IntelliDroid: A Targeted Input Generator for the Dynamic Analysis of Android Malware

JAN 5TH, 2016

论文下载：http://www.eecg.toronto.edu/~lie/papers/mwong_mascthesis.pdf

Abstract & Introduction

为动态分析工具提供输入。

传统的硬编码测试，自动fuzzing测试并不科学，虽然覆盖面广，但是可能还是无法触发恶意行为。 IntelliDroid最主要的两个新技术：

1. Android入口很多，包含很多event Handlers，仅仅触发包含敏感API的handler是不够的，很可能需要多个handler按一定顺序触发，才会唤醒恶意行为。IntelliDroid能够决定这些事件及事件的顺序。
2. 事件模拟是在整个framework层，更逼真。例如某个恶意软件会接收收到短信事件后，去数据库删掉该短信。如果仅仅在APP层面向APP发一个广播，而系统实际上没收到短信的话，不会触发恶意行为。IntelliDroid会让整

个系统都完整连贯的触发事件，这样就可以完美契合整个系统范围的动态分析工具，例如TaintDroid。

Design

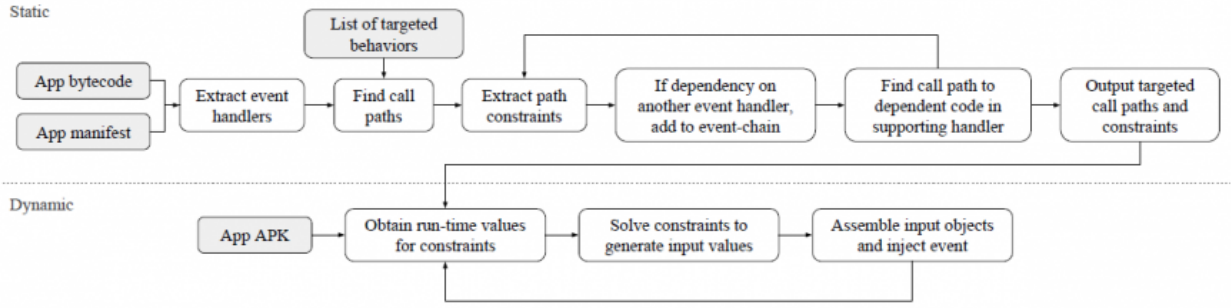

Fig. 1. IntelliDroid System Diagram

1 收集一个敏感API列表：之所以是API而不是system call或更底层的CPU表现等等，是因为监控敏感API的动态分析工具最多。

2 找到触发敏感API的路径：从manifest入手，一直向里找，遇到新的入口就加进来，这样拿到所有入口。从每个入口开始找敏感API，找到就记录下路线。这里遇到intent等隐式调用需特殊处理。

3 提取限制条件：通过控制流分析及数据流的程序分析，找到可行路径的触发条件，过于复杂的地方会有人工介入（a manually implemented function）。最终会得到一系列变量及对应的值，这些值会使程序触发敏感API

4 生成Event-Chains：如果限制条件仅和输入有关就简单了，实际上还和很多堆栈上的值有关。在处理这种复杂的情况时，需要知道什么时间是什么值，这个顺序就是Event-chains。

5 动态运行限制：虽然限制条件是静态生成的，但动态运行时会运用这些条件来实现限制，在程序运行时，动态获得某个变量的值，用来生成输入。作者说他这个功能强大，如果恶意APP从服务器下载一些信息，可以通过

网络监控拦截。如果固定时间触发的恶意行为，可以改系统时间。

6 输入控制：最关键的输入控制是直接向系统层注入的。例如收短信，通过直接想系统framework层插入一个数据来模拟短信。

```
1  class SmsReceiver extends BroadcastReceiver {
2    String sNum;
3
4    void onReceive(Context c, Intent i) {
5      if (i.getAction()=="SMS_RECEIVED") {
6        handleSms(i);
7      } else if (i.getAction()=="BOOT_COMPLETED"){
8        this.sNum = "99765";
9      }
10   }
11   void handleSms(Intent i) {
12     Bundle b = i.getExtras();
13     Object[] pdus = (Object[])b.get("pdus");
14
15     for (int x = 0; x < pdus.length; x++) {
16       SmsMessage msg =
             SmsMessage.createFromPdu(pdus[x]);
17       String addr = msg.getOriginatingAddress();
18       String body = msg.getMessageBody();
19       // Constraint depends on local function
20       if (needsReply(addr, body)) {
21         SmsManager sm = SmsManager.getDefault();
22         sm.sendTextMessage(addr, null, "YES", null,
               null);
23       }
24       // Constraint depends on heap variable
25       if (addr.equals(this.sNum)) {
26         abortBroadcast();
27       }
28     }
29   }
30   boolean needsReply(String addr, String body) {
31     if ((addr.startsWith("10658") &&
32           body.contains("RESPOND")) ||
33         (addr.startsWith("10086") &&
34           body.contains("REPLY"))) {
35       return true;
36     }
37     return false;
38   }
39 }
```

Listing 1. Code Example

上图可以找到路径4→11→22 4→11→26

Evaluation

- event chains:定时启动， 收到信号短信
- device-framework injection: 删除短信
- run time constraint data:网络获取的数据，sharedpreferences