Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.5, Oct. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.5.975

안드로이드 플랫폼 환경에서의 스미싱 차단에 관한 연구*

이 시 영,† 강 희 수, 문 종 섭‡
고려대학교 정보보호대학원

A Study on Smishing Block of Android Platform Environment*

Si-young Lee,† Hee-soo Kang, Jong-sub Moon‡
Center for Information Security Technologies, Korea University

요 약

스마트폰을 사용한 전자금융거래가 증가하면서, 스마트폰을 대상으로 하는 새로운 위협들이 증가하고 있다. 그중에서도 최근, 가장 위협이 되고 있는 스미싱은 갈수록 수법이 교묘해지고, 다양해지고 있다. 금융기관들은 증가하는 스미싱 위협에 대응하기 위한 방안으로 "알 수 없는 출처"의 설정을 통해 어플리케이션 설치를 차단하는 방법과 스미싱 탐지 어플리케이션의 설치를 권고하고 있다. 하지만 현재까지의 대응방안이 갖고 있는 한계적 문제로 인해 효과적으로 스미싱에 대응하지 못하고 있다. 본 논문에서는 안드로이드 RIL(Radio Interface Layer)에서 사용자 기기에 수신된 문자메시지를 수집하고, DB(Database) 내에 문자메시지가 저장되었는지 검사하여, 악성코드의 설치여부를 판단한다. 그리고 커널레벨에서의 시스템 콜 후킹을 통해 악성코드가 전송하는 문자메시지를 차단하는 기법을 제안한다. 또한 실제 구현을 통하여 기기에 적용이 가능함을 증명하였다.

ABSTRACT

As financial transactions with a smartphone has become increasing, a myriad of security threats have emerged against smartphones. Among the many types of security threats, Smishing has evolved to be more sophisticated and diverse in design. Therefore, financial institutions have recommended that users doesn't install applications with setting of "Unknown sources" in the system settings menu and install application which detects Smishing. Unfortunately, these kind of methods come with their own limitations and they have not been very effective in handling Smishing. In this paper, we propose a systematic method to detect Smishing, in which the RIL(Radio Interface Layer) collects a text message received and then, checks if message databases stores text message in order to determine whether Smishing malware has been installed on the system. If found, a system call (also known as a hook) is used to block the outgoing text message generated by the malware. This scheme was found to be effective in preventing Smishing as found in our implementation.

Keywords: Smishing, Malware, RIL, LKM

I. 서 론

오늘날 스마트기기의 발달과 전자금융거래의 편리성 때문에 스마트폰을 이용한 금융거래는 매년 빠른 속도로 증가하고 있다. 스마트폰을 이용한 금융거래가 증가하면서[13], 이에 따라 스마트폰을 노리는 악성코드의 수도 많이 증가하였으며, 특히 안드로이드를 기반으로 하는 공격이 눈에 띄게 증가하였다. 실제 카스퍼스키랩에서 발표한 모바일 위협 통계[1]에 의하

접수일(2014년 7월 30일), 수정일(20014년 9월 15일, 게재확정일(2014년 9월 22일)
* 본 연구는 미래창조과학부 및 한국인터넷진흥원의 "2014년도 고용계약형 정보보호 석사과정 사업"의 연구 결과로 진행되었음(과제번호 H2101-14-1001)
† 주저자, sherlocklee@korea.ac.kr
‡ 교신저자, jsmoon@korea.ac.kr(Corresponding author)

면 2013년 말까지 모바일 악성코드 중 98%가 안드로이드를 공격대상으로 삼고 있다. 이는 안드로이드가 역공학을 이용한 악성코드 제작이 용이하며[16], 개방형 마켓 구조를 통한 악성코드의 배포가 쉽기 때문이다. 2012년 11월 국내에서도 스미싱(SMS+Pishing)을 이용한 안드로이드 악성코드 'chest'가 출현하여 첫 금전피해를 발생시켰다[2]. 안드로이드를 공격대상으로 하는 스미싱은 시간이 지나면서 더욱더 교묘해지고 다양해져 최근에는 "모바일 무료쿠폰, 청첩장, 고지서" 등으로 위장하여 사회 공학적 기법까지 활용되고 있다.

본 논문의 목적은 사용자가 인지하지 못한 상태에서, 악성코드에 의한 무단 소액 결제가 발생하는 것을 탐지/방지 하는 방법을 제시하는 것이 목적이다. 이를 위하여, 안드로이드 Radio Interface Layer(RIL)에서 수집한 문자메시지가 DB 내에 저장되는지에 따라 문자메시지의 탈취를 판단하고, LKM(Loadable Kernel Module)으로 구현된 시스템 콜 후킹모듈을 이용하여 소액 결제 시 결제 회사로부터 주어지는 인증번호의 무단 외부 전송을 차단하는 기법을 제안한다.

본 논문의 구성은 2장에서 스미싱에 의한 무단 소액결제 발생 원리에 대해 설명하고, 3장에서는 관련연구 및 스미싱의 정의와 현재 존재하는 스미싱 대응방안에 대해 서술하였다. 4장에서는 제안방법의 특징에 대해 소개하며, 5장에서는 구현결과에 대해 확인하고, 6장에서 결론을 통한 본 논문의 정리와 향후연구에 대하여 설명한다.

II. 문제정의

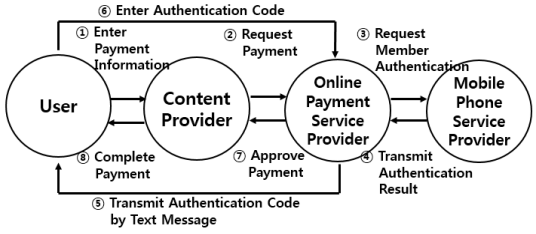
2.1 정상적인 소액결제 과정 및 스미싱을 통한 소액결제 피해

일반적으로 사용자가 소액결제를 하는 경우, Fig. 1.(A) 와 같은 과정이 진행된다.

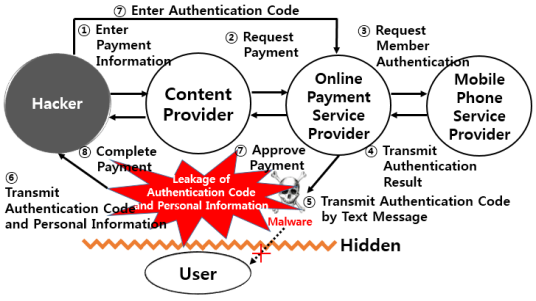
① 사용자가 상품을 판매하는 사이트(가맹점)에 접속하여 결제 정보를 입력
② 가맹점은 결제업체에 사용자가 선택한 상품에 대해 결제를 요청
③ 결제업체는 이동통신사에 가입자의 정보가 확실한지 인증 요청
④ 이동통신사는 결제업체에게 인증결과를 전송
⑤ 결제업체는 사용자의 휴대폰을 이용한 2차 인증을 진행하기 위하여 문자메시지 내에 인증번호를 전송
⑥ 사용자는 수신한 문자메시지에 포함된 인증번호를 결제업체에 전송
⑦ 결제업체는 사용자에 대한 2차 인증이 완료됐을 때, 결제를 승인하고 결제승인 메시지를 가맹점으로 전송
⑧ 가맹점은 사용자에게 결제승인 결과를 전송

이와 다르게, 공격자는 Fig.1.(B) 에서처럼 악성 어플리케이션을 사용하여 사용자 모르게 소액결제를 시도한다. 즉, 다음과 같은 과정으로 이루어진다.

① 해커(또는 해커의 조정을 받은 서버 컴퓨터)가 사용자 모르게 상품을 판매하는 사이트(가맹점)에 접속하여 사용자의 정보를 입력
② 가맹점은 결제업체에 해커가 선택한 상품에 대해 결제를 요청
③ 결제업체는 이동통신사에 가입자의 정보가 확실한지 인증 요청
④ 이동통신사는 결제업체에게 인증결과를 전송
⑤ 결제업체는 가맹점의 결제 요청이 맞는지 확인하기 위하여, 사용자 스마트폰의 문자 메시지 내에 인증번호를 삽입하여 사용자에게 직접 전송
⑥ 악성코드는 인증번호가 포함된 문자메시지를 탈취



(A) Process of normal mobile payment



(B) Process of acquiring money using Smishing by attacker

Fig. 1. Process of a normal mobile payment and an abnormal mobile payment by malware code

하여 해커의 서버로 전송, 사용자 화면에는 메시지가 표시되지 않음

⑦ 해커(또는 해커의 조정을 받은 서버 컴퓨터)가 수신한 문자메시지에 포함된 인증번호를 결제업체에 전달

⑧ 결제업체는 가맹점에 결제 승인

⑨ 가맹점은 해커에게 결제승인 결과를 전송, 이 결과는 사용자의 스마트폰에서 감추어짐

이 과정에서 수신된 문자메시지는 사용자 화면에 전혀 표시되지 않는다. 따라서 수신자가 인식하지 못하는 상태에서 금전적인 손실이 발생한다.

III. 관련연구

스미싱 위험에 대응하기 위한 연구로는 박상호[18] 등의 연구에서 문자메시지 수신 시, 첨부된 인증 값을 이용하여 인증 앱과 신뢰기관이 인증과정을 진행하고, 문자메시지가 안전한 발신지로부터 발신되었다는 것을 증명하는 방식을 사용하였다. Tiago Almeida[20] 등은 스미싱을 유도하는 특정 문자열을 검출하는데 여러가지 기계학습 알고리즘을 사용하고, 가장 적합한 알고리즘을 찾아내어 스미싱 메시지를 차단하는 방식을 소개하였다. Anna Kang[21] 등의 연구에서는 어플리케이션 레벨에서 수신된 문자메시지 내에 URL 클릭 시, 인터넷 접속에 사용되는 IP주소가 인가된 것인지(DB에 존재하는지) 확인하는 방식을 설명하였다. 그리고 최영석[12] 등의 연구에서는 커널레벨 시스템콜 후킹과 화이트리스트 방식을 사용하여 인가되지 않은 IP주소로의 접속을 차단하는 방식을 사용하였다.

3.1 스미싱의 정의

스미싱(Smishing)이란 문자메시지(SMS)와 피싱(Phishing)을 합성한 신조어로 스마트폰에서 작동원리는 다음과 같다. Fig.2.와 같이 악의적인 공격자는 피해자에게 URL 클릭을 유도하는 문자메시지를 발송한다. 사용자는 URL을 클릭하게 되고, 악성코드가 포함된 어플리케이션이 다운로드 된다. 그리고 사용자는 특별한 의심 없이 설치를 진행하고, 설치 후에는 개인정보가 무단으로 유출되며, 사용자 모르게 소액결제가 진행되어 금전적 피해를 발생시킨다[3].

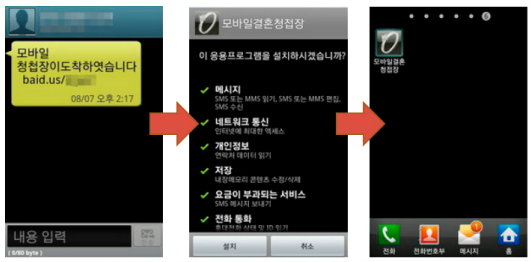


Fig. 2. Process of installing Malware, which inducing Smishing

3.2 스미싱 대응방안

3.2.1 스미싱 탐지 어플리케이션

꾸준하게 발생하고 있는 스미싱을 방지하기 위하여 통신사와 보안회사에서는 여러 가지 해결책을 내놓고 있다. Table 1. 를 보면 첫째, URL 및 문자열을 이용하여 탐지하는 방법은 사용자 기기에 수신된 문자메시지의 내용을 블랙리스트에 등록된 URL 및 문자열과 비교하여 탐지하는 방법이다. 하지만, URL Shor-tener를 이용하여 주소를 압축하거나[4][5], SNS를 통한 스미싱 기법[6]이 등장하면서 오탐율이 증가하고 있다. 둘째, 어플리케이션의 권한을 기반으로 안정성 검사를 하는 경우, 설치된 어플리케이션이 스미싱을 발생시키는데 필요한 특정 권한들을 가지고 있을 시, 사용자에게 경고 및 삭제를 요청하는 방식이다.

Table 1. Classification of solution by Smishing detection method

Dection Method	Application Name
The detection method of a string and URL	S-GUARD
	Safe SMS
	Smishing Defender
	SClean
	What is this SMS
The detection method of application permission	Phishing Guard
	Olleh Block Smishing
	Smishing Hacking Detection
The method of integrity check of application	Block Smishing
	What is this Application

하지만 검사 후 어플리케이션을 삭제하기 이전에, 악성 어플리케이션의 공격이 수행 된다면 피해를 입을 수 있는 위험이 존재한다[17][18]. 또한 권한 정보만으로는 어플리케이션이 어떤 행위를 하는지 정확히 판단할 수 없으므로, 정상 어플리케이션과 비정상 어플리케이션을 명확하게 구분할 수가 없다. 마지막으로 일반적인 어플리케이션의 무결성 검사 방법의 경우, 실행중인 어플리케이션의 APK 파일내용에 대한 해시값을 계산하여 서버로 안전하게 전송한 후, 서버에서 보관중인 원본 APK파일의 해시값과 비교하는 방식이다[7]. 하지만 APK 파일을 변조한 후, 실행중인 APK 파일의 경로를 얻어오는 자바 코드를 별도의 디렉토리에 보관된 원본 APK 파일의 경로로 반환되도록 수정하면, 무결성 검사가 우회되므로 취약점이 존재한다.

3.2.2 출처가 불분명한 어플리케이션 설치 차단

Fig.3. 에서처럼 공식마켓 이외에 다른 출처에서 어플리케이션을 다운로드하여 설치하지 못하도록 '환경설정 〉 보안 〉 알 수 없는 출처'를 통해 기본적인 설정이 가능하다. 그러나 특정 통신사 마켓이나 금융앱스토어에서 어플리케이션을 다운로드 받는 경우, 설정을 "허용"으로 변경해야만 설치가 가능하기 때문에, 이 상태에서 사용자의 기기는 악성 어플리케이션이 설치 될 가능성이 존재하게 된다[19].

Fig. 3. "Unknown Source" option

3.2.3 소액결제 금액 차단 및 제한

악성코드가 포함된 어플리케이션이 설치된 경우 사용자가 인지하지 못하는 사이에 소액결제가 이루어지므로, 피해를 입지 않기 위해 통신사에서 소액결제를 차단하도록 설정하거나, 금액의 이용한도를 제한할 수 있다. 그러나 소액결제를 차단할 경우에는 작은 금액이라도 결제 시에 카드나 금융거래를 이용해야 하므로, 추가적으로 발생하는 과정으로 인해 불편초래 할 수 있다. 또한 금액의 한도를 제한하는 방법의 경우에도, 스미싱을 당하면 제한범위 내의 금액도 피해를 입을 수 있기 때문에 완벽하게 스미싱을 차단하는 방법이라 할 수 없다. 그리고 많은 사용자들이 소액결제 차단 및 제한 서비스의 존재를 모르고 있어, 제대로 기능을 사용하지 못하고 있는 상태이다.

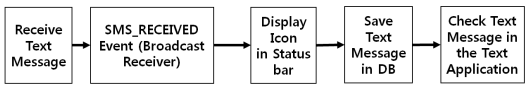
IV. 제안기법

4.1 기법 개요

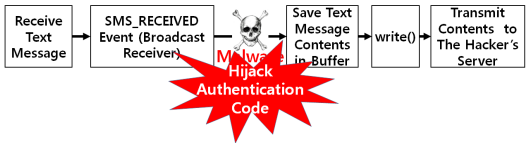
일반적으로 사용자 기기에 문자메시지가 수신되었을 때 악성 어플리케이션의 설치여부에 따라 정상동작과 악성 동작으로 나누어져 진행된다.

Fig.4.(A) 는 정상동작 과정으로 기기에서 문자메시지가 수신되면 SMS_RECEIVED 이벤트가 발생하고, 스마트폰의 상태를 나타내는 부분("상태바")에 편지모양의 아이콘이 표시되며, 문자메시지가 DB에 저장이 되어 메시지 어플리케이션에서 이를 확인 할 수가 있다.

악성어플리케이션이 설치되었다면, Fig.4.(B) 처럼 악성 동작이 진행된다. 정상동작과는 다르게 SMS_RECEIVED 이벤트가 발생한 후에 악성 어플리케이션이 제일 먼저 이벤트를 수신하고, 다른 어플리케이션으로는 이벤트가 전송되지 않도록 차단하는 방식을 사용한다[14]. 이때 상태바에도 아이콘이 표시 되



(A) Normal operation after receiving a text message



(B) Abnormal operation after receiving a text message

Fig. 4. Normal operation and abnormal operation after receiving a text message

지 않으며, 악성 어플리케이션은 수신된 문자메시지를 탈취하고, 공격자의 서버로 전송하기 위하여 네트워크 연결을 시도하게 된다. 이 때 최종적으로 안드로이드 커널에서 시스템 콜 write()가 호출되며, 자신이 원하는 정보와 인증번호가 포함된 문자메시지를 전송한다.

4.2 제안방법

4.2.1 개략적인 방법

악성 어플리케이션의 동작을 차단하기 위하여 제안하는 기법은 Fig.5. 와 같이 3가지 흐름으로 진행된다. 3가지 모듈이 추가되는데, 첫 번째로 악성 어플리케이션이 문자메시지를 탈취하기 전에 문자메시지를 수집할 수 있는 모듈을 추가한다. 그리고 두 번째로 사용자 기기의 DB안에 수집한 문자메시지가 저장이 되었는지 유무를 확인하는 모듈도 추가한다. 세 번째, 문자메시지가 저장이 되지 않았을 경우, 비정상이라고 판단하고 악성어플리케이션이 문자메시지를 서버로 전송하는 것을 차단하기 위하여, 네트워크 통신에 사용되는 시스템 콜 write()를 후킹하는 모듈을 커널에 삽입한다. 이 모듈은 서버로 전송할 때 사용하는 버퍼를 검사하도록 구현되어 있으며, 전송되는 내용 중 미리 수집한 문자메시지의 내용이 포함되어 있다면 전송을 차단하는 역할을 한다.

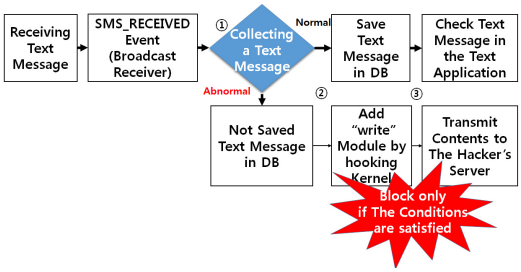


Fig.5. Flowchart of the suggested method

4.2.2 모듈 설명

4.2.2.1 Radio Interface Layer를 이용한 문자메시지 추출

· 문자메시지 포맷

스마트폰에서 문자메시지를 송신 할 때, 사용자가 입력한 문자가 그대로 전송되는 것이 아니라, 이동통신 표준화 기술협력기구인 3GPP에서 표준화한 데이터 포맷에 맞게 변경하여 전송한다[8]. 이때 데이터 포맷을 PDU라 부르며, 전송목적지와 사용하는 목적에 따라 다른 방식의 포맷을 사용하고 있다. 그 중 우리가 기본적으로 사용하는 형태는 두 가지가 존재한다. 첫 번째로 SMS_SUBMIT는 스마트폰에서 Short Message Service Center(SMSC)[15]로 전송할 때 사용하며 두 번째, SMS_DELIVER는 SMSC에서 스마트폰으로 전송할 때 사용한다. 여기서 SMSC란 송수신 중간 과정에서 문자메시지를 일시적으로 저장하였다가 수신번호로 전송을 하는 서비스 센터로써, 이는 수신자가 수신을 할 수 없는 상황이거나 전송오류 등 다양한 이유가 발생 할 있기 때문에 사용이 되고 있다. 본 논문에서는 Table 2. 와 같이 수신시에 사용되는 표준화된 데이터 포맷 SMS_DELIVER을 바탕으로 하며, 우리나라 통신사

Table 2. Structure of SMS_DELIVER Type in 3GPP

Name	Length	Purpose
SMSC address	variable	The address of Service center which sends a Text to a wireless device
MTI	2bit	Message type
MMS	1bit	Check bit if there are more messages to send.
LP	1bit	Check inhibiting automatic message generation that could cause infinite looping.
RP	1bit	Parameter indicating that Reply Path exists.
UDHI	1bit	The UserData field contains a Header.
SRI	1bit	Check if the Sender has requested a status report.
OA	2-12 Byte	Sender address
PID	1Byte	Protocol ID
DCS	1Byte	Data Coding Scheme
SCTS	7Byte	Time Stamp
UDL	1Byte	User Data Length
UD	variable	User Data

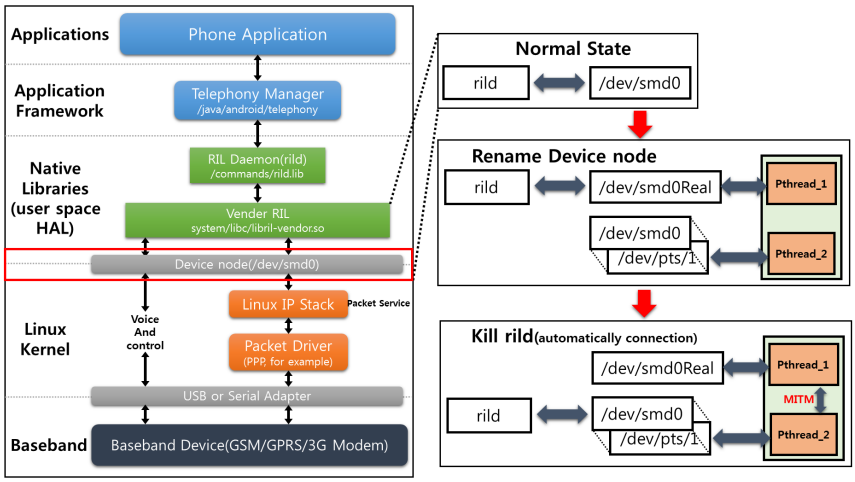


Fig. 6. Android's Telephony Architecture and modifying a Device node

에서는 약간 변형된 형식의 포맷을 사용하고 있다.

· RIL(Radio Interface Layer)

Fig.6. 의 왼쪽 그림은 안드로이드 전화통화 시스템 구조를 나타낸다. Application Layer의 어플리케이션에서 전화에 관련된 서비스를 사용하기 위해 A-pplication Framework의 Telephony에서 제공하는 API를 호출한다. 그리고 중간 계층인 RIL 거쳐 해당 명령어가 Baseband까지 전달된다. 여기서 RI-L[9]이란 Telephony 서비스들과 Hardware간의 추상화된 계층을 의미하며, 구성요소로는 RIL Daemon과 Vendor RIL로 구성된다. RIL Daemon은 안드로이드 Telephony 서비스들로부터의 전화에 관련된 모든 통신을 처리하여 "solicited" 명령어 이용하여 Vendor RIL로 넘겨주며, baseband까지 전달되는 과정 중에 중간 역할을 한다. 그리고 데이터를 수신할 때는 이 과정이 역으로 수행된다. Vendor RI-L은 Hardware와의 모든 통신을 처리하며, Baseb-and에서 들어오는 정보를 "unsolicited" 명령어로 RIL Daemon에게 넘겨준다. 그리고 application Framwork의 Telephony를 거쳐 최종적으로 사용자의 어플리케이션에서 확인할 수가 있다. "solicite-d" 명령어는 DIAL(전화걸기)과 HANGUP(전화끊기)등과 같은 내용으로 60개 이상 명령어들이 존재하며, RIL 라이브러리에 의해 시작되는 명령이다. 그리고 unsolicited 명령어는 CALL_STATE_CHAN-GED(전화 상태 변화)와 NEW_SMS(새 문자 메시지 알림)등과 같은 내용으로 10개 이상의 명령어가 존재하고 Baseband로부터 시작되는 명령을 의미한다.

· 문자메시지 수집

Application Layer에서 동작하는 악성 어플리케이션 보다 먼저 문자메시지를 수신하기 위하여 Colli-n Mulliner가 제안한 방법[10]을 이용한다. Fig.6. 의 오른쪽 그림처럼 Vendor RIL과 Baseband가 통신할 때 사용하는 Deivce node의 파일 이름을 변경한다. 그리고 rild 프로세스를 다시 시작시켜, MIT-M(Man In The Middle) 형태의 모니터링을 통해 수신된 PDU 형식의 문자메시지를 수집한다.

4.2.2.2 악성 어플리케이션 설치여부 판단

일반 Android의 레퍼런스 폰에서 SMS, MMS가 저장되는 데이터베이스 파일은 "/data/data/com.a-ndroid.providers.telephony/databases/mms-sms.db"이다. sqlite 쿼리문을 사용하여 이 파일에서 문자메시지 내용이 저장되어 있는 "SMS" 테이블의 "body"을 읽어온다. 그리고 3.2.1에서 생성한 더미 D-evice를 이용하여 수집한 문자메시지가 DB에 저장되었는지를 검사하고, 저장여부에 따라 문자메시지가 악성 어플리케이션에 의해 탈취되었는지를 판단할 수 있으며, 또한 악성 어플리케이션의 설치 여부까지 판단이 가능하다.

4.2.2.3 시스템 콜 후킹을 이용한 메시지 전송차단

이 모듈은 운영체제의 커널 내에서 동적으로 삽입/제거가 가능하며, 모듈 삽입 시 Fig.9. 처럼 시스템 콜을 후킹 하여 정상적인 시스템 콜 보다 선행동작하게 된다. 그리고 문자메시지의 전송을 차단하기 위해 후킹 하는 시스템 콜 목록은 Table 3. 과 같으며, L-KM으로 구현된 모듈은 소켓통신에서 사용되는 writ-e()함수를 후킹 한다. 모듈의 역할은 기기에서 서버로 전송하는 내용이 저장되어 있는 buf 내에 특정 문자열의 포함여부를 확인하는 것이다. 본 논문에서는 '인증번호'라는 문자열과 미리 수집해 두었던 문자메시지에서 추출한 인증번호가 존재하는지 검사한다. 그리고 검사에 사용되는 알고리즘으로는 기계학습에서 사용하는 많은 알고리즘들이 존재하므로, 실제 악성코드가 어떤 형태로 버퍼에 인증번호를 저장하는가에 따라 가장 효과적인 알고리즘을 사용한다. 결과적으로 Fig.7. 처럼 조건이 달성되었을 경우, 공격자의 서버로 전송되는 것을 차단함으로써 스미싱을 차단할 수 있다.

Table 3. List of system calls that hooking

Systemcall Name	Function Definition
write()	ssize_t write(int fd, const void *buf, size_t count);

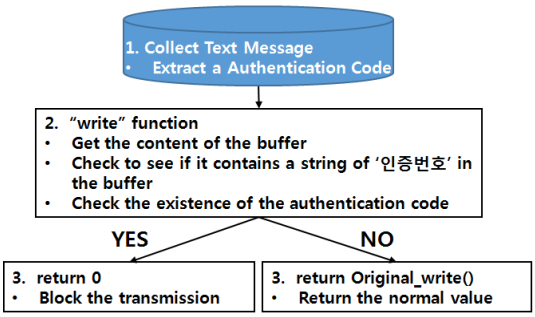


Fig. 7. Flowchart of blocking method to transfer a text message

V. 구현 및 결과

5.1 구현환경

본 논문에서 제안하는 기법을 구현한 환경은 Table 4.와 같다.

Table 4. Implemented environment

Item	Specification
CPU	Intel core i7-3770 @3.4GHz
RAM	4.00GB
OS	ubuntu 10.24 32bit
Test Phone	Nexus One(kernel 2.6.35.7), Android 2.3.6

5.2 구현결과

· RIL(Radio Interface Layer)을 이용한 문자메시지 수집모듈

Fig.8.은 Fig.6.의 오른쪽 그림과 같이 RIL에서 Vendor RIL과 Baseband가 통신할 때 사용하는 Device node의 파일이름을 /dev/smd0에서 /dev/smd0real로 변경한다. 그리고 kill 명령어를 이용하여 rild 프로세스를 죽이면, 다시 살아나면서 rild과 /dev/smd0는 다시 연결이 된다. 따라서 /dev/smd0real이 더미 Device node가 되는 실제 진행 과정을 보여준다. 본 논문에서 Device node의 파일이름은 HTC사를 기준으로 하였다. 그리고 Device node의 파일이름은 테스트를 진행하는 기기의 제조사에 따라 변경될 수 있다.

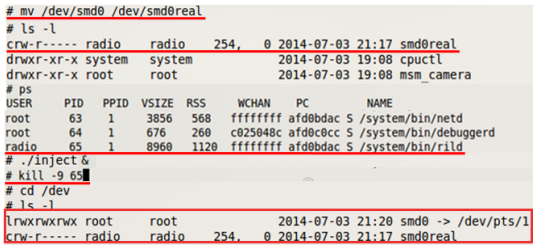


Fig. 8. Process of generating dummy Device node

· DB(Data Base)를 이용한 악성코드 탐지모듈

Fig.10.과 같이 더미 Device node인 smd0real를 통하여 PDU 포맷으로 이루어진 문자메시지를 모니터하고 수집 할 수 있다. PDU 포맷은 UCS-2로 인코딩 되어 있으며, 사용자 기기의 DB는 UTF-8로 인코딩 되어 있다. 따라서 문자메시지의 body부분을 UTF-8 인코딩하여, 사용자 기기의 DB안에 저장되

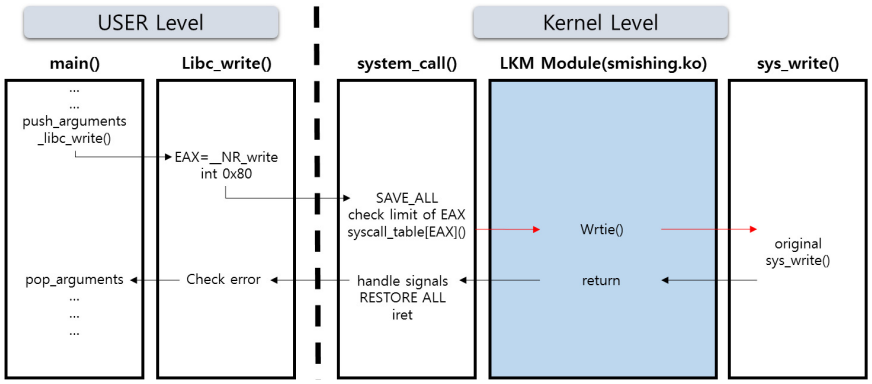


Fig. 9. Insertion of LKM module for hooking the write method

어 있는 문자메시지 내용과 비교한다. 그리고 수신된 문자가 저장이 되었는지를 확인한다.

```
# ./inject
read 191 bytes from smdreal:
+CMT: ,81
07912801929190610408A11060425790F900084170109134026 33EBCF8C778C77
8C778C99DBC88D638B2940020003400300035003900390038 0020C785B
2C8B2E4002E0020C815D655D7880020C785B825D574C8FCC138C694002E
MessageBody :
BCF8C778C778C99DBC88D638B2940020003400300035003900390038 0020C785B2
C8B2E4002E0020C815D655D7880020C785B825D574C8FCC138C694002E
******************** DB Check ********************
abc아
00 61 00 62 00 63 C5 44
Abcf얼
00 41 00 62 00 63 00 66 C5 BC
KB국민체크(9*0*)
김재기129
00 4B 00 42 AD 6D BB FC CC B4 D0 6C 00 28 00 39 00 2A 00 30 00 2A 00
29 00 0A AE 40 C7 AC AE 30 00 31 00 32 00 39
```

Fig. 10. Checking whether collected PDU is stored DB or not

· write() 시스템 콜 후킹 모듈

Fig.9.와 같이 네트워크 연결에서 데이터 전송에 관한 시스템 콜을 후킹하기 위한 모듈은 LKM으로 구현되어 동적으로 커널에 추가가 가능하다[11][12]. 구현된 모듈이 동작하는지 시험하기 위하여 테스트 폰에 스미싱 행위를 하는 악성 어플리케이션을 설치하고, 문자메시지 인증을 과정을 진행하였다. 그리고 탈취된 문자메시지가 전송되는 서버의 IP주소(163.152.126.90:8080)로 접속해보았다. Fig.11. 와 같이 후킹모듈을 삽입하기 전에는 문자메시지가 탈취되어 서버 데이터베이스에 저장되는 결과를 볼 수 있다. 하지만, 구현된 모듈을 삽입한 후에는 Fig.12. 에서처럼 전송되는 문자메시지의 내용이 Fig.7. 의 조건과 일치한다면 차단이 가능하다.

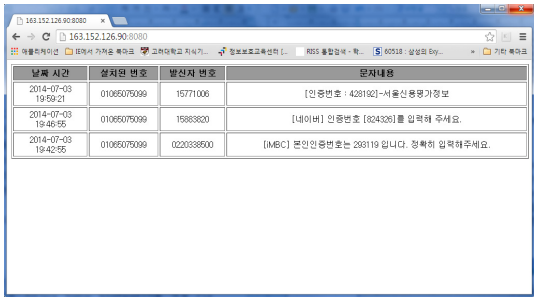


Fig. 11. The server which store the hijacked text message

```
<4>[  914.181915] ************************************************************
<4>[  914.182189] sockfd    = 34
<4>[  914.182312] UID NUM   = 10057
<4>[  914.182464] write_buf = POST /index.php HTTP/1.1
<4>[  914.182495] Content-Length: 233
<4>[  914.182495] Content-Type: application/x-www-form-urlencoded
<4>[  914.182495] Host: 163.152.126.90:8080
<4>[  914.182495] Connection: Keep-Alive
<4>[  914.182525] User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)
<4>[  914.182525]
<4>[  914.182525] telnum=01062475099&textvalue=%EB%B3%B8%EC%9D%B8%EC%9D%B8%EC%A6
9D%EB%B2%88%ED%98%B8%EB%8A%94+405998+%EC%9E%85%EB%8B%88%EB%8B%A4.+%EC%A0%95%ED
%95%B4%EC%A3%BC%EC%84%B8%EC%9A%94.&phonenum=01065075099
<4>[  914.184600] String Detection : %EC%D%B8%EC%A6%D%EB%B2%ED%B8
<4>[  914.184753] Find String!!!
<4>[  914.184875] number Detection : 405998
<4>[  914.185150] Find Number!!!Blocking transmission!!!!!
```

Fig. 12. Internal operation of hooked module

· 문자메시지 수집모듈 추가 시 문자처리시스템 성능 비교

본 논문에서 제안하는 문자메시지 수집모듈은 Fig.5.의 ①처럼 추가가 가능하다. 수집모듈의 추가로 인한 문자처리속도를 평가하기 위하여, "DB에 문자메시지가 저장된 시간 - SMS_RECEIVED 이벤트가 수신된 시간"을 모듈 추가 전/후로 500개씩 각각 측정하고, 100개씩 무작위로 추출해 내어 Fig.13. 와 같이 그래프로 나타내었다. 모듈을 추가하기 전 문자메

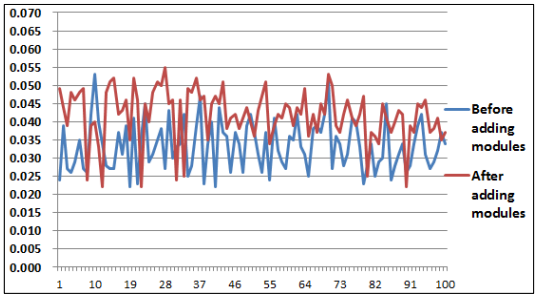


Fig. 13. Velocity measurement of text message processing system

시지의 평균 저장속도는 0.033초였으며, 모듈을 추가한 후 평균 저장속도는 0.042초였다. 실제 0.009초 차이 밖에 나지 않으므로, 수집모듈은 문자처리시스템에 많은 영향을 미치지 않는다.

VI. 결 론

최근 스미싱을 이용한 범죄가 증가하면서 안전한 전자금융 환경을 위협하고 있다. 금융감독원이나 금융결제원, 금융보안연구원에서도 스미싱에 대응하기 위한 방안들을 제시하고 있지만, 제시된 대응방안들이 갖고 있는 한계점으로 인해 스미싱에 효과적으로 대응하지 못하고 있다.

본 논문에서는 스미싱에 대응하기 위한 방법으로 R-adio Interface Layer을 이용하여 수신된 문자메시지를 수집하고, DB 내에 저장유무에 따라 악성코드의 감염여부를 판단하였으며, 커널 레벨에서 시스템 콜을 후킹 하여 전송되는 문자메시지의 탈취를 차단하였다. 기존에 제안된 기법들과는 다르게 사용자가 이용하는 어플리케이션 Layer에서의 대응책이 아닌, 더 낮은 Layer에서 실시간으로 탐지와 차단이 가능하도록 하였다. 하지만 본 논문에서 제안하는 모듈이 root 권한을 필요로 하는 단점이 존재하기 때문에, 이를 해결하기 위하여 플랫폼 제조사에서 더미 디바이스와 커널 모듈을 추가하면 해결이 가능하다. 향후 연구로는 악성코드의 인증번호 탈취를 차단하기 위하여, 스미싱을 유도하는 특정 문자열 및 인증번호를 검출하는데, 다양한 기계학습 알고리즘을 적용하여 보고, 차단확률을 높일 수 있는 방법에 관해 연구 할 계획이다.

References

[1] Kaspersky Lab, "Kaspersky Security Bulletin 2013, Overall Statistics for 2013," http://media.kaspersky.com/pdf/KSB_2013_EN.pdf.

[2] ZDNET Korea, "AhnLAB, New version of malware 'chest' is rapidly increasing," http://www.zdnet.co.kr/news/news_view.asp?artice_id=20130307162220, Mar. 2013.

[3] Smishing, http://www.police.go.kr/portal/main/contents.do?menuNo=200287

[4] N. Megiddo, P. Alto, and K. S. McCurley, "Efficient Retrieval of Uniform Resource Locators," U.S. Patent No. 6957224 B1, Oct. 2005.

[5] D. Antoniades, I. Polakis, G. Kontaxis, E. Athanasopoulos, S. Ioannidis, E.P.Markatos and T. Karagiannis, "We.b: the web of short urls," WWW '11 Proceedings of the 20th international conference on World wide web, pp. 715-724, Mar. 2011.

[6] Nprotect response team official blog, " circulation of APK using mobile messenger," http://erteam.nprotect.com/464.

[7] Soonil Kim, Sunghoon Kim and Donghoon Lee, "A study on the vulnerability of integrity verification functions of android-based smartphone banking applications," Journal of The Korea Institute of Information Security & Cryptology, 23(4), pp. 743-755, Aug. 2013.

[8] 3GPP, "3GPP TS 23.040 - Technical realization of the Short Message Service(SMS)," 3GPP, Dec. 2013.

[9] Radio Layer Interface, http://www.kandroid.org/online-pdk/guide/telephony.html.

[10] Collin Mulliner and Charlie Miller, "Injecting SMS Messages into Smart Phones for Security Analysis," WOOT'09 Proceedings of the 3rd USENIX confer-

ence on Offensive technologies, pp. 5-5, Aug. 2009.
〔11〕 Peter Jay Salzman, Michael Burian and Ori Pomerantz, "The Linux Kernel Module Programming Guide," http://www.tldp.org/LDP/lkmpg/2.6/lkmpg.pdf, May. 2007.
〔12〕 Youngseok Choi, Sunghoon Kim and Donghoon Lee, "Study to detect and block leakage of personal information : Android-platform environment," Journal of The Korea Institute of Information Security & Cryptology, 23(4), pp. 757-766, Aug. 2013.
〔13〕 Jeonghyeok Kim and Moonsun Bae, "Utilization status of domestic Online banking in 2013," The Bank of Korea, Feb. 2014.
〔14〕 Chao Yang, Vinod Yegneswaran, Phillip Porras and Guofei Gu, "Detecting money-stealing apps in alternative Android markets," CCS '12 Proceedings of the 2012 ACM conference on Computer and communications security, pp. 1034-1036, Oct. 2012.
〔15〕 SMSC, http://en.wikipedia.org/wiki/Short_message_service_center.
〔16〕 Vibha Manjunath and Martin Colley ,"Reverse Engineering Of Malware On Android," 2011 The SANS Institute, Aug. 2011.
〔17〕 W. Enck, M. Ongtang, and P. McDaniel, "On Lightweight Mobile Phone Application Certification," Proceeding CCS '09 Proceedings of the 16th ACM conference on Computer and communications security, pp. 235-245, May. 2009.
〔18〕 Sangho Park and Junhyeong Lee, "Proposal of Smishing Prevention System through Android Permission and Authentication," KIISC REVIEW, 23(6), pp. 5-12, Dec. 2013.
〔19〕 Tae Oh, Bill Stackpole, Emily Cummins, Carlos Gonzalez and Rahul Ramachandran, "Best Security Practices for Android, BlackBerry, and iOS," Enabling Technologies for Smartphone and Internet of Things (ETSIoT), 2012 First IEEE Workshop on, pp. 42-47, Jun. 2012.
〔20〕 Tiago Almeida, Jose Maria Gomez Hidalgo, Tiago Pasqualini Silva, "Towards SMS Spam Filtering: Results under a New Dataset," International Journal of Information Security Science, vol. 2 no. 1, pp. 1-18, Mar. 2013.
〔21〕 Anna Kang, Jaedong Lee, Wonmin Kang, Leonard Barolli and Jonghyuk Park, "Security Considerations for Smart Phone Smishing Attacks," Advances in Computer Science and its Applications, Lecture Notes in Electrical Engineering vol. 279, pp. 467-473, 2014

〈저자소개〉

이 시 영 (Si-young Lee) 학생회원
2013년 2월: 수원대학교 산업정보공학과 졸업
2013년 3월~현재: 고려대학교 정보보호대학원 금융보안학과 석사과정
〈관심분야〉 모바일보안, 시스템보안, 금융보안

강 희 수 (Hee-soo Kang) 학생회원
2013년 2월: 중앙대학교 컴퓨터공학부 졸업
2013년 3월~현재: 고려대학교 정보보호대학원 금융보안학과 석사과정
〈관심분야〉 정보보증, 정보보호관리체계, 모바일 보안

문 종 섭 (Jong-sub Moon) 종신회원
1981년 1월: 서울대학교 계산통계학과 졸업
1983년 1월: 서울대학교 대학원 계산통계학과 석사
1991년 5월: Illinois Insitute of Technology 전산학 박사
2002년 3월~현재: 고려대학교 전자 및 정보공학과 교수
〈관심분야〉 정보보호, 전자공학, 통신공학