Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.25, NO.3, Jun. 2015 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2015.25.3.603

안드로이드 UI 이벤트를 이용한 공격 기법 연구

윤 석 언,[1†] 김 민 성,[2] 이 상 진[1‡]
[1]고려대학교 정보보호대학원, [2]금융보안원

A Study on the attack technique using android UI events

Seok-Eon Yoon,[1†] Min-Sung Kim,[2] Sang-jin Lee[1‡]
[1]Graduate School of Information Security, Korea University,
[2]Financial Security Institute

요 약

스마트폰 어플리케이션은 UI(User Interface)로 구성되어 있다. 버튼 클릭, 화면 스크롤 등 사용자가 어플리케이션을 사용하는 과정 중 많은 UI의 변화가 발생하면서 UI 이벤트가 시스템에 전송된다. 이러한 UI 이벤트에는 다양한 정보가 포함되어 있는데, 사용자 입력과 관련된 정보도 함께 포함되어 있다. 안드로이드 환경에서 키로깅 등 기존의 알려진 입력정보 획득 기법은 루팅과 같은 제한적인 환경을 필요로 했으나, UI 이벤트는 일반권한만으로도 접근이 가능하다는 장점이 있다. 또한, 입력정보 보호를 위해 많은 어플리케이션에서 보안키패드를 적용 중에 있으나, 이와는 관계없이 UI 이벤트를 활용한 기법을 통해 입력정보 획득이 가능하다는 것이 실험을 통해 조사되었다. 본 논문에서는 UI 이벤트를 활용한 사용자 입력정보 유출 위협을 증명하고, 이를 응용하여 시나리오를 기반으로 재생공격(Replay Attack)이 가능함을 보이면서 대응 방안을 제시하도록 한다.

ABSTRACT

Smart-phone Applications are consists of UI(User Interface). During using applications, UI events such as button click and scroll down are transmitted to Smart-phone system with many changes of UI. In these UI events, various information including user-input data are also involved. While Keylogging, which is a well-known user-input data acquisition technique, is needed a restrictive condition like rooting to obtain the user-input data in android environment, UI events have advantage which can be easily accessible to user-input data on user privileges. Although security solutions based keypad in several applications are applied, we demonstrate that these were exposed to vulnerability of application security and could be obtained user-input data using UI events regardless of presence of any security system. In this paper, we show the security threats related information disclosure using UI events and suggest the alternative countermeasures by showing the replay-attack example based scenarios.
Keywords: Android Security, AccessibilityEvent, Secure Keypads, UIAutomator, Replay Attack

I. 서 론

현 송금 서비스는 크게 보안성 중심의 스마트폰 뱅킹과 편의성 중심의 간편한 송금 서비스로 나누어 볼 수 있다. 스마트폰 뱅킹에서의 송금 서비스는 보안카드번호 및 공인인증서, 각종 비밀번호를 통해 사용자를 보호하고, 간편한 송금 서비스는 현재 국내에 출시된 서비스 대부분이 가입 시 설정된 비밀번호로 사용자를 보호한다. 또한, 사용자가 입력하는 보안카드번호 및 각종 비밀번호가 외부로 유출되는 것을 예방하기 위해 암호화 등의 기술이 포함된 보안키패드를 송금 어플리케이션에 공통적으로 탑재하고 있다.

접수일(2015년 4월 14일), 수정일(2015년 6월 2일), 게재확정일(2015년 6월 3일)
† 주저자, slaxcore@korea.ac.kr
‡ 교신저자, sangjin@korea.ac.kr(Corresponding author)

반면, 이러한 입력정보 보호를 위한 노력에도 불구하고, 키로깅 및 화면 캡쳐 등 사용자의 입력정보를 획득할 수 있는 여러 공격기술이 발표되고 있다. 다만, 이러한 공격기술 대부분은 루트권한을 요구하므로 어플리케이션이 강력한 루팅탐지 기능을 제공한다면 공격이 성공할 수 없다는 제약이 발생하는데, 안드로이드에서 제공하는 기본 기능을 이용한다면 루트권한 등의 특별한 제약 없이도 입력정보 유출이 가능하다.

본 논문에서는 이를 증명하기 위해 안드로이드에서 기본 제공하는 프레임워크(UIAutomator)를 이용, 송금 서비스 어플리케이션을 대상으로 UI(User Interface) 이벤트가 발생될 때 마다 비밀번호 획득이 가능하다는 것을 증명하고, 더 나아가 사용자 계좌에서 공격자의 계좌로 부당송금이 가능하다는 것을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 입력정보 보호를 위해 현재 국내에서 가장 널리 사용중인 보안키패드에 대해 소개 후, 기존에 알려진 입력정보 획득 기술을 살펴본다. 3장에서는 안드로이드 UI 이벤트와 관련된 배경 지식에 대해 설명 후, 4장에서 UIAutomator를 활용하여 발생할 수 있는 보안 위협에 대해 SMS 및 메신저 어플리케이션을 예제로 설명한다. 5장과 6장에서는 4장에서 설명한 보안 위협을 증명하기 위해 보안키패드가 적용된 국내 송금 어플리케이션을 선별하여 실험을 통해 비밀번호 유출 및 부당송금 위협을 증명한다. 7장에서는 대응방안을 제시하고, 마지막 8장에서는 결론 및 향후 연구방향을 기술한다.

II. 관련 동향 및 연구

본 장에서는 입력정보 보호를 위해 가장 널리 사용 중인 보안키패드 솔루션에 대해 소개 후, 안드로이드 환경에서 기존에 알려진 입력정보 획득 기술을 살펴본다.

2.1 보안키패드 소개

Fig.1.은 어플리케이션에 탑재된 보안키패드의 모습으로서 국내 스마트폰 어플리케이션에서 입력정보 보호를 목적으로 가장 활발히 사용되고 있는 솔루션이다.

보안키패드의 기능으로는 입력정보의 유출·유추 방지가 대표적이다. 유출 방지는 자판에 해당하는 문자를 백그라운드에서 암호화 혹은 치환함으로서 원문 정보가 유출되지 않도록 보호하고, 유추 방지는 자판 배열을 랜덤하게 배치함으로써 터치좌표 로깅 및 훔

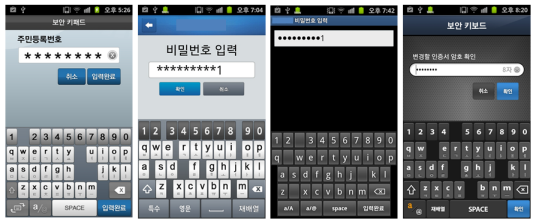


Fig. 1. Secure keypad

쳐보기 등으로부터 보호한다.

보안 기능의 세부적인 구현과 적용된 암호화 방식, 자판 배열의 모습에는 제품마다 다소 차이가 있으나 해당 기능의 목적은 각 제품마다 동일하다. 그러나 규칙성 없이 무작위로 자판배열이 변경됨에 따른 사용자의 불편함을 예상하여 가변 위치의 범위가 다소 제한적이고, 입력되는 마지막 문자를 일정시간 노출시키도록 구현된 점은 보안 측면에서 문제점으로 거론되고 있다[1-4].

2.2 알려진 입력정보 획득 기법

사용자가 입력한 정보를 가로채어 획득할 수 있는 범위는 스마트폰 내부 영역과 네트워크 전송 영역으로 크게 나눌 수 있지만, 중요정보를 다루는 대부분의 어플리케이션은 SSL 등의 암호화 통신이 이루어지므로 본 절에서는 스마트폰 내부 영역에서의 입력정보 획득 기법에 대해 설명한다.

2.2.1 터치 좌표 및 메모리 해킹에 의한 키로깅

키로깅이란 키패드를 통해 입력하는 문자를 식별하여 사용자가 입력한 내용을 획득하는 기법을 의미한다.

먼저 터치좌표 로깅은 Fig.2.에서 보듯이 화면 터치 시에 시스템 내부에서 발생하는 터치이벤트를 가로채어 화면의 X·Y좌표 값을 확인[5-7], 입력정보를 유추하는 기술이다. 일반적인 키패드는 자판이 고정되

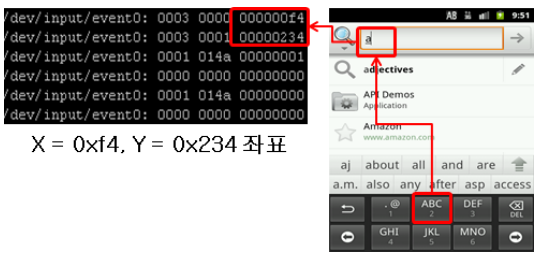


Fig. 2. Make X.Y coordinates of the touch screen

어 있다는 점, 보안키패드의 경우는 자판 배열이 매번 변하지만 앞 절에서 언급했듯이 가변 범위가 크지 않다는 점을 이용하여 입력값 획득이 가능하다.

메모리 해킹은 어플리케이션이 실행되면 관련된 모든 정보는 메모리에 기록된다는 점을 이용하여 메모리의 특정 정보를 획득 혹은 위·변조하는 기술이다. 2013년 HITCON 컨퍼런스에서는 안드로이드 시스템 라이브러리에서 이벤트 처리를 위해 정의된 특정 함수(ProcessKey())의 메모리 주소를 조작하여 입력정보를 가로채는 기술이 소개된 바 있으며[8], PTRACE 시스템 호출(System Call)을 이용하여 실행된 어플리케이션의 메모리 덤프 및 검색을 통해서도 입력 정보 획득이 가능하다[9]. 다만, 메모리 해킹을 위해서는 루트권한이 반드시 필요하다는 전제가 있고, 스마트폰 환경에 따라 메모리 주소가 상이하여 선행분석이 필요하다는 점에서 공격의 효율성이 다소 떨어지는 방법이다.

2.2.2 화면 캡쳐

키패드는 사용자 편의를 위해 터치된 자판의 강조 효과 기능이 있다. 따라서 Fig.3.에서 보듯이 사용자가 비밀번호 등 중요정보 입력 시 터치이벤트를 감지하여 화면 캡쳐가 진행될 경우 입력정보 획득이 가능해진다. 화면 캡쳐 방법으로는 프레임버퍼 출력값을 이미지 포맷으로 파싱하는 방법[10][11]과 안드로이드 API를 이용하는 방법[12]이 있다. 하지만 프레임버퍼를 이용하는 방법은 앞서 언급한 기술과 마찬가지로 루트권한이 필요하고, API를 이용하는 방법은 이미 화면 캡쳐 방지 API[13][14]를 안드로이드에서 제공하므로 쉽게 방어가 가능하다.

Table 1. Main event type on the predefined 'AccessibilityEvent' class

Event Type	Description
TYPE_VIEW_CLICKED	represents the event of clicking on a View like Button, CompoundButton, etc.
TYPE_VIEW_TEXT_CHANGED	represents the event of changing the text of an EditText.
TYPE_VIEW_TEXT_SELECTION_CHANGED	represents the event of changing the text selection of an EditText.

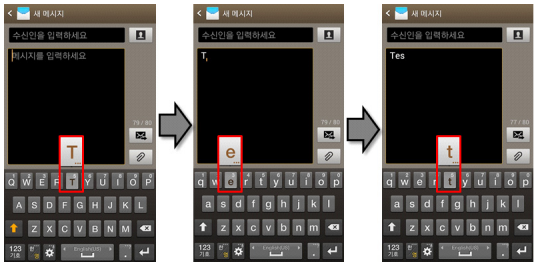


Fig. 3. Example for display capture attack

III. 배경 지식

2장에서 살펴본 것처럼 스마트폰 내에서 사용자 입력 정보를 획득할 수 있는 기술은 다양하지만 루트권한이 필요하거나 일정 부분 대응이 가능함을 알 수 있다.

이 장에서는 기존에 알려진 방식 외에도 UI를 통해 사용자 정보를 획득할 수 있다는 것을 증명하기 위한 배경지식으로서 안드로이드에서 제공하는 UIAutomator 프레임워크를 설명하고 최근 발표된 UI 공격 방식에 대한 소개 및 본 논문에서 사용된 방식과의 차이점을 설명한다.

3.1 UI 이벤트

안드로이드 시스템은 버튼터치, 창 전환 등 UI가 변화할 때마다 특정한 이벤트를 시스템에 전송한다. 이러한 각각의 이벤트 유형과 유형별 세부 정보는 안드로이드의 AccessibilityEvent 클래스에 정의되어 있는데[15], 본 논문에서 언급할 보안위협과 관련된 핵심 UI 이벤트 유형은 Table 1.에, 세부 정보(주요 메쏘드 및 반환값)는 Table 2.에 정리하였다.

Table 2. Main method and return value by processing UI events

Method	Return Value
getClassName()	Gets the class name of the source.
getText()	Gets the text of the source or source's sub-tree.
isPassword()	Gets if the source is a password field.
getContentDescription()	Gets the description of the source.

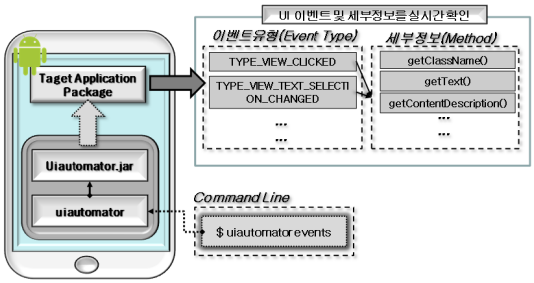


Fig. 4. UI events monitoring

3.2 UIAutomator

안드로이드에서는 어플리케이션 개발과정에서 UI에 대한 테스트를 지원하기 위해 UIAutomator라는 프레임워크를 제공한다[16]. UIAutomator는 관련 라이브러리 및 실행파일(Launcher), Windows GUI 형태의 Viewer로 구성되어 있다.

UIAutomator는 크게 두 가지 기능을 지원한다. 첫 번째로, UI 변화에 따른 이벤트를 실시간 모니터링 할 수 있는 기능을 지원하는데, Fig.4.와 같이 앞 절에서 언급한 AccessibilityEvent 클래스에 정의된 UI 이벤트 및 세부정보를 실시간 확인 가능하다.

두 번째로, 개발된 UI를 자동으로 구동되게 함으로써 테스트의 편의성을 제공한다. Fig.5.는 일련의 과정을 도식화한 것으로서, 앞서 확인한 실시간 UI 이벤트 정보를 토대로 JAVA 개발환경에서 자동화를 구현한 모듈을 스마트폰에 삽입하면 UIAutomator가 실행파일을 통해 테스트를 자동으로 진행한다[17].

특히 스마트폰에 내장된 UIAutomator 실행파일 및 라이브러리 파일은 Fig.6.에서 보듯이 일반 권한으로도 실행 및 읽기가 가능하므로 루트 등의 특별권한이 없이도 사용이 가능하다.

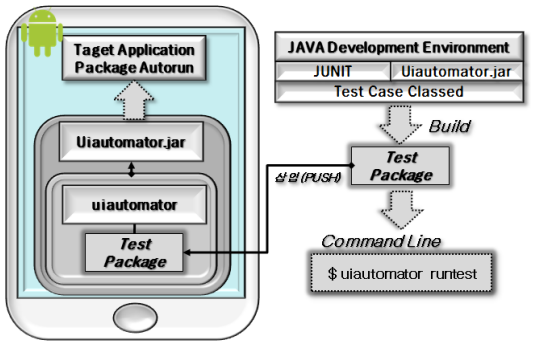


Fig. 5 UI autorun test

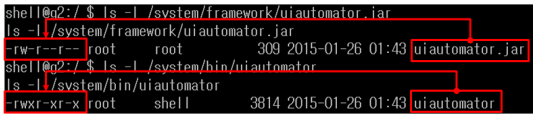


Fig. 6. UIAutomator permissions

3.3 UI State Inference Attack

USENIX 2013 컨퍼런스에서 소개된 기법으로, Fig.7.처럼 스마트폰 화면의 UI 상태 변화를 실시간 관찰하여 공격자가 원하는 UI로 변조 후 사용자가 입력한 정보를 유출하는 공격을 의미한다[18]. 예를 들어, 사용자가 로그인 화면을 클릭하게 될 경우 악성코드는 사전에 제작해 놓은 원본 화면과 동일한 가짜 화면으로 교체하게 된다. 이를 인지하지 못한 사용자는 가짜 화면에 비밀번호를 입력하게 되고, 이때 악성코드가 비밀번호를 가로챌 수 있다고 소개하고 있다.

해당 기법은 안드로이드 어플리케이션 UI 변화를 이용한다는 점은 본 논문과 동일하다. 그러나 UI 변화 감지를 위해 OS의 Shared Virtual Memory 사이즈 변화를 이용한 점은 안드로이드에서 제공하는 기본 프레임워크를 이용한 본 논문과는 기술적 접근 방식에 차이가 있다. 그리고 본 논문에서는 UI 변화에 따른 이벤트 로깅을 통해 입력정보 유출 등의 위협을 제기하였으나, 해당 발표에서는 마치 피싱(Phishing)과 같이 UI 바꿔치기를 통해 입력정보를 유출할 수 있다고 소개한 점 또한 차이가 있다. 더욱이 해당 기법에서는 메모리 사이즈 변화 감지를 위해 statm 명령어가 사용되었는데, 해당 명령어는 타겟 프로세스의 메모리 사이즈를 단순 숫자 형태로 출력해주므로 가짜UI로 바꿔치기 할 시점 판단에 있어서 모호해질 수 있고, 이에 따라 비밀번호를 가로채기 전에 사용자에게 인지될 가능성이 내재되어 있다. 반면, 본 논문에서 사용된 기법은 안드로이드 기본 기능을 이용해 화면의 모든 UI이벤트를 백그라운드에서 모니터링하여 로깅된 비밀번호를 가로챈다는 점에서 공

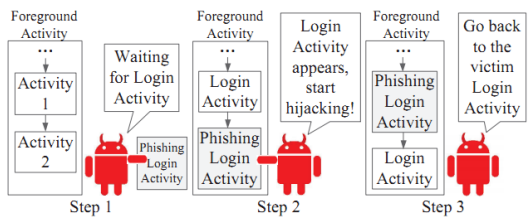


Fig. 7. Workflow of UI state inference attack

격의 정확성과 안정성에서 큰 차이가 있다.

IV. UIAutomator를 이용한 보안 위협

2장에서 설명한 바와 같이 기존의 공격 방식은 루트권한이 필요하기 때문에 사용자가 스스로 루팅을 진행했거나, 혹은 추가적인 공격을 통해 루트권한을 획득해야만 하는 선행조건이 있어야 한다. 3장에서 살펴본 바와 같이 UIAutomator는 일반권한으로도 접근이 가능하다는 점에서 기존의 전제사항을 벗어날 수 있음을 알 수 있으며, 이는 보안측면에서 악용될 소지가 있다. 본 장에서는 이러한 UIAutomator의 특징을 이용하여 정보 유출 및 부당송금 등의 위·변조 위협을 설명한다.

4.1 사용자 비밀정보 유출

사용자는 모든 비밀정보를 키패드로 입력한다. 예를 들어, 금융 어플리케이션에서의 비밀번호 입력이나 메신저 대화 및 SMS 입력 시에도 키패드를 이용한다. 즉, 버튼 터치로 인해 발생하는 UI 변화로 인해 특정 이벤트가 발생하고, 해당 이벤트의 세부정보를 통해 비밀정보가 유출될 수 있다.

Fig.8.은 안드로이드 기기에서 SMS 문자 입력 시 발생한 UI 이벤트 정보를 확인한 결과이다. SMS 입력 시 키패드를 통해 입력되는 순간마다 입력상자(EditText) 영역이 변하고 UI이벤트가 발생하게 된다. 이후 UI이벤트 타입에 따른 세부정보가 기록되면서 SMS 입력 내용도 함께 기록되는 것을 확인할 수 있다.

또한, Fig.9.처럼 메신저 서비스도 SMS와 동일한 이유로 사용자가 입력한 대화내용이 그대로 UI 이벤트 정보에 기록되게 된다. 메신저의 경우 프라이버시 보호가 사회적인 이슈로 부각되면서 대화 내용의 기밀성 보장 여부가 중요 쟁점으로 떠오르고 있다. 스마트폰 내 대화내용 유출 가능성 이슈는 최근 공개된 스마트폰 메신저 보안성 점검[19] 결과에서도 제기되었으나, 이는 루트권한을 통한 메모리 덤프를 이용한 것으로서, UIAutomator를 이용할 경우에는 루트권한 여부에 관계없다는 점에서 더 큰 위협이 될 수 있다.

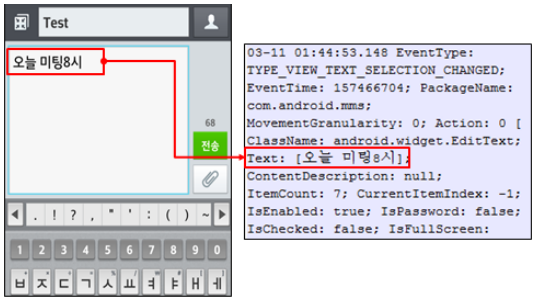


Fig. 8. UI events when user typing SMS

이 외에도 어플리케이션 잠금 비밀번호 유출도 가능하다. Fig.10.은 어플리케이션 실행 시 비밀번호 입력을 요구하는 일반적인 화면이다. 이미지 형태의 숫자 버튼 클릭 시 발생하는 이벤트 정보와 함께 입력된 비밀번호 숫자 정보는 클릭 이벤트의 세부정보(getText()와 getContentDescription()의 반환 값인 Text와 Description 정보)에서 확인할 수 있다.

4.2 어플리케이션 자동 재생에 의한 부당송금

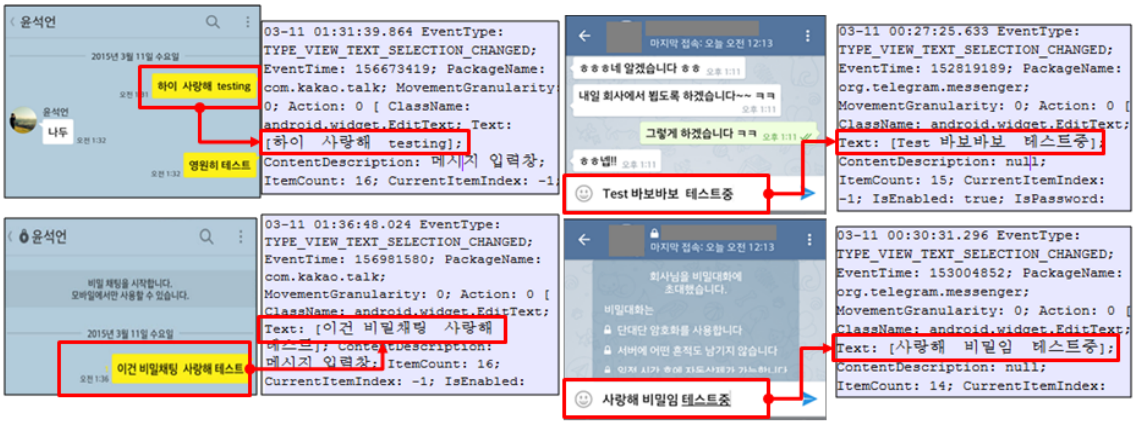


Fig. 9. UI events when user typing data on messenger

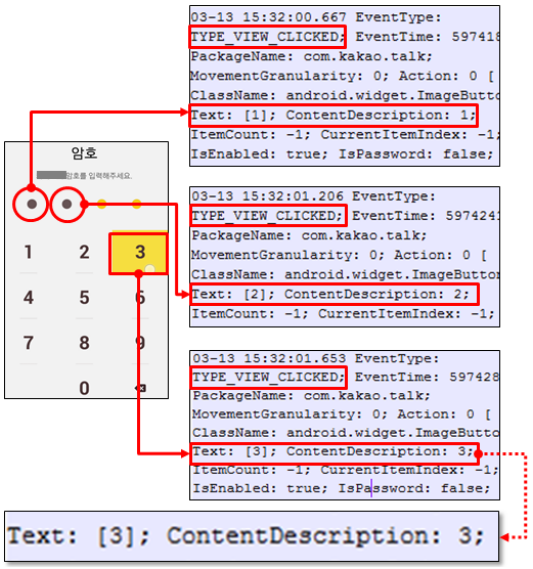


Fig. 10. UI events when user typing the lock-password(1,2,3,4)

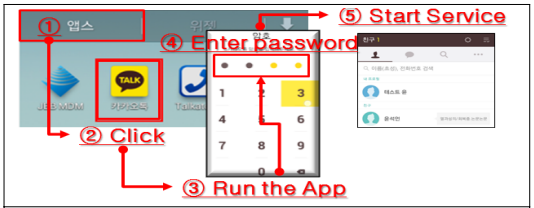
앞 절에서 설명한 노출된 비밀번호와 함께 UI 이벤트 정보들의 재구성을 통해 특정 어플리케이션 자동 실행 및 UI 이동, 비밀번호 입력 등을 자동화하여 이른바 재생공격(Replay Attack)이 가능하며, 이 과정에서 공격자가 임의의 정보를 입력시킬 수 있다. 송금과정을 예로 들면 수취인 계좌번호 혹은 금액정보를 공격자가 원하는 값으로 입력하여 부당송금을 유도할 수 있다.

Table 3.은 메신저 어플리케이션의 일반적인 실행과정을 슈도코드로 표현한 것이다. 이 과정에서 '④ Enter password'는 사전에 사용자로부터 비밀번호를 획득해야 하고, 획득한 비밀번호를 자동으로 입력되도록 구성할 수 있다. 이와 마찬가지로 송금 어플리케이션을 재생시켜 사용자 비밀번호 입력, 수취인 계좌로는 공격자의 계좌번호가 입력되도록 구성하여 사용자가 인지하지 못하는 시간대에 재생공격을 진행한다면 부당송금과 같은 위·변조 공격이 가능하다.

V. 국내 송금 서비스 어플리케이션의 UI 이벤트 로깅 및 비밀정보 평문 노출 실험 결과

본 절에서는 국내 스마트폰 송금 서비스를 제공하는 주요 어플리케이션에 대하여 UIAutomator를 활용한 UI 이벤트 로깅과 비밀정보 노출 여부에 대해 2015년 3월 10일에서 20일 사이에 조사 및 실

Table 3. Automation content on the step by step



	Description
①	Tab <- "*Apps Tab*"; if Tab is exist goto click();
②	AppName <- "*Application Name*"; if AppName is exist goto click();
③	*application doing progress itself*
④	Button1 <- Text is "1"; Button2 <- Text is "2"; ... if Button1 is exist goto Click(); if Button2 is exist goto Click(); ...
⑤	*application doing progress itself*

험한 결과를 Table 4.에 제시하였다.

실험에 사용된 스마트폰은 갤럭시S3(SHV-E210S)와 G2(LG-F320S)이고, 각각의 안드로이드 버전 4.2.1/4.3(갤럭시S3)와 5.0.1(G2)이다. 실험대상 분류는 보안키패드가 적용된 뱅킹 및 간편 송금 어플리케이션을 대상으로, 서비스 제공사의 대표성을 갖는 어플리케이션을 구글 플레이스토어를 통해 설치했다. 이후 어플리케이션에 적용된 보안키패드 제품을 조사하여 동일한 제품이 적용된 어플리케이션 별로 묶어서 무작위 선별하였다. 실험 방법으로는 각 어플리케이션 송금과정(로그인~송금완료)을 진행하면서 UIAutomator를 통해 UI 이벤트 모니터링을 실시한 결과를 확인하여 송금과정에서 입력된 각종 비밀번호가 평문으로 로깅되어 있는지 여부를 확인하였다. 그 결과 한 개 대상을 제외한 모든 실험대상에서 비밀번호가 평문으로 노출됨을 확인하였다.

VI. UIAutomator를 활용한 계좌번호 및 이체금액 위조

본 장에서는 Fig.11.의 시나리오를 바탕으로 사용자 스마트폰에 악성프로그램이 설치될 수 있고, 사용자가 인지하지 못하는 시점(예를 들어 잠이 든 새벽시간)이라는 가정 아래, UIAutomator를 활용하여 재생공격을 진행, 입금계좌번호 및 이체금액을 위조

Table 4. Experimental results about the disclosure of sensitive information(O : logging success, Disclosure / X : logging failure, keep the information)

Category	Application List	Keypad Solution	Experimental Result	
			UI Event Logging	Sensitive Information Disclosure
Smart-phone Banking	Bank A	Solution A	O	O
	Bank B		O	O
	Bank C		O	O
	Bank D		O	O
	Bank E	Solution B	O	O
	Bank F		O	O
	Bank G		O	O
	Bank H		O	O
	Bank I	Solution C	O	O
	O's Securities Module	Solution D	O	O
'Easy' Money Transfer	Application A	Built-in secure key-pad	O	O
	Application B		O	O
	Application C	Solution A	O	X

하여 공격자에게 부당송금이 가능하다는 것을 설명한다. 여기서 위조란 서버는 마치 사용자에 의해 정상적으로 송금이 진행되는 것처럼 인식하지만, 사실은 악성프로그램에 의해 자동으로 공격자의 계좌번호를 입력시켜 해당 계좌로 송금되는 것을 의미한다.

실험에서 위조 공격이 유효하다는 것은 Table 4.의 다른 송금 어플리케이션에서도 동일하게 적용될 수 있다는 것을 의미한다. 단, 보안카드번호 입력이 필요한 경우는 보안카드의 특성상 공격자가 사전에 사용자의 보안카드번호 수집이 선행되어야 한다는 전제사항이 있다. 하지만 최근 빈번하게 발생하는 피싱, 파밍 등의 금융사기 사건 사례에서 나타났듯이 금융사고 피해자의 상당수는 보안카드를 공격자에게 탈취 당한 경우이므로, 본 장

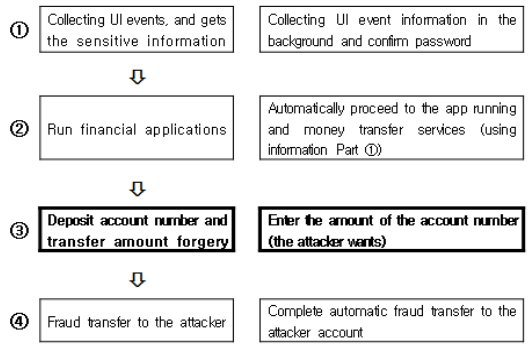


Fig. 11. Scenarios on the fraud money transfer

에서 언급하는 재생공격 시나리오도 그와 동일하게 공격자가 사용자의 보안카드번호를 알고 있다고 가정한다.

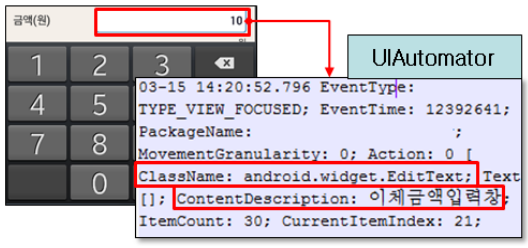


Fig. 12. Check the attribute information of the amount and account number entry box

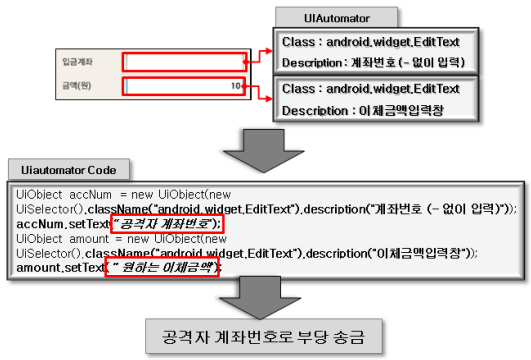


Fig. 13. Automation on deposit account and transfer amount input

입금계좌번호 및 이체금액은 일반적으로 기밀성을 요구하는 정보는 아니므로 일반키패드를 통해 입력이 진행된다. 이때 Fig.12.과 같이 UIAutomator를 통해 확인해 보면 이체금액 및 계좌번호가 입력되는 영역은 EditText 클래스로 구성되어 있으며, 해당 영역의 Description 정보는 '이체금액입력창'으로 Fig.13.과 같이 입금계좌 및 이체금액 입력상자에 공격자가 원하는 계좌번호와 금액을 설정(Setting) 해주면 해당 계좌로 부당송금 설정이 완료된다.

VII. 대응 방안

본 장에서는 5, 6장에서 보인 보안위협에 대해 안드로이드 운영체제와 단말기 제조사 측면, 서비스 제공사 측면으로 나누어 대응 방안을 제시하고자 한다.

7.1 안드로이드 운영체제 및 단말기 제조사 측면

본 논문에서 제기한 보안위협은 안드로이드 SDK와 단말기에 기본 제공되는 UIAutomator 프레임워크로부터 시작되었다.

테스트 해본 바, '/system/framework' 경로에 존재하는 uiautomator.jar와 uiautomator.odex 파일 등 관련된 파일이 삭제될 경우 본 논문에서 언급한 UI 이벤트 수집이 어렵고, 기존 설치된 어플리케이션은 아무런 영향 없이 정상 동작함을 확인하였다. 따라서 안드로이드 측에서는 모든 사용자가 아닌 개발자 및 테스터에 한하여 제한적인 프레임워크 제공 정책을 갖고, 단말기 제조사 측에서는 스마트폰에 탑재된 안드로이드에 해당 프레임워크 존재 여부에 대한 체크 과정이 함께 진행된다면 본 논문에서 제기한 문제의 해소가 가능하다.

7.2 서비스 제공사 측면

금융 서비스 제공사 측에서는 OTP(One Time Password) 형식의 일회용 인증을 강화함으로서 본 논문에서 제기한 일정부분의 문제점 해소가 가능해진다. 즉, 비밀번호가 유출되더라도 본 논문에서 보인 공격기법만으로는 송금서비스 진행이 완료될 수 없다. 또한, 본 논문에서 사용된 기법의 탐지 기능을 어플리케이션에 적용하여 대응할 수 있다.

7.2.1 2채널 및 거래연동 OTP 기술

OTP는 인증 시 마다 해당 세션에서만 사용할 수 있는 일회성 비밀번호를 생성하는 보안 시스템으로써, 한번 사용한 세션이 끝나면 즉시 폐기되기 때문에 재사용이 불가능하여 해킹이 발생하기 어려운 솔루션이다[20]. 따라서 사용자 비밀번호가 유출되더라도 이후의 송금서비스는 안전하게 진행될 수 있다.

현재 금융 서비스에서는 2채널 OTP를 시행 중에 있으나, 사용자가 OTP 단말을 발급받기 위해서 일정금액 비용이 발생한다는 점으로 인해 아직까지는 보편화되지 못하고 있는 실정이다. 또한 항상 휴대해야 한다는 불편함이 있는데, 이를 개선한 카드형 OTP 단말은 비용 발생이 더욱 가중된다. 하지만 현 인증시스템 중에서는 보안성이 가장 높은 수단으로 알려져 있으므로 비용 발생에 대한 문제가 제도적 보완을 거쳐 해결될 필요가 있다. 하지만 높은 보안수준을 제공하는 OTP라도 생성된 비밀번호의 생명주기(약 60초 가량)를 악용하여 MITM(Man-In-the Middle) 공격이 진행된다면 보안위협이 발생 될 수 있다는 문제점은 이미 많이 거론되어왔다. 이를 해결하기 위해 거래연동 OTP 기술[21]이 연구된 바가 있으며, 실제 송금 서비스에 적용된다면 보안문제를 상당부분 해소할 수 있다. 다만, 거래연동 OTP는 단말에 계좌번호를 입력해야 하므로 숫자패드가 부착되어야 하므로 크기가 커진다는 단점이 거론되고 있다.

7.2.2 CAPTCHA 형식의 OTP

또 다른 OTP 형태의 인증방식으로 CAPTCHA를 들 수 있다. CAPTCHA는 인증 시 마다 해당 세션에만 사용할 수 있는 특정 이미지를 사용자에게 시각적으로 보임으로서, 사용자가 이를 인식하여 이미지의 문자열 등을 입력하여 인증하는 방법이다.

기존에 발표된 이상호 등의 연구[22]에 의하면 인터넷 뱅킹 서비스를 보호하기 위한 CAPTCHA 기술로는 Arcot사의 VPS와 MS사의 MS워터마크가 있으며. 이들은 자동화 프로그램이 CAPTCHA 문자열을 인식하는 것을 어렵게 하고, 사용자가 인식 가능한 무작위성 CAPTCHA를 제공한다. 하지만 OTP 글자 색상이 배경과 서로 다르다는 점을 이용하여 특정 색상 또는 군집화 알고리즘을 통해, 혹은 특정 패턴을 통해 추출이 가능하다고 설명하고 있다.

이를 해결하기 위해 복잡도를 높여 OTP 추출을

더욱 어렵게 함으로서 개선이 가능하나 사용자도 인식 못하는 경우가 발생하므로 원본 이미지에 대한 정보가 사용자와 공유된 상태 그리고 사용자 선택에 의한 입력이 바탕이 된다면 복잡도가 높은 CAPTCHA라 하더라도 사용자는 인식이 가능하다고 밝히고 있다.

7.2.3 프로세스 감시를 통한 탐지기능 적용

일반적으로 운영체제에서는 파일명과 동일한 이름으로 프로세스가 동작한다는 점을 이용하여 현재 실행중인 프로세스 목록을 확인, 관련 프로세스가 실행 중일 경우 탐지하는 방법을 생각해 볼 수 있다. 본 논문에서는 순정 안드로이드 스마트폰에 기본으로 내장된 실행파일(uiautomator)과 라이브러리(uiautomator.jar)를 악용한 공격기법에 대해서 다루었고, 해당 파일들은 루트계정만이 쓰기가 허용되어 있으므로 일반권한으로는 파일명을 변조할 수 없다. 따라서 프로세스 감시를 통한 탐지기능을 어플리케이션에 적용한다면 본 논문에서 제기한 문제의 대응이 가능해진다.

VIII. 결 론

본 논문에서는 안드로이드 기반의 현 스마트폰 송금서비스가 UI 이벤트를 통해 입력정보 유출이 가능하고, 더 나아가 재생공격으로 이어질 수 있다는 점에 대해 논하였다. 구글 플레이스토어에서 배포 중인 실제 송금 서비스 어플리케이션의 대부분이 취약하다는 점을 실험을 통해 증명하였고 이를 응용한 공격 시나리오를 토대로 재생공격으로 이어질 수 있다는 설명과 그에 대한 대응방안을 제시함으로써, 현재보다 더 나은 환경의 송금 서비스가 구축될 수 있도록 기여했다는 데에 의의가 있다.

본 논문에서 증명한 내용은 현재 가장 활발하게 적용된 보안키패드만으로는 입력정보가 보호될 수 없다는 점에 대해 환기시키고, 많은 관련자가 간과하고 있는 화면 변화에 의해 발생되는 UI 이벤트를 통해서도 입력정보가 유출될 수 있다는 점을 주지시켰다.

기존에 알려진 공격기법은 루팅 등이 전제되어야 하는 제약이 있지만, 이러한 제약 없이도 입력정보 유출이 가능하다는 것을 본 논문을 통해 보였다. 다만, 본 논문에서 설명한 재생공격은 수집된 UI 이벤트 로그에 Text 혹은 Description 정보와 같은 식별 정보가 포함되어 있는 경우에 한정되어 있다. 이 같은 점을 극복하기 위해 화면에 구성된 각 UI 컴포넌트들의 좌표정보를 결합한 방법을 생각해 볼 수 있는데 관련 기술 및 대응 방안은 별도 연구가 필요할 것으로 보인다.

References

[1] Hwa-jeong Seo and Ho-won Kim, "cure Keypad with Encrypted Input Message," journal of the Korea Institute of Information and Communication Engineering, 18(12), pp. 2899-2910, Dec. 2014.

[2] Hyunjin Kim, "Virtual financial keypad with resistance to Shoulder-Surffing," Review of KIISC, 23(6), pp. 21-29, Dec. 2013.

[3] Sarang Na, "A Rolling Image based Virtual Keyboard Resilient to Spyware on Smartphones," Journal of The Korea Institute of information Security & Cryptology, 23(6), pp. 1219-1223, Dec.

[4] Yunho Lee, "An Analysis on the Vulnerability of Secure Keypads for Mobile Devices," Journal of Internet Computing and Services, 14(3), pp. 15-21, Jun. 2013.

[5] Getevent, https://source.android.com/devices/input/getevent.html

[6] Android Builders Summit 2013 - Integrating Sensors into Android Hardware, https://video.linux.com/videos/android-builders-summit-2013-integrating-sensors-into-android-hardware

[7] Multi-touch (MT) Protocol, https://www.kernel.org/doc/Documentation/input/multi-touch-protocol.txt

[8] Android Hooking Attack, http://hitcon.org/2013/download/[I2]%20Secret%20-%20AndroidHooking.pdf

[9] PTRACE, http://man7.org/linux/man-pages/man2/ptrace.2.html

[10] Android Native Screen capture application using the Framebuffer, http://www.pocketmagic.net/android-native-screen-capture-application-using-the-framebuf

fer/
〔11〕 pixelflinger.h, https://android.googlesource.com/platform/system/core.git/+/donut-release/include/pixelflinger/pixelflinger.h
〔12〕 View, http://developer.android.com/reference/android/view/View.html
〔13〕 WindowManager.LayoutParams, http://developer.android.com/reference/android/view/WindowManager.LayoutParams.html
〔14〕 Fernandes, Earlence, et al. "TIVOs: Trusted Visual I/O Paths for Android," CSE-TR-586-14, University of Michigan, May. 2014.
〔15〕 AccessibilityEvent, http://developer.android.com/reference/android/view/accessibility/AccessibilityEvent.html
〔16〕 Anbunathan. R and Anirban Basu. "An event based test automation framework for Android mobiles," Contemporary Computing and Informatics(IC3I) and 2014 International Conference on. IEEE, pp. 76-79, Nov. 2014.
〔17〕 UI Testing, tp://developer.android.com/tools/testing/testing_ui.html
〔18〕 Chen, Qi Alfred, Zhiyun Qian, and Z. Morley Mao, Peeking into your app without actually seeing it: Ui state inference and novel android attacks, 23rd USENIX Security Symposium, Aug. 2014.
〔19〕 Security Analysis on Secure Chat Messenger, http://www.hacknsecurity.com/blog/secure_chat.pdf
〔20〕 Byung-Tak Kang, "A study on the vulnerability of OTP implementation by using MITM attack and reverse engineering," Journal of The Korea Institute of information Security & Cryptology, 21(6), pp. 83-99, Dec. 2011.
〔21〕 Financial Security Agency, "Understanding authentication technology of transaction signing," Issue-report 2010-1, Financial Security Agency, Jan. 2010.
〔22〕 Sang-ho Lee, Sung-ho Kim, Jeon-Il Kang, Je-Sung Byun, Dea-Hun Nyang and Kyung-Hee Lee, "A Method of Enhancing Security of Internet Banking Service using Contents-Based CAPTCHA," Journal of The Korea Institute of information Security & Cryptology, 23(4), pp. 571-583, Aug. 2013.

〈저자소개〉

윤 석 언 (Seok-Eon Yoon) 정회원
2008년 2월: 수원대학교 컴퓨터학과 졸업
2009년 2월~2015년 4월: 사단법인 금융보안연구원 주임연구원
2010년 9월~2013년 2월: 고려대학교 정보보호대학원 정보보호학과 석사 수료
2015년 4월~현재: 사단법인 금융보안원 재직
〈관심분야〉 정보보호, 역공학, 모바일보안

김 민 성 (Min-sung Kim) 정회원
2009년 8월: 선문대학교 전자공학과 졸업
2012년 2월~2015년 4월: 사단법인 금융보안연구원 연구원
2015년 4월~현재: 사단법인 금융보안원 재직
〈관심분야〉 정보보호, 전자공학

이 상 진 (Sang-jin Lee) 종신회원
1989년 2월~1999년 2월: 한국전자통신연구원 선임 연구원
1999년 2월~2001년 8월: 고려대학교 자연과학대학 조교수
2001년 9월~현재: 고려대학교 정보보호대학원 교수
〈관심분야〉 대칭키 암호, 정보은닉이론, 디지털 포렌식