"Method Invocation Monitoring in PDFBox"

Part 2

Prepared for

Prof. Benoit Baudry

Prepared by

Ali Maher

4th of Nov. 2024

Abstract

Building on the prior case study, which explored monitoring method invocations and system properties within the "PDFBox" project, this report documents a follow-up effort to refine the tracking of runtime calls to third-party methods. This report introduces the use of AspectJ, which was developed to monitor method invocations directly in a demo project with simple encryption.

Brief Summary

First, I had a problem using AspectJ. I tried to write a custom agent to monitor the method invocations. I did something, but the result was not accurate. Then I again switched to AspectJ with more focus and reading some related resources (I mentioned the resources in the references section).

My latest attempt is as follows. I created a new workload. It consists of three parts: **aspectjdemo**, **aspectjpdfbox**, and **aspectjbouncycastle**.
The first two packages were about to test the AspectJ and learn how to work with it. And the central part is the aspectjbouncycastle.

As Bouncy Castle is related to encryption and signing pdf, I created a simple encryption class that encrypts a string using bouncycsatle packages. This folder consists of two files. The main file is used to run the encryption sample. The second file is our AspectJ file, which is used to configure our monitoring format.
I got help from two websites that were very useful for implementing the AspectJ file. The StackOverflow website's [link] and the Spring website's [link].

About the output

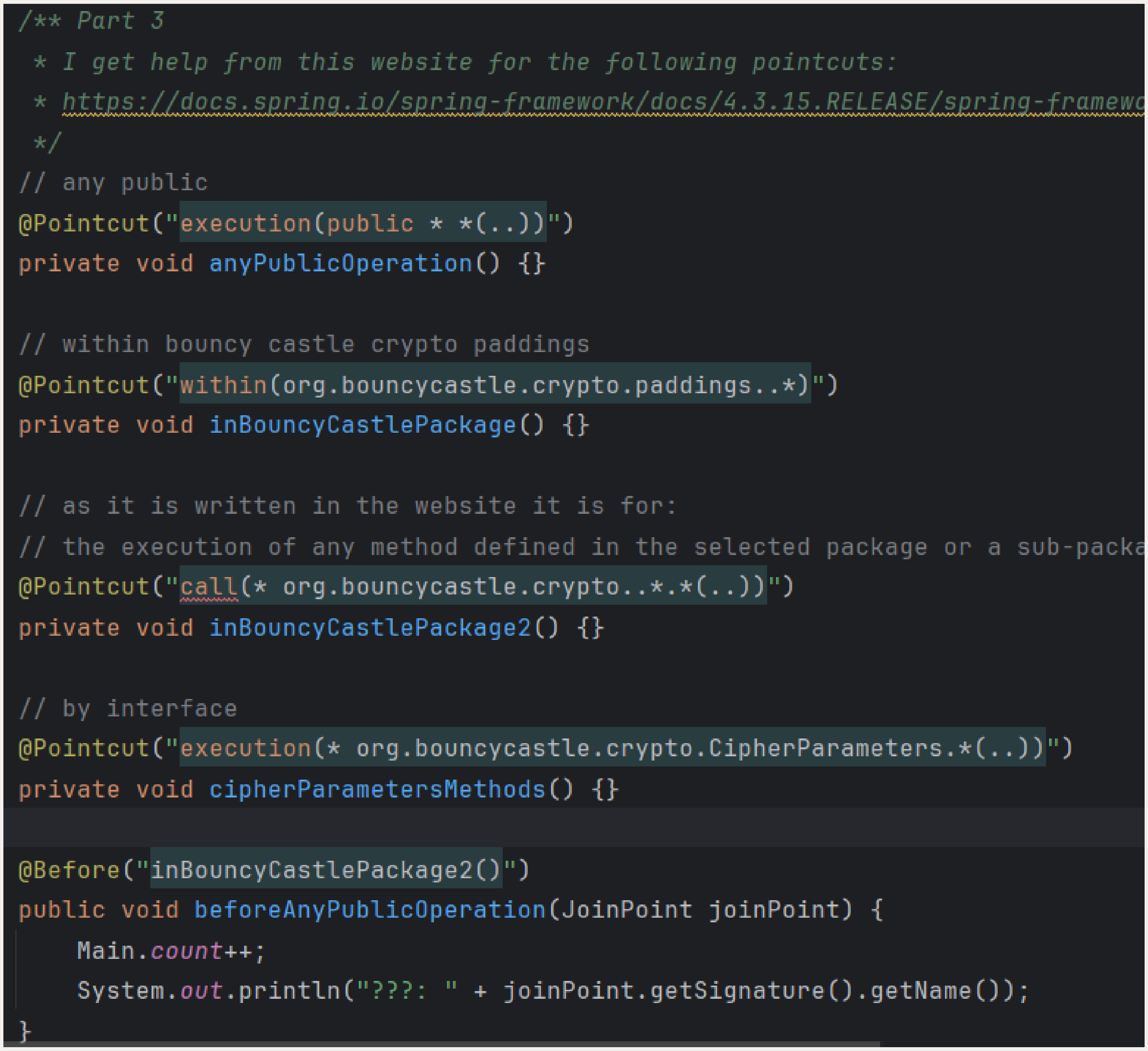
I classified the logs with some signs:

- All of the workload invocations start with ">>>".
- The Bouncy Castle's invocated methods monitored with the first approach begin with "**444**".
- The Bouncy Castle's invocated methods monitored with the second approach begin with "**???**".
- And we have two lines that indicate the encrypted text and decrypted text.

- Finally, we have "count", which shows the number of bouncy castle method invocations.

Other needed configurations

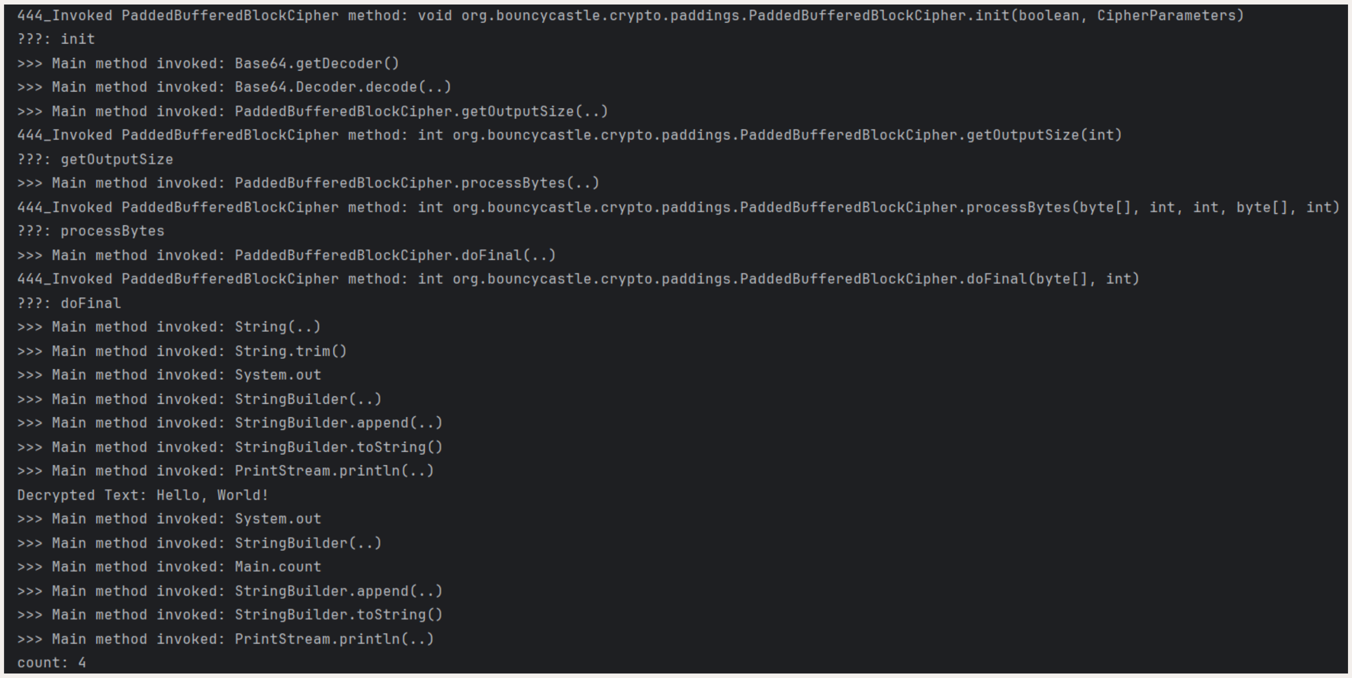
1. Make sure to add required dependencies to the project (like Bouncy Castle and AspectJ).
2. "aop.xml" file.



```
/** Part 3
 * I get help from this website for the following pointcuts:
 * https://docs.spring.io/spring-framework/docs/4.3.15.RELEASE/spring-framewo
 */
// any public
@Pointcut("execution(public * *(..))")
private void anyPublicOperation() {}

// within bouncy castle crypto paddings
@Pointcut("within(org.bouncycastle.crypto.paddings..*)")
private void inBouncyCastlePackage() {}

// as it is written in the website it is for:
// the execution of any method defined in the selected package or a sub-packa
@Pointcut("call(* org.bouncycastle.crypto..*.*(..))")
private void inBouncyCastlePackage2() {}

// by interface
@Pointcut("execution(* org.bouncycastle.crypto.CipherParameters.*(..))")
private void cipherParametersMethods() {}


@Before("inBouncyCastlePackage2()")
public void beforeAnyPublicOperation(JoinPoint joinPoint) {
    Main.count++;
    System.out.println("???: " + joinPoint.getSignature().getName());
}
```

Figure 1: AspectJ code, defining pointcuts, and where to monitor

Results



```
444_Invoked PaddedBufferedBlockCipher method: void org.bouncycastle.crypto.paddings.PaddedBufferedBlockCipher.init(boolean, CipherParameters)
???: init
>>> Main method invoked: Base64.getDecoder()
>>> Main method invoked: Base64.Decoder.decode(..)
>>> Main method invoked: PaddedBufferedBlockCipher.getOutputSize(..)
444_Invoked PaddedBufferedBlockCipher method: int org.bouncycastle.crypto.paddings.PaddedBufferedBlockCipher.getOutputSize(int)
???: getOutputSize
>>> Main method invoked: PaddedBufferedBlockCipher.processBytes(..)
444_Invoked PaddedBufferedBlockCipher method: int org.bouncycastle.crypto.paddings.PaddedBufferedBlockCipher.processBytes(byte[], int, int, byte[], int)
???: processBytes
>>> Main method invoked: PaddedBufferedBlockCipher.doFinal(..)
444_Invoked PaddedBufferedBlockCipher method: int org.bouncycastle.crypto.paddings.PaddedBufferedBlockCipher.doFinal(byte[], int)
???: doFinal
>>> Main method invoked: String(..)
>>> Main method invoked: String.trim()
>>> Main method invoked: System.out
>>> Main method invoked: StringBuilder(..)
>>> Main method invoked: StringBuilder.append(..)
>>> Main method invoked: StringBuilder.toString()
>>> Main method invoked: PrintStream.println(..)
Decrypted Text: Hello, World!
>>> Main method invoked: System.out
>>> Main method invoked: StringBuilder(..)
>>> Main method invoked: Main.count
>>> Main method invoked: StringBuilder.append(..)
>>> Main method invoked: StringBuilder.toString()
>>> Main method invoked: PrintStream.println(..)
count: 4
```

Figure 2: part of our output from running the workload

References

1. https://arxiv.org/pdf/2208.01321
2. https://pdfbox.apache.org/3.0/commandline.html
3. https://github.com/apache/pdfbox/
4. https://www.bouncycastle.org/
5. https://www.youtube.com/watch?v=5jeZjmwhIbQ
6. https://www.baeldung.com/aspectj
7. https://docs.spring.io/spring-framework/docs/4.3.15.RELEASE/spring-framework-reference/html/aop.html
8. https://stackoverflow.com/questions/60295366/how-to-monitor-method-execution-at-third-party-library
9. https://eclipse.dev/aspectj/doc/released/progguide/semantics-pointcuts.html
10. https://stackoverflow.com/questions/16318426/aspectj-pointcut-for-constructor-using-java-lang-reflection

The project files with doc are also available on my GitHub.

Thank you for your time and consideration.