安卓逆向学习之KGBMessenger的writeup（3）

原创

CowboyBeboppp 于 2021-05-21 16:51:46 发布 23 收藏 1

分类专栏： 安卓逆向

本文链接：https://blog.csdn.net/weixin_38109420/article/details/117091277

安卓逆向学习之KGB Messenger的writeup（3）

总结

1.java运算符的优先级。

2.若逆向解密无法完成的话可以考虑正向破解（穷举）（虽然效率低，但也是没有办法的办法）

Social Engineering (Hard)

- 进入聊天界面就可以看到特工们的对话了，看起来好像有人不会保守秘密，这波是使用社会工程学（其实没啥关系）来套密码。

思路

- 根据签名的方法分析java代码：MessengerActivity，发现一个重要的函数onSendMessage，其逻辑就是发送一句话就在RecyclerView组件中显示出来，而里面有两个重要的函数会触发两句话，其中一句还包含了FLAG：

```
public void onSendMessage(View paramView) {
  EditText editText = (EditText)findViewById(2131165225);
  String str = editText.getText().toString();
  if (!TextUtils.isEmpty(str)) {
    this.o.add(new a(2131558449, str, i(), false));
    this.n.c();
    if (a(str.toString()).equals(this.p)) {
      Log.d("MessengerActivity", "Successfully asked Boris for the password.");
      this.q = str.toString();
      this.o.add(new a(2131558434, "Only if you ask nicely", i(), true));
      this.n.c();
    }
    if (b(str.toString()).equals(this.r)) {
      Log.d("MessengerActivity", "Successfully asked Boris nicely for the password.");
      this.s = str.toString();
      this.o.add(new a(2131558434, "Wow, no one has ever been so nice to me! Here you go friend: FLAG{" + i() + "}", i(), true));
      this.n.c();
    }
    this.m.b(this.m.getAdapter().a() - 1);
    editText.setText("");
  }
}
```

- 所以a函数和b函数明显是分析的重点，首先分析a函数：

```
private String a(String paramString) {
  char[] arrayOfChar = paramString.toCharArray();
  for (byte b = 0; b < arrayOfChar.length / 2; b++) {
    char c1 = arrayOfChar[b];
    arrayOfChar[b] = (char)(char)(arrayOfChar[arrayOfChar.length - b - 1] ^ 0x32);
    arrayOfChar[arrayOfChar.length - b - 1] = (char)(char)(c1 ^ 0x41);
  }
  return new String(arrayOfChar);
}
```

- 发现其逻辑就是对输入的字符串的前半部分后后半部分分别与两个数字相与后替换位置然后返回，最后在onSendMessage中与this.p进行比较：

```
private String p = "V@]EAASB\022WZF\022e,a$7(&am2(3.\003";
```

- 所以这里逆推就能获得相应的输入啦：

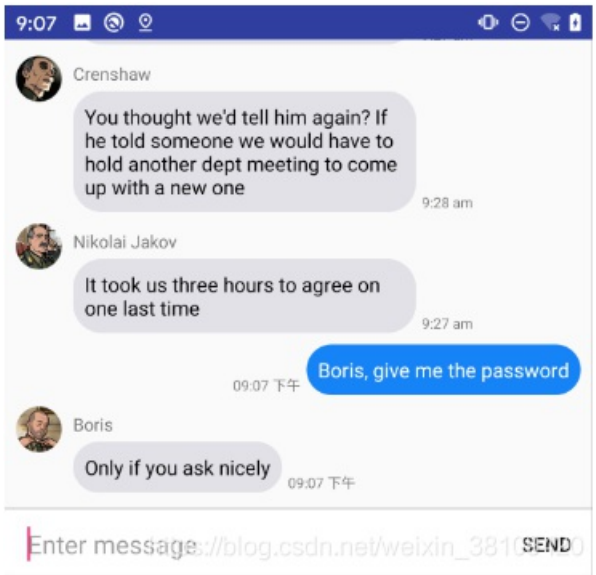
```
public class kgb3{
    public static void main(String [] args){
        String key = "V@]EAASB\022WZF\022e,a$7(&am2(3.\003";
        System.out.print(reverse(key));
    }

    private static String reverse(String paramString) {
    char[] arrayOfChar = paramString.toCharArray();
    for (byte b = 0; b < arrayOfChar.length / 2; b++) {
      char c1 = arrayOfChar[b];
      arrayOfChar[b] = (char)(char)(arrayOfChar[arrayOfChar.length - b - 1] ^ 0x41);
      arrayOfChar[arrayOfChar.length - b - 1] = (char)(char)(c1 ^ 0x32);
    }
    return new String(arrayOfChar);
    }
}
```

- 输出：

```
PS C:\Users\DELL\Desktop\java\ctf> javac kgb3.java
PS C:\Users\DELL\Desktop\java\ctf> java kgb3
Boris, give me the password
```

- 发出该消息就可以得到回复了：



- 想要触发flag就应该分析b函数：

```
private String b(String paramString) {
```

```
  byte b2;
  byte b1 = 0;
  char[] arrayOfChar = paramString.toCharArray();
  char c1 = Character.MIN_VALUE;
  while (true) {
    b2 = b1;
    if (c1 < arrayOfChar.length) {
      arrayOfChar[c1] = (char)(char)(arrayOfChar[c1] >> c1 % 8 ^ arrayOfChar[c1]);
      c1++;
      continue;
    }
    break;
  }
  while (b2 < arrayOfChar.length / 2) {
    c1 = arrayOfChar[b2];
    arrayOfChar[b2] = (char)arrayOfChar[arrayOfChar.length - b2 - 1];
    arrayOfChar[arrayOfChar.length - b2 - 1] = (char)c1;
    b2++;
  }
  return new String(arrayOfChar);
}
```


- 首先分析第一个while循环发现里面其实有一个冗余信息b2=b1，可能是jd-gui反编译的问题。
- 循环里面其实对整个字符串的每一位都进行了一个重复操作，分析此操作：

```
arrayOfChar[c1] = (char)(char)(arrayOfChar[c1] >> c1 % 8 ^ arrayOfChar[c1]);
```

- 通过查看java运算符的优先级：

表1 运算符的优先级

优先级	运算符	结合性
1	()、[]、{}	从左向右
2	!、+、-、~、++、--	从右向左
3	*、/、%	从左向右
4	+、-	从左向右
5	«、»、>>>	从左向右
6	<、<=、>、>=、instanceof	从左向右
7	==、!=	从左向右
8	&	从左向右
9	^	从左向右
10	|	从左向右
11	&&	从左向右
12	||	从左向右
13	?:	从右向左
14	=、+=、-=、*=、/=、&=、|=、^=、~=、«=、»=、>>>=	从右向左

- 可以看出是先对下标整除8后，原字符右移该数值后再与自己相与

```
arrayOfChar[c1] = (char)(char)(  (arrayOfChar[c1] >> (c1 % 8)) ^ arrayOfChar[c1] );
```

- 我们知道一般加密和解密的操作如下：

```
加密：
明文^key = 密文
解密：
密文^key = 明文
其中的逻辑：
某一值与同一数值相与两次该值不变
```

- 而这里相与的操作与自己进行了相与，所以无法通过密文推出明文。
- 而可以看到下面的循环是进行了一个reverse的操作。
- 这时候我们的思路不是通过密文反推明文，因为改变无法实现，这时候就只能用最简单的办法——**正向穷举破解**。
- 我们正向对一个字符进行上述循环中的操作然后观察其输出值是否等于“密文”中的字符，如果相等那么就是原始的字符。
- 由于我们不知道原字符到底是个什么字符，这里我们通过python中的string库得到可能的字符：

```
>>> string.printable
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~ \t\n\r\x0b\x0c
'
```

- 然后编写代码如下

```
public class kgb3{
 public static void main(String [] args){
  String key = "\000dslp}oQ\000 dks$|M\000h +AYQg\000P*!M$gQ\000";
  search(key);
 }

 private static void search(String paramString) {
  paramString = reverse(paramString);
  String test = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!\"#$%&\'()*+,-./:;<=>?@[\\]^_`{|
}~ \t\n\r";
  char [] arrayOftest = test.toCharArray();
  char [] arrayOfparam = paramString.toCharArray();
  //System.out.print(test);
  for(int i = 0 ; i < arrayOfparam.length; i++)
  {
   for(int j = 0; j < arrayOftest.length ; j++ ){
    char c = arrayOftest[j];
    c = (char)(char)((c >> (i % 8) )^ c);
    if(c == arrayOfparam[i]){
     System.out.print(arrayOftest[j]);
     break;
    }
   }
  }
 }

 public static String reverse(String str){
  return new StringBuilder(str).reverse().toString();
 }
}
```

- 最后输出为

```
aay I *PaEASE* have the aassworda
```

- 输出的字符串里奇怪的字符a其实是因为原字符为\000，而当i % 8= 0 时，没有进行移位，也就是相当于自己和自己异或，结果就是0，等于\000，于是就直接输出a了。不过不影响结果，最后得到flag~