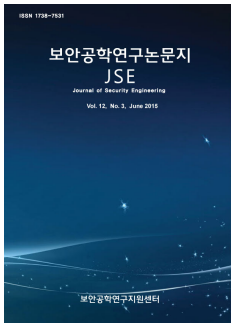



보안공학연구논문지
Vol.12 No.3

ISSN : 1738-7531(Print)

# 스마트 워치 보안 취약점 및 해결 기법에 대한 고찰

이융희, 김현성

To cite this article : 이융희, 김현성 (2015) 스마트 워치 보안 취약점 및 해결 기법에 대한 고찰, 보안공학연구논문지, 12:3, 191-206

보안공학연구논문지 *Journal of Security Engineering*
Vol.12, No.3 (2015), pp.191-206
*http://dx.doi.org/10.14257/jse.2015.06.06*

# 스마트 워치 보안 취약점 및 해결 기법에 대한 고찰

이융희[1], 김현성[2]

# Remarks on Smart Watch Security Vulnerability and Solution

Yunghee Lee[1], Hyunsung Kim[2]

요 약

본 논문에서는 스마트 워치에 대한 기술적 현황과 보안 취약점 및 해결책에 대한 연구 현황을 고찰한다. 먼저 애플, 삼성, 소니 등의 다양한 회사에서 제작된 스마트 워치의 기술적 현황을 운영체제와 기능 관점에서 분석한다. 특히, 이들 시스템의 보안 취약점을 다양한 관점에서 분석하고, 각 취약점에 대한 보안 해결책들에 대한 기존 연구의 현황을 살펴본다. 마지막으로 우리는 향후 스마트 워치 보안에 관한 연구 방향을 제시한다.

핵심어 : 정보보호, 웨어러블 컴퓨팅, 스마트 워치, 보안 취약성, 보안 기법

Abstract

The purpose of this paper is to provide remarks on the current technical status, security vulnerabilities, and security solutions on smart watch. First of all, we will review the current technical status on smart watch focused on the operating system and functionalities from various companies including Apple, Samsung, Sony, and so on. Especially, this paper will provide the previous research work analyses on security vulnerabilities by following the classification and security solutions on them. Finally, we will provide the future research directions on smart watch security.

Keywords : Information security, Wearable computing, Smart watch, Security vulnerability, Security scheme

## 1. 서론

2012년부터 기존 스마트폰 제조사들은 스마트폰 시장에서의 성능 경쟁이 곧 한계에 직면할 것을 예상하고 웨어러블 기기(Wearable Device)에 주목하기 시작했다. 웨어러블 기기의 역사는 반세기 전부터 이미 시작되었다. 1960년에 MIT 미디어랩에서 초기 부착형 타입의 웨어러블 컴퓨팅에

접수일(2015년04월13일), 심사의뢰일(2015년04월14일), 심사완료일(1차:2015년04월30일)
게재확정일(2015년06월04일), 게재일(2015년06월30일)

[1]702-701 경상북도 경산시 하양읍 가마실길 50 경일대학교 사이버보안학과.
email: dldbdgml99@naver.com
[2](교신저자) 702-701 경상북도 경산시 하양읍 가마실길 50 경일대학교 사이버보안학과.
email: kim@kiu.ac.kr

* 본 논문은 2014년 융합/스마트/클라우드 컴퓨팅 학술대회의 "스마트 워치 보안 취약점 및 해결 방안에 대한 고찰"을 확장한 논문입니다.

*ISSN: 1738-7531 JSE*

대한 연구가 시작되었고, 1961년에는 Thorp와 Shannon이 룰렛의 휠을 예측하는 데 사용할 수 있는 최초의 웨어러블 기기를 개발하였다[1]. 1980년대 후반에는 웨어러블 기기가 미군의 군복으로 채택되었고, 1990년대에는 전자기기가 점차 경량화 되면서 각종 산업에서 본격적으로 적용이 가능해졌다[2]. 21세기 들어서는 의료와 엔터테인먼트를 목적으로 하는 웨어러블 기기들이 본격적으로 등장하였는데, 최초의 상업 의류형 웨어러블 재킷인 ICD+가 필립스(PHILIPS)와 리바이스(Levi's)의 공동 개발로 출시되었다. ICD+는 음성인식 기능을 활용하여 전화를 거는 기능과 전화가 오면 재생 중이던 MP3가 자동으로 멈추는 기능 등을 가졌다. 또한 Nike+iPod 역시 같은 시기에 출시되었고, 이는 나이키(Nike)와 애플(Apple)의 합작으로 사용자의 움직임을 자동으로 동기화해 주는 스포츠 킷이었다. 2009년에는 Glacier Computers에서 W200 웨어러블 컴퓨터를 출시하였는데, 이것은 최근에 출시된 스마트 워치(Smart Watch)와 가장 가까운 디자인으로 손목에 밴드형으로 착용하여 주로 응급 상황에서 손을 자유롭게 사용하면서 많은 정보를 찾을 수 있도록 디자인되었다. 2009년 애플의 주도로 시작된 컴퓨터의 소형화 및 경량화와 무선 인터넷의 발전은 웨어러블 기기의 발전에 많은 영향을 끼쳤다[3-6]. 구글은 구글 글래스(Google Glass)라는 헤드 마운티드 디스플레이(Head Mounted Display, HMD)가 장착된 웨어러블 기기를 개발하였다[7]. 삼성전자는 스마트 워치인 갤럭시 기어(Galaxy Gear)를 개발하였다[8].

본 논문에서는 다양한 종류의 웨어러블 기기들 중 스마트 워치에 대한 각종 보안 위협과 현재까지 진행된 스마트 워치 보안 관련 연구에 대해서 살펴본다. 이렇게 스마트 워치에 대한 보안 위협과 이를 위한 보안 기술들에 대한 분석을 제시함으로써 향후 개발될 스마트 워치에서 고려해야 할 보안 요구사항을 도출하고, 현재까지 연구되지 못한 스마트 워치를 중심으로 한 보안 기법에 대한 연구 방향을 제시하고자 한다.

## 2. 스마트 워치 동향

본 장에서는 기 개발된 스마트 워치들의 특징에 대해서 살펴본다. [표 1]은 대표적인 스마트 워치의 운영체제 현황을 보여준다. 디스플레이의 크기나 해상도 등에 대한 정보는 본 논문에서 다루는 주제와 큰 관련이 없으므로 제외한다.

애플 워치는 외관과 기본 기능 소개가 있었을 뿐, 자세한 정보는 아직 공개되지 않았다. 이 기기는 애플에서 스마트 워치에 맞게 개발한 Watch OS라는 운영체제를 탑재하고 있다. 애플 워치는 다른 iOS 기기들과 블루투스를 통해 연동되어 부가적인 기능을 제공한다[9]. 반면 삼성 기어2는 USIM(Universal Subscriber Identity Module, 범용 사용자 식별 모듈)을 장착하여 독자적으로 작동한다. 즉 다른 스마트폰 없이 스스로 전화를 걸거나 메시지를 보내는 등의 기능을 할 수 있다[8].

삼성 기어핏은 애플 워치와 기능적으로 비슷하다. 삼성 기어핏은 다른 안드로이드 기기들과 연동할 수 있다. 기어2와 기어핏은 타이젠 운영체제를 적용하였다. 기어핏이 처음 출시될 때는 RTOS(Real Time Operating System, 실시간 운영체제) 커스텀 웨어러블 운영체제를 사용했으나,

이후 타이젠 운영체제를 사용하도록 펌웨어 업그레이드를 제공한다. 타이젠은 인텔과 삼성전자 등 IT 개발업체들이 새롭게 개발한 오픈 소스 모바일 운영체제인데, 안드로이드가 JAVA를 기반으로 한 것과 달리 HTML5와 C++ 을 기반으로 한다[10]. 삼성전자가 스마트 워치 제품에 타이젠 운영체제를 채용한 것은 커스터마이징에 제한이 있는 안드로이드 운영체제로부터 탈피하고자 함이다. 삼성전자는 스마트 워치 외에도 타이젠을 탑재한 스마트폰인 삼성Z의 출시 계획을 발표했다[11].

LG G워치R은 구글 안드로이드 웨어 운영체제를 탑재하였다. 안드로이드 웨어는 구글에서 웨어러블 제품에 특화되도록 수정한 안드로이드 운영체제인데, 안드로이드 버전 4.3 이상에서 사용할 수 있다[12]. G워치R을 포함한 대부분의 스마트 워치는 안드로이드 웨어 운영체제를 탑재하고 있다[13]. 소니 스마트 워치3 역시 안드로이드 웨어 운영체제를 탑재하였으며, 현재 아직 출시일이 확정되지 않은 상황이다. 이전 제품인 스마트 워치2는 안드로이드 웨어가 아닌 안드로이드를 운영체제로 탑재하였다[14].

[표 1] 스마트 워치 운영체제
[Table 1] Operating Systems on Smart Watch

| 스마트 워치 | 운영체제 | 주요 기능 |
|---|---|---|
| 애플 워치 | Watch OS | - 문자, 메일 읽고 답장하기 - 전화 걸고 받기<br>- 심박센서를 통한 심박수 측정 - 이동거리 계산(GPS와 Wi-Fi 활용) |
| 삼성 기어2 | 타이젠 | - 문자, 메일 읽고 답장하기 - 전화 걸고 받기<br>- 심박센서를 통한 심박수 측정 - 이동거리 계산(GPS와 Wi-Fi 활용)<br>- 만보계 기능(가속도센서 활용) - 독자적인 통신 기능(USIM 칩 장착) |
| 삼성 기어핏 | RTOS, 타이젠 | - 문자, 메일 읽기 - 전화 받기<br>- 이동거리 계산(GPS와 Wi-Fi 활용) - 만보계 기능(가속도센서 활용) |
| LG G워치R | 구글<br>안드로이드 웨어 | - 심박센서 - 가속도, 자이로, 나침반 센서<br>- 기압센서 |
| 소니 스마트<br>워치3 | 구글<br>안드로이드 웨어 | - 블루투스 - 가속도, 나침반 센서<br>- GPS - 마이크 |

이들 모두 블루투스를 지원하여 특정 스마트폰 제품군과 연동이 가능하다. 또한 마이크나 가속도계, 심박센서 등 사용자의 건강이나 활동 상태를 측정하는 각종 센서가 탑재되어 독자적인 기능이 가능하도록 설계되었다.

## 3. 스마트 워치의 보안 위협

현재까지 스마트 워치에 대한 해킹 사례는 보고된 바가 없다. 이는 아직 스마트 워치 자체가 널리 보급되지 않았고 해킹에 대한 수요도 없기 때문으로 생각된다. 하지만 스마트 워치가 보급되고 활성화 된다면 이에 대한 해킹 시도는 분명히 많아질 것이다.

스마트 워치 제조사들은 대부분 이전에 이미 스마트폰 개발을 통해 모바일 스마트 기기의 구축

경험이 있으므로 스마트 워치를 제작할 때에도 다른 모바일기기의 경험을 바탕으로 보안을 고려했을 것이다. 이런 관점에서 보면 구글의 안드로이드 웨어를 운영체제로 채택한 LG G워치R과 소니 스마트 워치3는 비교적 보안 상 안전하다고 볼 수 있다. 또한 삼성의 스마트 워치군이 채용한 타이젠(Tizen)은 첫 상용화이기 때문에 보안 상 문제가 많을 것이다.

안드로이드 웨어를 채택한 스마트 워치는 기존의 안드로이드 스마트폰에 대한 악의적인 접근, 해킹 등에 그대로 취약할 가능성이 높다. 안드로이드 운영체제를 웨어러블 기기에 맞게 최적화 하였지만 아직 초기 버전인 만큼 기존 안드로이드 운영체제가 가지고 있던 여러 가지 특성들을 공유하고 있다. 이는 애플의 Watch OS에도 똑같이 적용된다. 타이젠의 경우에는 응용 프로그램의 생명주기 전체적으로 보안에 상당한 중점을 두고 설계하였다고 발표하였다. 타이젠은 타이젠 SDK(Software Development Kit)를 사용하여 만들어진 응용 프로그램이 스토어에 등록될 때 보안 분석을 통해 한 번 보안 체크를 수행하고, 응용 프로그램이 사용자 기기에 설치된 이후에 바이러스 탐지 기능이 지속적으로 작동한다. 또한 응용 프로그램이 실행될 때 역시 리눅스 커널 단에서 지원하는 스맥(Smack) 기술로 허용하지 않은 접근을 차단하거나 위협 요인을 격리하는 샌드박싱(Sandboxing) 기술 등을 통해 보안을 강화하였다[15]. 하지만 타이젠은 출시된 지 얼마 안 되어서 이에 대한 보안 이슈들 역시 아직까지 확실하게 검증되지 않은 상태이다. 그러므로 다음 장에서는 각 스마트 기기에 대한 알려진 보안 이슈들에 대해 살펴본다.

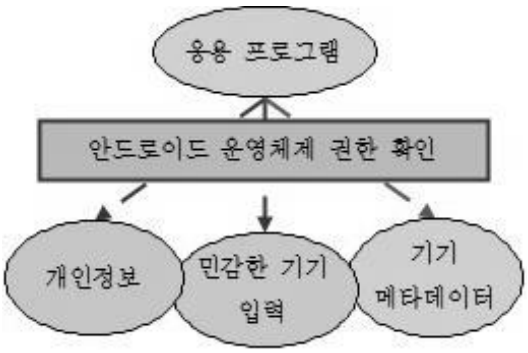
### 3.1 안드로이드 보안 이슈

안드로이드 기반의 기기들은 제조사에서 임의로 운영체제를 커스터마이징 할 수 있기 때문에 시스템의 구조도 굉장히 다양하다. 안드로이드는 운영체제 자체가 가진 취약점과 PC 환경에 비해 취약한 웹 브라우저 등으로 인해 해킹 시도가 굉장히 많았다. 안드로이드 스마트폰이 본격적으로 보급되기 시작한 2009년부터 지속적으로 안드로이드 보안에 대한 연구가 활발하게 진행되었다[16].

안드로이드는 역공학 공격에 취약하고, 그 이유는 안드로이드의 특성에 있다. 안드로이드 응용 프로그램의 달빅(Dalvik) 바이트코드는 자바와 특성이 비슷하여 실행 파일로부터 소스 코드를 거의 그대로 복원할 수 있다. 특히, 이렇게 복원된 코드 분석을 통한 불법적 응용 프로그램의 복제가 가능하다. 또한, 구글은 LVL(License Verification Library)를 제공하는데, LVL을 통해 응용 프로그램을 실행했을 때 서버에 접속하여 사용자에 대한 확인 절차를 거치도록 하였다. 하지만 역공학 공격을 통해 소스를 분석하고, LVL 관련 라이브러리 호출 코드를 수정하면 이러한 보안 시스템을 무력화 할 수 있다[17]. 또한, 안드로이드에서는 응용 프로그램 내에서 브라우저를 호출해 웹페이지의 내용을 표시하거나 제어하기 위해 WebView를 구성요소로 사용한다. 특정 페이지에 접속 시도 시 해당 페이지 내에서 URL(Uniform Resource Locator)을 포함한 메시지를 사용자에게 보내 결과적으로 응용 프로그램 내에서 악의적인 웹페이지를 로드시킬 수 있다[18].






[그림 1] 안드로이드 권한 확인 과정
[Fig. 1] Android Permission Check Process

안드로이드 개발자는 응용 프로그램의 접근 권한을 명시할 수 있다. 이렇게 명시된 권한은 [그림 1]과 같은 권한 확인 과정을 통해 동작되는데, 응용 프로그램이 설치될 때 사용자에게 팝업을 띄워 해당 응용 프로그램의 실행 가능 권한 목록을 보여주고, 이 권한에 대한 동의를 묻는다. 이 과정에서 사용자가 동의 버튼을 누르지 않으면 설치를 취소할 수밖에 없다. 즉, 권한 설정에 있어서 세분화된 권한 설정이 제시되지 못하므로 응용 프로그램이 필요 이상의 권한을 가질 수 있다[21]. 또한 안드로이드에서 응용 프로그램 간 데이터 교환 시, 일시적으로 접근하고자 하는 응용 프로그램의 권한을 얻을 수가 있는데, 이러한 보안 정책을 이용하는 공격 기법으로 권한 상승 공격(Privilege Escalation Attack)이 가능하다[20]. 권한 상승 공격은 해당 응용 프로그램이 가지지 않은 권한을, 필요한 해당 권한을 가진 다른 응용 프로그램을 이용하여 획득하는 공격 기법이다. 예를 들어 사용자가 카메라로 촬영한 사진에 접근하기 위해선 사진에 접근할 수 있는 API(Application Programming Interface)를 가진 다른 응용 프로그램의 취약점을 이용하여 API를 획득하는 과정을 거친다. 안드로이드에서는 응용 프로그램 간에 직접적인 데이터 교환이 가능하기 때문에 이런 권한 상승 공격이 가능하다.


[그림 2] 루팅 중인 안드로이드 스마트 폰
[Fig. 2] Android Smart Phone on Rooting

또한 [그림 2]에서 보여주는 바와 같이 안드로이드의 가장 흔한 보안 취약점은 바로 루팅(Rooting)이다. 이는 iOS 기반 기기의 탈옥(Jailbreak)과 비슷한 개념으로, 사용자가 운영체제의 루트 권한을 얻어내는 것을 말한다. 안드로이드에서 제한하는 기능이나 성능을 이용하기 위해서 사용자가 자의적으로 행하는 경우가 많다. 하지만, 이러한 루팅을 통한 안드로이드의 루트 권한을 악용하면 사용자가 의도치 않은 과금을 하게 만들거나 개인정보를 빼내는 등 심각한 문제를 야기할 수 있다. 또한 사용자의 운영체제를 악의적인 공격자가 자신의 지배하에 두어 DDoS(Distribution Denial of Service) 공격 등에 활용할 수 있다. DDoS 공격의 주체가 되는 것은 현재까지는 좀비PC가 대부분인 것으로 알려져 있지만, 스마트폰 등 모바일 기기의 통신 속도와 성능이 향상됨에 따라 모바일 기기들마저 DDoS 공격의 수단으로 활용되고 있다. PC와 달리 스마트폰과 같은 모바일 기기는 항상 켜져 있으며 네트워크에 연결되어 있는 특성으로 인해 DDoS 공격에 악용되면 훨씬 더 강력한 공격을 야기할 수 있다[23].

안드로이드 기반의 모바일 기기가 DDoS 공격에 이용될 수 있다는 위험성은 여러 차례 제기된 바가 있고, 악성코드나 바이러스 등에 감염되어 임의의 공격자가 악의적인 목적으로 제어할 수 있는 스마트폰을 좀비 스마트폰이라고 한다. 물론 스마트폰의 모든 권한이 공격자에게 있기 때문에 DDoS 공격이 아니라도 개인정보 유출 등의 공격 역시 가능하다[27]. 또한, 안드로이드 기반의 모바일 기기, 특히 스마트폰은 스미싱(SMiShing) 공격에 취약하다. 스미싱이란 SMS(Short Message Service)와 Phishing이 합성된 단어로, SMS를 통해 위변조된 웹페이지의 URL을 흥미로운 문구와 함께 보내 수신자가 무심코 해당 웹사이트를 방문하게 만들어 악성코드를 설치하거나 개인정보를 빼내는 수법을 말한다[28]. 사실 이러한 스미싱 공격에 대한 명확한 해결책은 아직까지 제시되지 않고 있는 상황이다. 기본적으로 사용자가 주의를 해야 해결될 수 있는 문제이지만 날이 갈수록 문자메세지의 내용이 정교해짐에 따라 사용자들 역시 정부 기관에서 발송한 문자메세지인지 스미싱인지 구분하기가 힘들다.

안드로이드 운영체제의 보안 이슈들을 해결하기 위해 많은 연구가 진행되었고, 실제로 그 중 많은 부분이 최신 버전 운영체제에 적용 되었다. 이에 대한 내용은 Ⅳ장에서 다룬다. 이러한 연구 결과로 도출된 보안 기법들은 안드로이드를 토대로 웨어러블 기기에 최적화하여 설계한 안드로이드 웨어 운영체제에도 동일하게 적용될 것으로 예상되나, 분명히 지금까지 없었던 새로운 공격 방식이 생겨날 것이므로 그것에 충분히 대비하는 것이 바람직하다.

### 3.2 iOS 보안 이슈

iOS 기반의 기기들은 안드로이드에 비해 비교적 안전하다는 평가를 받고 있다. 이는 iOS에서 응용 프로그램 계층의 모든 활동을 샌드박스(Sandbox) 상에서 실행되기 때문이다. 샌드박스란 개별 응용 프로그램이 시스템 리소스나 메모리에 접근하지 못하게 하거나, 다른 프로세스의 데이터에 접근하지 못하게 하는 환경을 말한다. 응용 프로그램의 홈 디렉토리 외의 경로에 접근하는 것을

원천적으로 차단하며, 카메라, GPS 등의 하드웨어 리소스를 사용할 수 있도록 인터페이스 클래스를 제공한다. 하지만 이 때문에 폐쇄적이라는 비판을 많이 받아오고 있으며, 버전을 업데이트하면서 다른 시스템 리소스를 이용할 수 있는 인터페이스를 개방함으로써 이를 해결하고 있다[32].

[표 2] 앱스토어 등록 결격 사유
[Table 2] The Reject Reasons from Appstore Registration

| 분 류 | 등록 결격 사유 |
|---|---|
| UI 관련 | - 아이콘과 리뷰 화면에 수영복 등장<br>- 아이폰, 아이팟, 아이팟 용 이어폰과 유사한 이미지를 사용하는 경우<br>- 폴라로이드 사진을 연상할 수 있는 이미지를 사용(저작권 침해)<br>- 알림창을 사용자의 입력을 받을 수 있도록 변경한 경우<br>- 여성의 가슴골이 보이는 경우<br>- 성적으로 자극할 수 있는 내용이 존재하는 경우<br>- 어플리케이션 소개에 어플리케이션 판매 수익금을 기부하겠다는 내용이 존재하는 경우<br>- 소개글에 “이런 기능은 애플 SDK 정책상 불가능 합니다”라고 쓰여져 있는 경우<br>- App name과 Product name이 다르게 표시되는 경우<br>- Play, Pause 버튼을 미디어 플레이어와 슬라이드 외에 사용한 경우<br>- 웹 뷰로 웹 페이지를 불러올 때, 로딩 이미지를 구현하지 않은 경우<br>- 스티브 잡스 사진 및 패러디 사진을 사용하는 경우<br>- 비속어 단어를 사용하는 경우<br>- 프로그램 설명이나 이미지 등에 가격 정보를 적는 경우 |
| 시스템 관련 | - 진동 기능이 들어갈 때 아이팟에서 처리를 하지 않는 경우<br>- 기기의 고유 번호나 핸드폰 번호를 화면에 노출<br>- 무형의 컨텐츠를 애플 결제 시스템 외에 다른 방식(사파리)을 통한 결제<br>- 표준으로 지원하지 않는 API를 사용하여 개발<br>- 탭 바에 별 아이콘이 즐겨찾기 메뉴가 아닌 다른 메뉴로 변경된 경우<br>- 게임 스코어 및 내용을 아이폰에서 서버로 전송할 때, 사용자의 동의를 받지 않은 경우<br>- 어플리케이션 테스트 시 다운 현상이 발생하는 경우 |
| 편의성 관련 | - 어플리케이션 기능이 부실한 경우<br>- 기능 미 구현<br>- 사용법이 어렵다고 판단되는 경우<br>- 설명문과 실제 기능이 다른 어플리케이션인 경우 |
| 기타(법적 규정 등) | - 컨텐츠 중에 특정 업체에 대한 비방이 존재하는 경우<br>- 특정 이미지를 표절하여 유사하게 사용하고 있다고 판단되는 경우<br>- 스크린샷으로 인한 거절(저작권, 상표권, 초상권, 성인/폭력 등)<br>- 패러디 이미지를 사용한 경우 |

이러한 특징들로 인해 iOS 기기들은 피싱 사이트나 악성코드 등으로부터 안전하다. 안드로이드는 특정 웹페이지에서 자동으로 악성 응용 프로그램을 설치하거나 시스템의 리소스를 가져와 민감한 개인정보를 빼내는 공격이 가능하다. 하지만 iOS 기반의 기기들에서는 인터넷을 탐색하는 기본 프로그램인 Safari에서도 시스템 리소스에 함부로 접근하는 것을 제한하기 때문에 개인정보를 훔칠 수가 없다. 또한 허가되지 않은 응용 프로그램의 설치를 차단하기 때문에 악성 응용 프로그램이 설치될 우려도 없다. iOS 기반의 기기들은 응용 프로그램을 애플에서 운영하는 앱스토어에서만 찾

고 설치할 수 있는데, 앱스토어는 플레이스토어와 달리 응용 프로그램 등록 신청을 하면 일주일 정도의 기간 동안 심사를 한 이후에 통과된 것들만 실제 스토어에 등록한다. 이 심사의 기준에 대해 자세히 밝혀진 사항은 없지만, 현재 개발자들의 경험을 통해 알려진 응용 프로그램 등록 결격 사유는 [표 2]와 같다[33].

또한, 애플에서는 기본적으로 iOS 응용 프로그램에서 멀티테스킹과 네트워크 사용, 이벤트 탐지 등을 제한함으로써 악성코드의 활동을 막았다. 멀티테스킹의 경우, 대부분의 악성코드가 백그라운드 상에서 사용자 몰래 활동을 하므로 응용 프로그램이 백그라운드에 있을 때 할 수 있는 활동을 제한한다. 또한 네트워크의 사용을 제한함으로써 DDoS를 발생시키거나 다른 좀비PC를 만드는 데 이용되는 것을 막는다. 이러한 정책으로 인해 개발자가 자유롭게 개발하지 못하고 사용자의 입장에서도 다양한 기능을 제공받지 못하는 단점이 있지만 원천적으로 해킹에 대한 위험을 막는 목적에 대해서는 성공했다는 평가를 받는다. 하지만, 이러한 모든 보안 정책은 iOS 기기가 애플에서 제공받은 순수한 운영체제를 사용한다는 가정 하에 성립된다. 폐쇄적인 정책으로 인한 불편함을 해결하기 위하여 몇몇 해킹 팀에서 iOS 기기의 탈옥툴을 제작하여 배포하였다. 탈옥이란 기본적으로 iOS에서 제한하는 기능들을 펌웨어나 커널을 수정함으로써 사용 가능하도록 허용하는 것을 말한다. 탈옥을 한 iOS 기기는 앱스토어 이외의 웹사이트 등에서 애플에서 허가하지 않은 응용 프로그램을 자유롭게 설치하고 사용할 수 있다. 그리고 애플에서 제한한 권한을 모두 획득하여 어떤 경로에든 접근하고 파일을 보거나 수정할 수 있게 된다. [그림 3]에서 보여주는 바와 같이 탈옥을 함으로써 애플에서 제한한 기능들을 사용할 수 있게 되므로 2011년을 기준으로 전세계의 iOS 기반 기기들 중 15% 정도가 탈옥을 한 것으로 추정되는 시기도 있었다.


[그림 3] 탈옥된 아이폰

[Fig. 3] Jailbreaked iPhone

하지만 iOS6, iOS7 등 운영체제가 버전을 거듭하면서 탈옥한 iOS 기기에서만 가능했던 기능들 중 일부를 애플에서 순정 운영체제 상에서 사용할 수 있도록 열어주었고, iOS 탈옥에 사용되던 방식들 역시 점차 애플이 기술적으로 차단함에 따라 현재는 탈옥을 한 iOS 기기를 사용하는 사용자가 5% 이하로 소수이다[34].

기본적으로 iOS 기기를 탈옥하면 애플에서 적용한 대부분의 보안 시스템이 무력화된다. 따라서 애플은 사용자들에게 기기의 프로그램을 임의로 변조하는 행위를 자제하라고 권고하고, 탈옥 툴을 제작하는 해커들이 이용하는 수법을 기술적으로 차단하는 등의 노력을 기울이고 있지만, 그럼에도 새로운 iOS 버전이 업데이트되면 늦어도 수개월 이내에는 그에 대한 탈옥 툴이 개발되는 실정이다. 탈옥 과정을 거친 iOS 기기에 대한 해킹 공격은 굉장히 많이 알려져 있다. 피싱사이트로 유도하여 악성 응용 프로그램을 설치할 수 있는 것은 물론, 스파이웨어를 설치하여 개인 데이터를 훔칠 수도 있다. 그 외에도 수많은 공격 방식이 있지만, 사용자로서 기기의 보안을 지키는 방법은 탈옥을 하지 않고 비밀번호를 다른 사람이 알 수 없도록 복잡하게 설정하는 방법 외에는 딱히 없다. 애플이 iOS의 소스 코드를 공개하지 않고 있기 때문이다.

### 3.3 스마트 워치 보안 이슈

스마트 워치는 항상 사용자의 몸에 착용되어있고 많은 기능을 제공해 주는 만큼 보안에 대해서도 민감하다. 예를 들어 현재 출시된 대부분의 스마트 워치는 의학기기로서 사용될 수 있도록 심박센서 등이 부착되어 있다. 이는 사용자의 생명과 관련된 중요한 부분인데도 불구하고 스마트 워치가 해킹 등의 공격에 놓일 경우 심각한 상황을 야기할 수 있다. 실제로 블랙햇 보안 컨퍼런스에서는 2012년에 특정인의 인슐린 펌프를 조작하여 신체에 치명적인 복용량을 주입할 수 있도록 조작이 가능하다는 사례를 보였다[35]. 이처럼 스마트 워치는 이제 단순한 IT 기기가 아닌 사용자의 생활에 많은 영향을 끼치는 기기가 되어, 스마트 워치에 대한 보안이 더욱 더 중요해졌다.

스마트 워치는 심박센서 뿐만 아니라 가속도 센서, GPS 등을 활용하여 사용자의 일거수일투족을 기록한다. 이에 따라 스마트 워치가 공공기관 또는 군사용으로 쓰이기 위해서는 반드시 충분한 연구가 선행되어야 한다. 스마트 워치는 기존의 스마트 기기들과 통신 방식이나 작동 기능 등이 다르므로, 이전과는 달리 정보보호 위협도 굉장히 다양해질 것으로 예상된다. 실제로 애플은 2010년 6월 발표된 iOS4버전부터 사용자의 위치정보를 저장하고 서버에서 수집한 것으로 알려졌다[36]. 애플은 아이폰 사용자의 위치정보를 'consolidated.db'라는 숨겨진 파일에 저장해 사용자의 위치정보를 1초 단위로 저장한 후 12시간마다 서버로 전송해 왔다. 또한 해당 아이폰을 동기화한 컴퓨터에도 그 위치 정보가 남고 데이터베이스로 누적되었기 때문에 사용자에 따라 몇 년 전의 위치추적 기록이 남기도 하였다. 심지어 이러한 정보는 암호화되지 않은 채 저장되어 컴퓨터가 해킹되면 사용자의 지난 몇 년간의 이동 기록이 고스란히 공격자에게 전해질 수 있게 되었다.

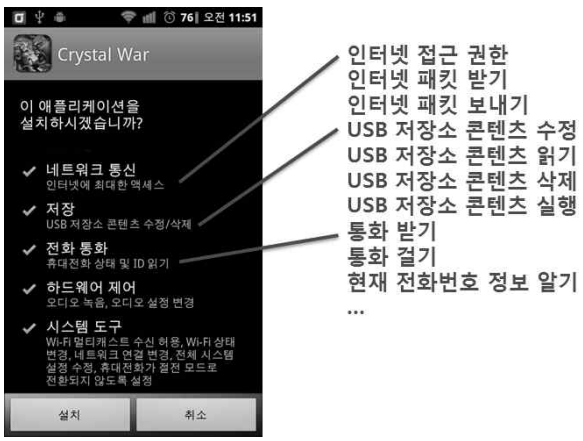
흔히 보안의 수준과 사용자의 편의성은 반비례 관계에 있다고 한다. 스마트 워치의 핵심은 훌륭한 보안을 제공하면서도 해당 보안 시스템이 사용자를 방해하지 않고 서비스를 자연스럽게 제공할 수 있어야 한다는 것이다. 예를 들어 기능을 하나 실행할 때마다 비밀번호, 동의 등을 입력받는 경우 보안은 상당히 향상되겠지만 사용자에게는 불편을 초래할 것이다[37].

이러한 조건을 충족하면서 보안 관점에서의 요구사항도 만족하기 위해 여러 가지 요구사항이

제시되었다. 스마트 워치는 기본적으로 확장성, 가용성, 유연성, 자가 구성을 만족해야 하고, 외부 공격자에 대한 보안 요구사항으로 기밀성, 무결성 등을 만족해야 한다. 또한 내부 공격자로부터의 공격에 대비해 노드에 대한 탄력성, 급격한 성능저하 방지 기능 등을 제공하여야 한다[37].

## 4. 안드로이드 보안 위협에 대한 보안 기법

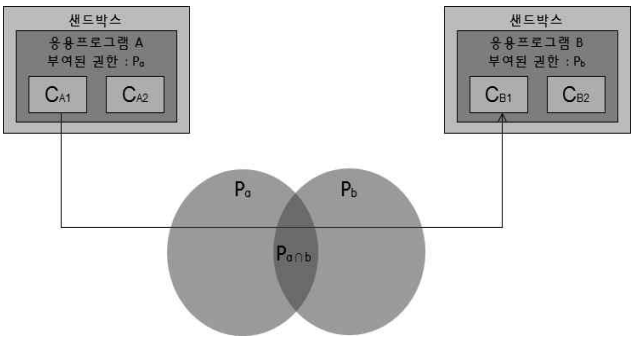
3.1에서 언급한 응용 프로그램 계층에서 이루어지는 두 가지 공격으로부터 사용자를 보호하기 위한 여러 가지 방법이 제시되었다. 첫 번째 대책은 응용 프로그램 암호화를 하는 것이다. 이는 응용 프로그램 실행 파일을 암호화하여 설치하고, 실행 시 복호화하는 방식으로, 응용 프로그램이 복제되더라도 복호화하지 못하면 응용 프로그램의 소스를 공격자가 분석할 수 없다. 이 기법은 아직 운영체제에 적용되지 않았지만 차후 구글에서 새 버전의 안드로이드에 탑재할 가능성이 높은 보안 기법이다. 또 다른 방법으로는 난독화 기법이 있는데, 이는 응용 프로그램을 디컴파일해도 소스 코드의 의미와 제어 흐름을 알아볼 수 없도록 변형하는 기법이다[19-20]. 또한, 안드로이드의 권한이 세분화되지 않아서 일어나는 각종 문제에 대해서는, [그림 4]와 같이 구글이 안드로이드의 권한들을 세분화하여 제공하는 방식으로 해결이 가능하다.




[그림 4] 안드로이드의 권한 세분화

[Fig. 4] Detailed Permission Control on Android

그리고 권한 상승 공격에 대응하기 위해 여러 가지 기법들이 제안되었는데, 그중 대표적인 것으로 Felt등이 제안한 기법이 있다[22]. 이 기법의 핵심은 [그림 5]와 같이 응용 프로그램 간에 데이터를 교환하거나 다른 응용프로그램의 API를 이용하여 접근할 때 두 응용 프로그램이 가진 권한의 교집합을 취하여 제한적인 권한을 사용하도록 한다는 것이다. 예를 들어 악의적인 응용 프로그램




[그림 5] 안드로이드의 권한 상승 공격 보안

[Fig. 5] Security Scheme against Android's Permission Upgrade Attack

이 사진에 접근할 권한을 가지지 않았을 때 사진에 접근할 권한을 가진 다른 응용 프로그램의 API(Application Programming Interface)를 사용하려고 하면 사진에 접근할 권한을 가지지 않은 악의적인 응용 프로그램의 권한과 AND 연산을 적용하여 사진에 접근할 권한이 없어진다. 이러한 공격 기법들을 포함한 권한 관련 공격들은 시스템 계층에서 이루어지는데, 이에 대한 해결책은 이미 다수 제시된 바 있다. Nauman등은 안드로이드 기반의 권한에 관한 보안 해결책을 발표하였다[24]. 이들은 안드로이드의 권한 체계가 세분화되어있지 않음을 지적했으며, 권한 체계를 개선하여 더욱더 세분화한 권한 모델을 제안하였다. 하지만 이러한 해결책을 사용하기 위해서는 안드로이드의 프레임워크 자체를 수정해야 하는 어려움이 따랐다. 그 후, 새로운 권한 시스템을 응용 프로그램 계층에서 적용할 수 있는 기법이 제안되었고, 시스템의 권한 설정을 수정하지 않고도 응용 프로그램이 사용하는 권한을 분석하여 악의적인 응용 프로그램을 가려내는 기법 역시 제안되었다[25-26]. 하지만 정확도가 높지 않아서 실제로 활용되기에는 부족한 점이 많은 실정이다.

안드로이드 운영체제의 루팅에 대한 보안 기법은 현재까지 제시되지 못했다. 대부분의 경우 루팅은 사용자가 편의를 위해 자발적으로 하고 있으며, 운영체제의 제작사인 구글에서도 직접적으로 차단하지는 못하고 있다.

안드로이드 운영체제를 탑재한 스마트폰에 대한 스미싱 공격에 대응해 제안된 기법들이 있는데, 내용기반 필터링(Content-based Filtering), 블랙리스트, 화이트리스트 등의 기법들이다[29-31]. 내용기반 필터링 기법의 경우 문자메시지의 내용을 분석하여 스미싱 공격을 추측하는 방식인데, 단축 URL을 사용하는 등의 방식이 많이 쓰임에 따라 사실상 사용이 어려워졌다. 블랙리스트 기법은 말 그대로 스미싱 공격에 사용된 발신자를 블랙리스트로 등록하여 아예 스미싱 공격 자체를 하지 못하도록 막는 방식이다. 하지만 스미싱 피해 사례가 도착해서 그것을 등록하는 데에 시간이 걸리므로 비효율적이다. 또한 화이트리스트 기법은 가장 강력하고 확실한 방법이지만 사용자의 불편을 야기하는 단점이 있다.

## 5. 고찰

본 논문에서는 웨어러블 기기들 중 스마트워치에 대한 각종 보안 위협과 현재까지 진행된 스마트워치 보안 관련 연구에 대한 분석을 제시하였다. 스마트 워치에 채택될 안드로이드 웨어와 Watch OS는 각각 이들 운영체제가 기반으로 하는 안드로이드와 iOS의 특징을 가질 것으로 예상되며, 그 특징들로 인한 보안 상 취약점들 중 몇 가지는 현재 선행된 연구를 통해 보안기법들이 제시되었다. 안드로이드 환경의 보안 위협 요소와 보안 기법은 [표 3]과 같이 정리할 수 있다. 또한 운영체제에 따른 공격 가능 여부는 [표 4]와 같다.

[표 3] 안드로이드 환경의 보안 위협 요소
[Table 3] Security Threats on Android Environment

| 분 류 | 위협 요소 | 보안 기법 |
|---|---|---|
| 응용프로그램 계층 | 역공학 공격, WebView 취약점 | 난독화, 암호화, 소프트웨어 워터마킹, 소프트웨어 버스마크 |
| 시스템 계층 | 퍼미션, 루팅 | 세분화된 퍼미션 모델, 모바일 IDS, 테인트 분석 |
| 기타 | DDos 공격, 스미싱 공격 | 내용기반 필터링, 화이트리스트/블랙리스트 |

[표 4] 운영체제 별 공격 가능 여부
[Table 4] Available Possibility based on Operating System

| 공격방식 | 안드로이드 | iOS |
|---|---|---|
| 역공학 | 가능 | 가능 |
| WebView | 가능 | 불가능 |
| 퍼미션 | 취약 | 양호 |
| 루팅(탈옥) | 가능 | 가능 |
| DDos | 가능 | 불가능 |
| 스미싱 | 가능 | 가능 |

하지만 스마트 워치가 출시되어 상용화되면 수많은 스마트워치 유저들을 노린 해커들이 이전보다 더욱더 많은 공격 수법을 만들어낼 것이고, 이러한 공격 수법에 발 빠르게 대응하지 못하면 웨어러블 기기의 특성 상 굉장히 민감한 개인정보 유출이 일어날 수 있다. 이를 방지하기 위하여 다양한 요구조건이 제안되었고, 또 현재 제작되는 많은 종류의 스마트워치에 적용되고 있다. 스마트 워치 환경의 보안성을 높이기 위하여 이러한 환경에서의 위협 유형에 대한 체계적인 대응 방식이나 수준 높은 선제적 예방 기법의 연구를 진행하여야 하며, 사용자와 서비스에 대한 안정성 및 신뢰성을 제공하기 위한 보안 기술이 개발되어야 한다. 요즘 대두되고 있는 헬스케어 역시 웨어러블 기기와 밀접한 관련이 있는데, 이와 같이 인간의 건강 또는 생명과 밀접한 역할을 수행하는 웨어

러블 기기가 악의적인 해킹에 노출된다면 큰 인명피해를 발생시킬 수 있다. 또한 스마트워치 자체의 해킹 공격 외에도 스마트워치를 소지한 사용자의 악의로 인한 사생활 침해 등의 문제가 생길 수 있다는 사실도 이슈가 되고 있다[37]. 스마트워치는 항상 손목에 차고 다닌다는 특징으로 인하여 스마트폰에 비해 도청, 도찰 등의 사생활 침해가 일어날 가능성이 높으며, 이를 방지하기 위한 법적 제도나 시스템적 장치가 마련되어야 할 것이다. 또한 각국 정부에서는 웨어러블 기기로 인한 사생활, 개인정보 침해를 막기 위해 관련법을 만들어 사전에 범죄를 예방하여야 한다.



*ISSN: 1738-7531 JSE*

## References


[1] http://www.media.mit.edu/wearables/lizzy/timeline.html.

[2] M. J. Zieniewicz, D.C. Johnson, C. Wong and J. D. Flatt, The Evolution of Army Wearable Computers, IEEE Pervasive Computing. (2002), Vol. 1, No. 4, pp. 30-40.

[3] http://rack.0.mshcdn.com/media/ZgkyMDE0LzA1LzEyLzc4L3RlY2g5LjYzMTI2LmpwZwpwCXRodW1iCTEyMDB4OTYwMD4/9d56ecba/d2a/tech9.jpg/.

[4] http://www.gizmag.com/w200-wearable-computer/11443/, April 10 (2009).

[5] T. H. Kim, M. K. Hwang and H. M. Jung, Current, Future, and Issue of Future Wearable Computing, IITA IT Trends. (2014), Vol. 1637, pp.16-18.

[6] http://ko.wikipedia.org/wiki/착용_컴퓨터, April 29 (2015).

[7] G. David, Google unveils 'Project Glass' virtual-reality glasses, CNN, 2012.

[8] http://www.anandtech.com/show/7785/samsung-announces-tizenbased-gear-2-gear-2-neo-smartwatches, February 23 (2014).

[9] http://www.apple.com/kr/watch/ (2015).

[10] http://ko.wikipedia.org/wiki/tizen January 15 (2015).

[11] http://www.kbench.com/?q=node/136900, July 29 (2014).

[12] http://www.android.com/wear/ (2015).

[13] http://techcrunch.com/2014/08/27/lg-g-watch-r/, August 27 (2014).

[14] http://www.sonymobile.com/global-en/products/smartwear/smartwatch-3-swr50/features/ (2014).

[15] http://www.zdnet.co.kr/news/news_view.asp?artice_id=20140410145656, April 10 (2014).

[16] J. Jang, S. Han, Y. Cho. U. J. Choe and J. Hong, Survey of Security Threats and Countermeasures on Android Environment, Journal of Security Engineering. (2014), Vol. 11, No. 1, pp. 1-12.

[17] T. K. Chawla and A. Kajala, Transfiguring of and Android App Using Reverse Engineering, International Journal of Computer Science and Mobile Computing. (2014), Vol. 3, No. 4, pp. 1204-1208.

[18] E. Chin and D. Wagner, Bifocals: Analyzing WebView Vulnerabilities in Android Applications, Lecture Notes in Computer Science. (2013), Vol. 8267, pp. 138-159.

[19] W. Enck, D. Octeau, P. McDaniel and S. Chaudhuri, A Study of Android Application Security, Proc. of the 20th USENIX conference on Security, (2011) pp. 21-21, Aug. 8-12; San Francisco, USA.

[20] L. Davi, A. Dmitrienko, A. Sadeghi and M. Winandy, Privilege Escalation Attacks on Android, Proc. of the 13th International Conference on Information Security and Cryptology, (2010) pp. 346-360, Dec. 1-3; Seoul, Korea.

[21] A. P. Felt, S. Hanna and E. Chin, Permission Re-delegation: Attacks and Defenses, Proc. of the 20th

USENIX Security Symposium, (2011) Aug. 8-12; San Francisco, USA.

[22] A. P. Felt, M. Finifter, E. Chin, S. Hanna and D. Wagner A survey of mobile malware in the wild, Proc. of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices, (2011) pp. 3-14, Oct. 17-21; Chicago, USA.

[23] T. Bläsing, L. Batyuk, A. D. Schmidt and S. A. Camtepe, An Android Application Sandbox system for suspicious software detection, Proc. of 5th International Conference on Malicious and Unwanted Software, (2010) pp. 55-62, Oct. 19-20; Nancy, France.

[24] M. Nauman, S. Khan and X. Zhang, Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints, Proc. of the 5th ACM Symposium on Information, Computer and Communications Security, (2010) pp. 328-332, Apr. 13; Beijing, China.

[25] J. Jeon, K. K. Micinski, J. A. Vaughan, N. Reddy, Y. Zhu, J. S. Foster and T. Millstein, Dr. Android and Mr. Hide: Fine-grained Security Policies on Unmodified Android, UM Computer Science Department, (2011), pp. 1-14.

[26] D. Barrera, H. G. Kayacik, P. C. Oorschot and A. Somayaji, A Methodology for Empirical Analysis of Permission-Based Security Models and its Application to Android, Proc. of the 17th ACM Conference on Computer and Communications Security, (2010) pp. 73-84, Oct. 4-8; Chicago, USA.

[27] K. Jang, S. CHoi and H. Yeom, Smartphone DDoS Attacks, Information Security. (2011), Vol. 21, No. 5, pp. 65-70.

[28] J. Mu, A. Cui, and J. Rao, Android Mobile Security – Threats and Protection, Proc. of International Conference on Computer, Networks and Communication Engineering, (2013) pp. 683-685, May 23-24; Beijing, China.

[29] Y. J. Won, H. Kim and J. H. Huh, Hybrid Spam Filtering for Mobile Communication, Computers & Security. (2010), Vol. 29, No. 4, pp. 446-459.

[30] G. Xiang, J. Hong, C. P. Rose and L. Cranor, Cantina+: A Feature-rich Machine Learning Framework for Detecting Phishing Web Sites, ACM Transactions on Information and System Security. (2011), Vol. 14, No. 2, article No. 21.

[31] C. Ye, W. Han and Y. Le, Anti-Phishing based on Automated Individual White-list, Proc. of the 4th ACM Workshop on Digital Identity Management, (2008) pp. 51-60, Nov. 8; Berlin, Germany.

[32] Jonathan Zdziarski, Hacking and Securing iOS Applications, O'Reilly Media, (2012).

[33] J. S. Oh, Study on the leakage of personal information through the iOS jailbreak, M. S. Thesis, Chungnam National University, (2012).

[34] http://techcrunch.com/2013/02/04/jailbreaking-is-back-new-evasi0n-software-works-on-most-ios-6-06-1-devices-including-iphone-5/ February 4 (2013)

[35] AhnLab, Imagination becomes reality, IoT, Security Issue & Issue, (2014).

[36] http://arstechnica.com/apple/2011/04/how-apple-tracks-your-location-without-your-consent-and-why-it-matters/, April 21 (2011).

[37] H. S. Park, Privacy issues and implications surrounding the wearable computer, KOITA, (2013).

## Authors

**이융희 (Yunghee Lee)**

2012년 2월 ~ 현재 : 경일대학교 사이버보안학과 학생
2012년 2월 ~ 현재 : 경일대학교 정보융합보안연구소 연구원
2012년 2월 ~ 현재 : 경일대학교 정보보호연구실 연구원
관심분야 : 정보보호, 스마트 기기 보안, 정보융합보안, 스마트그리드 보안, 무선 보안, 암호프로토콜

**김현성 (Hyunsung Kim)**

2002년 2월 : 경북대학교 컴퓨터공학과 박사
2012년 3월~현재 : 경일대학교 사이버보안학과 교수
2010년 2월~현재 : 정보융합보안연구소 소장
2002년 3월~2012년 2월 : 경일대학교 컴퓨터공학부 교수
2012년 9월~2014년 4월 : 경일대학교 학술정보원 원장
2009년 1월~2010년 1월 : 더블린시립대학 컴퓨팅학부 방문교수
관심분야 : 인지무선네트워크 보안, 네트워크 보안, 암호 프로토콜, 암호구현, 정보보호