看雪 2017 安全开发者峰会

Kanxue 2017 Security Developer Summit

2000-2017

移动APP 灰色产业案例分析与防范

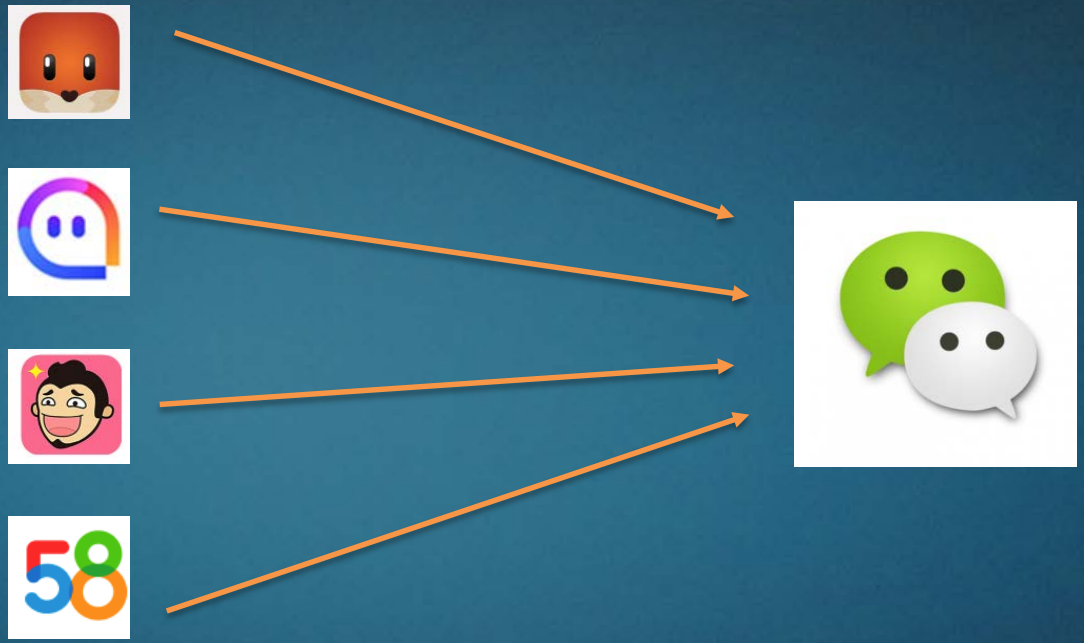
无名侠

看雪峰會
2017

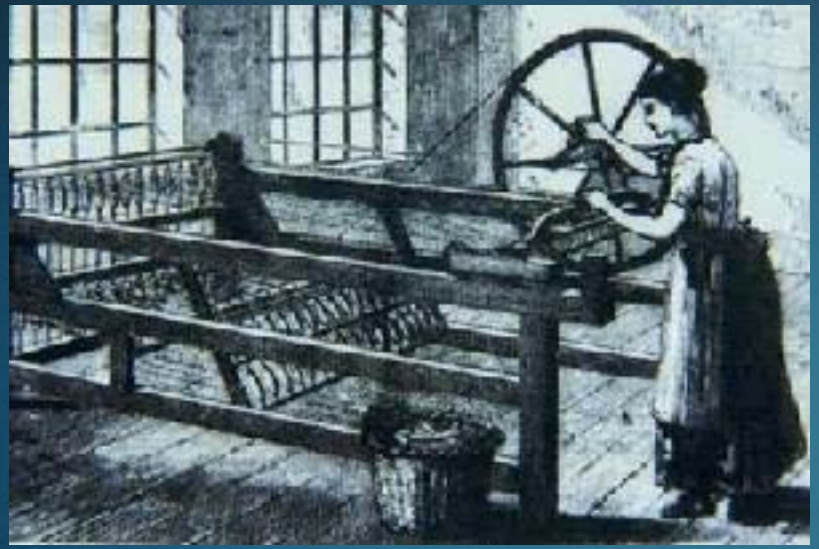
薅羊毛，你玩过吗？

- 羊毛党的工业革命
- Android APP为何成为利益突破口？
- 谈谈小米抢购软件的实现
- 如何从应用层面提高业务安全？

引流



羊毛党的“工业革命”



手动薅羊毛 => 自动化脚本

提前知道x活动 -> 找合伙人 / 注册小号 / 准备vps / 找技术

活动上线 -> 找人脱壳 -> 找人分析协议 -> 找人逆向算法

Android APP为何成为利益突破口？

1. Android 应用极易被反编译和调试
2. 大量能被利用的应用均有Android端
3. 脱壳、协议分析、算法逆向分工明确，高效合作

【已经和谐】

原理？

模拟数据包
类似于爬虫

难点 – 协议分析

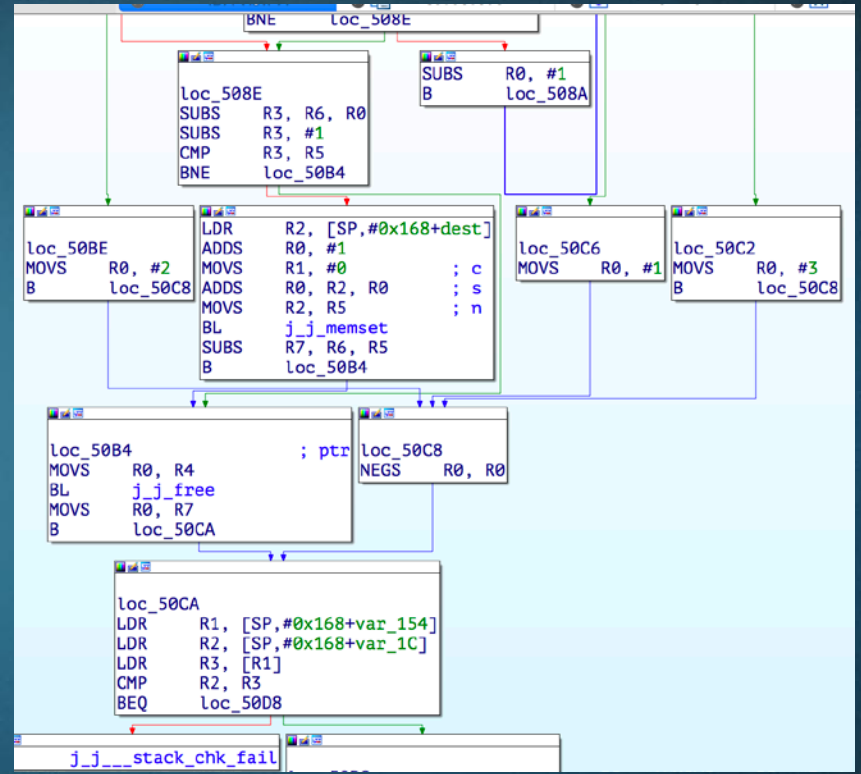
已经和谐

逆向分析APP，寻找password算法

```
public void a(String arg4, String arg5, boolean arg6) {
    if(arg4 != null && arg5 != null) {
        this.y = arg4;
        this.j.a("username", this.y);
        this.x = arg5;
        this.j.a("password", this.x);
        if(arg6) {
            this.j.a("sync", 1);
        }

        this.j.a("appid", "yidian");
```

Description | Source



密钥 / 加密数据嗅探

CryptoFucker，Xposed插件，用于抓取javax.crypto.* 与 javax.security.* 算法参数（包括加密数据、密钥、IV、结果等数据）的工具。

https://github.com/Chenyuxin/CryptoFucker

使用这个工具可测试应用内是否使用了不安全的固定密钥。

密钥 / 加密数据嗅探（续）

```
10-10 02:33:29.218 958-958/? I/Xposed: [me.yidui]MD5 update data:
                                       com.tanliani.utils.MD5->getSign  MD5.java(39)
                                       ──────────────────────────────────
                                       com.yidui.NewLoginActivity->apiPutCaptcha  NewLoginActivity.java(293)
                                       ──────────────────────────────────

                                       0x00000000 30 30 30 31 31 33 33 33 34 35 35 35 36 37 37 37 0001133345556777
                                       0x00000010 38 38 38 39 39 00 00 00 00 00 00 00 00 00 00 00 88899...........
                                       0x00000020 00 00 00 00 00                                  .....
                                       ──────────────────────────────────
10-10 02:33:29.242 958-958/? D/q_me.yidui: SHA-1 update data:
                                           com.android.volley.InternalUtils->sha1Hash  InternalUtils.java(32)
                                           ──────────────────────────────────
                                           com.android.volley.Request->createIdentifier  Request.java(633)
                                           ──────────────────────────────────

                                           0x00000000 52 65 71 75 65 73 74 3A 30 3A 68 74 74 70 3A 2F Request:0:http:/
                                           0x00000010 2F 73 74 61 74 2E 6D 69 6C 69 61 6E 74 65 63 68 /stat.miliantech
                                           0x00000020 2E 63 6F 6D 2F 61 70 69 2F 63 6F 75 6E 74 3A 31 .com/api/count:1
                                           0x00000030 35 30 37 35 37 34 30 30 39 32 34 36 3A 38 00 00 507574009246:8..
                                           0x00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
                                           ──────────────────────────────────
10-10 02:33:29.246 958-958/? I/Xposed: [me.yidui]SHA-1 update data:
 6: Android Monitor   0: Messages   9: Version Control   Terminal
```

千里之提，毁于蚁穴

千里之提，毁于蚁穴

- 某米电视盒子“自带抓包”“辅助调试”

  截图已经和谐

千里之提，毁于蚁穴

- 安全类So如果有x86版本…… **Instagram**

```
20
21  v3 = a3;
22  v4 = a1;
23  v19 = _stack_chk_guard;
24  v5 = (a1->functions->GetByteArrayElements)();
25  v15 = (v4->functions->GetArrayLength)(v4, v3);
26  std::string::string(&v17, "fb26667d85c4432ee34e8e69876575a2", &v16);
27  v6 = Scrambler::getString(&v17);
28  std::string::~string(&v17);
29  sub_9B8();
30  v7 = strlen(v6);
31  sub_98C(&v18, v6, v7);
32  sub_A7C(&v18, v5, v15);
33  sub_A88(&v18, &v14, v8);
34  (v4->functions->ReleaseByteArrayElements)(v4, v3, v5, 0);
35  v9 = sub_9B8();
36  v10 = 0;
37  v11 = operator new[](2 * v9 + 1);
38  while ( v10 < sub_9B8() )
39  {
40    snprintf(&v11[2 * v10], 3u, "%02x", *(&v14 + v10));
41    ++v10;
42  }
43  v12 = (v4->functions->NewStringUTF)(v4, v11);
44  operator delete[](v11);
45  result = v12;
46  if ( v19 != _stack_chk_guard )
47    _stack_chk_fail(v12);
48  return result;
```


载入ELF文件

↓

修复重定位

↓

执行

Ollvm 混淆后的算法如何利用？

- Unicorn framework：https://github.com/unicorn-engine/unicorn/
- X86 生产环境中模拟执行SO中的代码
- 防范建议：增加SO中目标函数的上下文依赖性。

千里之提，毁于蚁穴 —— 套壳保平安

- 你加的壳很难脱，但是你用的加密算法很简单。
- 你加的壳很难脱，但是你用的壳只会保护dex文件。

如何增加安全强度？

自改标准算法 —— 两种变异的Base64

1、替换Base64符号，某APP某处算法（2016）替换“=”为“B”
2、置换初始化表+异或改进

```
63      v9 = 0;
64      v10 = "OPWvYny#Nopz0$HI34QRSG@dJKq7FghD9Zi*kAB8rsFu56L&Ca^2tTUVEewx1m+/=";
65      do
```

```
do
{
  if ( *(_BYTE *)v11 == *(_BYTE *)(v10 - 2) )
    v13 = v12 ^ (v12 >> 4);
  ++v12;
  v11 = (__int128 *)((char *)v11 + 1);
}
while ( v12 < 0x40u );
v14 = &v26;
v15 = -1;
v16 = 0;
do
{
  if ( *(_BYTE *)v14 == *(_BYTE *)(v10 - 1) )
    v15 = v16 ^ (v16 >> 4);
  ++v16;
```

自改算法 —— Tea

```
unsigned int *__fastcall sub_514(unsigned int *result, unsigned int *a2, int a3)
{
  unsigned int v3; // [sp+14h] [bp-1Ch]@1
  unsigned int v4; // [sp+18h] [bp-18h]@1
  int v5; // [sp+1Ch] [bp-14h]@1
  signed int i; // [sp+20h] [bp-10h]@1

  v3 = *result;
  v4 = *a2;
  v5 = 0;
  for ( i = 0; i <= 7; ++i )
  {
    v5 -= 0x61C88647;
    v3 += ((v4 >> 5) + *(_DWORD *)(a3 + 4)) ^ (16 * v4 + *(_DWORD *)a
    v4 += ((v3 >> 5) + *(_DWORD *)(a3 + 12)) ^ (16 * v3 + *(_DWORD *)
  }
  *result = v3;
  *a2 = v4;
  return result;
}
```

```
int __fastcall TeaEncryptECB(int a1, int a2, int a3)
{
  unsigned int v3; // r4@1
  unsigned int v4; // r3@1
  int v5; // r5@1
  int v6; // r1@3
  int result; // r0@5
  int v8; // [sp+10h] [bp-28h]@2
  int v9; // [sp+14h] [bp-24h]@4
  int v10; // [sp+18h] [bp-20h]@4
  int v11; // [sp+1Ch] [bp-1Ch]@4

  v3 = (*a1 >> 24) | (*a1 << 24) | ((*a1 & 0xFF00) << 8) | ((*a1 & 0xFF0000u) >> 8);
  v4 = (*(a1 + 4) << 24) | (*(a1 + 4) >> 24) | ((*(a1 + 4) & 0xFF00) << 8) | ((*(a1 + 4) & 0xFF0000u) >> 8);
  v5 = 0;
  do
  {
    *(&v8 + v5) = (*(a2 + v5) >> 24) | (*(a2 + v5) << 24) | ((*(a2 + v5) & 0xFF00) << 8) | ((*(a2 + v5) & 0xFF0000u) >> 8);
    v5 += 4;
  }
  while ( v5 != 16 );
  v6 = 0;
  do
  {
    v6 -= 0x61C88647;
    v3 += (v4 + v6) ^ (16 * v4 + v8) ^ ((v4 >> 5) + v9);
    v4 += (v3 + v6) ^ (16 * v3 + v10) ^ ((v3 >> 5) + v11);
  }
  while ( v6 != 0xE3779B90 );
  result = (v4 >> 24) | (v4 << 24) | ((v4 & 0xFF00) << 8);
  *a3 = (v3 >> 24) | (v3 << 24) | ((v3 & 0xFF00) << 8) | ((v3 & 0xFF0000) >> 8);
  *(a3 + 4) = result | ((v4 & 0xFF0000) >> 8);
  return result;
}
```

使用Ollvm 编译

1、Ollvm 混淆强度非常高，混淆后难以阅读分析。
2、代码加强上下文依赖。

自实现vm

看雪CTF很多类似的题目。

1、非通用vm实现难度低，可根据自身业务需求设计。
2、bytecode 更新灵活，可通过network随时更新or动态生成。
3、可改python、lua等解释器

Http采用一些序列化框架

- Google Protobuf
- Facebook Thirft
- …….

- String -> Bin

谢谢