UnderPass

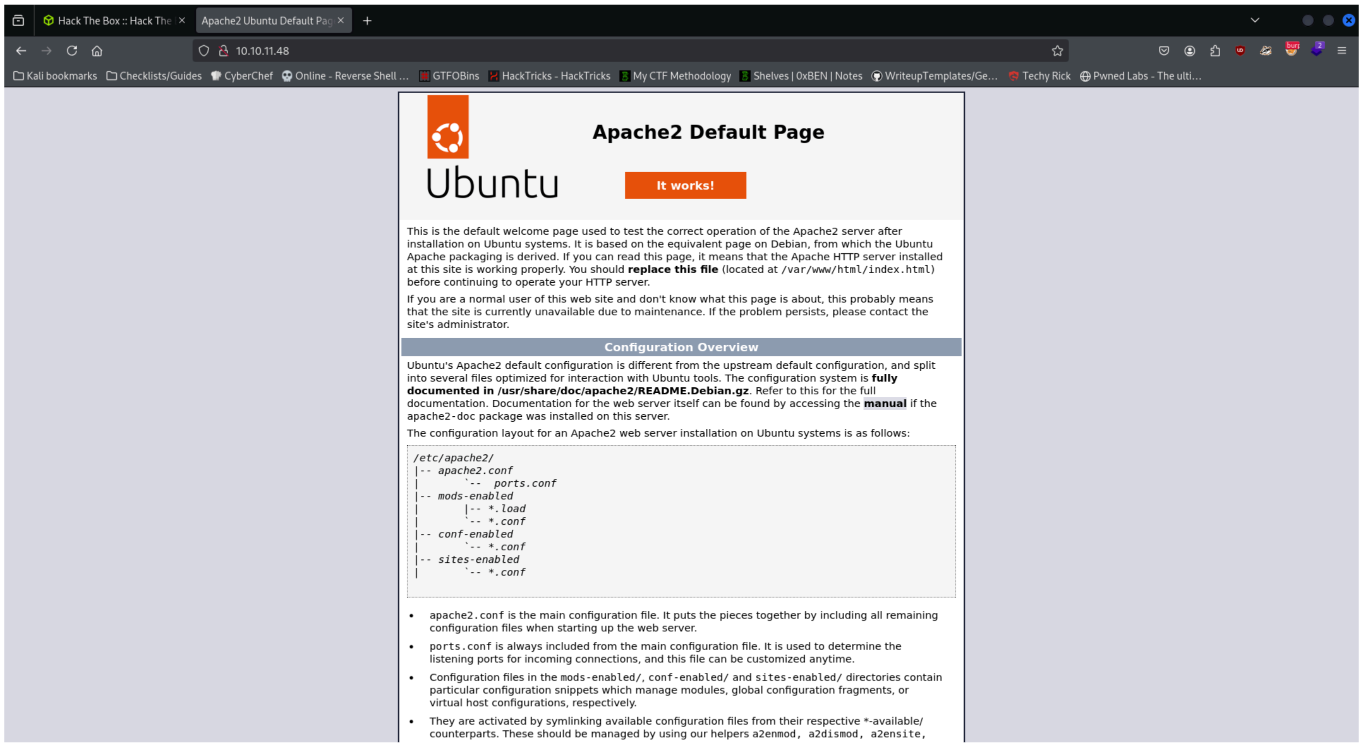
Tags: #Linux/Ubuntu #Easy #Apache #PHP #SNMP #RADIUS #Plaintext-Creds #Default-Creds #Sudo-Misconfiguration

Nmap Results

```
Nmap scan report for 10.10.11.48
Host is up (0.086s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   256 48:b0:d2:c7:29:26:ae:3d:fb:b7:6b:0f:f5:4d:2a:ea (ECDSA)
|_  256 cb:61:64:b8:1b:1b:b5:ba:b8:45:86:c5:16:bb:e2:a2 (ED25519)
80/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.09 seconds
```

Service Enumeration

Nmap scan results are relatively basic, 2 open ports both being SSH and an HTTP webserver. Let's see what we find when we navigate to this site in our browser:



Apache2 Default Page

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`,

It's Apache's default page after installation. There's nothing interesting here so I'll rely on directory enumeration. I ran the usual command `feroxbuster -u http://10.10.11.48 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt`, however I didn't get any results, even after using a few other dictionary files.

So I messed around with `nmap` more. I ran a scan against all possible ports with `sudo nmap -p- 10.10.11.48` but the same 2 ports appeared. The last option I had was running a UDP scan with `sudo nmap -sU -T4 10.10.11.48`. The "-T4" flag is important because it makes nmap scan faster, which is needed since these kinds of scans usually take a long time. Fortunately, we found something of interest. Here is the output of that UDP scan:

```
Nmap scan report for 10.10.11.48
Host is up (0.087s latency).
Scanned at 2024-12-26 18:12:18 EST for 1044s
Not shown: 965 closed udp ports (port-unreach)
PORT     STATE         SERVICE
22/udp   open|filtered ssh
80/udp   open|filtered http
161/udp  open          snmp
162/udp  open|filtered snmptrap
1038/udp open|filtered mtqp
1066/udp open|filtered fpo-fns
1434/udp open|filtered ms-sql-m
1812/udp open|filtered radius
```

```
1813/udp  open|filtered radacct
9020/udp  open|filtered tambora
17332/udp open|filtered unknown
17490/udp open|filtered unknown
17549/udp open|filtered unknown
18319/udp open|filtered unknown
18888/udp open|filtered apc-necmp
19722/udp open|filtered unknown
19936/udp open|filtered unknown
20019/udp open|filtered unknown
20540/udp open|filtered unknown
24279/udp open|filtered unknown
25240/udp open|filtered unknown
26415/udp open|filtered unknown
27707/udp open|filtered unknown
32778/udp open|filtered sometimes-rpc20
33866/udp open|filtered unknown
35702/udp open|filtered unknown
41896/udp open|filtered unknown
44190/udp open|filtered unknown
49204/udp open|filtered unknown
49214/udp open|filtered unknown
51717/udp open|filtered unknown
51972/udp open|filtered unknown
54321/udp open|filtered bo2k
57813/udp open|filtered unknown
62154/udp open|filtered unknown
Final times for host: srtt: 86555 rttvar: 603  to: 100000

Read from /usr/share/nmap: nmap-protocols nmap-service-probes nmap-services.
Nmap done: 1 IP address (1 host up) scanned in 1044.58 seconds
```

There is one port that is marked as **open**, not **open|filtered** like the rest of them, which is port 161 running **SNMP**. Nmap knows for sure that this port is opened and the service on it because of how it takes advantage of the nature of UDP.

Nmap and the UDP protocol

Unlike TCP, UDP does not require an acknowledgement (ACK) to confirm a connection. UDP services usually only respond to specific requests, so an opened UDP port will not respond if it receives data it doesn't understand. Nmap can't distinguish this from a potential firewall dropping the request, so it will mark that port as open|filtered, since it didn't receive a response back either way. However if a service on a port receives a valid request, it will respond and nmap will know that that port is opened and what service is running on it.

Even without the -sV flag, UDP scans automatically probe service info because to determine if the port is opened in the first place, it has to send data that the service on that port will accept. The only way to do this is by guessing. Nmap has a few lists that include ports associated with the most common service(s) and custom crafted packets for determining a service. It uses these lists together to generate accurate results in UDP scans and version detection scans. For example, when checking port 53, nmap will look in those lists to find the most common service on that port, which is DNS, and look for a DNS packet to send. If it receives a response, it knows that DNS is running on that port.

Now that I know SNMP runs on port 161, I'll run a more detailed scan on it by passing the -sC flag for default scripts and -sV to enumerate versions:

`sudo nmap -sU -p 161 -sC -sV -o nmap/snmp-port 10.10.11.48`

Here are the results:

```
Nmap scan report for 10.10.11.48
Host is up (0.090s latency).

PORT    STATE SERVICE VERSION
161/udp open  snmp    SNMPv1 server; net-snmp SNMPv3 server (public)
| snmp-info:
|   enterprise: net-snmp
|   engineIDFormat: unknown
|   engineIDData: c7ad5c4856d1cf6600000000
|   snmpEngineBoots: 29
|_  snmpEngineTime: 10h17m48s
| snmp-sysdescr: Linux underpass 5.15.0-126-generic #136-Ubuntu SMP Wed Nov 6
10:38:22 UTC 2024 x86_64
|_  System uptime: 10h17m47.80s (3706780 timeticks)
Service Info: Host: UnDerPass.htb is the only daloradius server in the basin!

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.78 seconds
```

Looking at the Service Info line, the scan discovered a hostname `underpass.htb` and also specifies that it is a daloRADIUS server, a GUI web based program that interacts with a RADIUS server and makes the entire workflow for an admin easier.

Crash course on the RADIUS protocol

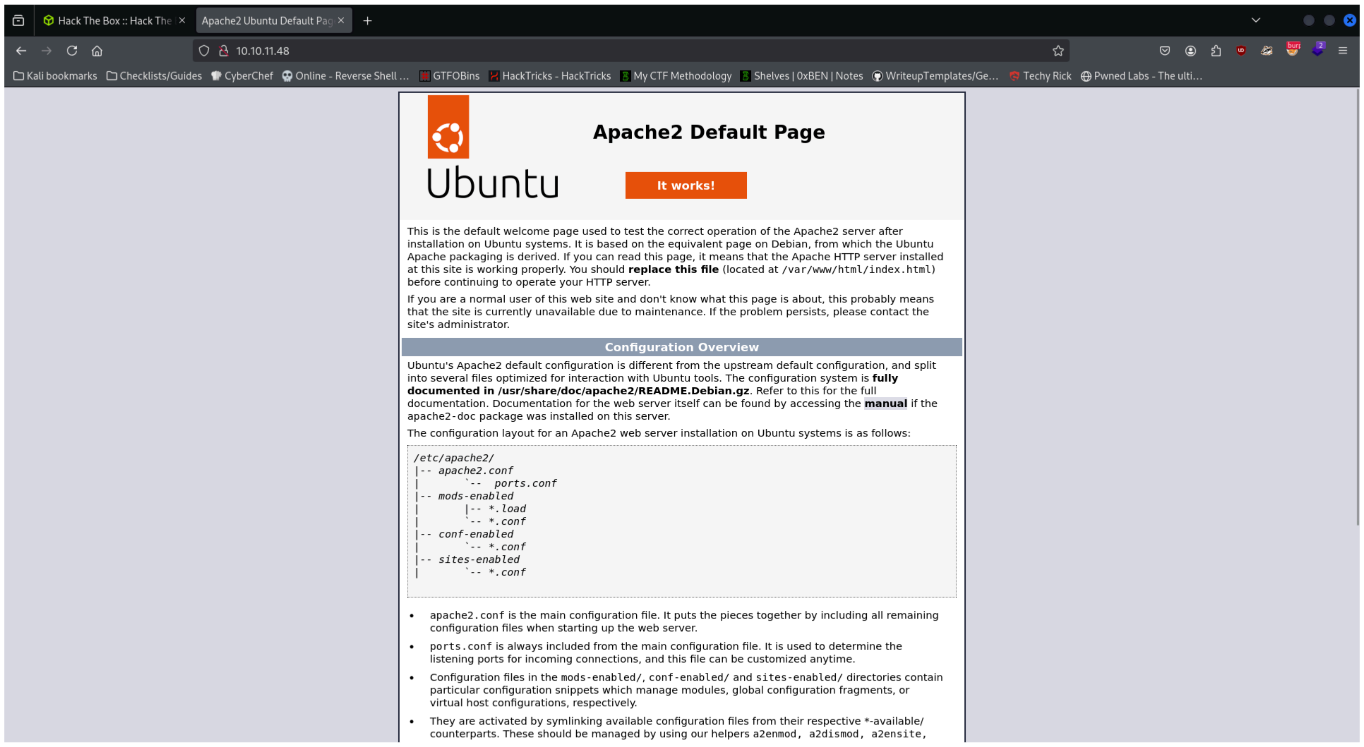
RADIUS stands for Remote Authentication Dial-In User Service. It is a client-server networking protocol that implements the **AAA Framework**, meaning it handles user authentication, authorization, and accounting. Essentially, the RADIUS protocol controls *who* can access the network, *what* they're allowed to do, and logs what they actually do.

The radius client, or network access server (NAS), is a device like a switch, router, wireless APs, etc., whose purpose is to communicate with the radius server and enforce the decisions it makes and the rules it sets. The radius server, as previously mentioned, is the "brain" responsible for telling the NAS(s) what to do.

Here's how a typical RADIUS setup authenticates end-users:

1. The end-user sends a request to a radius client (NAS) asking to connect to the network and supplies their username and password
2. The client sends an access request message to the radius server over UDP (usually port 1812) containing the end-users credentials and some info about the NAS. The user's password is encrypted using a **shared secret** unique to each client that is stored on that NAS and the radius server. This is to ensure the message's integrity. The shared secret is never transmitted for security purposes.
3. With this info, the radius server does 2 things, **verify** the NAS's identity and determine whether the user should be allowed or denied access to the network. For the NAS's identity, it looks at the shared secret it has on file for that specific NAS and uses it to decrypt the user's password. If decryption is successful then the radius server knows that the NAS is who it says it is. If not then it will drop the entire request without notifying the NAS even if the user's credentials are valid, preventing a potential attacker from gathering info about the server. To actually authenticate the user, it validates the user's credentials by comparing them against entries in the database, as discussed earlier.
4. After the radius server sends its decision to the NAS, the NAS takes the proper action, either accept or deny the user.

Common implementations of the RADIUS protocol are found in WPA2-Enterprise networks (unlike WPA2-Personal networks where only a single wifi password is required for access) and VPN networks.



Apache2 Default Page

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

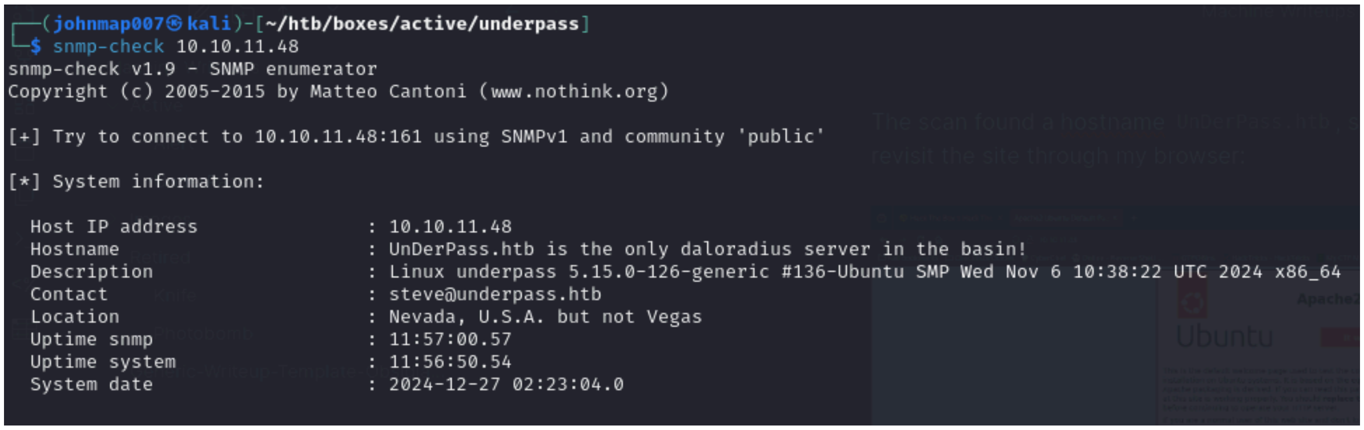
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

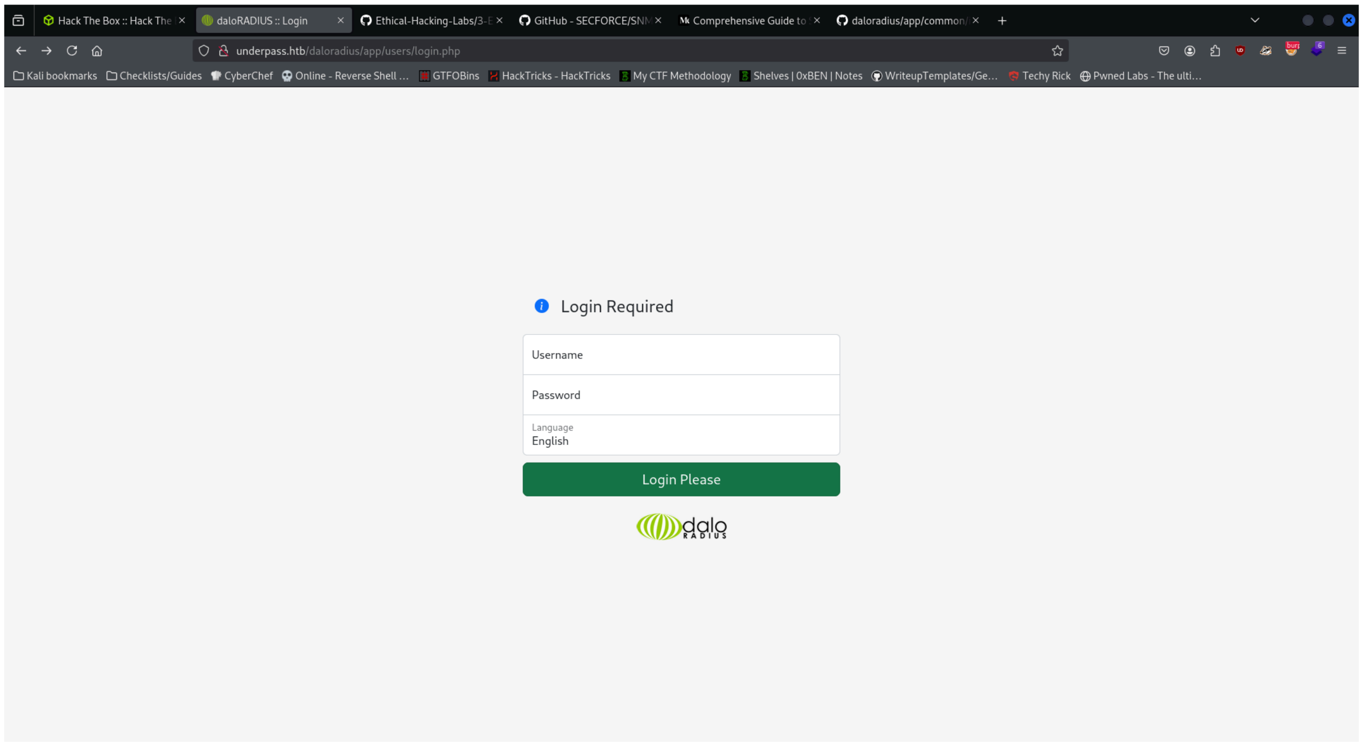
- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod, a2dismod, a2ensite,`

The same Apache default page is still being displayed, so I'll explore SNMP a little more with `snmp-check` to see what else we can find. All we need to do is pass the ip address. Here's the results:



```
┌──(johnmap007㉿kali)-[~/htb/boxes/active/underpass]
└─$ snmp-check 10.10.11.48
snmp-check v1.9 - SNMP enumerator
Copyright (c) 2005-2015 by Matteo Cantoni (www.nothink.org)

[+] Try to connect to 10.10.11.48:161 using SNMPv1 and community 'public'

[*] System information:

  Host IP address               : 10.10.11.48
  Hostname                      : UnDerPass.htb is the only daloradius server in the basin!
  Description                   : Linux underpass 5.15.0-126-generic #136-Ubuntu SMP Wed Nov 6 10:38:22 UTC 2024 x86_64
  Contact                       : steve@underpass.htb
  Location                      : Nevada, U.S.A. but not Vegas
  Uptime snmp                   : 11:57:00.57
  Uptime system                 : 11:56:50.54
  System date                   : 2024-12-27 02:23:04.0
```

I found a potential user **"Steve"**. Could be a user on the daloRADIUS server I found earlier. But first we need to locate a login page of some sort. A quick google search reveals that the default path is just **/daloradius**, but upon navigating to the page, I get a 403 Forbidden status code. Fortunately daloradius is open source and available on GitHub, so I went there to find where the login panel is located. The most likely path I found was **/app/users/login.php**, so I tried that and finally got a login page:

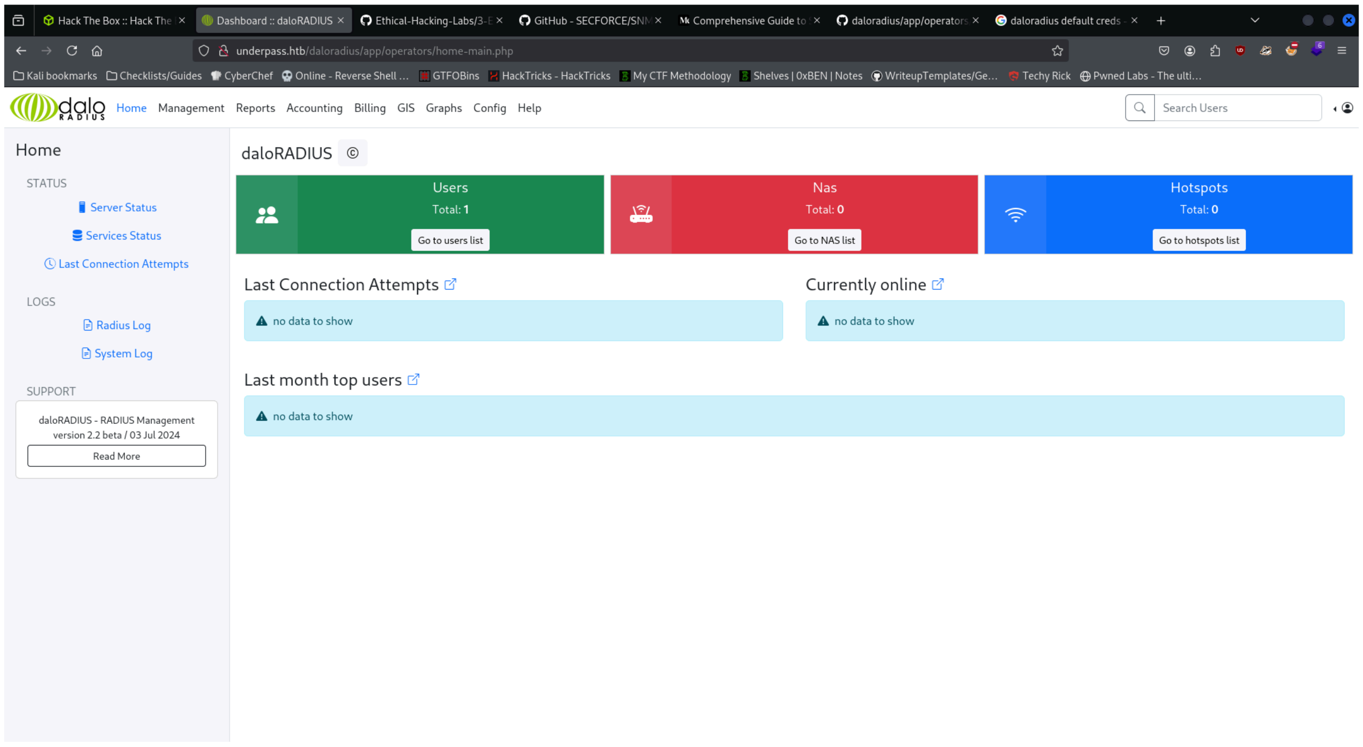


Exploitation

Initial Access

The default creds are "administrator" as the username and "radius" as the password, but those didn't work. We don't know Steve's password so we can't log in as him. I tried inputting some basic SQL Injection payloads like `' OR 1=1--` and using `sqlmap` against it but neither worked. There was no software version displayed on the page or hidden in the source code, so I couldn't look for an exploit online.

I ended up going back to the GitHub page and seeing what else I could find. It turns out there is a 2nd log in page at **/app/operators/login.php** that looks very similar to the login page before. The default creds worked here, and we are now logged in as administrator. This is what we see on the dashboard:



daloRADIUS

Users Total: 1

Nas Total: 0

Hotspots Total: 0

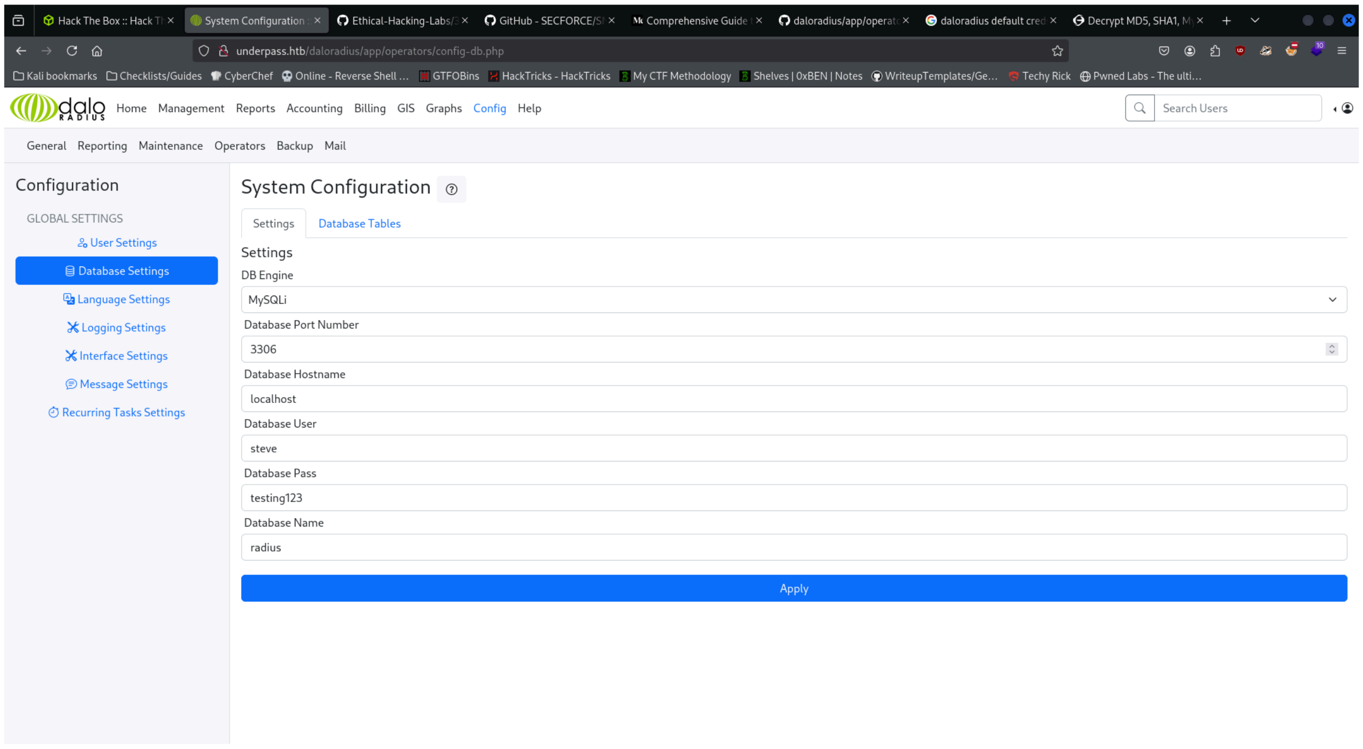
Last Connection Attempts: no data to show

Currently online: no data to show

Last month top users: no data to show

daloRADIUS - RADIUS Management version 2.2 beta / 03 Jul 2024

My first instinct was to head over to the config tab to see if I can find some sensitive data. Looking in the database settings tab, I discovered that this machine is using a MySQL database and also found the credentials to access it:

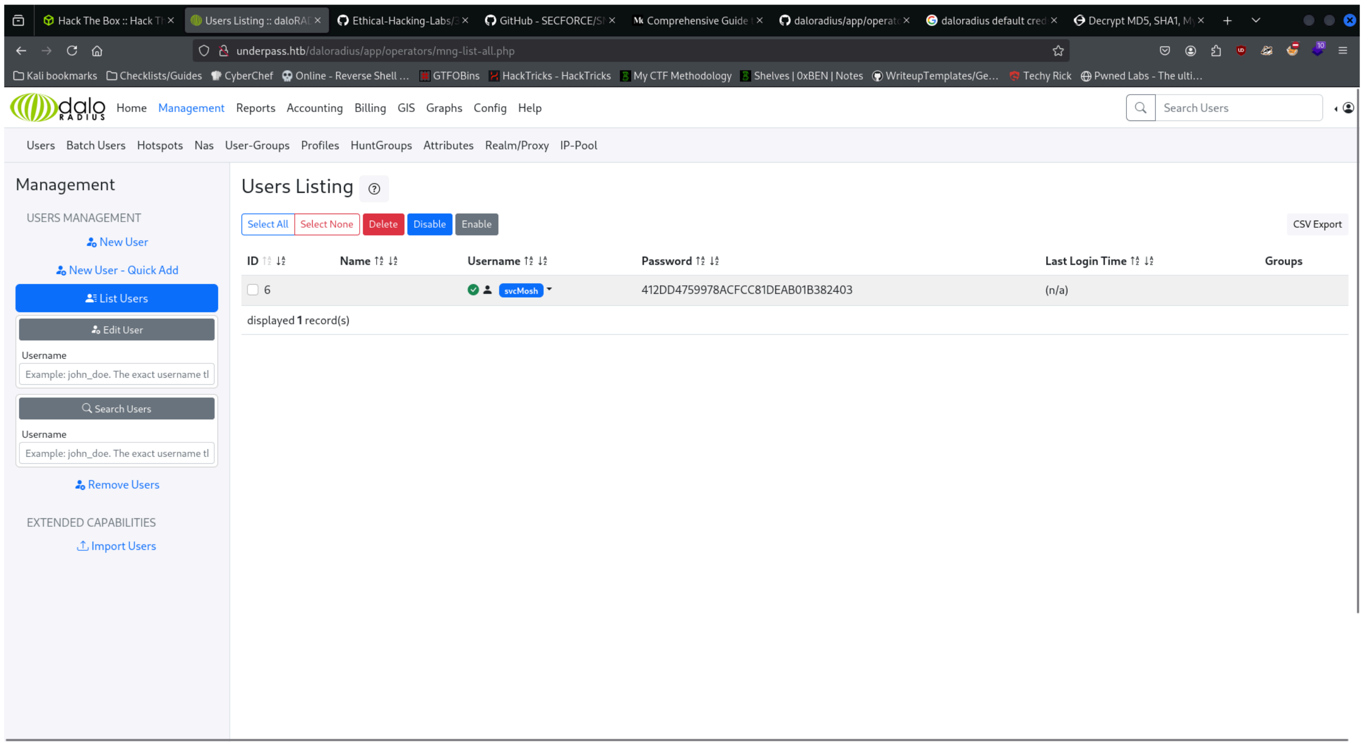


System Configuration

DB Engine	MySQLi
Database Port Number	3306
Database Hostname	localhost
Database User	steve
Database Pass	testing123
Database Name	radius

The username is **steve** and the password is **testing123**, but the database is only listening for connections locally, so we can't access it until we get a shell. Let's keep looking around

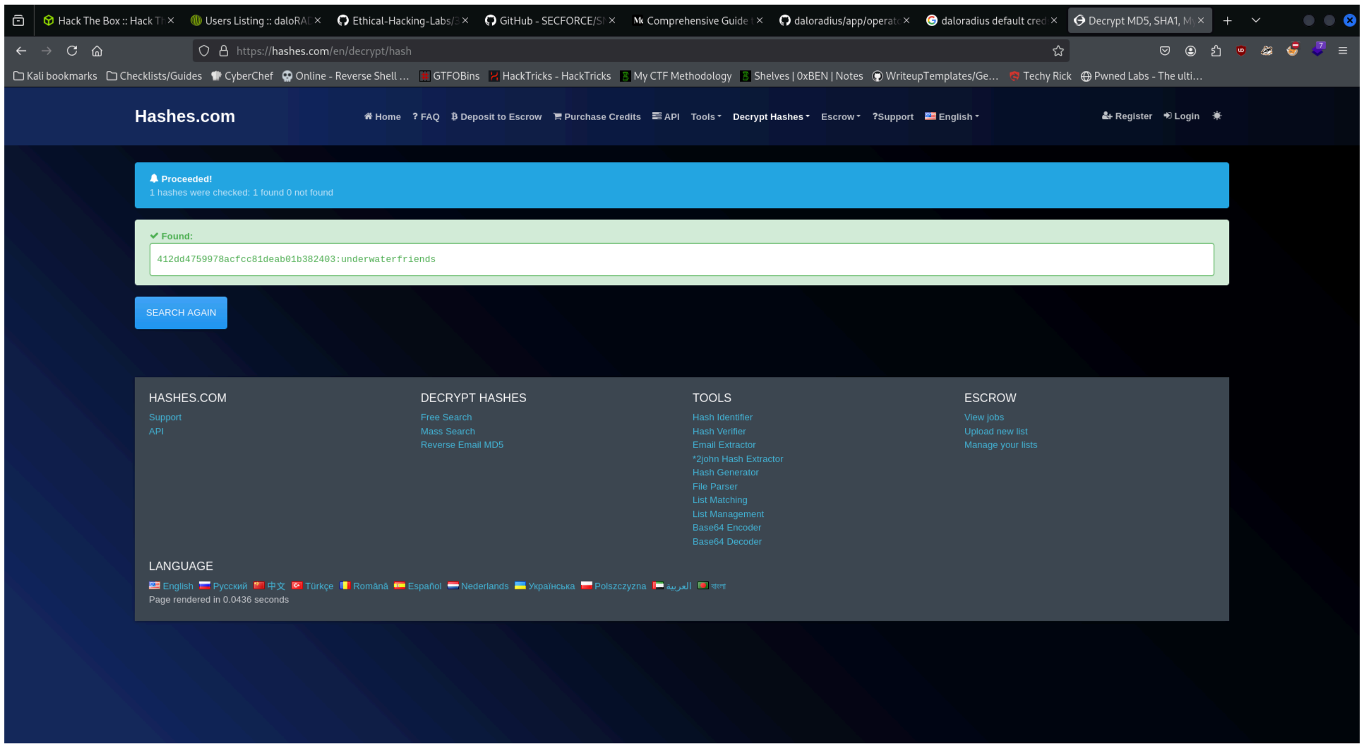
I checked out the management tab and found another user **svcMosh** and their password hash right next to it:



ID	Name	Username	Password	Last Login Time	Groups
6		svcMosh	412DD4759978ACFCC81DEAB01B382403	(n/a)	

displayed 1 record(s)

At a glance, it looks to be an MD5 hash, but I wanted to be sure, so I went over to the hash type identifier at hashes.com and the site confirmed it. I wanted to see if this hash was decrypted in their database and luckily, it was. svcMosh's password is **underwaterfriends**:



Proceeded!
1 hashes were checked: 1 found 0 not found

Found:

```
412dd4759978acfcc81deab01b382403:underwaterfriends
```

I tried to log in as svcMosh in both **/app/operators/login.php** and **/app/users/login.php** but to no avail. However, I was able to **SSH** into the machine with those creds and got user.

Post-Exploit Enumeration

Operating Environment

Current User

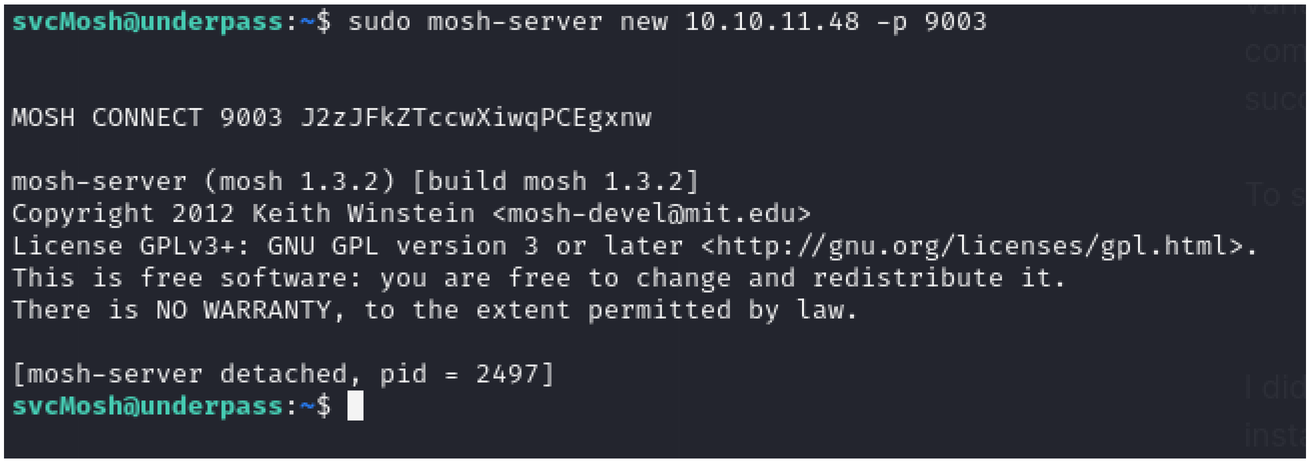
- `id` output: `uid=1002(svcMosh) gid=1002(svcMosh) groups=1002(svcMosh)`
- `sudo -l` output: `Matching Defaults entries for svcMosh on localhost: env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty User svcMosh may run the following commands on localhost: (ALL) NOPASSWD: /usr/bin/mosh-server`

Privilege Escalation

The only thing that was of interest while enumerating the local machine was the sudo privileges of svcMosh which I checked with `sudo -l`. It showed that I can run `/usr/bin/mosh-server` as root without a password.

Mosh (Mobile Shell) is a program similar to SSH in how it allows you to establish a remote shell session, except it operates over UDP instead of TCP and focuses more on low-latency and connection stability than SSH does. When a session is initiated with the included command `mosh-server`, it generates a session key which is used to encrypt communication between the server and the client. The client must define the **MOSH_KEY** environment variable, set it to the session key that the server generated, and use the `mosh-client` command with the IP address of the server and the port it is listening on to connect successfully and get a shell.

To start up the server, I just need to execute `sudo mosh-server new 10.10.11.48 -p 9003`:

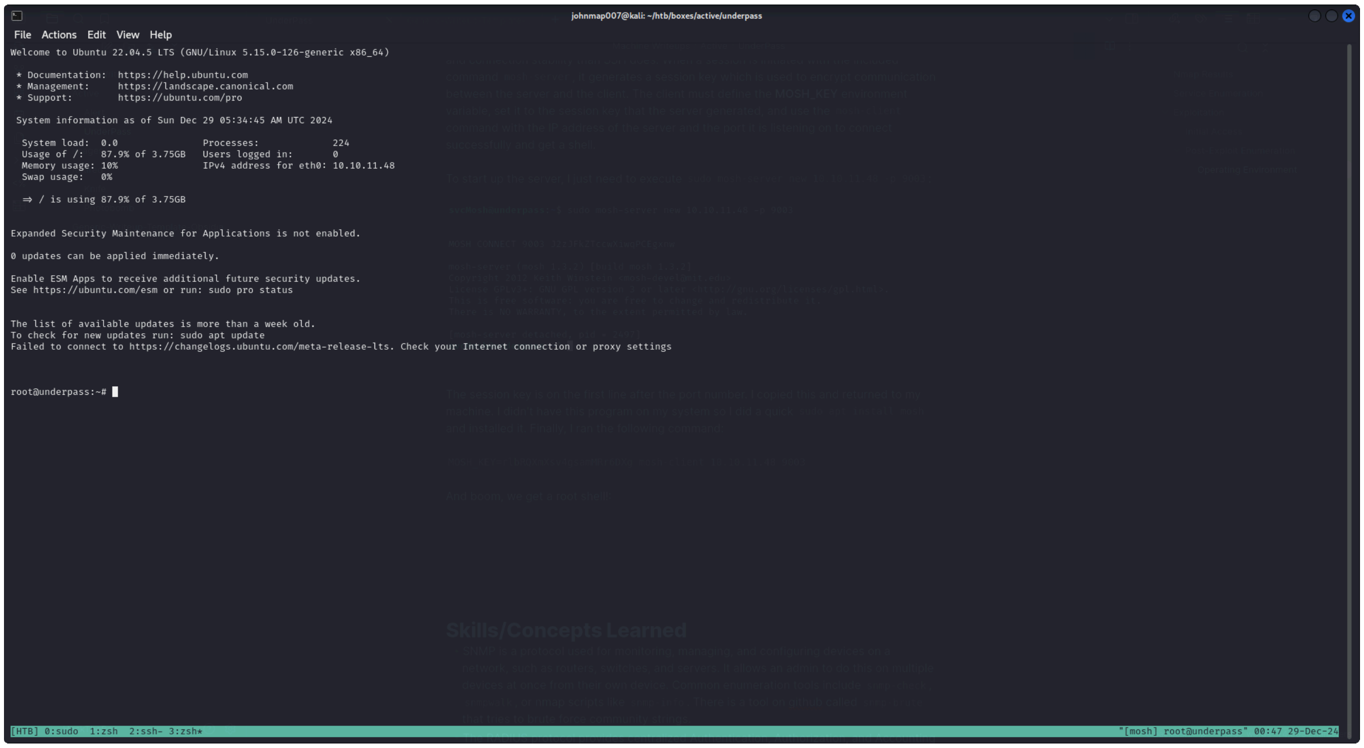


```
svcMosh@underpass:~$ sudo mosh-server new 10.10.11.48 -p 9003


MOSH CONNECT 9003 J2zJFkZTccwXiwqPCEgxnw

mosh-server (mosh 1.3.2) [build mosh 1.3.2]
Copyright 2012 Keith Winstein <mosh-devel@mit.edu>
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

[mosh-server detached, pid = 2497]
svcMosh@underpass:~$
```

Since I started the server as root, I should get a shell as root once I connect. The session key is on the first line after the port number. I copied this and returned to my machine. I didn't have this program on my system so I did a quick `sudo apt install mosh` and installed it. Finally, I ran the following command:

```
MOSH_KEY=rlbRQXmXsv4gsamMRr6DXg mosh-client 10.10.11.48 9003
```

And boom, we get a root shell!


johnmap007@kali: ~/htb/boxes/active/underpass
File Actions Edit View Help

```
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Dec 29 05:34:45 AM UTC 2024

  System load:  0.0               Processes:             224
  Usage of /:   87.9% of 3.75GB   Users logged in:       0
  Memory usage: 10%               IPv4 address for eth0: 10.10.11.48
  Swap usage:   0%

  ⇒ / is using 87.9% of 3.75GB


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings



root@underpass:~#

[HTB] 0:sudo  1:zsh  2:ssh- 3:zsh*                                  "[mosh] root@underpass" 00:47 29-Dec-24
```

Skills/Concepts Learned

- SNMP is a protocol used for monitoring, managing, and configuring devices on a network, such as routers, switches, and servers. It allows an admin to do this on multiple devices at once from their own device. Misconfigurations can lead to sensitive data being exposed, which was the case in this machine. Common enumeration tools include `snmp-check`, `snmpwalk`, or nmap scripts like `snmp-info`. There is a tool on GitHub called `snmp-brute` that tries to brute force community strings.
- The RADIUS protocol provides centralized Authentication, Authorization, and Accounting (AAA) management for users connecting to a network. It's widely used in environments requiring secure access control, such as VPNs and wireless networks and operates via a client-server model where devices like routers or APs act as clients and communicate with a central RADIUS server (see above section titled "Crash course on the RADIUS protocol" for more details)
- Given an open source product on GitHub, look everywhere for important pages, potential sources of valuable info, anything. There were 2 login pages when I thought there was just 1, another reason why enumeration is a very important step.
- Mosh is short for **Mobile Shell**. It's a program like SSH that allows you to obtain a remote shell session, except it operates over UDP instead of TCP. It includes a command `mosh-server`, which is used to start a server and listen for connections. Since this command was allowed to be executed as root, anyone who connects to the server will obtain a root shell.

Proof of Pwn

https://www.hackthebox.com/achievement/machine/391579/641