XFOCUS TEAM
from internet,for internet

用ring3代码可靠地枚举Windows进程

于　旸

X'CON 2004

枚举进程的一般方法

- Psapi
  - EnumProcesses()
- ToolHelp32
  - Process32First()
  - Process32Next()
- NtQuerySystemInformation()

SystemProcessInformation = 5

à ExpGetProcessInformation()

à 遍历ActiveProcessLinks

à 定位EPROCESS

à 获得进程信息

EPROCESS

- EPROCESS的部分结构：

```
.....
DWORD      UniqueProcessId
LIST_ENTRY  ActiveProcessLinks
.....
Char        ImageFileName[16]
```

- ETHREAD的部分结构：

```
.....
PEPROCESS  ThreadsProcess
.....
```

ActiveProcessLinks

```
kd> da poi(PsInitialSystemProcess) + 1fc
81a2fc5c  "System"
kd> da poi(poi(PsInitialSystemProcess)+a0) -a0 + 1fc
8132af5c  "SMSS.EXE"
kd> da poi(poi(poi(PsInitialSystemProcess)+a0)) -a0 + 1fc
8134af5c  "CSRSS.EXE"
kd> da poi(poi(poi(poi(PsInitialSystemProcess)+a0))) -a0 + 1fc
8119375c  "WINLOGON.EXE"
```

检测隐藏进程的流程

定位所有EPROCESS

↓

取出进程信息

常规方法枚举进程

↓

得到进程信息

↓ ↓

比较，找出隐藏进程

核心问题：定位EPROCESS

隐藏和检测之间的对抗

- Hook NtQuerySystemInformation()
  - Hook SDT
  - SystemProcessInformation=5
    SystemHandleInformation=16
  - 实现：
    - Hacker Defender (holy_father@phreaker.net)

- 直接遍历ActivateProcessLinks枚举进程
  - 可以检测Hook NtQuerySystemInformation()
  - 实现：
    - KProcCheck (chewkeong@security.org.sg)

隐藏和检测之间的对抗

- 从ActiveProcessLinks上摘除自身
  - 实现：
    - FU_Rootkit (fuzen_op@yahoo.com)

- 利用线程调度链表检测隐藏进程
  - KiWaitInListHead、KiWaitOutListhead、KiDispatcherReadyListHead
  - 可以检测Hook NtQuerySystemInformation()
  - 可以检测从ActiveProcessLinks上摘除自身
  - 无法在Windows XP和Windows 2003上实现
  - 实现
    - Klister (joanna@mailsnare.net)
    - KProcCheck (chewkeong@security.org.sg)

隐藏和检测之间的对抗

- 绕过基于内核调度链表的进程检测
  - http://www.xfocus.net/articles/200404/693.html (Kinsephi@hotmail.com)

- Hook SwapContext()

  SwapContext()函数原型：

```
__fastcall SwapContext
(
        PETHREAD SwapIn,
        PETHREAD SwapOut
)
```

Hook SwapContext()

- 支持Windows 2000/XP/2003
- 从环境切换级别上检测，可靠
- 可以检测目前所有已知隐藏进程的方法
- SwapContext()并不是一个引出函数
  - 通用性问题
  - 可能导致系统崩溃
- 实现：
  - http://www.rootkit.com/newsread_print.php?newsid=170 (kkasslin@cc.hut.fi)

对EPROCESS的新探

- Windows NT 5.0、5.1
  - EPROCESS. SessionProcessLinks
    - 不包含System和smss.exe
  - EPROCESS.Vm.WorkingSetExpansionLinks

- Windows NT 5.2
  - EPROCESS. MmProcessLinks

```
kd> dt _EPROCESS ImageFileName poi(MmProcessList)-238
   +0x154 ImageFileName : [16]  "Idle"
kd> dt _EPROCESS ImageFileName poi(poi(MmProcessList))-238
   +0x154 ImageFileName : [16]  "System"
```

找到第一个EPROCESS

- Ntoskrnl.exe导出的PsInitialSystemProcess

```
kd> dt _EPROCESS ImageFileName
   poi(PsInitialSystemProcess)
   +0x1fc ImageFileName : [16]  "System“
```

- NtQuerySystemInformation()
  - SystemHandleInformation = 16
  - SYSTEM_HANDLE_INFORMATION.Object

找到第一个EPROCESS

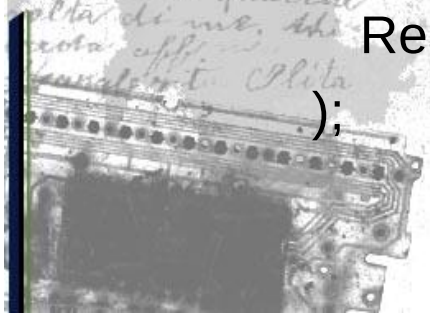
- 利用KPCR

```
kd> dt _KPCR PrcbData.CurrentThread ffdff000
   +0x120 PrcbData                 :
      +0x004 CurrentThread         : 0xff91e740
kd> dt _ETHREAD ThreadsProcess 0xff91e740
   +0x22c ThreadsProcess : 0xff9011c0
kd> dt _EPROCESS ImageFileName 0xff9011c0
   +0x1fc ImageFileName : [16]  "kd.exe"
```

读取内核数据-NT 5.0

- 读取物理内存

```
RtlInitUnicodeString (
   &PhyMemString,
   L\\Device\\PhysicalMemory
);
ZwOpenSection (
   &hPhyMem, SECTION_MAP_READ,
   &PhyMemAttribs
);
MapAddress = MapViewOfFile (
   hPhyMem, FILE_MAP_READ, 0, ReadAddress,
   ReadLength + GetPageSize()
);
```



读取内核数据-NT 5.0

- 线性地址à物理地址

```
PVOID __stdcall LameGetPhysicalAddress( PVOID KernelAddress )
{
    PVOID PhysAddress = 0;

    if((DWORD)KernelAddress < 0x80000000L ||
      (DWORD)KernelAddress >= 0xA0000000L)
        (DWORD)PhysAddress= (DWORD)KernelAddress & 0x0FFFFFFF;
    else
        (DWORD)PhysAddress = (DWORD)KernelAddress & 0x1FFFFFFF;
    return PhysAddress;
}
```

– Flier Lu (flier_AT_nsfocus.com)
《自动验证 Windows NT 系统服务描述表的完整性》

读取内核数据-NT 5.1/5.2

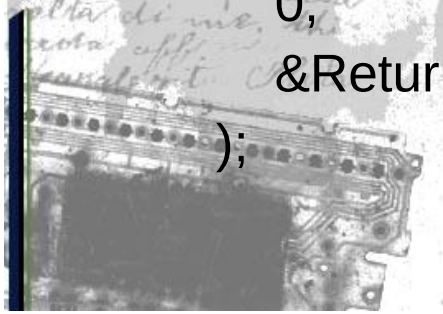
- ZwSystemDebugControl()

```
NTSTATUS ZwSystemDebugControl (
    IN SYSDBG_COMMAND Command,
    IN PVOID InputBuffer,
    IN ULONG InputBufferLength,
    OUT PVOID OutputBuffer,
    IN ULONG OutputBufferLength,
    OUT PULONG ReturnLength
    );
```

读取内核数据-NT 5.1/5.2

```
typedef struct _MEMORY_CHUNKS {
    ULONG Address;
    PVOID Data;
    ULONG Length;
}MEMORY_CHUNKS, *PMEMORY_CHUNKS;
MEMORY_CHUNKS QueryBuff;

ZwSystemDebugControl (
    SysDbgReadKernelMemory,
    &QueryBuff,
    sizeof(MEMORY_CHUNKS),
    NULL,
    0,
    &ReturnLength
);
```



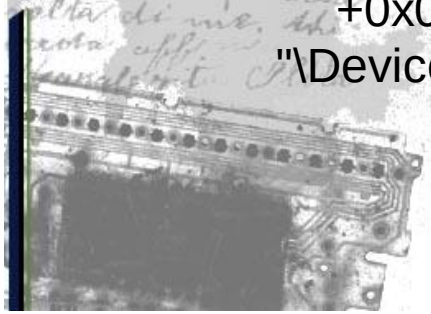
获取进程映像全路径

- Windows NT 5.0

```
kd> dt _EPROCESS pImageFileName
    poi(poi(PsInitialSystemProcess)+a0)-a0
   +0x284 pImageFileName : 0x81363fb8  "\WINNT\system32\SMSS.EXE“
```

- Windows NT 5.1/5.2

```
lkd> dt _eprocess SeAuditProcessCreationInfo.ImageFileName->Name
    0ff894218
nt!_EPROCESS
   +0x1d4 SeAuditProcessCreationInfo                :
      +0x000 ImageFileName                          :
         +0x000 Name                                : _UNICODE_STRING
"\Device\HarddiskVolume1\WINDOWS\system32\cmd.exe"
```



处理进程残像

- 进程结束后，操作系统并不总能把相应的EPROCESS从链表上摘掉
- EPROCESS其它成员已经破坏
  - 判断合法性
- MmProcessLinks会枚举出完整的EPROCESS残像

参考资源

[1]Hacker Defender Holy_Father(holy_father_AT_phreaker.net)
- http://rootkit.host.sk/

[2]KprocCheck Tan Chew Keong(chewkeong_AT_security.org.sg)
- http://www.security.org.sg/code/kproccheck.html

[3]FU_Rootkit fuzen_op(fuzen_op_AT_yahoo.com)
- https://www.rootkit.com/vault/fuzen_op/FU_Rootkit.zip

[4]Klister Joanna Rutkowska(joanna_AT_mailsnare.net)
- http://www.rootkit.com/vault/joanna/klister-0.4.zip

参考资源

[5]绕过内核调度链表进程检测
SoBeIt(Kinsephi_AT_hotmail.com)

- http://www.xfocus.net/articles/200404/693.html

[6]Detecting Hidden Processes by Hooking the SwapContext Function kkasslin(kkasslin_AT_cc.hut.fi)

- http://www.rootkit.com/newsread_print.php?newsid=170

[7]Playing with Windows /dev/(k)mem crazylord (razylord_AT_thins.net)

- http://www.phrack.org/phrack/59/p59-0x10.txt

参考资源

[8]自动验证 Windows NT 系统服务描述表的完整性 Flier Lu (flier_AT_nsfocus.com)

- http://www.nsfocus.net/index.php?act=magazine&do=view&mid=2119

[9]对Native API NtSystemDebugControl的分析 tombkeeper (tombkeeper_AT_xfocus.org)

- http://www.xfocus.net/articles/200408/721.html

[10]获取Windows 系统的内核变量 tombkeeper (tombkeeper_AT_xfocus.org)

- http://www.xfocus.net/articles/200408/724.html