APKBUS

Android应用安全提升攻略

侯海飞

大纲

1. 应用安全风险案例 & 常见安全隐患
2. 应用安全活动周期
   - 安全开发
   - 应用审计
   - 安全增强
   - 渠道监控
3. 安卓加固常见策略
   - 应用加固
   - 源码混淆

应用安全风险案例

网易易盾

■ **密码学误用导致防御绕过**

某外卖APP的数据加密实现于native层，密钥硬编码，且加密逻辑未保护，导致加密数据被破，进一步致使网络访问验证被攻陷，可以用于伪造任意链接发起攻击，比如撞库获取用户信息、伪造电话攻击、业务薅羊毛等。

■ **Webview组件使用不当导致应用被克隆**

file协议的权限设置不当，攻击者发送一条恶意链接给用户即可完成攻击，通过Url Scheme方式打开被攻击APP中导出的Webview activity，使之加载恶意链接中的html文件，导致本地数据泄露，并进一步实现应用克隆。

■ **易被逆向，对核心业务逻辑保护不足**

某照片恢复app，未采取任何逆向对抗措施，导致核心付费逻辑被改写，重打包后可免费使用付费功能。

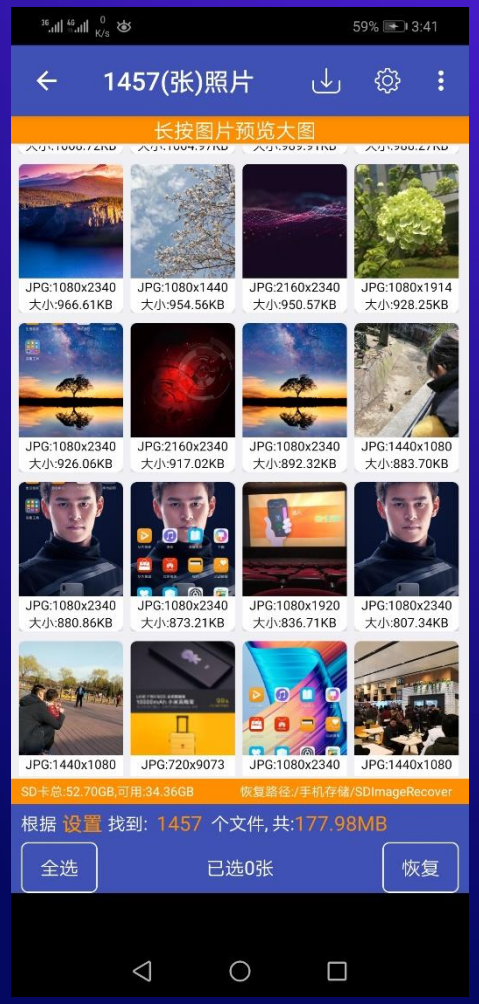
注：相关漏洞均已修复

应用安全风险案例/某照片恢复app的攻防策略

网易易盾

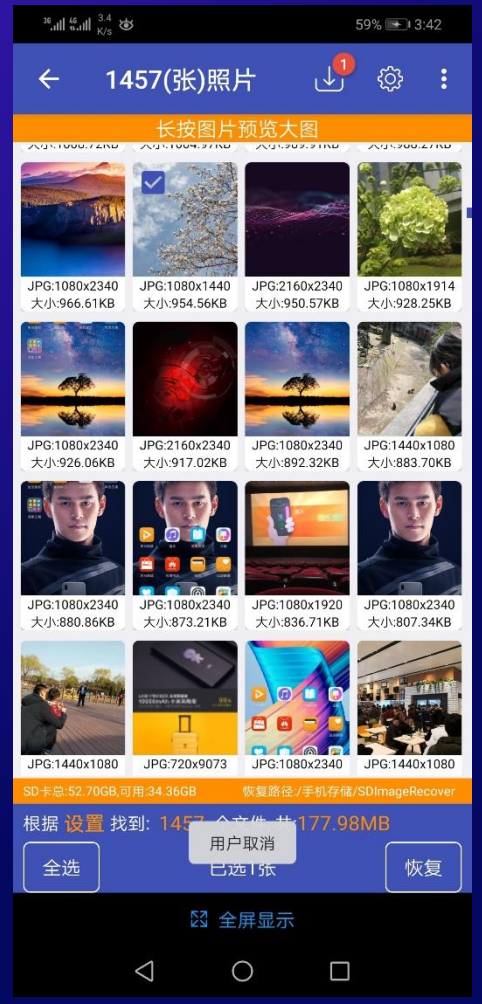
a) 业务场景

(1) 初始



(2) 扫描照片

(3) 付费流程



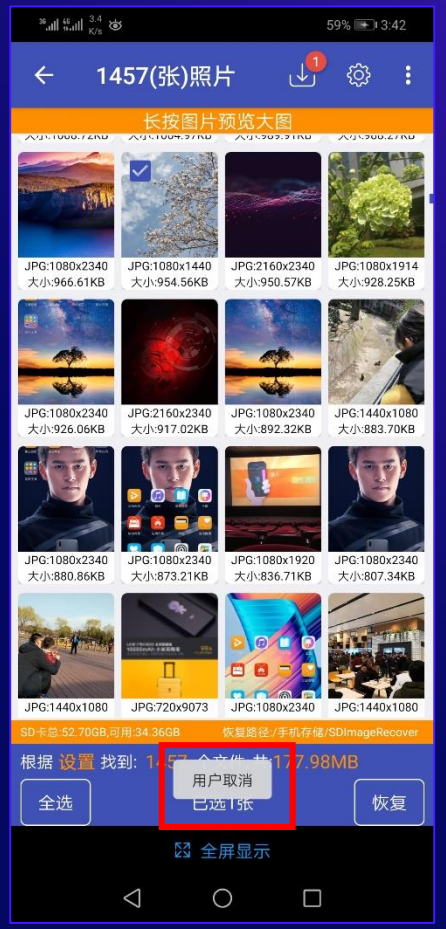
(4) 付费结果校验

应用安全风险案例/某照片恢复app的攻防策略

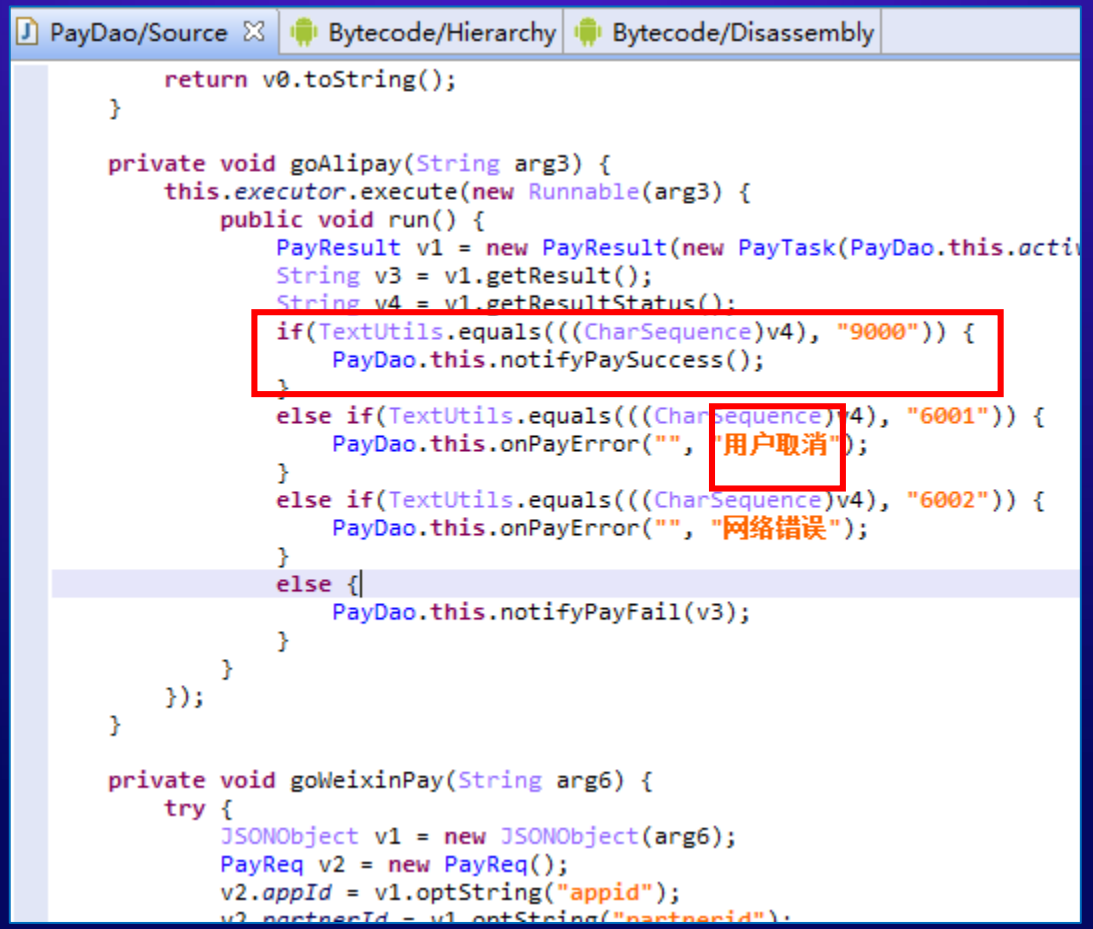
网易易盾

b) 攻击过程

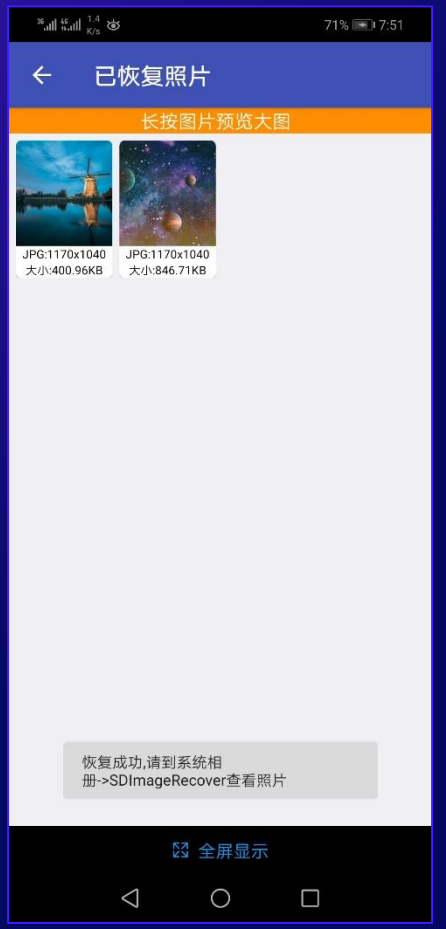
基本思路：目标场景执行在客户端，直接逆向客户端，更改代码控制流即可



(1) 寻找突破口



(2) 逆向，根据线索定位相关控制流，并更改



(3) 效果确认

应用安全风险案例/某照片恢复app的攻防策略

网易易盾

c) 防护策略

- 核心业务与服务端关联

  照片恢复的步骤拆分，将部分内容传入服务端，由服务端校验付费结果，再下发内容给客户端

- 字符串加解密

  代码的明文字符串，需要加密，防止破解者直接搜索到线索

- 防重打包

  启动校验签名，非原签名则直接挂机，或者无法正常运行核心逻辑

- 核心逻辑抽取至native层实现

  增加逆向门槛

- 代码混淆

  java、资源、c/c++均可混淆，增加逆向难度

常见安全隐患汇总

网易易盾

代码可逆向

App可被逆向，轻易获取代码逻辑，进一步导致控制流被hook，防线被破

功能泄漏

客户端APP高权限行为和功能被其他未授权的应用程序调用访问

APP可调试

客户端APP能够被调试，动态的提取、修改运行时的程序数据和逻辑

日志信息泄漏

客户端APP将开发调试信息打印泄露，包含敏感参数和执行流程信息

可二次打包

APP可被修改， 被套壳加入攻击者代码，重新打包发布

密码学误用

客户端APP代码中使用了不安全的密码学实现，例如固定硬编码的对称加密，ECB模式的对称加密，CBC模式中IV固定等

敏感信息泄漏

客户端APP代码中泄漏敏感数据，如认证使用的共享密钥、不应被暴露的后台服务器ip地址等

通信数据明文传输

客户端APP与服务器端交互的数据通过明文的通信信道传输，或者加密传输，但数据依然可以被解密

大纲

1. 应用安全风险案例 & 安全隐患总结
2. 应用安全活动周期
   - 安全开发
   - 应用审计
   - 安全增强
   - 渠道监控
3. 安卓加固常见策略
   - 应用加固
   - 源码混淆

应用安全活动周期

网易易盾

应用安全活动周期/安全开发

网易易盾

开展SDL培训，强化安全开发意识，使用较成熟的安全组件，相关技术点参考下表

密码输入	攻击点	/dev/input/event可以读取到按键和触屏，且全局录屏，也可观察到输入过程。
	开发建议	实现安全键盘随机布局，并禁止输入界面的录屏、截屏。
加密算法与秘钥存储	攻击点	硬编码秘钥极易被逆向获取；低强度加密算法不安全。
	开发建议	a）避免硬编码密钥在代码中，建议将加密过程置于native层，并增加字符串加密，逻辑混淆　等，也可考虑使用白盒密钥; b）Hash算法使用SHA-256代替MD5；AES不要使用ECB模式，初始化向量IV不要使用固定的常量；使用/dev/urandom或者/dev/random来初始化伪随机数生成器，代替SecureRandom。
数据通信	攻击点	中间人攻击，数据协议破解。
	开发建议	网络访问数据加密可置于native层；推荐使用https代替http，并做防抓包对抗。
运行环境	攻击点	攻击者通常需要root、模拟器、xposed框架环境。
	开发建议	初级对抗攻击，检测运行环境。
应用签名	攻击点	篡改应用代码，重打包正常运行。
	开发建议	native层校验签名（也可是其它文件）。
核心逻辑保护	攻击点	Java层代码攻击门槛低，易于逆向。
	开发建议	核心逻辑在native层实现，结合逻辑混淆，提高逆向门槛。
调试日志	攻击点	日志可能包含运行过程和数据信息，可能存在攻击点。
	开发建议	发布版本，去除日志信息。
应用数据	攻击点	本地存储信息，可能存在敏感数据。
	开发建议	本地存储转至native层操作，并加密敏感数据。

应用安全活动周期/安全扫描与渗透测试

网易易盾

（一）安全扫描

扫描内容
- 常见漏洞，可参考“公安三所”漏洞项
- 容易被攻击的技术点

扫描方法
- 静态扫描 反编译apk，静态代码查漏
- 动态扫描 运行时分析，运行时查漏

第三方服务
- 各大应用加固厂商
- 国家安全部门（公安部、网安等）
- 开源扫描框架 Mobile Security Framework

（二）渗透测试

人工测试
- 测试人员需要有专业的安全渗透技能，一般需要找特定安全审查机构

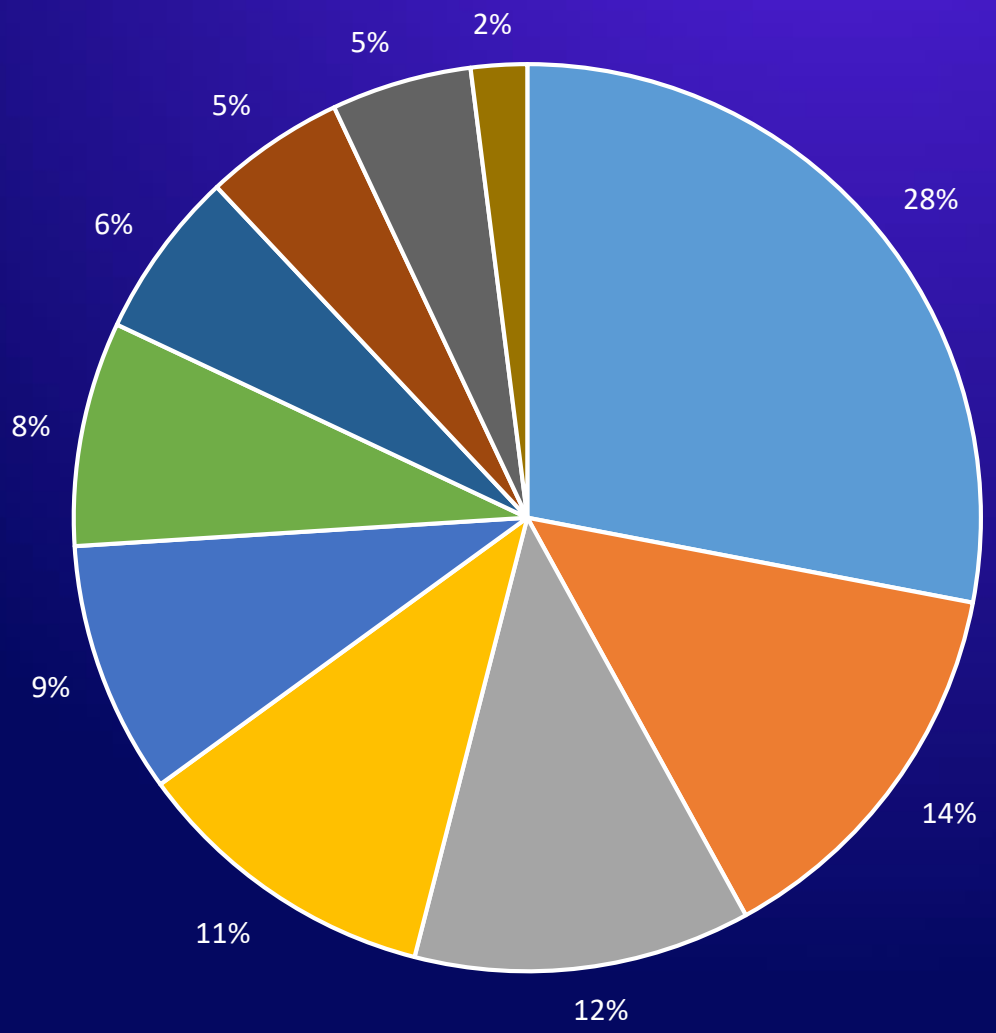
目标场景
- 针对特定场景做攻击，场景涵盖核心逻辑，例如用户登录、支付场景、数据安全、通信安全、拒绝服务等

周期较长
- 相较于安全扫描，渗透测试根据应用场景规模，测试周期一般需要一周或更长

应用安全活动周期/安全扫描与渗透测试

网易易盾

2018 年金融类应用高危漏洞 TOP10



- ContentProvider敏感信息泄露
- 不安全Zip文件解压
- 服务端证书弱校验
- 客户端XML外部实体注入
- Intent Scheme URL攻击
- WebView远程代码执行
- 不安全的内部存储权限
- Fragment注入
- Janus签名漏洞

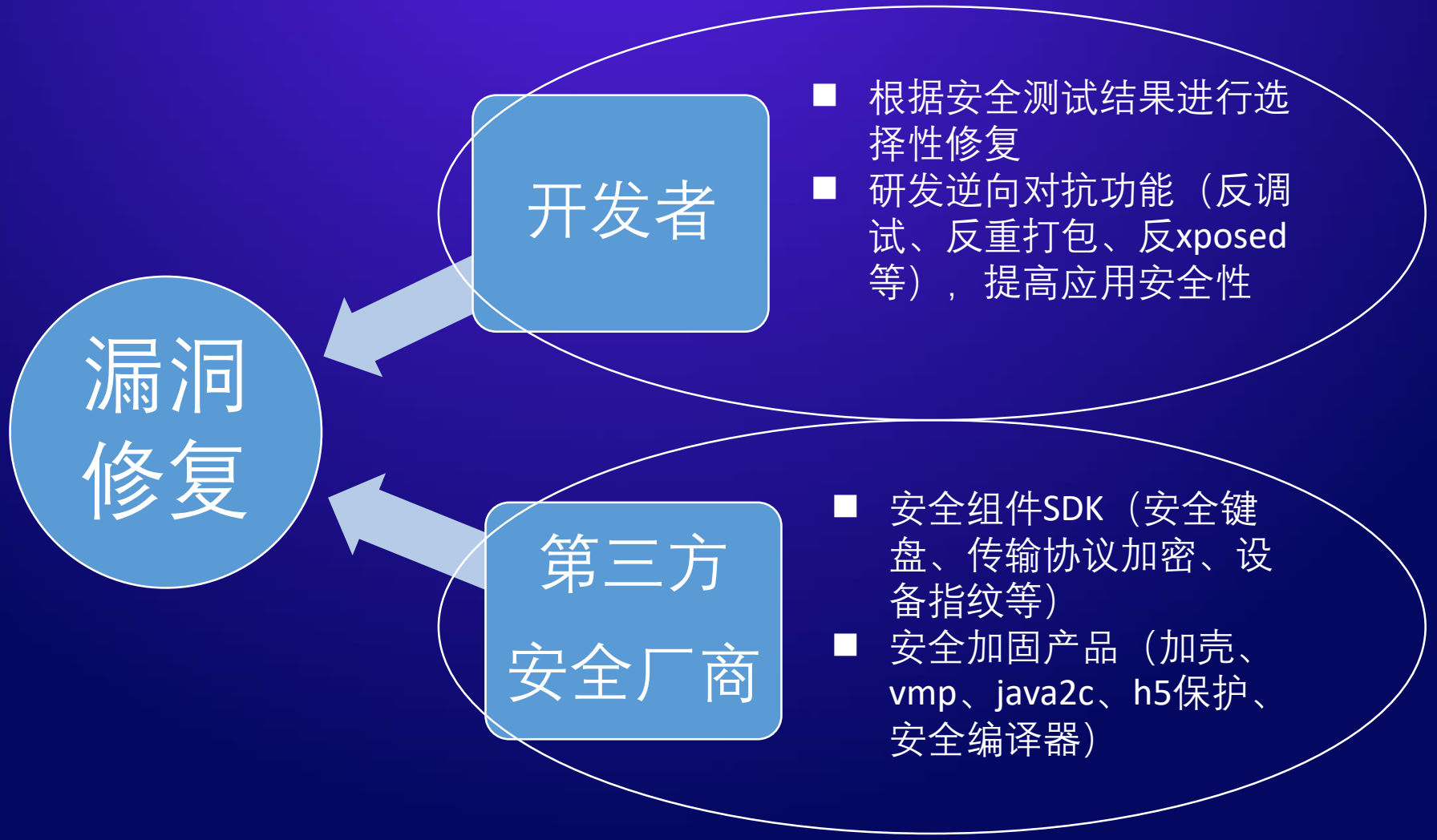
数据来源：云测

	测评项目	危险等级	测评结果
自身安全（6项）			
1	权限信息	低	安全
2	行为信息	低	安全
3	病毒扫描	高	安全
4	敏感词信息	中	存在敏感词(发现3处)
5	广告SDK检测	低	安全
6	第三方SDK检测		
7	加固壳识别		
8	Java代码反编译风		
9	So文件破解风险		
10	篡改和二次打包风险	高	安全
11	Janus签名机制漏洞	高	存在漏洞(发现1处)
12	资源文件泄露风险	中	存在风险(发现31处)
13	应用签名未校验风险	中	存在风险(发现1处)
14	代码未混淆风险	低	安全
15	使用调试证书发布应用风险	低	存在风险(发现1处)
本地数据存储安全（24项）			
16	Webview明文存储密码风险	高	安全
17	Webview File同源策略绕过漏洞	高	安全
18	明文数字证书风险	高	存在风险(发现1处)
19	调试日志函数调用风险	高	存在风险(发现2处)
20	数据库注入漏洞	高	安全
21	AES/DES加密方法不安全使用漏洞	高	安全
22	RSA加密算法不安全使用漏洞	高	安全
23	密钥硬编码漏洞	高	安全
24	动态调试攻击风险	高	存在风险(发现1处)
25	应用数据任意备份风险	中	安全
26	敏感函数调用风险	中	安全
27	数据库文件任意读写漏洞	中	安全
28	全局可读写的内部文件漏洞	中	安全
29	SharedPreferences数据全局可读写漏洞	中	安全
30	SharedUserId属性设置漏洞	中	安全
31	Internal Storage数据全局可读写漏洞	中	安全
32	getDir数据全局可读写漏洞	中	安全
		中	
		中	
		中	
通信数据传输安全（5项）			
40	HTTP传输数据风险	高	安全
41	HTTPS未校验服务器证书漏洞	中	安全
42	HTTPS未校验主机名漏洞	中	安全
43	HTTPS允许任意主机名漏洞	中	安全
44	Webview绕过证书校验漏洞	低	安全
身份认证安全（3项）			
45	界面劫持风险	高	存在风险(发现1处)
46	输入监听风险	高	安全
47	截屏攻击风险	中	安全
内部数据交互安全（10项）			
48	动态注册Receiver风险	高	安全
49	Content Provider数据泄露漏洞	高	安全
50	Activity组件导出风险	中	存在风险(发现9处)
51	Service组件导出风险	中	安全
52	Broadcast Receiver组件导出风险	中	存在风险(发现4处)
53	Content Provider组件导出风险	中	安全
54	PendingIntent错误使用Intent风险	中	安全
55	Intent组件隐式调用风险	低	安全
56	Intent Scheme URL攻击漏洞	低	安全
57	Fragment注入攻击漏洞	低	安全
恶意攻击防范能力（13项）			
58	动态注入攻击风险	高	存在风险(发现1处)
59	Webview远程代码执行漏洞	高	安全
60	未移除有风险的Webview系统隐藏接口漏洞	中	安全
		中	安全
		中	安全
		中	存在漏洞(发现7处)
		低	安全
		低	安全
		低	存在风险(发现7处)
67	未使用地址空间随机化技术风险	低	存在风险(发现1处)
68	模拟器运行风险	低	存在风险(发现1处)
69	Root设备运行风险	低	存在风险(发现1处)
70	不安全的浏览器调用漏洞	低	安全

并非所有漏洞都需要修复，开发者可根据实际情况选择

来源：公安三所

应用安全活动周期/安全增强



应用安全活动周期/渠道监控

网易易盾

- 爬虫

监测主流渠道应用下载数据，包括第三方市场、论坛等

- 盗版识别

应用图标、名称、包名、资源文件、代码指纹等，使用相似度算法分析

- 数据分析

盗版渠道分布、下载量、盗版溯源、篡改内容等数据分析

序号	版本	版本个数	正版个数	可疑盗版个数	正版分布渠道	可疑盗版分布渠道
1	6.3.8.0	1	1	0	木蚂蚁	无
2	8.4.7.4	1	0	1		吾爱破解
3	6.2.3.0	1	1	0	木蚂蚁	无
4	6.3.7.0	1	1	0	木蚂蚁	无
5	8.4.6.1	1	0	1		木蚂蚁

应用安全活动周期/总结

网易易盾

安全开发	应用审计	安全增强	渠道监测
数据通信	安全扫描	漏洞修复	盗版监控
数据安全	渗透测试	应用加固	数据分析
核心逻辑保护			
调试日志			

大纲

1. 应用安全风险案例 & 安全隐患总结
2. 应用安全活动周期
   - 安全开发
   - 应用审计
   - 安全增强
   - 渠道监控
3. **安卓加固常见策略**
   - 应用加固
   - 源码混淆

安卓加固常见策略

网易易盾

（一）应用加固

- DEX文件
- SDK文件
- SO文件

（二）源码混淆

- JAVA
- C/C++
- Javascript

安卓加固常见策略/应用加固

网易易盾

- DEX加固

- 整体DEX加固（加壳）
- 拆分DEX加固（函数抽取及动态恢复，增加内存Dump的难度）
- 虚拟机加固 (自定义指令虚拟机解释器，指令可混淆)
- Java2C加固 （将DEX指令转换为native指令执行，可增加混淆、字符串加密功能）

安卓加固常见策略/应用加固

- DEX加固/java2c

```
17  @   private void encrypt(String str){
18          if (str == null || str.length() == 0) {
19              throw new IllegalArgumentException("String to encript
20          }
21
22          StringBuffer hexString = new StringBuffer();
23          try {
24              MessageDigest md = MessageDiges
25              md.update(str.getBytes());
26              byte[] hash = md.digest();
27              for (int i = 0; i < hash.lengt
28                  if ((0xff & hash[i]) < 0x1(
29                      hexString.append("0" +
30                  } else {
31                      hexString.append(Integ
32                  }
33              }
34          } catch (NoSuchAlgorithmException e) {
35              e.printStackTrace();
36          }
37      }
```

加固前

```
package com.fireboy.myappli

import android.os.Bundle;
import android.support.v7.a

public class MainActivity e
    public MainActivity() {
        super();
    }

    private native void enc
    }

    protected native void o
    }
}
```

加固后

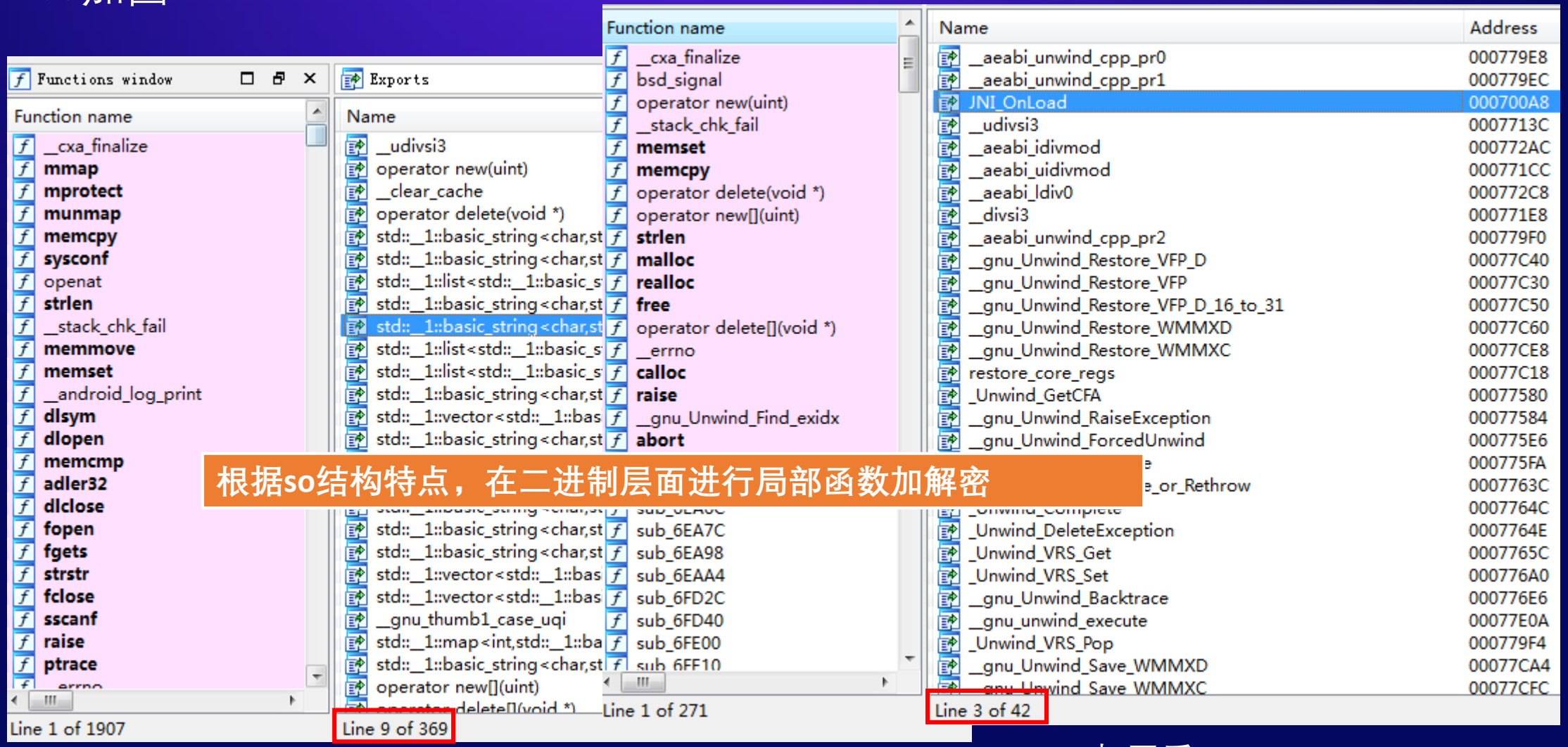
IDA View-A | Pseudocode-A | Hex View-1 | Structures

```
271   v15 = j_dvmRemoveFromReferenceTable(v211, v202, 1, v14);
272   *v205 = v191;
273   v79 = (v196 - 2623);
274   v187 = v15;
275   if ( j_catchTargets(v211, &v79, &v79, v188) != 202 )
276   {
277     v79 = v190;
278     v186 = "getInstance";
279     v185 = &GLOBAL_OFFSET_TABLE_;
280     v184 = j_invokeStaticObjectMethod(
281              v211,
282              "java/security/MessageDigest",
283              "getInstance",
284              "(Ljava/lang/String;)Ljava/security/MessageDigest;");
285     v183 = v184;
286     v16 = j_dvmAddToReferenceTable(v211, v202, v184);
287     v17 = *v205;
288     v79 = v184;
289     v182 = v16;
290     v181 = 1;
291     v18 = j_dvmRemoveFromReferenceTable(v211, v202, 1, v17);
292     *v205 = v184;
293     v79 = (v185 - 2623);
294     v180 = v18;
295     if ( j_catchTargets(v211, &v79, &v79, v181) != 202 )
296     {
297       v79 = "()[B";
298       v179 = &GLOBAL_OFFSET_TABLE_;
299       v178 = j_invokeObjectMethod(v211, v209, "java/lang/String", "getBytes");
300       v177 = v178;
301       v19 = j_dvmAddToReferenceTable(v211, v202, v178);
302       v20 = *v207;
303       v79 = v178;
304       v176 = v19;
305       v21 = j_dvmRemoveFromReferenceTable(v211, v202, 2, v20);
306       *v207 = v178;
307       v79 = (v179 - 2623);
308       v175 = v21;
309       if ( j_catchTargets(v211, &v79, &v79, 1) != 202 )
310       {
311         v80 = v177;
312         v79 = "([B)V";
313         v174 = "update";
314         v173 = &GLOBAL_OFFSET_TABLE_;
315         j_invokeVoidMethod(v211, v183, "java/security/MessageDigest", "update");
316         v79 = v173 + 0xFFFFD704;
317         v172 = &v79;
318         v171 = &v79;
319         if ( j_catchTargets(v211, &v79, &v79, 1) != 202 )
320         {
```

安卓加固常见策略/应用加固

网易易盾

- so加固



加固前

加固后

安卓加固常见策略/源码混淆

网易易盾

- Java代码混淆

```
public final class com.bumptech.glide.a.a$a
{
    private final b b;
    private final boolean[] c;
    private boolean d;

    com.bumptech.glide.a.a$a(a arg1, b arg2, com.
        this(arg1, arg2);
    }

    private com.bumptech.glide.a.a$a(a arg2, b ar
        this.a = arg2;
        super();
        this.b = arg3;
        boolean[] v0 = b.d(arg3) ? null : new boo
        this.c = v0;
    }

    static b a(com.bumptech.glide.a.a$a arg1) {
        return arg1.b;
```

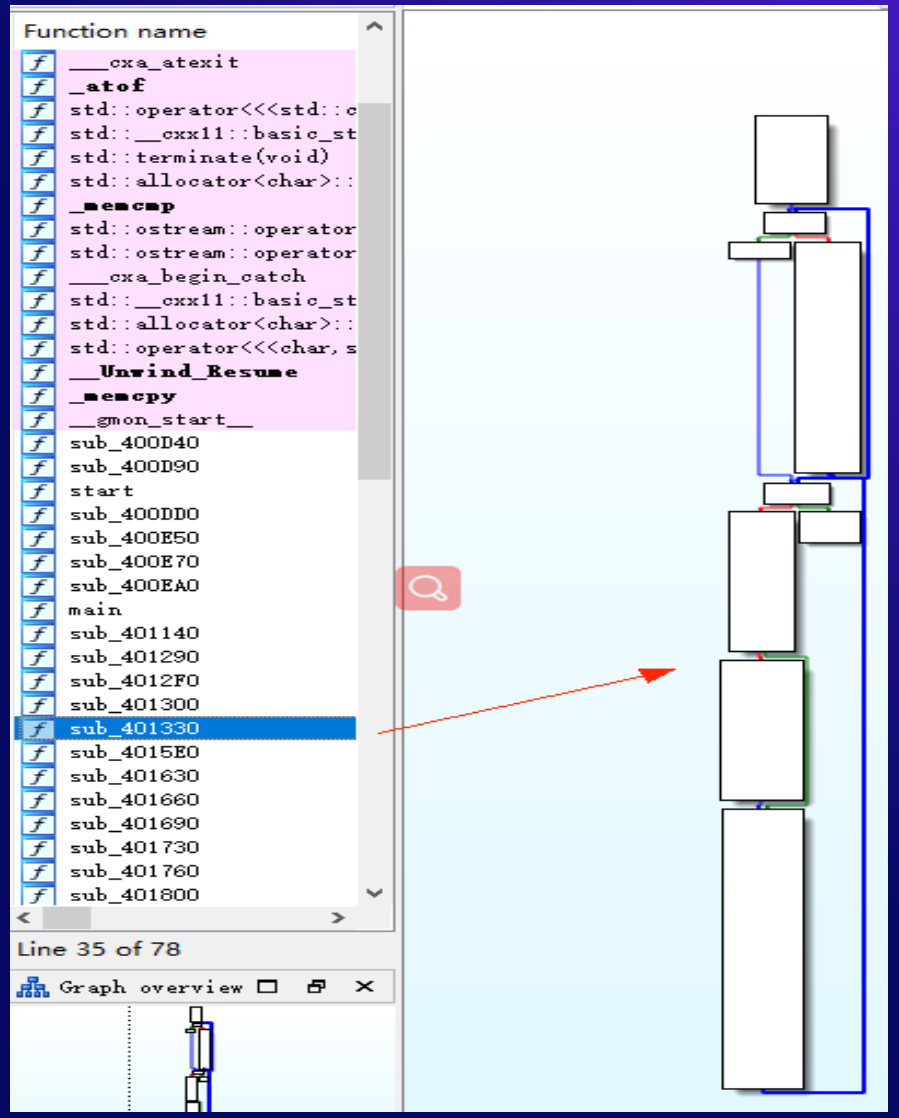
混淆前

```
public final class d
{
  private DocumentContents 士;
  private long 始 = -1L;
  private int 式 = -1;
  private String 示;
  private boolean 蔫 = false;
  private int 慕 = -1;
  private int 藟 = 0;
  private DocumentId 驶;
```

混淆后

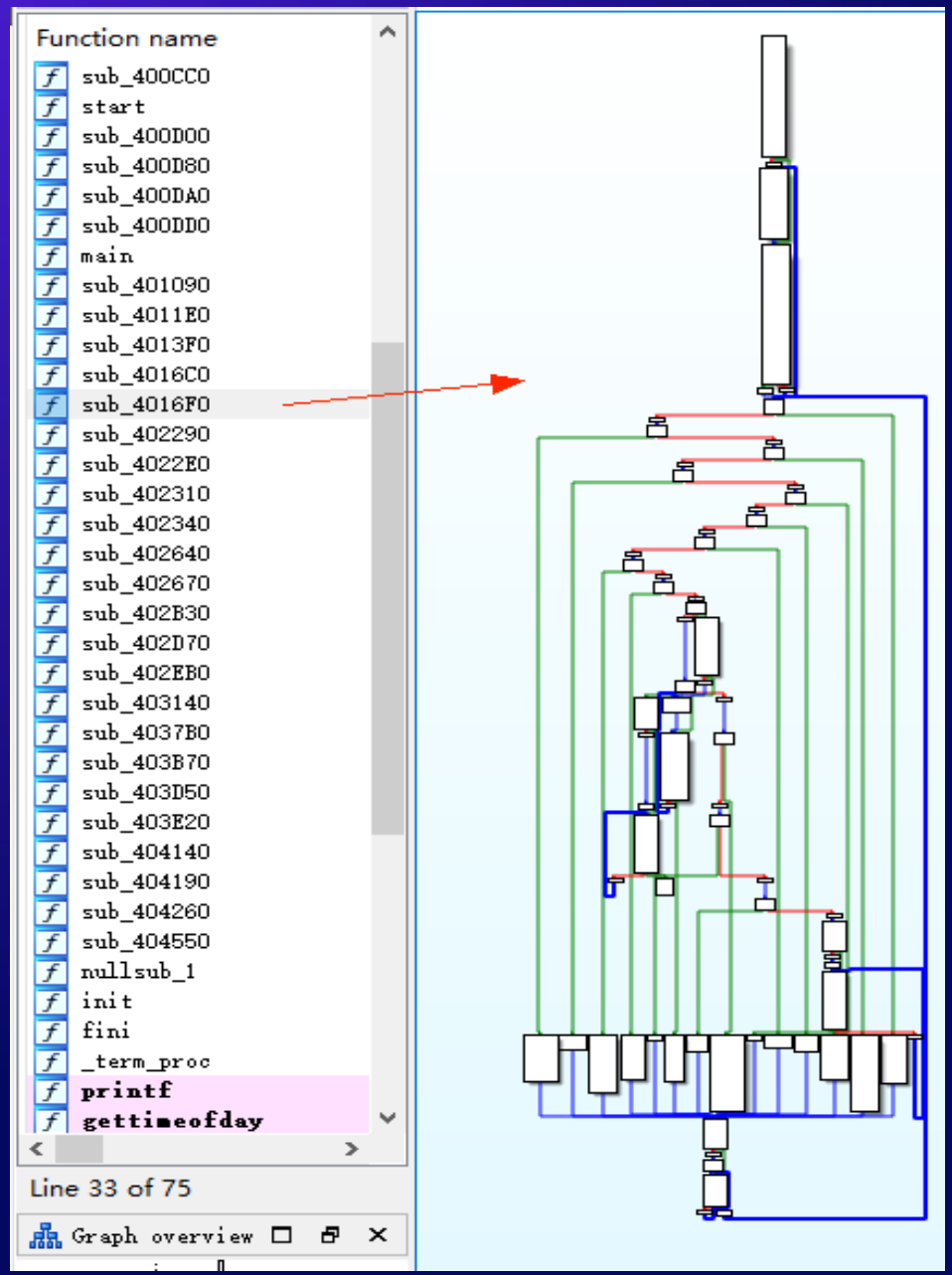
安卓加固常见策略/源码混淆

- c/c++代码混淆



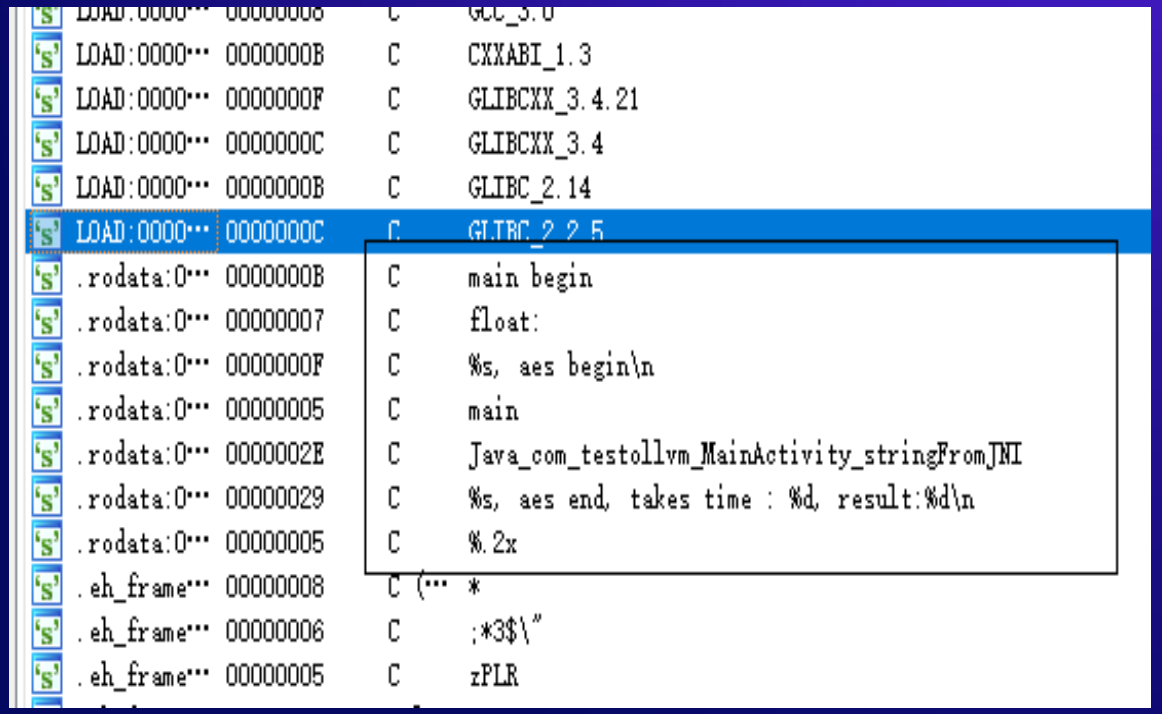
保护之前的
代码控制流

保护后被逻辑混
淆的代码控制流

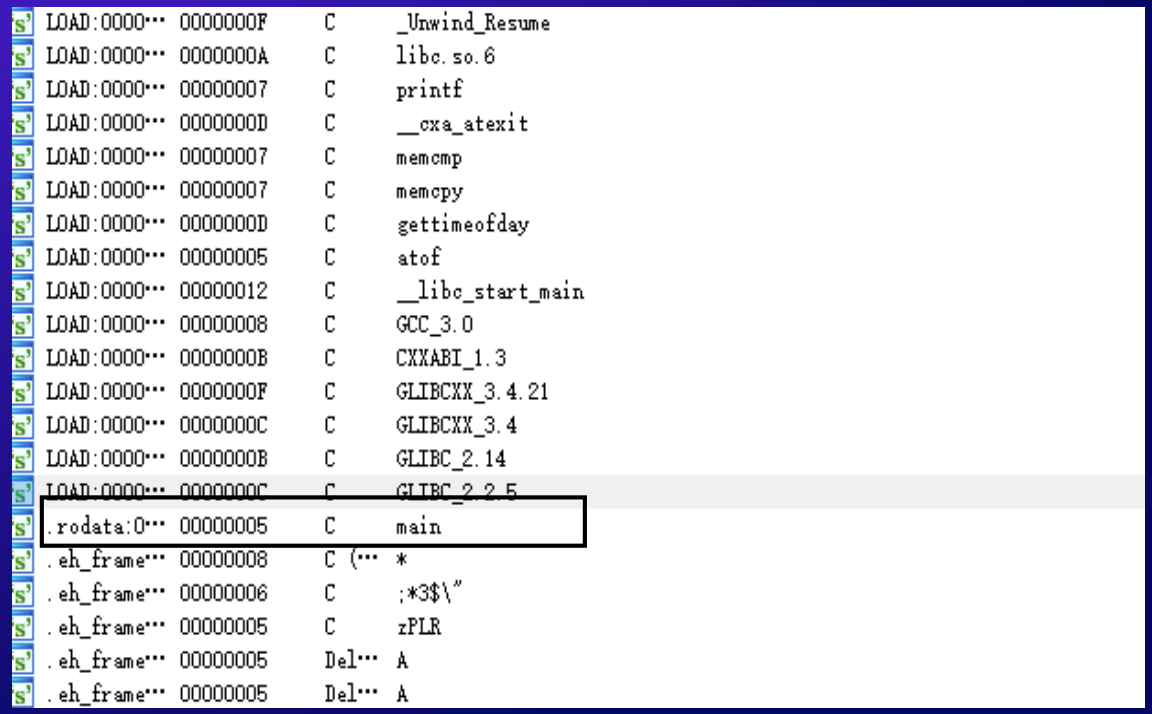


安卓加固常见策略/源码混淆

- c/c++代码混淆



保护前 字符串明文



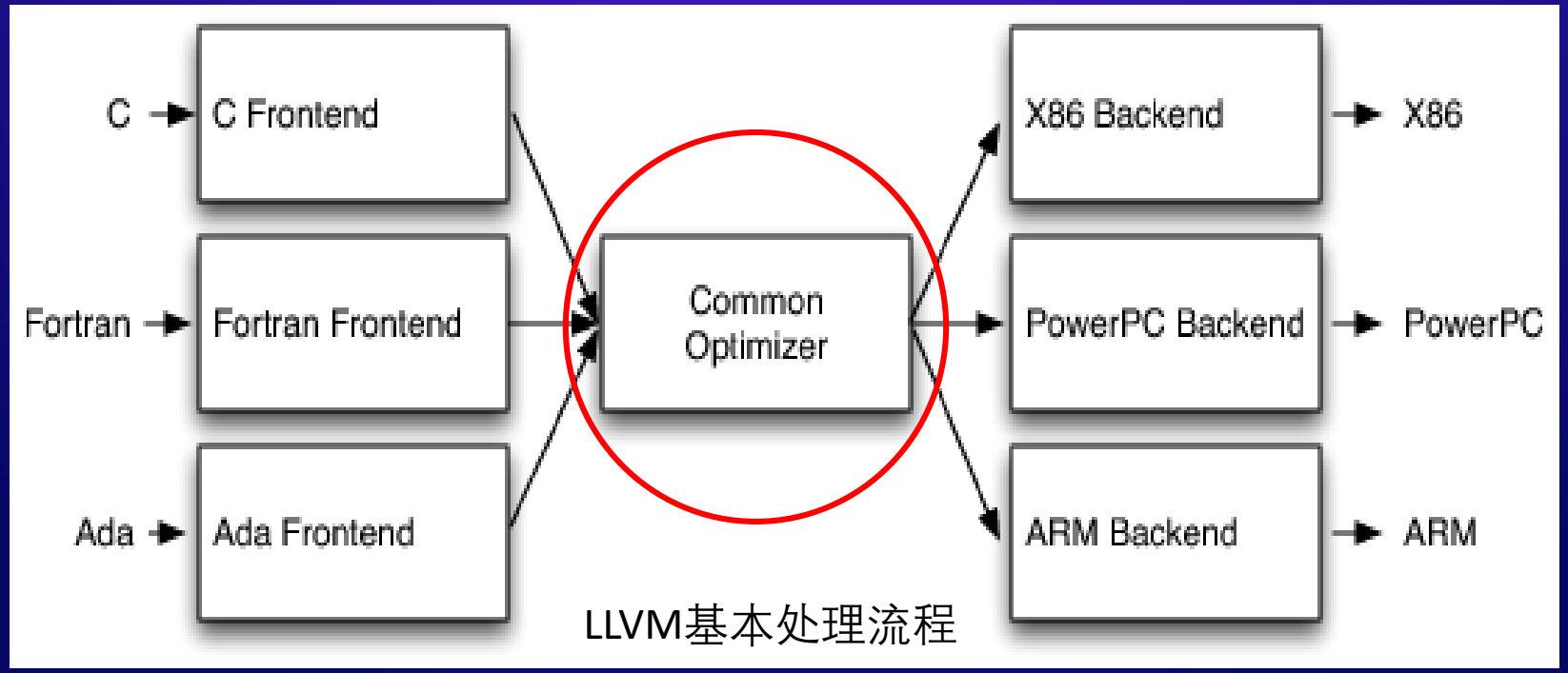
保护后 字符串隐藏

安卓加固常见策略/源码混淆

网易易盾

- c/c++代码混淆

LLVM : 逆向对抗利器



LLVM基本处理流程

此处不仅仅是混淆，通过多重Optimize(优化器)，可以实现多种效果，例如代码控制流扁平化，虚假控制流，字符串加密，符号混淆，指令替换等

图片来源：LLVM官网

安卓加固常见策略/源码混淆

- h5 脚本混淆

```
if ($.mobile_download.isMobile()) {
    var language_arr = ['de','es','br','fr','it','nl'];
    var path_name = window.location.pathname;
    var language = path_name.substr(1,2);
    var link_url = '';
    $('a[href^="http://                    "],a[href^="http:/                    ]').each(function(){
        var download_url = $(this).attr('href');
        if(typeof(download_url) != 'undefined' ){
            if(download_url.indexOf('.dmg') > -1 || download_url.indexOf('.exe') > -1){
                if($.mobile_download.in_Array(language,language_arr)){
                    $(this).attr('href', 'https:/                    e.com/'+language+'/get-downloac
                }else{
                    language = 'en';
                    $(this).attr('hr
                }
                $.mobile_download.cli
            }
        }
    });
}else{
    var filetypes = /\.(zip|exe|dmg)$
```

保护前

保护后

总结

网易易盾

逆向对抗要点：

- 核心逻辑置于native层
- Native层代码保护，借助LLVM
- 直接使用第三方安全加固产品（可选）

为您的App保驾护航

http://dun.163.com

欢迎关注安卓巴士公众号

www.apkbus.com