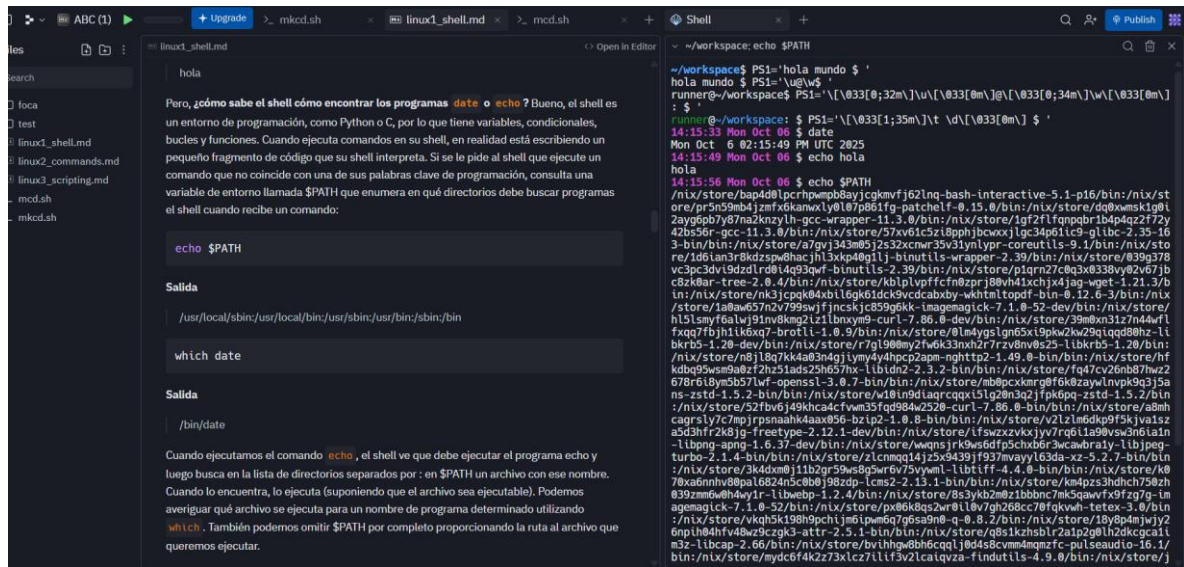


Taller Linux 1



```
#!/bin/sh

hola

Pero, ¿cómo sabe el shell cómo encontrar los programas date o echo? Bueno, el shell es un entorno de programación, como Python o C, por lo que tiene variables, condicionales, bucles y funciones. Cuando ejecuta comandos en su shell, en realidad está escribiendo un pequeño fragmento de código que su shell interpreta. Si se le pide al shell que ejecute un comando que no coincide con una de sus palabras clave de programación, consulta una variable de entorno llamada $PATH que enumera en qué directorios debe buscar programas el shell cuando recibe un comando:

echo $PATH

Salida

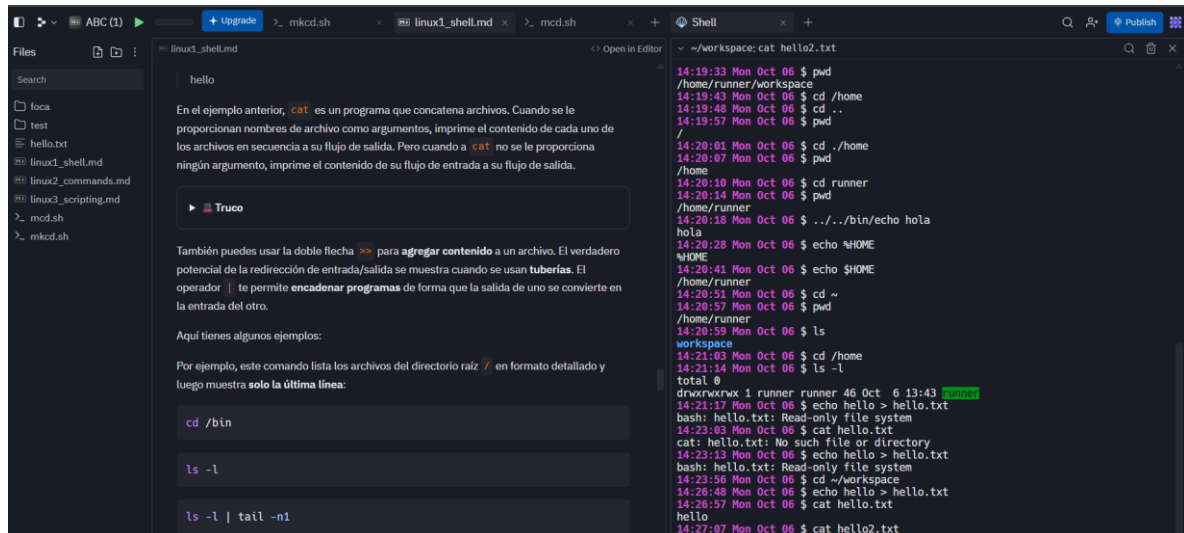
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

which date

Salida

/bin/date

Cuando ejecutamos el comando echo, el shell ve que debe ejecutar el programa echo y luego busca en la lista de directorios separados por : en $PATH un archivo con ese nombre. Cuando lo encuentra, lo ejecuta (suponiendo que el archivo sea ejecutable). Podemos averiguar qué archivo se ejecuta para un nombre de programa determinado utilizando which. También podemos omitir $PATH por completo proporcionando la ruta al archivo que queremos ejecutar.
```



```
#!/bin/sh

hello

En el ejemplo anterior, cat es un programa que concatena archivos. Cuando se le proporcionan nombres de archivo como argumentos, imprime el contenido de cada uno de los archivos en secuencia a su flujo de salida. Pero cuando cat no se le proporciona ningún argumento, imprime el contenido de su flujo de entrada a su flujo de salida.

Truco

También puedes usar la doble flecha >> para agregar contenido a un archivo. El verdadero potencial de la redirección de entrada/salida se muestra cuando se usan tuberías. El operador | te permite encadenar programas de forma que la salida de uno se convierte en la entrada del otro.

Aquí tienes algunos ejemplos:

Por ejemplo, este comando lista los archivos del directorio raíz / en formato detallado y luego muestra solo la última línea:

cd /bin

ls -l

ls -l | tail -n1
```

Files

Search

loca

hacker-lab

scanner.sh

test

hello.txt

linux1_shell.md

linux2_commands.md

linux3_scripting.md

mod.sh

mkcd.sh

3. Cambio de Permisos y Ejecucion del Script

Paso 6: Hacer el Script Ejecutable

Cambia los permisos del script para que pueda ejecutarse.

chmod +x scanner.sh

Paso 7: Ejecutar el Script

Ejecuta el script y analiza los resultados.

./scanner.sh

4. Análisis de Resultados

Paso 8: Interpretar la Salida

- Revisa los archivos listados por `grep` para ver si contienen información sensible como contraseñas.
- Examina los archivos ejecutables encontrados por `find` y evalúa si presentan algún riesgo de seguridad.

5. Desafío Adicional (Opcional)

Paso 9: Mejorar el Script

Shell

```
~/workspace/hacker-lab: ./scanner.sh
14:28:57 Mon Oct 06 $ cd ~/workspace
bash: cd: /home/runner/workspace: No such file or directory
14:29:08 Mon Oct 06 $ pwd
/home/runner/workspace
14:29:13 Mon Oct 06 $ mkdir hacker-lab
14:29:22 Mon Oct 06 $ cd hacker-lab
14:29:27 Mon Oct 06 $ touch scanner.sh
14:30:04 Mon Oct 06 $ code scanner.sh
14:30:17 Mon Oct 06 $ chmod +x scanner.sh
14:31:21 Mon Oct 06 $ ./scanner.sh
Iniciando el escaneo...
/etc/login.defs
grep: /etc/shadow: Permission denied
/etc/default/useradd
/etc/security/pwhistory.conf
grep: /etc/security/opasswd: Permission denied
grep: /etc/gshadow: Permission denied
/etc/debconf.conf
grep: /etc/.pwd.lock: Permission denied
/etc/pam.d/chpasswd
/etc/pam.d/other
/etc/pam.d/newusers
/etc/pam.d/su
/etc/pam.d/su-l
/etc/pam.d/common-password
/etc/pam.d/chsh
/etc/pam.d/login
/etc/pam.d/chfn
/etc/pam.d/passwd
grep: /etc/gshadow: Permission denied
grep: /etc/shadow: Permission denied
grep: /etc/ssl/private: Permission denied
/etc/ssl/openssl.cnf
grep: /var/cache/ldconfig: Permission denied
grep: /var/cache/apt/archives/lock: Permission denied
grep: /var/cache/debconf/passwords.dat: Permission denied
/var/cache/debconf/templates.dat
/var/cache/debconf/templates.dat~old
/var/log/bootstrap.log
```