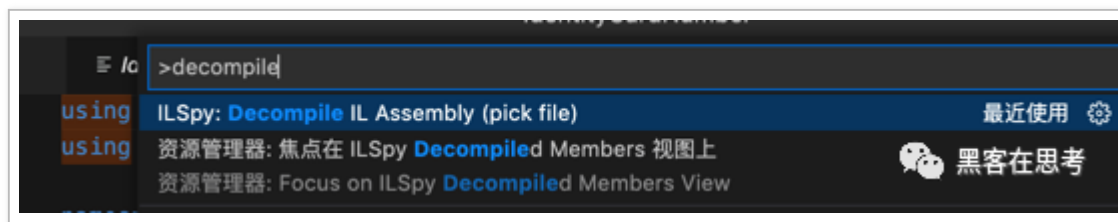


「学习记录」.NET 程序的数据库密码解密

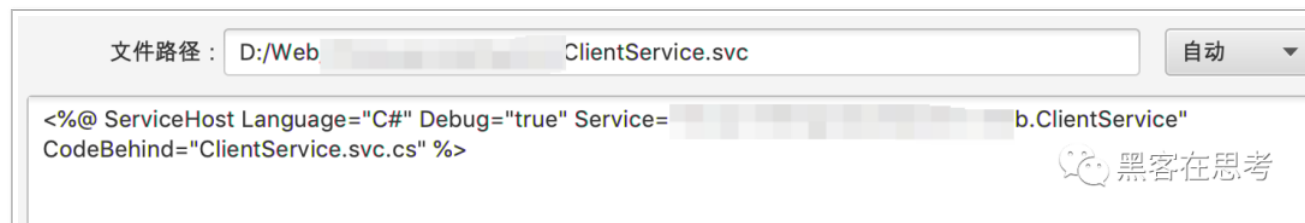
做项目过程中遇到的，关于简单的.NET 程序的数据库解密流程过程记录，学习，阿巴阿巴

PS: MAC 反编译.NET 的 DLL 可以直接在 `VSCODE` 下载插件 `ILSPY`，然后打开 `VSCODE` 的命令行输入 `>Decompile`，即可选择 DLL 反编译。



程序一

进入正题，打开 webshell 管理工具，连接小伙伴给的 shell，切换至根目录查看到存在文件 `ClientService.svc`



这里面有两个重要的参数：`Service` 和 `CodeBehind`。

- `Service` 是属性值是 WCF 的服务实现类的完全限定名。

- `CodeBehind` 是服务实现类所在的文件名。

在运行的时候，宿主程序从 svc 文件中的 Service 属性得到 WCF 服务的完全限定名，然后从配置文件中找到同名的 service，进而找到所有的 EndPoint，并根据其属性进行实例化。

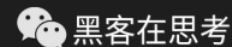
这说明此 WEB 服务为 WCF 服务，直接进入 / bin 下打开 `ClientService.svc.cs`，查看到存在 UserLogin 接口方法的实现

```
4      /// <summary>
5      /// 用户登录
6      /// type=0: 端
7      /// type=1: 端
8      /// type=2: 端
9      /// </summary>
10     public User UserLogin(string userName, string password, int type)
11     {
12         if (type == 2)
13         {
14             var judge = Business.AnonymousUsers.JudgeUserLogin(userName, password);
15
16             return new User()
17             {
18                 Credential = judge.CredentialID.ToString(),
19                 DepartmentCode,
20                 th() + judge.UserModel.AvatarUrl,
21                 rmissions.Select(p => p.Code).ToList()
22             };
23         }
24         var user = Business.AnonymousUsers.UserLogin(userName, password);
25         if (type == 0)
26         {
27             if (!user.UserModel.IsSystemUser)
28             {
29                 throw new Exception("您没有任何权限");
30             }
31         }
32         else
33         {
34             if (user.UserModel.IsSystemUser == true)
35             {
36                 throw new Exception("用户名与密码错误。");
37             }
38         }
39     }
40 }
```

可以看到会根据登陆类型的不同进入不同的处理，登陆类型为 2 时，进入 `Business.AnonymousUsers.JudgeUserLogin()` 方法处理；登陆类型为 0 时，进入 `Business.AnonymousUsers.UserLogin()` 处理，我们只关注第二个登陆类型就好。

`Business.AnonymousUsers.UserLogin()` 在 `XXX.ASystem.Business.dll` 中实现，

```
104 public static CurrentOnlineUser UserLogin(string userName, string password)
105 {
106     try
107     {
108         .SDK.User user = new Management[GlobalParameters.PlatformUrl].UserLogin(userName, password);
109         CurrentOnlineUser obj = new CurrentOnlineUser
110         {
111             CredentialID = Guid.NewGuid(),
112             LastAccessTime = DateTime.Now,
113             UserName = userName,
114             UserModel = new em.Business.PublicDataTransfer.User
115             {
116                 Age = user.Age,
117                 AvatarUrl = user.AvatarUrl,
118                 Birthday = user.Birthday,
```



发现其会实例化以下结构的 user，这里不必关注，然后进入 `GlobalParameters.PlatformUrl` 中的 `UserLogin` 方法

```
1 using System;
2
3 namespace [REDACTED].SDK
4 {
5     public class User
6     {
7         public string UserName { get; set; }
8
9         public string TrueName { get; set; }
10
11         public string [REDACTED] { get; set; }
12
13         public DateTime [REDACTED]day { get; set; }
14
15         public string [REDACTED]entCode { get; set; }
16
17         public string [REDACTED]entName { get; set; }
18
19         public int Score { get; set; }
20
21         public string [REDACTED] { get; set; }
22
23         public string [REDACTED] { get; set; }
24
25         public bool IsUser { get; set; }
26
27         public string [REDACTED] { get; set; }
28
29         public int Age { get; set; }
30     }
31 }
```

接着查看 `GlobalParameters.PlatformUrl` 为

```
219  
220     public static string PlatformUrl => ConfigurationManager.AppSettings["APIurl"];  
221
```

到 Web.Config 中查看，此配置为本地 15801 开启的 WEB 服务，

```
<appSettings>  
  <add key="APIurl" value="http://127.0.0.1:15801/" />  
  <!--<add key="APIurl" value="http://localhost:5282/" />-->  
  <add key="OutputFileClass" value="OutputFile" />  
  <add key="APIPassword" value="123456" />  
  <add key="APIIP" value="." />  
  <add key="StaticKey" value="1234567890" />  
</appSettings>
```

发现上面实例化 user 的时候调用的 DLL 名字就与我们开始反编译的 DLL 不一样了，开始的为

`XXX.ASystem.Business.dll`，而实例化 user 的时候声明的类型为 `XXX.BSystem.SDK.dll` 中实现的。

在这个 WEB 目录同级看到以 BSystem 命名的目录，那就是这个 WEB 又会去本地的 `15801` 端口进行进一步的登陆处理（套娃？）。

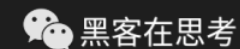
进入此目录再次查看 `ClientService.svc` 文件，bin 目录又没有这个 `ClientService.svc.cs` 文件了，所以就直接打开 Service 参数指定的 `XXX.BSystem.WEB.dll` 文件，查看 `UserLogin()` 方法，发现明显处理逻辑有点不一样

```
public User UserLogin(string userName, string password)
{
    //IL_004c: Unknown result type (might be due to invalid IL or missing references)
    //IL_0051: Unknown result type (might be due to invalid IL or missing references)
    //IL_006b: Unknown result type (might be due to invalid IL or missing references)
    //IL_0077: Unknown result type (might be due to invalid IL or missing references)
    //IL_00b5: Unknown result type (might be due to invalid IL or missing references)
    //IL_00c6: Unknown result type (might be due to invalid IL or missing references)
    //IL_00d7: Unknown result type (might be due to invalid IL or missing references)
    //IL_00ed: Unknown result type (might be due to invalid IL or missing references)
    //IL_0113: Unknown result type (might be due to invalid IL or missing references)
    //IL_0124: Unknown result type (might be due to invalid IL or missing references)
    //IL_0135: Unknown result type (might be due to invalid IL or missing references)
    //IL_0146: Unknown result type (might be due to invalid IL or missing references)
    //IL_0157: Unknown result type (might be due to invalid IL or missing references)
    //IL_0168: Unknown result type (might be due to invalid IL or missing references)
    //IL_0179: Unknown result type (might be due to invalid IL or missing references)
    //IL_018a: Unknown result type (might be due to invalid IL or missing references)
    //IL_01bb: Expected 0, but got Unknown
    CurrentOnlineUser currentOnlineUser = AnonymousUsers.UserLogin(userName, password);
    if (currentOnlineUser == null)
    {
        throw new Exception("用户名或密码错误。");
    }
    Li...usUsers.GetSystemPermissions(currentOnlineUser
    Gl...ype.登录, "用户登录", current
    Us...
    va...alID.ToString());
}
```

又进入了 `AnonymousUsers.UserLogin()`，处理，虽然和上面第一层的名字一样，但这次是进入到

`XXX.BSystem.Business.dll` 中的 `AnonymousUsers` 类，继续跟过去看


```
93 public static CurrentOnlineUser UserLogin(string userName, string password)
94 {
95     UserLoginRecord userLoginRecord = _userLoginRecords.FirstOrDefault(p => p.UserName ==
96     if (userLoginRecord != null)
97     {
98         TimeSpan timespan = DateTime.Now - userLoginRecord.LastLoginTime;
99         if (timespan.TotalSeconds < 600.0)
100         {
101             TimeSpan timeSpan2 = TimeSpan.FromSeconds(new TimeSpan(0, 0, 600).TotalSeconds - timespan);
102             throw new Exception("此用户已被锁定, 请" + (int)timeSpan2.TotalMinutes + "分" + timeSpan2.TotalSeconds + "秒后重试");
103         }
104         userLoginRecord.LastLoginTime = DateTime.Now;
105         userLoginRecord.IsLock = false;
106         userLoginRecord.TryLoginCount = 0;
107     }
108     IGlobalCache globalParameters = AccessFactory.GetGlobalParameters();
109     try
110     {
111         CurrentOnlineUser result = globalParameters.UserLogin(userName, password);
112         if (userLoginRecord != null)
113         {
114             lock (_userLoginRecords)
115             {
116                 _userLoginRecords.Remove(userLoginRecord);
117             }
118         }
119     }
120     catch { }
```




我们关注的字符串 `password` 又进入了 `globalParameters.UserLogin()`，再往上两行 `globalParameters` 是由 `IGlobalCache` 这个接口声明的，后面的 `AccessFactory.GetGlobalParameters()` 我也没找到在哪。

日了个 DJ 了，给爷整麻了，找不动了。

直接去 `IGlobalCache` 看发现确实有 `UserLogin` 方法，再去 `XXX.BSystem.Cache.dll` 中的 `GlobalCache` 类，发现确实是这个接口的实现，那就对了

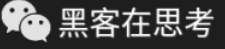

 黑客在思考

继续查看，发现终于找到了加密处理的地方

 黑客在思考

Utility.GetMD5 在 xxx.common.dll 中实现，最终发现其会在 md5 处理前的数据加上一段估计字符

```
62     public static string GetMD5(string input)
63     {
64         string text = "123456789@xj",
65         MD5CryptoServiceProvider mD5CryptoServiceProvider = new MD5CryptoService
66         byte[] bytes = Encoding.UTF8.GetBytes(text + input);
67         byte[] array = mD5CryptoServiceProvider.ComputeHash(bytes);
68         mD5CryptoServiceProvider.Clear();
69         StringBuilder stringBuilder = new StringBuilder();
70         for (int i = 0; i < array.Length; i++)
71     }
```



最后就可以指定这个盐让 hashcat 去跑密码了。

程序二

大体也非常简单，也是有解密数据库密码的需求，随便翻目录翻到文件 `AddUser.aspx`

名称 ^	修改日期	类型	大小
_vti_cnf	2021/10/19 3:31	文件夹	
AddFunctionClass	2006/9/20 9:54	ASP.NET Server...	3 KB
AddFunctionClass.aspx.cs	2006/3/28 3:18	Visual C# Sour...	2 KB
AddFunctions	2006/9/20 9:54	ASP.NET Server...	3 KB
AddFunctions.aspx.cs	2006/3/28 3:18	Visual C# Sour...	4 KB
AddObjectSecurity	2006/9/20 9:54	ASP.NET Server...	4 KB
AddObjectSecurity.aspx.cs	2006/3/28 3:18	Visual C# Sour...	2 KB
AddObjectSecurity.aspx	2006/12/21 17:55	.NET Managed R...	6 KB
AddRole	2006/9/20 9:54	ASP.NET Server...	13 KB
AddRole.aspx.cs	2006/3/28 3:18	Visual C# Sour...	20 KB
AddRole.aspx	2006/12/21 17:55	.NET Managed R...	6 KB
AddUser	2006/9/20 9:54	ASP.NET Server...	5 KB
AddUser.aspx.cs	2006/3/28 3:18	Visual C# Sour...	3 KB
AddUser.aspx	2006/12/21 17:55	.NET Managed R...	6 KB
ChooseRoles	2006/9/20 9:54	ASP.NET Server...	3 KB
ChooseRoles.aspx.cs	2006/3/28 3:18	Visual C# Sour...	2 KB
ChooseRoles.aspx	2006/12/21 17:55	.NET Managed R...	6 KB
Default	2006/9/20 9:54	ASP.NET Server...	3 KB

直接打开看旁边的.cs源文件，这个一目了然，直接看 `SecurityMgr` 中的 `CreateUser` 方法就行

```
using BizEngine.Common;
using BizEngine.Security;
using BizEngine.UI;
using System;
namespace IntelligentOffice.security
{
    public class AddUser : WebPage
    {
        private void InitializeComponent()
        {
            base.Load += new EventHandler(this.Page_Load);
        }

        protected override void OnInit(EventArgs e)
        {
            this.InitializeComponent();
            base.OnInit(e);
        }

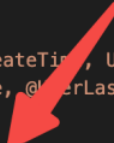
        private void Page_Load(object sender, EventArgs e)
        {
            base.VerifyPage("");
            if (base.Request.Form["username"] != null)
            {
                string text1 = "employee";
                SecurityMgr mgr1 = new SecurityMgr();
                this.AddResult = mgr1.CreateUser(base.Request.Form["username"].ToString(), base.Request.Form["userpwd"].ToString(), text1);
                if (mgr1.GetUserEmplByEmplID(base.Request.Form["EmplID"].ToString()) != null)
                {
                    base.Response.Write("<script>alert('&#x26b1&#x6b49 \u2014 \u60a8\u6240\u586b\u7684\u5458\u55df\u5df2\u7ecf\u6709\u5bf9");
                }
                else
                {
                    if (this.AddResult)
                    {

```

 黑客在思考

找到对应的 BizEngine.Security.dll , 反编译

```
463 public bool CreateUser(string UserName, string UserPwd, string UserID, string EmplID)
464 {
465     try
466     {
467         string sql = "select * from Users where UserName='" + UserName + "'";
468         BECommand bECommand = new BECommand(sql, "", "");
469         BEDataReader bEDataReader = new BEDataReader();
470         bEDataReader.SetReader(bECommand.ExecuteReader());
471         if (bEDataReader.Read())
472         {
473             bEDataReader.Close();
474             bECommand.Close();
475             return false;
476         }
477         bEDataReader.Close();
478         bECommand.Close();
479         sql = "insert into Users (UserName, UserPwd, UserCreateTime, UserLastLogon, UserLogonTimes, UserLastLogoffTime)";
480         sql += " values(@UserName, @UserPwd, @UserCreateTime, @UserLastLogon, 0, @UserTypeID)";
481         BECommand bECommand2 = new BECommand(sql, "", "");
482         bECommand2.SetParameters("@UserName", UserName);
483         bECommand2.SetParameters("@UserPwd", Encryption.Encrypt(UserPwd));
484         bECommand2.SetParameters("@UserCreateTime", DateTime.Now);
```



黑客在思考

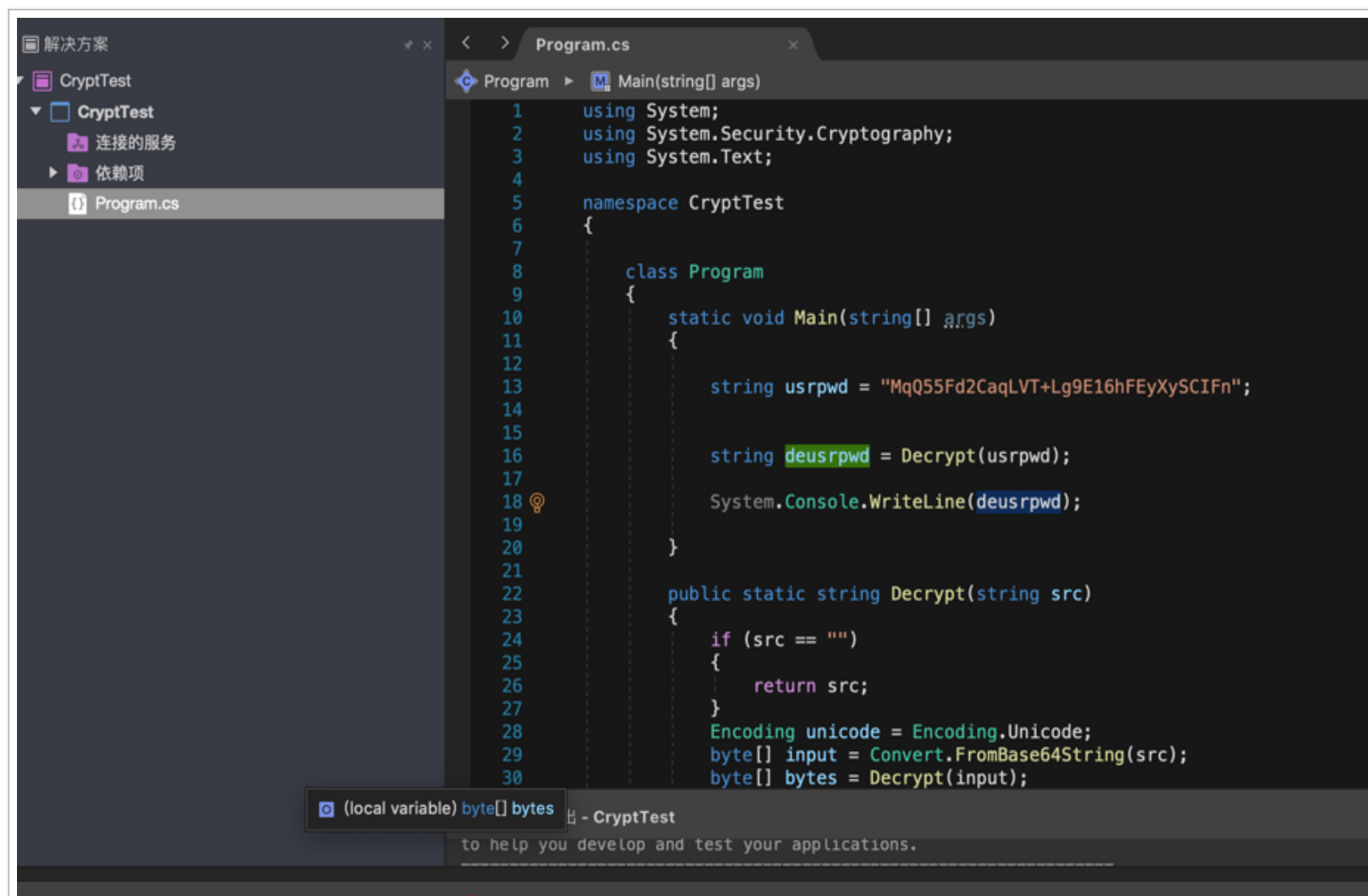
密码进入到 `Encryption.Encrypt()` ，然后找到对应的 DLL 跟过去就看到了

```
4
5 namespace [REDACTED].Common
6 {
7     public class Encryption
8     {
9         public static string Decrypt(string src)
10         {
11             if (src == "")
12             {
13                 return src;
14             }
15             Encoding unicode = Encoding.Unicode;
16             byte[] input = Convert.FromBase64String(src);
17             byte[] bytes = Decrypt(input);
18             return unicode.GetString(bytes);
19         }
20
21         public static byte[] Decrypt(byte[] input)
22         {
23             byte[] rgbKey = new byte[8] { [REDACTED] 147, 19, 128, 18 };
24             byte[] rgbIV = new byte[8] { [REDACTED], 153, 49 };
25             return new DESCryptoServiceProvider().CreateDecryptor(rgbKey, rgbIV).TransformFinalBlock(input, input.Length, 0);
26         }
27
28         public static byte[] Encrypt(byte[] input)
29         {
30             byte[] rgbKey = new byte[8] { 19, [REDACTED] 19, 128, 18 };
31             byte[] rgbIV = new byte[8] { 8, 1, [REDACTED], 49 };
32             return new DESCryptoServiceProvider().CreateEncryptor(rgbKey, rgbIV).TransformFinalBlock(input, input.Length, 0);
33         }
34
35         public static string Encrypt(string src)
36         {
37             if (src == "")
38             {
39                 return src;
40             }
41             byte[] bytes = Encoding.Unicode.GetBytes(src);
42             byte[] inArray = Encrypt(bytes);
43             return Convert.ToBase64String(inArray);
44         }
45     }
46 }
```

```
44     }  
45 }
```

进行了 DES，我们只需要把这个方法原模原样拿出来就行。

编写解密程序：



```
1  using System;  
2  using System.Security.Cryptography;  
3  using System.Text;  
4  
5  namespace CryptTest  
6  {  
7  
8      class Program  
9      {  
10         static void Main(string[] args)  
11         {  
12  
13             string usrpwd = "MqQ55Fd2CaqLVT+Lg9E16hFEyXySCIFn";  
14  
15             string deusrpwd = Decrypt(usrpwd);  
16  
17             System.Console.WriteLine(deusrpwd);  
18  
19         }  
20  
21         public static string Decrypt(string src)  
22         {  
23             if (src == "")  
24             {  
25                 return src;  
26             }  
27  
28             Encoding unicode = Encoding.Unicode;  
29             byte[] input = Convert.FromBase64String(src);  
30             byte[] bytes = Decrypt(input);
```



成功解开密码

两个案例是这两天项目中遇到的，比较简单，.NET 程序看着也比较简单，漏洞也多，那么话说回来，你打上单出什么装备？