

ProjectSauron 中持久化模块分析



2016-08-23/ by nEINEI

Agenda

- 1 ProjectSauron背景介绍
- 2 持久化思路
- 3 各种技术细节
- 4 思考？如何长时间隐藏

ProjectSauron背景介绍

- ProjectSauron从**2011年6月就已经开始**，到2016年仍然在活跃。ProjectSauron入侵受害者网络所使用的最初感染途径仍然未知。
- ProjectSauron幕后的网络罪犯是经验丰富的传统攻击者，并且花费了大量精力学习其他高级攻击者的技术，包括Duqu、Flame、Equation和Regin：吸收了这些攻击中最具创新的技术，并且改善了其攻击策略，以确保自身不被发现。

ProjectSauron背景介绍

- **卡巴斯基发现窃取政府通讯信息的顶级网络间谍平台 ProjectSauron (又名Remsec)**
- <http://news.kaspersky.com.cn/news2016/08n/160808.htm>
- **赛门铁克, Backdoor.Remsec indicators of compromise**
- http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/Symantec_Remsec_IOCs.pdf

ProjectSauron背景介绍

- 其中有一个并不十分引人注意的细节：
- BUS manager stage 1 : bootstrap library
- MD5:2A8785BF45F4F03C10CD929BB0685C2D52736
- DLL 32 9.0 2009.07.14 01:07:24
- 该模块可能从2009年就已经开始工作，但直到目前才被发现。

Agenda

- 1 ProjectSauron背景介绍
- 2 持久化思路
- 3 各种技术细节
- 4 思考？如何长时间隐藏

Project Sauron持久化模块

- Loader :This description is based on the analysis of the file with the MD5 hash of 2a8785bf45f4f03c10cd929bb0685c2d which was seen with the file name of MSAOSSPC.DLL. As mentioned, the loader is responsible for loading files from disk and executing them
- Persistence : The loader is implemented as a (fake) Security Support Provider. Implementing the export of **InitSecurityInterfaceW** effectively functions as a loadpoint for the module.

2 持久化思路,为什么能隐藏这么久的时间?

- 1 TimeDateStamp 00000108 4 4A5BDA4C 周二 7月 14 09:07:24 2009
- 2 ITW 时间 :
 - First seen ITW 2009-07-14 01:15:44 (这个时间是否真实呢?)
 - First submission 2016-04-06 14:28:07
 - Last submission 2016-08-16 14:58:29
- 3 编译器:
 - Microsoft Visual C++ ver. 6/7 DLL Linker 9 - MS Visual Studio 2008
- 4 SSPI服务提供模块
- 5 Description AOL Client for 32 bit platforms
- 6涉及到的算法, ADLER32,RC5/RC6 / ZLIB
-

一次确定命名的时间

K7GW	-	9.221.19209	20160404
Kaspersky	HEUR:Trojan.Multi.R emsec.gen	15.0.1.13	20160406
Kingsoft	-	2013.8.14.323	20160406
Malwarebytes	-	2.1.1.1115	20160406
McAfee	-	6.0.6.653	20160406
McAfee-GW-Edition	-	v2015	20160406
Microsoft	-	1.1.12603.0	20160406
MicroWorld-eScan	-	12.0.250.0	20160406
NANO-Antivirus	-	1.0.18.7201	20160406

McAfee VSE启发式命名的时间

McAfee	RDN/Generic.dx	6.0.6.653	20160804
McAfee-GW-Edition	Artemis!Trojan	v2015	20160804

2 持久化思路,为什么能隐藏这么久的时间?

- 0000BCC1 4800D6C1 0 AcquireCredentialsHandleW
- 0000BCA7 4800D6A7 0 AcquireCredentialsHandleA
- 0000BC91 4800D691 0 AcceptSecurityContext
- 0000BCDB 4800D6DB 0 ApplyControlToken
- 0000BE7A 4800D87A 0 VerifySignature
- 0000BE64 4800D864 0 RevertSecurityContext
- 0000BE4A 4800D84A 0 QuerySecurityPackageInfoW
- 0000BE30 4800D830 0 QuerySecurityPackageInfoA
- 0000BE18 4800D818 0 QueryContextAttributesW
- 0000BE00 4800D800 0 QueryContextAttributesA
- 0000BDF2 4800D7F2 0 MakeSignature
- 0000BDD7 4800D7D7 0 **InitializeSecurityContextW**
- 0000BD8C 4800D78C 0 InitializeSecurityContextA
- 0000BDA5 4800D7A5 0 InitSecurityInterfaceW
- 0000BD8E 4800D78E 0 InitSecurityInterfaceA
- 0000BD73 4800D773 0 ImpersonateSecurityContext
- 0000BD5D 4800D75D 0 FreeCredentialsHandle
- 0000BD4B 4800D74B 0 FreeContextBuffer
- 0000BD30 4800D730 0 EnumerateSecurityPackagesW
- 0000BD15 4800D715 0 EnumerateSecurityPackagesA
- 0000BCFF 4800D6FF 0 DeleteSecurityContext
- 0000BCED 4800D6ED 0 CompleteAuthToken

PE
导出
的
函数

这些函数属于哪类调用呢？

Security Support Provider Interface:

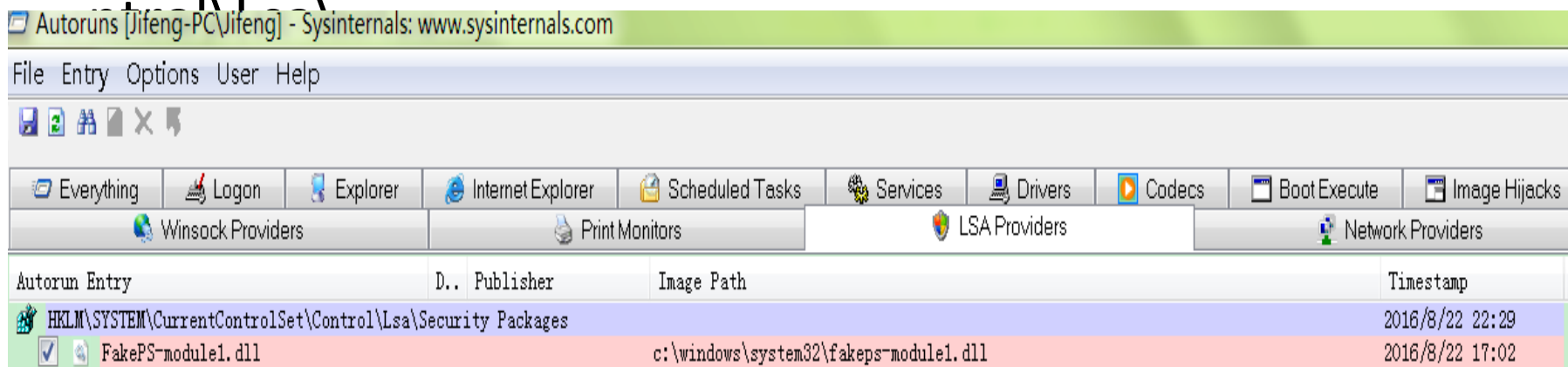
(SSPI) A common interface between transport-level applications, such as Microsoft Remote Procedure Call (RPC), and security providers, such as Windows Distributed Security. SSPI allows a transport application to call one of several security providers to obtain an authenticated connection. These calls do not require extensive knowledge of the security protocol's details.

相关的详细解释

- <https://technet.microsoft.com/en-us/library/bb742535.aspx>
- DllHandle = (void *)LoadLibrary(TEXT("security.dll"));
- INIT_SECURITY_INTERFACE InitSecurityInterface;
- PSecurityFunctionTable SecurityInterface = 0;
- SecPkgInfo PAPI * SecurityPackages;
- DWORD NumOfPkgs;
- SECURITY_PROVIDER_INFO PAPI * List;
- InitSecurityInterface = GetProcAddress(DllHandle, SECURITY_ENDPOINT);

如何自动加载支持SSPI的DLL

- <https://github.com/gentilkiwi/mimikatz/blob/bb371c2acba397b4006a6cddc0f9ce2b5958017b/mimilib/kssp.c>
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Co



Autoruns [Jifeng-PC\Jifeng] - Sysinternals: www.sysinternals.com

File Entry Options User Help

Everything Logon Explorer Internet Explorer Scheduled Tasks Services Drivers Codecs Boot Execute Image Hijacks

Winsock Providers Print Monitors LSA Providers Network Providers

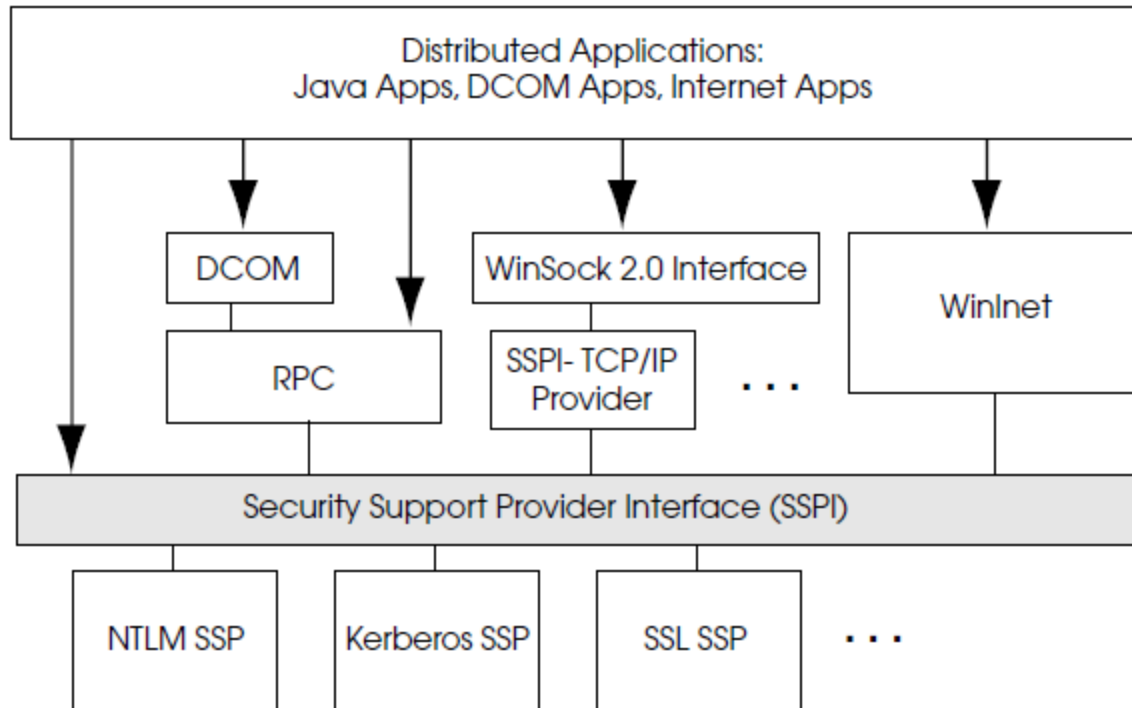
Autorun Entry	D..	Publisher	Image Path	Timestamp
HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages				2016/8/22 22:29
<input checked="" type="checkbox"/> FakePS-module1.dll			c:\windows\system32\fakeps-module1.dll	2016/8/22 17:02

- tspkg
- Pku2u ; FakePS-module1.dll

Agenda

- 1 ProjectSauron背景介绍
- 2 持久化思路
- 3 各种技术细节
- 4 思考？如何长时间隐藏

SSPI 架构模型



SSPI函数的调用关系

- 编程的话，导入sspi.h (secure32.dll -> security.dll -> sspicli.dll (这里是真正的实现，就是把内部的一个表给导出出来))
- 一般包含InitSecurityInterface 的模块在系统中：
- c:\Windows\System32\credssp.dll 22,528 2014/10/29 09:27 -
a--
- c:\Windows\System32\dbgeng.dll 4,417,536 2015/04/02
06:20 -a--
- c:\Windows\System32\dbnetlib.dll 130,560 2014/10/29
10:42 -a--
- c:\Windows\System32\dnsapi.dll 657,920 2014/11/05 09:44 -
a—
- ...

伪造了一份SSPI的导出函数

Unexplored ■ Instruction ■ External symbol

Name	Address	Ordinal
AcceptSecurityContext	480015D4	1
AcquireCredentialsHandleA	48001913	2
AcquireCredentialsHandleW	48001913	3
ApplyControlToken	480017C5	4
CompleteAuthToken	480017DC	5
DeleteSecurityContext	4800175D	6
DllEntryPoint	48009935	[main entry]
EnumerateSecurityPackagesA	4800194F	7
EnumerateSecurityPackagesW	48001AAA	8
FreeContextBuffer	480017CD	9
FreeCredentialsHandle	480013F2	10
ImpersonateSecurityContext	480017E1	11
InitSecurityInterfaceA	4800190D	12
InitSecurityInterfaceW	48001A5C	13
InitializeSecurityContextA	4800192B	14
InitializeSecurityContextW	4800192B	15
MakeSignature	48001858	16
QueryContextAttributesA	480019E8	17
QueryContextAttributesW	48001B5A	18
QuerySecurityPackageInfoA	4800196E	19
QuerySecurityPackageInfoW	48001AC9	20
RevertSecurityContext	480017E1	21
VerifySignature	480018A8	22

PE 导出模块

- 这些导出模块仅有1个函数具有执行多很多函数的代码的功能，其他的函数都是内部运行完毕直接返回。
- SECURITY_STATUS __stdcall
CompleteAuthToken(PCtxtHandle phContext,
PSecBufferDesc pToken)
- {
- return 0;
- }

PE 导出模块

- 这些导出模块仅有1个函数具有执行多很多函数的代码的功能，其他的函数都是内部运行完毕直接返回。
- SECURITY_STATUS __stdcall
AcquireCredentialsHandleW(LPWSTR pszPrincipal, LPWSTR pszPackage, unsigned __int32 fCredentialUse, void *pvLogonId, void *pAuthData, SEC_GET_KEY_FN pGetKeyFn, void *pvGetKeyArgument, PCredHandle phCredential, PTimeStamp ptsExpiry)
- {
- return A_atomic_3(ptsExpiry, fCredentialUse, (int)pAuthData, (int)phCredential);
- }

资源节上看

- BLOCK "StringFileInfo"
 - {
 - BLOCK "040904b0"
 - {
 - VALUE "CompanyName", "Microsoft Corporation"
 - VALUE "FileDescription", "AOL Client for 32 bit platforms"
 - VALUE "FileVersion", "6.00.7755"
 - VALUE "InternalName", "MSAOSSPC"
 - VALUE "LegalCopyright", "Copyright © 1995-1996 Microsoft Corporation"
 - VALUE "OriginalFilename", "MSAOSSPC.DLL"
 - VALUE "ProductName", "Microsoft® Internet Services"
 - VALUE "ProductVersion", "6.00.7755"

.rdata段看到的一段信息

- 代码里面可以看到的一段数据信息，
- ```
.rdata:4800B14C 41 4F 4C 00 Str2 db 'AOL',0 ; DATA XREF:
EnumerateSecurityPackagesA+4o
.rdata:4800B14C ; QuerySecurityPackageInfoAo
.rdata:4800B150 41 4F 4C 20 53 65+aAolSecurityPac db 'AOL Security Package',0
.rdata:4800B150 63 75 72 69 74 79+ ; DATA XREF:
QuerySecurityPackageInfoA+65o
.rdata:4800B165 00 00 00 align 4
.rdata:4800B168 44 65 66 61 75 6C+aDefaultUser db 'Default user',0 ; DATA XREF:
QueryContextAttributesA+63o
.rdata:4800B175 00 00 00 align 4
.rdata:4800B178 ; WCHAR pszPackageName
.rdata:4800B178 pszPackageName: ; DATA XREF:
EnumerateSecurityPackagesW+4o
.rdata:4800B178 ; QuerySecurityPackageInfoW+1o
.rdata:4800B178 41 00 4F 00 4C 00+ unicode 0, <AOL>,0
.rdata:4800B180 aAolSecurityP_0: ; DATA XREF:
QuerySecurityPackageInfoW+6Co
```

# AOL 是什么缩写?

- [https://en.wikipedia.org/wiki/AOL\\_Mail](https://en.wikipedia.org/wiki/AOL_Mail)
- 美国在线（AOL Inc.，前身為：American Online，AOL），著名的互联网服务提供者，現為電信商威訊的子公司。
- 其它的AOL不是很具有关联性，看起来最大的可能是美国在线的产品的AOL缩写

# 有些神秘的函数

- ```
int __fastcall sub_48002677(int a1, _DWORD *a2)
{
    int result; // eax@5

    if ( (unsigned int)"@SUVWATAUAVAWH该 >= (unsigned int)nullsub_2
        || (unsigned int)((char *)nullsub_2 - (char *)"@SUVWATAUAVAWH该) > 0x800 )
    {
        result = 0;
    }
    else
    {
        if ( a2 )
            *a2 = (char *)nullsub_2 - (char *)"@SUVWATAUAVAWH该;
        result = (int)"@SUVWATAUAVAWH该;
    }
    return result;
}
```

[微软的解释](#)

<https://social.technet.microsoft.com/Forums/en-US/9c9a0898-f335-4d42-ab5f-5aa972400be0/strange-repeating-strings-in-multiple-files-possible-virus?forum=w7itprosecurity>

<https://forums.malwarebytes.org/topic/157201-trojan-or-virus-doing-weird-things-to-files/>

涉及到算法相关的代码片段

- 一个代码片段,
- ```
switch (v79 & 3)
{
 case 1u:
 *(_DWORD *)(v2 + 76) = &unk_4800BD00;
 *(_DWORD *)(v2 + 84) = 9;
 *(_DWORD *)(v2 + 80) = &unk_4800C500;
 *(_DWORD *)(v2 + 88) = 5;
 *(_DWORD *)v2 = 18;
 break;
 case 2u:
 *(_DWORD *)v2 = 15;
 break;
 case 3u:
 *(_DWORD *)(v1 + 24) = "invalid block type";
 *(_DWORD *)v2 = 27;
 break;
```

[搜索到了一份相关的一份源代码:](http://dlib.net/dlib/external/zlib/inflate.c.html)

<http://dlib.net/dlib/external/zlib/inflate.c.html>

# 相关的一份源代码:

- case 1:                    /\* fixed block \*/
- fixedtables(state);        Tracev((stderr, "inflate:  fixed codes block%s\n",
- state->last ? " (last)" : ""));
- state->mode = LEN\_;        /\* decode codes \*/
- if (flush == Z\_TREES) {
- DROPBITS(2);            goto inf\_leave;        }
- break;
- case 2:   /\* dynamic block \*/
- Tracev((stderr, "inflate:  dynamic codes block%s\n",
- state->last ? " (last)" : ""));
- state->mode = TABLE;       break;
- case 3:
- strm->msg = (char \*)"invalid block type";
- state->mode = BAD;

# 所有API函数名称都加密

- ```
void A_write_log_to_dir()
{
    const char *v0; // eax@2
    CHAR ExistingFileName; // [sp+0h] [bp-210h]@2
    char Dest; // [sp+104h] [bp-10Ch]@2
    char v3; // [sp+207h] [bp-9h]@2
    char Dst; // [sp+208h] [bp-8h]@2

    if ( dword_4800E208 )
    {
        A_Unknown_Maybe_get_filename_call(&ExistingFileName);
        v0 = (const char *)A_Unknown_Maybe_get_filename((int)"彝熵 娠珺tE€.", &Dst, 6, 1);
        sprintf(&Dest, 0x103u, v0, &ExistingFileName, dword_4800E218);
        v3 = 0;
        if ( A_write_file_some_call1(&Dest) || GetLastError() == 2 )
        {
            if ( MoveFileA(&ExistingFileName, &Dest) )
            {
                if ( ++dword_4800E218 > dword_4800E208 )
                    dword_4800E218 = 1;
            }
        }
    }
}
```

所有API函数名称都加密

- `.rdata:4800C8B0 8B 96 8D A8 90 8D+aLcnirnpMmRtJn db ' 嬬嶧恹瘡寤穢悞馳嗒' ; DATA XREF: sub_48001C5C+270`
`.rdata:4800C8C2 FF db 0FFh`
`.rdata:4800C8C3 FF db 0FFh`
`.rdata:4800C8C4 9E 9A 8D BC 9A AD+aUNNLLrtNcliu db ' 濃崑鑿臻臻悞殛棲淚' ; DATA XREF: sub_48001DB4+9E0`
`.rdata:4800C8D6 FF db 0FFh`
`.rdata:4800C8D7 FF db 0FFh`
`.rdata:4800C8D8 8D BC 8B B1 9A 8B+aNLLuNcliu db '崑 嬬臻濃殛棲淚' ; DATA XREF: sub_48001DB4+14A0`
`.rdata:4800C8E6 FF db 0FFh`
`.rdata:4800C8E7 FF db 0FFh`
`.rdata:4800C8E8 8D BC 8B B1 9A 8B+aNLLuNclzlu db '崑 嬬臻濃殛棲齧淚' ; DATA XREF: sub_48001DB4+1380`

调用前在内存中解密

- ```
A_decode_buffer_by_API_Name(
 &ProcName,
 (_DWORD*)"娑嶧恹瘳寤殍悞馳哇濃崑鑿臻臻悞殍
 棲悞崑嫫臻濃殍棲悞崑嫫臻濃殍棲嚙悞kernel32");
v8 = (unsigned int)GetProcAddress(v6, &ProcName);
if (!v8)
 return 0;
if (!dword_4800E25C)
{
 if (A_find_sepcial_char((unsigned int)v6, v8)
 || (A_decode_buffer_by_API_Name(&v11,
 &unk_4800C954), sub_48003C15(&v11)))
```

# 解密的函数

- ```
_BYTE * __usercall A_decode_buffer_by_API_Name@<eax>(_BYTE *result_buff@<eax>,
_DWORD *buff@<edx>){
    signed int v2; // esi@1
    int v3; // ecx@2
    unsigned int v4; // ecx@2
    v2 = 0;
    do {
        v3 = ~*buff;
        ++buff;
        v4 = _byteswap_ulong(v3);
        do {
            if ( (_BYTE)v4 )
                *result_buff++ = v4;
            else
                v2 = 1;
            v4 >>= 8;
        }
        while ( v4 ); }
    while ( !v2 );
    *result_buff = 0;
    return result_buff;
}
```

古老的int2E调用重新出现

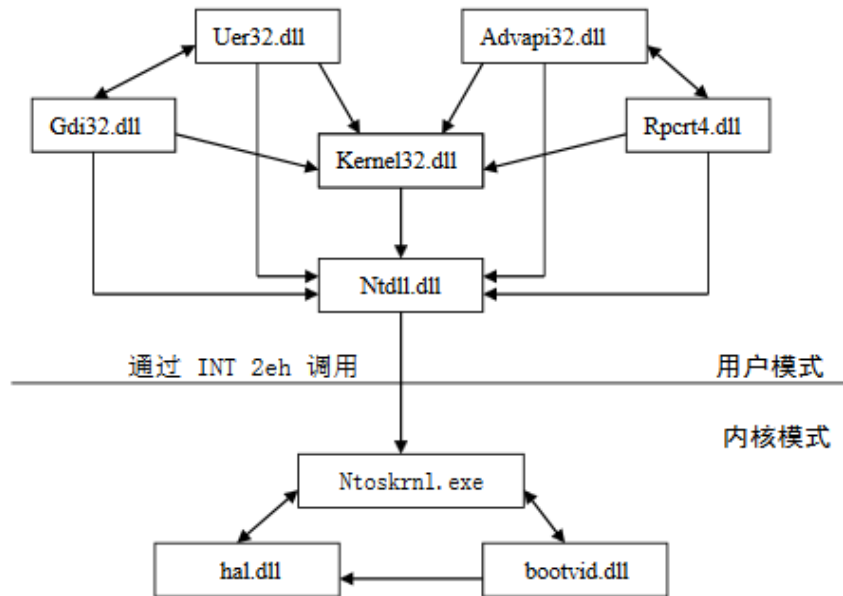
```
.text:48003E19
.text:48003E19 loc_48003E19:                                ; CODE XREF: A_Fake_NtCall_x1+20↑j
.text:48003E19      push    [ebp+arg_20]
.text:48003E1C      push    [ebp+arg_1C]
.text:48003E1F      push    [ebp+arg_18]
.text:48003E22      push    0
.text:48003E24      push    1
.text:48003E26      push    [ebp+arg_10]
.text:48003E29      push    [ebp+arg_8]
.text:48003E2C      push    [ebp+arg_4]
.text:48003E2F      push    [ebp+arg_0]
.text:48003E32      push    1FFFFFFh
.text:48003E37      push    esi
.text:48003E38      call   A_int2E
.text:48003E3D      leave
.text:48003E3E      retn
.text:48003E3E A_Fake_NtCall_x1 endp

.text:4800987C A_int2E      proc near                                ; CODE XREF: A_Fake_NtCall_x1+A1↑p
.text:4800987C      arg_0      = byte ptr 4
.text:4800987C      mov     eax, dword_4800E018
.text:48009881      lea    edx, [esp+arg_0]
.text:48009885      int    2Eh                                ; DOS 2+ internal - EXECUTE COMMAND
.text:48009885                                ; DS:SI -> counted CR-terminated command string
.text:48009887      retn    2Ch
.text:48009887 A_int2E      endp
.text:48009887
```

利用int2E目的是什么呢？

- 1 想一下最早程序编译的时间出现在2009年。
- 2 WIN7操作系统出现的时间是2009年。
- 3 当时，绝大多数系统都是WINXP的系统为主，win7为辅，还有一部分是win2000。
- 而在win2000之后，winxp以后系统的系统调用取消了int2e，改为SYSENTER指令。

Win2000时的调用体系



(图一)

00783DA8	↵ 88 20000000	mov eax,20	
00783DAD	. 8D5424 04	lea edx,dword ptr ss:[esp+4]	
00783DB1	. CD 2E	int 2E	→ 由此处进入内核
00783DB3	. C2 2C00	ret 2C	
00783DB6	↵ 57	push edi	
00783DB7	. 8B7C24 0C	mov edi,dword ptr ss:[esp+C]	
00783DBB	. 8B5424 08	mov edx,dword ptr ss:[esp+8]	
00783DBF	. C702 00000000	mov dword ptr ds:[edx],0	
00783DC5	. 897A 04	mov dword ptr ds:[edx+4],edi	
00783DC8	. 0BFF	or edi,edi	

利用int2E目的是什么呢？

- 1 说明当时被攻击目标系统包含win2000的机器。
- 2 =》得出一个推论，这个模块的编译时间是2009年的可能性非常大。

```
v7 = (unsigned __int8)GetVersion();
v8 = v7 < 6;
if ( v7 >= 6 )
    v9 = A_Fake_NtCall_x1_x(0, a1, 0, a2, a3, a4, a5, a6, (int)&v17);
else
    v9 = A_Gen_syscall_somevalue_x(a1, 0, a2, a3, a4, a5, a6, a7, (int)&v17, (int)&v14);
v10 = v9;
if ( !v9 )
    return 0;
if ( v8 && v15 && v16 && !A_Get_Ntdll_API_x5_something((int)&v14, a1, v9)
    || !(a7 & 1)
    && !(a7 & 4) ? (v12 = A_Get_Ntdll_API_x6_something(v10)) : (v12 = A_Use_Csrss_call_API(a1, (HANDLE)v10, (int)&v17)),
```

防止被HIPS监控

- 不走正常的API 调用流程
 - 伪造一个FakeNtCall例程

 - 例如，
 - (ring3)
 - CreateProcess->CreateProcessW->**ntCreateProcess/zwCreateProcess** ->SSDT Dispatcher (kernel)
- \\
hooked by HIPS

一种伪造的Ntcall方式

- LONG __declspec(naked) NtCall(DWORD FunctionIndex,DWORD ClassIndex,...)
- {
- __asm
- {
- push ebp
- mov ebp,esp
- mov eax,FunctionIndex
- mov ecx,ClassIndex
- lea edx,[ebp+0x10]
- call fs:[0xC0] // - > KiFastSystemCall
- add esp,0x4
- leave
- retn
- }
- }

另外一个方式是，绕过被NT保护的开始5个字节，防止被HIPS hook，自己填充一个调用的入口参数，直接调用。

一种伪造的NTcall方式

- ```
if (A_Get_Kernel32_API())
```
- ```
{
```
- ```
 v7 = GetModuleHandleA("ntdll");
```
- ```
  if ( !v7 )
```
- ```
 return 0;
```
- ```
  A_decode_buffer_by_API_Name(&ProcName, *(void **)((char *)&off_4800C9F8 + v1));
```
- ```
 v8 = GetProcAddress(v7, &ProcName);
```
- ```
  if ( !v8 || *(_BYTE *)v8 != -72 )
```
- ```
 return 0;
```
- ```
  v5 = *(_DWORD *)((char *)v8 + 1);
```
- ```
}
```
- ```
else
```
- ```
{
```
- ```
  A_decode_buffer_by_API_Name(&ProcName, *(void **)((char *)&off_4800C9BC + v1));
```
- ```
 v6 = A_magic_code_CALL2((int)&ProcName);
```
- ```
  if ( !v6 || *(_DWORD *)v6 != 0xB8D18B4C ) // 针对x64平台
```
- ```
 return 0;
```
- ```
  v5 = *(_DWORD *)(v6 + 4);
```
- ```
}
```
- ```
if ( v5 )
```
- ```
 goto LABEL_9;
```
- 
- [https://src.chromium.org/svn/releases/4.0.223.9/src/sandbox/wow\\_helper/service64\\_resolver.cc](https://src.chromium.org/svn/releases/4.0.223.9/src/sandbox/wow_helper/service64_resolver.cc)

# 被GetProcAddress导出的函数

```
PSecurityFunctionTableW __stdcall InitSecurityInterfaceW()
{
 struct _SECURITY_FUNCTION_TABLE_W *v0; // esi@1
 struct _OSVERSIONINFOW VersionInformation; // [sp+4h] [bp-114h]@1

 v0 = 0;
 VersionInformation.dwOSVersionInfoSize = 276;
 if (GetVersionExW(&VersionInformation))
 {
 if (VersionInformation.dwPlatformId == 2)
 {
 if (A_AA_MainDll())
 return (PSecurityFunctionTableW)&SSPI_FuncTableList;
 }
 else if (VersionInformation.dwPlatformId == 1)
 {
 return (PSecurityFunctionTableW)&SSPI_FuncTableList;
 }
 }
 return v0;
}
```

# 被GetProcAddress导出的函数

- 由A\_AA\_MainDll 执行读取特定文件，执行，写入特定目录log，加密等操作。
- 具体见iDB文件.

# 为什么能隐藏这么长时间？

- 可否构造出一个类似的样本？

```
__declspec(dllexport)int InitializeSecurityContextW()
{
 int v0 = 0;
 _OSVERSIONINFO VersionInformation;
 VersionInformation.dwOSVersionInfoSize = 276;
 if (GetVersionExW(&VersionInformation))
 {
 if (VersionInformation.dwPlatformId == 2)
 {
 if (A_AA_MainDll())
 return (int)&g_Fake_SSPI[0];
 }
 else if (VersionInformation.dwPlatformId == 1)
 {
 return (int)&g_Fake_SSPI[0];
 }
 }
 return v0;
 return (int)&g_Fake_SSPI[0];
}

__declspec(dllexport)int VerifySignature()
{
 return g_Fake_Global[1];
}

__declspec(dllexport)int AcquireCredentialsHandleW()
{
 return g_Fake_Global[4];
}

__declspec(dllexport)int AcquireCredentialsHandleA()
```

| Ordinal      | Function RVA | Name Ordinal | Name RVA | Name                              |
|--------------|--------------|--------------|----------|-----------------------------------|
| (nFunctions) | Dword        | Word         | Dword    | szAnsi                            |
| 00000001     | 000012C0     | 0000         | 000023A7 | ?AcceptSecurityContext@@YAH...    |
| 00000002     | 000012B0     | 0001         | 000023C5 | ?AcquireCredentialsHandleA@...    |
| 00000003     | 000012A0     | 0002         | 000023E7 | ?AcquireCredentialsHandleW@...    |
| 00000004     | 000012A0     | 0003         | 00002409 | ?ApplyControlToken@@YAHXZ         |
| 00000005     | 00001320     | 0004         | 00002423 | ?CompleteAuthToken@@YAHXZ         |
| 00000006     | 00001340     | 0005         | 0000243D | ?DeleteSecurityContext@@YAH...    |
| 00000007     | 00001350     | 0006         | 0000245B | ?EnumerateSecurityPackagesA...    |
| 00000008     | 00001340     | 0007         | 0000247E | ?EnumerateSecurityPackagesW...    |
| 00000009     | 00001330     | 0008         | 000024A1 | ?FreeContextBuffer@@YAHXZ         |
| 0000000A     | 00001320     | 0009         | 000024BB | ?FreeCredentialsHandle@@YA...     |
| 0000000B     | 00001310     | 000A         | 000024D9 | ?ImpersonateSecurityContext@...   |
| 0000000C     | 00001300     | 000B         | 000024FC | ?InitSecurityInterfaceA@@YAHXZ    |
| 0000000D     | 000012F0     | 000C         | 0000251B | ?InitSecurityInterfaceW@@YAH...   |
| 0000000E     | 000012F0     | 000D         | 0000253A | ?InitializeSecurityContextA@@Y... |
| 0000000F     | 00001220     | 000E         | 0000255D | ?InitializeSecurityContextW@@...  |
| 00000010     | 000012F0     | 000F         | 00002580 | ?MakeSignature@@YAHXZ             |
| 00000011     | 000012E0     | 0010         | 00002596 | ?QueryContextAttributesA@@Y...    |
| 00000012     | 00001290     | 0011         | 000025B6 | ?QueryContextAttributesW@@Y...    |

# 没能触发太多的报警



SHA256: 60c4f332cc122e4bab9c4fa3114cf9e2e4e3dfedbaa0d072d77a3aee696eb60a

File name: MSAOSS.dll

Detection ratio: 1 / 55

Analysis date: 2016-08-22 07:59:44 UTC ( 18 hours, 37 minutes ago )



Analysis

File detail

Additional information

Comments 0

Votes

| Antivirus     | Result                 | Update   |
|---------------|------------------------|----------|
| CAT-QuickHeal | (Suspicious) - DNAScan | 20160822 |
| ALYac         | ✓                      | 20160822 |
| AVG           | ✓                      | 20160822 |

Activate Wind  
Go to PC settings to

# 为什么能隐藏这么长时间？

卡巴 & 赛门2份报告的提到了要加载A00000002.dll，卡巴提到该Dll是由services.exe来激活。

The screenshot shows the x32dbg debugger interface. The main window displays assembly code for the module ms.dll at address 480011B9. The code includes instructions like `push eax`, `call ms.48004905`, `add esp,14`, `test eax,eax`, `je ms.4800124B`, `push ebx`, `call ms.48001091`, and `mov ebx,eax`. The EIP register is highlighted at 480011BF. The right-hand pane shows the register values: EAX: 00000000, EBX: 7FFDB000, ECX: 001DEC2C, EDX: 00000003. A message box is visible with the text: "L'failed to open module c:\\System volume Information\\\_restore{ED650925-A32C-4E9C-8...". Below the assembly view, there are memory dump windows. The 'Dump 1' window shows a memory dump starting at address 001DEC2C, with the ASCII column containing the string "Failed to open module c:\\System volume Information\\\_restore{ED650925-A32C-4E9C-8...". The 'Dump 5' window shows a memory dump starting at address 001DF5A8, with the ASCII column containing the string "user32.76726573".

# 为什么能隐藏这么长时间

在模拟中我们发现需要父进程是services来模拟加载，否则不能触发行为。但是services.exe是伪造的还是系统本身的呢？如果是系统本身的则需要格外的代码来支持DLL运行，这是一个疑点。但2份报告都没有给出答案。

|         |              |      |                    |                                                                                              |                    |                         |
|---------|--------------|------|--------------------|----------------------------------------------------------------------------------------------|--------------------|-------------------------|
| 7:43... | services.exe | 1464 | CreateFile         | C:\Users\win7x86\Desktop\tg\MFC140ENU.DLL                                                    | NAME NOT FOUND     | Desired Access: R...    |
| 7:43... | services.exe | 1464 | CreateFile         | C:\Users\win7x86\Desktop\tg\MFC140ENU.DLL.DLL                                                | NAME NOT FOUND     | Desired Access: R...    |
| 7:43... | services.exe | 1464 | CreateFile         | C:\Users\win7x86\Desktop\tg\MFC140ENU.DLL                                                    | NAME NOT FOUND     | Desired Access: R...    |
| 7:43... | services.exe | 1464 | CreateFile         | C:\Users\win7x86\Desktop\tg\MFC140ENU.DLL.DLL                                                | NAME NOT FOUND     | Desired Access: R...    |
| 7:43... | services.exe | 1464 | CreateFile         | C:\Users\win7x86\Desktop\tg\MFC140ENU.DLL                                                    | NAME NOT FOUND     | Desired Access: R...    |
| 7:43... | services.exe | 1464 | CreateFile         | C:\Users\win7x86\Desktop\tg\MFC140ENU.DLL.DLL                                                | NAME NOT FOUND     | Desired Access: R...    |
| 7:43... | services.exe | 1464 | CreateFile         | C:\Users\win7x86\Desktop\tg\MFC140ENU.DLL                                                    | NAME NOT FOUND     | Desired Access: R...    |
| 7:43... | services.exe | 1464 | CreateFile         | C:\Users\win7x86\Desktop\tg\MFC140ENU.DLL.DLL                                                | NAME NOT FOUND     | Desired Access: R...    |
| 7:43... | services.exe | 1464 | QueryNameInfo...   | C:\Users\win7x86\Desktop\tg\services.exe                                                     | SUCCESS            | Name: \(\$Recycle.B...  |
| 7:44... | services.exe | 1464 | CreateFile         | C:\Users\win7x86\Desktop\tg\ms.dll                                                           | SUCCESS            | Desired Access: R...    |
| 7:44... | services.exe | 1464 | QueryBasicInfor... | C:\Users\win7x86\Desktop\tg\ms.dll                                                           | SUCCESS            | CreationTime: 8/23/...  |
| 7:44... | services.exe | 1464 | CloseFile          | C:\Users\win7x86\Desktop\tg\ms.dll                                                           | SUCCESS            |                         |
| 7:44... | services.exe | 1464 | CreateFile         | C:\Users\win7x86\Desktop\tg\ms.dll                                                           | SUCCESS            | Desired Access: R...    |
| 7:44... | services.exe | 1464 | CreateFileMap...   | C:\Users\win7x86\Desktop\tg\ms.dll                                                           | FILE LOCKED WIT... | SyncType: SyncTy...     |
| 7:44... | services.exe | 1464 | CreateFileMap...   | C:\Users\win7x86\Desktop\tg\ms.dll                                                           | SUCCESS            | SyncType: SyncTy...     |
| 7:44... | services.exe | 1464 | Load Image         | C:\Users\win7x86\Desktop\tg\ms.dll                                                           | SUCCESS            | Image Base: 0x480...    |
| 7:44... | services.exe | 1464 | CloseFile          | C:\Users\win7x86\Desktop\tg\ms.dll                                                           | SUCCESS            |                         |
| 7:44... | services.exe | 1464 | ReadFile           | C:\Users\win7x86\Desktop\tg\ms.dll                                                           | SUCCESS            | Offset: 38,400, Leng... |
| 7:44... | services.exe | 1464 | ReadFile           | C:\Users\win7x86\Desktop\tg\ms.dll                                                           | SUCCESS            | Offset: 9,216, Lengt... |
| 7:44... | services.exe | 1464 | CreateFile         | C:\System Volume Information\_restore{ED650925-A32C-4E9C-8A73-8E6F0509309A}\RP0\A0000002.dll | PATH NOT FOUND     | Desired Access: G...    |
| 7:44... | services.exe | 1464 | CreateFile         | C:\Windows\System32\en-US\KernelBase.dll.mui                                                 | SUCCESS            | Desired Access: G...    |
| 7:44... | services.exe | 1464 | CreateFileMap...   | C:\Windows\System32\en-US\KernelBase.dll.mui                                                 | FILE LOCKED WIT... | SyncType: SyncTy...     |
| 7:44... | services.exe | 1464 | QueryStandardl...  | C:\Windows\System32\en-US\KernelBase.dll.mui                                                 | SUCCESS            | AllocationSize: 786...  |
| 7:44... | services.exe | 1464 | CreateFileMap...   | C:\Windows\System32\en-US\KernelBase.dll.mui                                                 | SUCCESS            | SyncType: SyncTy...     |

# 4 思考？如何长时间隐藏

- 1 需要参数，条件的才能触发恶意程序的调用路径
- 2 不需要加壳等额外引发终端安全报警的功能
- 3 全程加密字符串，函数名，说明等
- 4 分散功能设计
- 5 终端启发式检测，在分析粒度上将越来越难于检测