Java RASP技术介绍

feng 2017-07-29

唯品会安全应急响应中心

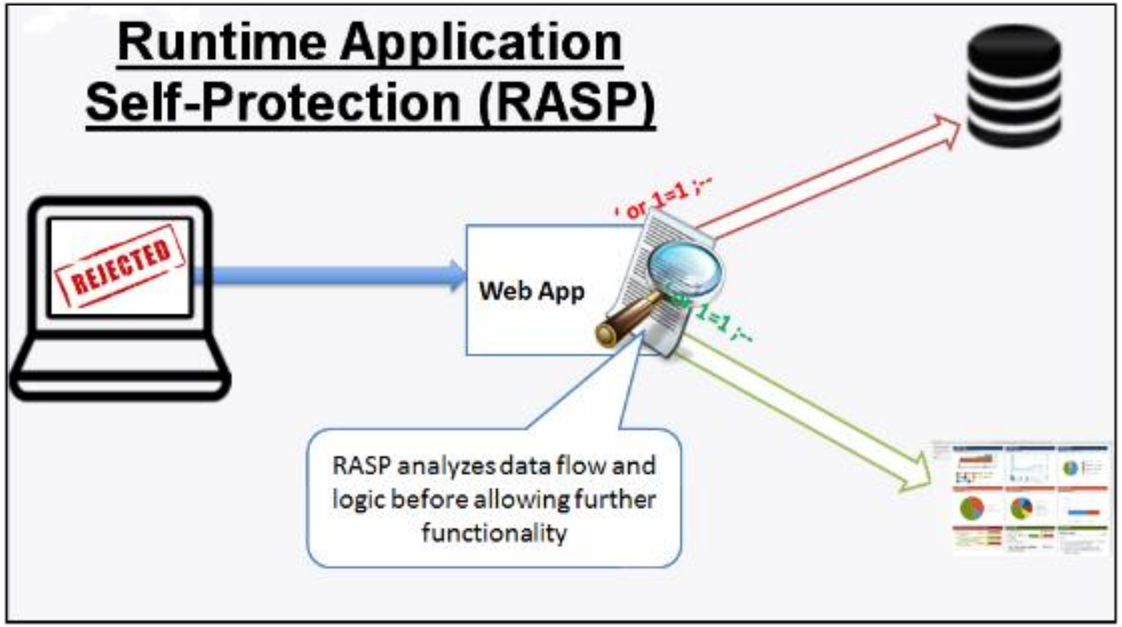
VIP Security Response Center (VSRC)

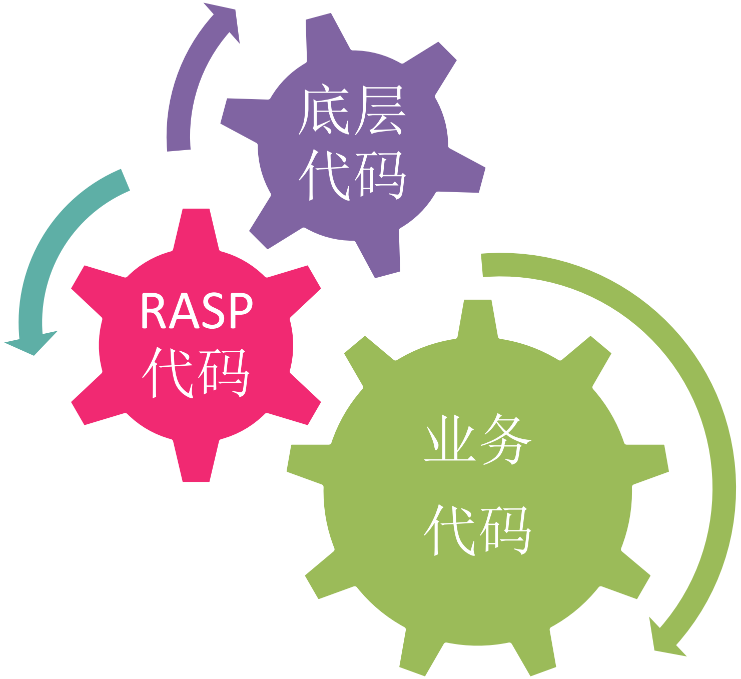
目 录

CONTENTS

01 什么是 RASP

RASP，即Runtime Application Self-Protection，实时应用自我保护。


Runtime Application Self-Protection (RASP)
REJECTED
Web App
' or 1=1 ;--
1=1 ;--
RASP analyzes data flow and logic before allowing further functionality



RASP实现原理示意

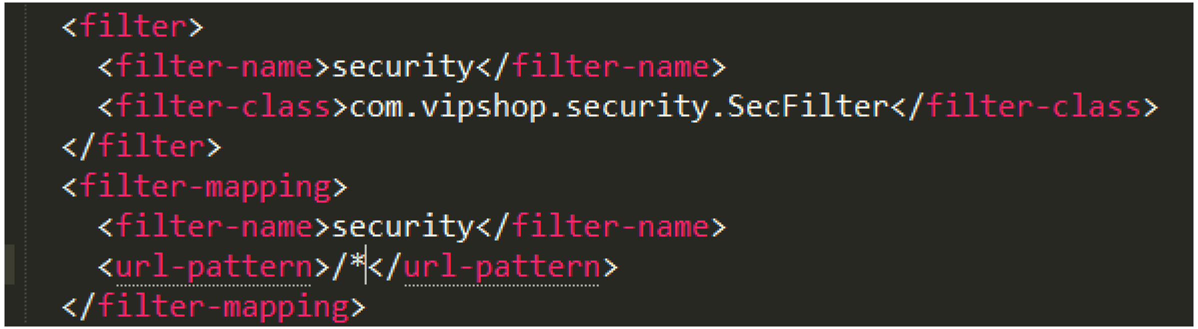
02

RASP的实现方案

Java Web

Filter

- 开发难度相对低
- 漏洞覆盖面小
- 对性能影响大
- 推广难度大

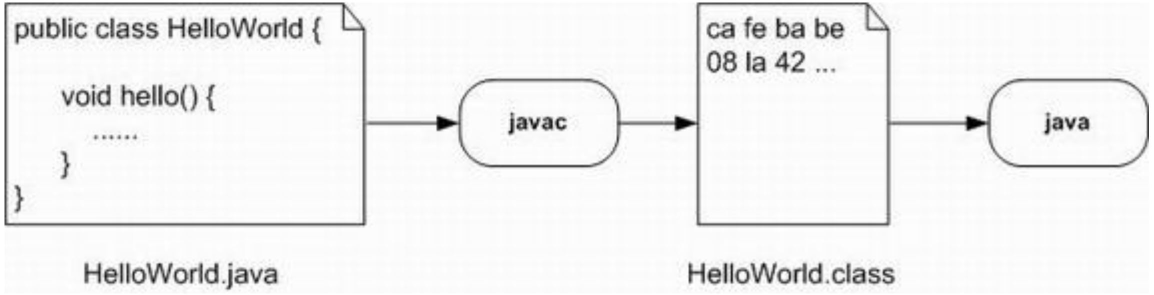


```
<filter>
  <filter-name>security</filter-name>
  <filter-class>com.vipshop.security.SecFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>security</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

AOP

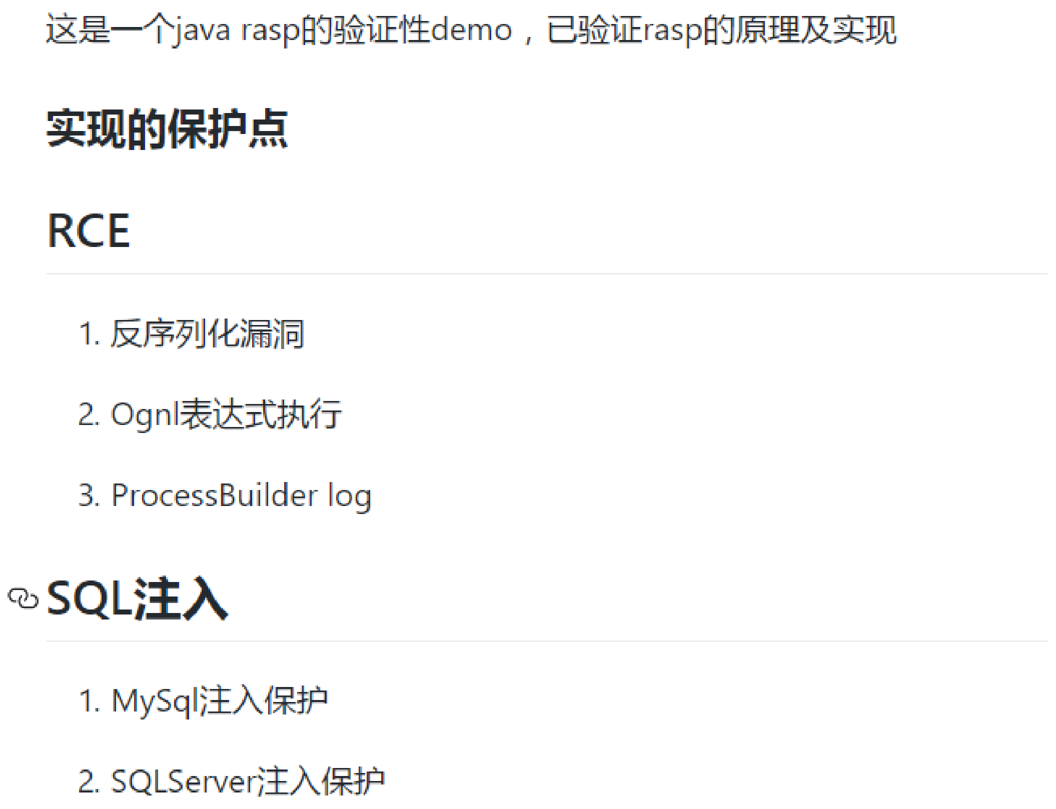
- ~~编译期（静态代理）~~
- **字节码加载前（**Instrumentation**）**
- ~~字节码后（动态代理）~~

- 开发难度高
- 漏洞覆盖面广
- 对性能影响小
- 推广难度低



03 案例分析

Java Open Rasp



这是一个java rasp的验证性demo，已验证rasp的原理及实现

实现的保护点

RCE

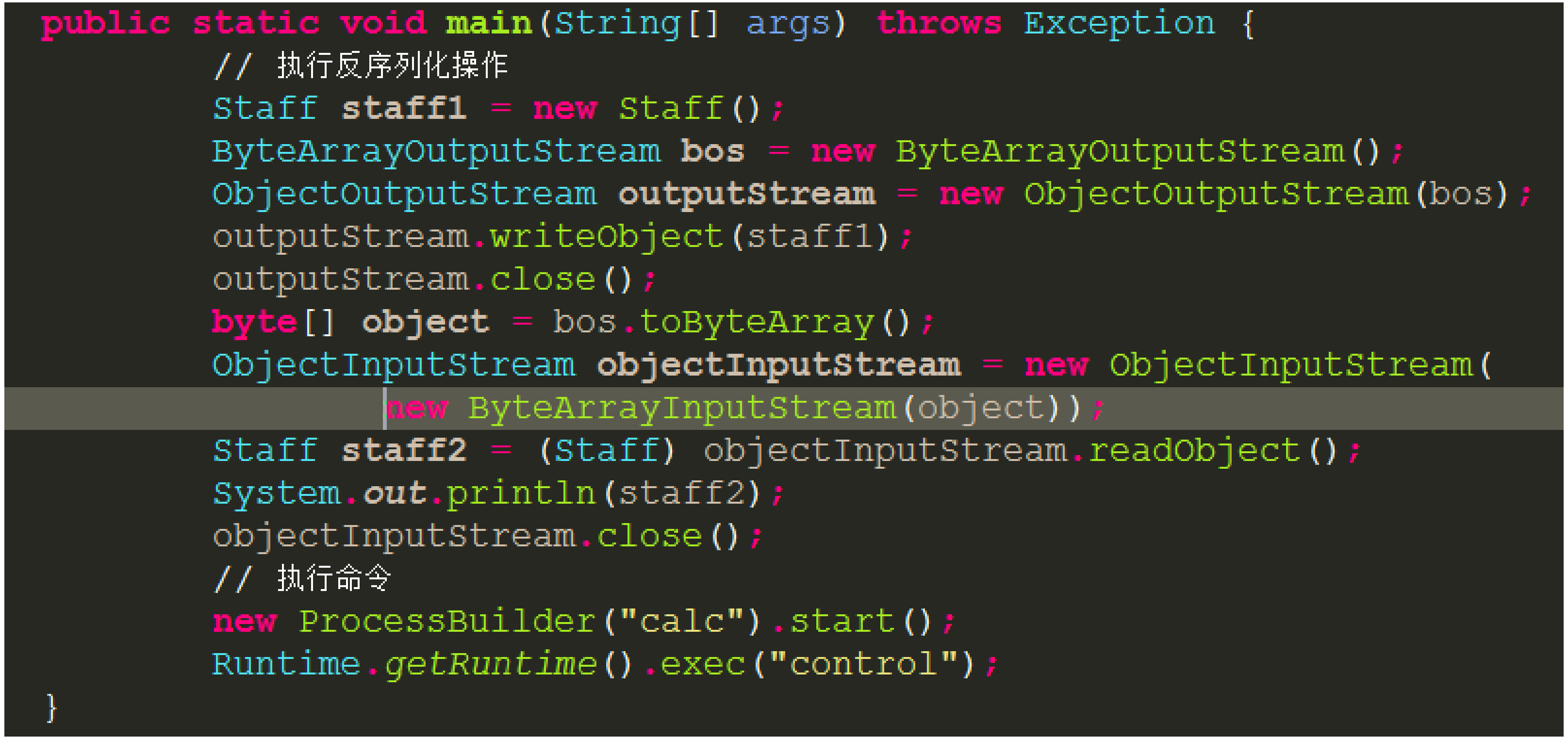
1. 反序列化漏洞
2. Ognl表达式执行
3. ProcessBuilder log

SQL注入

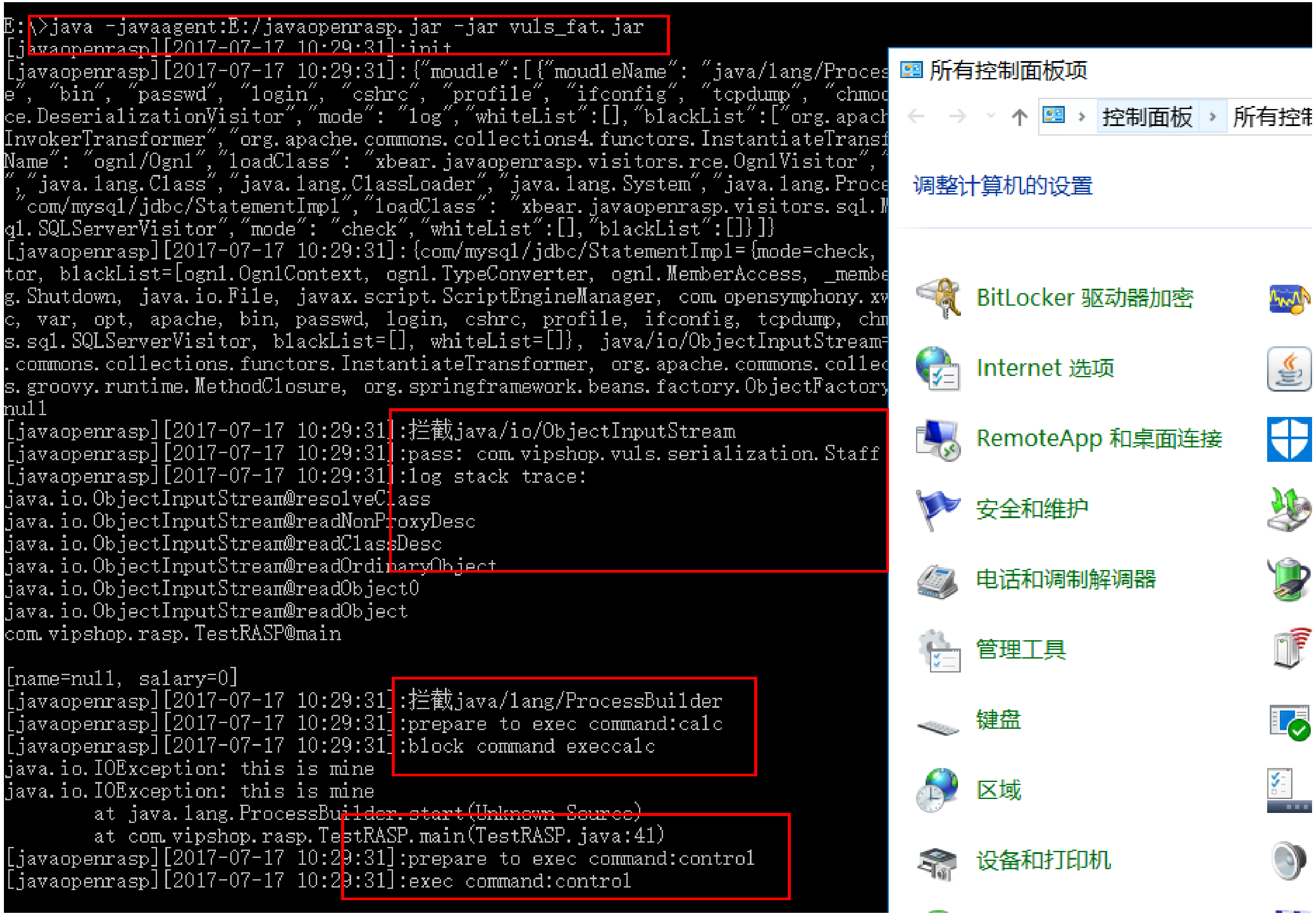
1. MySql注入保护
2. SQLServer注入保护

- Instrumentation
- ASM

使用示例



```java
public static void main(String[] args) throws Exception {
        // 执行反序列化操作
        Staff staff1 = new Staff();
        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        ObjectOutputStream outputStream = new ObjectOutputStream(bos);
        outputStream.writeObject(staff1);
        outputStream.close();
        byte[] object = bos.toByteArray();
        ObjectInputStream objectInputStream = new ObjectInputStream(
                new ByteArrayInputStream(object));
        Staff staff2 = (Staff) objectInputStream.readObject();
        System.out.println(staff2);
        objectInputStream.close();
        // 执行命令
        new ProcessBuilder("calc").start();
        Runtime.getRuntime().exec("control");
}
```



```
E:\>java -javaagent:E:/javaopenrasp.jar -jar vuls_fat.jar
[javaopenrasp][2017-07-17 10:29:31]:init
[javaopenrasp][2017-07-17 10:29:31]:{"moudle":[{"moudleName": "java/lang/Proces
e", "bin", "passwd", "login", "cshrc", "profile", "ifconfig", "tcpdump", "chmod
ce.DeserializationVisitor","mode": "log","whiteList":[],"blackList":["org.apach
InvokerTransformer","org.apache.commons.collections4.functors.InstantiateTransf
Name": "ognl/Ognl","loadClass": "xbear.javaopenrasp.visitors.rce.OgnlVisitor","
","java.lang.Class","java.lang.ClassLoader","java.lang.System","java.lang.Proce
 "com/mysql/jdbc/StatementImpl","loadClass": "xbear.javaopenrasp.visitors.sql.M
ql.SQLServerVisitor","mode": "check","whiteList":[],"blackList":[]}]}
[javaopenrasp][2017-07-17 10:29:31]:{com/mysql/jdbc/StatementImpl={mode=check,
tor, blackList=[ognl.OgnlContext, ognl.TypeConverter, ognl.MemberAccess, _membe
g.Shutdown, java.io.File, javax.script.ScriptEngineManager, com.opensymphony.xw
c, var, opt, apache, bin, passwd, login, cshrc, profile, ifconfig, tcpdump, chm
s.sql.SQLServerVisitor, blackList=[], whiteList=[]}, java/io/ObjectInputStream=
.commons.collections.functors.InstantiateTransformer, org.apache.commons.collec
s.groovy.runtime.MethodClosure, org.springframework.beans.factory.ObjectFactory
null
[javaopenrasp][2017-07-17 10:29:31]:拦截java/io/ObjectInputStream
[javaopenrasp][2017-07-17 10:29:31]:pass: com.vipshop.vuls.serialization.Staff
[javaopenrasp][2017-07-17 10:29:31]:log stack trace:
java.io.ObjectInputStream@resolveClass
java.io.ObjectInputStream@readNonProxyDesc
java.io.ObjectInputStream@readClassDesc
java.io.ObjectInputStream@readOrdinaryObject
java.io.ObjectInputStream@readObject0
java.io.ObjectInputStream@readObject
com.vipshop.rasp.TestRASP@main

[name=null, salary=0]
[javaopenrasp][2017-07-17 10:29:31]:拦截java/lang/ProcessBuilder
[javaopenrasp][2017-07-17 10:29:31]:prepare to exec command:calc
[javaopenrasp][2017-07-17 10:29:31]:block command execcalc
java.io.IOException: this is mine
java.io.IOException: this is mine
        at java.lang.ProcessBuilder.start(Unknown Source)
        at com.vipshop.rasp.TestRASP.main(TestRASP.java:41)
[javaopenrasp][2017-07-17 10:29:31]:prepare to exec command:control
[javaopenrasp][2017-07-17 10:29:31]:exec command:control
```

背景知识

Instrumentation

- 使用 Instrumentation，开发者可以构建一个独立于应用程序的代理程序（Agent），用来监测和协助运行在 JVM 上的程序，甚至能够替换和修改某些类的定义。

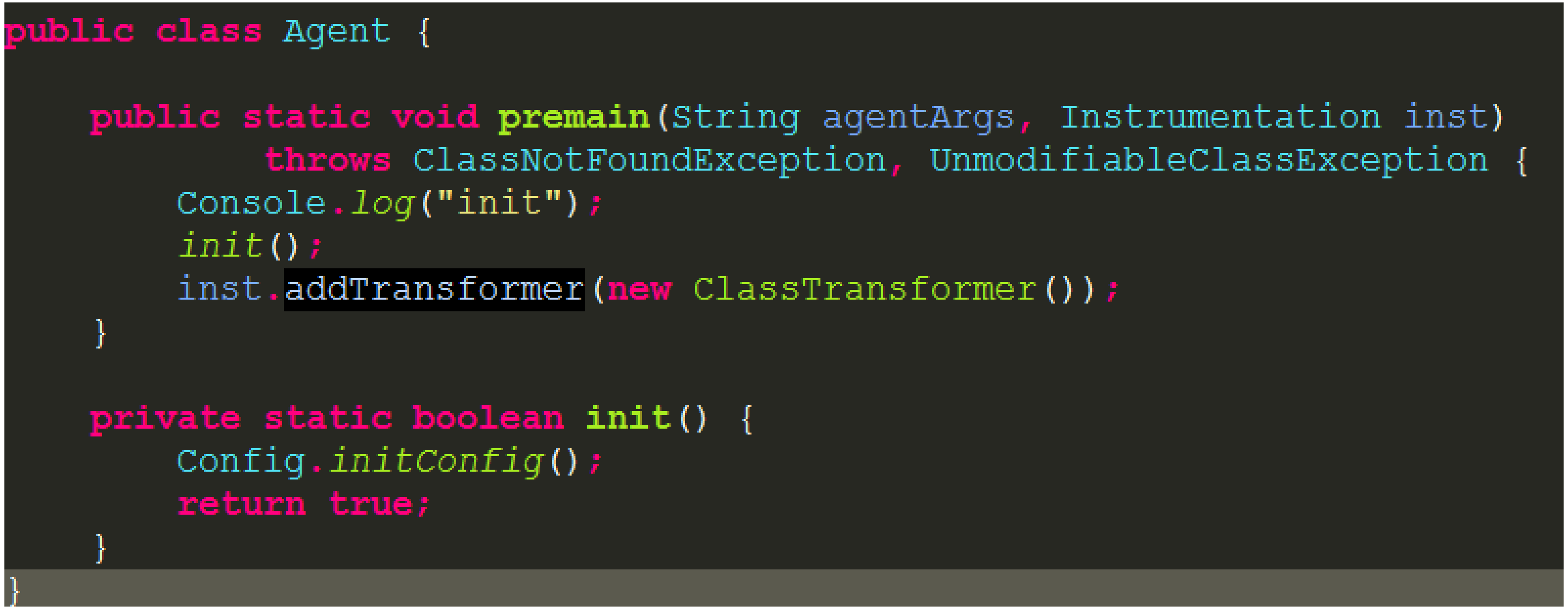
- 有了这样的功能，开发者就可以实现更为灵活的运行时虚拟机监控和 Java 类操作了，这样的特性实际上提供了一种虚拟机级别支持的 AOP 实现方式，使得开发者无需对 JDK 做任何升级和改动，就可以实现某些 AOP 的功能了。

- 编译后，将"-javaagent:/path/agent.jar"添加至JVM的启动参数

ASM

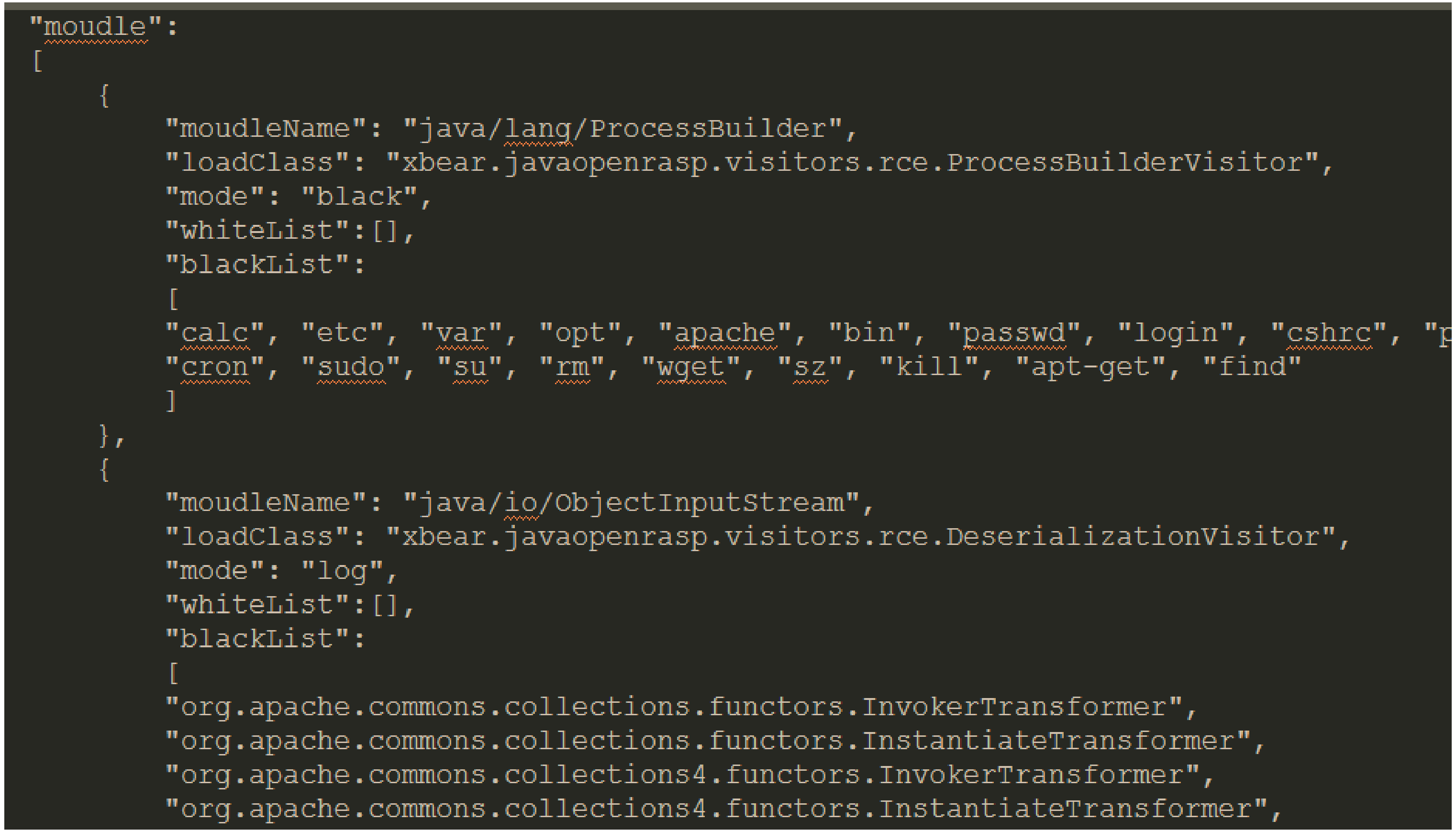
- ASM是一个通用的Java字节码操作和分析框架。
- 它可以用于直接以二进制形式修改现有类或动态生成类。
- 提供通用的转换和分析算法可以轻松组合定制复杂的转换和代码分析工具。
- ASM提供与其他字节码框架类似的功能，但它的重点是简单的使用和性能。
- 因为它的设计和实现尽可能小且尽可能快，所以它使它在动态系统中非常有吸引力。

代码分析



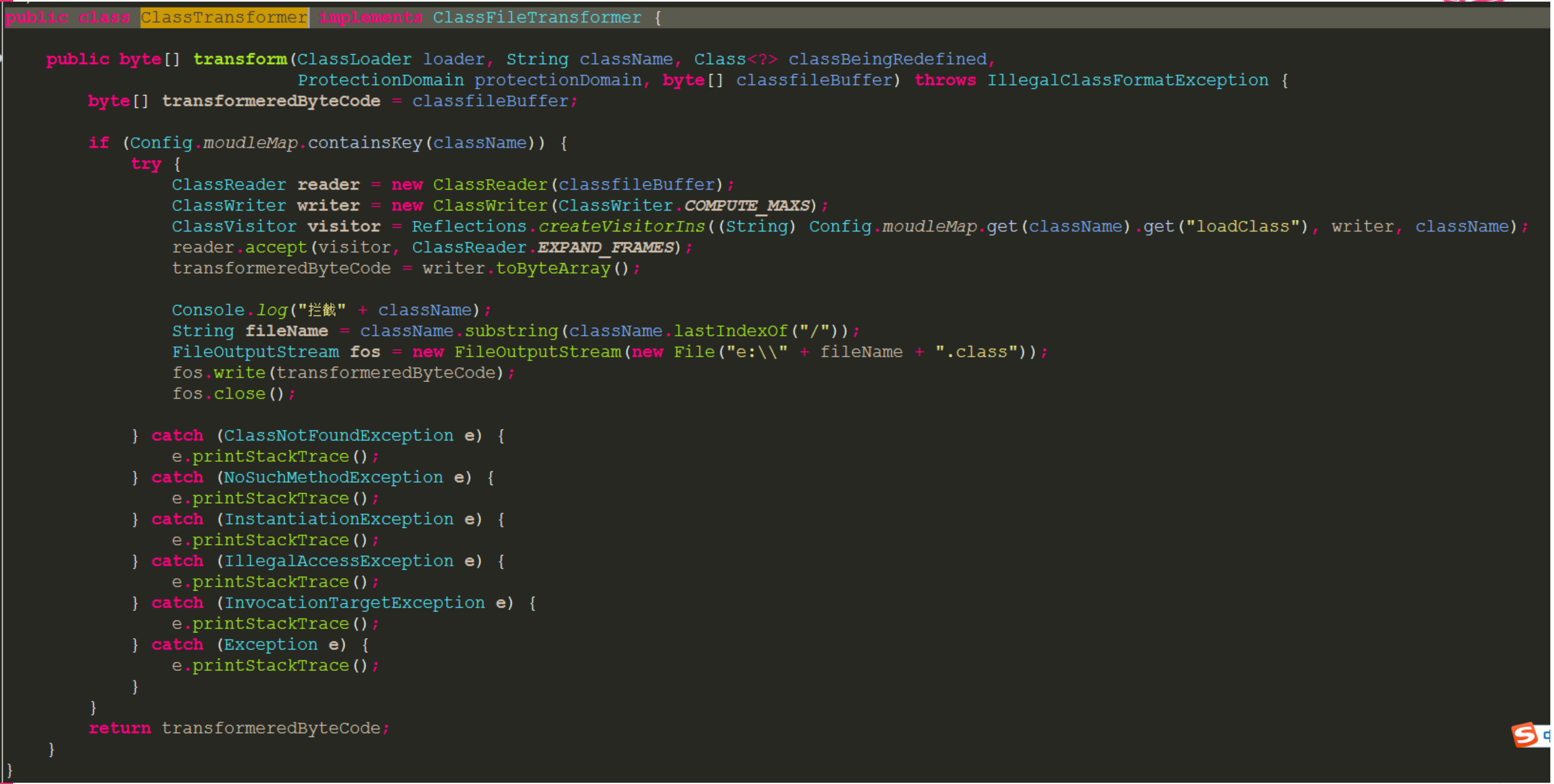
```java
public class Agent {

    public static void premain(String agentArgs, Instrumentation inst)
            throws ClassNotFoundException, UnmodifiableClassException {
        Console.log("init");
        init();
        inst.addTransformer(new ClassTransformer());
    }

    private static boolean init() {
        Config.initConfig();
        return true;
    }
}
```

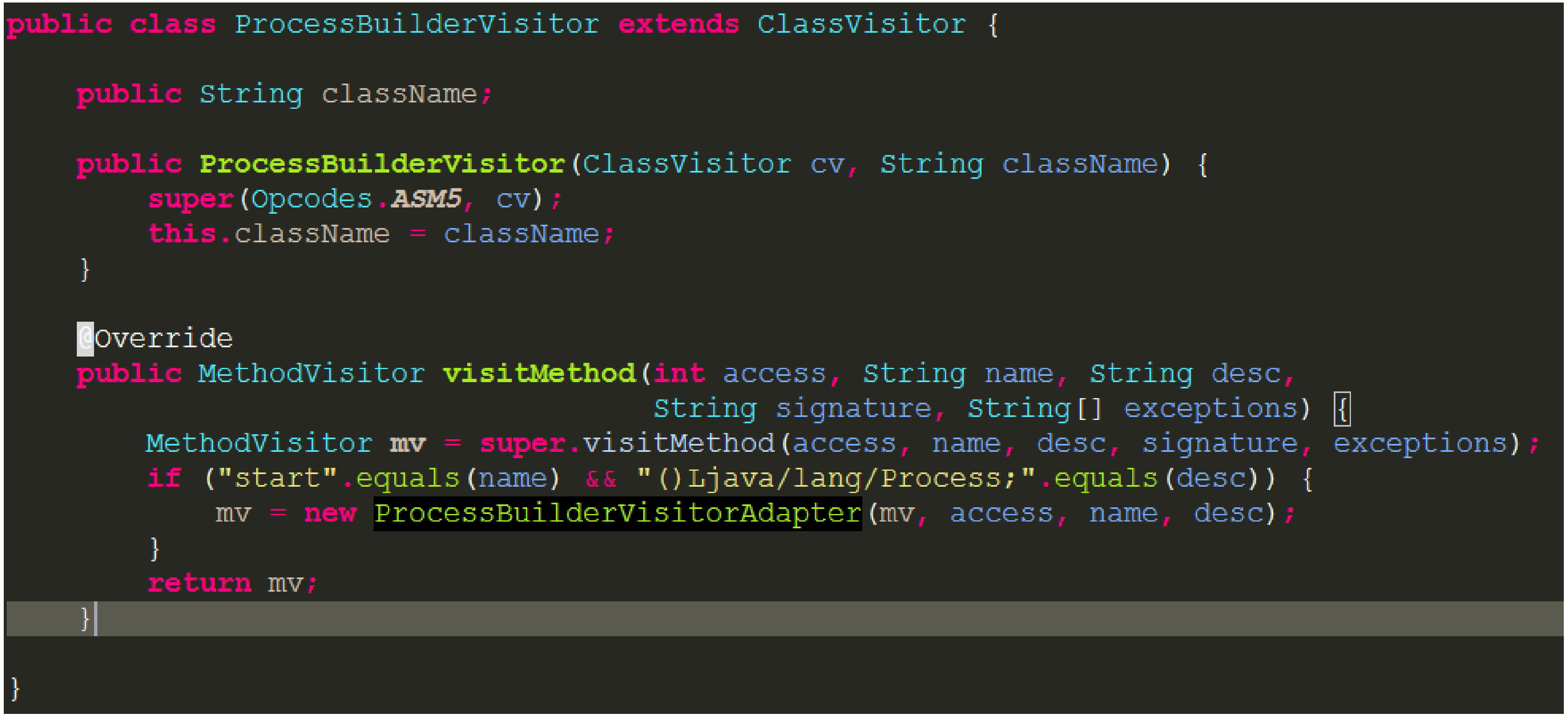
- 使用 Instrumentation



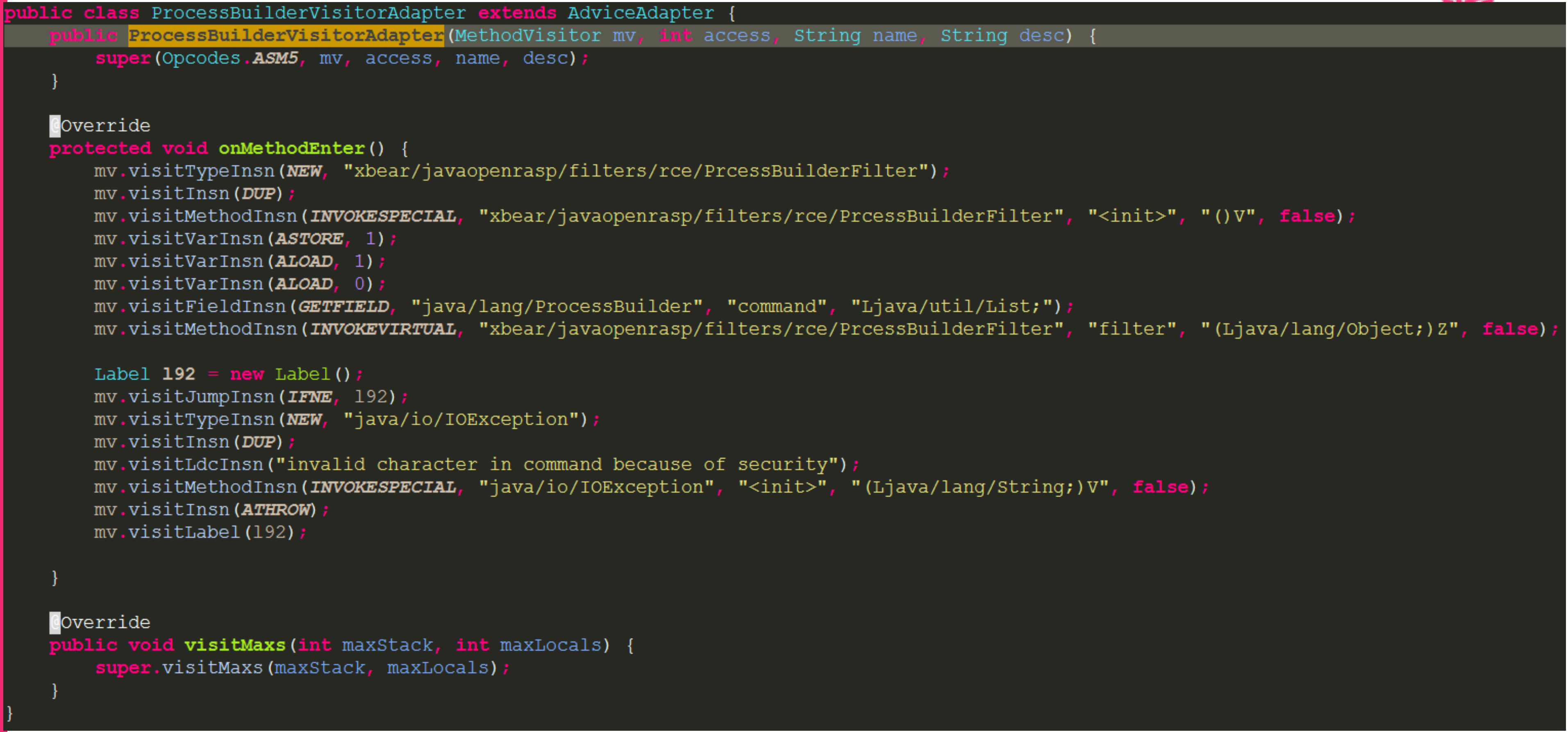
```
"moudle":
[
    {
        "moudleName": "java/lang/ProcessBuilder",
        "loadClass": "xbear.javaopenrasp.visitors.rce.ProcessBuilderVisitor",
        "mode": "black",
        "whiteList":[],
        "blackList":
        [
        "calc", "etc", "var", "opt", "apache", "bin", "passwd", "login", "cshrc", "p
        "cron", "sudo", "su", "rm", "wget", "sz", "kill", "apt-get", "find"
        ]
    },
    {
        "moudleName": "java/io/ObjectInputStream",
        "loadClass": "xbear.javaopenrasp.visitors.rce.DeserializationVisitor",
        "mode": "log",
        "whiteList":[],
        "blackList":
        [
        "org.apache.commons.collections.functors.InvokerTransformer",
        "org.apache.commons.collections.functors.InstantiateTransformer",
        "org.apache.commons.collections4.functors.InvokerTransformer",
        "org.apache.commons.collections4.functors.InstantiateTransformer",
```

Config文件

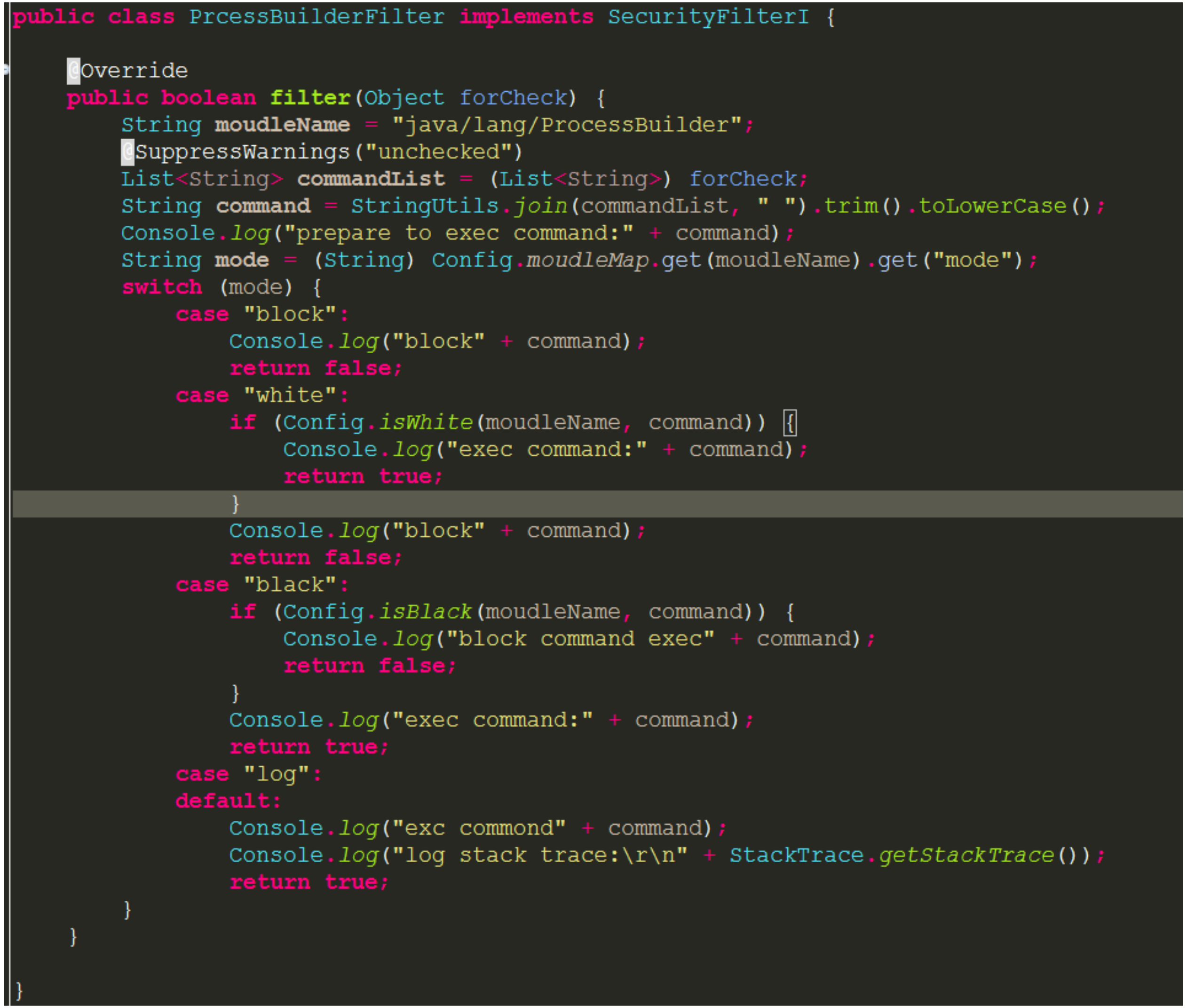

```java
public class ClassTransformer implements ClassFileTransformer {

    public byte[] transform(ClassLoader loader, String className, Class<?> classBeingRedefined,
                            ProtectionDomain protectionDomain, byte[] classfileBuffer) throws IllegalClassFormatException {
        byte[] transformeredByteCode = classfileBuffer;

        if (Config.moudleMap.containsKey(className)) {
            try {
                ClassReader reader = new ClassReader(classfileBuffer);
                ClassWriter writer = new ClassWriter(ClassWriter.COMPUTE_MAXS);
                ClassVisitor visitor = Reflections.createVisitorIns((String) Config.moudleMap.get(className).get("loadClass"), writer, className);
                reader.accept(visitor, ClassReader.EXPAND_FRAMES);
                transformeredByteCode = writer.toByteArray();

                Console.log("拦截" + className);
                String fileName = className.substring(className.lastIndexOf("/"));
                FileOutputStream fos = new FileOutputStream(new File("e:\\" + fileName + ".class"));
                fos.write(transformeredByteCode);
                fos.close();

            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            } catch (NoSuchMethodException e) {
                e.printStackTrace();
            } catch (InstantiationException e) {
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                e.printStackTrace();
            } catch (InvocationTargetException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return transformeredByteCode;
    }
}
```



```java
public class ProcessBuilderVisitor extends ClassVisitor {

    public String className;

    public ProcessBuilderVisitor(ClassVisitor cv, String className) {
        super(Opcodes.ASM5, cv);
        this.className = className;
    }

    @Override
    public MethodVisitor visitMethod(int access, String name, String desc,
                                     String signature, String[] exceptions) {
        MethodVisitor mv = super.visitMethod(access, name, desc, signature, exceptions);
        if ("start".equals(name) && "()Ljava/lang/Process;".equals(desc)) {
            mv = new ProcessBuilderVisitorAdapter(mv, access, name, desc);
        }
        return mv;
    }

}
```

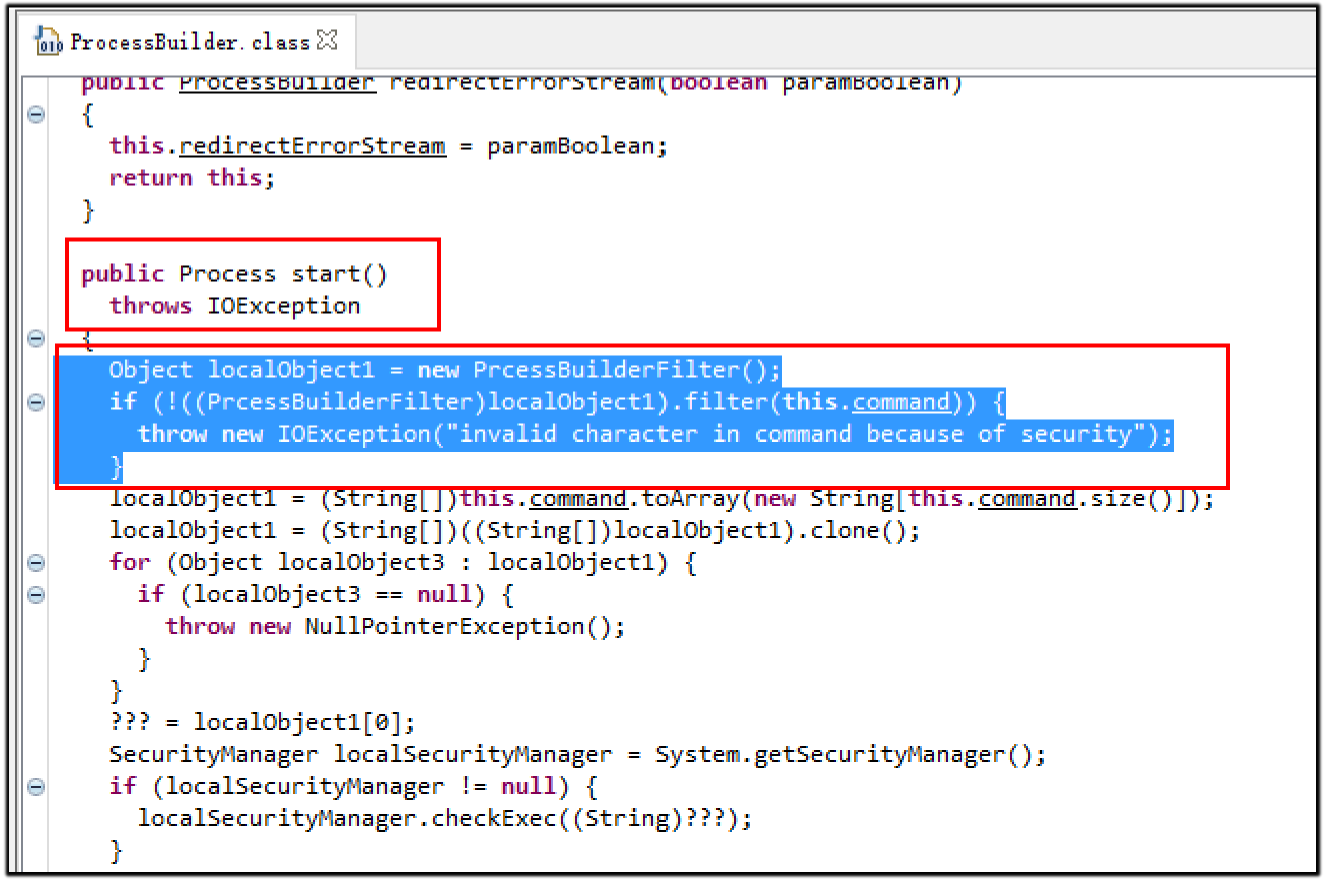


```java
public class ProcessBuilderVisitorAdapter extends AdviceAdapter {
    public ProcessBuilderVisitorAdapter(MethodVisitor mv, int access, String name, String desc) {
        super(Opcodes.ASM5, mv, access, name, desc);
    }

    @Override
    protected void onMethodEnter() {
        mv.visitTypeInsn(NEW, "xbear/javaopenrasp/filters/rce/PrcessBuilderFilter");
        mv.visitInsn(DUP);
        mv.visitMethodInsn(INVOKESPECIAL, "xbear/javaopenrasp/filters/rce/PrcessBuilderFilter", "<init>", "()V", false);
        mv.visitVarInsn(ASTORE, 1);
        mv.visitVarInsn(ALOAD, 1);
        mv.visitVarInsn(ALOAD, 0);
        mv.visitFieldInsn(GETFIELD, "java/lang/ProcessBuilder", "command", "Ljava/util/List;");
        mv.visitMethodInsn(INVOKEVIRTUAL, "xbear/javaopenrasp/filters/rce/PrcessBuilderFilter", "filter", "(Ljava/lang/Object;)Z", false);

        Label l92 = new Label();
        mv.visitJumpInsn(IFNE, l92);
        mv.visitTypeInsn(NEW, "java/io/IOException");
        mv.visitInsn(DUP);
        mv.visitLdcInsn("invalid character in command because of security");
        mv.visitMethodInsn(INVOKESPECIAL, "java/io/IOException", "<init>", "(Ljava/lang/String;)V", false);
        mv.visitInsn(ATHROW);
        mv.visitLabel(l92);

    }

    @Override
    public void visitMaxs(int maxStack, int maxLocals) {
        super.visitMaxs(maxStack, maxLocals);
    }
}
```



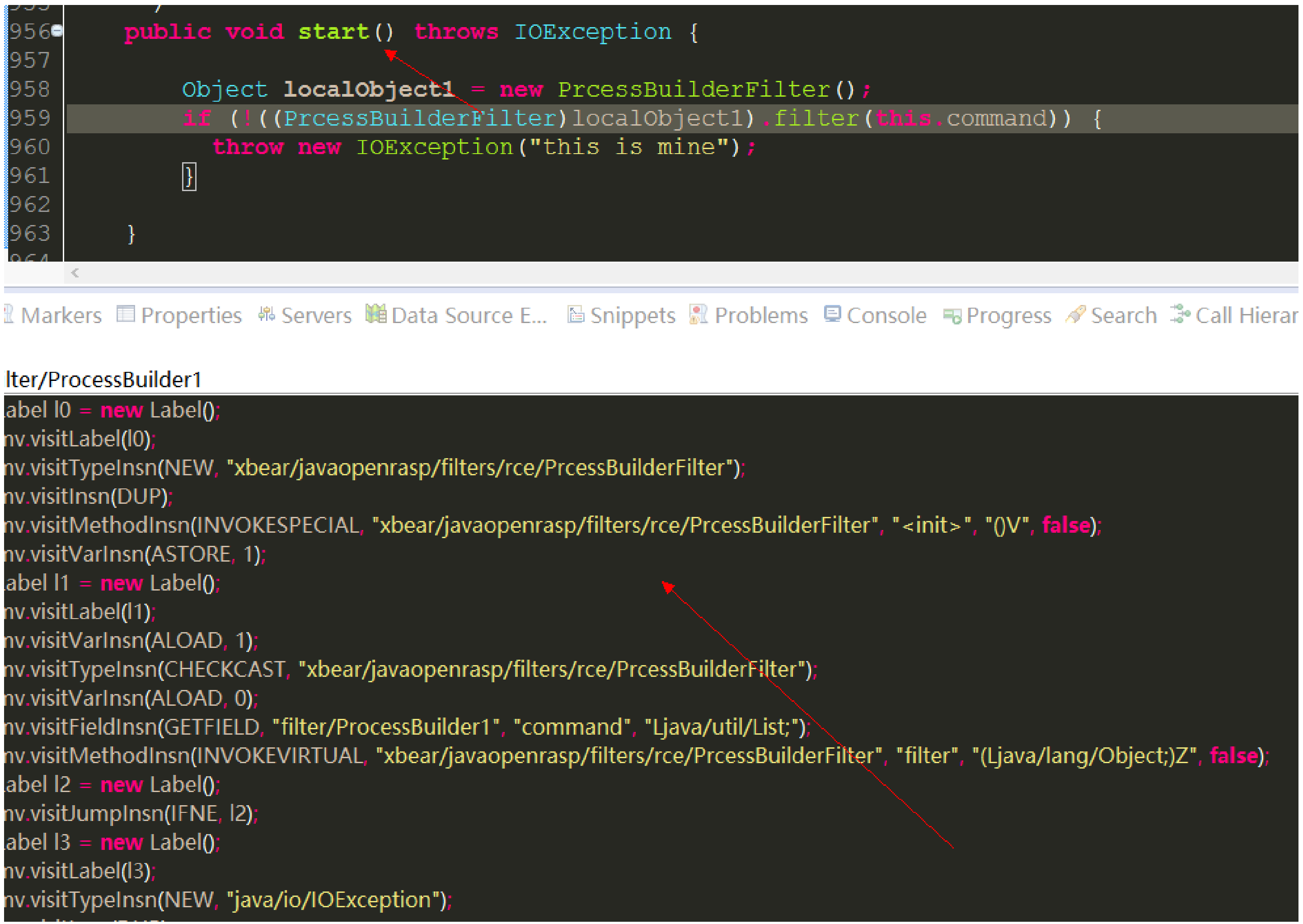
```
public class PrcessBuilderFilter implements SecurityFilterI {

    @Override
    public boolean filter(Object forCheck) {
        String moudleName = "java/lang/ProcessBuilder";
        @SuppressWarnings("unchecked")
        List<String> commandList = (List<String>) forCheck;
        String command = StringUtils.join(commandList, " ").trim().toLowerCase();
        Console.log("prepare to exec command:" + command);
        String mode = (String) Config.moudleMap.get(moudleName).get("mode");
        switch (mode) {
            case "block":
                Console.log("block" + command);
                return false;
            case "white":
                if (Config.isWhite(moudleName, command)) {
                    Console.log("exec command:" + command);
                    return true;
                }
                Console.log("block" + command);
                return false;
            case "black":
                if (Config.isBlack(moudleName, command)) {
                    Console.log("block command exec" + command);
                    return false;
                }
                Console.log("exec command:" + command);
                return true;
            case "log":
            default:
                Console.log("exc commond" + command);
                Console.log("log stack trace:\r\n" + StackTrace.getStackTrace());
                return true;
        }
    }

}
```

结果分析


ProcessBuilder.class

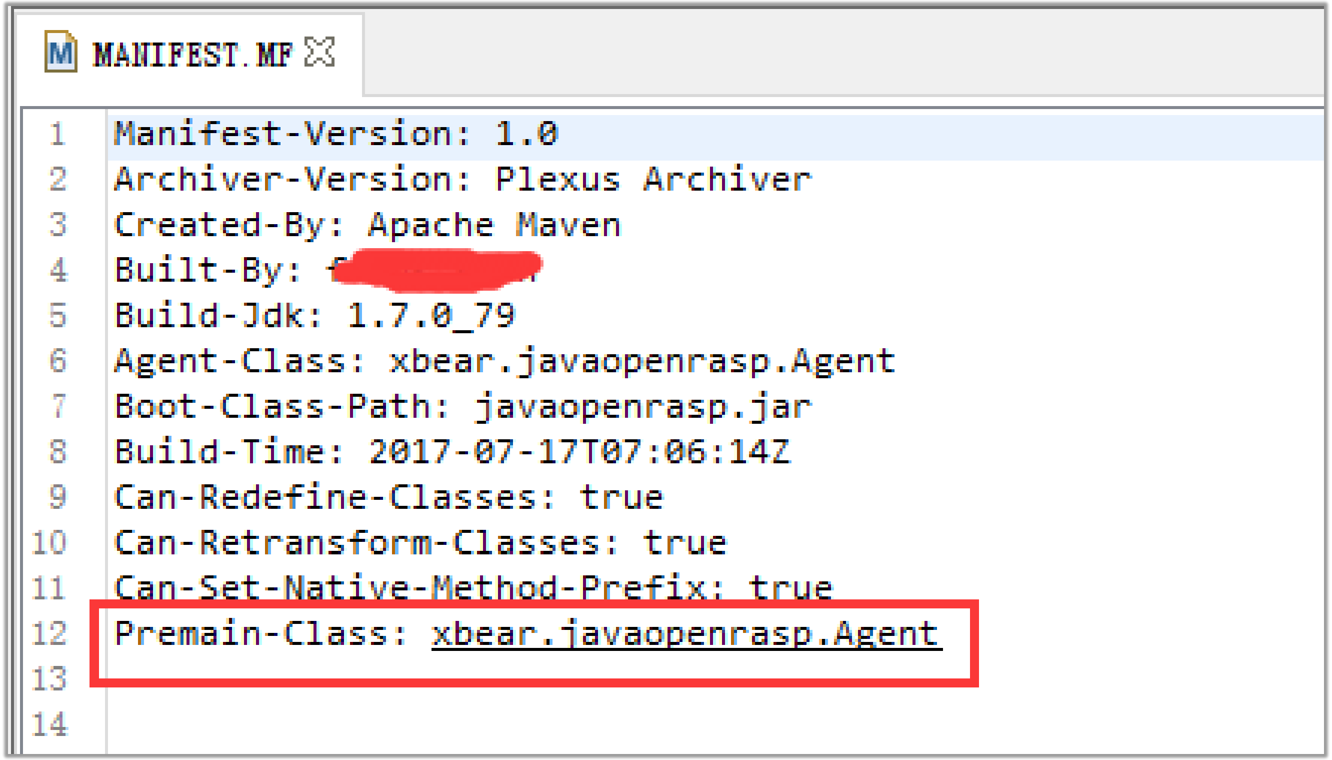
```
  public ProcessBuilder redirectErrorStream(boolean paramBoolean)
  {
    this.redirectErrorStream = paramBoolean;
    return this;
  }

  public Process start()
    throws IOException
  {
    Object localObject1 = new PrcessBuilderFilter();
    if (!((PrcessBuilderFilter)localObject1).filter(this.command)) {
      throw new IOException("invalid character in command because of security");
    }
    localObject1 = (String[])this.command.toArray(new String[this.command.size()]);
    localObject1 = (String[])((String[])localObject1).clone();
    for (Object localObject3 : localObject1) {
      if (localObject3 == null) {
        throw new NullPointerException();
      }
    }
    ??? = localObject1[0];
    SecurityManager localSecurityManager = System.getSecurityManager();
    if (localSecurityManager != null) {
      localSecurityManager.checkExec((String)???);
    }
```

开发技巧


```
    public void start() throws IOException {

        Object localObject1 = new PrcessBuilderFilter();
        if (!((PrcessBuilderFilter)localObject1).filter(this.command)) {
          throw new IOException("this is mine");
        }

    }
```

lter/ProcessBuilder1

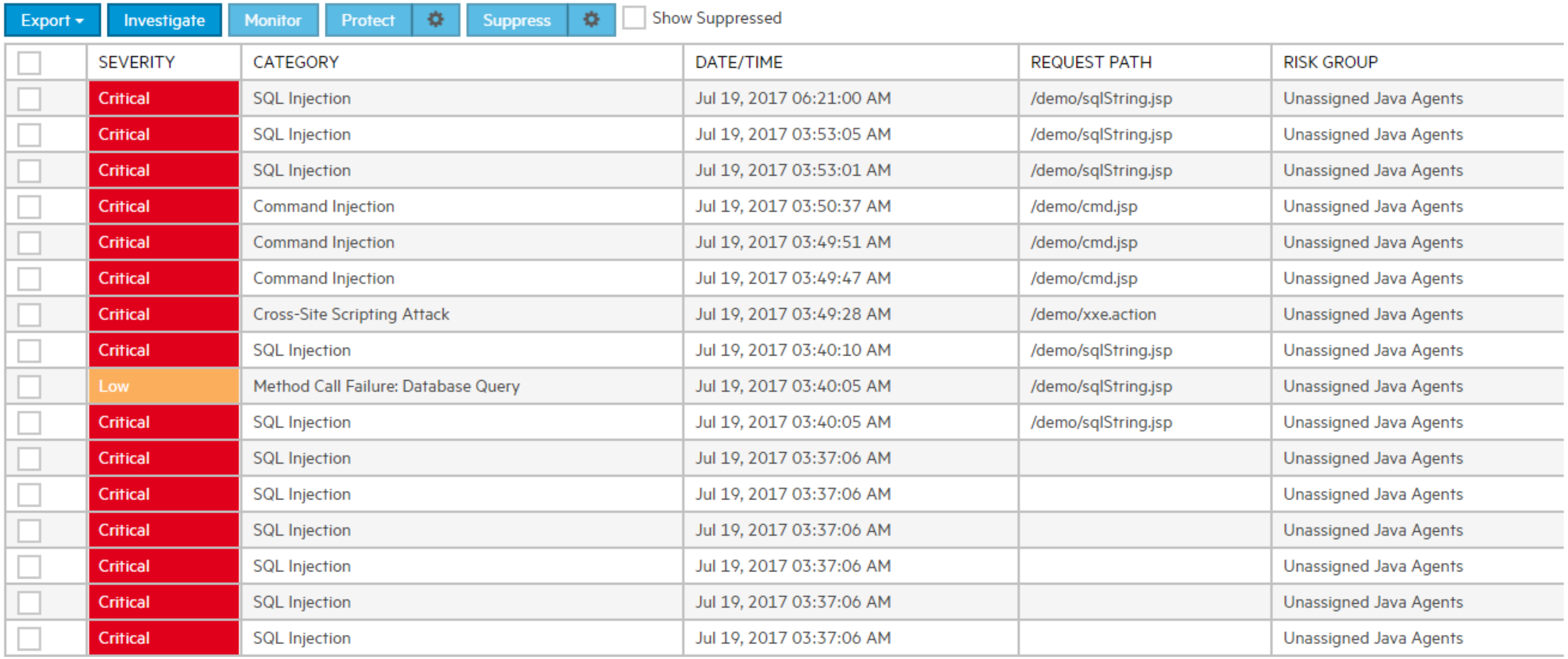
```
abel l0 = new Label();
nv.visitLabel(l0);
nv.visitTypeInsn(NEW, "xbear/javaopenrasp/filters/rce/PrcessBuilderFilter");
nv.visitInsn(DUP);
nv.visitMethodInsn(INVOKESPECIAL, "xbear/javaopenrasp/filters/rce/PrcessBuilderFilter", "<init>", "()V", false);
nv.visitVarInsn(ASTORE, 1);
abel l1 = new Label();
nv.visitLabel(l1);
nv.visitVarInsn(ALOAD, 1);
nv.visitTypeInsn(CHECKCAST, "xbear/javaopenrasp/filters/rce/PrcessBuilderFilter");
nv.visitVarInsn(ALOAD, 0);
nv.visitFieldInsn(GETFIELD, "filter/ProcessBuilder1", "command", "Ljava/util/List;");
nv.visitMethodInsn(INVOKEVIRTUAL, "xbear/javaopenrasp/filters/rce/PrcessBuilderFilter", "filter", "(Ljava/lang/Object;)Z", false);
abel l2 = new Label();
nv.visitJumpInsn(IFNE, l2);
abel l3 = new Label();
nv.visitLabel(l3);
nv.visitTypeInsn(NEW, "java/io/IOException");
```


MANIFEST.MF

```
Manifest-Version: 1.0
Archiver-Version: Plexus Archiver
Created-By: Apache Maven
Built-By: [redacted]
Build-Jdk: 1.7.0_79
Agent-Class: xbear.javaopenrasp.Agent
Boot-Class-Path: javaopenrasp.jar
Build-Time: 2017-07-17T07:06:14Z
Can-Redefine-Classes: true
Can-Retransform-Classes: true
Can-Set-Native-Method-Prefix: true
Premain-Class: xbear.javaopenrasp.Agent
```

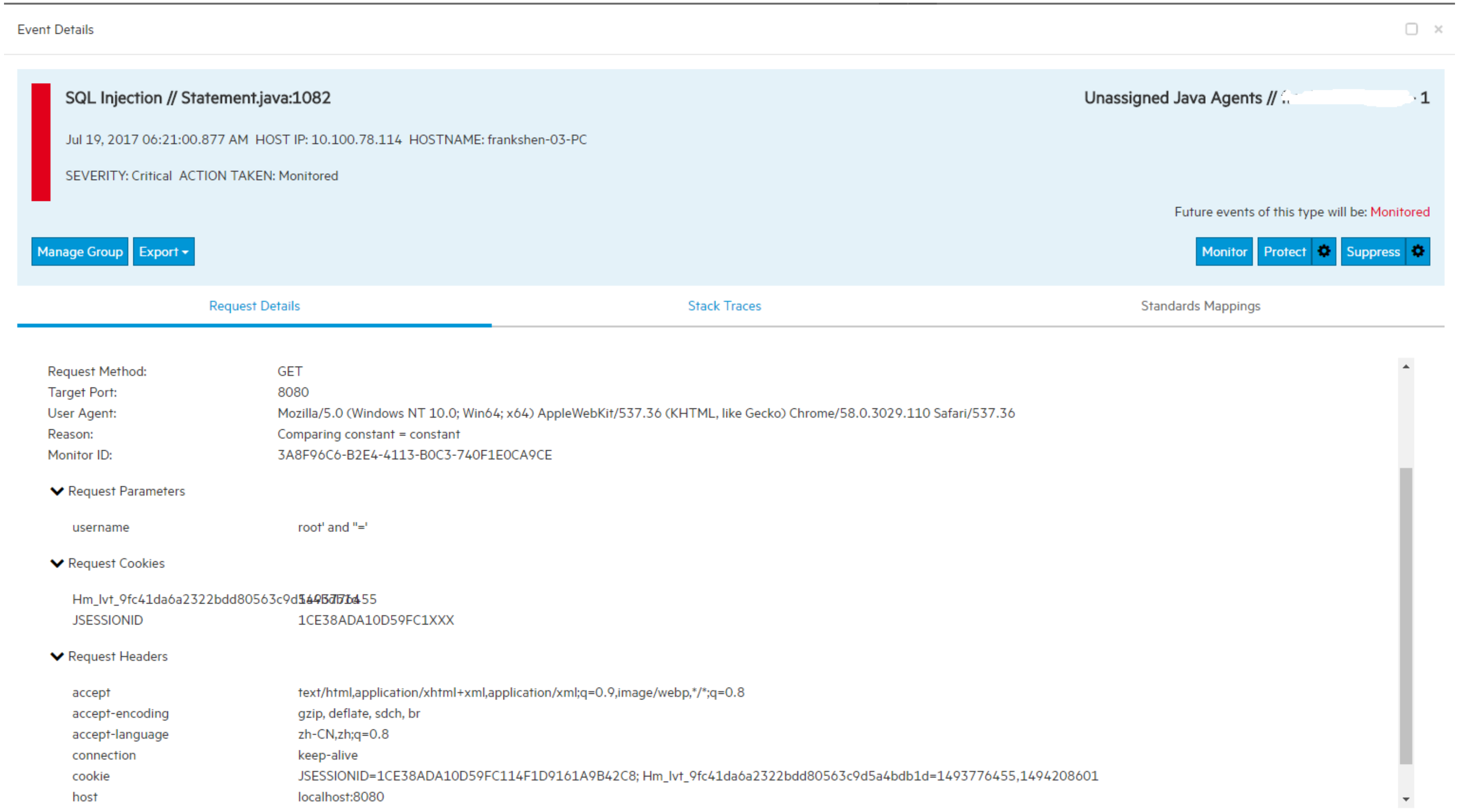
适用场景

- 漏洞监测
- 灰盒测试



	SEVERITY	CATEGORY	DATE/TIME	REQUEST PATH	RISK GROUP
☐	Critical	SQL Injection	Jul 19, 2017 06:21:00 AM	/demo/sqlString.jsp	Unassigned Java Agents
☐	Critical	SQL Injection	Jul 19, 2017 03:53:05 AM	/demo/sqlString.jsp	Unassigned Java Agents
☐	Critical	SQL Injection	Jul 19, 2017 03:53:01 AM	/demo/sqlString.jsp	Unassigned Java Agents
☐	Critical	Command Injection	Jul 19, 2017 03:50:37 AM	/demo/cmd.jsp	Unassigned Java Agents
☐	Critical	Command Injection	Jul 19, 2017 03:49:51 AM	/demo/cmd.jsp	Unassigned Java Agents
☐	Critical	Command Injection	Jul 19, 2017 03:49:47 AM	/demo/cmd.jsp	Unassigned Java Agents
☐	Critical	Cross-Site Scripting Attack	Jul 19, 2017 03:49:28 AM	/demo/xxe.action	Unassigned Java Agents
☐	Critical	SQL Injection	Jul 19, 2017 03:40:10 AM	/demo/sqlString.jsp	Unassigned Java Agents
☐	Low	Method Call Failure: Database Query	Jul 19, 2017 03:40:05 AM	/demo/sqlString.jsp	Unassigned Java Agents
☐	Critical	SQL Injection	Jul 19, 2017 03:40:05 AM	/demo/sqlString.jsp	Unassigned Java Agents
☐	Critical	SQL Injection	Jul 19, 2017 03:37:06 AM		Unassigned Java Agents
☐	Critical	SQL Injection	Jul 19, 2017 03:37:06 AM		Unassigned Java Agents
☐	Critical	SQL Injection	Jul 19, 2017 03:37:06 AM		Unassigned Java Agents
☐	Critical	SQL Injection	Jul 19, 2017 03:37:06 AM		Unassigned Java Agents
☐	Critical	SQL Injection	Jul 19, 2017 03:37:06 AM		Unassigned Java Agents
☐	Critical	SQL Injection	Jul 19, 2017 03:37:06 AM		Unassigned Java Agents

VSRC



Event Details

SQL Injection // Statement.java:1082

Unassigned Java Agents // [redacted] 1

Jul 19, 2017 06:21:00.877 AM HOST IP: 10.100.78.114 HOSTNAME: frankshen-03-PC

SEVERITY: Critical ACTION TAKEN: Monitored

Future events of this type will be: Monitored

Manage Group | Export | Monitor | Protect | Suppress

Request Details | Stack Traces | Standards Mappings

Request Method:	GET
Target Port:	8080
User Agent:	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36
Reason:	Comparing constant = constant
Monitor ID:	3A8F96C6-B2E4-4113-B0C3-740F1E0CA9CE

Request Parameters

username	root' and "=''

Request Cookies

Hm_lvt_9fc41da6a2322bdd80563c9d5a4bdb1d	1493776455
JSESSIONID	1CE38ADA10D59FC1XXX

Request Headers

accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
accept-encoding	gzip, deflate, sdch, br
accept-language	zh-CN,zh;q=0.8
connection	keep-alive
cookie	JSESSIONID=1CE38ADA10D59FC114F1D9161A9B42C8; Hm_lvt_9fc41da6a2322bdd80563c9d5a4bdb1d=1493776455,1494208601
host	localhost:8080

04 思考

优劣

优点

- 对应用开发来说无感知
- 漏洞发现准确率高
- 实时发现与阻断
- 应用场景多
- 推广容易

缺点

- 对性能有5%-10%的影响
- 开发难度高

  1、熟练使用ASM框架

  2、熟悉JVM字节码指令

  3、深入分析漏洞原理，找准切点

  4、了解容器原理，找准切点
- 要不断运营规则
- 更新规则需重启容器

参考资料

- 《 Java中的RASP实现》http://mp.weixin.qq.com/s/Qk_0ZxlWqAn2fAn8m2kObQ
- 《什么是实时应用程序自我保护（RASP）？》https://segmentfault.com/a/1190000004160109
- 《Rasp技术介绍与实现》http://www.jianshu.com/p/53b50edb4a04
- 《聊聊最近挺热的RASP技术》https://mp.weixin.qq.com/s/lhBdm3_jHZCWK5xVuvg2RQ
- https://github.com/xbeark/javaopenrasp
- http://asm.ow2.org/
- 《AOP 的利器：ASM 3.0 介绍》https://www.ibm.com/developerworks/cn/java/j-lo-asm30/
- 《Instrumentation 新功能》https://www.ibm.com/developerworks/cn/java/j-lo-jse61/index.html

感谢聆听

—

THANKS!

唯品会安全应急响应中心
VIP Security Response Center (VSRC)