


CVE-2022-22972_VMware_Workspace_ONE_Access_身份认证绕过漏洞

漏洞描述

5 月 18 日, VMware 发布了一份公告 (VMSA-2022-0014), 以解决多个 VMware 产品中的两个漏洞, 其中包括 CVE-2022-22972, 该漏洞在身份认证处理时存在一定缺陷。远程攻击者可通过伪造相关请求信息来绕过身份验证, 从而获取相关应用程序的管理权限。



vmware® Multi-Cloud App Platform Cloud & Edge Infrastructure Anywhere Workspace Security Partners

VMware S. > Advisories > VMSA-2022-0014

Critical

Advisory ID: VMSA-2022-0014.1
CVSSv3 Range: 7.8-9.8
Issue Date: 2022-05-18
Updated On: 2022-05-27
CVE(s): CVE-2022-22972, CVE-2022-22973
Synopsis: VMware Workspace ONE Access, Identity Manager and vRealize Automation updates address multiple vulnerabilities.

RSS Feed
Download PDF
Download Text File
Share this page on social media
Sign up for Security Advisories
Enter your email address

相关介绍

VMware 是一家提供全球桌面到数据中心虚拟化解决方案的厂商, 其推出的产品包括我们最熟悉的 VMware Workstation, 一款桌面虚拟计算软件。此次漏洞涉及的多个产品介绍如下:

VMware Workspace ONE Access 是 VMware 公司开发的一款智能驱动型数字化工作空间平台, 通过 Workspace ONE Access 能够随时随地在任意设备上轻松、安全地交付和管理任意应用。VMware vRealize Automation 是自动化部署方案云管平台。VMware Cloud Foundation 是 VMware 公司混合云平台。vRealize Suite Lifecycle Manager 是 vRealize Suite 生命周期和内容管理平台。

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) 4.4, 4.3.x, 4.2.x, 4.1, 4.0.x

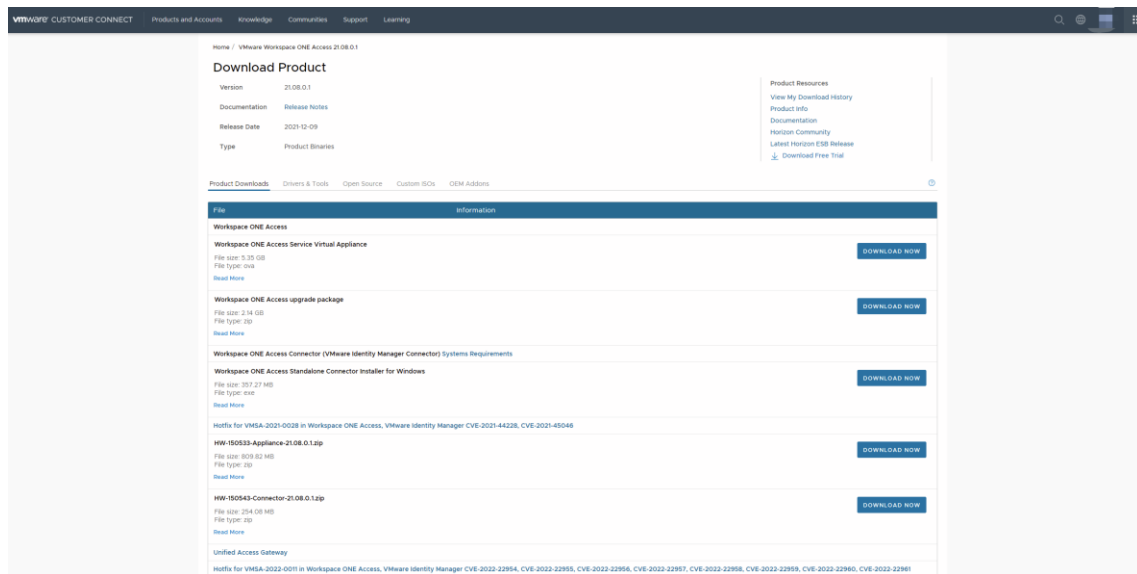
- VMware Cloud Foundation (vRA) 3.x
- vRealize Suite Lifecycle Manager(vIDM) 8.x

漏洞分析

简单回顾下 CVE-2022-22954 (VMware Workspace ONE Access SSTI 漏洞), 属于模板注入漏洞, 恶意攻击者可以利用此漏洞在未经过身份验证的情况下进行远程任意代码执行; CVE-2022-22957 (VMware Workspace ONE Access JDBC 注入漏洞), 由于相关参数完全可控, 恶意攻击者可实行 JDBC 注入, 通过写入任意文件等方式获取系统权限。此次的 CVE-2022-22972, 同样选择 VMware Workspace ONE Access 漏洞版本来进行分析, 并详细记录环境搭建和漏洞分析复现过程。

环境搭建

先从[官网](#)下载 VMware Workspace ONE Access 21.08.0.1 OVA 文件



使用 VMware Workstation 导入 OVA 文件, 配置 FQDN (主机名设置为随机域名, 不然后续配置数据库时会报错)

导入虚拟机

属性
此虚拟机的其他属性。

Application	Networking Properties
Host Name (FQDN)	vmware.xx.com
Default Gateway	192.168.43.1
Domain Name	
Domain Search Path	
DNS	192.168.43.1
IP Address	192.168.43.222
Netmask	255.255.255.0

帮助 < 上一步(B) 导入(I) 取消

导入成功后会进行初始化，完成后出现如下信息

```
VMware Workspace ONE Access - 21.08.0.1 Build 19010796
Welcome to the VMware Workspace ONE Access Appliance (192.168.43.222)

To configure the appliance and complete the installation,
point a Web browser to your site's URL at
  https://vmware.xx.com

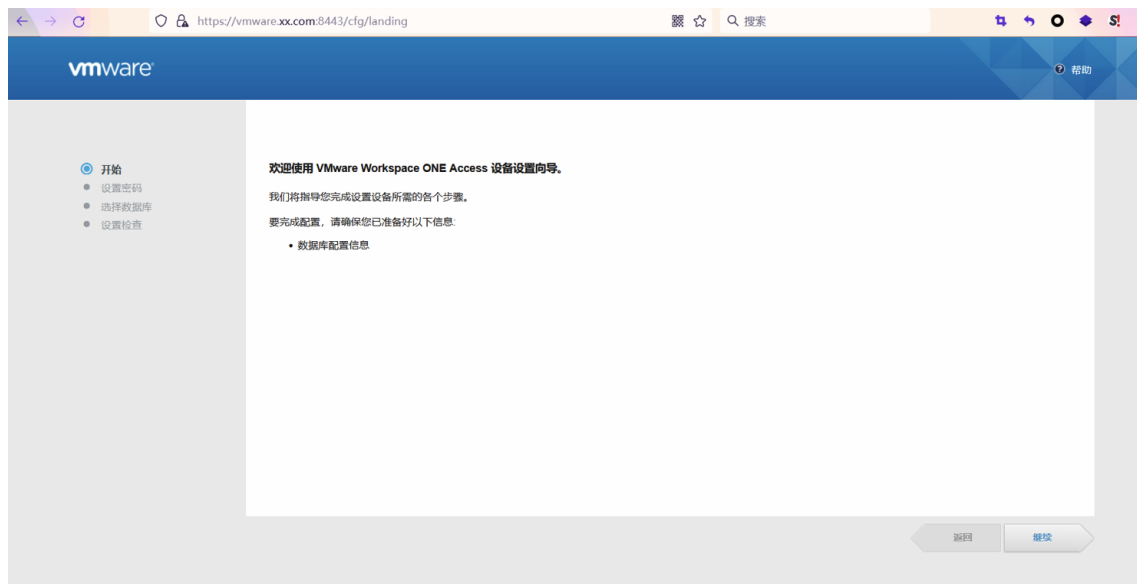
After the initial configuration,
you can find the administrative services links here
  https://vmware.xx.com:8443

When using a load balancer, use the load balancer URL for end user login.

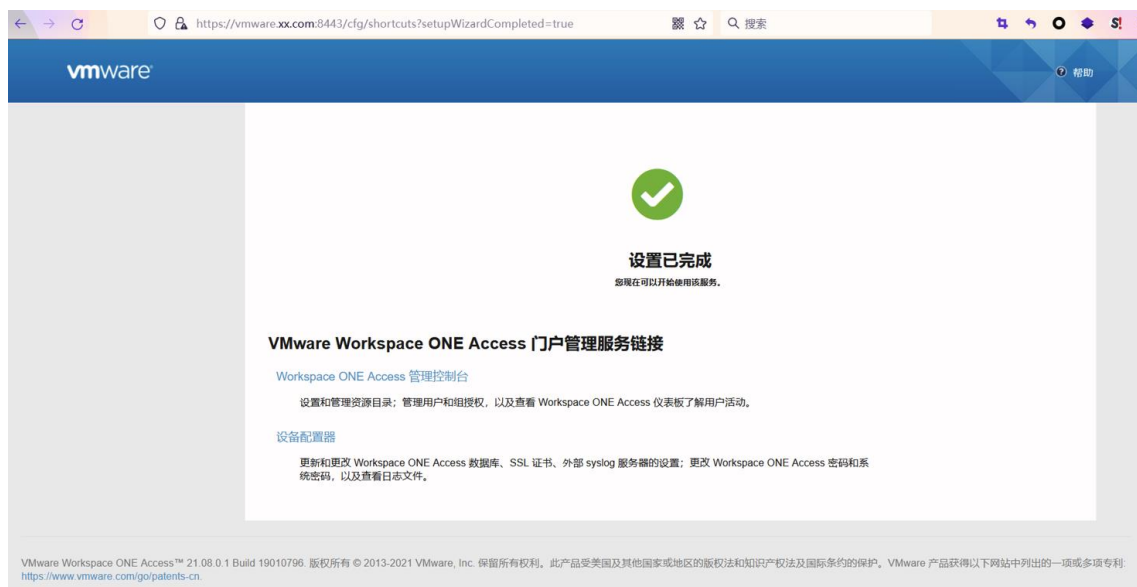
*Login
Set Timezone (Current:UTC)

Use Arrow Keys to navigate
and <ENTER> to select your choice.
```

访问 <https://<域名: 8443>>，根据提示进行账号和数据库配置

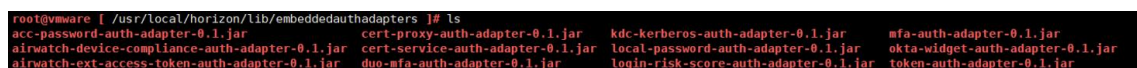


按照要求，一步一步完成配置即可

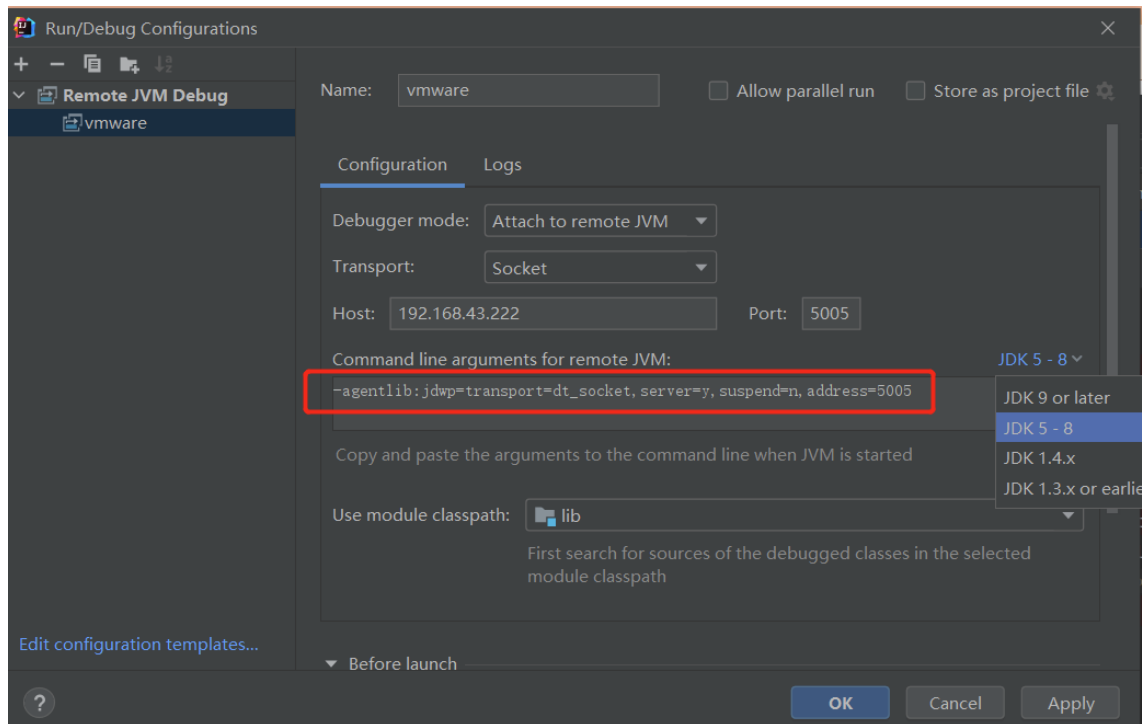


为搭建动态调试环境，需将相关源码保存到本地，并使用 IDEA 开启远程调试

需要的 lib 文件位于 `/usr/local/horizon/lib/embeddedauthadapters` 目录下



IDEA 配置远程调试



将远程调试命令写入 `/opt/vmware/horizon/workspace/bin/setenv.sh`

```

DH_KEYSIZE=1024

#No need to specify heap size with Xmx. These settings will automatically adapt to the memory size of the VM.
#Do not use the -Xms or -Xmx options in conjunction with -XX:+AggressiveHeap
. /usr/local/horizon/scripts/flag.inc
. /usr/local/horizon/scripts/manageCaCerts.inc

FIPS_JAR=$(ls -lr /usr/local/horizon/jre-endorsed/bc-fips-*.jar | head -1)
CLASSPATH=${FIPS_JAR}

if [ -f $FLAG_FIPS_MODE ]; then
    . /usr/local/horizon/scripts/tcCiphers.inc

    FIPS_MODE_OPTIONS="-Djava.security.properties=${CATALINA_BASE}/conf/ldm_fips.security \
        -Djdk.tls.namedGroups=${APPROVED_CURVES}"

    DH_KEYSIZE=2048

    #Fips disable flag is only supported for connector-only installations
    if [ ! -f $FLAG_FIPS_MODE_DISABLED ]; then
        FIPS_MODE_OPTIONS="${FIPS_MODE_OPTIONS} \
            -Dldm.fips.mode.required=true \
            -Dorg.bouncycastle.fips.approved_only=true "
    fi
fi

JVM_OPTS="-server -Djdk.tls.ephemeralDHKeySize=${DH_KEYSIZE} -XX:+AggressiveOpts \
    -Dliquibase.should.run=true \
    -Djavax.net.ssl.trustStore=${IDM_CA_KEYSTORE} \
    -Dset.rmi.server.hostname=true \
    -Dvertex.disableFileCaching=true \
    -XX:MaxMetaspaceSize=768m -XX:MetaspaceSize=768m \
    -Xss1m -Xmx3968m -Xms3968m \
    -XX:+UseParallelGC -XX:+UseParallelOldGC \
    -XX:NewRatio=3 -XX:SurvivorRatio=12 \
    -Dorg.apache.xml.security.ignoreLineBreaks=true \
    -XX:+DisableExplicitGC \
    -XX:+UseBiasedLocking -XX:-LoopUnswitching -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005"

echo Tomcat memory params are $JVM_OPTS

#Uncomment to produce garbage collector output suitable for viewing in GCViewer
#GC_LOG="-Xloggc:/opt/vmware/horizon/workspace/logs/gc.log -XX:+PrintHeapAtGC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:-HeapDumpOnOutOfMemoryError"
setenv.sh 180L, 65998
58,119

```

重启服务之后，配置 `iptables` 防火墙，允许数据包通过 `INPUT` 链和 `OUTPUT` 链

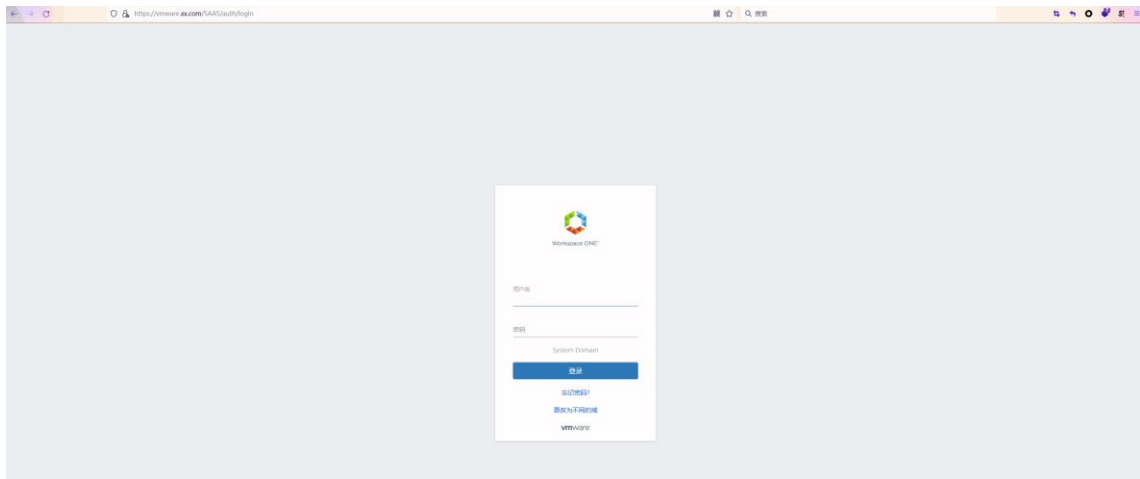
`iptables -P INPUT ACCEPT && iptables -P OUTPUT ACCEPT`

```

root@vmware [ / ]# iptables -P INPUT ACCEPT && iptables -P OUTPUT ACCEPT

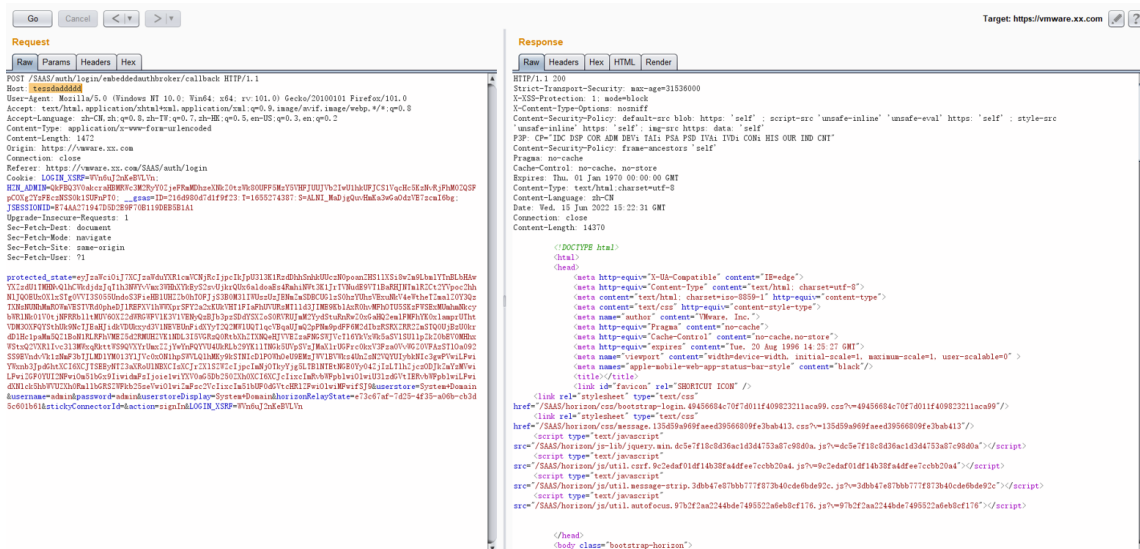
```

最后访问 `https://<域名>`，到登陆页面，环境搭建成功



动态分析

抓取登陆数据包，修改 Host 头为 tesseddddd 后进行重放



回到 vmware 中，查看/opt/vmware/horizon/workspace/logs/horizon.log

在日志中发现，vm 对任意输入的 host 头，发送了 HTTP 请求，并且因为无法解析而抛出异常

```

java.net.UnknownHostException: tessadddd: temporary failure in name resolution
at java.net.InetAddressImpl.lookupAllHostAddr(Native Method) ~[?:1.8.0_312]
at java.net.InetAddress$2.lookupAllHostAddr(InetAddress.java:929) ~[?:1.8.0_312]
at java.net.InetAddress.getAddressesFromNameService(InetAddress.java:1324) ~[?:1.8.0_312]
at java.net.InetAddress.getByAddress(InetAddress.java:1277) ~[?:1.8.0_312]
at java.net.InetAddress.getByAddress(InetAddress.java:1183) ~[?:1.8.0_312]
at java.net.InetAddress.getByAddress(InetAddress.java:1127) ~[?:1.8.0_312]
at org.apache.http.impl.conn.SystemDefaultTlsResolver.resolve(SystemDefaultTlsResolver.java:45) ~[local-password-auth-adapter-0.1.jar:?]
at org.apache.http.impl.conn.DefaultHttpClientConnectionOperator.connect(DefaultHttpClientConnectionOperator.java:112) ~[local-password-auth-adapter-0.1.jar:?]
at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.connect(PoolingHttpClientConnectionManager.java:376) ~[local-password-auth-adapter-0.1.jar:?]
at org.apache.http.impl.execchain.MainClientExec.establishRoute(MainClientExec.java:393) ~[local-password-auth-adapter-0.1.jar:?]
at org.apache.http.impl.execchain.MainClientExec.execute(MainClientExec.java:236) ~[local-password-auth-adapter-0.1.jar:?]
at org.apache.http.impl.execchain.ProtocolExec.execute(ProtocolExec.java:186) ~[local-password-auth-adapter-0.1.jar:?]
at org.apache.http.impl.execchain.RetryExec.execute(RetryExec.java:89) ~[local-password-auth-adapter-0.1.jar:?]
at org.apache.http.impl.client.InternalHttpClient.doExecute(InternalHttpClient.java:185) ~[local-password-auth-adapter-0.1.jar:?]
at org.apache.http.impl.client.CloseableHttpClient.execute(CloseableHttpClient.java:83) ~[local-password-auth-adapter-0.1.jar:?]
at org.apache.http.impl.client.CloseableHttpClient.execute(CloseableHttpClient.java:188) ~[local-password-auth-adapter-0.1.jar:?]
at com.vmware.horizon.adapters.local.LocalPasswordAuthService.authenticate(LocalPasswordAuthService.java:81) [local-password-auth-adapter-0.1.jar:?]
at com.vmware.horizon.adapters.local.LocalPasswordAuthAdapter.login(LocalPasswordAuthAdapter.java:178) [local-password-auth-adapter-0.1.jar:?]
at com.vmware.horizon.federationbroker.FederationBrokerService.loginFromEmbeddedAuthAdapter(FederationBrokerService.java:537) [federation-broker-business-0.1.jar:21.08.0.1 Build 19010796]
at com.vmware.horizon.federationbroker.FederationBrokerService.loginUsingEmbeddedAuthBroker(FederationBrokerService.java:322) [federation-broker-business-0.1.jar:21.08.0.1 Build 19010796]
at com.vmware.horizon.federationbroker.FederationBrokerService.doEmbeddedAuthBrokerLogin(FederationBrokerService.java:256) [federation-broker-business-0.1.jar:21.08.0.1 Build 19010796]
at com.vmware.horizon.federationbroker.FederationBrokerService$$FastClassBySpringCGIIB541a26567.invoke(<generated>) [federation-broker-business-0.1.jar:21.08.0.1 Build 19010796]
at org.springframework.cglib.proxy.MethodProxy.invoke(MethodProxy.java:218) [spring-core-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invokeJoinpoint(CglibAopProxy.java:729) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:163) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:750) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at com.vmware.vim.common.tracing.aop.AopAspect.processAroundAdvice(AopAspect.java:45) [vim-common-distributed-tracing-0.39.jar:?]
at sun.reflect.GeneratedMethodAccessor1254.invoke(Unknown Source) ~[?:?]
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[?:1.8.0_312]
at java.lang.reflect.Method.invoke(Method.java:498) ~[?:1.8.0_312]
at org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethodWithGivenArgs(AbstractAspectJAdvice.java:644) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethod(AbstractAspectJAdvice.java:633) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at org.springframework.aop.aspectj.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:175) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:750) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:95) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:750) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:692) [spring-aop-5.2.15.RELEASE.jar:5.2.15.RELEASE]
at com.vmware.horizon.federationbroker.FederationBrokerService$$EnhancerBySpringCGIIB541a26567.doEmbeddedAuthBrokerLogin(<generated>) [federation-broker-business-0.1.jar:21.08.0.1 Build 19010796]
at com.vmware.horizon.service.controller.auth.LoginController.performEmbeddedLoginRequestWithRelayState(LoginController.java:1666) [classes/21.08.0.1 Build 19010796]

```

同时，根据日志信息中整个认证的调用栈，我们将分析的开端定位在 `local-password-auth-adapter-0.1.jar` 中

分析 `com.vmware.horizon.adapters.local.LocalPasswordAuthAdapter#login` 函数

```

@NotNull
public AuthAdapterResponse login(@NotNull String tenantId, @NotNull Map<String, String> config, @Nullable HttpServletRequest request, @Nullable HttpServletResponse response, @Nullable Map<String, String> inputParameters) throws AuthAdapterException {
    AuthAdapterResponse authAdapterResponse = new AuthAdapterResponse();
    log.info("login for local password auth adapter is called.");
    Locale locale = request == null ? Locale.getDefault() : request.getLocale();
    this.messages.setLocale(locale);
    if (inputParameters == null) {
        inputParameters = new HashMap();
    }
    if (null == ((Map)inputParameters).get("multiAttempt")) {
        ((Map)inputParameters).put("multiAttempt", "0");
        return this.generateResponseForCollectingUserInputFirstTime((Map)inputParameters, locale, authAdapterResponse);
    } else {
        try {
            this.assertRequiredUserInput((Map)inputParameters, locale);
        } catch (AuthAdapterConfigException var16) {
            authAdapterResponse = this.generateErrorResponse((Map)inputParameters, locale, authAdapterResponse);
            locale = "en_US";
            authAdapterResponse.getConfigAttributes().add(this.createErrorAttribute("error.local.password.multiple.login", var16.getMessage()));
            return authAdapterResponse;
        }
        String username = ((String)((Map)inputParameters).get("username")).trim();
        String password = ((String)((Map)inputParameters).get("password")).trim();
        String domain = ((String)((Map)inputParameters).get("domain"));
        String endpoint = this.getLocalUrl(request);
        String userAgent = request.getHeader("user-agent");
        if (null != endpoint && this.getLocalPasswordService(config).authenticate(endpoint, tenantId, username, password, domain, userAgent)) {
            if (response != null) {
                response.setStatus(200);
            }
            return generateSuccessResponse(username, domain);
        }
    }
}

```

在获取到账号密码等信息之后，通过 `getLocalUrl` 函数来提取参数 `endpoint`

跟进

`com.vmware.horizon.adapters.local.LocalPasswordAuthAdapter#getLocalUrl` 函数

```

private String getLocalUrl(HttpServletRequest request) {
    if (null == request) {
        return null;
    } else {
        try {
            return fromURL(SSLConst.HTTP, request.getServerName(), request.getServerPort(), request.getContextPath() + "/api/v1/identity/auth/endpoint").toString();
        } catch (URISyntaxException var3) {
            log.error("failed to create URL: " + var3.getMessage(), var3);
            return null;
        }
    }
}

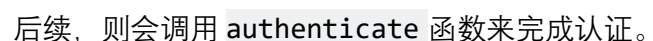
```

variables

- endpoint = Cannot find local variable 'endpoint'
- request = Cannot find local variable 'request'
- SSLConst.HTTP = 80
- request.getServerName() = tessadddd
- this = LocalPasswordAuthAdapter@91805

继续跟进将回到

此时的 `endpoint` 即为新构造的 `HTTPS` 请求，主机名则为我们任意输入的 `tessdadddd`



在

```

1 public Boolean authenticate(@NotNull String endpoint, @NotNull String tenantId, @NotNull String username, @NotNull String password, @Nullable String domain, @NotNull String userAgent) {
2     if (StringUtils.isEmpty(endpoint) || StringUtils.isEmpty(tenantId)) {
3         throw new IllegalArgumentException("endpoint = " + username + ", tenantId = " + endpoint);
4     }
5
6     try {
7         RequestBuilder requestBuilder = RequestBuilder.create("POST").setUri(endpoint).addHeader("name", "Access").addHeader("application/json", "application/json").addHeader("name", "Content-Type").addHeader("name", "User-Agent", userAgent);
8         requestBuilder.setUri(new String[] { endpoint }).setUri(new LocalPasswdService.getLoginEndpoint(username, password, domain)).setUri(new LocalPasswdService.getLoginEndpoint(username, password, domain));
9         HttpResponse response = requestBuilder.execute().getHttpResponse();
10         HttpStatusCode statusCode = response.getStatusCode();
11         String responseBody = response.getBody().asString();
12         if (statusCode == HttpStatusCode.OK) {
13             responseCode = HttpStatusCode.OK.getStatusCode();
14             if (200 == responseCode) {
15                 boolean userIsAuthorized = responseBody.contains("username");
16                 Boolean valid = true;
17                 return valid;
18             }
19         }
20         LOG.warn("failed to authenticate user " + username + ", status code " + responseCode);
21     } catch (IOException var10) {
22         LOG.warn("failed to authenticate user " + username);
23         this.logger.addException(var10);
24     } finally {
25         this.logger.addSubsegment();
26     }
27 }

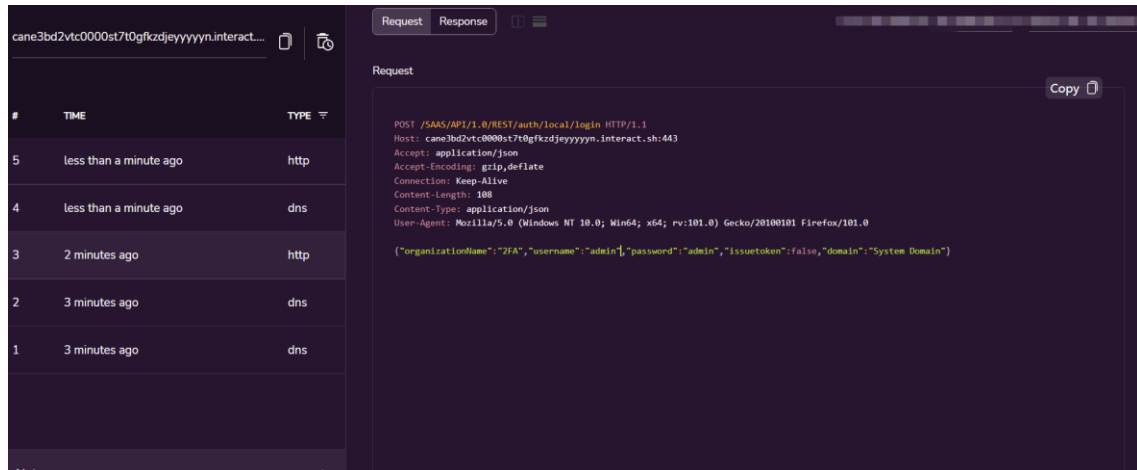
```

通过之前的 `HTTPS` 请求，使用 `POST` 方式发送，后续会直接根据请求返回的状态码来判断是否认证成功，若状态码为 `200`，即认证成功。因此可通过伪造 `HOST` 头和伪造

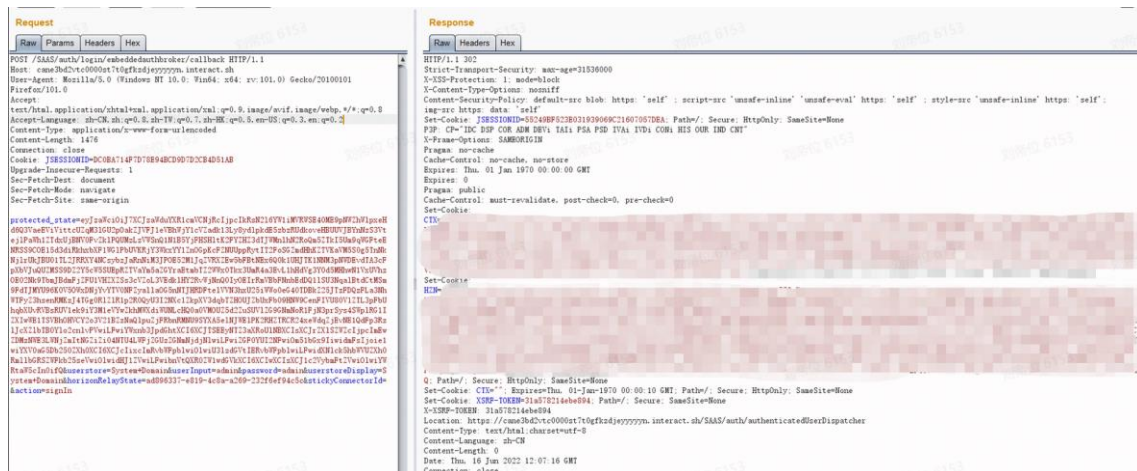
HTTPS 服务器并保证对任何请求返回状态 200，即可实现认证绕过。

漏洞复现

伪造 HTTPS 服务器，满足对任意请求都返回 200



修改 HOST 为伪造的 HTTPS 服务器地址，成功绕过认证并获取有效 cookie



注：在漏洞复现中发现，vm 在认证之后还会对用户名进行检查，所以使用的用户必须是存在的。

后续可通过 JDBC 注入（参考 CVE-2022-22957）实现 RCE