前言

首先申明，本文只是从技术的角度来分析下怎样破解带签名的C#写的dll文件。大家如有遇到收费的软件或类库还是应该去购买正版，程序员何苦为难程序员呢。

不带签名的破解

不带签名的dll文件的破解很简单，通常有下面三个步骤：

1. 使用反编译工具对dll文件进行反编译，找到校验过期的相关代码，反编译工具可以使用ILSpy或Reflector;

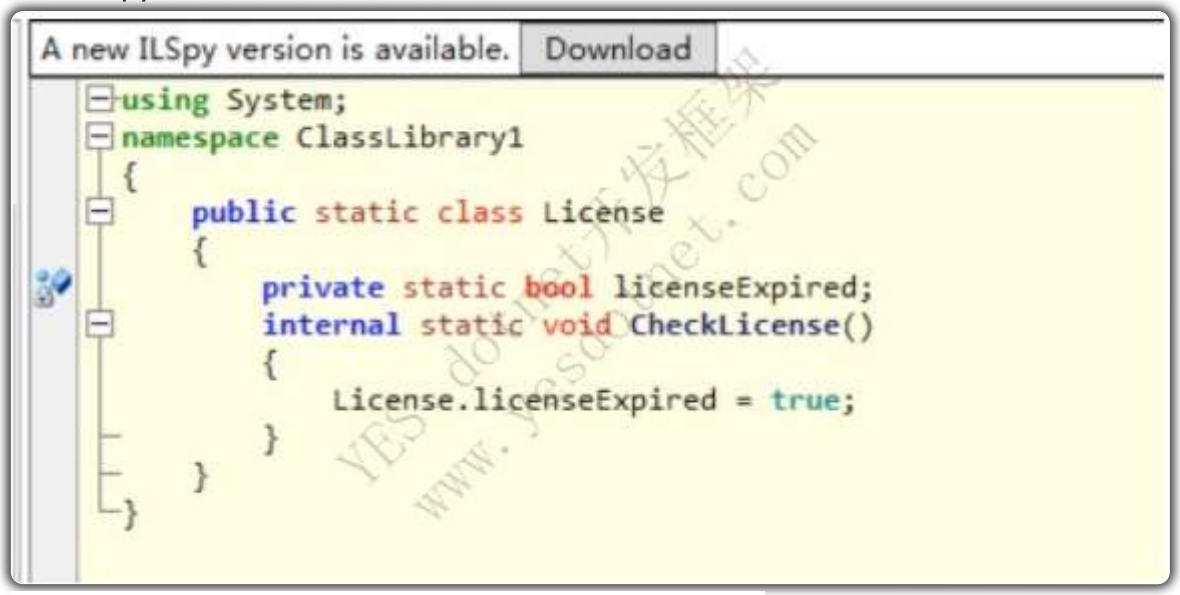
2. 使用ildasm.exe工具将dll导出成il文本文件，在该文件中找到相关的代码进行修改；

3. 使用ildasm.exe工具将修改后的il文件编译成dll文件。

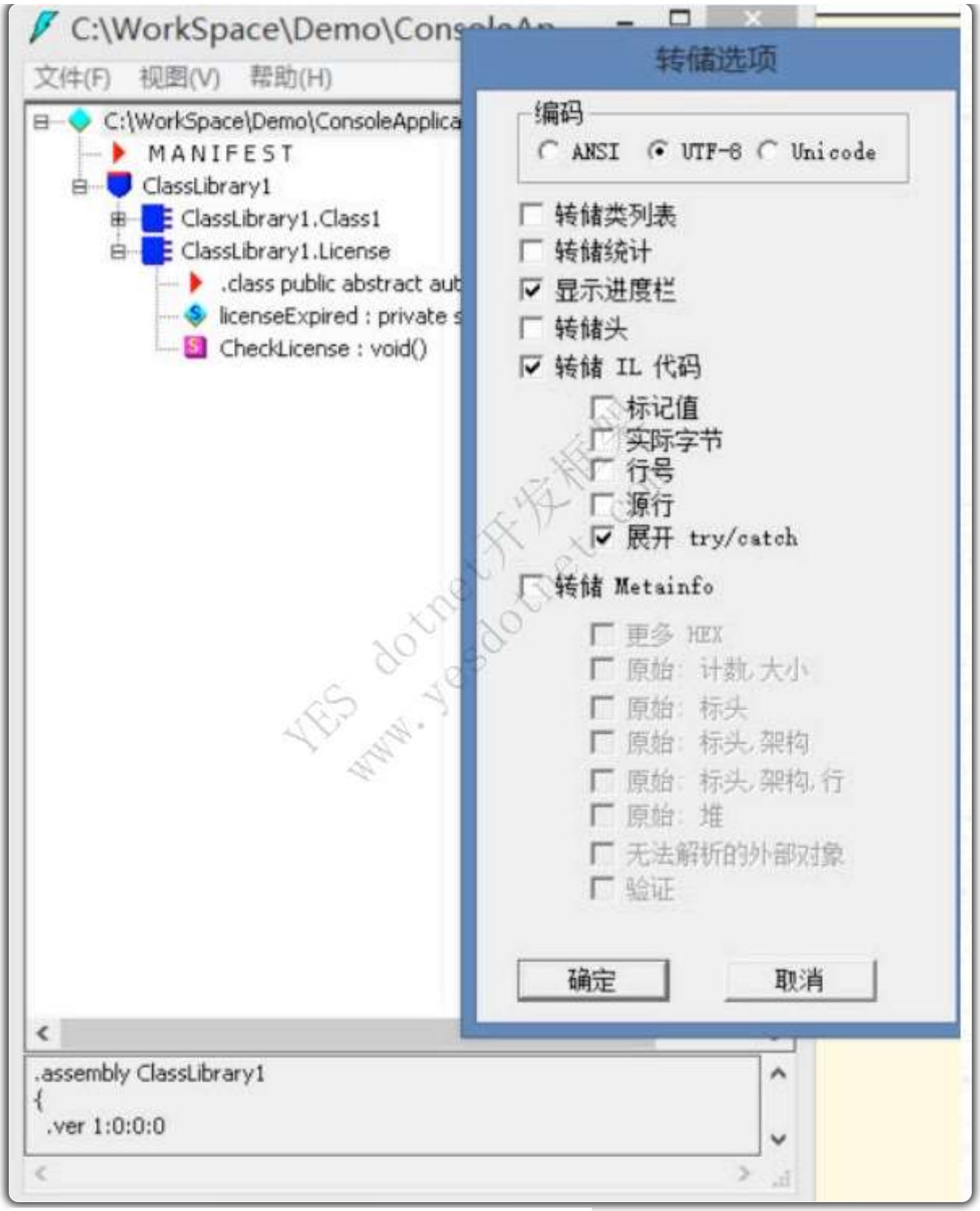
下面看一个例子，假设有ClassLibrary1.dll文件，该类库中的有关校验过期的代码如下：

```
public static class License { private static bool licenseExpired; internal static void CheckLicense() { //if语句中判断是否过期 if(true) { //标示是否过期，设置成true表示过期 License.licenseExpired = true; } } }
```

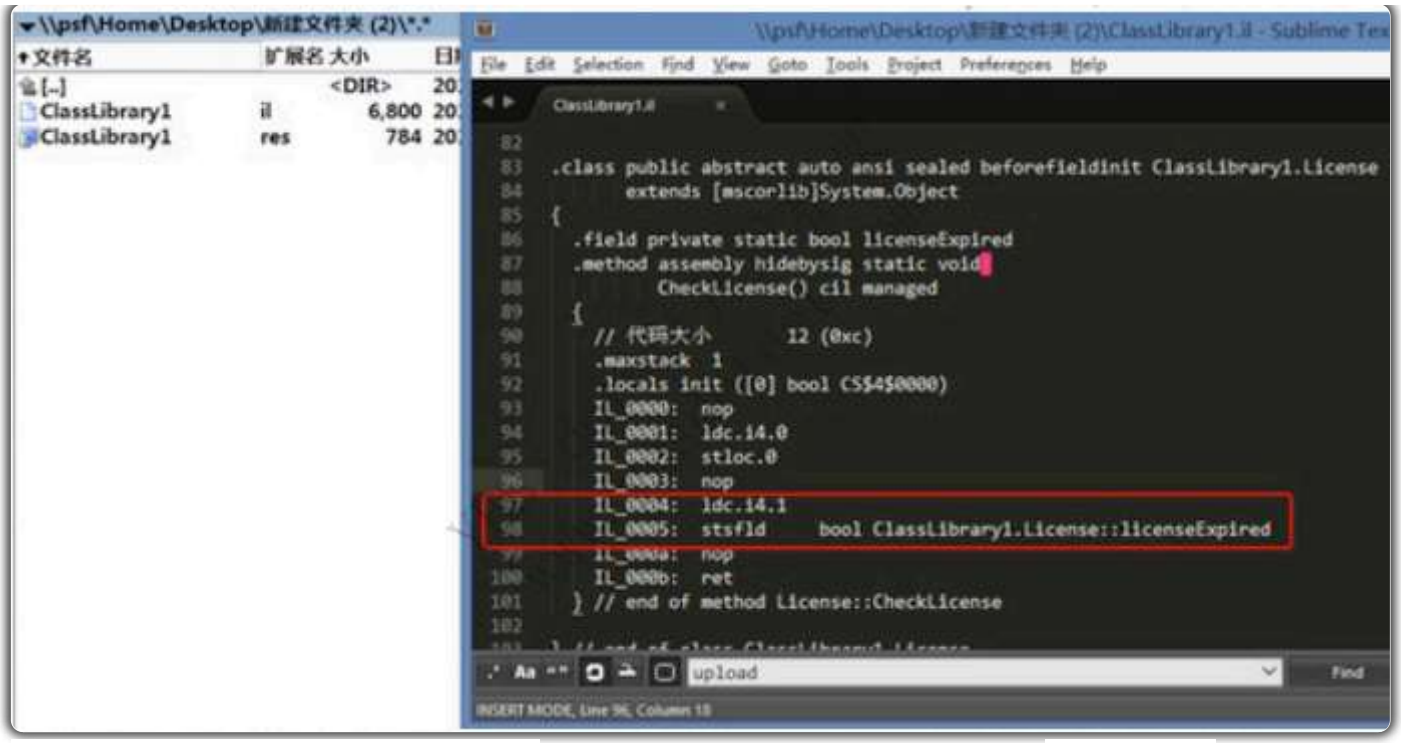
使用ILSpy进行反编译看到的代码如下：



现在使用ildasm.exe对该dll文件进行导出成il文本文件：



使用文本编辑器打开il文件，找到校验对相关代码：



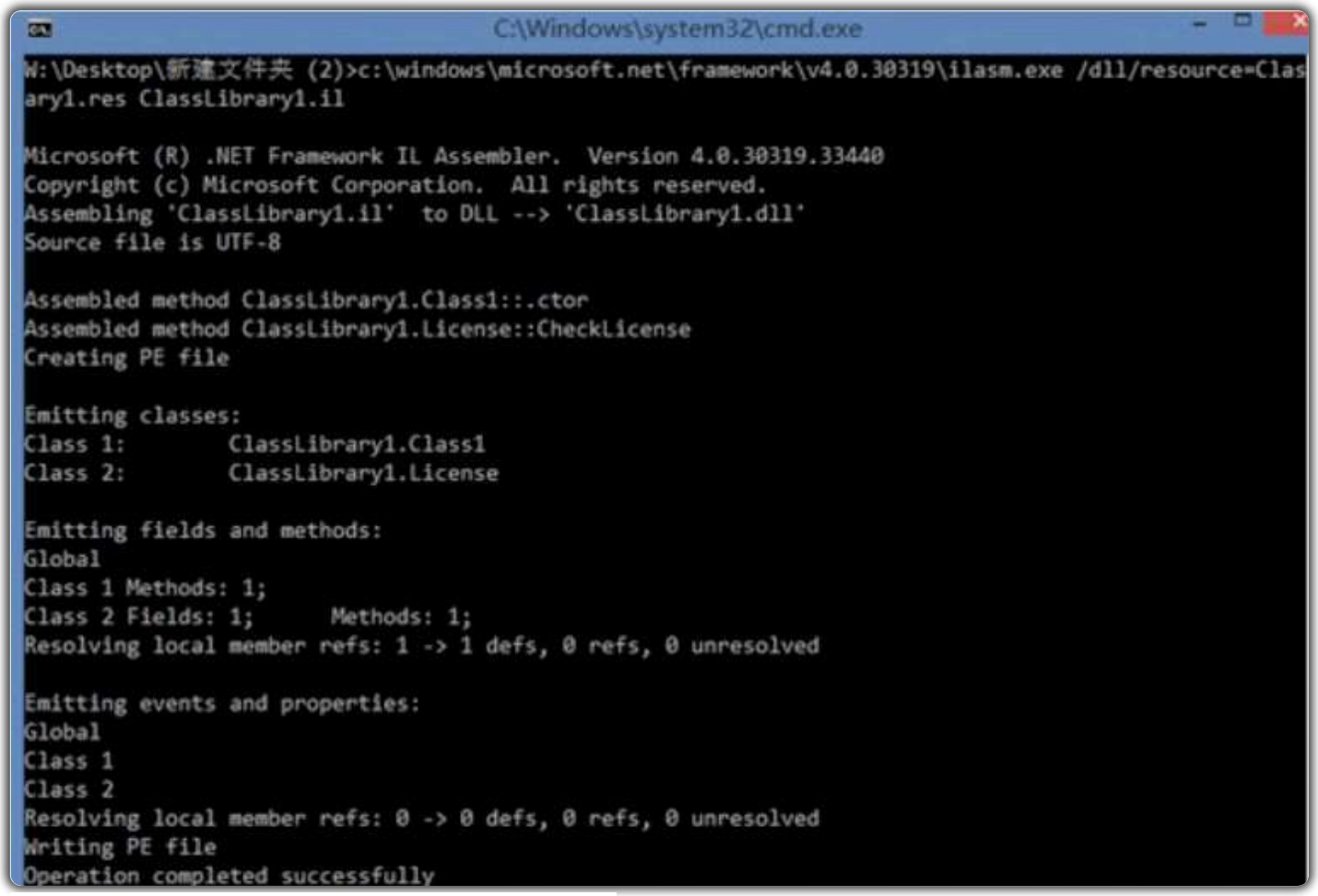
1. 上图中的红框部分代码对应的就是 `License.licenseExpired = true;` 这行代码；

2. 第97行代码 `IL_0004:ldc.i4.1` 代表的就是true，等待着赋值给下面的licenseExpired；

3. 修改97行的代码为 `IL_0004:ldc.i4.0` ，然后保存il文件。

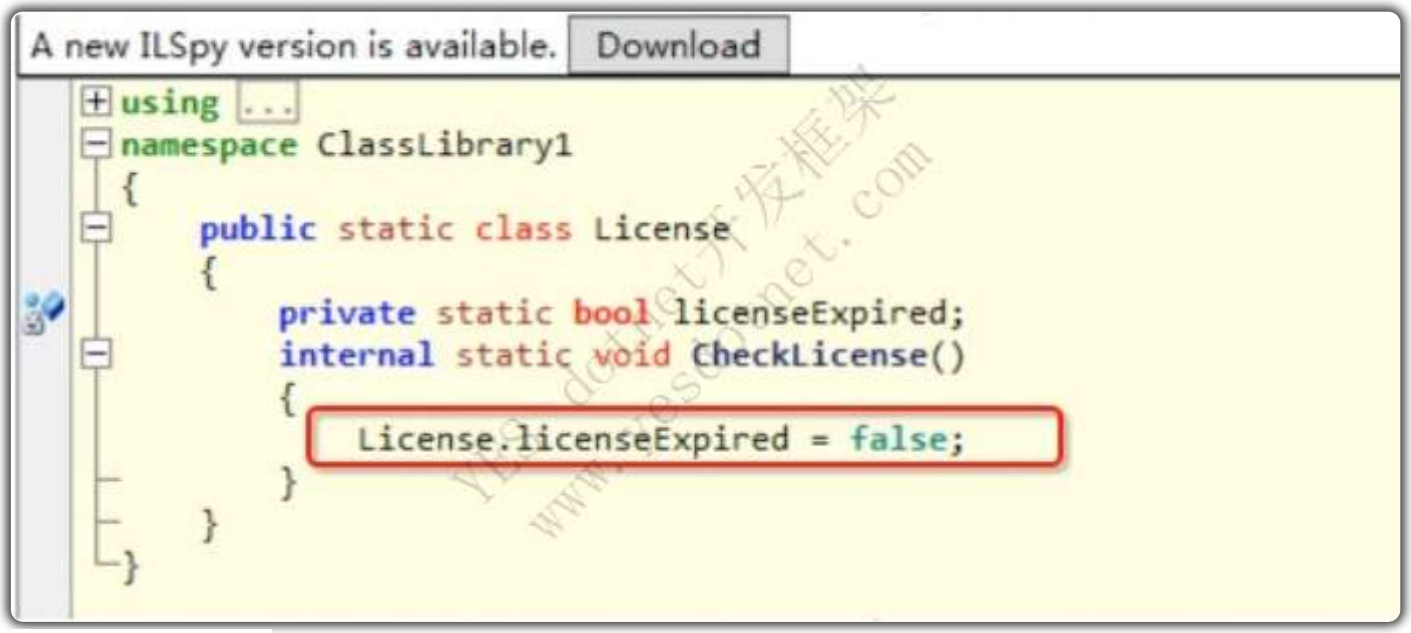
打开命令行，进入到il文件所在到目录，执行下面的命令；

```
c:\windows\microsoft.net\framework\v4.0.30319\ilasm.exe
/dll/resource=ClassLibrary1.res ClassLibrary1.il
```



现在在il文件的目录中可以看到生成的dll文件：

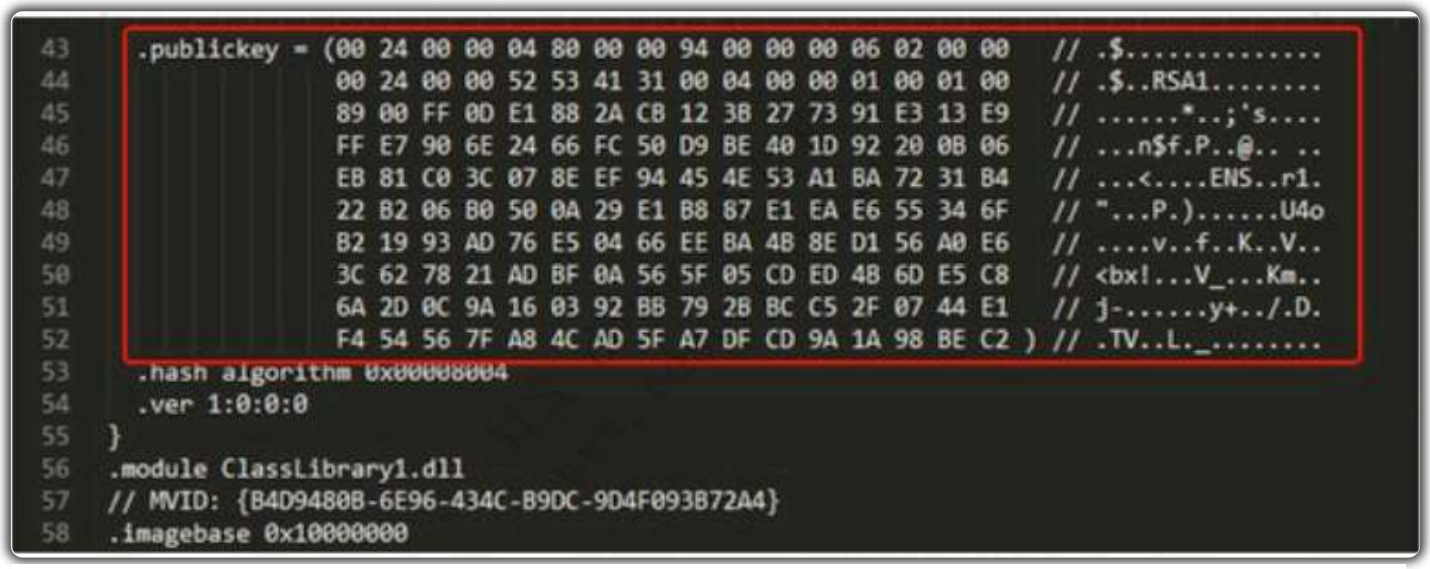
反编译生成的dll文件，可以看出代码已经被修改，如下图：



带签名dll的破解

如果程序集是带签名的程序集，在经过上面的步骤进行破解后，dll文件依然不能 使用，通常会报如下错误：

未能加载文件或程序集"ClassLibrary1"或它的某一个依赖项。未能验证强名称签名。此程序集可能已被篡改，或者已被延迟签名，但没有用正确的私钥进行完全签名。 (异常来自 >HRESULT:0x80131045) 经过对比发现经过签名的dll文件和未签名的dll文件的区别在于签名的dll文件生成的il代码中会多处如下的代码：



将il代码中上面红框部分代码删除，重新生成的dll文件就是去掉了签名的dll文件。不出什么意外的话此时的dll文件可以正常使用了，但有时又会出现如下的错误：

重写成员“xxx”时违反了继承安全-重写方法的安全可访问性必须与所重写方法的安全可访问性匹配

解决该问题需要在Assemblyinfo.cs文件中添加如下代码：

```
[assembly: System.Security.SecurityRules(System.Security.SecurityRuleSet.Level1)]
```

上面添加的代码对应的il代码为：

```
 .custom instance void
[mscorlib]System.Security.SecurityRulesAttribute::.ctor(valuetype
[mscorlib]System.Security.SecurityRuleSet) = ( 01 00 01 00 00 )
```

将上面的代码添加到il的相应位置，重新生成dll文件就OK了。

总结

本文是以技术研究学习为目的，不提倡对收费的软件或类库进行破解使用。

取消打印水印：

修改：FastReport.Print.DefaultPrintController类中的PrintPage方法取消水印代码

```
// Token: 0x060012B3 RID: 4787
public override void PrintPage(object sender, PrintPageEventArgs e)
{
    base.StartPage(e);
    Graphics graphics = e.Graphics;
    graphics.PageUnit = GraphicsUnit.Pixel;
    graphics.TranslateTransform(-e.PageSettings.HardMarginX / 100f * graphics.DpiX, -e.PageSettings.HardMarginY / 100f
    FRPaintEventArgs e2 = new FRPaintEventArgs(graphics, graphics.DpiX / 96f, graphics.DpiY / 96f, base.Report.GraphicC
    base.Page.Print(e2);
    base.Page.Dispose();
    base.FinishPage(e);
    e.HasMorePages = base.HasMorePages();
}

// Token: 0x060012B4 RID: 4788
```

打印页数限制：

修改：FastReport.Print.DefaultPrintController→FastReport.Print.PrintControllerBase类

→FastReport.PageNumbersParser类中的 构造方法：

```
// Token: 0x060012B8 RID: 4792
public PageNumbersParser(Report report, int curPage)
{
    this.FPages = new List<int>();
    int num = report.PreparedPages.Count;
    if (report.PrintSettings.PageRange == PageRange.Current)
    {
        this.FPages.Add(curPage - 1);
    }
    else if (!this.Parse(report.PrintSettings.PageNumbers, num))
    {
        for (int i = 0; i < num; i++)
        {
            this.FPages.Add(i);
        }
    }
    num = report.PreparedPages.Count;
    for (int j = 0; j < this.FPages.Count; j++)
    {
        if (this.FPages[j] >= num || this.FPages[j] < 0)
        {
            this.FPages.RemoveAt(j);
            j--;
        }
    }
    if (report.PrintSettings.PrintPages == PrintPages.Odd)
    {
        int k = 0;
        while (k < this.FPages.Count)
        {
            if (this.FPages[k] % 2 == 0)
```

取消导出水印：

修改：FastReport.Export.ExportBase类中的GetOverlayPage方法,相关代码

```
// FastReport.Export.ExportBase
// Token: 0x0600263D RID: 9789
protected ReportPage GetOverlayPage(ReportPage page)
{
    return page;
}
```

取消导出页数限制：

修改：FastReport.Export.ExportBase类中的Export方法,相关代码

```
public void Export(Report report, Stream stream)
{
    base.SetReport(report);
    this.FStream = stream;
    this.PreparePageNumbers();
    this.GeneratedFiles.Clear();
    this.FExportTickCount = Environment.TickCount;
    if (this.FPages.Count > 0)
    {
        if (!string.IsNullOrEmpty(this.FileName))
        {
            this.GeneratedFiles.Add(this.FileName);
        }
        this.Start();
        report.SetOperation(ReportOperation.Exporting);
        if (this.ShowProgress)
        {
            Config.ReportSettings.OnStartProgress(base.Report);
        }
        else
        {
            base.Report.SetAborted(false);
        }
        try
        {
            for (int i = 0; i < report.PreparedPages.Count; i++)
            {
                if (this.ShowProgress)
                {
                    Config.ReportSettings.OnProgress(base.Report, string.Format(Res.Get("Mes
                      + 1, this.FPages.Count);
                }
                if (base.Report.Aborted)
                {
                    break;
```

23.2.18