Lab2

16302010039_崔帅帅

Challenge1

在看过 smali 代码之后发现该 challenge 主要在于一个 check(string input)函数

这段是函数签名，即 public int check（String input）

```
.method public check(Ljava/lang/String;)I
    .registers 9
    .param p1, "input"    # Ljava/lang/String;
```

定义 a，b 两个字符串变量

String a = “qwerfdsazxcvbgtyhjuioplkmnb”;

String b = “czgstdubqvz”;

```
.line 54
const-string/jumbo v0, "qwerfdsazxcvbgtyhjuioplkmnb"

.line 55
.local v0, "a":Ljava/lang/String;
const-string/jumbo v1, "czgstdubqvz"

.line 56
.local v1, "b":Ljava/lang/String;
```

先对输入的 input 长度做初步判断，如果长度不合适，就直接返回 0。

```
invoke-virtual {p1}, Ljava/lang/String;->length()I

move-result v5

invoke-virtual {v1}, Ljava/lang/String;->length()I

move-result v6

if-ne v5, v6, :cond_29
```

从此处可以看出，该函数有一个 for 循环，循环长度为 input 的 length

```
.line 58
const/4 v2, 0x0

.local v2, "i":I
:goto_13
invoke-virtual {p1}, Ljava/lang/String;->length()I

move-result v5

if-ge v2, v5, :cond_2d
```

```
.line 68
:cond_2d
const/4 v4, 0x1
```

将 input 字符串做处理判断，如果不符合 input 的输入要求，则返回 0，符合则继续循环

```
.line 59
invoke-virtual {p1, v2}, Ljava/lang/String;->charAt(I)C

move-result v5

add-int/lit8 v3, v5, -0x61

.line 60
invoke-virtual {v0, v3}, Ljava/lang/String;->charAt(I)C

move-result v5

invoke-virtual {v1, v2}, Ljava/lang/String;->charAt(I)C

move-result v6

if-eq v5, v6, :cond_2a
```

```
.line 58
.restart local v2    # "i"
:cond_2a
add-int/lit8 v2, v2, 0x1
```

然后整理之后得到如下 check 函数：

```
public int check(String input){
    String a = "qwerfdsazxcvbgtyhjuioplkmnb";
    String b = "czgstdubqvz";
    int j = 0;
    if (input.length() == b.length()){
        for (int i = 0; i < input.length(); i++) {
            if (a.charAt(input.charAt(i) - 'a') != b.charAt(i))
                return 0;
        }
        return 1;
    }
    return 0;
}
```

如果 input 输入正确则返回 1，否则返回 0。

通过解密得到 input 为“kingofsmali”

不知道是手误还是什么那个字符串 a 好像多了一个字母 b，然后我发现字符串“kingofs{ali”也是正确的。

Challenge2

通过 challenge2 的 smali 代码发现主要逻辑就是 verify 函数，大致如下：

这是函数签名，private boolean verify（String key）

```
.method private verify(Ljava/lang/String;)Z
    .registers 6
    .param p1, "key"    # Ljava/lang/String;
```

声明定义字符串变量 flag

String flag = “&we[hkra[aj_nulpekjy”

```
.line 48
:cond_4
const-string v1, "&we[hkra[aj_nulpekjy"

.line 51
.local v1, "flag":Ljava/lang/String;
```

声明定义一个 StringBuilder

```
new-instance v2, Ljava/lang/StringBuilder;

invoke-direct {v2}, Ljava/lang/StringBuilder;-><init>()V

.line 53
.local v2, "stringBuilder":Ljava/lang/StringBuilder;
```

一个以 key 长度的 for 循环

```
.line 53
.local v0, "i":I
:goto_c
invoke-virtual {p1}, Ljava/lang/String;->length()I

move-result v3

if-ge v0, v3, :cond_1f

.line 57
.end local v0    # "i":I
:cond_1f
```

将输入字符串减少 4

```
.line 54
invoke-virtual {p1, v0}, Ljava/lang/String;->charAt(I)C

move-result v3

add-int/lit8 v3, v3, -0x4

int-to-char v3, v3

invoke-virtual {v2, v3}, Ljava/lang/StringBuilder;->append(C)Ljava/lang/StringBuilder;
```

整理之后得到如下 verify 函数

```
private boolean verify(String key){
    if (key.length() == 0) {
        return false;
    }
    String flag = "&we[hkra[aj_nulpekjy";
    StringBuilder stringBuilder = new StringBuilder();
    for (int i = 0; i < key.length(); i++) {
        stringBuilder.append((char)(key.charAt(i) - 4));
    }
    return stringBuilder.toString().equals("&we[hkra[aj_nulpekjy");
}
```

如果输入值正确会返回 `true`，否则返回 `false`
不知道是我解读的问题还是什么，他的 flag 变量好像没有用到。
不过最终解密还是没有任何影响，
我得到的结果是*{i_love_encryption}最后验证结果正确