The Role of the Explosion Force Attacks Combinational in gross

Papel de la explosión combinacional en ataques de fuerza bruta

V López.

Keywords:
Attack combinations, Combinational explosion, brute force.

Abstract
This paper presents a work that describes how through the technique known as brute force cracking attempts to identify unknown brand by trying all possible combinations of an alphabet, its effectiveness is that the alphabet taken as input to the algorithm contains all the letters , numbers and / or symbols of the string being searched. Otherwise it is important to note that as the length of the growing chain grows exponentially the number of combinations to be tested so that generates a combinatorial explosion.

Palabras clave:
Ataque, combinaciones, Explosión combinacional, fuerza bruta.

A través de la técnica de craking denominada fuerza bruta se intenta identificar una cadena desconocida probando con todas combinaciones posibles de un abecedario, su eficacia radica en que el abecedario tomado como entrada al algoritmo contenga todas las letras, números y/o símbolos que conforman la cadena que se busca. Por lo demás es importante hacer notar que a medida que crece la longitud de la cadena, crece exponencialmente el número de combinaciones a probar de tal manera que genera una explosión combinacional.

I. INTRODUCCION

Hoy por hoy la protección a la confidencialidad es día a día una tarea más difícil de mantener, es por eso que las tecnologías que protegen el acceso a los recursos son cada vez más complejas y diversas. En el campo de la autenticación generalmente alguno de estos tres factores o una combinación de estos, algo que tú conoces, algo que tú eres y algo que tú tienes, refiriéndose en su orden por ejemplo a un password, un token o una huella, nos centraremos en el primero de los ítems que hoy por hoy es el más común, por lo que lo vemos muy seguido en sistemas que requieren de una autenticación a través de unas credenciales por parte del usuario como usuario y contraseña para poder acceder a un recurso; esta medida actualmente se considera un control básico y elemental de acceso, a veces el único, que de no estar bien estructurado se convierte en una vulnerabilidad al momento en que se confía ciegamente en la dificultad para descubrir la cadena o cadenas que hacen parte de esas credenciales.

II. FUERZA BRUTA

Actualmente el cracking de password es una actividad en alto crecimiento y que a través de herramientas cada vez más automatizadas le facilitan la tarea al craker para la obtención de información confidencial. Una de las técnicas más famosaspara realizar esta tarea se llama "Ataque por Fuerza bruta" y se basa en la combinación de las letras de un abecedario, números y hasta símbolos especiales variando la longitud hasta dar con la cadena desconocida, esta técnica que básicamente estaría incluida dentro de los métodos de "Prueba y Error", es un ataque sencillo en su concepción y certero si se posee todas los valores de entrada necesarios para descubrir las cadenas que conforman las credenciales.

Las etapas que hacen parte de este ataque se podrían resumir en: Identificación del objetivo, identificación de los valores de entrada, generación de datos aleatorios, monitoreo de las excepciones y por ultimo determinar la explotabilidad del objetivo [1]; a continuación se muestra una implementación en Java de un algoritmo que busca por fuerza bruta una cadena desconocida. [2]

```
public class BruteForceStringSearcher implements StringSearcher {
private final CharSequence _pattern;

publicBruteForceStringSearcher(CharSequence pattern) {
assert pattern != null : "pattern can't be null";
assertpattern.length() > 0 : "pattern can't be empty";
_pattern = pattern;
}
publicStringMatch search(CharSequence text, int from) {
assert text != null : "text can't be null"; assert from >= 0 :
"from can't be < 0"; int s = from;

while (s <= text.length() - _pattern.length()) {
inti = 0;
while (i< _pattern.length() && _pattern.charAt(i) ==
text.charAt(s + i)) {
++i;
}
if (i == _pattern.length()) {
return new StringMatch(_pattern, text, s);
}
++s;
}
return null;
} }
```

III. EXPLOSION COMBINACIONAL

La implementación al parecer sencilla del ataque de fuerza bruta en realidad posee un gran problema matemático, ya que cada vez que se incrementa el abecedario y/o la longitud de la cadena, lo que sería respectivamente el número de valores de entrada "n" y/o la longitud de la cadena "l", la cantidad de combinaciones que puede llegar a surgir crece estrepitosamente; a tal punto que en algunos casos tardaría años en encontrarla, a este fenómeno se le denomina *Explosión Combinacional*.

En la Tabla 1 se muestra el caso en que una sola credencial este formada por letras del abecedario español formado por 27 caracteres sin contar la "ch" y la "ll" y solo en minúsculas; el número de permutaciones posibles en cada caso es n^l , en la tabla se muestra la cantidad de posibilidades, variando únicamente la longitud de la cadena

Tamaño del abecedario	Longitud de la clave	Número de Permutaciones Posibles	Tiempo Estimado (s)	Crecimiento
27	1	27	<1	
27	2	27^2=279	<1	2600%
27	3	27^3=19683	<1	2600%
27	4	27^4=531441	<1	2600%
…	…	…		
27	8	27^8=282.429.536.481	22	2600%
27	9	27^9=7.625.597.484.987	589	2600%
27	10	27^{10}=205.891.132.094.649	15899	2600%

Tabla 1. Número de permutaciones con respecto a la longitud de la clave

Analizando la tabla anterior nos podemos dar cuenta que cada vez que se incrementa la longitud de la cadena l a $l+1$, el número de posibles permutaciones crece, por lo que se maximiza el tiempo requerido para crakearla En este caso se muestra además el tiempo estimado para recorrer todas las posibles combinaciones en cada caso teniendo como base un procesador AMD Athlon 64 Dual Core el cual realiza 18,500MIPS(Millones de instrucciones por segundo), lógicamenteno todo el procesador estaría disponible para la realización de esta tarea por lo que seleccionamos el 70% (12.950 MIPS) dejando el 30% restantes para procesos base del sistema operativo.

El tiempo promedio para realizar cada operación sería el resultado de dividir el número total de combinaciones por cada caso entre las operaciones por segundo que es capaz de realizar el procesador tomado como ejemplo y este sería el tiempo promedio que se gastaría la maquina antes descrita en encontrar la cadena a través de una ataque de fuerza bruta, sin embargo este tiempo puede variar dependiendo del sistema operativo utilizado, memoria RAM disponible e implementación algorítmica del ataque.

Además del análisis de la Tabla 1 se nota que a medida que se incrementa l en una unidad el porcentaje de crecimiento es del 2600% de manera continua por lo que notamos un crecimiento lineal entre el número de combinaciones. Ahora veamos el caso en que se incrementa en una unidad la cantidad de elementos n y se deja constante la longitud l como es el caso de la Tabla 2.

Tamaño del abecedario	Longitud de la clave	Número de Permutaciones Posibles	Tiempo Estimado (s)	Crecimiento
10	4	10.000	<1	52.42%
11	4	14.641	<1	46%
12	4	20.736	<1	42%
…	…	…		…
25	4	390.625	<1	18%
26	4	456.976	<1	17%
27	4	531.441	1	16%

Tabla 2. Número de permutaciones con respecto al tamaño del abecedario

En la Tabla 2 se nota que a medida que aumentamos el tamaño del abecedario n a $n+1$ existe un crecimiento en el número de combinaciones, pero este crecimiento se va haciendo cada vez menor a medida que esto pasa. Ahora en la Tabla 3 se observan los datos con el crecimiento parejo de la variable n y l.

Tamaño del abecedario	Longitud de la clave	Número de Permutaciones Posibles	Tiempo Estimado (s)	Crecimiento
10	1	10	<1	52.42%
11	2	121	<1	46%
12	3	1.728	<1	42%
…	…	…		…
25	16	23.283.064.365.387.000.000.000	<1	18%
26	17	1.133.827.315.385.150.000.000.000	<1	17%

Tabla 3. Número de permutaciones con respecto al aumento de la longitud de la clave y del abecedario

Esta que es definitivamente la mejor opción nos brinda un crecimiento netamente exponencial, lo que para el cracker se traduce en términos computacionales en un problema que se ve expresado en la relación exponencial que existe entre el número de combinaciones y el costo para determinar cada una de estas combinaciones; hasta llegar a punto donde es difícil para la computación actual llegar al objetivo en un tiempo prudente, es tan así que en algunos casos el número de cadenas posibles por probar es tan grande que hasta un equipo de cómputo con grandes características de hardware tardaría años en descifrar la cadena correcta.

Aquí es donde la explosión computacional nos pone un pare y nos señala lo difícil de la tarea a menos que se optimice la búsqueda de diversas maneras. Cada vez escuchamos más la importancia de endurecer las contraseñas y siempre dentro de esas recomendaciones se encuentran unas muy importantes como lo son que las contraseñas deben ser largas, de recomendable mínimo ocho caracteres, de los cuales se deben combinar mayúsculas, minúsculas, números y caracteres especiales entre otros. Tangible queda el hecho de que si siguiéramos las recomendaciones para elaborar una contraseña segura o al menos medianamente segura estaríamos asegurando cada vez más una mayor

dificultad para que el atacante obtuviera exitosamente nuestra contraseña

El caso de las actuales contraseñas para el sistema bancario es algo no menos preocupante el que tan solo permitan longitud de cuatro caracteres y que estos estén limitados a 10 números, es un caso preocupante al notar que en tan solo 10*10*10*10 da 10000 combinaciones podrían dar con la atesorada contraseña. Lo que llevaría a pensar que se debe tener más seguridad en las cuentas de Facebook, Hotmail, hayo u otros sitios similares que una cuenta bancaria, al menos en cuanto a número de combinaciones se refiere.

IV. ALTERNATIVAS

A. Programación dinámica.

La técnica de programación dinámica hace referencia a una técnica que en realidad no posee un problema tipo sino que se puede aplicar a un amplio grupo de problemas ya que se basa en el principio de optimalidad de Bellman "Una política optima está constituida de subpolíticas óptimas"[3]. Esta técnica evita explorar todas las secuencias posibles por medio de la resolución de subproblemas de tamaño creciente y almacenamiento en una tabla de las soluciones óptimas de esos subproblemas para facilitar la solución de los problemas más grandes, por lo que la recursividad juega un papel muy importante dentro de esta técnica. Los elementos básicos que debe tener un problema de programación dinámica son:

1. El problema se puede dividir en etapas o subproblemas(En el caso del cracking podríamos decir que cada carácter de la cadena es una etapa)
2. Cada etapa tiene un estado(encontrado o no)
3. La política de cada estado es transformarse para servir de punto de partida de la siguiente etapa(armando paso a paso la cadena valida)

B. Backtracking

Técnica también denominada "vuelta atrás" que selecciona inteligentemente las posibles soluciones a medida que recorre un árbol de todas o las más probables soluciones, a medida que realiza el recorrido va guardando cada paso en el conjunto de soluciones parciales [4]; este es la cadena buscada no se encuentra en una Subárea del árbol la solución se devuelve al proceso original y se ensaya por otra ruta diferente a las que se haya probado.

```
procBactrackingEnum (↕X[1 . . . i ]: TSolución, ↑num: N)
variables L: ListaComponentes inicio
siEsSolución (X) entonces num=num+1
EscribeSolución (X)
en otro caso
L  Candidatos (X)
mientras ¬Vacía (L) hacer
X[i + 1]  Cabeza (L); L  Resto (L) BacktrackingEnum (X, num)
finmientras
finsi
fin
```

C. Ataque de diccionario.

Este ataque es muy similar al ataque por fuerza bruta, la diferencia radica en que este prueba con una serie de palabras probables en lugar de generar combinaciones [5], generalmente las más utilizadas o las que tengan relación con la víctima. Si la cadena utilizada está en el diccionario se consigue reducir sustancialmente el tiempo necesario para encontrarla; además la utilización de palabras existentes y comunes como "secreto", "admin" o "1234" facilita este tipo de ataques. En la Figura 1 se muestra el top 10 de las contraseñas más utilizadas por todos los tiempos en el idioma inglés, según la firma de seguridad ZoneAlarm de Checkpoint.

Fig. 1 Top 20 de los passwords más utilizados de todos los tiempos en el idioma inglés.

D. Criptoanálisis a través de tablas arco iris.

Actualmente la mayoría de las contraseñas no se almacenan en texto claro, en cambio se pasan como entrada a un algoritmo de hash como MD5, aunque la mayoría de estos algoritmos han sido cripto analizados con éxito, con esto se logra almacenar la huella o resumen de la cadena que hace parte de la credencial de

un usuario. Pero surgió otro problema, ante una misma contraseña se generaba un mismo hash, es decir H(p)=h, por lo cual se podían generar ficheros con una relación contraseña-hash. Estos ficheros son el nacimiento de las tablas arco iris o por su nombre en inglés RainbowTables.

Una tabla arco iris utiliza una enorme lista de valores de hashes precalculados de casi todas las posibles combinaciones de contraseñas con casi todos los posibles caracteres especiales, letras y símbolos. Las tablas de arco iris son más rápidas que los ataques de fuerza bruta y diccionario además que requiere menos memoria. El tamaño del archivo de la tabla del Rainbow depende de si desea cargar los valores hash de las letras solamente, letras y números, o todos los caracteres pero generalmente son de varios gigas. Ante la aparición de las Rainbowtables se creó el concepto de semilla o pizca de sal (Salt), que no es más que un valor de un tamaño fijo que se introduce en el proceso de cifrado de tal forma que ante una misma contraseña existen tantos hashes como posibles valores pueda valer la semilla. Por ejemplo, si yo tengo una semilla de 2 bits puedo generar cuatro hashes distintos ante una misma contraseña:

H(p+00)=h1
H(p+01)=h2
H(p+10)=h3
H(p+11)=h4

De las técnicas anteriormente comentadas esta es la más utilizada en la actualidad para realizar las tareas de cracking de passwords, por esto se recomienda asegurar que el algoritmo de hash utilice la semilla o salt antes mencionada, para aleatorizar el resultado del hash.

V. CONCLUSION

A lo largo del tiempo se han optimizado los métodos de cracking de passwords haciéndole la tarea cada vez más sencilla al cracker, y al mismo tiempo la tarea más complicada al usuario, por lo que cada vez más, hay que preocuparse más por construir una buena cadena al momento de crear o actualizar sus credenciales.

La técnica de fuerza bruta es una de las formas más antiguas para desarrollar un ataque que permita el descubrir una cadena desconocida, este básicamente se basa en el método de prueba y error por lo que posee el inconveniente de la eficiencia y más aún cuando la longitud y/o el tamaño del abecedario es grande, lo que produce una generación exagerada de combinaciones aumentando el universo de posibilidades, esto último denominado explosión combinacional; por lo tanto el costo computacional para descubrir la combinación correcta por lo que este método es muy alto cuando la cadena está bien construida.

La generación de cadenas seguras dependen no solo de un factor sino de una combinación de estos como lo son la utilización de un abecedario grande que incluya caracteres alfanuméricos y símbolos, a su vez debe tener una longitud recomendada de no mínimo de 8 caracteres, se deben actualizar periódicamente aproximadamente una vez al mes y no se deben utilizar en más de un sistema de información. Es importante enfrentar el hecho que es mejor incrementar la longitud de la cadena que incrementar el abecedario esto en el caso de que el sistema sea excluyente, pero mejor aún es incrementar las dos variables ya que se denoto que está asociado a un crecimiento exponencial del número total de posibilidades lo que incrementa la dificultad para el cracker que le haya seleccionado como objetivo.

Por otro lado, se mostraron técnicas alternativas a la fuerza bruta como la programación dinámica, el backtracking, la Heurística, el ataque por diccionarios, el criptoanálisis con tablas arco iris, este último el más utilizado en la actualidad por lo que descubre una cadena almacenada en texto no claro.

Todas las técnicas anteriormente mencionadas minimizan unas más que otras, el número de posibilidades a explorar en comparación con el ataque de fuerza bruta, lo que lógicamente minimiza el tiempo para descubrir la cadena correcta; es por lo anterior que hoy más que nunca debemos tener claro que las matemáticas nos ayudan a entender que tan fácil o difícil puede llegar a ser el proceso de cracking, pero el que a final decide si toma acciones es el usuario.

Pero claro está el hecho de que así usted haya logrado la formación de una credencial fuerte y aceptación de políticas de actualización constante, esto no lo protege del todo y menos de un ataque de ingeniería social, por lo que recordemos que la autenticación depende de un sin número de factores, por lo que es importante se conozcan y estén bien administrados**.**

REFERENCIAS

[1] M. Sutton, A. Greene, P. Amini,*Fuzzing: Brute Force Vulnerability Discovery,* 1Ed, Adisson Wesley, 2007.

[2] S. Harris, J. Ross, *Beginning Algorithms*, 1 Ed, Wrox, 2005.

[3] M. Begoña, *Programación matemática para la economía y la empresa*, Universidad de Valencia, 2009.

[4] M. Begoña, Programación matemática para la economía y
la empresa, Universidad de Valencia, 2009.

[5] W. Stallings, *Fundamentos de seguridad en redes*, 2 Ed, Prentice Hall, 2004.