Androguard补完计划——提取加密字符串

2016-10-24 penguin\_wwy 京东安全应急响应中心

点击上方蓝字关注

0X01 概述

上篇分析了Androguard如何读取dex，而且还提到Androguard很适合进行扩展或者移植成为自己项目的某一模块。

本篇文章就来研究一下如何在Androguard基础上进行扩展。

App对抗静态分析的方法之一就是利用反射，如果对反射的字符串进行加密会得到更好的效果，

而且不但反射可以通过加密字符串，凡是动态注册、加载都可以通过加密字符串提高隐蔽性，对

抗静态分析。

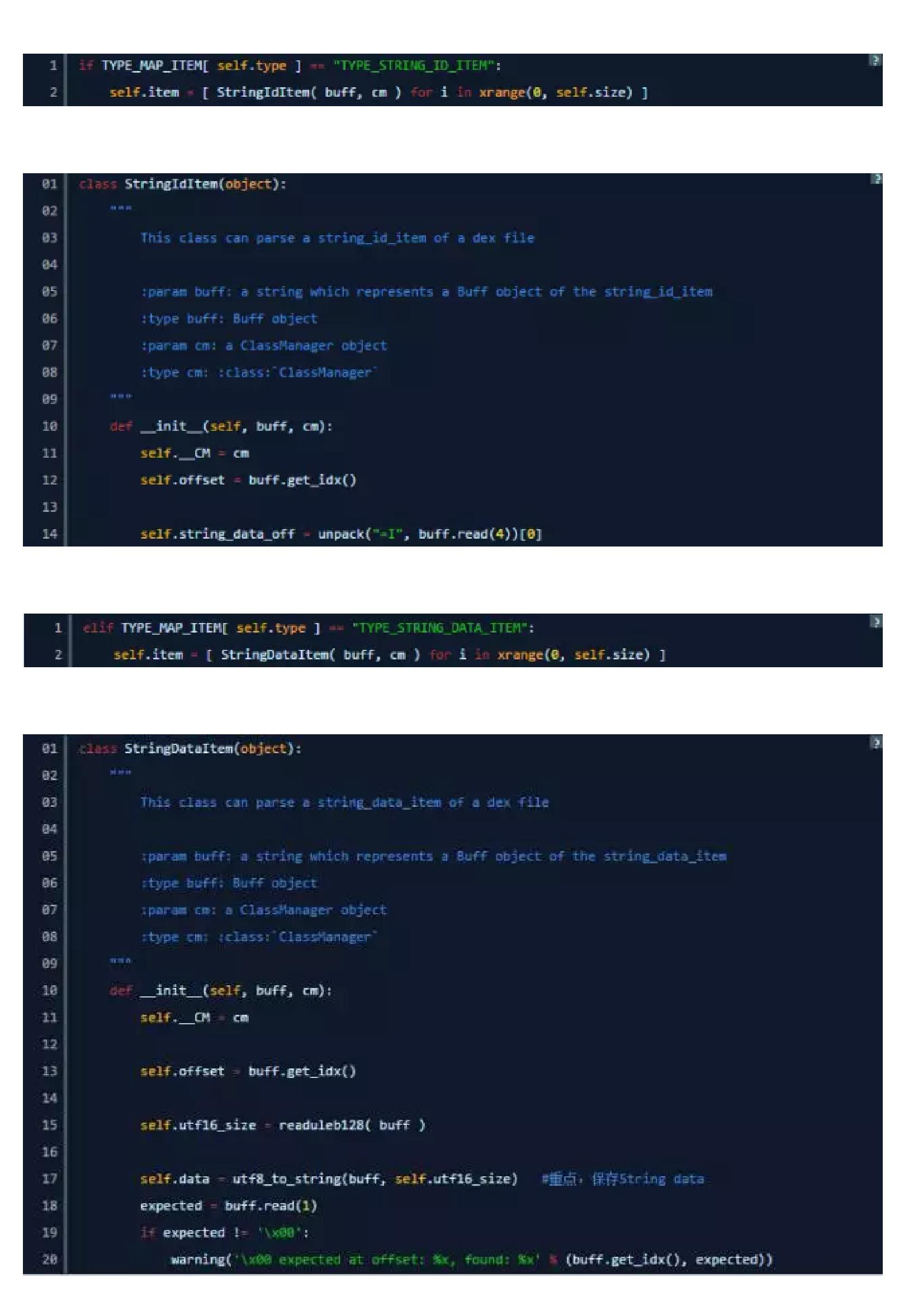
比如这样：

或者这样：

下面我们就给Androguard补充一个功能，提取dex文件中经过加密的字符串。要提取dex文件中的加密字符串，主要两个步骤：1、提取dex文件中的字符串池，获取全部字符串；2、判断是否为加密字符串。

0X02 提取

首先如何提取？上篇提到Androguard读取dex文件对各个item进行处理，处理逻辑在MapItem.next函数中看一下源码，找到跟字符串相关的Item，有两处。

一个是StringIdItem：

主要记录String的偏移：

一个是StringDataItem：

看看代码：

重点在self.data，通过utf8\_to\_string函数将字节码转换为字符串。

我们知道了每个字符串保存在每个StringDataItem.data中，那我们如何获得它们呢。回到next函数，MapItem.Item保存所有StringDataItem组成的列表：

而这个 MapItem 会被加入到 MapList.map\_item 这个队列以及 self.CM 中，当然加入到ClassManager中的过程更复杂。如果从map\_item中获取到字符串，需要首先找到处理StringDataItem的mi，然后遍历map\_item中的所有MapItem对象，依次拿到MapItem.data，这无疑很复杂。那就让我们把目光放到ClassManager上，看看add\_type\_item：

咦，似乎有了意外的发现，当处理到StringDataItem时，会设置一个标志位。当标志位为真时，self.\_\_strings\_off这个字典才会保存数据，也就是StringDataItem相关的数据。

我们来仔细研究一下这段代码，先理解参数。type\_item表示Item的类型，c\_item则是mi = MapItem( buff, self.CM )的mi，也就是一个完整的MapItem对象。参数中的item则是 mi.get\_item( )，也就是MapItem.item。所以当type 为StringDataItem时item就是保存StringDataItem对象的列表。

整理一下思路，现在的情况是我们可以从ClassManager中的\_\_strings\_off字典根据偏移得到每个StringDataItem。但是悲催的是ClassManager当中并没有获得\_\_strings\_off的方法，我们只能自己先加一个：

只要遍历\_\_strings\_off，拿到每个Item，获取data就可以得到字符串了。

类似如下处理：

str\_list就会保存dex文件中的所有字符串了。

0X03 判断加密字符串

得到所有字符串之后，我们就依次判断它是否是加密字符串。如何判断呢？公司倒是有一个判断随机字符串的工具（也就是人类无法识别的字符串），但毕竟是公司的东西，也没有源码。搞一个字典太费劲，而且字典越大也会影响运行时间。我暂时想了一个办法来判断随机字符串。

首先，先弄个小字典，大概十几二十个有关单词（果然是小字典···），先用小字典过滤一下。对于剩下的字符串，将字符分为大写英文，小写英文，和其他字符（除 \ / . ; 以及空格）。对于一般有意义的字符串，ASCII 相对集中，以大写或小写为主。比如ACCESS\_CHECKIN\_PROPERTIES这是权限，Landroid/os/Debug这是类名。而加密后的字符串ASCII分布就会相对随机比如KS9FRUc6HgxFByUhQ1A=:MN1$gNqcet，这三种字符或者其中两种的字符数量相差就不会太大。我们可以统计一个字符串中三种字符的数量，如果其中两种或三种数量相对接近，就认为是随机字符。



这是我写的代码，以供参考。

0X04 测试