Bienvenidos a esta nueva sesión donde vamos a trabajar para hacer un pequeño ejemplo en bas, a trabajar con hases y en este caso también palabras.

Bueno, lo que vamos a hacer es un pequeño script, vamos a generar un pequeño script en base el cual va a leer un fichero y va a ir hasheando todas las palabras de ese fichero en diferentes algoritmos.

Estos datos al final lo que vamos a hacer es almacenarlos en un fichero separado por comas, delimitado lo que queramos, puntos, coma, lo que será rollo CSV.

¿Nosotros vamos a utilizar cómo?

Pues porque sea algo similar a un CSV.

Bien, vamos a pasar a la máquina Kali, donde vamos a empezar a trabajar directamente sobre el ejemplo.

Vamos a abrir el editor, vamos a poner por ejemplo, vamos a empezar indicando cuál es la shell por defecto que queremos utilizar en nuestro script.

Lo primero que vamos a hacer es que nuestro script le demos pasar una serie de parámetros.

La idea que tenemos, vamos a poner aquí por ejemplo un comentario, nuestra idea va a ser que el uso de nuestro script sea, oye, pues pones el nombre del script y le pasas un parámetro, que el parámetro un va a ser realmente el fichero de palabras, y luego pasamos un parámetro dos, que va a ser realmente el fichero donde queremos almacenar los hashes.

Si metemos un parámetro o ejecutamos con tres, cuatro parámetros, el script no debe funcionar, por lo cual lo primero que vamos a hacer es aplicar aquí una condición donde digamos, si el número de argumentos que recibe el script es diferente a dos, pues yo lo que hago es indicarte no es correcto.

Entonces tú tienes que, en este caso hasworks sh, pues vamos a decirle que el parámetro un debe ser el fichero word, txt o el que sea, y el parámetro dos, vamos, realmente

estamos esperando los parámetros, tiene que ser el nombre del output, el fichero hash.
Txt y esto es un echo y aquí ejecutamos un exit para que el script degenere funcionar.
¿Si por ejemplo nosotros vamos a poner aquí a modo de prueba, vamos a ponerle, si yo no le paso parámetros, le paso por ejemplo un parámetro o le paso tres parámetros, veis?
Esto nos va a ir fallando, pero si metemos dos, dos parámetros, sería esto y está funcionando.
Lógicamente, claro, no son los parámetros esperados, nosotros esperamos que le metan parámetros reales, pero bueno, es una primera aproximación.
Vamos a continuar, vamos a el echo go, lo vamos a quitar y ahora que ya tenemos este control del skip, vamos a empezar.
Si llego a este punto significa que han introducido parámetros, damos por hecho, podemos hacer alguna comprobación más con impresiones regulares de si lo que me han pasado existe o no existe, utilizar operadores que existen en bas para saber si lo que me ha pasado como parámetro un es un fichero, realmente es un fichero de tal tipo, pero de momento eso a dejar para otros momentos.
¿Qué hacemos con este bucle?
Estamos iterando y estamos leyendo líneas.
¿La sintaxis aquí en base es un poquito, a veces un poco rara, porque al final dices están pasando a ese while, están pasando ese dólar input file que es ese es el primer parámetro que nos han pasado, que realmente lo que estamos esperando es que nos pasen el fichero de palabras 1.
Vez que nos han pasado esto, lo que vamos a hacer es, bueno, vamos a empezar a aplicar qué algoritmos queremos tratar?
Pues el MD.
Y fijaros, antes de pasar o de seguir continuando con esto, vamos a hacer un pequeño inciso.

Yo por ejemplo, yo por ejemplo si pongo esto, fijaros que el eco hola, si yo le paso, lo paso, él me va a sacar y el guión, esto es importante, fijaos que con el n y sin n el hash es diferente, porque con el n estamos haciendo el hash de la palabra hola, sin el n realmente estamos haciendo lo si le quitamos el n estamos haciendo hola y salto de línea.

Nosotros realmente queremos hacer el hash de hola, no de hola.

Entonces y luego claro, si yo obtengo dos valores, yo me quiero quedar con esto, no quiero quedarme con el resto.

¿Entonces yo lo que voy a hacer es utilizar comando cut, le puedo decir oye, quiero quedarme con el campo un, esto no estoy indicando el delimitador, cuál es el campo un? Fdfile campo un hay un parámetro dentro del put, podéis mirar en la ayuda del comando, que es d, y con ese parámetro nosotros podemos decirle el espacio, claro, fijaros, él corta este espacio de aquí, por lo cual ahí dice, bueno, el campo un es el de la izquierda, el campo dos será el siguiente.

Entonces bueno, pues ahí me quedo con el hash y eso sí lo que me interesa, me interesa porque ahora ya sí vamos a volver a describir, me quedo con el hash y eso sí lo que me interesa, me interesa porque ahora ya sí vamos a volver al script.

Ahora sí, por ejemplo, yo puedo decir, voy a crear una variable m, creo contexto aquí en bas para poder ejecutar instrucciones y le digo echo de dólar line, acordaos que lo que hace esto es leer las líneas del fichero que estamos pasando, se lo pasamos a m sum y luego hacemos la parte del cut.

De esta forma, claro, vamos a poner aquí por ejemplo m si tenemos esta forma, claro vamos a poner aquí por ejemplo m, claro vamos a poner aquí por ejemplo m si tenemos un fichero words, por ejemplo pájaro cat python, python por ejemplo, vamos poniendo diferentes animales, vamos poniendo y le pasamos animales, pues vamos poniendo y le pasamos ahora a nuestro script el segundo parámetro de momento nos da igual, fijaos como si nos está funcionando, esos son hashes m que nos está generando.

Entonces ahora qué es lo que vamos a hacer ahora en vez de pintarlos aquí y ahora vamos a irnos a saco sobre la línea, pero en vez de m vamos a coger el comando samos, la misma operativa ahora vamos a imprimir el sa primero, entonces lo que vamos a obtener es la palabra ver m y la palabra ver en sound, la palabra do m o la palabra dos con sauron y así vamos a ir.

Vamos a ir generando todo.

¿Qué ocurre?

Pues que lo que ocurre es que lo que nos interesa es almacenar un fichero, como hemos dicho antes, la salida, queremos almacenar todos estos hases separados por comillas o gomas mejor dicho, en un fichero, también queremos almacenar el sa y los que queramos añadir realmente.

Entonces no operativa, generamos el sa y cortamos y luego decimos bueno lo que queremos hacer es echo md coma sa, esto si lo generamos, fijaros como aquí hay una línea, lo que pasa que es muy larga y tal, pero fijaos las comas, ya lo tenemos, lo tenemos.

Entonces ahora lo que vamos a hacer es, yo cojo la salida y lo redirijo pues al dólar output file, si lo ejecutamos ya no va a salir por pantalla, hemos dicho que va a ser por ejemplo la palabra hases que no existe, se creará el fichero, fijaos que no sale nada.

Hacemos ahora el fichero hashes, vamos a abrirlo por ejemplo en editor de texto por líneas, aquí se ven muy bien las líneas, esta fue una línea y ahí tenéis el primer parámetro es el has en m, el sa y el sa.

¿Por qué es interesante esto así?

¿Pues es interesante porque separa por comillas como un CSV, luego a la hora de procesarlo, si nosotros tuviéramos que procesarlo después también sería bastante sencillo, porque pensad que yo puedo decir quiero leer por ejemplo el fichero que me pasa, que va a ser el dólar output file por ejemplo, y queremos ir line y hacemos un cut campo dos, por ejemplo lo quiero es a y digo que el delimitador es la coma, fijaros esto

debería funcionar, bueno voy a borrar un momento el fichero hashes para que lo volvamos a generar al final veis?

¿Me estoy quedando con el campo, pero es más, podemos ampliarlo a un argumento más, no?

¿Podemos decir pues en vez de en vez de ser dos argumentos lo que esperamos es me pueden pasar tres?

Podríamos hacerlo, me pasas tres argumentos o me pasas dos o me pasas tres, podríamos decir si no es igual a dos o no es igual a tres entonces te muestro ese mensaje, pero si es dos o tres luego tendría que ver si es dos voy por un camino, si es tres significa que me has pasado como tercer argumento

por ejemplo el tipo de hash que quieres sacar de un fichero.

Al final ya cada uno puede trabajarlo de 1000 maneras como mejor le venga.

Entonces como veis es un ejemplo sencillo donde se trabaja el uso simplemente de herramientas de hashing y también hemos enlazado con utilizar scripts de bash, es decir, ahora es útil para nosotros utilizar scripts de bas, siempre va a ser útil por tema automatización, cuando veáis posteriormente, por ejemplo en sistema linux es muy importante la post explotación con script devastación al final nos va a permitir hacer cosas bastante potentes de post explotación.

Pero aquí en este caso pues también empezamos a trabajar de concepto criptográfico de los hases.

Lo estoy viendo a través de un script de path que me permite, como digo, me permite poder generar un ejemplo muy básico, pero por ejemplo si nosotros quisiéramos hacer una pequeña herramienta para oye, qué hash, para encontrar un hash, tener un diccionario y ver qué hash corresponde con ciertas palabras, sería el proceso contrario.

Imaginaos que nos dan un fichero con hashes, por ejemplo hashes m, hases a y nos dicen oye tengo este diccionario de palabras, tienes que encontrar qué palabras se encuentran

en esos hashes.

Eso es crackear, eso es el cracking, el password cracking que también se utiliza en el hacking ético, por lo cual al final aquí se abre una ventana importante de posibilidades y de cosas que se pueden trabajar.

¿A Modo de resumen de todo esto, vamos a analizar de nuevo el script o a modo de resumen esta parte de aquí importante, porque esta es la parte de oye, yo como desarrollador, como creador del script, digo mi script va a funcionar bien si le pasas en este caso dos argumentos?

Puede ser que le pasen dos, puede ser que dice que le pasen cuatro, oye, dependiendo lo que necesites, o puede ser que dice que pueda funcionarte con dos o con cinco porque son casos diferentes que tú estás implementando en tu script, pues también lo puedes implementar con este control que al principio luego es verdad que cada argumento para ser más finos tendríamos que validarlo también, es decir, oye, si esperamos que el primer parámetro sea un fichero, existen mecanismos en Bas para poder identificar que el primer parámetro que va a pasar es un fichero 1 tipo en concreto para saber si existe o no existe, si de tal tipo, si tenemos permiso para hacer cosas sobre él, eso se puede meter, son cosas que se pueden evaluar a través del comando test con los operadores unarios se nos escapa un poco del objetivo de la sesión pero se puede hacer.

Y luego el parámetro dos exactamente lo mismo, habría que validarlo, lo que pasa es que es un parámetro de salida, en este caso es un output donde oye, yo voy a generar unos datos y los quiero almacenar en este segundo parámetro que me están pasando, con lo cual realmente no hace falta tanto validar esa existencia.

Luego el while está la parte central del script, pues el while lo que va a hacer es, oye, el fichero que me ha pasado como entrada, como palabras que quieres hashear, pues lo vamos leyendo línea a línea, aquí está la operada en la instrucción, se genera el m, se genera el sa, se genera el sa de la misma manera, como veis, se hace uso del comando

cat y luego se van almacenando línea, línea, esos hashes, esa palabra haseada.
También podríamos haber puesto aquí, quedaría bien también decir, oye, dólar line, por ejemplo, dos puntos tal, pues sí, también podríamos haberlo puesto.
Vamos a generarlo, vamos a generar esta parte de aquí que hemos hecho antes a modo de ejemplo, por ejemplo, ver dos puntos has m hasa un has a 512.
Entonces ahí vais viendo un poco el resultado y ahí se puede ir jugando bastante, se puede jugar hasta donde nuestra imaginación nos quiera llevar, nuestras necesidades dentro de un jagnético nos necesite.
Bien, pasamos otra vez, vamos a pasar de nuevo a las slides.
A modo de conclusión deciros nada, hemos hecho este script, ya lo hemos resumido, o sea que tampoco voy a entrar más otra vez a resumirlo, un script interesante que nos permite ya empezar a trabajar con algoritmos de hashing y haremos otro ejemplo más, trabajaremos con más conceptos criptográficos a través de Bash, que yo creo que también es lo más buena forma para tanto poder medir los conceptos criptográficos básicos y también practicar un poquito la programación en Bash.