Wonderland 设计

小组成员：
杨舒晗 20307130386
王楠欣 20307130377
李丹琦 20307130353
李昊霖 20307130260

Display

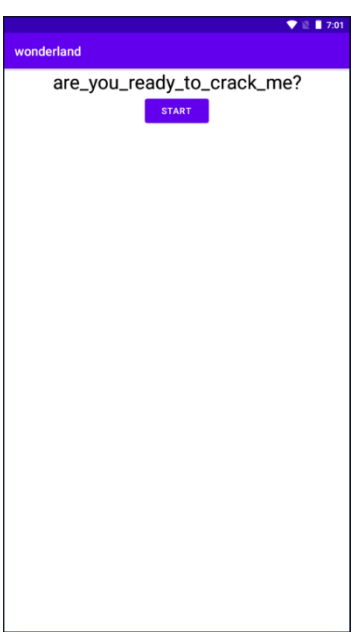
名称：wonderland

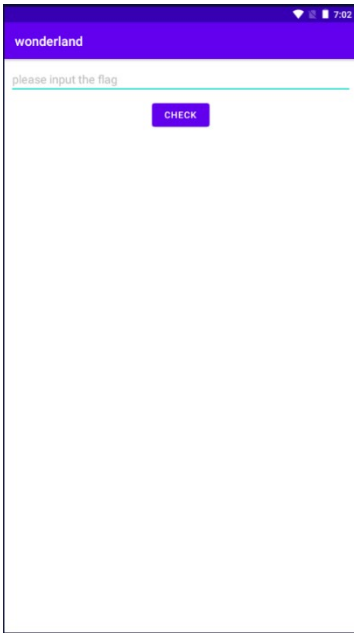
简介：欢迎来到摸仙堡的 wonderland，只要你能找到藏在一堆 button 里的四个真正的 flag pieces 并按照 button 的顺序正确拼接就能 crack me。

Flag：
flag{pore_2022_group_project:after_all_tomorrow_is_another_day!**!!!!######___!!!******___#####_##!**_*!reverse_it}
或
flag{pore_2022_group_project:after_all_tomorrow_is_another_day!**!!!!######___!!!******___####_###!**_*!reverse_it}
【注】匹配任一 flag 即可

基本设计思路

flag 总共由四部分组成，每个部分都由一个独立的 flag 呈现，由一个单独的页面检查。





需要在 apk 中获得四个正确的部分然后按照对应按钮的顺序拼接即可 crackme。

Apk 由首页，点击首页按钮后进入的七个 button 的页面，以及每个 button 对应的 flag 检查页面组成。apk 首页点击“start”按钮后，onTouch 会一直占用点击状态，使程序不能执行到 onClick，阻碍 attacker 进入之后的页面。若通过修改 smali 顺利进入通过首页后，可以看到七个按钮，每个按钮都对应一个 flag 的检查页面，其中只有四个对应真正的 flag pieces。

七个 button 的设计思路如下：

Button 1：flag{pore_2022_group_project:

首页的页面跳转运用了 onTouch 函数的 return true 占用状态导致不能触发 onclick；Button1 的页面跳转用到当前时间 service 和 killprocess 程序；两者都需要修改 smali 代码重打包 apk，如此 attack 方才能进入 Button1 的 flag check 部分。flag check 运用自定义函数的解密拿到 DES 加密的密钥，同时提供 DES 密文和 iv，写出 DES 解密代码解密得到这部分的 flag。

Button 3：flag{after_all_tomorrow_is_another_day}

将输入的字符串按照对角线的方式放在二维数组中，'_'对应一个空格，然后按照行顺序读取新的字符串。若输入的字符串经过变换后与题目本身给出的数组一致，则成功。比如输入“get_the_flag_by_yourself”，经过下表中的变换后将得到“g el ue et abyrl thfgyosf”。然后对算法进行控制流混淆。

g		e	l			u	e		
	e	t		a	b	y	r	l	
		t	h	f	g	y	o	s	f

Button 4：flag{!!!!!######___!!!******___#####_##!**_*!} 或**
flag{!!!!!######___!!!******___####_###!**_*!}**

检查 flag 的核心逻辑在 native code 中实现，用 C 实现推箱子游戏，能将箱子推到目标上的最短字符串即为 flag pieces。通过限制用户输入长度使 flag 只有两种可能。需要逆向 native code 猜测出这个游戏并获取游戏的地图及控制方向的字符，寻找最短的路径来获得 flag pieces。

Button 7：flag{reverse_it}

运用 C++实现简化版的 AES 加密过程（去掉了盒变换和列混淆两个环节）。

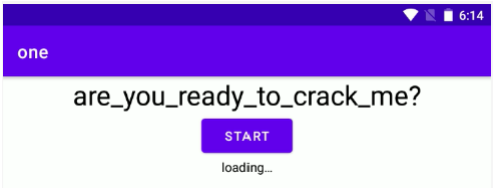
Button 2：Button 5：Button 6：flag{fakeflag}

在 native code 中检查 flag 实现，将输入字符串中的字符与字符下标异或，将异或后的字符串与给定字符串比较，若相等会给出“YOU ARE FOOLED!”的提示，表明是虚假 flag。

如何逆向 wonderland

一、通过首页的 start 进入 wonderland 找 flag 的界面

(1)安装 apk 后，打开 app，点击 start 按钮，会发现界面被卡住不跳转，一直显示“loading…”。



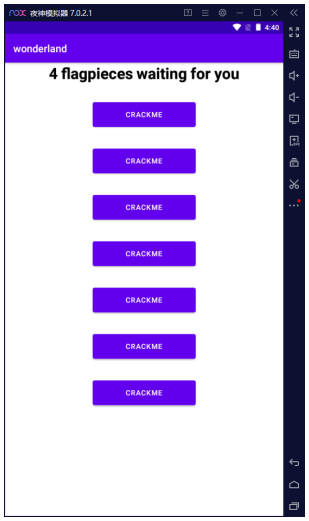
（2）通过 monitor.bat 查按钮 id，找到 start 对应点击/触摸事件的函数。jeb 反汇编 apk，找到这部分代码在 ccc 类中，分析得知，因为 onTouch 中有 return true 语句，所以 onTouch 会一直占用点击状态，使程序不能执行到 onClick 的部分。此处修改 smali 代码将 return true 删掉或将 onTouch 代码段删掉，就能顺利进入找 flag 的界面了。

```
public class ccc extends AppCompatActivity {
    @Override  // androidx.fragment.app.FragmentActivity
    protected void onCreate(Bundle arg4) {
        super.onCreate(arg4);
        this.setContentView(0x7F0B002D);  // layout:hhh
        ((TextView)this.findViewById(0x7F0801A0)).setText(this.getResources().getString(0x7F0E0034));
        Button start = (Button)this.findViewById(0x7F0801A1);  // id:ui_2
        start.setOnClickListener(new View.OnClickListener() {
            @Override  // android.view.View$OnClickListener
            public void onClick(View arg6) {
                ((TextView)ccc.this.findViewById(0x7F0801A2)).setTextColor(0xFF000000);  // id:ui_3
                Thread.currentThread();
                try {
                    Thread.sleep(5000L);
                }
                catch(InterruptedException e) {
                    e.printStackTrace();
                }

                Intent v2 = new Intent(ccc.this, fff.class);
                ccc.this.startActivity(v2);
            }
        });
        start.setOnLongClickListener(new View.OnLongClickListener() {
            @Override  // android.view.View$OnLongClickListener
            public boolean onLongClick(View arg3) {
                ((TextView)ccc.this.findViewById(0x7F0801A2)).setTextColor(0xFF000000);  // id:ui_3
                return true;
            }
        });
        start.setOnTouchListener(new View.OnTouchListener() {
            @Override  // android.view.View$OnTouchListener
            public boolean onTouch(View arg3, MotionEvent arg4) {
                ((TextView)ccc.this.findViewById(0x7F0801A2)).setTextColor(0xFF000000);  // id:ui_3
                return true;
            }
        });
    }
}
```

二、在 Wonderland 页面中寻找七个 button 后的 flag

每个“CRACKME”按钮对应一个 flag 检查页面，其中有四个 real flag，三个 fake flag



三、进入按钮后的 check 逻辑（从上到下为 button1-7）

1. Button 1：

需要修改 smali 并进行 DES 解密

（1）点击 button1 时，会弹出“wrong time！”消息，在 jeb 中查看 button1 对应的代码。

（2）aaa 中将当前时间和一个以前的时间比较，当不等的时候弹出“wrong time！”，但根据代码，不能只考虑这一点，因为出了 if 部分，后续还有 killprocess 的代码。这里同样需要修改 smali 代码，如删除 if 和 killprocess 的代码，进入的 checkflag 界面。

```
public class aaa extends Service {
    @Override  // android.app.Service
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override  // android.app.Service
    public void onCreate() {
        super.onCreate();
    }

    @Override  // android.app.Service
    public void onDestroy() {
        super.onDestroy();
    }

    @Override  // android.app.Service
    public int onStartCommand(Intent intent, int flags, int startId) {
        Date m = new Date(System.currentTimeMillis());
        String str = new SimpleDateFormat(this.getResources().getString(0x7F0E002C)).format(m);  //
        if(!str.equals(new String(new Random().nextInt(2022) + str.substring(4, str.length())))) {
            Toast.makeText(this, this.getResources().getString(0x7F0E002E), 1).show();  // string:hh
            return super.onStartCommand(intent, flags, startId);
        }

        Process.killProcess(Process.myPid());
        this.startActivity(new Intent(this, eee.class));
        return super.onStartCommand(intent, flags, startId);
    }
}
```

（3）分析 checkflag 的代码，这里利用 bbb.door 方法对输入内容进行 DES 加密，并与给定字符串 " aPxWIYSMYLOOsgI1QEyZ+dIMD1RxV+SBKJqY6lFunLI="比较，当相等时验证消息为“Right！”，其中 DES 加密的密钥”moon&sun”通过一个给出的 bbb.hhh 解密函数解密字符串"umpqq*x{"得到。

eee 代码：

```
public class eee extends AppCompatActivity {
    private EditText et;

    @Override  // androidx.fragment.app.FragmentActivity
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setContentView(0x7F0B001C);  // layout:adbaaa
        Button btn = (Button)this.findViewById(0x7F080054);  // id:bb1
        this.et = (EditText)this.findViewById(0x7F080051);  // id:b1
        btn.setOnClickListener(new View.OnClickListener() {
            @Override  // android.view.View$OnClickListener
            public void onClick(View view) {
                String in = eee.this.et.getText().toString();
                try {
                    if(bbb.door(eee.this.getResources().getString(0x7F0E0076), eee.this.getResources().getString(0x7F0
                        Toast.makeText(eee.this, "Right!", 1).show();
                        return;
                    }

                    Toast.makeText(eee.this, "Wrong!", 1).show();
                }
                catch(Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
}
```

If 条件关键部分：

```
1B), in, bbb.hhh(eee.this.getResources().getString(0x7F0E0032))).equals(eee.this.getResources().getString(0x7F0E006C))) {  // string:type "DES"
```

bbb.hhh 方法（解密字符串得到 DES 算法密钥）：

```
public static String en(String str) {
    char[] back = new char[str.length()];
    int i;
    for(i = 0; i < str.length() - 1; ++i) {
        back[i] = (char)(str.charAt(i + 1) - i);
    }

    back[str.length() - 1] = (char)(str.charAt(0) - (str.length() - 1));
    return new String(back);
}

public static String hhh(String origin) {
    return bbb.en(origin);
}
```

bbb.door 方法（DES 加密）：

```
public static String door(String e1, String e2, String e3, String e4) throws Exception {
    IvParameterSpec zeroIv = new IvParameterSpec(bbb.iv);
    Cipher cipher = Cipher.getInstance(e2);
    cipher.init(1, new SecretKeySpec(e4.getBytes(), e1), zeroIv);
    return ddd.encode(cipher.doFinal(e3.getBytes()));
}
```

为了得到正确的输入，攻击方需要将解密函数 copy 出来，运行拿到密钥，同时写出 DES 的解密函数，对给定字符串解密，得到明文“flag{pore_2022_group_project:”，在 check 界面输入该明文，弹出“Right！”消息，找到 button1 的 flag。

DES 解密代码如下：

```
public static String decryptDES(String decryptString, String decryptKey)
        throws Exception {
    byte[] byteMi = new BASE64().decode(decryptString);
    IvParameterSpec zeroIv = new IvParameterSpec(iv);
    SecretKeySpec key = new SecretKeySpec(decryptKey.getBytes(), "DES");
    Cipher cipher = Cipher.getInstance("DES/CBC/PKCS5Padding");
    cipher.init(Cipher.DECRYPT_MODE, key, zeroIv);
    byte decryptedData[] = cipher.doFinal(byteMi);

    return new String(decryptedData);
}

```


```
public static String decryptDES(String decryptString, String decryptKey)
        throws Exception {
    byte[] byteMi = new BASE64().decode(decryptString);
    IvParameterSpec zeroIv = new IvParameterSpec(iv);
    SecretKeySpec key = new SecretKeySpec(decryptKey.getBytes(), "DES");
    Cipher cipher = Cipher.getInstance("DES/CBC/PKCS5Padding");
    cipher.init(Cipher.DECRYPT_MODE, key, zeroIv);
    byte decryptedData[] = cipher.doFinal(byteMi);

    return new String(decryptedData);
}
```


2、Button3

先从变量 next 和 j 入手，破解 switch 中各个基本块之间的执行顺序，将扁平化的控制流复原。然后看出 x，y 及一些不透明谓词的混淆作用，将函数化简。最后可以自己编写代码，测试一些字符串找到规律，也可以直接看代码想象出它的加密方式。

3、Button4

1、使用 monitor 定位页面的 check 按钮点击事件所在的类为 ggg，然后使用 jeb 逆向 apk，发现检查 flag 的逻辑在 native code 中，并且 flag 格式为 flag{}，长度不超过 49

2、使用 ida 分析 native code，找到 ggg 类使用的 native 函数 Check，以此为入口分析，猜测出这是一个推箱子游戏

1）需要在 djiehiu 函数中看出三个被折叠的常量，地图的长，宽以及箱子的数目，分别为 ida 反编译得到的 v10, v9, v8，式中重复出现的(v7 % 11 - (v7 % 10 - v7 / 10) % 11 + 1) % 11 等于 1，(v7 % 11 - (v7 % 10 - v7 / 10) % 11) % 11 等于 0，故可知地图的长为 11，宽为 9，箱子的数目为 2

```
v1 = random();
v7 = sub_9CA0(v1) % 100;
v10 = (v7 % 11 - (v7 % 10 - v7 / 10) % 11 + 1) % 11 + 10;
v9 = (v7 % 11 - (v7 % 10 - v7 / 10) % 11) % 11 + 9;
v8 = (v7 % 11 - (v7 % 10 - v7 / 10) % 11 + 1) % 11 + 1;
v6 = strlen(a1);
Jogoheuibwnr(v13, v14, v10, v9, v8, &v16);
```

2）在 Jogoheuibwnr 函数中获取地图数字矩阵。函数中初始化了一个很大的数组，查看地址&unk_2D744 中存放的数据，根据 dest 中元素类型为_DWORD 为无符号 32 位数，可以推测对应整型数据，因此从地址&unk_2D744 开始，每四个字节对应一个数组中一个元素。

```
_DWORD dest[65029]; // [esp+44h] [ebp-3F814h] BYREF

dest[65025] = __readgsdword(0x14u);
memcpy(dest, &unk_2D744, (size_t)&unk_3F804);
```

```
unk_2D744       db    0
                db    0
                db    0
                db    0
                db    1
                db    0
                db    0
                db    0
                db    1
                db    0
                db    0
                db    0
                db    1
                db    0
                db    0
                db    0
                db    1
                db    0
                db    0
                db    0
                db    1
                db    0
                db    0
                db    0
```

函数中通过 for 循环，根据 dest 数组给二维数组 a1 赋值，并且当 dest 数组元素为 3 或 6 时，将元素坐标存放到二维数组 a2 中；当 dest 数组元素为 5 或 7 时，将元素坐标存放到数组 a7 中。可以猜测 a2 和 a7 记录了游戏中会移动的小人和箱子，而 a2 中存放了多个位移向量，可能对应箱子的坐标。

循环结束条件使用了 djiehiu 函数中获得的常量 v10 和 v9，因此，需要跳过 dest 中的无效数据，然后，每四个字节确定 a1 中的一个有效数据。

通过分析 iojugyub，jwehuibipjohio 两个更新数组 a7 和 a2 中的位移向量的函数，以及 jwhbiriwrnwjoqj 在每次移动后更新地图数组 a1 的函数可以判断地图数组 a1 中各个数字的含义，0-7 分别代表墙外的空地，墙壁，墙内的空地，箱子，目的地，小人，有箱子在的目的地，有小人在的目的地，然后可以根据初始化的 a1 数组绘制出游戏地图

3）确定方向控制的字符

在 djiehiu 函数中的 switch 语句根据输入字符确定 v11，而后根据 v11 在二维数组 s 中确定位移向量，其中给出了八个 case，每个 case 都给 v11 赋值，需要还原常量来判断每个字符对应的位移向量。可以发现‘! ’，‘#’，‘*’，‘_’分别给 v11 赋值 0,3,2,1，对应位移（-1,0），（1,0），（0，-1），（0,1），可以猜测对应向上，向右，向左，向下而其余 case 都给 v11 赋值 0，对应位移（0,0）

```
case '!':
  v11 = (v12 + 3) * (v12 + 2) * (v12 + 1) * v12 % 2;
  break;
case '#':
  v11 = (v12 + 5) % 2 + (v12 + 4) % 2 + (v12 + 3) % 2 + (v12 + 2) % 2 + (v12 + 1) % 2 + v12 % 2;
  break;
```

```
case '*':
  v11 = (v12 + 1) * (v12 + 1) * v12 * v12 % 4 + 2;
  break;
```

```
case '_':
  v4 = v12 % 100;
  v11 = (v12 % 100 % 11 - (v12 % 100 % 10 - v12 % 100 / 10) % 11 + 1) % 11;
  break;
```

```
memset(s, 0, sizeof(s));
s[0] = -1;
s[2] = 1;
s[5] = -1;
s[7] = 1;
```

5) 正确得到地图及控制方向的字符后可以进行推箱子游戏并记录小人移动的字符，由 Check 函数中对 flag 长度的限制可以推断要寻找最短的移动路径字符串，Check 函数最后返回 jwhbiriwrnwjoqj 函数的返回值，该函数更新地图的同时记录推入目的地的箱子数目，当计数等于箱子总数时返回 1，即 Check 返回 1。因此，能使所有箱子推入目的地的最短的字符串即为 flag。

4、Button7

逆向 native code 得到种子密钥、加密函数和密文。第一步需要利用密钥扩展函数得到扩展后的密钥（函数可以在 native code 中找到），第二步需要根据加密函数的逻辑写出对应的解密函数，再根据密文和扩展密钥来还原密码明文。

具体思路如下：

Jeb 反编译定位 button 点击操作，关联到 xxx class：

```
g7.setOnClickListener(new View.OnClickListener() {
    @Override  // android.view.View$OnClickListener
    public void onClick(View view) {
        fff.this.startActivity(new Intent(fff.this, xxx.class));
    }
});
```

阅读代码，定位检查函数 checkstr

```
if(xxx.this.Checkstr(xxx.this.mEditText.getText().toString())) {
    context = xxx.this.getApplicationContext();
    v0 = "RIGHT!";
}
```

```
static {
    System.loadLibrary("one");
}

public native boolean Checkstr(String arg1) {
}
```

发现为 native 引用，使用 IDA 打开 apk 解压后的.so 文件，搜索找到 checkstr 函数：

```
1int __cdecl checkstr(int a1)
2{
3  char *v1; // eax
4  char *v3; // [esp+18h] [ebp-130h]
5  char *v4; // [esp+24h] [ebp-124h]
6  int j; // [esp+34h] [ebp-114h]
7  int i; // [esp+38h] [ebp-110h]
8  char v7; // [esp+3Fh] [ebp-109h]
9  int v8; // [esp+40h] [ebp-108h] BYREF
10  int v9[16]; // [esp+48h] [ebp-100h] BYREF
11  char v10[176]; // [esp+88h] [ebp-C0h] BYREF
12  _DWORD v11[4]; // [esp+138h] [ebp-10h] BYREF
13
14  v11[1] = __readgsdword(0x14u);
15  if ( sub_B650(a1) == 16 )
16  {
17    v4 = v10;
18    do
19    {
20      sub_A760(v4);
21      v4 += 4;
22    }
23    while ( v4 != (char *)v11 );
24    KeyExpansion(&Key, v10);
25    v3 = (char *)v9;
26    do
27    {
28      sub_B270(v3);
29      v3 += 4;
30    }
31    while ( v3 != v10 );
32    for ( i = 0; i != 16; ++i )
33    {
34      v1 = (char *)sub_B680(a1, i);
35      sub_B0C0(&v8, *v1, (unsigned __int64)*v1 >> 32);
```

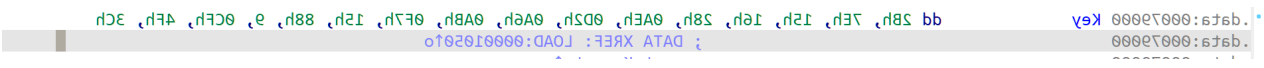
首先拷贝出 keyexpansion 函数，根据运行逻辑来作出还原，得到密钥扩展部分

```
unsigned int __cdecl KeyExpansion(int a1, int a2)
{
  int i; // [esp+2Ch] [ebp-4Ch]
  int j; // [esp+2Ch] [ebp-4Ch]
  int v5; // [esp+30h] [ebp-48h] BYREF
  int v6[10]; // [esp+38h] [ebp-40h] BYREF
  int v7; // [esp+60h] [ebp-18h] BYREF
  int v8[4]; // [esp+68h] [ebp-10h] BYREF

  v8[1] = __readgsdword(0x14u);
  sub_A760(v8);
  for ( i = 0; i < 4; ++i )
  {
    v6[8] = *(_DWORD *)(a1 + 16 * i);
    v6[6] = *(_DWORD *)(a1 + 16 * i + 4);
    v6[4] = *(_DWORD *)(a1 + 16 * i + 8);
    v6[2] = *(_DWORD *)(a1 + 16 * i + 12);
    Word(&v7);
    *(_DWORD *)(a2 + 4 * i) = v7;
  }
  for ( j = 4; j < 44; ++j )
  {
    v8[0] = *(_DWORD *)(a2 + 4 * (j - 1));
    if ( j % 4 )
    {
      sub_AB50(&v5);
      *(_DWORD *)(a2 + 4 * j) = v5;
    }
    else
    {
      sub_AB50(v6);
```

```
void KeyExpansion(byte key[4 * Nk], word w[4 * (Nr + 1)]) {
    word temp;
    int i = 0;
    while (i < Nk) {
        w[i] = Word(key[4 * i], key[4 * i + 1], key[4 * i + 2], key[4 * i + 3]);
        i++;
    }
    i = Nk;
    while (i < 4 * (Nr + 1)) {
        temp = w[i - 1];
        if (i % Nk == 0)
            w[i] = w[i - Nk] ^ rcon[i / Nk - 1];
        else
            w[i] = w[i - Nk] ^ temp;
        i++;
    }
}
```

输入反编译得到的种子密钥，得扩展密钥



```cpp
int main() {
    word w[4 * (Nr + 1)];
    KeyExpansion(key, w);

    for (int i = 0; i != 4 * (Nr + 1); i++) {
        cout <<hex<< w[i].to_ulong() << endl;
    }
    return 0;
}
```

```
Microsoft
23619d9a
2c7e1516
6d0c7b0
8559c72e
a6385ab4
247e1516
22aed2a6
a7f71588
1cf4f3c
347e1516
16d0c7b0
b127d238
b0e89d04
147e1516
2aed2a6
b389009e
3619d9a
547e1516
56d0c7b0
e559c72e
e6385ab4
d47e1516
82aed2a6
67f71588
81cf4f3c
cf7e1516
4dd0c7b0
2a27d238
未找到相关问题
```

再根据 encrypt 函数，逆向构造 decrypt 解密函数

```cpp
unsigned int __cdecl encrypt(int a1, int a2)
{
  int *v3; // [esp+10h] [ebp-38h]
  int k; // [esp+1Ch] [ebp-2Ch]
  int j; // [esp+20h] [ebp-28h]
  int i; // [esp+24h] [ebp-24h]
  int v7[4]; // [esp+28h] [ebp-20h] BYREF
  unsigned int v8; // [esp+38h] [ebp-10h] BYREF

  v8 = __readgsdword(0x14u);
  v3 = v7;
  do
    sub_A760(v3++);
  while ( v3 != (int *)&v8 );
  for ( i = 0; i < 4; ++i )
    v7[i] = *(_DWORD *)(a2 + 4 * i);
  RKey_Add(a1, v7);
  for ( j = 1; j <= 10; ++j )
  {
    ShiftRow(a1);
    for ( k = 0; k < 4; ++k )
      v7[k] = *(_DWORD *)(a2 + 4 * (k + 4 * j));
    RKey_Add(a1, v7);
  }
  return __readgsdword(0x14u);
}
```

加密部分主要包括两种操作：位运算异或（rkey_add）和行移动操作（shiftrow），分别根据函数逻辑写出逆运算函数：

```
void InvShiftRow(byte sta_matr[4 * 4]) {
    for (int i = 0; i < 4; i++) {
        byte temp[4];
        for (int j = 0; j < i; j++)
            temp[j] = sta_matr[i * 4 + 3 - j];
        for (int j = 0; j < 4 - i; j++)
            sta_matr[i * 4 + 3 - j] = sta_matr[i * 4 + 3 - j - i];
        for (int m = 0; m < i; m++)
            sta_matr[i * 4 + m] = temp[i - m - 1];
    }
}

void decrypt(byte sta_matr[4 * 4], word w[4 * (Nr + 1)]) {
    word key[4];
    for (int i = 0; i < 4; i++)
        key[i] = w[4 * Nr + i];
    RKey_Add(sta_matr, key);
    for (int r = Nr - 1; r >= 0; r--) {
        InvShiftRow(sta_matr);
        for (int i = 0; i < 4; i++)
            key[i] = w[4 * r + i];
        RKey_Add(sta_matr, key);
    }
}
```

最终通过运行 decrypt 函数，输入 code 密文，即可得到答案明文:

```
.data:00079040 code            dd 0D1h, 71h, 0F6h, 0, 3Ch, 30h, 64h, 0C3h, 0B7h, 0EFh, 73h, 70h, 0EEh, 0F1h, 67h, 6Dh
```

5、Button2，Button5，Button6

利用简单的异或判断，核心逻辑在 native code 中

```
int __cdecl djiehiuyeah(char *a1, int a2)
{
  int i; // [esp+1Ch] [ebp-1Ch]
  int v5[5]; // [esp+24h] [ebp-14h] BYREF

  v5[2] = __readgsdword(0x14u);
  if ( a2 != 14 )
    return 0;
  qmemcpy(v5, "f`ifbig`", 8);
  for ( i = 5; i < 13; ++i )
    a1[i] ^= i;
  return strcmp(a1, (const char *)v5) != 0;
}
```

对抗逆向手段

1、apk 开始页面及 Button1

命名模糊处理、控制流混淆、base64 加密、DES 加密密钥通过自定义加密函数的密文解密得到、DES 加密、防模拟器（程序在模拟器中有 killprocess 操作，需要通过修改 smali 代码绕过）、垃圾代码（fff.class 中多余的 Onclick 事件、aaa.class 中启动的时间服务将当前时间与过去时间比较等等，与 flagpiece 的寻找没有直接关系，去掉这部分代码，仍然能根据剩余的代码找到该 flagpiece）

2、Button3 控制流混淆

1、源程序代码十分简单，接下来主要是对控制流进行混淆：

```
int row = 5;
int cal = bytes1.length/row + 4;
int index;
byte[] bytes2 = new byte[row*cal];

for (int i = 0; i < bytes1.length; i++) {
    if(bytes1[i]=='_') {
        continue;
    }
    index = i % 5 * (cal + 1) + i / 5;
    bytes2[index] = bytes1[i];
}
  System.out.println(Arrays.toString(bytes2));
return Arrays.toString(bytes2).equals(Arrays.toString(bytes));
```

2、首先，增加两个冗余变量 x,y，一方面利用它们增加伪造分支，另一方面通过均值不等式将 i++复杂化，使得 i,x,y 看上去似乎相互联系，增加阅读难度。

```
if(x < y || i < 5){
    x += 1; y += 2;
    i += x/y + y/(4*x); //均值不等式，即i++
}
else{                   //增加伪造分支
    x += 2; y += 1;
    i += y/(4*x);
}
```

3、其次，将求 bytes2 数组下标 index 的算式单独放到另一个函数中，将 x，y 两个无用的变量同时传进函数，将"i % 5 * cal"变换成"(i – i / 5 * 5) * cal"，将"i % 5"用"x % 5"代替，将"i / 5"用"(int)(y / 10)"代替。

```
index = encode(i,cal,x,y);
bytes2[index] = bytes1[i];
```

```
public static int encode(int i, int cal, int x, float y){
    return (i - i / 5 * 5) * cal +  x % 5 + (int)(y / 10);
}
```

4、然后，增加不透明谓词构造一个伪分支

```
if ((bytes1[i] ^ 0x1f) == 0) //always false
    i += 2;
```

5、最后，进行控制流扁平化。定义一个数组 a 和参数 j，next 第一次赋值为 0，进入 case0；next 为 a[1]-a[0]进入 case1，j 为 1；next 为 a[3]-a[2]进入 case5，j 为 4；next 为 a[5]/a[2]进入 case4，j 为 3；next 为 a[4]/a[2]进入 case3，j 为 1；若 bytes1 已经遍历完成，则 next 为 a[1]*a[0]进入 case2，返回，否则 next 为 a[2]/a[1]/2 进入 case1，j 为 1，继续进行 case1->case5->case4->case3 的循环。

```
int[] a = {1, 2, 4, 9, 12, 16};
int j = 0;
int next = 0;
while(true) {
    switch (next) {
        case 0:
            if ((bytes1[i] ^ 0x1f) == 0) //always false
                i += 2;
            next = a[j + 1] - a[j];
            j++;
            break;
        case 1: //1, j==1
            if (bytes1[i] == '_') {
                x += 1; y += 2;
                i++;
            }
            next = a[j + 2] - a[j + 1];
            j += 3;
            break;
        case 2:
            System.out.println(Arrays.toString(bytes2));
            return Arrays.toString(bytes2).equals(Arrays.toString(bytes));
```

```
        case 3: //4, j==1
            if(i == bytes1.length){
                next = a[j] * a[j - 1];
                j += 2;
            }
            else{
                next = a[j + 1] / a[j] / 2;
            }
            break;
        case 4: //3, j==3
            if (x < y || i < 5) {
                x += 1; y += 2;
                i += x / y + y / (4 * x); //均值不等式，即i++
            } else {                      //增加伪造分支
                x += 2; y += 1;
                i += y / (4 * x);
            }
            next = a[j + 1] / a[j - 1];
            j -= 2;
            break;
        case 5: //2, j==4
            index = encode(i, cal, x, y);
            bytes2[index] = bytes1[i];
            next = a[j + 1] / a[j - 2];
            j--;
            break;
    }
}
```

3、Button4 推箱子及 Button2、5、6 fakeflag：native code，常量折叠，手动混淆函数名

1）手动修改函数名称，将游戏地图的长，宽，箱子数量等常量作为函数参数传递

```
bool djiehiu(char* c);
void Jogoheuibwnr(int map[12][12], int box[255][2], int WIDTH, int LENGTH, int BOX,int* p);
bool jwhbiriwrnwjoqj(int map[12][12], int box[255][2], int dir, int WIDTH, int LENGTH, int BOX, int* player);

int jwehuibipjohio(int map[12][12], int* player, int dir, int WIDTH, int LENGTH, int DIR[5][2]);
int jwiubhiwbhwb(int map[12][12], int* player, int box[255][2], int dir, int DIR[5][2], int BOX);
int iojugyub(int map[12][12], int* player, int box[255][2], int dir, int WIDTH, int LENGTH, int DIR[5][2], int BOX);
```

2）常量折叠，并且在推箱子的方向控制中手动加入混淆的 case

```
//constant unfold
int WIDTH = 2, LENGTH = 9, BOX = 0;
int a=WIDTH,b=LENGTH, c=BOX;
int r = abs(random())%100;
a = a+b+(r%11-((r%10-r/10)%11)+1)%11-1;//11
c = a+r;c=c-b;c=c-r;//2
b += (r%11-((r%10-r/10)%11))%11;//9
WIDTH = a; LENGTH = b; BOX = c;
```

```
switch (ch) {
    case '!'://'w':
        dir = i*(i+1)*(i+2)*(i+3)%2;//0;
        break;
    case '_'://'s':
        int z ;
        z = i%100;
        dir = (z%11-((z%10-z/10)%11)+1)%11;//1
        break;
    case '*'://'a':
        dir = i*i*(i+1)*(i+1)%4+2;//2
        break;
    case '#'://'d':
        dir = i%2+(i+1)%2+(i+2)%2+(i+3)%2+(i+4)%2+(i+5)%2;//3
        break;
```

```
    //obfuscation
case '@':
    dir = i*(i+2)%2+4;
    break;
case '$':
    dir = i%2+(i+1)%2+(i+2)%2+(i+3)%2+(i+4)%2+(i+5)%2 + (i+6)%2+(i+7)%2;//4
    break;
case '%':
    dir = (z%11-((z%10-z/10)%11)+4)%11;//4
    break;
case '^':
    dir = i%2+(i+1)%2+(i+2)%2+(i+3)%2 +2;//4
    break;
case '&':
    dir = (int)(((char)(i%255)|0xff)^0xfb);//4
    break;
case '+' :
    dir = (((char)(i%255)&0x00)^0x34)-'0';//4
default://other input the player will stand still
    int m = dir%4;
    dir = 4 - m + i; dir = dir + m; dir = dir + i ;//4
```

3）游戏地图为二维矩阵常量，设置为整型二维数组，数组大小大于地图实际大小。
由于数组中每个元素为小于 15 的正数，因此只有最低位的字节的数据有效，而高位的三个字节都为 0；并且由于数组大小大于地图实际大小，有效数据间会有很多无效的 0，对逆向造成一定难度。如下图对应四个整型数据 0，1，1，1

```
unk_2630        db    0
                db    0
                db    0
                db    0
                db    1
                db    0
                db    0
                db    0
                db    1
                db    0
                db    0
                db    0
                db    1
                db    0
                db    0
                db    0
```

4、Button7 AES 加密反逆向处理

1、将密钥扩展、加密函数从整体拆分为多个部分，增加整体代码阅读理解难度

```
⊞void RKey_Add(byte sta_matr[4 * 4], word w[4]) { ... }

⊞void ShiftRow(byte sta_matr[4 * 4]) { ... }

⊟void encrypt(byte sta_matr[4 * 4], word w[4 * (Nr + 1)]) {
    word key[4];
    for (int i = 0; i < 4; i++)
        key[i] = w[i];
    RKey_Add(sta_matr, key);
⊟   for (int r = 1; r <= Nr; r++) {
        ShiftRow(sta_matr);
        for (int i = 0; i < 4; i++)
            key[i] = w[4 * r + i];
        RKey_Add(sta_matr, key);
    }
    cout << endl;
}
```

2、混合加入位运算操作和操作数类型转换，使其最终反编译结果呈现多个函数套壳的情况

```
  while (i < 4 * (Nr + 1)) {
      temp = w[i - 1];
      if (i % Nk == 0)
          w[i] = w[i - Nk] ^ rcon[i / Nk - 1];
      else
          w[i] = w[i - Nk] ^ temp;
      i++;
  }


oid RKey_Add(byte sta_matr[4 * 4], word w[4]) {
  for (int i = 0; i < 4; i++) {
      word k0 = w[i] >> 24;
      word k1 = (w[i] << 8) >> 24;
      word k2 = (w[i] << 16) >> 24;
      word k3 = (w[i] << 24) >> 24;

      sta_matr[i] = sta_matr[i] ^ byte(k0.to_ulong());
      sta_matr[i + 4] = sta_matr[i + 4] ^ byte(k1.to_ulong());
      sta_matr[i + 8] = sta_matr[i + 8] ^ byte(k2.to_ulong());
      sta_matr[i + 12] = sta_matr[i + 12] ^ byte(k3.to_ulong());
```

小组成员参与细节

杨舒晗（组长）:
fakeflag 设计：native code 简单 XOR 实现；Button4 推箱子游戏（native code）设计；
小组项目以及设计文档整合

王楠欣:
小组前期工作思路、日程整理；加密算法合集、安卓开发教学资源等资料分享；apk 首页 start 拦截和点击 start 按钮后进入的 wonderland 页面设计；Button1 flagpiece 设计：时间 service 和 killprocess 代码拦截页面跳转+自定义加解密函数+BASE64 加密+DES 加密

李丹琦:
Button3 加密变换，控制流混淆设计

李昊霖:
Button7 AES 加密（native code）设计