용하도록 한다.(SIMON, SPECK, LEA 등)
- 무결성 검증에 실패한 경우 다운로드 받거나 복사한 파일은 삭제시킨다.
  (중요 파일의 무결성 검사를 파일 실행 전 또는 부팅 시점에 수행하여야 한다.)

펌웨어 업데이트의 배포 및 패치 과정에서도 보안 위협이 발생될 수 있으며, 이에 펌웨어 업데이트 시 파일 고유 해시 값을 비교하여 변조 여부에 대한 무결성 검증을 실시해야 한다. 또한 무결성 인증 시 SHA-256 이상의 보안 알고리즘 사용을 권장한다. 또한 공개키 알고리즘을 활용하여 프로그램 코드에 대한 디지털 서명을 통해 펌웨어 업데이트 시 무결성과 인증 과정을 제공해야 한다. <그림 4-8>는 펌웨어에 대한 디지털 서명 과정이다.



<그림 4-8> 펌웨어 디지털 서명 과정

이와 같은 방법으로 펌웨어 업데이트 시 악의적으로 변조된 펌웨어에 대한 방어 기능을 제공할 할 수 있다. <표 4-6,7>는 디지털 서명 과정에 대한 설명이다.

<표 4-6> 펌웨어 디지털 서명 과정 1

번호	설명
1	- 펌웨어 제조사는 공개키와 개인키를 생성하고 공개키를 인증기관(CA)에 전달하여 인증기관(CA)으로부터 서명된 인증서를 발급받음
2	- 펌웨어 제조사는 발급받은 디지털 인증서(공개키)와 개인키를 디지털 서명 용도의 안전한 시스템에 보관

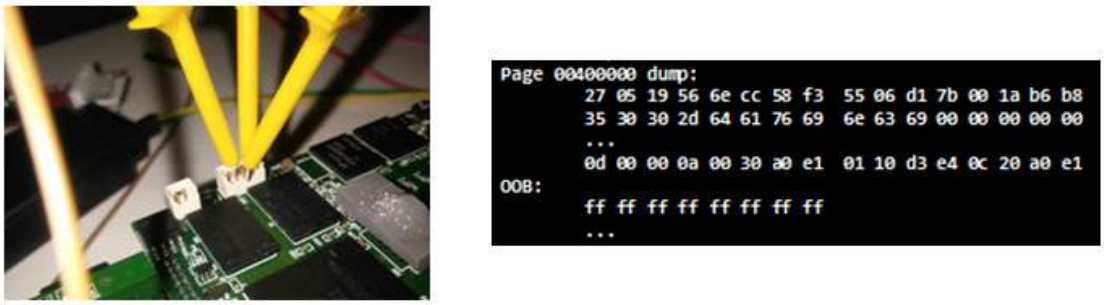
<표 4-7> 펌웨어 디지털 서명 과정 2

번호	설명
3	- 펌웨어 제조사에서 개발되어 배포 준비된 펌웨어 이미지를 디지털 서명 센터에 전달
4	- 디지털 서명 센서는 펌웨어에 개인키로 서명한 디지털 서명 정보를 추가하여 배포
5	- 디바이스는 로드된 펌웨어 이미지를 기기에 내장된 공개키를 이용하여 검증
6	- 검증 결과 일치 시, 펌웨어 업데이트 진행

■ 물리적 보안

- USB 포트 등 외부 포트를 사용하지 않도록 설정하는 기능을 제공하여야 한다.
- 디버그 포트(UART, JTAG)를 통해 중요 정보가 노출되지 않도록 하여야 한다.

USB, RS-232/485, UART, JTAG 등의 물리적인 접근포트가 노출되어 있을 경우(기기 외부 노출이든 내부 노출이든) 공격자가 노출된 포트에 프로브를 연결하여 로우 레벨의 신호로 공격할 수 있다. 제품이 업그레이드되면서 불필요해진 물리적인 포트를 제거하더라도 소프트웨어가 함께 제거되지 않으면 공격자가 이를 활성화하여 악용할 수도 있다.

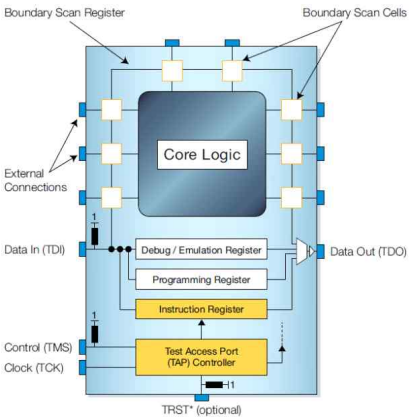


<그림 4-9> 노출된 UART 핀에 대한 물리적 공격

따라서, 공격의 대상이 될 수 있는 불필요한 하드웨어 입력 채널은 물리적으로 제거되어야 한다.

① USB - 엔지니어의 펌웨어 업데이트용 포트가 외부로 노출되었다면 반드시 제거 및 임의의 메모리를 자동으로 마운팅하지 않도록 조치
② RS-232/485 - 시리얼 라인으로 제어 명령이 암호화되지 않은 상태(텍스트 및 바이너리 코드)로 전송될 경우 태핑으로 인한 도청 위험이 있으므로 시리얼 라인을 물리적으로 노출되지 않도록 배치
③ UART, JTAG - UART (RX, TX, VCC, GND)핀과 JTAG (TDI, TDO, TCK, TMS, TRST)핀이 쉽게 노출되지 않도록 기판 위의 물리적 배열을 고려, 식별이 용이하게 하는 라벨(legend symbol name)을 지움

개발 단계에서는 JTAG(Joint Test Action Group), ICSP(In Circuit Serial Programming) 등의 하드웨어 프로그래머 또는 디버거를 이용하여 펌웨어 추출 및 프로그래밍(펌웨어를 변경하여 기록하는 것)을 수행할 수 있으나 출시 후에는 이러한 기능이 제한되어야 한다. 다음은 JTAG핀 표준 사양을 설명하고 있다. 칩 제조사는 테스트 상호호환성을 위해 JTAG핀을 하드웨어에 포함하고 있으므로 별도 제한 설정을 하지 않는 한 JTAG 접근 경로는 활성화되어 있다.



<그림 4-10> JTAG의 핀 구조

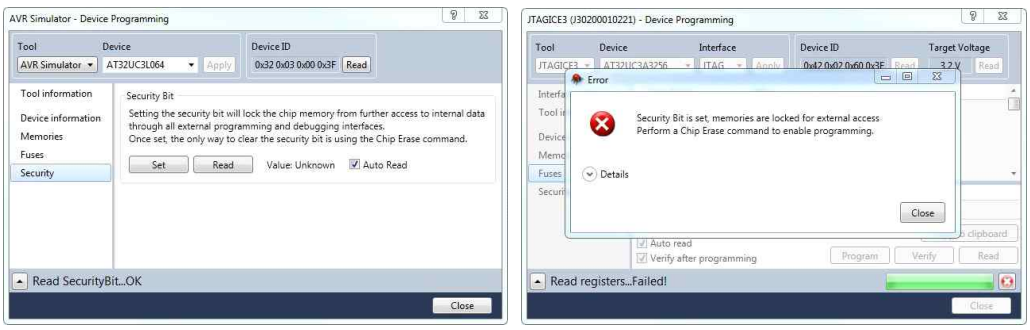
플래시 메모리는 프로세서가 엑세스할 수 있도록 읽기, 쓰기, 삭제 명령을 SPI(Serial Peripheral Interface) 프로토콜 형태로 제공한다. 일반적으로 플래시메모리의 데이터는 프로세서를 통해서 접근할 수 있다고 알고 있으나 메모리 드라이버 수준의 로우 레벨 프로그래밍 방법을 알 수만 있으면 프로세서의 도움 없이 플래시 핀에 직접 연결하여 데이터를 추출 또는 변경할 수 있다. 다음은 칩 제조사의 데이터시트를 참고하여 플래시 메모리 핀으로부터 직접 데이터를 추출하는 공격 사례를 보여주고 있다.

<그림 4-11> 물리적인 메모리 액세스

위와 같은 두 가지 유형의 물리적 메모리 공격을 방어하기 위해서는 하드웨어에서 지원하는 보안 설정을 해야 한다.

① JTAG 디버그 기능을 사용할 수 없도록 설정
② 칩 삭제(erase) 시에 EEPROM에 저장된 내용을 보호하기 위한 설정
③ 엔지니어 모드 진입 시 필요한 암호키 설정
④ 내부 데이터를 보호하기 위한 OTP(one time programming) 영역 사용

JTAG 접근 제한을 위해서는 칩 제조사에서 제공하는 프로그래밍 도구를 사용하여 퓨즈 비트(fuze bit)를 활성화해야 한다.



<그림 4-12> AVR 계열 시큐리티 비트 설정

※ http://www.atmel.com/webdoc/atmelstudio/atmelstudio.section.securitybit.html

플래시 메모리의 OTP 영역은 메모리 제조사의 데이터시트를 참고하여 사용할 수 있으며 한 번만 저장할 수 있는 영역이므로 제품 개발 공정 마지막에 수행하는 것이 좋다.

11.1.6 Status Register Protect (SRP1, SRP0)

The Status Register Protect bits (SRP1 and SRP0) are non-volatile read/write bits in the status register (S8 and S7). The SRP bits control the method of write protection: software protection, hardware protection, power supply lock-down or one time programmable (OTP) protection.

SRP1	SRP0	/WP	Status Register	Description
0	0	X	Software Protection	/WP pin has no control. The Status register can be written to after a Write Enable instruction, WEL=1. [Factory Default]
0	1	0	Hardware Protected	When /WP pin is low the Status Register locked and can not be written to.
0	1	1	Hardware Unprotected	When /WP pin is high the Status register is unlocked and can be written to after a Write Enable instruction, WEL=1.
1	0	X	Power Supply Lock-Down(1)	Status Register is protected and can not be written to again until the next power-down, power-up cycle.(2)
1	1	X	One Time Program(1)	Status Register is permanently protected and can not be written to.

<그림 4-13> 메모리의 OTP(one time programming) 사용을 위한 매뉴얼