Java安全漫谈 - 03.反射篇(3)

> 这是代码审计知识星球中Java安全的第三篇文章。

上次讲了个简单的命令执行Payload，但遗留下来两个问题：

- 如果一个类没有无参构造方法，也没有类似单例模式里的静态方法，我们怎样通过反射实例化该类呢？
- 如果一个方法或构造方法是私有方法，我们是否能执行它呢？

第一个问题，我们需要用到一个新的反射方法`getConstructor`。

和`getMethod`类似，`getConstructor`接收的参数是构造函数列表类型，因为构造函数也支持重载，所以必须用参数列表类型才能唯一确定一个构造函数。

获取到构造函数后，我们使用`newInstance`来执行。

比如，我们常用的另一种执行命令的方式ProcessBuilder，我们使用反射来获取其构造函数，然后调用`start()`来执行命令：

```
Class clazz = Class.forName("java.lang.ProcessBuilder");
((ProcessBuilder)
clazz.getConstructor(List.class).newInstance(Arrays.asList("calc.exe"))).star
t();
```

ProcessBuilder有两个构造函数：

- `public ProcessBuilder(List<String> command)`
- `public ProcessBuilder(String... command)`

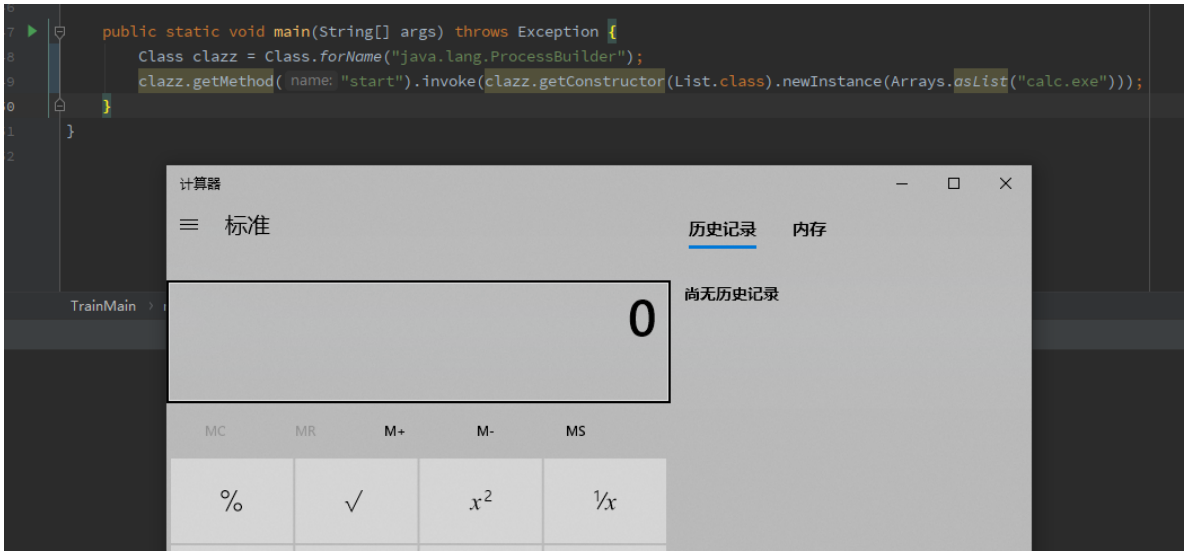
我上面用到了第一个形式的构造函数，所以我在`getConstructor`的时候传入的是`List.class`。

但是，我们看到，前面这个Payload用到了Java里的强制类型转换，有时候我们利用漏洞的时候（在表达式上下文中）是没有这种语法的。所以，我们仍需利用反射来完成这一步。

其实用的就是前面讲过的知识：

```
Class clazz = Class.forName("java.lang.ProcessBuilder");
clazz.getMethod("start").invoke(clazz.getConstructor(List.class).newInstance(
Arrays.asList("calc.exe")));
```

通过`getMethod("start")`获取到start方法，然后`invoke`执行，`invoke`的第一个参数就是ProcessBuilder Object了。



那么，如果我们要使用`public ProcessBuilder(String... command)`这个构造函数，需要怎样用反射执行呢?

这又涉及到Java里的可变长参数（varargs）了。正如其他语言一样，Java也支持可变长参数，就是当你定义函数的时候不确定参数数量的时候，可以使用`...`这样的语法来表示"这个函数的参数个数是可变的"。

对于可变长参数，Java其实在编译的时候会编译成一个数组，也就是说，如下这两种写法在底层是等价的（也就不能重载）：

```
public void hello(String[] names) {}
public void hello(String...names) {}
```

也由此，如果我们有一个数组，想传给hello函数，只需直接传即可：

```
String[] names = {"hello", "world"};
hello(names);
```

那么对于反射来说，如果要获取的目标函数里包含可变长参数，其实我们认为它是数组就行了。

所以，我们将字符串数组的类`String[].class`传给`getConstructor`，获取`ProcessBuilder`的第二种构造函数：

```
Class clazz = Class.forName("java.lang.ProcessBuilder");
clazz.getConstructor(String[].class)
```

在调用`newInstance`的时候，因为这个函数本身接收的是一个可变长参数，我们传给`ProcessBuilder`的也是一个可变长参数，二者叠加为一个二维数组，所以整个Payload如下：

```
Class clazz = Class.forName("java.lang.ProcessBuilder");
((ProcessBuilder)clazz.getConstructor(String[].class).newInstance(new
String[][]{{"calc.exe"}})).start();
```

有兴趣可以尝试把这个Payload改成完全反射编写。

再说到今天第二个问题，如果一个方法或构造方法是私有方法，我们是否能执行它呢?

这就涉及到`getDeclared`系列的反射了，与普通的`getMethod`、`getConstructor`区别是：

- `getMethod`系列方法获取的是当前类中所有公共方法，包括从父类继承的方法
- `getDeclaredMethod`系列方法获取的是当前类中"声明"的方法，是实在写在这个类里的，包括私有的方法，但从父类里继承来的就不包含了

自己理解一下差别吧。

`getDeclaredMethod`的具体用法和`getMethod`类似，`getDeclaredConstructor`的具体用法和`getConstructor`类似，我就不再赘述。

举个例子，前文我们说过Runtime这个类的构造函数是私有的，我们需要用`Runtime.getRuntime()`来获取对象。其实现在我们也可以直接用`getDeclaredConstructor`来获取这个私有的构造方法来实例化对象，进而执行命令：

```
Class clazz = Class.forName("java.lang.Runtime");
Constructor m = clazz.getDeclaredConstructor();
m.setAccessible(true);
clazz.getMethod("exec", String.class).invoke(m.newInstance(), "calc.exe");
```

可见，这里使用了一个方法`setAccessible`，这个是必须的。我们在获取到一个私有方法后，必须用`setAccessible`修改它的作用域，否则仍然不能调用。