Information Security

Information security project,
realized by automatizing known exploits

carried out by

Andrea Righi

Matr. ID: 15483

Cristiano Lucian

Matr. ID: 15850

Bolzano, 6 June 2020

Contents

List of Figures

Listings

1 Introduction

This collection of exploits found on Exploit-DB and SecLists.Org, were tested by Andrea Righi and Cristiano Lucian as project for the Information Security Course at Unibz. The majority of the scripts for the automation of the exploits found in this project were developed by us. We used three operating system environments: Kali Linux, Microsoft Windows 10 and Mac OS Catalina. The applications present in this project have been patched with new releases, thus in every exploit description we specified the correct version that is affected.

2 Vulnerable applications

2.1 HideMyAss Pro VPN

HideMyAss Pro VPN is a pay-per-use VPN service: it is an Avast product and its application is available for almost all the operating systems; it counts 400.000 million users worldwide. The vendor website can be found at: HideMyAss Pro VPN.

2.2 MEmu Play

MEmu Play software is an emulator of the Android environment, it is developed for Windows and it is widely used to play Android games. The vendor website can be found at: MEmu Play Android Emulator.

2.3 Gym Management System

Gym Management System is a free software, developed in PHP and MySql: it is sold as an easy way to use gym and health gym membership system. The vendor website can be found at: Gym Management System.

2.4 Hospital Management System

Gym Management System is a free software, developed in PHP and MySql: it is sold as a web application for the hospital, which manages doctors and patients. The vendor website can be found at: Hospital Management System.

2.5 PHP-Fusion Content Management System

PHP-Fusion is an all in one integrated and scalable open-source content management system written in PHP. The software is highly extensible with plugins that make it suitable for any kind of personal or commercial website. The vendor website can be found at: PHPFusion CMS.

2.6 Cockpit Next Content Management System

Cockpit Next is a self-hosted, open-source content management system fully customizable due to its api-driven structure. The vendor website can be found at: Cockpit Next.

3 Exploits

3.1 Privilege escalation and Reverse Shell attack

3.1.1 HideMyAss Pro VPN

Version

The exploit was tested on HideMyAss Pro VPN v3.3.0.3 for macOS (Catalina 10.15.04).

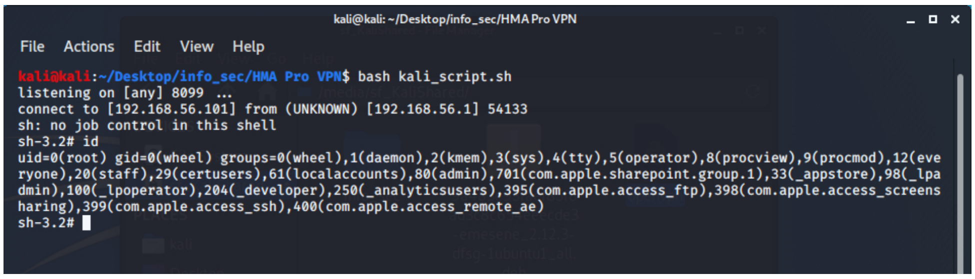
Cause of the vulnerability

When HideMyAss Pro VPN is installed in a macOS system, it install also a an helper binary *com.privax.hmaprovpn.helper* in the PriviligedHelperTools folder; this folder contains all the programs authorized to run as root. This behaviour is encouraged by Apple, because placing helpers in the PriviligedHelperTools folder will limit the possibilities of a potential privileged escalation compared to the case if the entire application is running as root, and it will also make the application more reliable, as if there is a crash in this tool, the main application can still run, and the helper can be restarted safely.

The vulnerability found in the HMA application, is that the helper is responsible for opening VPN connections with correct security and connection profile settings: however, it does not perform any security check on the *openvpn* executable, which can be replaced to run malicious code.

Implementation

We automatized this exploit by using two machines: the attack was carried out by a virtual one, a Kali OS (2020.1 running on VirtualBox 6.1.8) while the victim was the host machine,


kali@kali: ~/Desktop/info_sec/HMA Pro VPN
File Actions Edit View Help

```
kali@kali:~/Desktop/info_sec/HMA Pro VPN$ bash kali_script.sh
listening on [any] 8099 ...
connect to [192.168.56.101] from (UNKNOWN) [192.168.56.1] 54133
sh: no job control in this shell
sh-3.2# id
uid=0(root) gid=0(wheel) groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(procmod),12(eve
ryone),20(staff),29(certusers),61(localaccounts),80(admin),701(com.apple.sharepoint.group.1),33(_appstore),98(_lpa
dmin),100(_lpoperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),398(com.apple.access_screens
haring),399(com.apple.access_ssh),400(com.apple.access_remote_ae)
sh-3.2# █
```

Figure 1: Kali terminal view

a macOS (Catalina 10.15.04). The automatization is created thanks to a bash script: it should be run on the attacker machine: Its execution will generate two output files: *openvpn* and a bash file. They should be moved on the victim machine, and the bash should be executed. It will perform the exploit, visible when the HideMyAss Pro VPN will be activated.

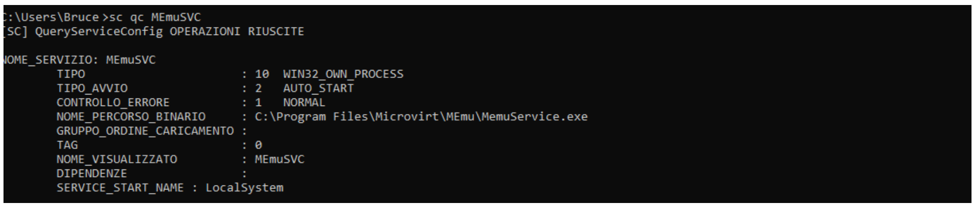
3.1.2 MEmu Play Android Emulator

Version

The exploit was tested on MEmu Play v6.0.6 for Windows 10 Pro (x64).

Cause of the vulnerability

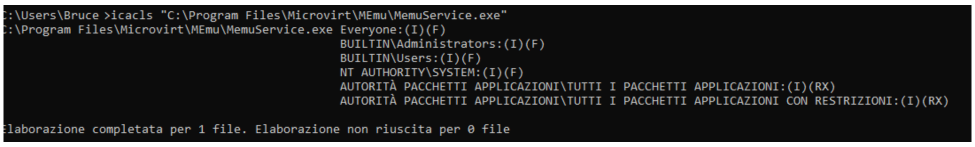
When MEmu Play is installed and launched for the first time, it needs to run the *Memu-Sevice.exe* process, that is executed as Local System process. Since during the installation process the installation folder's permissions are not edited, by default Authenticated Users group has access. Therefore *MemuService.exe* can be modified by any logged-in user, even with low privileges.



```
:\Users\Bruce >sc qc MEmuSVC
SC] QueryServiceConfig OPERAZIONI RIUSCITE

OME_SERVIZIO: MEmuSVC
        TIPO                       : 10  WIN32_OWN_PROCESS
        TIPO_AVVIO                 : 2   AUTO_START
        CONTROLLO_ERRORE           : 1   NORMAL
        NOME_PERCORSO_BINARIO      : C:\Program Files\Microvirt\MEmu\MemuService.exe
        GRUPPO_ORDINE_CARICAMENTO :
        TAG                        : 0
        NOME_VISUALIZZATO          : MEmuSVC
        DIPENDENZE                 :
        SERVICE_START_NAME : LocalSystem
```

(a) MEmuSVC runs as LocalSystem.



```
:\Users\Bruce >icacls "C:\Program Files\Microvirt\MEmu\MemuService.exe"
:\Program Files\Microvirt\MEmu\MemuService.exe Everyone:(I)(F)
                                               BUILTIN\Administrators:(I)(F)
                                               BUILTIN\Users:(I)(F)
                                               NT AUTHORITY\SYSTEM:(I)(F)
                                               AUTORITÀ PACCHETTI APPLICAZIONI\TUTTI I PACCHETTI APPLICAZIONI:(I)(RX)
                                               AUTORITÀ PACCHETTI APPLICAZIONI\TUTTI I PACCHETTI APPLICAZIONI CON RESTRIZIONI:(I)(RX)

laborazione completata per 1 file. Elaborazione non riuscita per 0 file
```

(b) Permissions of the MEmu folder, even Everyone has (F) = full access.

Figure 2: Permission vulnerabilities

This vulnerability can lead to a low-privileged user to substitute that process with a custom payload and trigger its execution with a restart of the system, since the MEmuSVC process auto-starts at every boot of the vulnerable machine.

Implementation

We automatized this exploit by using two machines: the attack was carried out by a machine running Kali Linux OS while the victim was a virtual machine running Windows 10 Pro on an another computer. These machines were connected on the same wireless network.

The *"kali_script.sh"* is a bash script and sets up the attacking machine preparing it for the attack. First it creates the payload and places it on the */var/www/html* folder, reading the current IP address and setting up an apache2 server (pre-installed with Kali). Then it opens up a tab for the listener of the payload on port 443. The payload can be injected in the victim machine in various manners, running a server on the attacking machine and then downloading the malicous batch script from there was for us the most suitable way for this project. The *"win_malicious.bat"* is a batch script for the victim's Windows machine. It asks the user for the attacking machine's IP, then connects to its webserver, renames the current *MemuService.exe* file (since it is running it cannot be deleted) and downloads the payload in the correct path (MEmu Installer has *"C:\Program Files\"* or *"C:\Program Files (x86)\"* as default installation paths).

When the system is restarted, a reverse shell opens in the listener tab on the attacking machine. Since the payload has full access on the system, the attacker can execute any kind of command.

Listing 1: Example of windows commands that could be executed in the reverse shell

```
C:\Windows\system32>whoami >> nt authority\system
net user hacker hacker /add
shutdown
```

3.2 Remote Code Execution

3.2.1 Gym Management System

Version

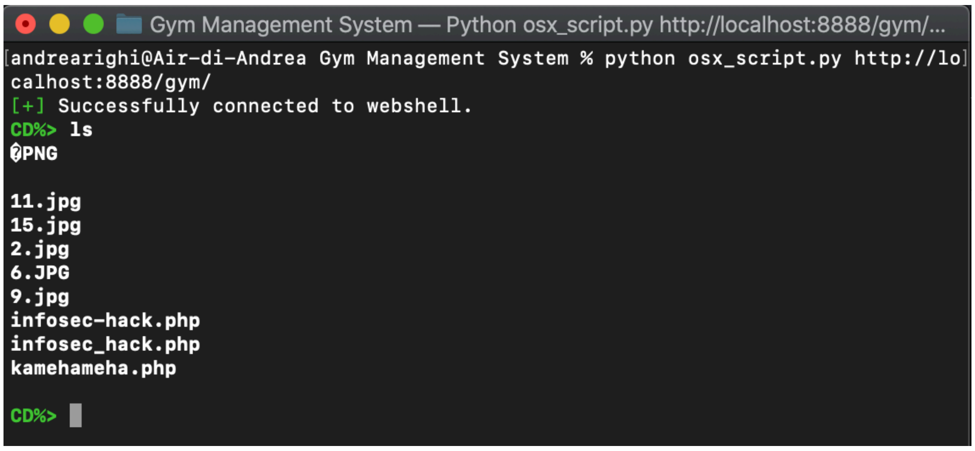
The exploit was tested on Gym Management System 1.0 for macOS (Catalina 10.15.04), using MAMP 5.7.

Cause of the vulnerability

In the *upload.php* file, there is no session control: therefore it allows uploading of files in an unauthenticated way. This behaviour allows for uploading of malicious file, in order to gain access to a remote code execution on the server. We achieved its automatization, by uploading a double extension file, called *infosec-hack.php.jpg*: this file is able to bypass the extension check for images.

Implementation

We automatized this exploit by using a macOS machine (Catalina 10.15.04). The automatization is created thanks to a Python script; we first get connection with the server: then we upload the malicious file, which contains a PHP script that connects our terminal with the server one. Here we can see the content of the folder, with our PHP file inserted in it.


Gym Management System — Python osx_script.py http://localhost:8888/gym/...

```
andrearighi@Air-di-Andrea Gym Management System % python osx_script.py http://lo
calhost:8888/gym/
[+] Successfully connected to webshell.
CD%> ls
�PNG

11.jpg
15.jpg
2.jpg
6.JPG
9.jpg
infosec-hack.php
infosec_hack.php
kamehameha.php

CD%> 
```

Figure 3: OSX terminal view

3.3 Persistent Cross-Site Scripting

3.3.1 Hospital Management System

Version

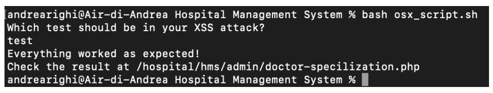
The exploit was tested on Hospital Management System 4.0 for macOS (Catalina 10.15.04), using MAMP 5.7.

Cause of the vulnerability

In the *doctor-specilization.php* file, the *doctorspecilization* parameter is not sanitzed: therefore it allows any kind of Persistent Cross-Site scripting.

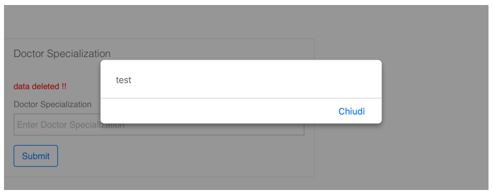
Implementation

We automatized this exploit by using a macOS machine (Catalina 10.15.04). The automatization is created thanks to a Bash script; it performs a CURL request, changing the payload of the *alert* function, in order to store what the attacker wants. It is then possibile to view the persistent XSS in the page */hospital/hms/admin/doctor-specilization.php*, logging as an admin.



```
andrearighi@Air-di-Andrea Hospital Management System % bash osx_script.sh
Which test should be in your XSS attack?
test
Everything worked as expected!
Check the result at /hospital/hms/admin/doctor-specilization.php
andrearighi@Air-di-Andrea Hospital Management System %
```

(a) OSX terminal view.



Doctor Specialization

data deleted !!

Doctor Specialization

Enter Doctor Specialization

test

(b) Browser alert view.

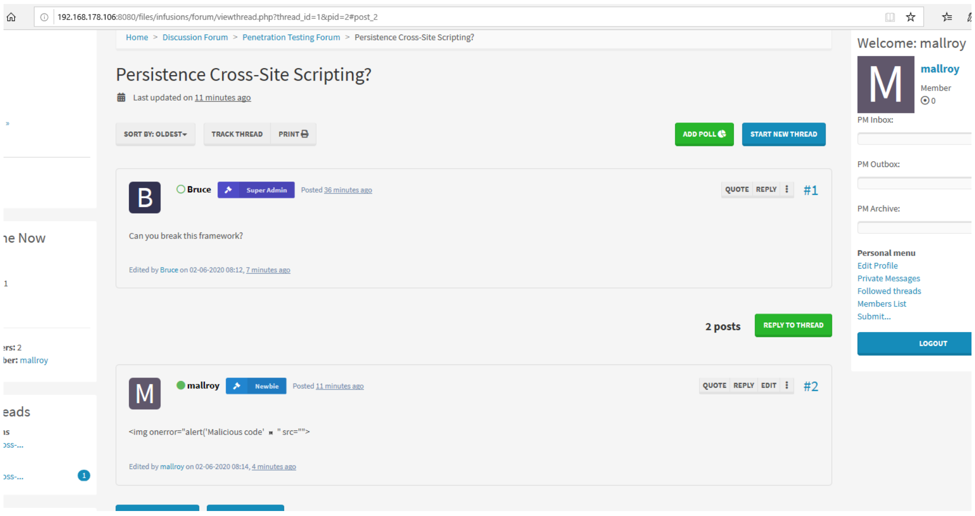
3.3.2 PHP-Fusion

Version

The exploit was tested on PHP-Fusion 9.03.50 with the "Forum infusion" plugin, on a virtual machine running Windows 10 Pro, using XAMPP 7.4.6 as local server and PHP 7.4.6.

Cause of the vulnerability

The vulnerability resides in the threads of comments in the Forum part of the framework. When a new comment is submitted, the input string is correctly sanitized, however there is a print function that lets the users to have a simplified textual view of the thread in order to be printed.



192.168.178.106:8080/files/infusions/forum/viewthread.php?thread_id=1&pid=2#post_2

Home > Discussion Forum > Penetration Testing Forum > Persistence Cross-Site Scripting?

Persistence Cross-Site Scripting?

Last updated on 11 minutes ago

Bruce Super Admin Posted 36 minutes ago #1

Can you break this framework?

Edited by Bruce on 02-06-2020 08:12, 7 minutes ago

2 posts

mallroy Newbie Posted 11 minutes ago #2

<img onerror="alert('Malicious code' " src="">

Edited by mallroy on 02-06-2020 08:14, 4 minutes ago

Welcome: mallroy

mallroy Member

Figure 5: View of the sanitized comments

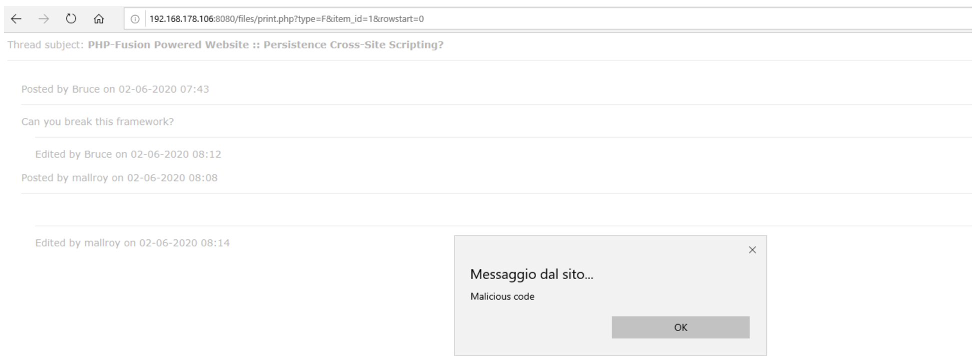
To generate the printable page, the *parse_textarea()* function is used in */print.php* from the *includes/core_functions_include.php* file. This function, as shown below, does not sanitize properly the text area of the message stored on the server, and therefore vulnerable to a Persistent XSS.

Listing 2: function without sanitation in /print.php

```
echo parse_textarea($data['post_message']);
```

Implementation

In order to replicate the attack, it is necessary to open a new thread and post *"<script>alert('Malicious code')</script>"* as a new comment (or edit an old one). Then at the top of the page, through the "Print" button, trigger the execution of the malicious code.



192.168.178.106:8080/files/print.php?type=F&item_id=1&rowstart=0

Thread subject: **PHP-Fusion Powered Website :: Persistence Cross-Site Scripting?**

Posted by Bruce on 02-06-2020 07:43

Can you break this framework?

Edited by Bruce on 02-06-2020 08:12

Posted by mallroy on 02-06-2020 08:08

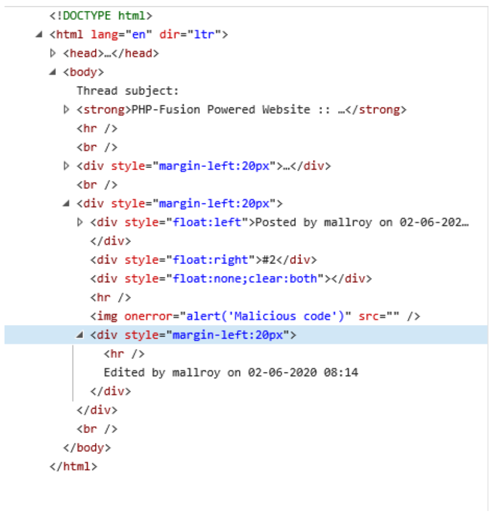
Edited by mallroy on 02-06-2020 08:14

Messaggio dal sito...

Malicious code

OK

Figure 6: View of the XSS execution



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>…</head>
  <body>
    Thread subject:
    <strong>PHP-Fusion Powered Website :: …</strong>
    <hr />
    <br />
    <div style="margin-left:20px">…</div>
    <br />
    <div style="margin-left:20px">
      <div style="float:left">Posted by mallroy on 02-06-202…
      </div>
      <div style="float:right">#2</div>
      <div style="float:none;clear:both"></div>
      <hr />
      <img onerror="alert('Malicious code')" src="" />
      <div style="margin-left:20px">
        <hr />
        Edited by mallroy on 02-06-2020 08:14
      </div>
    </div>
    <br />
  </body>
</html>
```

Figure 7: HTML code in the /print.php page

3.4 Directory Traversal

3.4.1 Cockpit Next

Version

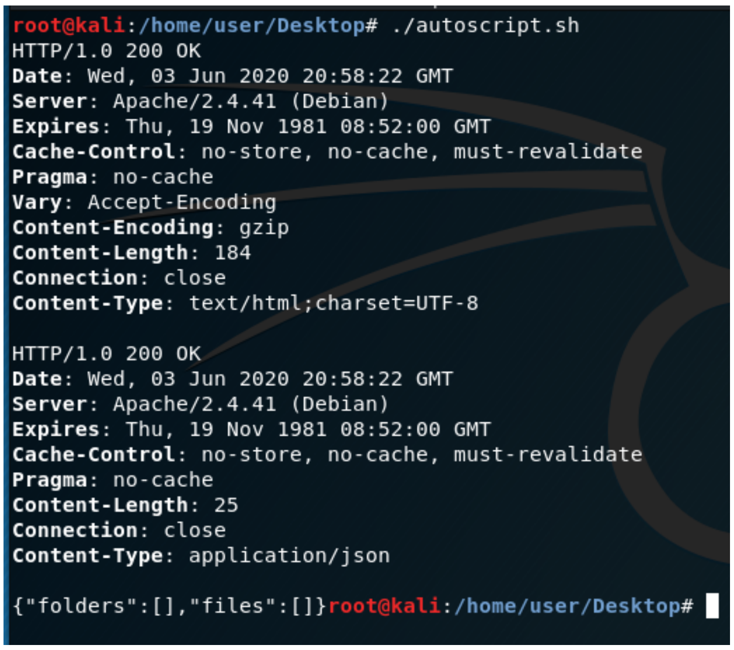
The exploit was tested on Cockpit Next CMS 0.6.2 on Kali Linux. We set up an Apache2 server to run the software, with PHP 7.3, SQLite and *mod_rewrite*, *mod_versions* enabled on Apache2.

Cause of the vulnerability

This software performs actions on files without an appropriate validation, and therefore allows an attacker to traverse the file system to unintended locations and access arbitrary files.

Implementation

The bash script we wrote to automatize the exploit performs two cURL POST requests, the first one accesses the cockpit system to let the second one perform the directory traversal. The second POST request accesses the */media/api* and sends an *ls* command to return the files in the folder traversed (*../../../../../* in this case).



```
root@kali:/home/user/Desktop# ./autoscript.sh
HTTP/1.0 200 OK
Date: Wed, 03 Jun 2020 20:58:22 GMT
Server: Apache/2.4.41 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 184
Connection: close
Content-Type: text/html;charset=UTF-8

HTTP/1.0 200 OK
Date: Wed, 03 Jun 2020 20:58:22 GMT
Server: Apache/2.4.41 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 25
Connection: close
Content-Type: application/json

{"folders":[],"files":[]}root@kali:/home/user/Desktop#
```

Figure 8: The responses of the two POST Requests

Unsuccessful Exploit Attempt

We performed also another, unsuccessful, attempt for a Server-Side Request Forgery on this software. In versions 0.5.5 and previous, there is a */assets/lib/fuc.js.php* file which uses a fetch_url_contents code vulnerable to SSRF to manipulate an url parameter, this allows remote attackers to read arbitrary files or send TCP traffic to the internet, as specified in this CVE Report.

We set up a server with XAMPP on a virtual machine running Windows 10 Pro, and sending a custom POST request from a machine running Kali Linux on the same local network. This because we wanted to replicate the exploit in an environment closer to a possible real situation. However before executing with the vulnerable parameter, the *fuc.js.php* file parses the HTTP_REFERER address from the request and compares it with the HTTP_HOST address in order to check that the request comes from the actual server. Since the referer address is parsed, they will result different in our custom request (one address with the port number and the other one without). Modifying them differently resulted in several 503 Service Unavailable errors related to XAMPP.

At this point we realized that executing the exploit from the same machine as the server's would probably work, but would be less reality-based since the attacker should first find a way to inject a script on the server and make the request from the server itself.

References

Boku (2020). *Gym Management System 1.0 - Unauthenticated Remote Code Execution*. URL: https://www.exploit-db.com/exploits/48506.

Coiffeur (2020). *PHPFusion 9.03.50 - Persistent Cross-Site Scripting*. URL: https://www.exploit-db.com/exploits/48497.

Fullshade (2020). *Hospital Management System 4.0 - Persistent Cross-Site Scripting*. URL: https://www.exploit-db.com/exploits/47841.

Mitre (2018). *CVE-2018-15540*. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-15540.

— (2020). *CVE-2020-5191*. URL: https://nvd.nist.gov/vuln/detail/CVE-2020-5191.

Sahin, **Han** (2017). *HideMyAss Pro VPN Client for macOS 3.x - Local Privilege Escalation*. URL: https://www.exploit-db.com/exploits/41952.

Sánchez, **Alejandra** (2019). *Memu Play 6.0.7 - Privilege Escalation (PoC)*. URL: https://www.exploit-db.com/exploits/46437.

Uvarov, **Simon** (2018). *Cockpit CMS Multiple Vulnerabilities*. URL: https://seclists.org/fulldisclosure/2018/Oct/30.

Wu, **Qian**/**Wang**, **Bo**/**Zhang**, **Jiawang** (2018). *SSRF (Server Side Request Forgery) in Cockpit 0.4.4-0.5.5*. URL: https://seclists.org/fulldisclosure/2018/May/10.