Lab II – AspectJ, IRM's, and Reverse Engineering

CPS 499-02/592-02

Software/Language Based Security

Fall 2020

Dr. Phu Phung

Evan Krimpenfort

Part I: Aspect-Oriented Programming and IRM's

Task I: Getting Started with AspectJ

This lab incorporates us working with AspectJ which will work on AOP and IRM's inside of java files. This is the beginning of getting used to the program.

```
[09/17/20]seed@VM:~/.../AspectJ$ ls
HelloAspect.aj  HelloAspect.class  Hello.class  Hello.java
[09/17/20]seed@VM:~/.../AspectJ$ java -cp ".:/usr/share/java/aspectjrt.jar" Hell
o AOP Test from Evan Krimpenfort. The time and date are September 17th, 4:05PM
AOP test: Hello World!
```

Figure 1: using an AspectJ program given to us

Task II: Writing in AspectJ

AspectJ involves modifications before, during, and after a called function. In figure 2, we added a print out after the call "greeting."

```
[09/17/20]seed@VM:~/.../AspectJ$ java -cp ".:/usr/share/java/aspectjrt.jar" Hell
o AOP Test from Evan Krimpenfort. The time and date are September 17th, 4:05PM
AOP test: Hello World!
: AOP Test Afterwards.
```

Figure 2: Modifying the AspectJ program to user an after method

Task III: Using AspectJ to Modify Java ByteCode

a. Packing Java ByteCode into a .jar file

You can also take this Aspect oriented program and weave it into a .jar file with the file you want to hook into. You can see the process in Figures 3, 4, and 5.

```
[09/17/20]seed@VM:~/.../AspectJ$ ls
HelloAspect.aj  HelloAspect.class  Hello.class  Hello.java
[09/17/20]seed@VM:~/.../AspectJ$ javac Hello.java
[09/17/20]seed@VM:~/.../AspectJ$ java Hello
Hello World!
[09/17/20]seed@VM:~/.../AspectJ$ jar -cvfe Hello.jar Hello Hello.class
added manifest
adding: Hello.class(in = 495) (out= 323)(deflated 34%)
[09/17/20]seed@VM:~/.../AspectJ$ java -jar Hello.jar
Hello World!
[09/17/20]seed@VM:~/.../AspectJ$ ls
HelloAspect.aj  HelloAspect.class  Hello.class  Hello.jar  Hello.java
[09/17/20]seed@VM:~/.../AspectJ$
```

Figure 3: Setting up Hello.jar

b. Weaving aspects to a .jar file

```
[09/17/20]seed@VM:~/.../AspectJ$ ajc -cp ".:/usr/share/java/aspectjrt.jar" -inpath Hello.jar
 HelloAspect.aj -outjar Hello-AOP.jar
[09/17/20]seed@VM:~/.../AspectJ$ ls
Hello-AOP.jar  HelloAspect.aj  HelloAspect.class  Hello.class  Hello.jar  Hello.java
[09/17/20]seed@VM:~/.../AspectJ$
```

Figure 4: Weaving AspectJ into Helslo.jar

```
[09/17/20]seed@VM:~/.../AspectJ$ java -cp ".:/usr/share/java/aspectjrt.jar" -javaagent:/usr/
share/java/aspectjweaver.jar -jar Hello-AOP.jar
AOP test: Hello World!
: AOP Test Afterwards.
[09/17/20]seed@VM:~/.../AspectJ$
```

Figure 5: running the weaved .jar file

Task IV: Getting Familiar with writing a security policy in AspectJ

a. Demonstration

In this section of the lab, we will be flashing back to the last experiment where we used ShoppingCart.java. We want to weave ourselves into the program and spy on Wallet.setBalance (int balance). When the function is referenced, our AOP file, ShoppingCartAspect.aj, will print out the exact date and time the function was called and what the balance parameter was. The demonstration seen in figures 6 and 7 below show where and when the AOP file makes its mark.

```
[09/17/20]seed@VM:~/.../ShoppingCart$ java -classpath ".:/usr/share/java/aspectjrt.jar:u
sr/share/java/aspectjtools.jar" -javaagent:/usr/share/java/aspectjweaver.jar -jar Shoppi
ngCart-IRM.jar
ShoppingCart program is running on port 8888
Waiting for connections from clients
A new customer is connected!
```

```
/bin/bash 88x26
[09/17/20]seed@VM:~/.../ShoppingCart$ telnet localhost 8888
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Welcome to Evan's ShoppingCart. The time now is Thu Sep 17 17:24:49 EDT 2020
Your balance: 29880 credits
Please select your product:
candies - 1
car - 30000
pen - 40
book - 100
What do you want to buy, type e.g., pen?
```

Figure 6: Starting the ShoppingCart-IRM.jar script

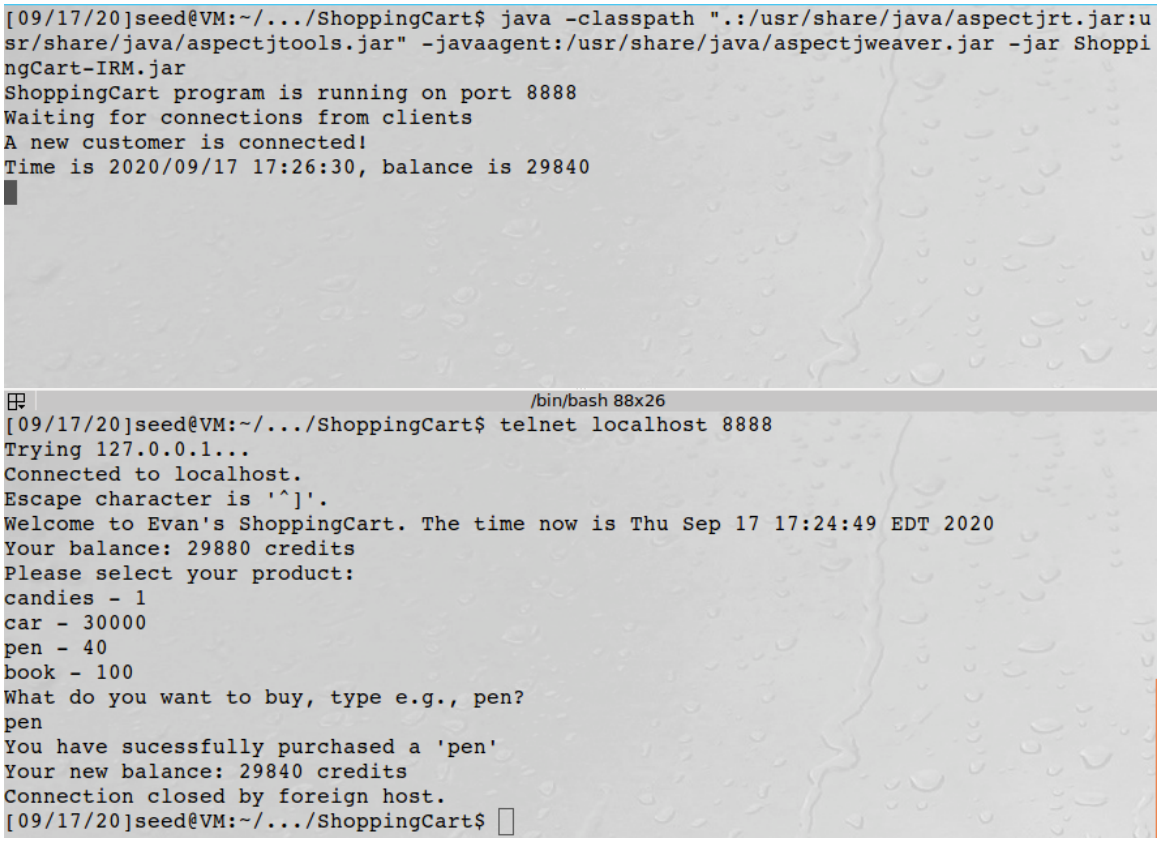

Figure 7: Showing the advice seen in the server

b. Test Description

When the setBalance(..) method is called, we see that the server shows the date, time, and balance after the function was called. How this happens is inside of the ShoppingCartAspect.aj program. Since I used the after method, this information comes to the server after a client purchases an item. This is nice in that now the user can see through the server on what the client's balance is after the purchase.

c. Aspect Source Code

To find the code, check it out in Appendix A.

Appendix A – ShoppingCartAspect.aj

```
import java.time.format.DateTimeFormatter;
import java.time.LocalDateTime;

public aspect ShoppingCartAspect
{
	after(int balance): call(* Wallet.setBalance(int)) && args(balance)
	{
		DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
		LocalDateTime now = LocalDateTime.now();
		System.out.println("Time is " + dtf.format(now) + ", balance is " +  balance);
	}
}
```

Part II: Reverse Engineering

Task V: Java ByteCode RE

a. Aspect Code Examination

When looking at figures 8 and 9, the difference between each .jar file is the added ShoppingCartAspect.class. This class imports the code that we will see in the next part within figure 10.

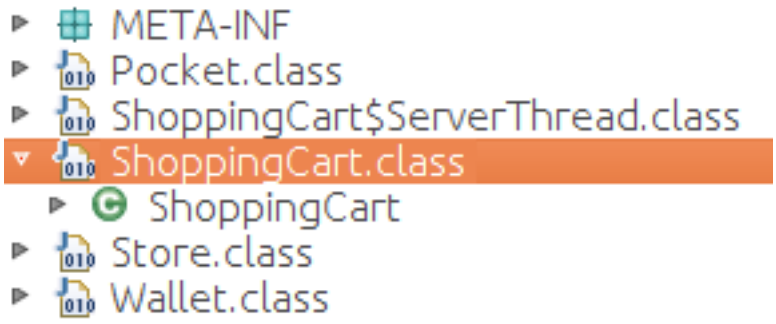

Figure 8: ShoppingCart.jar RE

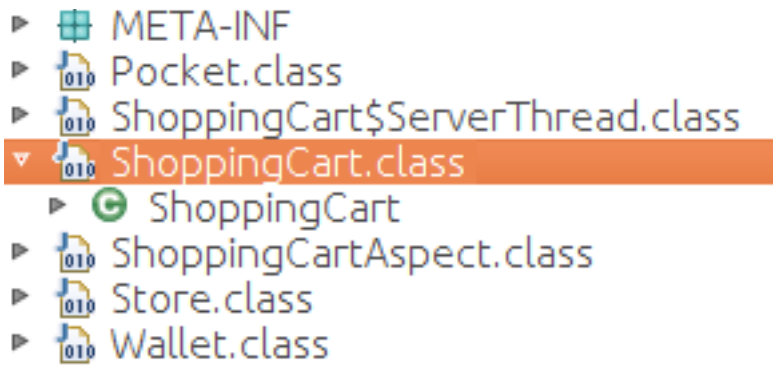

Figure 9: ShoppingCart-IRM.jar RE

b. Advice Code

```
61        int k = i - j;
          try {
61          wallet.setBalance(k);
          } catch (Throwable throwable) {
61          ShoppingCartAspect.aspectOf().ajc$after$ShoppingCartAspect$1$20aac00f(k);
61          throw throwable;
          }
61        ShoppingCartAspect.aspectOf().ajc$after$ShoppingCartAspect$1$20aac00f(k);
```

Figure 10: Aspect Code replacement

The Code that was injected in place of wallet.setBalance(k) was a try catch statement such that a throwable was made to run the aspect's after statement code. If the catch statement was never triggered, the same after statement was still ran, making sure that this code was always executed.

c. Code Mapping

Aspect-Oriented Programming is trying to add point cuts to an existing program without modifying the actual code of the program. These point cuts capture data at explicit execution points. AOP also involves breaking down those execution points into different interactions. This allows the programmer to focus on certain functions at certain times of execution, further enhancing the AOP experience. This level of abstraction is a huge part of AOP. Inline Reference Monitors are policies set inside of a finite-state automata. This provides a layer of policies between the program and anything outside of it. AOP can work inside of IRM's by placing the point cuts at locations in the program of which those functions go outside of itself and into the OS.

Task VI: Android RE

a. Running an PK Android pp in an android emulator

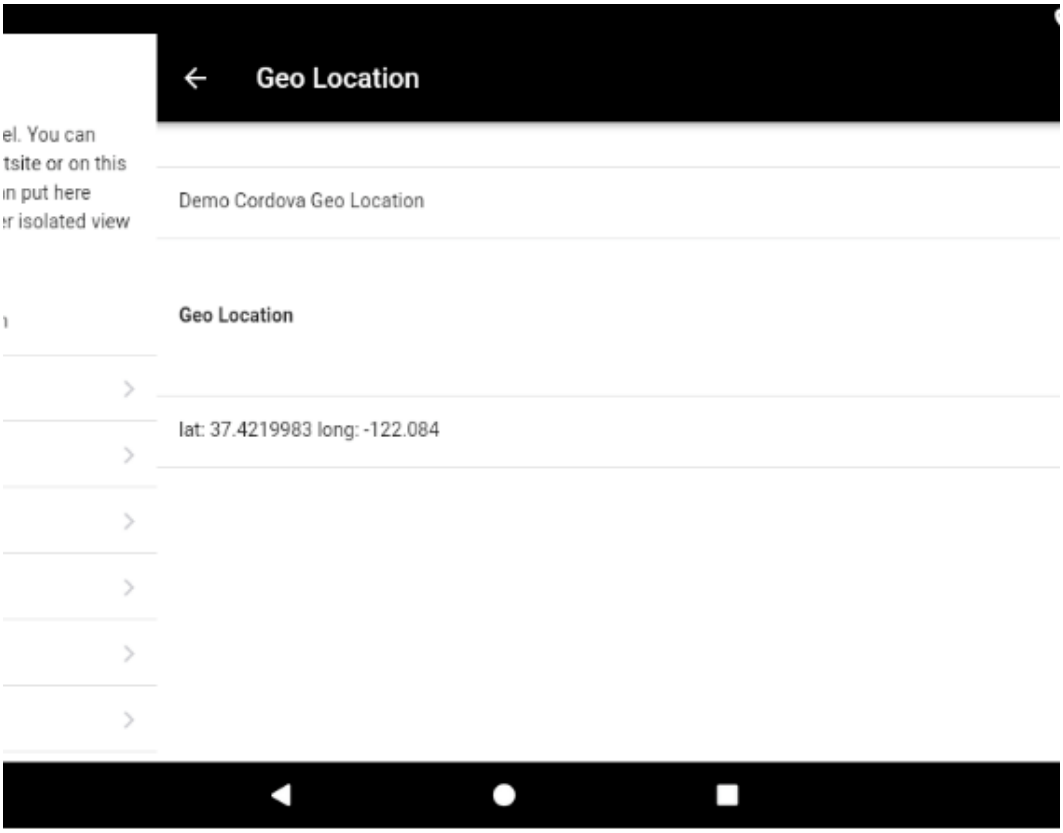

Figure 11: GPS of the Cordova App

b. Reverse engineer and modify a hybrid Android App

```
: \"material:menu\"\n                }\n              })\n            ],\n
    1\n          ),\n          _vm._v(\" \"),\n          _c(\"f7-nav-title\", [_vm
._v(\"Demo App - modified by Evan Krimpenfort\")]),\n          _vm._v(\" \"),\n
      _c(\n            \"f7-nav-right\",\n            [\n              _c(\"f7-li
nk\", {\n                staticClass: \"searchbar-enable\",\n                attrs
```

Figure 12: Demo App Modification

```
10854 /***/ 0:
10855 /*!**************************!*\
10856   !*** multi ./src/app.js ***!
10857   \**************************/
10858 /*! no static exports found */
10859 /***/ (function(module, exports, __webpack_require__) {
10860
10861 eval("module.exports = __webpack_require__(/*! ./src/app.js */\"./src/app.js\");\n
      \n\n//# sourceURL=webpack:///multi_./src/app.js?");
10862
10863 /***/ })
10864
10865 /******/ });
10866 alert('This Android App has been modified by Evan Krimpenfort.');
```

Figure 13: Alert at the end Modification

c. Rebuild a Modified Android App

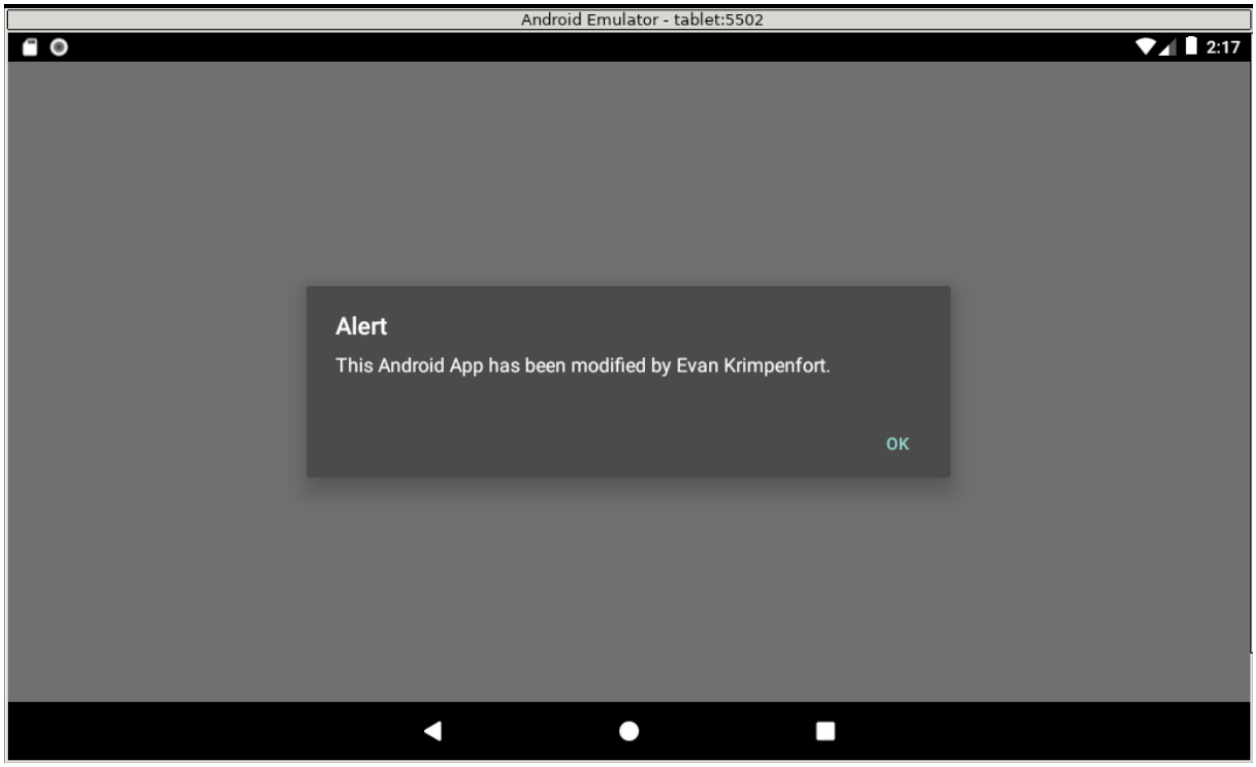

Figure 14: Alert shown in new app

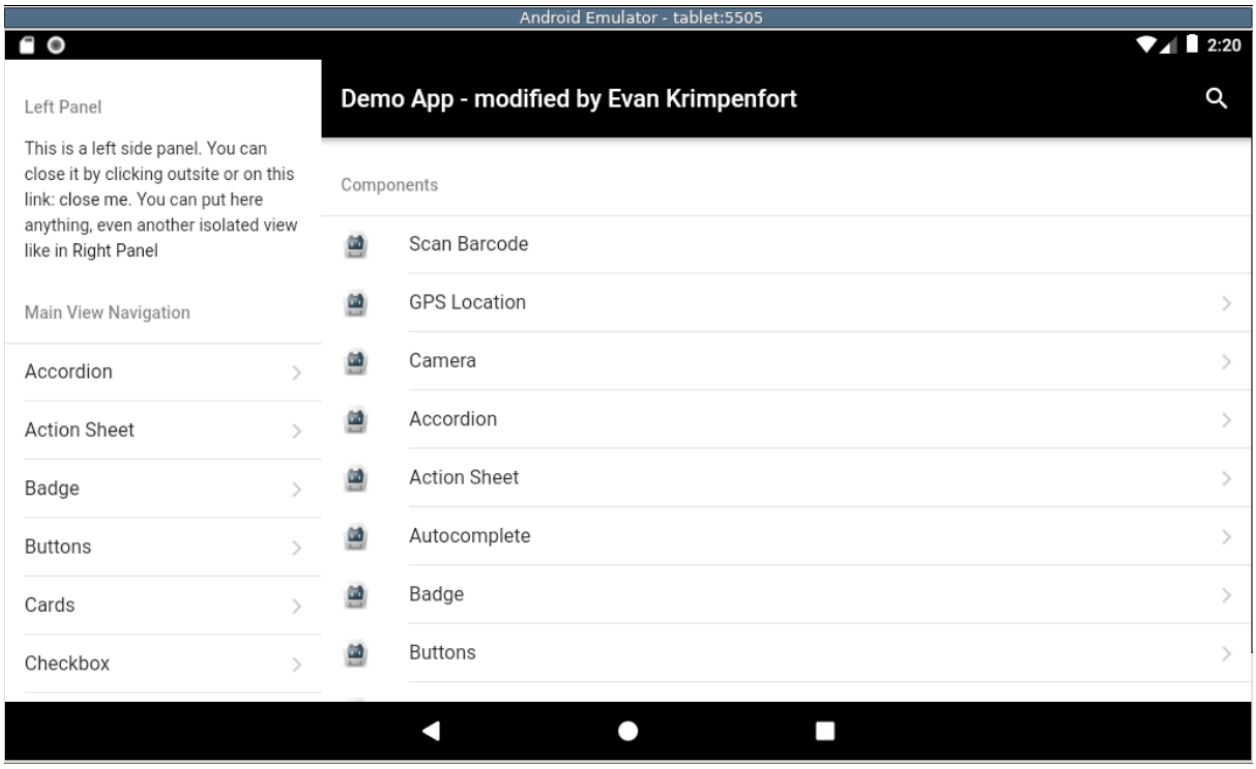

Figure 15: Name shown up by the Demo App space

The theme here is seeing how easy it is to take an apk, download it, unpack it, and modify it for your own benefit. And, from rewriting the .apk, you can set your name to it and pass it off like its yours since you added your Alias to it.