Bueno como introducción quiero decirles que ojalá les hayan dado un buen resultado en el examen, pero si así no fuera no se preocupen. Simplemente vean en lo que se equivocaron y retomen conciencia o relean si es necesario.

Una cosa que quería decir del examen es que quizás les marcó error por no haber puesto tildes. Bueno la idea es que empiecen a darse cuenta que tienen que escribir correctamente porque aunque el hacker muchas veces es un insocial incurable, tiene que tener capacidades correctas y altas para poder comunicarse. Luego cuando veamos el apartado de **ingeniería social** verán por qué.

Hoy vamos a ver algo muy valioso en el mundo del hacking.

Supongamos que logramos robar una base de datos pero las contraseñas estan hasheadas. **¿Cómo hacemos para obtener esas contraseñas tan buscadas?**

Bueno tenemos algo conocido como la **fuerza bruta**, es decir que si viésemos un candado con una combinación númerica para poder abrirla, probaríamos **todos** los números hasta dar en el clavo.

“**Pero esto llevaría tanto tiempo que mejor me consigo una novia y me dejo de estar con esto del hacking”**

Tranquilo Manolo. Eres muy feo para conseguir novia así que mejor consigue paciencia :D. Es broma, es broma.

En realidad tienes mucha razón, este ataque (aunque obviamente llega a ser automatizado por un programa haciendo muchísimos intentos por segundo) **es muy lento**. Más que nada cuando no tenemos ninguna información exacta sobre lo que buscamos. Imaginen que intentamos dar con una contraseña de 4 a 12 caracteres de largo con números, minúsculas, mayúsculas, caracteres especiales, etc.

Ni hablar si en vez de querer pegarle a la contraseña, tenemos que darle al **hash**. Esto haría que a cada contraseña le tenemos que aplicar el **hash correcto** (que a veces no tenemos idea de cual hash fue aplicado) y luego **compararlo** con el hash obtenido para ver si es correcto. Por más que sea un milisegundo por hash aplicado, esto hecho tantas veces se convierte en un número importante.

Entonces a alguien se le ocurrió que en vez de simplemente intentar con todas las combinaciones posibles del universo, reduzcamos ese número a algo más sensato y posible. Entonces creamos una tabla que se llamará **diccionario** y con esta tabla se intentarán los aciertos hasta que alguno haya sido el correcto. Claro que tenemos **menos posibilidades** de que ese momento llegue -más aún comparándolo con el ataque de fuerza bruta, que tarde o temprano dará en el blanco-. Pero esto no quita que sea un ataque muy efectivo y claramente más corto que el anterior.

Obvio que muchas veces se realiza un **estudio previo**, de la persona, para ver cuales palabras podrían ser las mejores posibles contraseñas. Otras veces, uno tiene un diccionario prearmado de muchas palabras y mucho peso (gigas, o teras incluso) con las más posibles dentro del rango universal.

Pero claramente, alguien tuvo que mezclar estos dos procedimientos anteriores para **balancear el tema del tiempo y el acierto**. Entonces se creo, nuestra amiga, la **rainbow table** (o tabla arco iris).

En ésta, podemos partir de **una palabra** y darle **muchas combinaciones y variaciones a la misma**. Supongamos que tenemos una palabra en un diccionario, que es “admin”. Entonces, las variaciones serían “@dmin”, “adm1n”, “ADMIN”, “Adm1n”, etc. Entonces de una palabra obtenemos una gran

gama de colores con los cuales jugar. E incluso también puede juntar palabras del diccionario y mezclarlas sobre sus variaciones para obtener mejores resultados.

Pero siempre tenemos el factor tiempo en contra y por esto se utilizan hashes fuertes, ya que si algún hash es débil o si tenemos una contraseña común, ya vemos que las complicaciones no son muchas. Ahora, si un micro hace una cantidad enorme de intentos por segundo, imaginen lo que hace una **GPU** -que es la unidad de procesamiento de la placa de video- que es mucho más rápida. A esta técnica de usar la GPU se le llama -aunque no estoy seguro, debo admitir- **cuda password cracking.**

“**Entiendo. Pero, del lado defensivo ¿Qué podemos hacer para proteger las contraseñas?”**

Bueno a parte de darle un buen hash y aplicar las políticas correctas, podemos utilizar algo llamado **salt**. El salt es un **fragmento aleatorio** de caracteres y números que se le agrega al hash. Esto hace que cada bit agregado por este salt, **duplique el tiempo** que un ataque de fuerza bruta -por cualquier a de las técnicas antes mencionadas- va a llevar hasta pegarle a la conraseña que tanto buscan. Pero obviamente no vamos a conformarnos con un bit, sino que vamos a utilizar 32 bits y ver que esto llevaría muchísimo más tiempo. Podemos calcular fácilmente que cada contraseña tendrá sumadas, $\mathbf{2^{32}}$ **posibilidades más**. Obviamente que el atacante no lo tiene fácil, ya que además de tener que conocer el hash aplicado, va a tener que darse cuenta de que el salt está allí también.

Ya veremos a fondo los ataques en el laboratorio y asi poder intentar crackear las contraseñas que queramos ;)

Cualquier cosa pueden mandarme mail a: r0add@hotmail.com

Para donaciones, pueden hacerlo en bitcoin en la dirección siguiente:

1HqpPJbbWJ9H2hAZTmpXnVuoLKkP7RFSvw

Roadd.
