Busqueda Binaria Aplicada a las Blind SQL Injection

Búsqueda Binaria Aplicada a las Blind SQL Injection
Antes de empezar me gustaría aclarar algunas cosas:

- No Pretenderé Ocupar conceptos enredados, ni tampoco caer en tecnicismo, lo explicare como yo lo entiendo, y como me fuese gustado que me lo explicaran.
- Esta Publicación esta enfocada a las personas que tengan nociones básicas de Blind SQL Injection, quienes no saben de qué estoy hablando, es mejor que busquen algún tutorial de este tipo de vulnerabilidad en Nuestra Comunidad. o lean los siguientes conceptos.

Conceptos :

1. Seccion SQL Injection en Undersecurity
2. ¿Que es el Formato ASCII?

Comandos en MYSQL :

1. MID
2. ASCII

Control De Flujos:

1. CASE
2. IF

Información Blind SQL Injection

1. Blind SQL Injection by Netting

Información Búsqueda Binaria

1. Busqueda Binaria
2. Arbol Binario

Ahora bien empecemos…

Las Blind SQL Injection, es un tipo de vulnerabilidad que es bastante común y de una explotación bastante sencilla, pero **tiene 1 solo problema**. **Es necesario generar una cantidad de peticiones bastante considerables hasta encontrar un valor verdadero. Por lo que se convierte en una explotación bastante lenta y tediosa.**
Por Ejemplo, Planteemos el Siguiente Escenario.

Tabla User

```
mysql> desc user;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| User     | varchar(20) | NO   |     | NULL    |                |
| Password | varchar(20) | NO   |     | NULL    |                |
| id       | int(20)     | NO   | PRI | NULL    | auto_increment |
+----------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Datos Contenidos en la tabla User:

- **Usuario:** Administrador
- **Password**: Undersecurity

```
mysql> select id,password,user from user;
+----+---------------+---------------+
| id | password      | user          |
+----+---------------+---------------+
|  1 | Undersecurity | Administrador |
+----+---------------+---------------+
1 row in set (0.00 sec)
```

Ahora bien, Para Empezar Nosotros Obtendremos el Primer Valor del Usuario, mediante la metodología Común para explotar las Blind SQL Injection. (En un Ambiente Controlado dentro de la consola, no es una situación real de explotación).
Para encontrar un valor verdadero tendremos que testear desde el Carácter 32 de la Tabla Ascii Hasta el Valor 122.
Como Podemos Observar en la Siguiente Imagen, el Primer valor del Usuario es “A”, transformado al formato Ascii obtendría el valor de 65.

```
mysql> select (select user from user);
+-------------------------+
| (select user from user) |
+-------------------------+
| Administrador           |
+-------------------------+
1 row in set (0.00 sec)

mysql> select mid((select user from user),1,1);
+----------------------------------+
| mid((select user from user),1,1) |
+----------------------------------+
| A                                |
+----------------------------------+
1 row in set (0.00 sec)

mysql> select ascii(mid((select user from user),1,1));
+-----------------------------------------+
| ascii(mid((select user from user),1,1)) |
+-----------------------------------------+
|                                      65 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

En la Practica Tendríamos que Generar **33 Peticiones (65 – 32)** para Recién Obtener el Primer valor del usuario.
Y si nos ponemos bien extremistas, si el usuario se llamara *zamorano*, tendríamos que generar **90 peticiones** para recién Obtener el Valor. Por lo que se hace bastante lento y tedioso este tipo de explotación.

Ahora viene la implementación de Búsqueda Binaria.
La Búsqueda Binaria se basa en la teoría de divide y vencerás. La Idea es ir dividiendo el rango en mitades.
Por Ejemplo : La clave que queremos encontrar es **9.**
Tenemos el Siguiente String
arreglo =(1,2,3,4,5,6,7,8,**9**,10,11);
Dividimos en 2 el String Obteniendo.

1. Restricción = (1, 2, 3, 4, 5,6) false
2. **Restricción = (6, 7, 8, 9, 10,11) true**

Si se fijan la cantidad de números dentro del arreglo, es impar, por lo que se necesita repetir un numero dentro de los 2 String resultantes (6), para obtener 2 String Con la misma cantidad de caracteres.
Y Ahora ¿Cuál seria Nuestro String a Seguir?, El que Tenga Nuestro Valor a Buscar, en este caso el segundo String o la segunda restricción.

1. Restricción = (6, 7,8) false
2. **Restricción = (9, 10,11) true**

1. **Restricción = (9,10) true**
2. Restricción = (10,11) false

1. **Restricción = (9) true**
2. Restricción = (10) false

Finalmente obtenemos nuestro valor, dentro de la primera restricción. Con una cantidad de 4 peticiones.

Esta Misma Metodología Ahora la Aplicaremos a una Blind SQL Injection.

Ahora Aplicaremos los Valores de la tabla Ascii desde el valor 32 hasta el 122.

- **Arreglo :** 32-122 [Space-z]
- **Restriccion_1:** 32-72 [Space-M]
- **Restricción_2:** 72-122 [M-z]

Utilizáramos el Siguiente Consulta SQL.

Query Global:

```
+and+ (SELECT+IF ((ARREGLO), (RESTRICCION_1), false)) –
```

Explicación:

Si El valor a encontrar esta entre los valores del arreglo, entonces consultar si están dentro de la primera restricción. por el contrario si el valor a buscar no esta dentro de la primera restricción, entonces esto significa, que esta dentro de la segunda restricción.

Query : Restricción_1

```
select+case+(ascii(mid((CONSULTA),SUBSTRING,1))+RESTRICCION_1)+when+true+then+
(true)+else+false+end
```

Explicacion :

En el caso que el primer valor de la consulta este dentro del rango de la restriccion 1, entonces retornara verdadero. por el contrario retornara falso.

Ahora bien, un Simple Ejemplo :

Consulta : `select+user()`

Valor de user () : Administrador@localhost.com

Valor a Obtener : A

Valor ASCII :65

- **Parametros:**

Arreglo: 32-122 [Space-z]
Restriccion_1: 32-72 [Space-M]
Restriccion_2: 72-122 [M-z]

Consulta Completa :

```
http://www.host.com/vulnz.php?id=1+and+(SELECT+IF((select+case+
(ascii(mid((select+user()),1,1))+BETWEEN+32+AND+122)+when+true+then+(true)
+else+false+end),(select+case+(ascii(mid((select+user()),1,1))
+BETWEEN+32+AND+77)+when+true+then+(true)+else+false+end),false))--
```

Explicacion :

Buscaremos el Primer valor de la consulta “select user()”, el cual es A, (Ascii: 65), primero consultaremos si el valor esta entre 32-122 [Space-z], si este resulta verdadero entonces retonara TRUE , o por el contrario si no esta dentro del rango 32-122 retornara **FALSE.**

Si el valor retornado por la primera consulta es **TRUE** , entonces consultara si el primer valor esta entre 32-77 [Space-M], si esto es verdadero retornara **TRUE**, por el contrario retonara **FALSE** y el valor estaria entre 72-122[M-z].

En Esta Consulta Reducimos Nuestras alternativas de 90 a 45.

¿Podrías Adivinar que Estado Retornara la Consulta , Si el valor a Buscar es 64?

- Si Pensaste en **TRUE**, estas en lo Correcto.

Luego En la Segunda vuelta , el rango que se encuentre dentro del valor a buscar (en este caso 32-77), se convertirá en nuestro arreglo, y se dividirá en 2 **([32-54]-[55-77]),** así sucesivamente, hasta encontrar el valor final, **en solo 7 consultas.**

Para Demostrar Esta Metodología Desarrolle un Pequeño Programa que solamente muestra 1 carácter, en 7 peticiones.

Aclaro : Es un PoC en Php.

Posteriormente dentro del Lab de Undersecurity.net Se desarrollara la Tool Completa.

Imagen :

```
undersec@Undersec:~/Escritorio/BLINDOOL$ php blind00l.php -web 'http://localhost/1/www/blind.php?x=9' -q 'select+user+from+user
+where+id=11' -sub 1
WEB :> http://localhost/1/www/blind.php?x=9
QUERY :>select+user+from+user+where+id=11
MID :> 1
[*]VALORES DE COMPARACION ORIGINALES : 22-22
        [-]ENTRE : =-z
        [-]ENTRE : =-[
        [-]ENTRE : =-L
        [-]ENTRE : =-D
        [-]ENTRE : A-D
        [-]ENTRE : A-B
        [-]ENTRE : A-A

        CARACTER ENCONTRADO :  A

SEGUNDOS TOTALES  :  0
```

Código del POC Llamado **BlindD00l:** http://lib.undersecurity.net/index.php?download=./Lab/BLIND00L_SOURCE.txt

- Testeado En Mysql 5.x
- Curl Activado

Gracias a toda la Comunidad Undersecurity.net

- Cic4tr1z : logística Infinita
- Nork : Correcciones y Apoyo en ideas extravagantes.
- 1995: Apoyo Incondicional.
- N0b0dy :Notable y leal animo a la comunidad.
- Lix : Por ser Simplemente Lix.
- y A muchos mas que quedaron en el camino.

#OzX [Undersecurity.net]