**Escuela Colombiana De Ingeniería**

**Julio Garavito**

**Seguridad y privacidad TI**

**Daniel Esteban Vela Lopez**

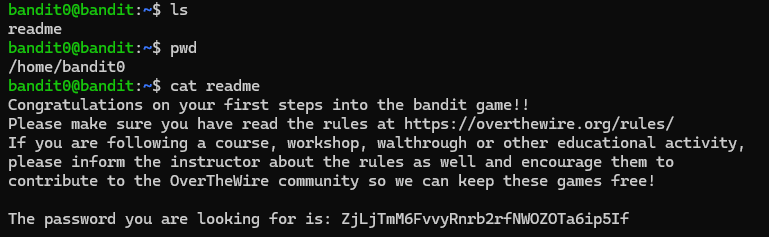
**Andres Felipe Montes**

**Laura Valentina Rodriguez Ortegon**

**Laboratorio No.2**

**2024-2**

Nivel 0 = ZjLjTmM6FvvyRnrb2rfNWOZOTa6ip5If

para el nivel 0 ya nos encontrábamos en el directorio de inicio y se utilizó el comando ls para listar los directorios que se encuentran dentro de la carpeta, luego usamos el pwd para mostrar el directorio de trabajo actual y por último cat readme, el cual nos muestra el contenido de los archivos



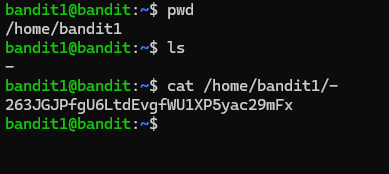
Nivel 1= 263JGJPfgU6LtdEvgfWU1XP5yac29mFx

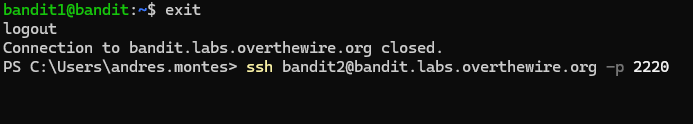
Para el nivel 1 vamos a ingresar con la contraseña del primer nivel y aplicamos los comandos anteriores, según el orden para mostrar los archivos dentro del directorio, buscamos nuestro archivo “-” pero con la ruta completa, entonces va ir desglosando desde el

/home

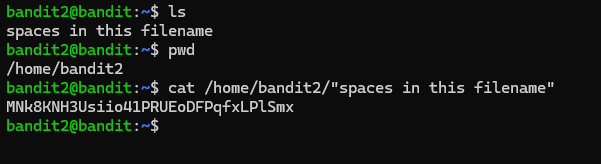
/bandit1

/-





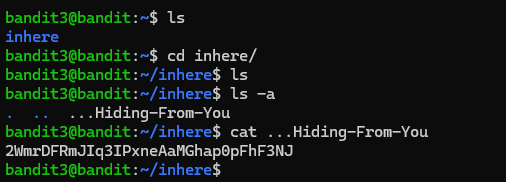
Nivel 2 = MNk8KNH3Usiio41PRUEoDFPqfxLPlSmx

Para este nivel 2, vamos a tener en cuenta la dirección de la ruta para bandit2 y como es un nombre con espacios debería ir entre comillas para que no afecten los espacios y no agregue otros símbolos especiales a la hora de la búsqueda, si no que más bien va tomar el nombre textualmente.



Nivel 3 = 2WmrDFRmJIq3IPxneAaMGhap0pFhF3NJ

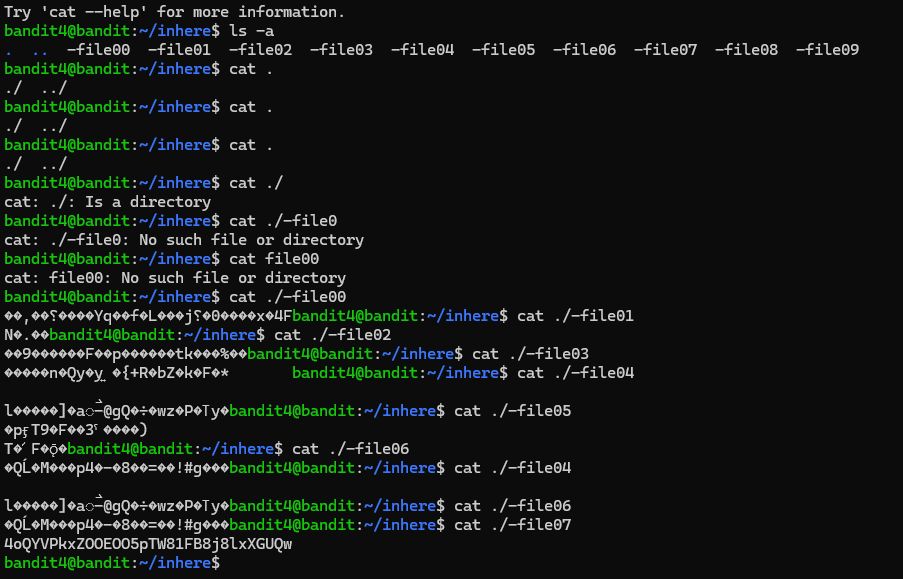
Para el nivel 3 seguimos en el directorio de inhere y con el comando ls -a vamos a listar los directorios que están ocultos, en este caso nos va salir el archivo oculto . .. …Hiding-From-You, lo leemos y nos dara la contraseña correspondiente





Nivel 4= 4oQYVPkxZOOEOO5pTW81FB8j8lxXGUQw

Volvemos a realizar el pasó anterior para listar los directorios ocultos desde el /bandit3, pero acá empezamos a revisar los archivos para localizar la contraseña y nos damos cuenta que estan corruptos, entonces buscamos el que nos muestre la contraseña correcta, hasta encontrarla en el archivo -file07, mostraremos la información con el comando de cat ./-file07, lo buscamos así para que no tenga inconvenientes por el símbolo especial





Nivel 5 = HWasnPhtq9AVKe0dmk45nxy20cvUa6EG

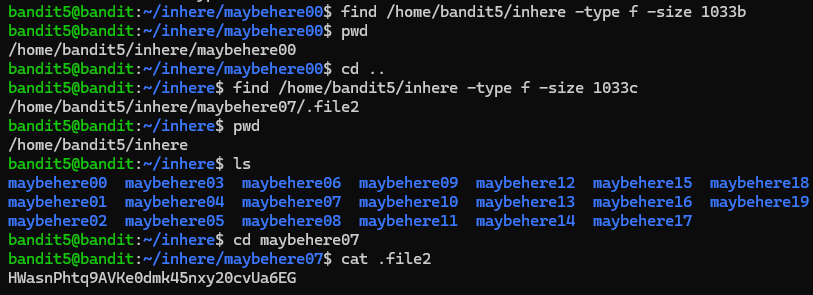
Como las búsquedas se hacen con muchos más archivos de los que teníamos antes, entonces ahora vamos a implementar la búsqueda con comando find rutaDelArchivo -type f -size 1033c, donde la idea es buscar los parámetros que se dan desde la busqueda hasta el tamaño de archivo, en este caso nos vamos a la ruta de

/home

/bandit5

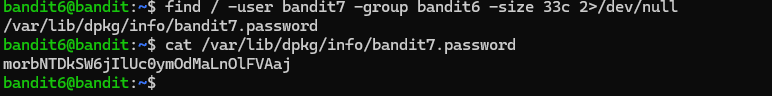
/inhere

y desde ahí va buscar el archivo correspondiente, obteniendo el archivo de 1033 bytes correspondiente



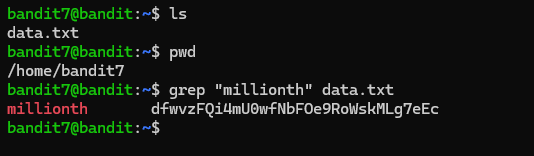
Nivel = 6 morbNTDkSW6jIlUc0ymOdMaLnOlFVAaj

Acá vamos a filtrar por el usuario, grupo y tamaño del archivo. Donde pedimos que busque cada información en el directorio deseado y al final del comando el redirecciona la salida estándar de errores (stderr), al dispositivo nulo (/dev/null)



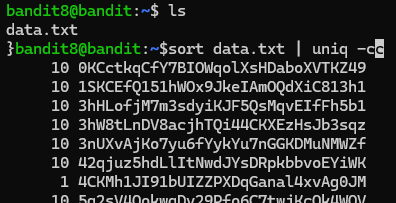
Nivel 7 = dfwvzFQi4mU0wfNbFOe9RoWskMLg7eEc

Para el nivel 7, vamos a usar grep en este caso la palabra millionth será nuestro filtro en el archivo .txt, donde va buscar esa palabra en específico y nos mostrará la contraseña para el siguiente nivel.



Nivel 8 = 4CKMh1JI91bUIZZPXDqGanal4xvAg0JM

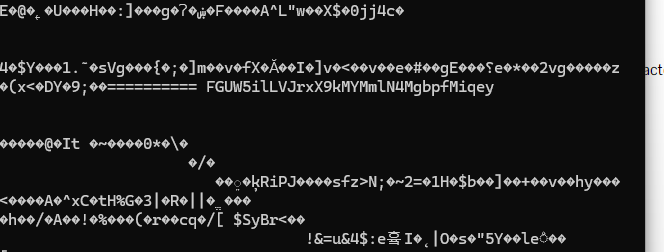
Para este punto vamos a buscar el archivo data.txt y vamos agrupar y ordenar las líneas con el comando sort, asegurando que se muestran el número de todas las líneas duplicadas con -cc, ya que elimina las líneas duplicadas y a la vez las va contando sobre el archivo data.txt.





Nivel 9 = FGUW5ilLVJrxX9kMYMmlN4MgbpfMiqey

Para este nivel solo es necesario leer la información del archivo data.txt con el comando cat y a partir de lo que se muestra es fácil descifrar la contraseña porque la otra información está corrupta, así que no hay que evaluar muchos puntos para resolver cual es la contraseña.





Nivel 10 = dtR173fZKb0RRsDFSGsg2RWnpNVj3qRr

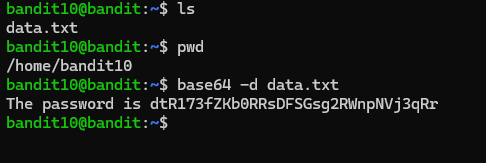
Primero revisamos en qué parte del sistema de archivos nos encontramos y cuál archivo está dentro de este directorio, en este caso será

/home

/bandit10

->data.txt

Ya que tenemos esto y tenemos datos que se encuentran codificados en base 64, entonces utilizamos el comando para decodificar datos que están codificados en base 64, entonces lo que hace que la información del archivo sea una distinta pero a la hora de descodificar la será la que se muestra en pantalla porque eso requerimos.

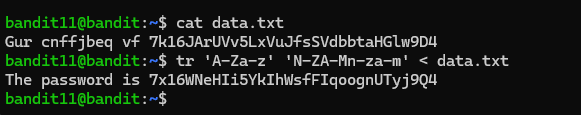




Nivel 11 = 7x16WNeHIi5YkIhWsfFIqoognUTyj9Q4

En este nivel leemos nuestro archivo y nos podemos dar cuenta que no hay ningún patrón ordenado, así que lo que se debe hacer es empezar a seguir la forma en que está almacenada la información, como no lo dice en el enunciado, “todas las letras minúsculas (az) y mayúsculas (AZ) se han rotado 13 posiciones”, entonces en este comando que se ha utilizado es para realizar una transformación de texto en un archivo, aplicando una técnica conocida como cifrado ROT13 . Este tipo de cifrado es un caso especial de un cifrado, donde cada letra del alfabeto es sustituida por la letra que se encuentra 13 posiciones después en el alfabeto.

Lo que está haciendo es transformar caracteres en la entrada estándar, la cual vendría siendo desde la a hasta la z, desde la entrada que se da que es de rotación cada 13 posiciones, haciendo que el contenido se organice como se esperaría del archivo data.txt





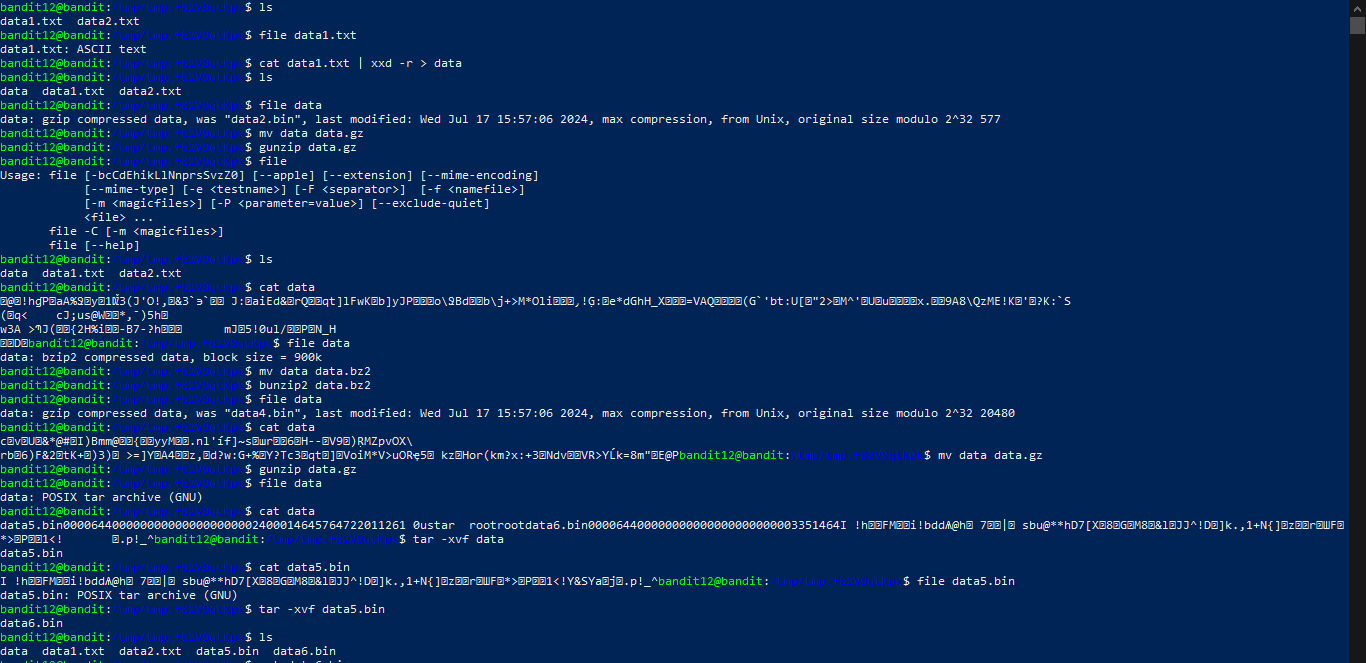
Nivel 12 = FO5dwFsc0cbaIiH0h8J2eUks2vdTDwAn

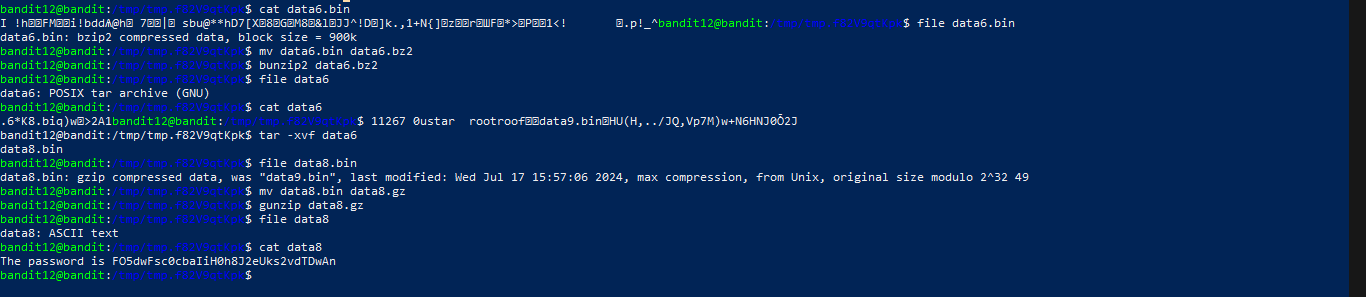
Para el nivel 12 se sugiere a los participantes crear un directorio temporal para poder trabajar más libremente, entonces eso fue lo primero que se hizo con el comando mktemp.

para posteriormente copiar el archivo data.txt al directorio temporal seguido de eso se usó el comando file para identificar el tipo de compresión que se había realizado sobre el archivo lo cual nos dio como respuesta un volcado hexadecimal entonces con el comando xxd -r se revirtió ese volcado y se renombro el archivo como data.

Después de eso se vuelve a usar el comando file para verificar si data seguía comprimido y de qué forma se había hecho, el método de compresión esta vez fue gzip por que se renombro el archivo con un .gz para descomprimirlo utilizando el comando gunzip dando como resultado otra vez data

En resumen y para no alargarnos demas se realizaron varias descompresiones mas pasando por gzip, bzip2 y tar hasta llegar a la contraseña de este nivel,

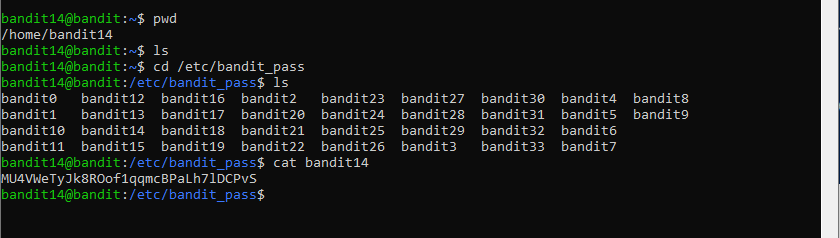




Nivel 13 = MU4VWeTyJk8ROof1qqmcBPaLh7lDCPvS

Es importante establecer conexión de forma segura a través de SSH a un servidor remoto, para este caso el mismo localhost, el cual va utilizar una clave privada en vez de ser una contraseña. con el fin de utilizar un nivel adicional de seguridad, luego de esto si podemos ubicarnos en el directorio de /etc/bandit\_pass/bandit14, acá ingresaremos a este directorio, leeremos la info de bandit14; la contraseña para el siguiente nivel y la encontraremos

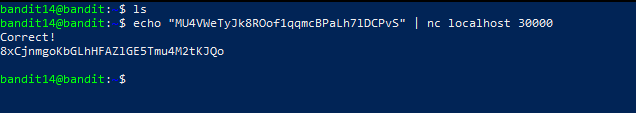






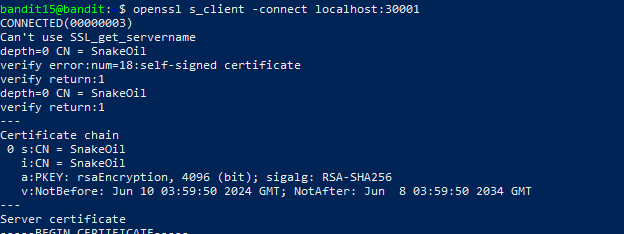
Nivel 14 = 8xCjnmgoKbGLhHFAZlGE5Tmu4M2tKJQo

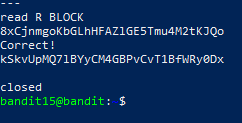
En este nivel usamos el comando el cual se utiliza para enviar un mensaje, que en este caso es la contraseña, que se usó para entrar al nivel, a un servicio o aplicación que está escuchando en un puerto específico; donde es el 30000 en la máquina local (localhost) utilizando nc (Netcat), este puede leer y escribir datos a través de conexiones de red utilizando TCP o UDP, en este caso va a leer la contraseña.



Nivel 15 = 8xCjnmgoKbGLhHFAZlGE5Tmu4M2tKJQo

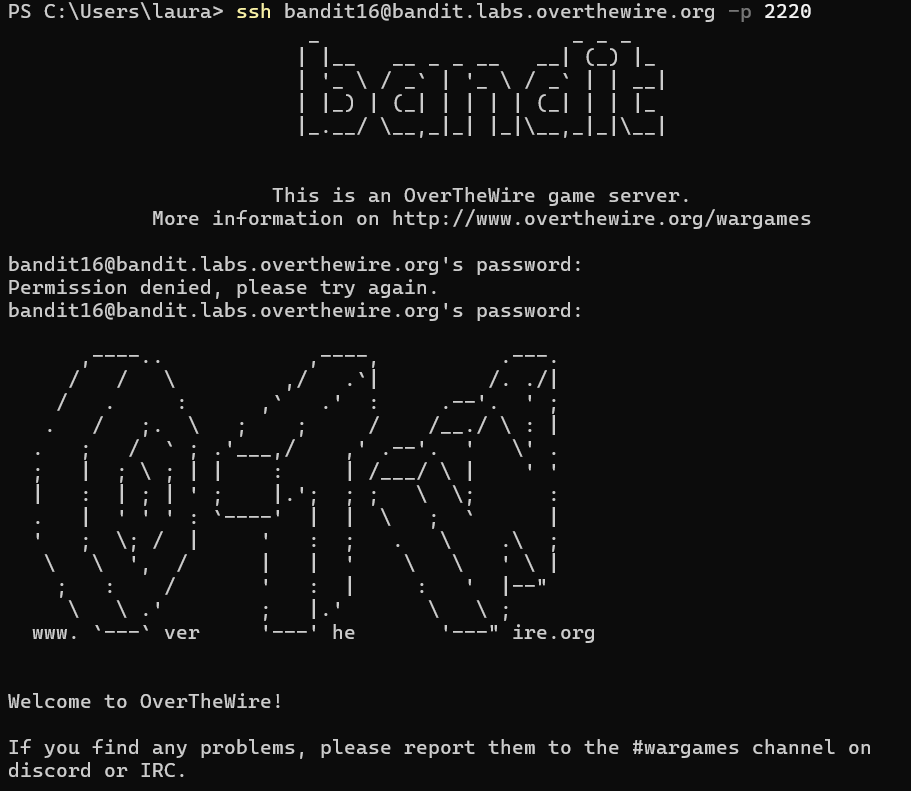
Al ejecutar openssl s\_client -connect localhost:30001, el comando establece una conexión SSL/TLS con el servicio que escucha en el puerto 30001 de la máquina local, permitiendo verificar que el servicio esté operando correctamente y que la configuración SSL/TLS sea válida. Además, proporciona detalles sobre la conexión, como el certificado del servidor, la cadena de certificados, y las suites de cifrado utilizadas. Una vez establecida la conexión, es posible interactuar con el servicio de forma segura, enviando y recibiendo datos a través de esta conexión, en este caso nuestro dato importante es la contraseña para el siguiente nivel.

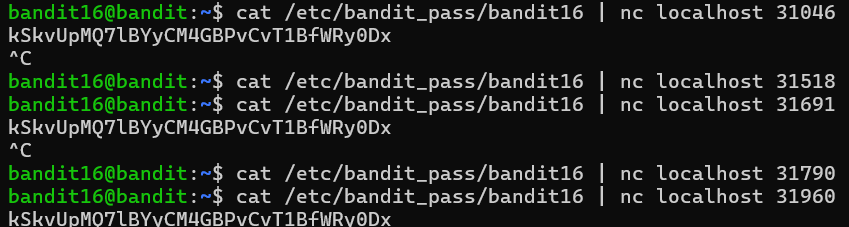
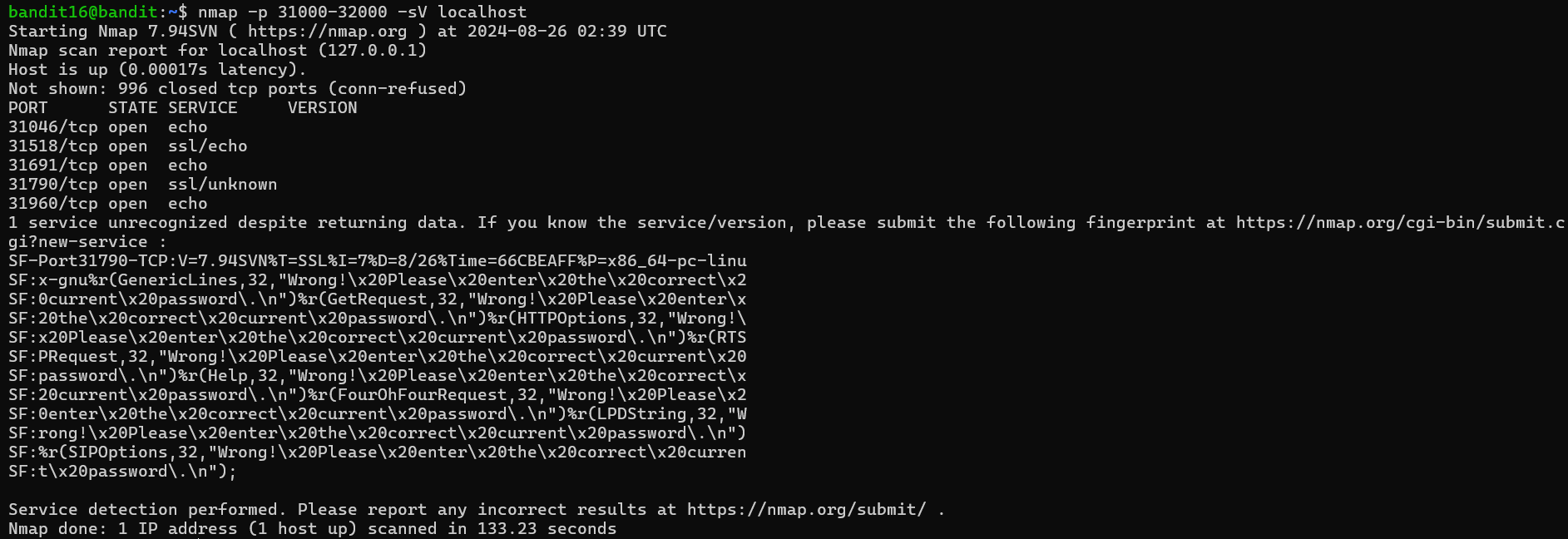


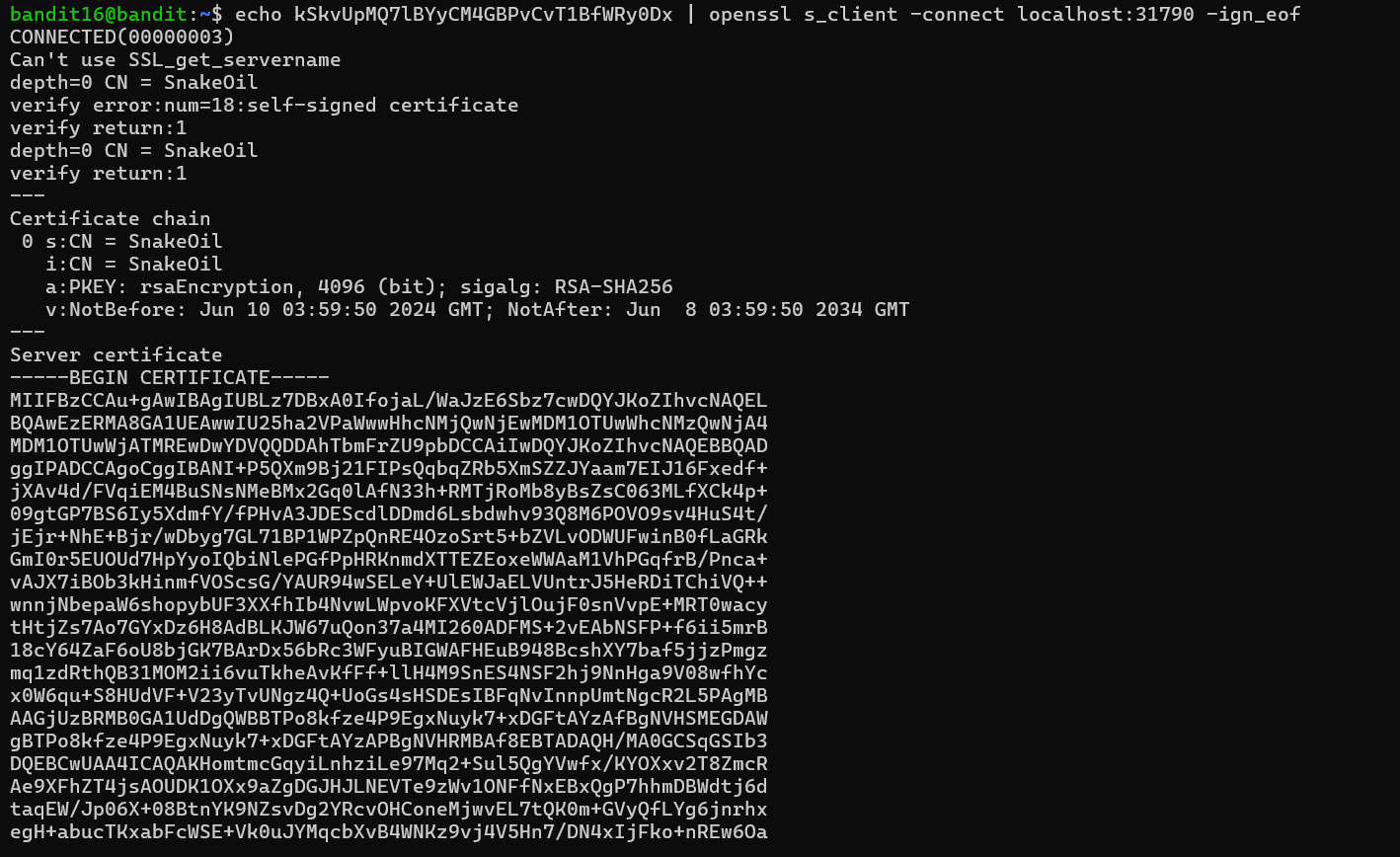


Nivel 16 = kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx

* Para el nivel 16 tiene varios puntos a tener en cuenta, primero debemos escanear el rango de puertos para realizar una búsqueda de cuáles están activos con nmap.
* Como hay 5 puertos habilitados, entonces verificamos cuales están en funcionamiento y cuáles no.
* En este caso, sabemos que el 31518 y 31790 están utilizando SSL, así que nos vamos con el puerto 31790 e implementamos un comando parecido del punto 14, donde tiene echo “contraseña” | openssl s\_client -connect localhost: 31790 -ign\_eof, con el fin de permitir enviar una cadena de texto a un servidor a través de una conexión SSL/TLS segura, manteniendo la conexión abierta para una mayor interacción con el servicio. Esto es útil para probar conexiones seguras, enviar datos sensibles, y recibir respuestas del servidor sin cerrar la conexión inmediatamente.
* Esto nos da un certificado de servidor, lo copiamos y creamos un directorio donde contenga un archivo para guardar la información, damos permisos al archivo de lectura y escritura y al crear la conexión con el ssh, ya estaremos en bandit17.
* Y podremos obtener la contraseña en /etc/bandit\_pass/bandit17







-----BEGIN RSA PRIVATE KEY-----

MIIEogIBAAKCAQEAvmOkuifmMg6HL2YPIOjon6iWfbp7c3jx34YkYWqUH57SUdyJ

imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMlOJf7+BrJObArnxd9Y7YT2bRPQ

Ja6Lzb558YW3FZl87ORiO+rW4LCDCNd2lUvLE/GL2GWyuKN0K5iCd5TbtJzEkQTu

DSt2mcNn4rhAL+JFr56o4T6z8WWAW18BR6yGrMq7Q/kALHYW3OekePQAzL0VUYbW

JGTi65CxbCnzc/w4+mqQyvmzpWtMAzJTzAzQxNbkR2MBGySxDLrjg0LWN6sK7wNX

x0YVztz/zbIkPjfkU1jHS+9EbVNj+D1XFOJuaQIDAQABAoIBABagpxpM1aoLWfvD

KHcj10nqcoBc4oE11aFYQwik7xfW+24pRNuDE6SFthOar69jp5RlLwD1NhPx3iBl

J9nOM8OJ0VToum43UOS8YxF8WwhXriYGnc1sskbwpXOUDc9uX4+UESzH22P29ovd

d8WErY0gPxun8pbJLmxkAtWNhpMvfe0050vk9TL5wqbu9AlbssgTcCXkMQnPw9nC

YNN6DDP2lbcBrvgT9YCNL6C+ZKufD52yOQ9qOkwFTEQpjtF4uNtJom+asvlpmS8A

vLY9r60wYSvmZhNqBUrj7lyCtXMIu1kkd4w7F77k+DjHoAXyxcUp1DGL51sOmama

+TOWWgECgYEA8JtPxP0GRJ+IQkX262jM3dEIkza8ky5moIwUqYdsx0NxHgRRhORT

8c8hAuRBb2G82so8vUHk/fur85OEfc9TncnCY2crpoqsghifKLxrLgtT+qDpfZnx

SatLdt8GfQ85yA7hnWWJ2MxF3NaeSDm75Lsm+tBbAiyc9P2jGRNtMSkCgYEAypHd

HCctNi/FwjulhttFx/rHYKhLidZDFYeiE/v45bN4yFm8x7R/b0iE7KaszX+Exdvt

SghaTdcG0Knyw1bpJVyusavPzpaJMjdJ6tcFhVAbAjm7enCIvGCSx+X3l5SiWg0A

R57hJglezIiVjv3aGwHwvlZvtszK6zV6oXFAu0ECgYAbjo46T4hyP5tJi93V5HDi

Ttiek7xRVxUl+iU7rWkGAXFpMLFteQEsRr7PJ/lemmEY5eTDAFMLy9FL2m9oQWCg

R8VdwSk8r9FGLS+9aKcV5PI/WEKlwgXinB3OhYimtiG2Cg5JCqIZFHxD6MjEGOiu

L8ktHMPvodBwNsSBULpG0QKBgBAplTfC1HOnWiMGOU3KPwYWt0O6CdTkmJOmL8Ni

blh9elyZ9FsGxsgtRBXRsqXuz7wtsQAgLHxbdLq/ZJQ7YfzOKU4ZxEnabvXnvWkU

YOdjHdSOoKvDQNWu6ucyLRAWFuISeXw9a/9p7ftpxm0TSgyvmfLF2MIAEwyzRqaM

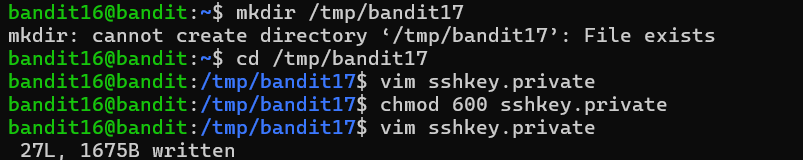
77pBAoGAMmjmIJdjp+Ez8duyn3ieo36yrttF5NSsJLAbxFpdlc1gvtGCWW+9Cq0b

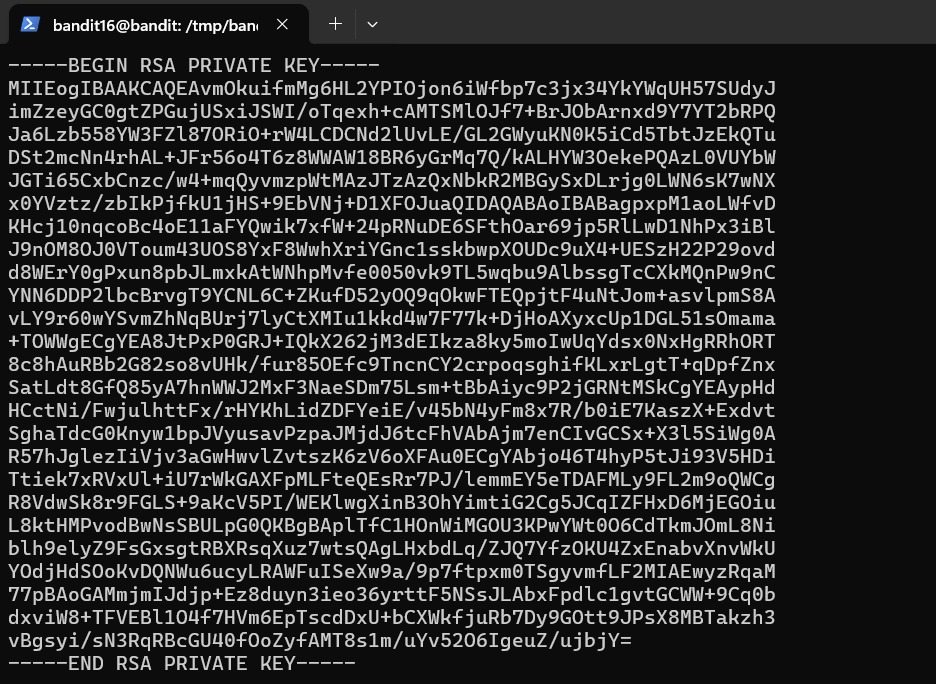
dxviW8+TFVEBl1O4f7HVm6EpTscdDxU+bCXWkfjuRb7Dy9GOtt9JPsX8MBTakzh3

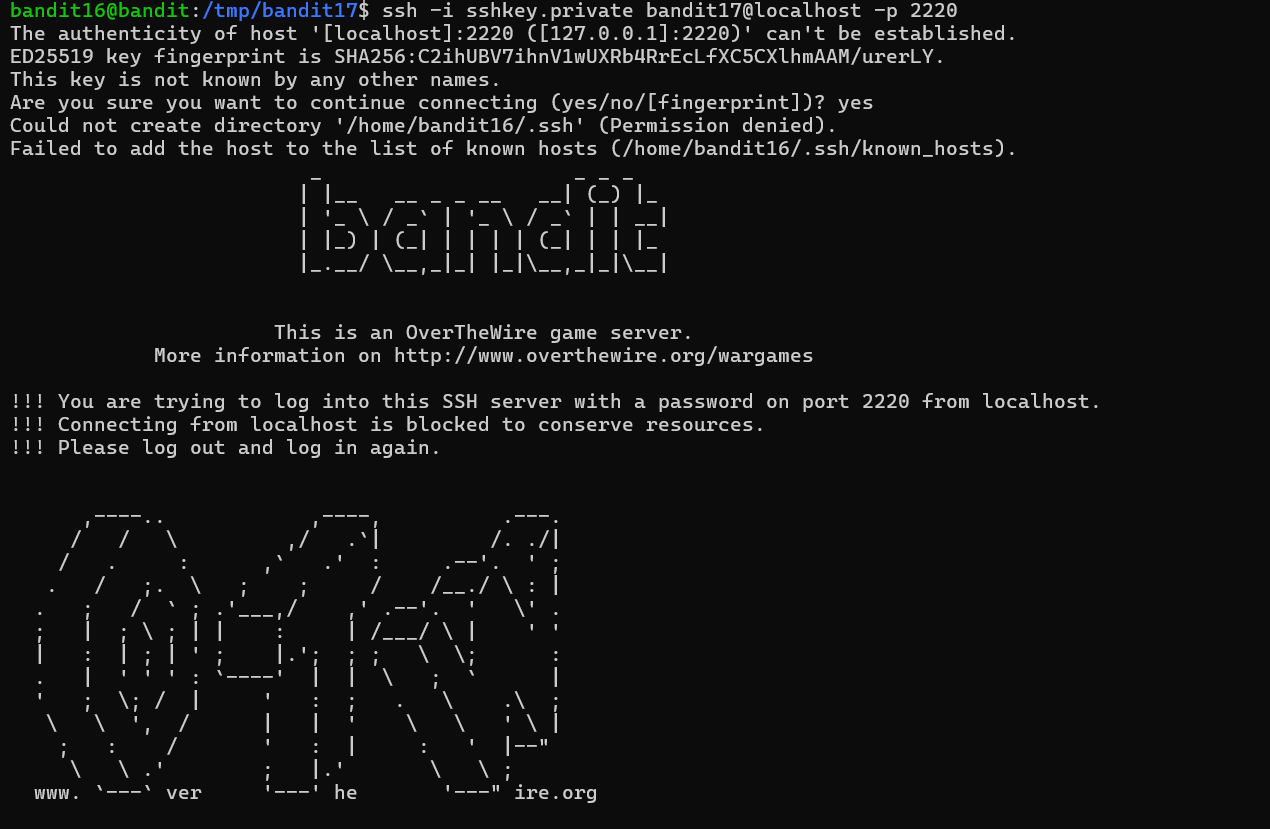
vBgsyi/sN3RqRBcGU40fOoZyfAMT8s1m/uYv52O6IgeuZ/ujbjY=

-----END RSA PRIVATE KEY-----

****

****

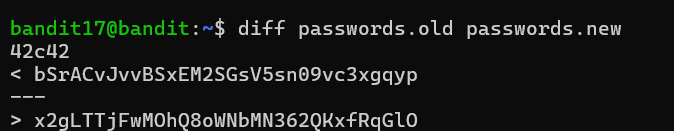
****

****

Nivel 17 = EReVavePLFHtFlFsjn3hyzMlvSuSAcRD

Para este nivel utilizamos el comando diff passwords.old passwords.new, el cual se utiliza para comparar dos archivos de texto, en este caso passwords.old y passwords.new, y mostrar las diferencias entre ellos. diff es una herramienta de línea de comandos que examina los archivos línea por línea y destaca las modificaciones entre versiones. Esto resulta útil para ver cómo ha cambiado el contenido entre dos versiones de un archivo, ya sea código fuente, configuraciones, o listas. En este contexto, passwords.old contiene la versión anterior de los datos de contraseñas, mientras que passwords.new tiene la versión más reciente.

****



La contraseña es la nueva o la última para el siguiente nivel

Nivel 18 = x2gLTTjFwMOhQ8oWNbMN362QKxfRqGlO

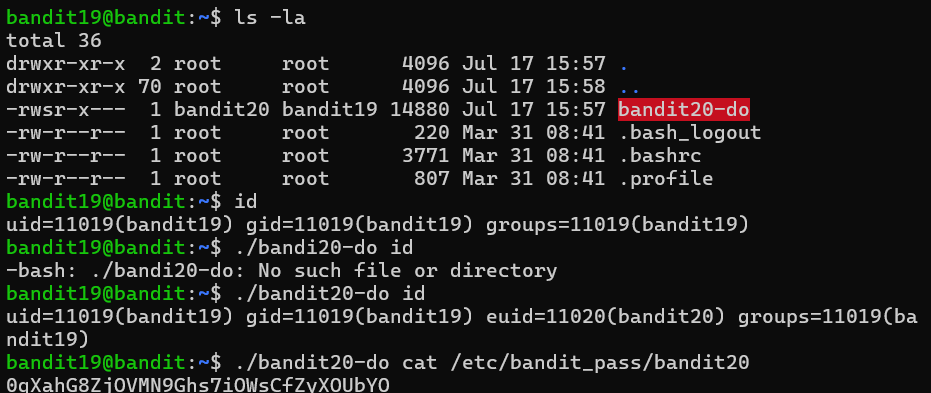
Tener en cuenta que para el comando es importante agregarle el “/bin/bash” al final de entrar al nivel, o se saldrá de la conexión automáticamente.

De resto, solo es utilizar los comandos que ya conocemos y con ello leer el readme, para que nos de la contraseña.



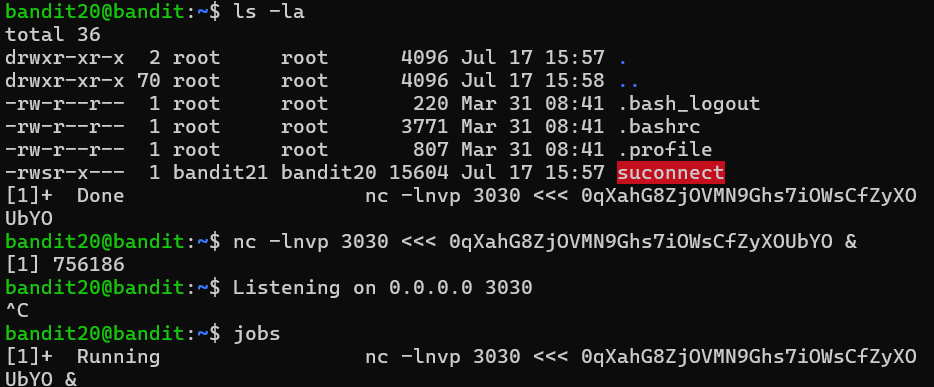
Nivel 19 = cGWpMaKXVwDUNgPAVJbWYuGHVn9zl3j8

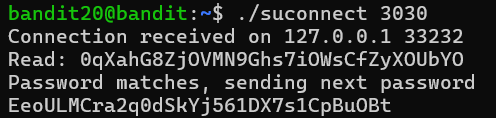
Primero debemos listar los archivos y directorios, mostrando así la información adicional; incluso los archivos ocultos. Luego, con el comando id mostramos la información del usuario actual y los grupos del cual pertenece el usuario. Así como ya tenemos nuestro archivo bandit20-do subrayado con anterioridad, se ejecuta este archivo en el directorio actual y pasará el id como argumento a ese archivo. Así podremos leer la contraseña en el directorio específico.



Nivel 20 = 0qXahG8ZjOVMN9Ghs7iOWsCfZyXOUbYO

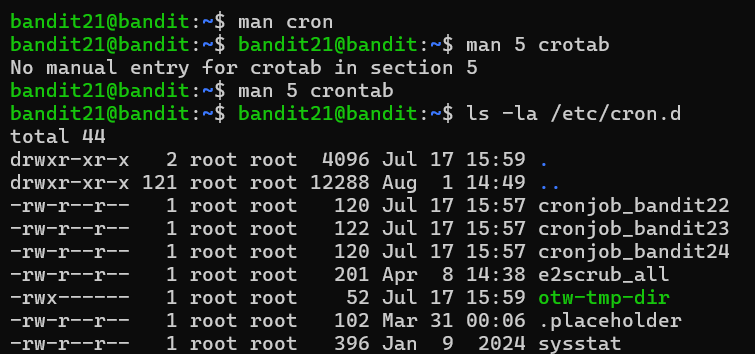
El comando ls -la lista los archivos y directorios en el directorio actual, mostrando detalles completos, incluidos los archivos ocultos. Luego, nc -nnvp 3030 <<< "contraseña" & usa nc (netcat) para iniciar un servidor en el puerto 3030 que escucha conexiones entrantes y envía "contraseña" a cualquier cliente que se conecte; el símbolo & ejecuta este comando en segundo plano. La salida "Listening on 0.0.0.0 3030" confirma que el servidor está activo y escuchando conexiones en todas las interfaces de red. El comando jobs lista los procesos en segundo plano, como el que se ejecutó con nc. Finalmente, ./subconect 3030 ejecuta un archivo o script llamado subconect con el puerto 3030 como argumento, para conectarse al servidor que se ha establecido con nc. Así que se estaría configurando un servidor de red que escucha en el puerto 3030, mientras que otro comando se prepara para interactuar con él

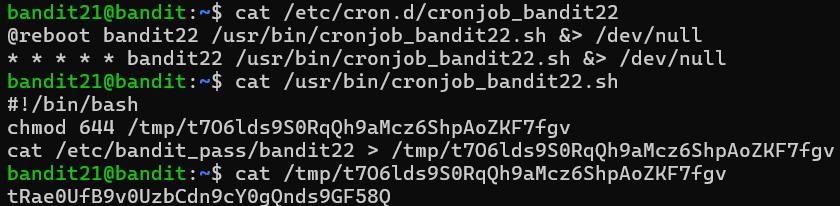




Nivel 21 = EeoULMCra2q0dSkYj561DX7s1CpBuOBt

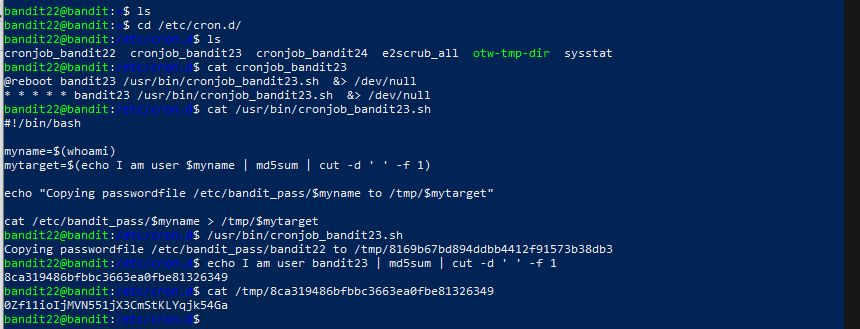
Primero, man cron y man 5 crontab proporcionan información sobre el funcionamiento de cron y el formato de los archivos crontab, respectivamente. Luego, ls -la /etc/cron.d lista los archivos en el directorio donde se almacenan las configuraciones de cron, permitiéndo ver qué tareas están programadas. Con cat /etc/cron.d/cronjob\_bandit22, se examina el contenido de un archivo de configuración de cron específico para entender qué tarea se ejecuta. Posteriormente, cat /usr/bin/cronjob\_bandit22.sh muestra el script asociado a esa tarea programada para revelar qué acciones realiza. Finalmente, cat /tmp/"dato dado anteriormente" revisa el contenido de un archivo temporal, cuyo nombre se refiere a información previamente obtenida. En conjunto, estos comandos ayudan a entender y verificar la configuración y ejecución de tareas automáticas en el sistema, para así poder obtener la contraseña del siguiente nivel





Nivel 22 = tRae0UfB9v0UzbCdn9cY0gQnds9GF58Q

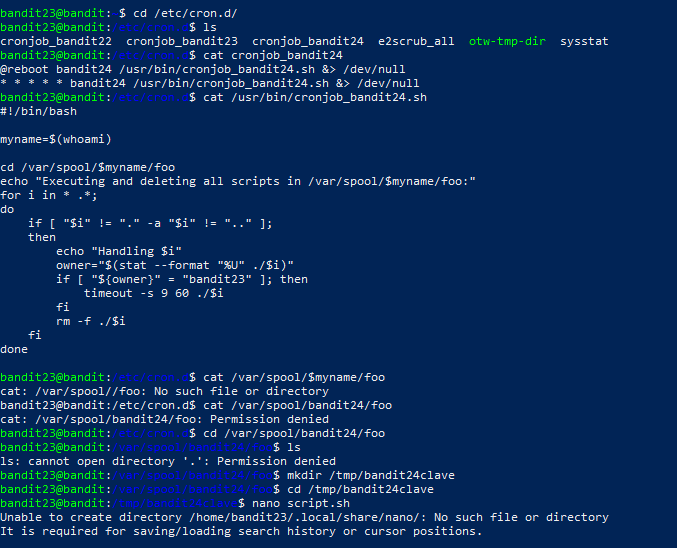
Se debe identificar un trabajo cron que se ejecuta automáticamente en el sistema. Comienza listando y accediendo al directorio /etc/cron.d/, donde están configurados los trabajos cron. Luego, se inspecciona el archivo cronjob\_bandit que define un trabajo cron específico, el cual ejecuta un script ubicado en /usr/bin/cronjob\_bandit23.sh. Al revisar el contenido de este script, puedes observar que genera un hash MD5 basado en una cadena específica. Este hash es la clave, el cual está contenido en /tmp/.

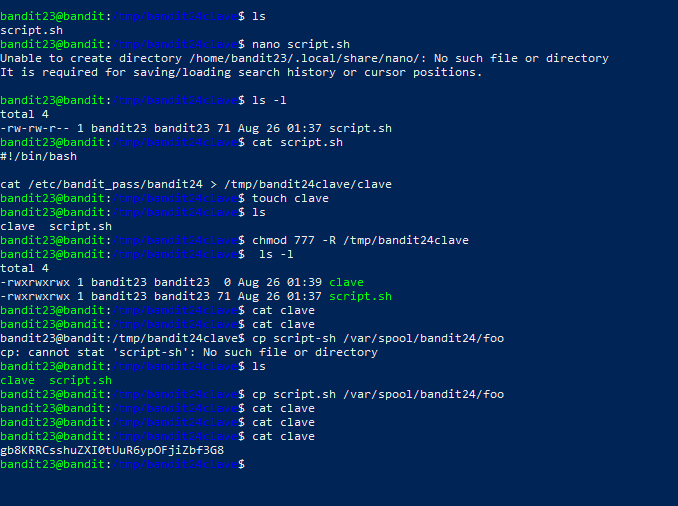


Nivel 23 = 0Zf11ioIjMVN551jX3CmStKLYqjk54Ga

Para pasar este nivel nos dirigimos al mismo directorio de donde sacamos las contraseñas anteriores y notamos que el sh está apuntando al usuario de bandit 24 por lo que creamos un script en la carpeta temporal y un archivo para guardar la clave.

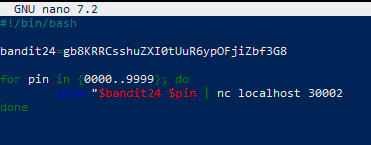
El script en cuestión es bastante simple solo es un apuntador que guarda la clave de bandit24 en el archivo que creamos en el directorio temporal de bandit23

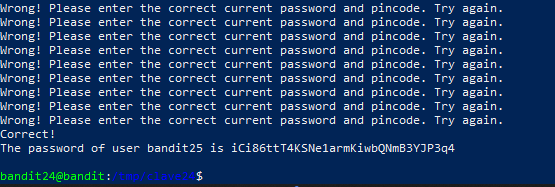




Nivel 24 = gb8KRRCsshuZXI0tUuR6ypOFjiZbf3G8

Para este nivel es necesario crear un archivo en /bin/bash, donde debemos colocar la contraseña con la que ingresamos a este nivel y así poder iterar sobre todos los rangos que van de 0000 a 9999 y así poder envíar el PIN y la contraseña, de esta forma podremos obtener la contraseña del siguiente nivel

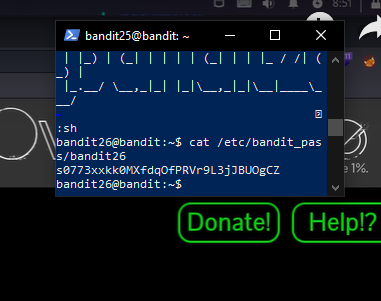




Nivel 25 = iCi86ttT4KSNe1armKiwbQNmB3YJP3q4

Para este caso solo realizaremos la conexión a SSH al servidor que se ejecuta en localhost, autenticandose como el usuario bandit26 utilizando una clave privada almacenada en el archivo bandit26.sshkey; donde la conexión se realiza a través del puerto 2220; luego solo es leer el archivo para la contraseña de bandit26 y ajustar el tamaño de pantalla para poder establecer la contraseña del siguiente nivel.





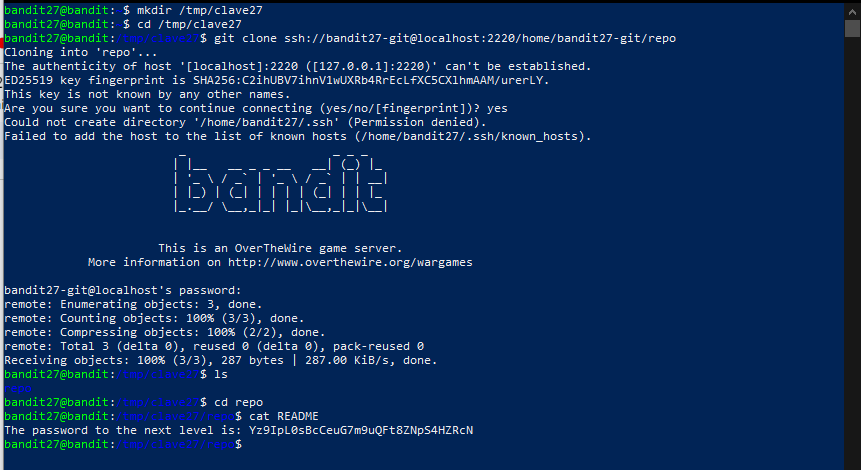
Nivel 26 = s0773xxkk0MXfdqOfPRVr9L3jJBUOgCZ

para este nivel seguimos aprovechando el “bug” de more que nos permite ejecutar comandos en vi y por medio del directorio de pass obtenemos la clave de bandit 27



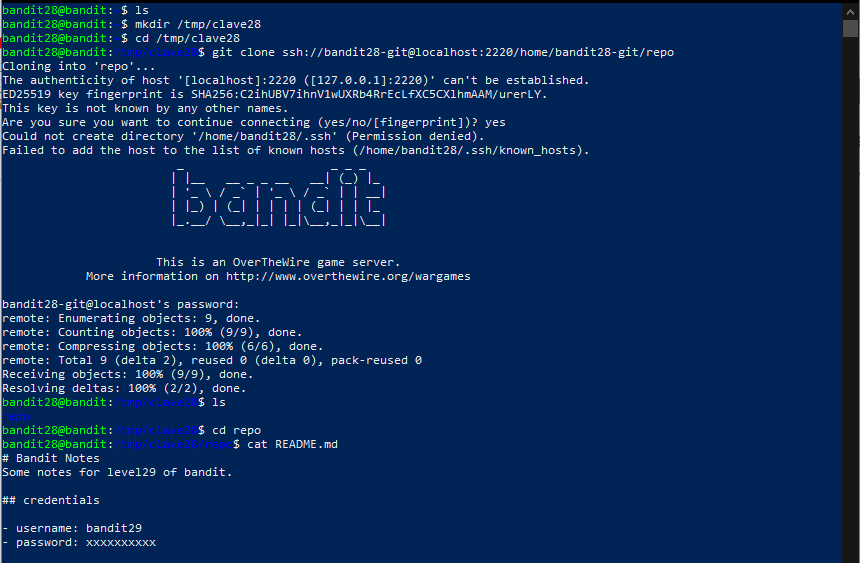
Nivel 27 = upsNCc7vzaRDx6oZC6GiR6ERwe1MowGB

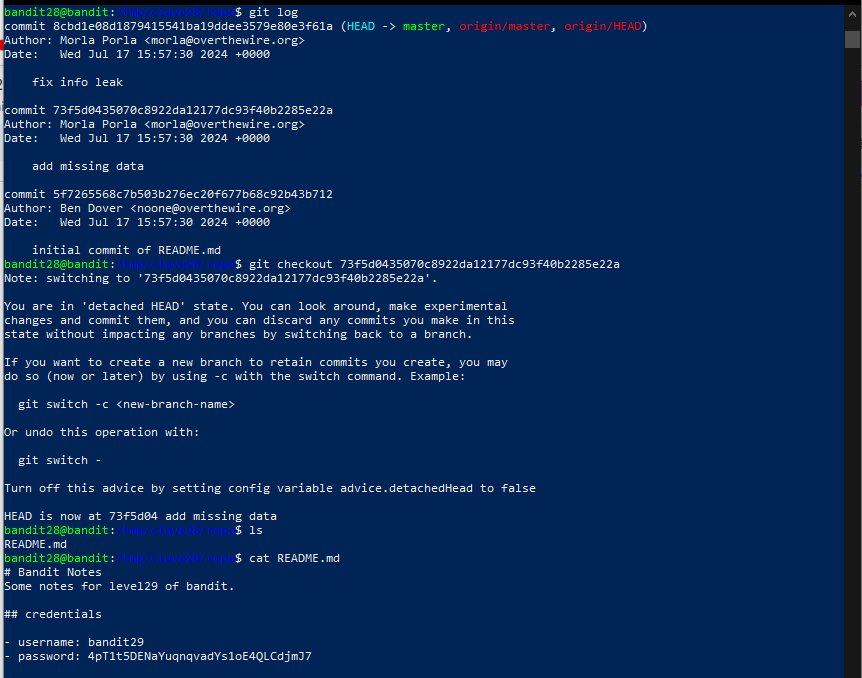
La idea de este nivel es poder clonar el repositorio desde una referencia establecida que nos dan en el enunciado del nivel, luego navegar sobre este y buscar el archivo readme, ya que por lo general en un repositorio la idea del readme es mostrar información adicional, para este caso es la contraseña del siguiente nivel.



Nivel 28 = Yz9IpL0sBcCeuG7m9uQFt8ZNpS4HZRcN

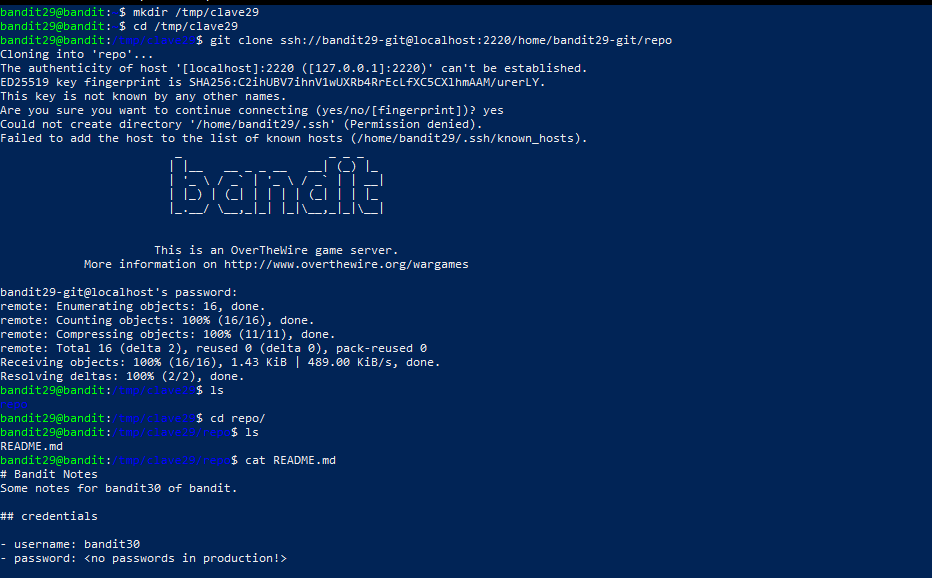
Para este nivel tenemos que encontrar la clave en un repositorio de git por lo que primero hacemos una carpeta en /tmp para clonar el repo y miramos el archivo que hay dentro, como ese archivo no nos proporciona la clave usamos git log para ver los commits anteriores a ese y nos cambiamos a la rama que hizo el ultimo commit, por ultimo abrimos el archivo readme y ahí encontramos las credenciales para el siguiente nivel.

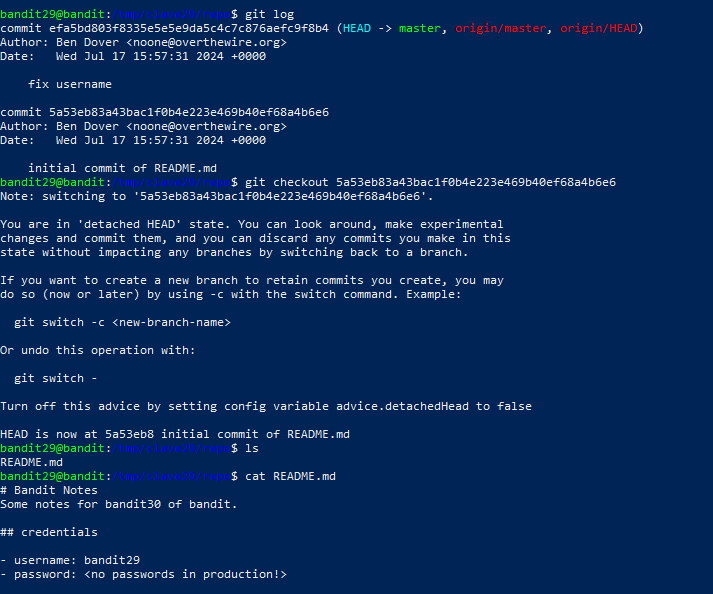


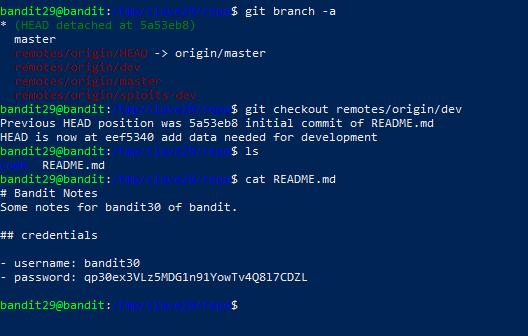


Nivel 29 = 4pT1t5DENaYuqnqvadYs1oE4QLCdjmJ7

Para este nivel hacemos los mismos pasos del punto anterior hasta que llegamos al readme.txt que nos dice “no password in production” después de eso verificamos las ramas y nos cambiamos a la rama dev donde podemos encontrar la clave para este nivel..



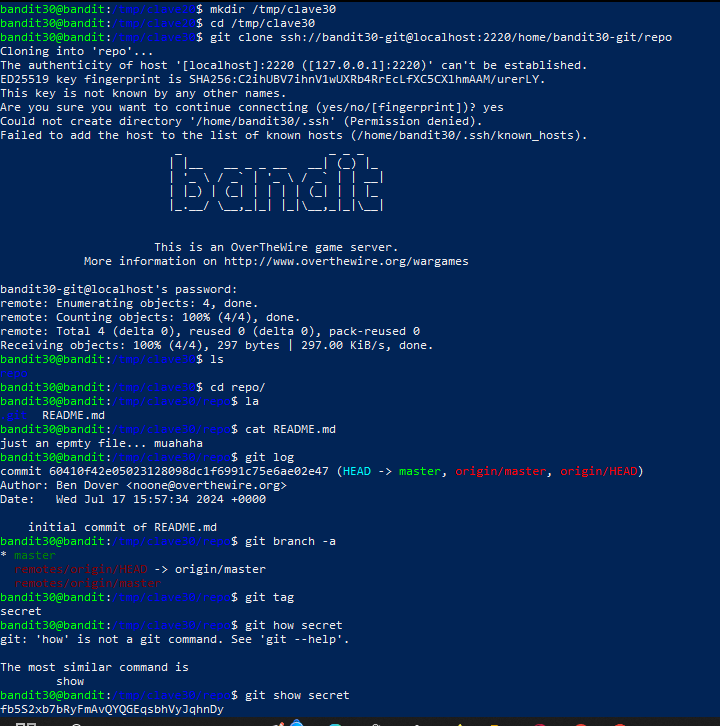




Nivel 30 = qp30ex3VLz5MDG1n91YowTv4Q8l7CDZL

Para este nivel y como en los anteriores clonamos el repo de git y intentamos buscar la clave en el archivo readme que nos ofrece, pero en este caso nos dice que está vacío lo cual es un poco sospechoso.

Entonces con el comando git tag listamos las etiquetas (si las hay) y encontramos que hay una llamada secret y con el comando git show mostramos el contenido asociado a esa etiqueta que es la clave para este nivel.



Nivel 31 = fb5S2xb7bRyFmAvQYQGEqsbhVyJqhnDy

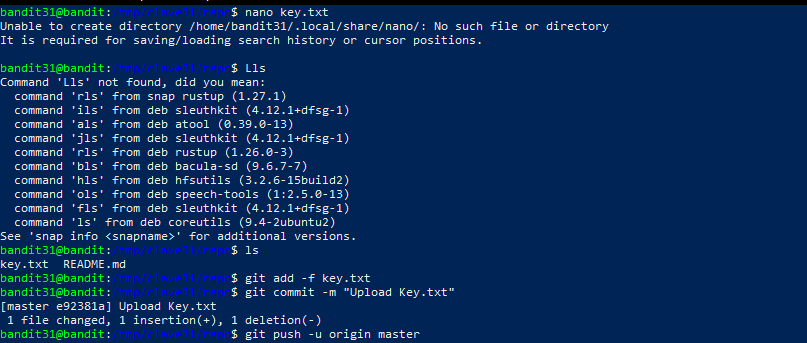
Para este nivel se nos pide modificar el archivo key.txt y ponerle un texto especifico “May I in you” y realizar los pasos respectivos para hacer un push, los cuales listare a continuación

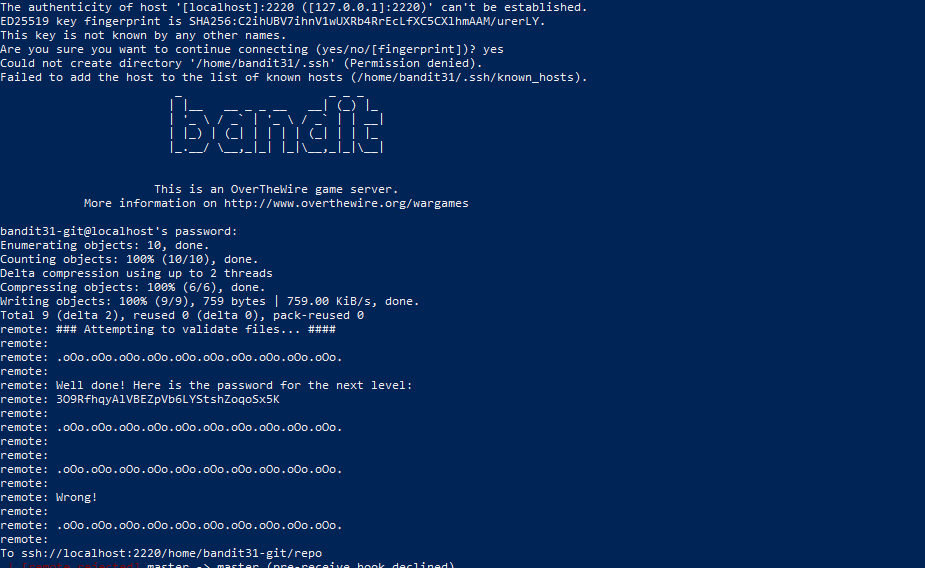
git add .

git commit -m “udapte”

git push -u origin

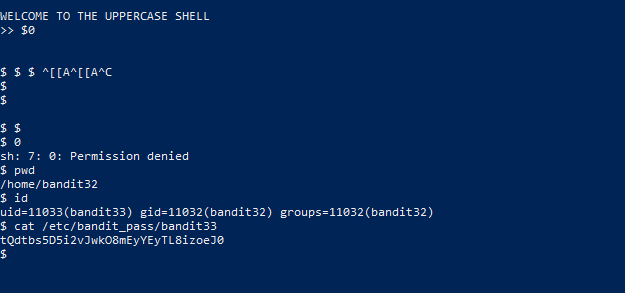
y de esta forma podemos obtener la clave para este nivel, cabe resaltar que para este nivel como en los anteriores se clonó el repositorio y se creo un directorio en /tmp.



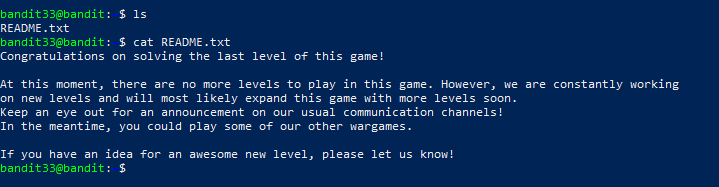


Nivel 32 = 3O9RfhqyAlVBEZpVb6LYStshZoqoSx5K

Para el último nivel solo tuvimos que buscar en /etc/bandit\_pass/bandit33 que es donde se encuentran las claves para los niveles de bandit.



Nivel 33 = tQdtbs5D5i2vJwkO8mEyYEyTL8izoeJ0

En este nivel solo leemos lo que está en el readme y así concluimos que acabamos bandit y sus niveles

Nivel 34 = no hay xd