[原创]某音1700版本gorgon argus ladon 算法逆向

南柯

极客

1天前


```
突然发现注册看雪到现在快十年了，一直都在暗戳戳的白嫖同行分享的经验和工具，正好在工作中发现一些小盆友在安卓逆向
多新入行的盆友不太好找到复杂保护下的算法逆向经验分享，就此拿去年分析还原的DY1700版本的几个签名参数的算法作为
验。
本贴主旨是逆向方法的分享，会提到算法以及实现过程，但是不会放出实现源码，这个大家应该也能理解，毕竟我本人不是
卓逆向有兴趣的同学一些启示，帮助吧！
首先跟大家讲的是，DY的这套算法并不难，整体逆向的难度算是一般，之所以很多人逆不出来，根本原因还是在基础不够扎
一下的朋友，稳住心态，树立信心，你想这条路上，信心和耐心很重要。另外，务必养成良好的习惯，比如记录分析过程，
```

使用工具：

```
jadx、frida、ida pro、eclipse、android studio、010editor
```

分析方法：

```
首先肯定是定位native函数的入口，这个在其它文章很多人已经说过了，主要是在JAVA层通过hook代码执行流或者代码分析
native函数，再确定该函数在so文件中的偏移位置，在动态分析的时候在此处下断点，这里是libmetasec_ml.so 入口偏移
然后构建动态调试环境，这里很多小伙伴使用各种工具甚至直接在原本的app里面进行动态调试，这种方式真的是非常的不合
方式是自己构建一个安卓工程，将相关模块代码抽取出来，在保证运行正常的情况下，使用自己构建的工程进行动态分析，
动态调试，这里要先知道，代码是树形结构，逆向过程就是还原这个树形结构的过程，通常有两种方式，一种是从根节点开
节点，确定各个分支和节点的功能。这种对于代码结构比较清晰，紧凑的实现是很有效的，也是很多人喜欢采用的。另外一
确定节点所在分支，从而回溯到主干，从主干再扩展到入口和出口，这种情况通常用来应对代码结构分散，执行流程被打乱
定代码执行流程为目的，梳理清楚代码主要结构。这里很多新手很容在没有理顺一个清晰的思路的情况下，一头扎进具体的
来。所以一定要记着，先搞清楚代码轮廓，先把代码的主干，分支，叶子节点画出来，再顺着代码执行流去正向或者反向分
```

DY算法分析：

```
正向执行顺序：8D49C（入口）->19B20(主函数，包括了参数处理和各处加密的调用)-> 19D20(主函数中判断是否有x-ss-
算)-> 1A440(gorgon 分支其中451F0 是gorgon原始参数拼接偏移)-> 1C34A（gorgon计算调用和结果拼装）->1D09A（X
和结果拼装）
```

这里贴出的是相对偏移地址，上面是整体的大概执行流程，按照这个偏移下断点动态调试，整个代码的执行流和功能结构是清晰的，就不做具体说明了，下面讲具体的算法：

一、xgorgon:下面贴出我在分析过程中记录的分析结果，不一定完全准确哈

```
1C346->7FA50->451F0（源数据拼装）->428C8(*)->45058->2CD58
```

x-gorgon 编码前数据构造偏移：44420、44594、44808

256位表变换偏移：440E8

1、对原url的参数部分做md5,，取结果的前四个字节作为加密内容

2、使用8位随机密钥与固定表生成256位的随机编码表：

```
随机密钥格式：1E 00 E0 8A 93 45 80 C0  其中第四位的0x8A和第八          位的0xC0 是随机生成的：
```

使用的固定表：

00 01 02 03 04 05 06 07

08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17

18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27

28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37

38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46 47

48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57

58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67

68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77

88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97
98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7
A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7
B8 B9 BA BB BC BD BE BF C0 C1 C2 C3 C4 C5 C6 C7
C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7
D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7
E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7
F8 F9 FA FB FC FD FE FF
最终生成的随即表如下：
1E 1F 1F F4 8B D5 5B 82
A8 B1 9B 96 35 87 15 44 72 83 75 78 1F 79 44 46
7C 95 8F 9A 49 AB 49 88 C6 E7 E9 FC B3 AB C3 9B
50 79 83 9E 5D CF 7D CC 8F 4B 5D 80 47 C1 77 CE
B3 5D 77 A2 71 F3 B1 72 6E AF D1 8B DB 65 9E 92
F8 AF 6B A6 85 46 E5 54 C2 78 65 88 6F B1 DF DF
CC AB 5F AA 99 A2 95 98 44 77 B9 35 F4 AD 93 8F
A0 B1 88 AE AD 98 46 DC 88 DB CF 90 97 78 92 DE
97 ED 92 B2 C1 83 81 C6 BE 72 A1 1F 9E F5 FB A2
F8 D1 A2 B6 D5 A7 B5 F4 75 A3 79 98 BF 99 AF 93
49 B5 CC BA E9 CB E9 A8 93 82 D1 49 88 F3 35 83
F0 B5 FC BE FD EF AB EC BA AE FD 93 E7 E1 46 EE
C4 83 46 C2 83 78 78 8F 15 CF DB B3 B2 F5 CB FD
49 77 96 CB AB CE F5 97 B9 80 E5 F0 44 79 C6 92
AD 65 FF 96 5D AA 83 C4 46 93 AB 5D 49 CE 80 77
6E 79 F3 F5 46 C6 C6 FC 9B FB CE BA CE DB 77 FE
1F 87 77 E5 77 49 E7 6E

3、使用2步骤生成的随即表，与一个固定表共同对原数据串进行编码：

04 00 00 F0 00 80 F2 A2 E2 00 00 00 00 00 00 00
00 00 00 03 03 04 61 C4 1E 4F
其中（04 80 是固定值，f0是2步骤中随机密钥的第四位f2 a2 e2 00 是url计算md5结果的前四位，03 03 04 是加密算法版本号v4.3.3 ，61 c4 1e 4f 是时间戳，也是X-Khronos 的值）
编码结果：
04 04 20 F0 00 80 23 43 60 05 77 72 72 CF 13 52
8F DE 2C 36 EF 35 38 A4 CA AB
其中第二个04 是固定值，20为步骤2中随机密钥的第八位

4、对步骤3的编码结果进行最后一步编码：

```
编码表为："0123456789abcdef"
```

（从上面的计算过程可以看出，X-GORGON的计算结果是可以再服务端解密的哦，当然也可以自己实现解密算法）

二、x-argus:

```
7FA50->457C4->4073C->404D0->4545C->(40616拼装摘要结果)->F3A0->F650->459A0->45BAE(重要计算偏移对称加密)
1、由两个固定串（y9b0xCYrB++Slss+AeCCmqg/7xiNzqCCzZ2pYLjGgVY=
```

jr36OAbsxc7nlCPmAp7YJUC8Ihi7fq73HLaR96qKovU=这两个串是作为参数传入的一个固定串解密出来的） 计算得出
Signkey 计算方法偏移：76230
8E BD FA 38 06 EC C5 CE E7 94 23 E6 02 9E D8 25
40 BC 22 18 BB 7E AE F7 1C B6 91 F7 AA 8A A2 F5

对 8E BD FA 38 06 EC C5 CE E7 94 23 E6 02 9E D8 25 做md5
signkey：F1 59 33 76 76 6E A9 8D 34 F3 1B 05 7A 9D 5B E4

对 40 BC 22 18 BB 7E AE F7 1C B6 91 F7 AA 8A A2 F5 做md5
1F E1 09 A4 12 52 83 F4 18 DE 9E 05 1A 96 9E 12

加密源串拼接：（使用protobuffer 的序列化方式）位置：F3A0->F650
固定串08 D2 A4 80 82 04 10 02 18 +随机值+1128+ddid+1588093228+1.0+v04.03.03-ml-android+..... （其中由当前时间戳变换计算）+ 某个加密结果的前六位+ 另一个个加密结果的前六位+一个结构体+当前时间戳变换结果

其中两个加密结果的前六位中，第一个是根据url的sm3摘要计算出来的，第二个是16位的空byte数组做sm3摘要计算出来的。

摘要算法偏移458A0。
08 D2 A4 80 82 04 10 02 18 E8 A4 FA 6A 22 04 31
31 32 38 32 0A 31 35 38 38 30 39 33 32 32 38 3A
03 31 2E 30 42 14 76 30 34 2E 30 33 2E 30 33 2D
6D 6C 2D 61 6E 64 72 6F 69 64 48 80 8C 98 40 52
08 00 80 00 00 00 00 00 00 60 A2 DE A9 9D 0C 6A
06 10 6E 34 A2 B8 C7 72 06 F4 A0 38 C4 4E E2 7A
0A 08 38 10 BE E1 54 18 BE E1 54 88 01 A2 DE A9
9D 0C A2 01 04 6E 6F 6E 65 A8 01 F0 04 00 C7 42

```
对加proto结果进行加密:
先对signkey 加上随机串 加上signkey 拼接的成的串进行sm3摘要，生成32位摘要结果。
```

如:
8E BD FA 38 06 EC C5 CE E7 94 23 E6 02 9E D8 25
40 BC 22 18 BB 7E AE F7 1C B6 91 F7 AA 8A A2 F5
01 63 54 55 8E BD FA 38 06 EC C5 CE E7 94 23 E6
02 9E D8 25 40 BC 22 18 BB 7E AE F7 1C B6 91 F7
AA,8A,A2,F5
其中01 63 54 55为随机值
2、使用1步骤的32位摘要结果作为密钥，对protobuf串进行对称加密这里为自定义的对称加密算法（具体算法就不贴了，自己分析实现一下，这里还是挺有意思的）。

```
3、对步骤2的结果，先在头部添加00 80 00 00 00 00 00 00然后从后向前，同随机byte数组（0xFF 0xFA 0x57 0x40，
做异或操作，结果为待组装的元数据：（关键偏移：
```

取值偏移：48DF4
取数据地址偏移：47E6C
异或计算偏移：48D9C
最终赋值偏移：47880
算法关键跳转偏移1：47C78
)
结果赋值偏移：47F44
4、对加密结果进行组装:
如:
35 E0 FA 83 69 01 10 0D 18 AE 96 E8 07 56 5E 64
84 97 EE FD C4 08 77 3B 25 EC 4B 15 DD 53 88 6E
37 DA B5 57 16 FE A5 CC B0 93 46 F6 86 1E 34 75
28 36 BD 22 CF C6 4B CA DE 77 61 09 04 3A 73 1B
E4 B5 B0 DB E7 88 A0 2E C5 08 F1 5D 19 6C B4 72
98 B6 AE 0E 53 DC 95 BE B8 6D 2B BE 38 EB AE 5C
2A 0F 05 26 6E 1E 9A 46 A6 B5 A8 26 B7 AC 26 B4
8E 3D 8C 84 2A F2 7E D2 A6 7D 5B 80 2D D1 CB A9
08 45 0A A0 7F BE 8D 70 07 FF FB F7 67 FF FB 77
67 81 1D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D

35 AC 50 86 00 01 10 0D 18 作为头部
其中35为固定值，AC 50 86 00为随机值01 10 0D 18为固定值（其中10 和0d 不是固定值，是由组装后计算出来的0d由源字符串的sm3结果的第一byte位经过(第一个byte位&0x3f)计算得到，10由x-ss-stub加密结果经过sm3摘要后的第一byte位&0x3f 得到）(81 1d)为步骤1中随机值的前两位，这里应该是解密时要用。

使用（F1 59 33 76 76 6E A9 8D 34 F3 1B 05 7A 9D 5B E4）作为key （1F E1 09 A4 12 52 83 F4 18 DE 9E 05 1A 96 9E 12）作为iv 对4的结果进行aes cbc模式的加密，填充模式为PKCS7Padding。

```
6、对步骤5的aes结果进行编码：
```

将aes的头部加入最开始的随机值的后两个字节，然后进行base64编码

 # 三、X-Ladon：

7FA50(md5)->2CD58(hexstr编码)-> 4648C(密钥扩展算法)-> 465CC(对称加密)-> 76134（base64）

1、随机串+1128 计算md5

2、对1 的md5结果进行hexstr编码

3、将2 的编码结果作为key与1644989778-1588093228-1128 进行自定义的对称加密

4、对3的加密结果前面拼接1的随机串，进行base464

```
我是将DY的算法还原成了java代码，对x-argus 实现了加密和解密，分析过程中发现DY的自定义加密还是有点意思的，这里
调试工程这里就不发了，不是敝帚自珍，而是我写本帖的目的就是希望分享分析过程和分析方法，希望我的分析文档和思路能
助，我认为我已经提供了最关键的信息，剩下的就是需要有耐心，愿意动手去自己分析了。至于其中涉及到的llvm混淆和vmp
段的LLVM的混淆水平，我认为还没到严重影响分析的程度，更不需要专门写工具去做去混淆。而所谓VMP并不是真正的讲指令
而已，这种方式通常都有关键的跳转地址计算函数，跟踪这里就很容易捋清楚调用过程。当然，这需要耐心！实际上自己动手
记，逆向实践技术，实践大于一切！
第一次在看雪写东西，不太会排版哈，专栏也刚申请还没结果，这就先写到这吧！有感兴趣的同学搞完这个还可以试试穿山甲
分享！
```

[看雪招聘平台创建简历并且简历完整度达到90%及以上可获得500看雪币～]

最后于 1天前 被南柯编辑 ，原因： 补充执行流

收藏 · 6　点赞 · 1　打赏　分享

最新回复 (3)

mb_rjdrqvpa 15小时前　2 楼　0

穿山甲的设备指纹我记得不难，和抖音有差距，为何说有趣？
我打算过段时间发帖，附成品算法

极客

最后于 15小时前 被mb_rjdrqvpa编辑 ，原因:

黑屏 3小时前　3 楼　0

顶一下 大佬 期待大佬更多作品

极客

南柯 29分钟前　4 楼　0

，穿山甲跟抖音的设备指纹是两个团队，差距是有点，但感觉也没那么大，我觉得是魔改的时候一些细节没注意到然后留下了尾巴！穿山甲的指纹采集是单独的一个模块，我弄了差不多一半然后有紧急任务给打断了……现在穿山甲和shopee都差一点没弄完，又在搞阿里了……回头都得补上！如果大佬分享出来，让我们可以学习学习当然是极好的啦，不过成品算法还是建议大佬谨慎哈……还是有点风险的

极客