Prepared Statement と Undefined Attack

ＮＴＴ コミュニケーションズ株式会社
ＩＴ マネジメントサービス事業部
セキュリティオペレーションセンタ

2010 年 03 月 09 日

Ver. 1.0

1. 調査概要

Prepared Statement(準備済み SQL 文)、プレースフォルダ、バインドメカニズム、パラメータ・クエリなど、SQL クエリをある程度事前にコンパイルしておき、DB 応答を向上させる機能を積極的に採用することが、SQL Injection 対策としても有効であるのは周知の事実である。

本文書では、プレースフォルダを未定義のままにしておくことで、思わぬ情報漏えいが起きうる可能性について検証した結果をここに記す。
ただし、今回検証したADO(ASP)、ADO.NET(ASP.NET)、Java、PDO(PHP)、DBI(perl)のうちで未定義の状態で脆弱性に発展しうるSQLクエリを実行したのは、古いバージョンの PDO(PHP)(「3.1 PDO(PHP)の場合 1」を参照)だけであった(Win32 版では発現しなかった)。
今一度、読者諸氏が開発したシステムで、プレースフォルダが未定義の状態で SQL が実行されることがないかどうか、またその場合どのような挙動を示すのかについて確認されることを推奨する。

2. Undefined Attack

2.1. Prepared Statement(準備済みSQL文)とは

一般的に Prepared Statement(準備済み SQL 文)と呼ばれる機能は、事前に大まかな SQL 文を用意し、かつ事前にある程度までコンパイルしておき、異なる部分だけを実行時にリアルタイムでコンパイルすることで、データベースの応答速度を改善するための機構である。

一般的に、Prepared Statement という機構では、SQL Injection 対策としての

- 汚染データを、文字列リテラルとして使う場合の SQL エスケープ
- 汚染データを、数値などのリテラルとして使う場合のデータ型の整合性チェック

を、実装しているため、この機構を積極的に採用することで、アプリケーション・プログラマが SQL Injection 対策を注意しながら開発するコストを削減することができるため、主な SQL Injection 対策として推奨されている。

3. Undefined Attack

一般的に Prepared Statement 内の「?(プレースフォルダ)」は、なにか値がセットされた上で、SQL クエリとして実行されうるものである。
しかしながら、Web アプリケーションのプログラミング上、または設計上の不備によって、値がセットされないまま実行される場合も考えられる。
このような場合、どのような挙動を示すのか検証してみた。
ほとんどの場合は、エラーや例外、または無反応となり、セキュリティ侵害となりうるような状況を確認することはできなかった。
しかしながら、PDO(PHP)の古いバージョン(Win32 版は除く)に関しては、値がセットされていない検索条件を無視した検索結果が出力された。このことは、Session Adoption などによって、本

来は値がセットしてあるべき変数に値がセットされていないような状況を作ることができる場合、検索条件がなくなり、データベース上の情報が大量に漏洩するような脆弱性となりうることを示している。

どちらにしても、セキュアなアプリケーションを開発するために、プレースフォルダやパラメータ・クエリのパラメータには何か値をセットするよう、プログラムを注意深く作成することが、アプリケーション・プログラマに求められている。

3.1. PDO(PHP)の場合 1

検証環境

- CentOS 5.4
- Apache ver2.2.3
- PHP ver5.1.6
- mySQL ver5.0.77

```
<html><head><title>Number</title></head><body><form method="get" action="">
<?php
$dsn = "mysql:dbname=toshi_db;host=localhost";
$conn = new PDO($dsn, 'toshi', 'acB6FTg98uTh');
$sql = "select * from str";
$str = "";
$flag_1 = "";
$flag_2 = "";
if(isset($_GET['str'])){
        $str = htmlspecialchars(trim($_GET['str'],ENT_QUOTES));
}
if(isset($_GET['flag_1'])){
        $flag_1 = htmlspecialchars(trim($_GET['flag_1'],ENT_QUOTES));
}
if(isset($_GET['flag_2'])){
        $flag_2 = htmlspecialchars(trim($_GET['flag_2'],ENT_QUOTES));
}
$bind_para = '';
if(isset($_GET['str'])){
        if($flag_2 == 'int'){
                $bind_para = PDO::PARAM_INT;
                $sql = $sql . " where int_1=?";
        }else{
                $bind_para = PDO::PARAM_STR;
                $sql = $sql . " where char_1=?";
        }
}
$sth = $conn->prepare($sql);
if($flag_1 == 'set'){
        $sth->bindValue(1, $str, $bind_para);
}
$sth->execute();
echo 'Str : <input type="text" name="str" value="' . $str . '"></br>';
echo 'Set or Unset : <input type="text" name="flag_1" value="' . $flag_1 . '" size="5">';
echo '  Int or Char : <input type="text" name="flag_2" value="' . $flag_2 . '" size="5">';
echo '<input type="submit" value="Display"></form><hr>';
echo '<h3>SQL Template : ' . $sql  . '</h3>';
echo '<hr><h3>Result:</h3><table border="1"><tr><td>INT</td><td>CHAR_1</td></tr>';
while ($row = $sth->fetch()){
        echo '<tr>';
        echo '<td>' . htmlspecialchars($row['int_1'],ENT_QUOTES) . '</td>';
        echo '<td>' . htmlspecialchars($row['char_1'],ENT_QUOTES) . '</td>';
        echo '</tr>';
}
echo '</table></body></html>';
?>
```

図 3.1-1 : 検証コード、

プレースフォルダの値と、それをセットする値の二つを指定して、

mySQL に準備済み SQL 文で接続する PHP

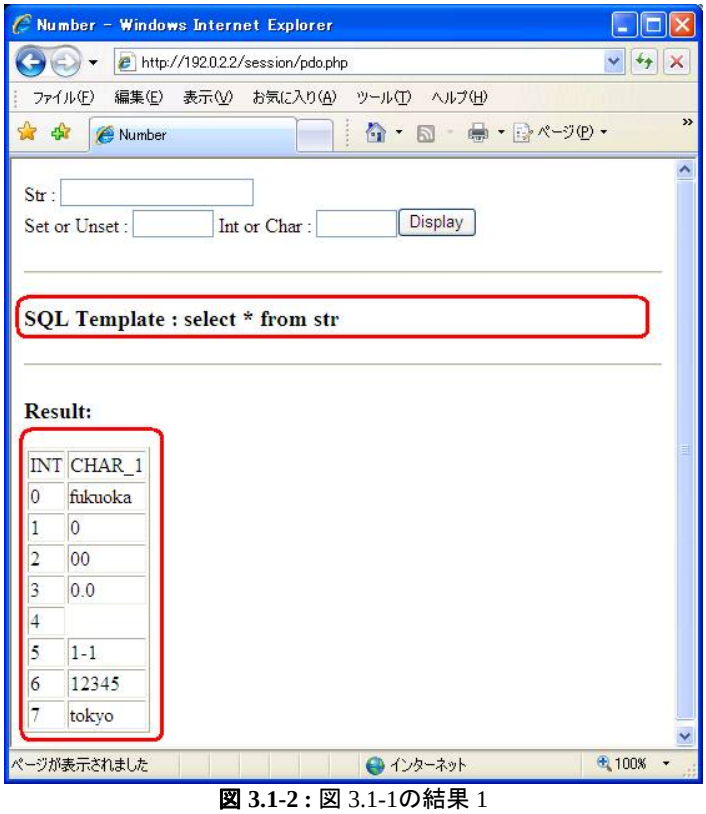

図 3.1-2 : 図 3.1-1の結果 1

何も指定しないと、プレースフォルダなしの全件検索となる

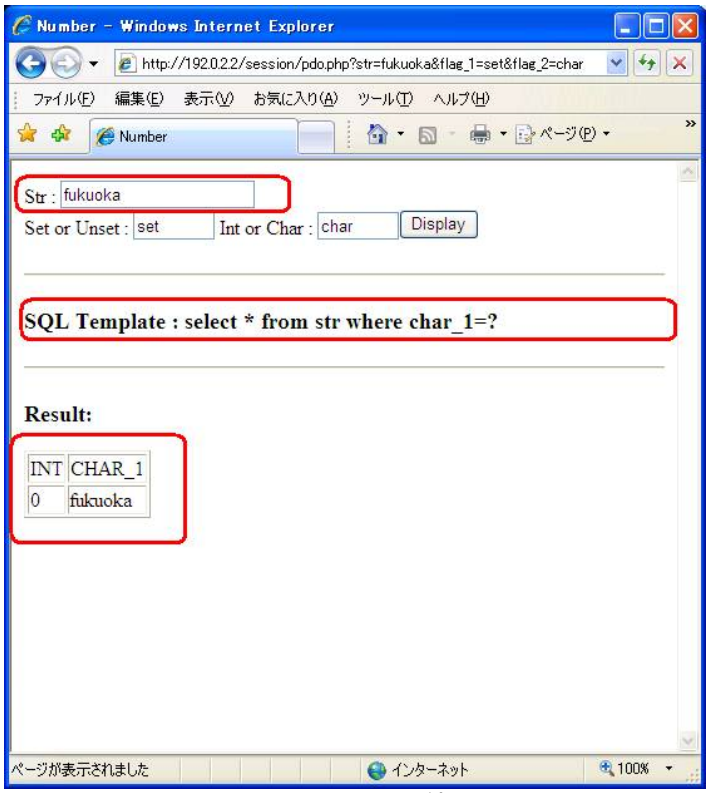

図 3.1-3 : 図 3.1-1の結果 2

プレースフォルダ(文字列型)を指定することで、該当レコードだけがヒットする

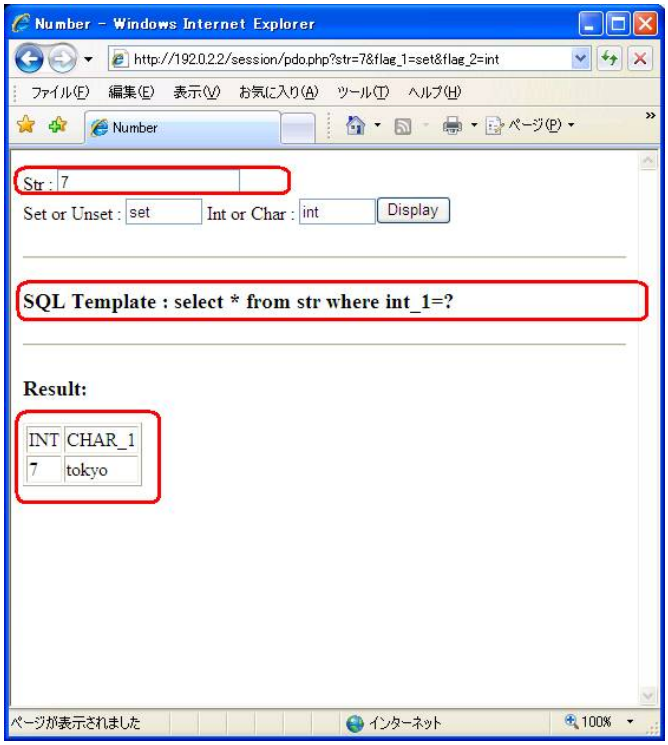

図 3.1-4 : 図 3.1-1の結果 3

プレースフォルダ(数値型)を指定することで、該当レコードだけがヒットする

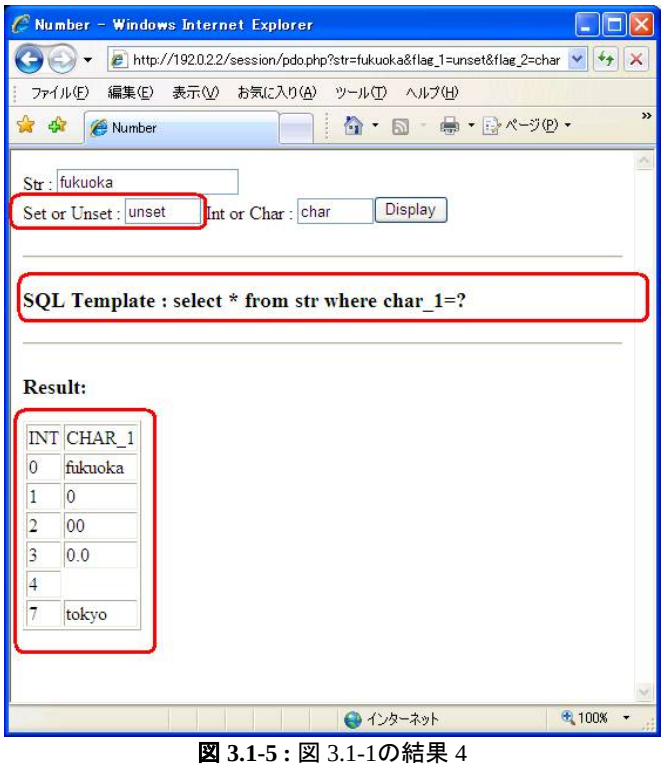

図 3.1-5 : 図 3.1-1の結果 4

文字列型のプレースフォルダ付の準備済みSQL文に対して値をセットしない場合、

文字列のカラム(空文字列を含)と値が「0」のレコードが出力される

(数値っぽくなるレコードが除外されている)

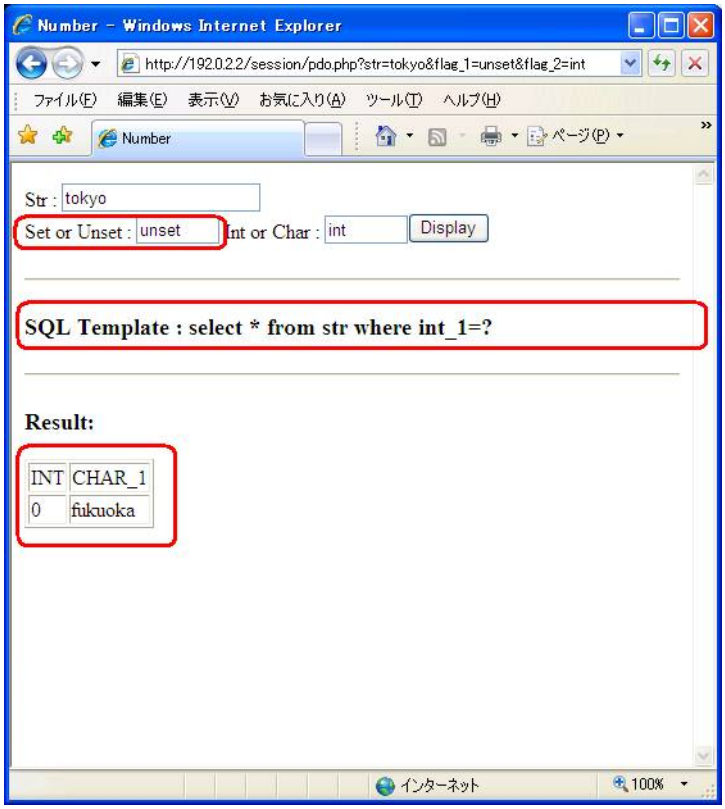

図 3.1-6 : 図 3.1-1の結果 5

数値型のプレースフォルダ付の準備済みSQL文に対して値をセットしない場合、

カラムの値が「0」のレコードだけが出力される

(未定義のデフォルト値として「0」が代入されている可能性がある)

3.2. PDO(PHP)の場合 2

検証環境

- CentOS 5.4
- Apache ver2.2.3
- PHP ver5.3.1
- mySQL ver5.1.42

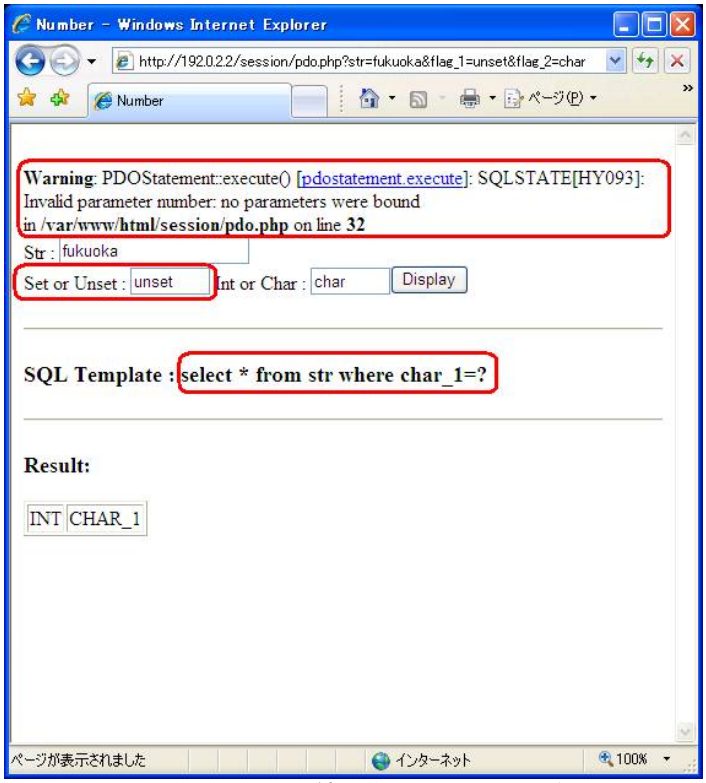

図 3.2-1 : 図 3.1-1の結果 6。PHP5.3.1 の場合、

数値型も文字列型もプレースフォルダを未定義のままにするとエラーとなる

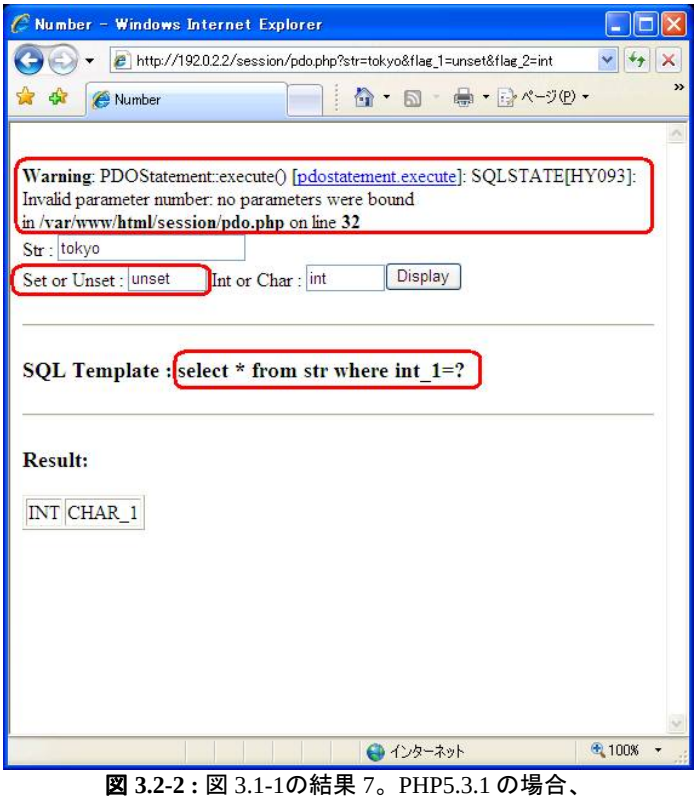

図 3.2-2 : 図 3.1-1の結果 7。PHP5.3.1 の場合、

数値型も文字列型もプレースフォルダを未定義のままにするとエラーとなる

3.3. PDO(PHP)の場合 3

検証環境

- MS-Windows 2000 Server SP4
- Apache ver2.2.14 (Win32)
- PHP ver5.1.6 for Win32
- mySQL ver5.0.77

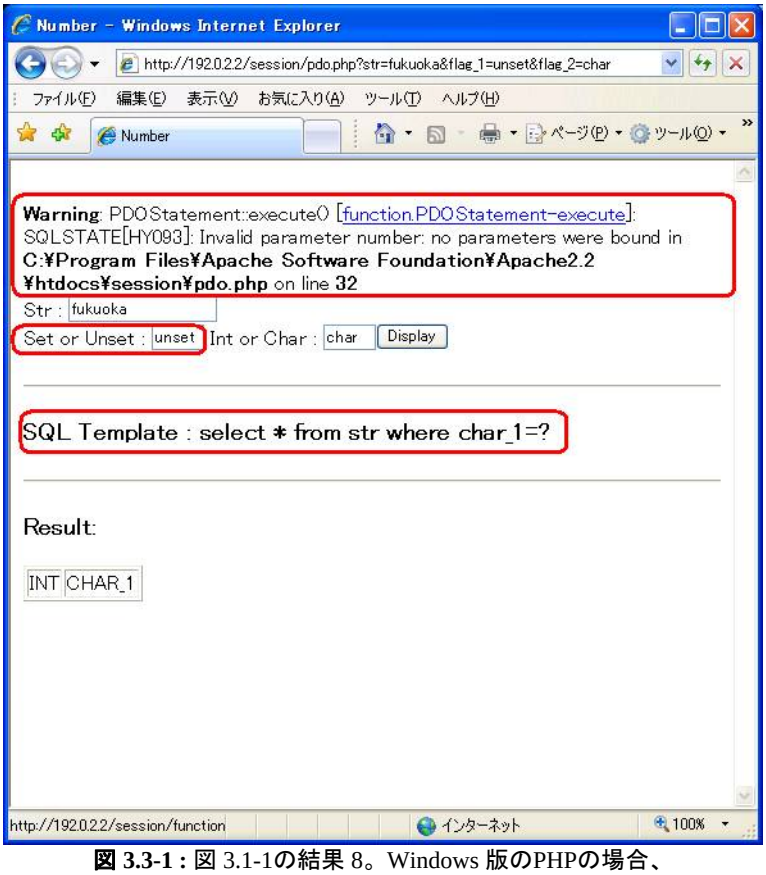

図 3.3-1 : 図 3.1-1の結果 8。Windows 版のPHPの場合、

数値型も文字列型もプレースフォルダを未定義のままにするとエラーとなる

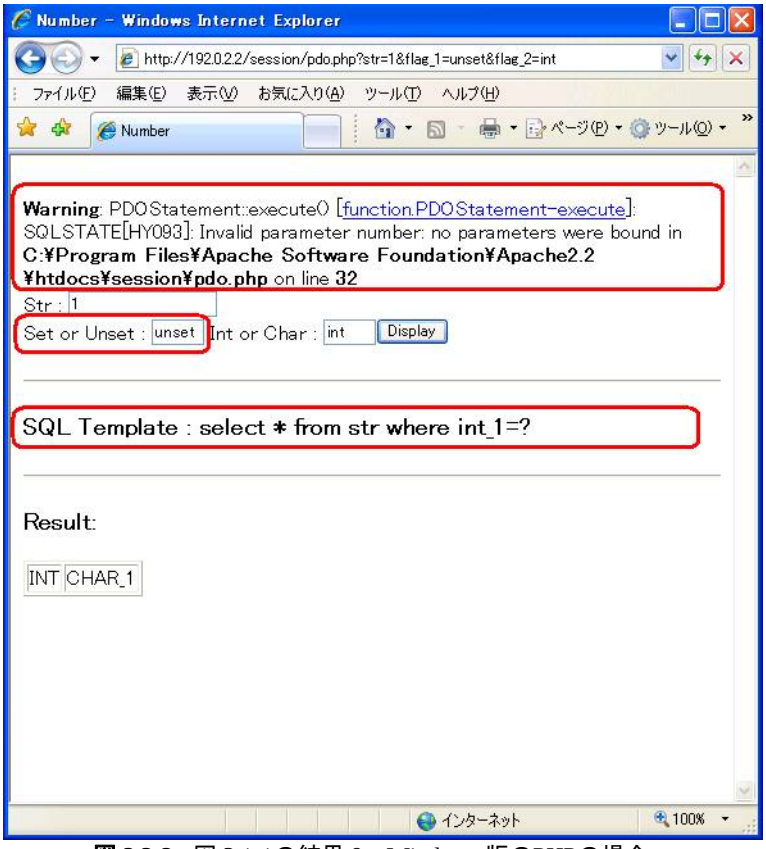

図 3.3-2 : 図 3.1-1の結果 9。Windows 版のPHPの場合、

数値型も文字列型もプレースフォルダを未定義のままにするとエラーとなる

3.4. PDO(PHP)の場合 4

検証環境

- MS-Windows 2000 Server SP4
- Apache ver2.2.14 (Win32)
- PHP ver5.1.6 for Win32
- PostgreSQL ver8.4.2

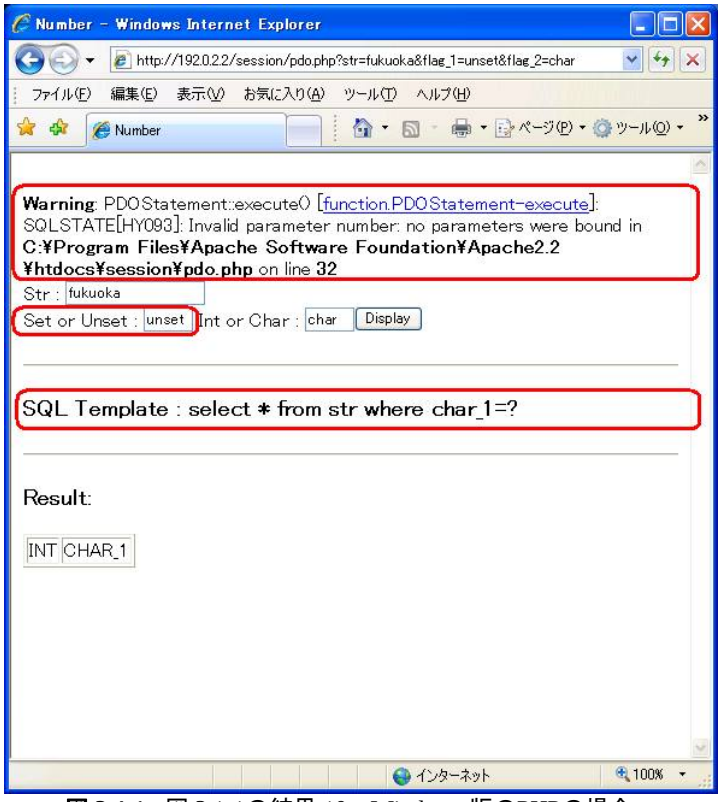

図 3.4-1 : 図 3.1-1の結果 10。Windows 版のPHPの場合、

数値型も文字列型もプレースフォルダを未定義のままにするとエラーとなる

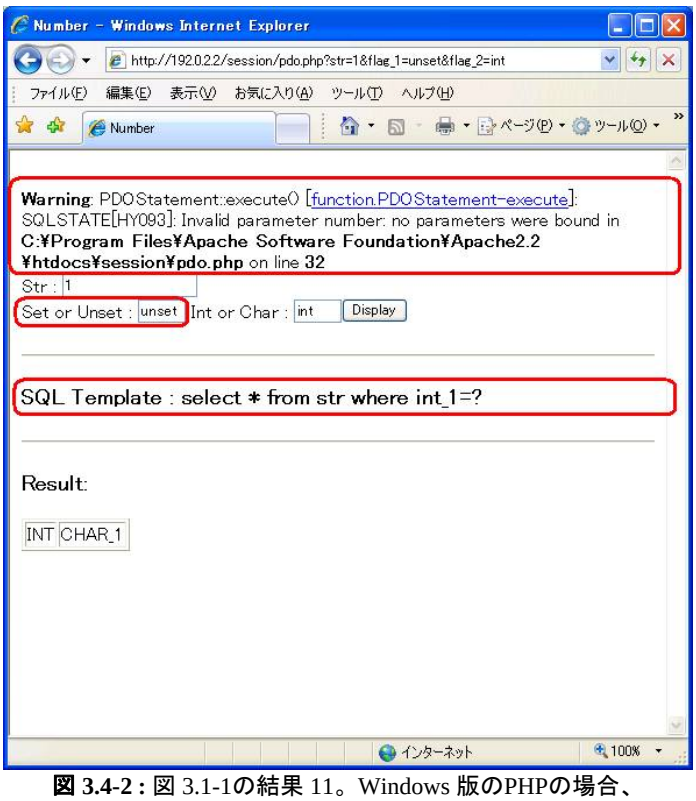

図 3.4-2 : 図 3.1-1の結果 11。Windows 版のPHPの場合、

数値型も文字列型もプレースフォルダを未定義のままにするとエラーとなる

3.5. ADO(ASP)の場合

検証環境

- MS-Windows Server 2003R2 SP2
- MS-SQL Server2005

```
<%
 Option Explicit
 Dim ADOParam
 Dim ADOCmd
 Dim DBDescription
 Dim DBObj
 Dim RSObj
 Dim SQLStr
 Dim SQLStrParam
 Dim FlgStr
 Dim i
 Dim iMax
 SQLStr = "SELECT * FROM testtbl"
 SQLStrParam = Request("SQLStrParam")
 FlgStr = Request("FlgStr")
 If 0 < Len(SQLStrParam) Then
  SQLStr = SQLStr + " WHERE uname=?"
 End If
 Response.Write("準備済み SQLStr=「" & Server.HTMLEncode(SQLStr) & "」<HR>")
 Response.Write("SQLStrParam=「" & Server.HTMLEncode(SQLStrParam) & "」<HR>")
%>
<html><body><FORM ACTION="" METHOD="get"><TABLE BORDER="1">
 <TR><TH>SQLStrParam</TH><TD><INPUT TYPE="text" NAME="SQLStrParam"
VALUE="<%=Server.HTMLEncode(SQLStrParam)%>"></TD></TR>
 <TR><TH>FlgStr</TH><TD><INPUT TYPE="text" NAME="FlgStr"
VALUE="<%=Server.HTMLEncode(FlgStr)%>"></TD></TR>
</TABLE><INPUT TYPE="submit"></FORM>
<%
 Set DBObj = Server.CreateObject("ADODB.Connection")
 DBObj.Open "SQLServer2005","testuser","testuserP"
 Set ADOCmd = Server.CreateObject("ADODB.Command")
 ADOCmd.CommandText = SQLStr
 ADOCmd.CommandType = 1
 ADOCmd.ActiveConnection = DBObj
 If FlgStr = "set" Then
  Set ADOParam = Server.CreateObject("ADODB.Parameter")
  ADOParam.Value = SQLStrParam
  ADOParam.Size = Len(SQLStrParam)
  ADOParam.Type = 200
  ADOCmd.Parameters.Append ADOParam
 End If
 Set RSObj = ADOCmd.Execute()
 If Not RSObj.EoF Then
  Response.Write("<HR><TABLE BORDER=""1"">")
  iMax = RSObj.Fields.Count -1
  Response.Write("<TR>")
  For i=0 To iMax
   Response.Write("<TH>" & RSobj(i).Name & "</TH>")
```

```
  Next
  Response.Write("</TR>")
  While Not RSObj.EoF
   Response.Write("<TR>")
   For i=0 To iMax
    Response.Write("<TD>" & Server.HTMLEncode(GetValue(RSObj(i))) & "</TD>")
   Next
   Response.Write("</TR>")
   RSObj.MoveNext
  Wend
  Response.Write("</TABLE>")
 End If
 RSObj.close
 Set RSObj = Nothing
 DBObj.Close
 Set DBObj = Nothing

 Function GetValue(iObj)
  On Error Resume Next
  Dim ansStr
  ansStr = ""
  ansStr = CStr(iObj)
  GetValue = ansStr
 End Function
%>
</body></html>
```

図 3.5-1 : 検証コード、

プレースフォルダの値と、それをセットする値の二つを指定して、

MS-SQL Server2005 に ADO の準備済み SQL 文で接続する ASP

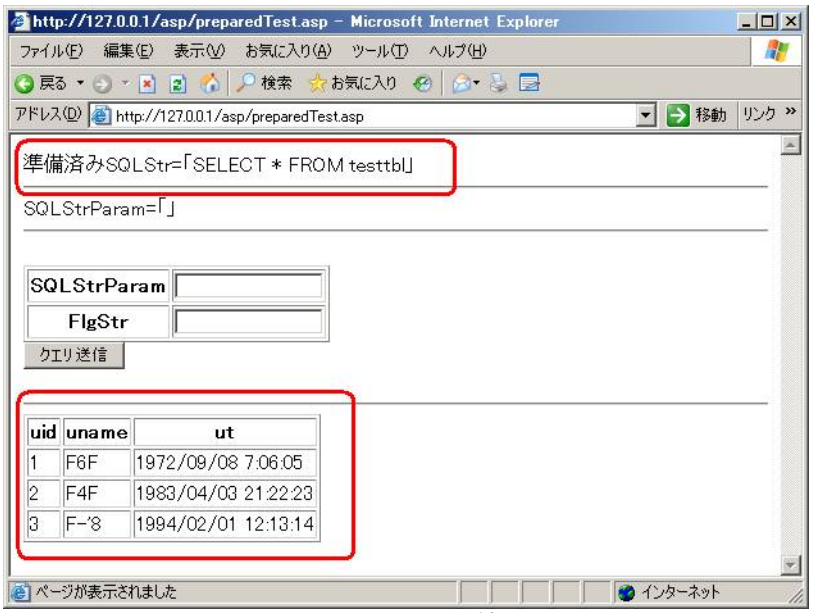

図 3.5-2 : 図 3.5-1の結果 1

何も指定しないと、プレースフォルダなしの全件検索となる

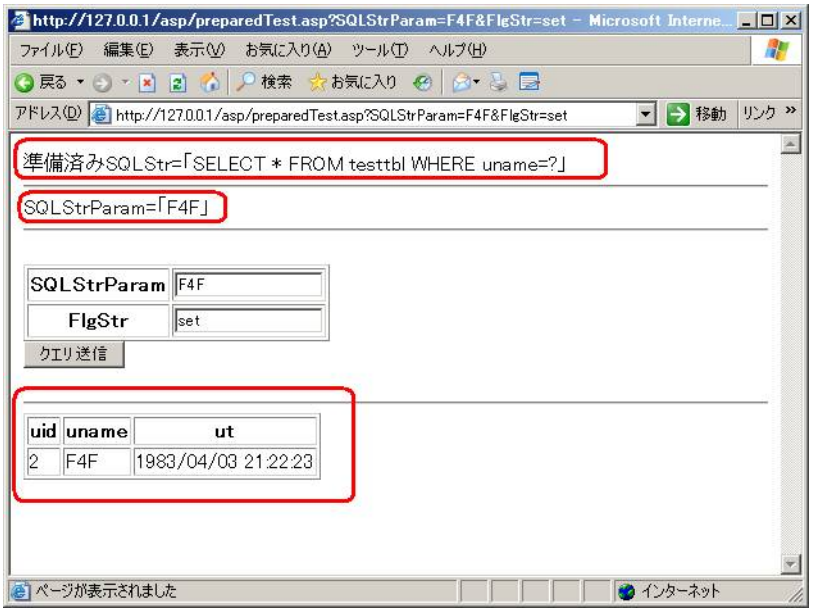

図 3.5-3 : 図 3.5-1の結果 2

プレースフォルダを指定することで、該当レコードだけがヒットする

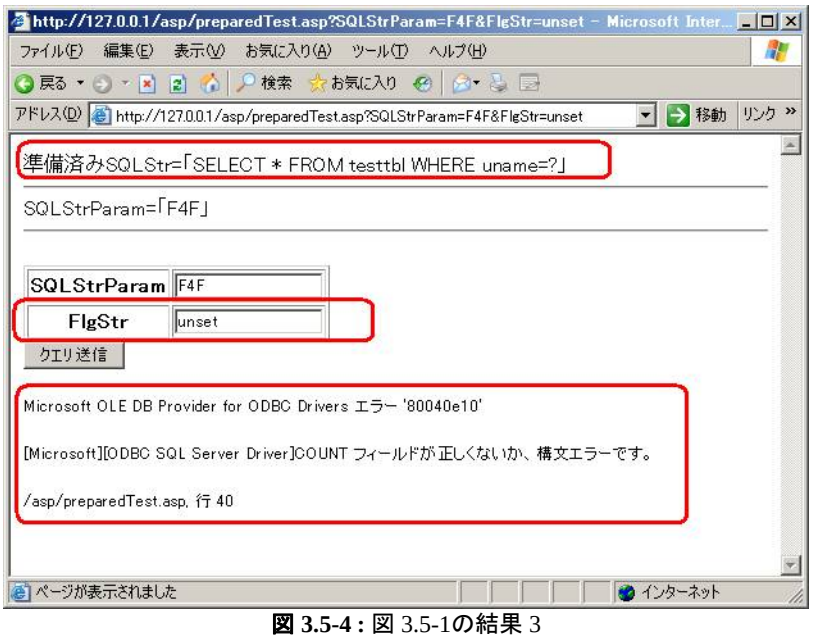

図 3.5-4 : 図 3.5-1の結果 3

プレースフォルダ付の準備済みSQL文に対して、値をセットしない場合はエラーとなる

3.6. ADO.NET(ASP.NET)の場合

検証環境

- MS-Windows Server 2008 SP2
- MS-C# 2.0 (C# 2005 Compiler version 8.00.50727.4016)
- MS-SQL Server2005

```
using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Data;
using System.Data.SqlClient;

class testWeb{
 public static void Main(string[] args){
  Console.WriteLine("Prepared SQL Test");
  String connStr = "server=tcp:127.0.0.1,1433;User ID=testuser;pwd=testuserP;";
  SqlConnection connObj = new SqlConnection(connStr);
  connObj.Open();
  Console.WriteLine("usage : sqltest.exe [Param] [Flg(IsSet?)] [Type(intORchar)]");
  String paramStr = null;
  String FlgStr = null;
  if(2 < args.Length){
   if(args[1] == "set"){
    paramStr = args[0];
   }
   FlgStr = args[2];
  }
  String sqlStr = "SELECT * FROM testdb..testtbl";
  if(FlgStr == "int"){
   sqlStr += " WHERE uid=@Key";
  }else if(FlgStr == "char"){
   sqlStr += " WHERE uname=@Key";
  }
  Console.WriteLine("SQL = " + sqlStr);
  SqlCommand cmdObj = new SqlCommand(sqlStr,connObj);
  cmdObj.CommandType = CommandType.Text;
  if(paramStr != null && FlgStr != null){
   if(FlgStr == "int"){
    cmdObj.Parameters.Add("@Key",SqlDbType.Int,50);
   }else{
    cmdObj.Parameters.Add("@Key",SqlDbType.NVarChar,50);
   }
   cmdObj.Parameters["@Key"].Value = paramStr;
  }
  SqlDataReader readObj = cmdObj.ExecuteReader();
  int iMax = readObj.FieldCount;
  while(readObj.Read()){
   for(int i=0;i<iMax;i++){
    Console.Write(readObj[i].ToString());
    Console.Write(" - ");
   }
   Console.Write("¥n");
  }
 }
}
```

図 3.6-1：検証コード

第一引数にプレースフォルダの値、第二引数にプレースフォルダに値をセットするかどうか

第三引数は、検索対象が文字列か数値型かどうかをセットする

```
C:¥>csc sqltest.cs
Microsoft(R) Visual C# 2005 Compiler version 8.00.50727.4016
for Microsoft(R) Windows(R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.


C:¥>sqltest.exe
Prepared SQL Test
usage : sqltest.exe [Param] [Flg(IsSet?)] [Type(intORchar)]
SQL = SELECT * FROM testdb..testtbl
1 - F6F - 1972/09/08 7:06:05 -
2 - F4F - 1983/04/03 21:22:23 -
3 - F-'8 - 1994/02/01 12:13:14 -

C:¥>sqltest.exe F6F set char
Prepared SQL Test
usage : sqltest.exe [Param] [Flg(IsSet?)] [Type(intORchar)]
SQL = SELECT * FROM testdb..testtbl WHERE uname=@Key
1 - F6F - 1972/09/08 7:06:05 -

C:¥>sqltest.exe F6F unset char
Prepared SQL Test
usage : sqltest.exe [Param] [Flg(IsSet?)] [Type(intORchar)]
SQL = SELECT * FROM testdb..testtbl WHERE uname=@Key

ハンドルされていない例外: System.Data.SqlClient.SqlException: スカラ変数 "@Key"を宣言してください。
   場所 System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean
breakConnection)
   場所 System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean
breakConnection)
   場所 System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject
stateObj)
   場所 System.Data.SqlClient.TdsParser.Run(RunBehavior runBehavior, SqlCommandcmdHandler,
SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject
stateObj)
   場所 System.Data.SqlClient.SqlDataReader.ConsumeMetaData()
   場所 System.Data.SqlClient.SqlDataReader.get_MetaData()
   場所 System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior
runBehavior, String resetOptionsString)
   場所 System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior,
RunBehavior runBehavior, Boolean returnStream, Boolean async)
   場所 System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior,
RunBehavior runBehavior, Boolean returnStream, String method, DbAsyncResult result)
   場所 System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior,
RunBehavior runBehavior, Boolean returnStream, String method)
   場所 System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String
method)
   場所 System.Data.SqlClient.SqlCommand.ExecuteReader()
   場所 testWeb.Main(String[] args)
```

図 3.6-2 : 図 3.6-1の結果。パラメータ変数(nvarchar型)に値をセットしないとエラーとなる

```
C:¥>sqltest.exe 2 set int
Prepared SQL Test
usage : sqltest.exe [Param] [Flg(IsSet?)] [Type(intORchar)]
SQL = SELECT * FROM testdb..testtbl WHERE uid=@Key
2 - F4F - 1983/04/03 21:22:23 -

C:¥>sqltest.exe 2 unset int
Prepared SQL Test
usage : sqltest.exe [Param] [Flg(IsSet?)] [Type(intORchar)]
SQL = SELECT * FROM testdb..testtbl WHERE uid=@Key

ハンドルされていない例外: System.Data.SqlClient.SqlException: スカラ変数 "@Key"を宣言してください。
   場所 System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean
breakConnection)
   場所 System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean
breakConnection)
   場所 System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject
stateObj)
   場所 System.Data.SqlClient.TdsParser.Run(RunBehavior runBehavior, SqlCommandcmdHandler,
SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject
stateObj)
   場所 System.Data.SqlClient.SqlDataReader.ConsumeMetaData()
   場所 System.Data.SqlClient.SqlDataReader.get_MetaData()
   場所 System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior
runBehavior, String resetOptionsString)
   場所 System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior,
RunBehavior runBehavior, Boolean returnStream, Boolean async)
   場所 System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior,
RunBehavior runBehavior, Boolean returnStream, String method, DbAsyncResult result)
   場所 System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior,
RunBehavior runBehavior, Boolean returnStream, String method)
   場所 System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String
method)
   場所 System.Data.SqlClient.SqlCommand.ExecuteReader()
   場所 testWeb.Main(String[] args)
```

図 3.6-3：図 3.6-1の結果。パラメータ変数(integer型)に値をセットしないとエラーとなる

3.7. DBI(perl)の場合

検証環境

- CentOS5.1
- perl ver5.8.8
- DBI ver1.607
- DBD-mysql ver4.008
- mySQL ver5.0.67

```
# cat perl-prepare.pl
#! /usr/bin/perl
use strict;
use warnings;
use DBI;

my ${db_source} = "DBI:mysql:database=testdb;host=127.0.0.1;port=3306";
my ${sqlstrparam} = $ARGV[0];
my ${sqlflg} = $ARGV[1];
my ${sqlstr} = "SELECT * FROM testdb.testtbl WHERE username=?";
my ${dbh} = DBI->connect(${db_source}, 'testuser', 'testuserP') || die DBI::errstr;
my ${sth} = ${dbh}->prepare(${sqlstr}) || die ${dbh}->errstr;
my ${result_set};
print "SQL  = " . ${sqlstr} . "¥n";
print "Param= " . ${sqlstrparam} ."¥n";
print "Flg  = " . ${sqlflg} . "¥n";
if(${sqlflg} =~ /^set$/){
 ${result_set} = ${sth}->execute(${sqlstrparam}) || die ${sth}->errstr;
}else{
 ${result_set} = ${sth}->execute() || die ${sth}->errstr;
}
if(${result_set}){
 my ${rows} = ${sth}->rows;
 print "${rows} records found¥n";
 my $names = ${sth}->{'NAME'};
 my ${num} = ${sth}->{'NUM_OF_FIELDS'};
 for(my ${j}=0; ${j} < ${num}; ${j}++){
  print "$$names[${j}],";
 }
 print "¥n";
 for(my ${i}=0; ${i} < ${rows}; ${i}++){
  my @result = ${sth}->fetchrow_array;
  my ${kosu} = @result;
  for(my ${j}=0; ${j} < ${kosu}; ${j}++){
   print $result[${j}] . ",";
  }
  print "¥n";
 }
}else{
 print "No Data";
}
${sth}->finish;
${dbh}->disconnect;
__END__
```

図 3.7-1：検証コード

第一引数にプレースフォルダの値、第二引数にプレースフォルダに値をセットするかどうか

```
# perl perl-prepare.pl sanaki set
SQL  = SELECT * FROM testdb.testtbl WHERE username=?
Param= sanaki
Flg  = set
1 records found
id,username,userdate,
1,sanaki,1972-09-03 12:13:14,

# perl perl-prepare.pl '' set
SQL  = SELECT * FROM testdb.testtbl WHERE username=?
Param=
Flg  = set
1 records found
id,username,userdate,
3,,2010-01-08 12:14:55,

# perl perl-prepare.pl sansaki unset
SQL  = SELECT * FROM testdb.testtbl WHERE username=?
Param= sanaki
Flg  = unset
0 records found
id,username,userdate,
```

図 3.7-2 : 図 3.7-1の結果。

最初と二つ目は「username=sanaki」と「username=''」のレコードが表示され正常動作している

最後はプレースフォルダを未定義で実行した結果だが、結果レコードが返ってこないという結果になった

3.8. Javaの場合

検証環境

- JDK ver1.5.0_16
- mySQL connector/J ver5.1.10
- mySQL ver5.0.67

```
# cat Test.java
import java.sql.*;
// import SQLite.*;
// import com.mysql.*;

class Test{
 public static void main(String[] args){
  String dburl = "jdbc:sqlite://home/SQLite/testdb.db";
  dburl = "jdbc:mysql://localhost/testdb?useUnicode=true&characterEncoding=SJIS";
  String sqlstr = "SELECT * FROM testtbl";
  String param = null;
  boolean IsSet = false;
  boolean DataTypeIsInt = false;
  if(2 < args.length){
   param = args[0];
   if(args[1].equals("set") == true){
    IsSet = true;
   }
   if(args[2].equals("int") == true){
    DataTypeIsInt = true;
    sqlstr += " WHERE id=?";
   }else{
    sqlstr += " WHERE username=?";
   }
  }
  System.out.println("SQL = " + sqlstr);
  if(IsSet == true){
   System.out.println("param = " + param);
  }
  int ParamInt = 0;
  try{
   ParamInt = Integer.parseInt(param);
  }catch(NumberFormatException e){
   ParamInt = 0;
  }
  if(sqlstr != null){
   try {
//    Class.forName("SQLite.JDBCDriver");
//    Class.forName("org.gjt.mm.mysql.Driver");
    Class.forName("com.mysql.jdbc.Driver");
   }catch(ClassNotFoundException e){
    errPrint(e,"Fail load JDBC Driver");
   }
   try{
    Connection con = DriverManager.getConnection(dburl,"testuser","testuserP");
    System.out.println("Connection Success");
    try{
```

```
      PreparedStatement prepStmt = con.prepareStatement(sqlstr);
      try{
       if(IsSet == true){
        if(DataTypeIsInt == true){
         prepStmt.setInt(1,ParamInt);
        }else{
         prepStmt.setString(1,param);
        }
       }
       ResultSetMetaData rsmd;
       int j;
       int jMin;
       int jMax;
       ResultSet rs = prepStmt.executeQuery();
       rsmd = rs.getMetaData();
       jMax = rsmd.getColumnCount();
       jMin = 1;
       for(j=1;j<=jMax;j++){
        System.out.print(rsmd.getColumnLabel(j) + ",");
       }
       System.out.println("");
       while (rs.next()) {
        for(j=1;j<=jMax;j++){
         System.out.print(rs.getString(j) + ",");
        }
        System.out.println("");
       }
       rs.close();
      }catch(SQLException e){
       errPrint(e,"Error ! ResultSet Display error");
      }
      prepStmt.close();
     }catch(SQLException e ){
      errPrint(e,"Error!! SQL Execute error");
     }
     con.close();
    }catch(SQLException e ){
     errPrint(e,"Connection Error!!!");
    }
   }
  }
  static void errPrint(java.lang.Exception e,String errStr){
   System.out.println(errStr);
   e.printStackTrace();
  }
}
```

図 3.8-1：検証コード

第一引数にプレースフォルダの値、第二引数にプレースフォルダに値をセットするかどうか

第三引数は、プレースフォルダが「int」か「varchar」かを指定する

```
# javac -version
javac 1.5.0_16

# java -version
java version "1.5.0_16"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_16-b02)
Java HotSpot(TM) Client VM (build 1.5.0_16-b02, mixed mode, sharing)

# javac Test.java

# java -classpath  .:/usr/local/share/java/mysql-connector-java-5.1.10-bin.jar Test
SQL = SELECT * FROM testtbl
Connection Success
id,username,userdate,
1,sanaki,1972-09-03 12:13:14.0,
2,suzuki,1981-10-13 14:15:16.0,
3,,2010-01-08 12:14:55.0,
4,saito,2010-01-08 12:15:30.0,

# java -classpath  .:/usr/local/share/java/mysql-connector-java-5.1.10-bin.jar Test 1 set int
SQL = SELECT * FROM testtbl WHERE id=?
param = 1
Connection Success
id,username,userdate,
1,sanaki,1972-09-03 12:13:14.0,

# java -classpath  .:/usr/local/share/java/mysql-connector-java-5.1.10-bin.jar Test suzuki
set char
SQL = SELECT * FROM testtbl WHERE username=?
param = suzuki
Connection Success
id,username,userdate,
2,suzuki,1981-10-13 14:15:16.0,

# java -classpath  .:/usr/local/share/java/mysql-connector-java-5.1.10-bin.jar Test suzuki
unset char
SQL = SELECT * FROM testtbl WHERE username=?
Connection Success
com.mysql.jdbc.PreparedStatement@b02e7a: SELECT * FROM testtbl WHERE username=** NOT
SPECIFIED **
Error ! ResultSet Display error
java.sql.SQLException: No value specified for parameter 1
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:1055)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:956)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:926)
        at com.mysql.jdbc.PreparedStatement.checkAllParametersSet(PreparedStatement.java:2513)
        at com.mysql.jdbc.PreparedStatement.fillSendPacket(PreparedStatement.java:2489)
        at com.mysql.jdbc.PreparedStatement.fillSendPacket(PreparedStatement.java:2415)
        at com.mysql.jdbc.PreparedStatement.executeQuery(PreparedStatement.java:2169)
        at Test.main(Test.java:60)

# java -classpath  .:/usr/local/share/java/mysql-connector-java-5.1.10-bin.jar Test 1 unset
int
SQL = SELECT * FROM testtbl WHERE id=?
Connection Success
```

```
com.mysql.jdbc.PreparedStatement@b02e7a: SELECT * FROM testtbl WHERE id=** NOT SPECIFIED **
Error ! ResultSet Display error
java.sql.SQLException: No value specified for parameter 1
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:1055)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:956)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:926)
        at com.mysql.jdbc.PreparedStatement.checkAllParametersSet(PreparedStatement.java:2513)
        at com.mysql.jdbc.PreparedStatement.fillSendPacket(PreparedStatement.java:2489)
        at com.mysql.jdbc.PreparedStatement.fillSendPacket(PreparedStatement.java:2415)
        at com.mysql.jdbc.PreparedStatement.executeQuery(PreparedStatement.java:2169)
        at Test.main(Test.java:60)
```

図 3.8-2：図 3.8-1の結果。

プレースフォルダを未定義のままDBを呼び出すと、

「No value specified for parameter」という例外が発生している

4. まとめ

そもそもプレースフォルダを未定義のままにしておくということ自体が、非常に稀な場面であると思われる。また、今回検証した多くの環境で適切にエラーとなっており、脆弱性になってしまう場面は、非常に稀であると思われる。

しかしながら、今一度、読者諸氏が開発したシステムで、プレースフォルダが未定義の状態で SQL が実行されることがないかどうか、またその場合どのような挙動を示すのかを確認されることを推奨する。
本文書作成にあたって、検証作業者である私どももセキュリティ機能をただ使うということではなく、「適切に使う」ことが、セキュリティの基本であることを改めて実感した。

5. 検証作業者

NTT コミュニケーションズ株式会社
IT マネジメントサービス事業部ネットワークマネジメントサービス部
セキュリティオペレーションセンター
　佐名木 智貴
　本城 敏信

6. 参考

1. セキュア Web プログラミング Tips 集(出版社:株式会社ソフト・リサーチ・センター)
ISBN=978-4883732562

7. 履歴

- 2010 年 03 月 09 日 ：ver1.0 最初の公開

8. 最新版の公開URL

http://www.icto.jp/security_report/index.html

9. 本レポートに関する問合せ先

NTT コミュニケーションズ株式会社
IT マネジメントサービス事業部ネットワークマネジメントサービス部
セキュリティオペレーションセンター

e-mail: scan@ntt.com

以 上