Windows_Kernel_Shellcode_Exploit

chrO.ot

Nanika

Summary

- What is Exploit
- Windows_Shellcode
- Windows_Kernel_shellcode
- CreateRemoteThread backdoor “byshell”
- Exploit Demo
- 0day_exploit

What is Exploit

- Buffer overflow
- Heap overflow
- Format string
- CGI or SQL inject

Buffer overflow

```
void func(void)
{
char buffer[256]; // *
for(i=0;i<512;i++)
buffer[i]='A'; // !
 return;
 }
```

STACK

Local Variables

ESP

Buffer

EBP-> Old Value of EBP

Return Address

Heap overflow

```
buf1 = HeapAlloc(hHeap, 0, 32);

strcpy(buf1,mybuf);

buf2 = HeapAlloc(hHeap, 0, 32);

HeapFree(hHeap, 0, buf1);
HeapFree(hHeap, 0, buf2);
```

- mov [ecx], eax
- mov [eax+4], ecx

Format string

```
int main(int argc, char
    *argv[])
    {
    char buffer[512]="";
    strncpy(buffer,argv[
    1],500);
        printf(buffer);
        return 0;
    }
```

```
printf
%.123456x%.123456x%n
```

CGI or SQL inject

- ../../../../../../
- ‘!@#$%^&*()
- http://127.0.0.1/a.asp?s=xxxxx

Windows_Shellcode

- bind shellcode
- connect back shellcode

bind shellcode

- 編碼
- Kernel 基址定位
- API函數address定位
- Create Shell
- WinSocket
- 產生TCP協議參數為押入堆疊6,1,2
- Bind;綁定一個Port來做通訊
- Listen;等對方連結
- Accept;連結成功
- Send;發送
- Recv;接收

connect back shellcode

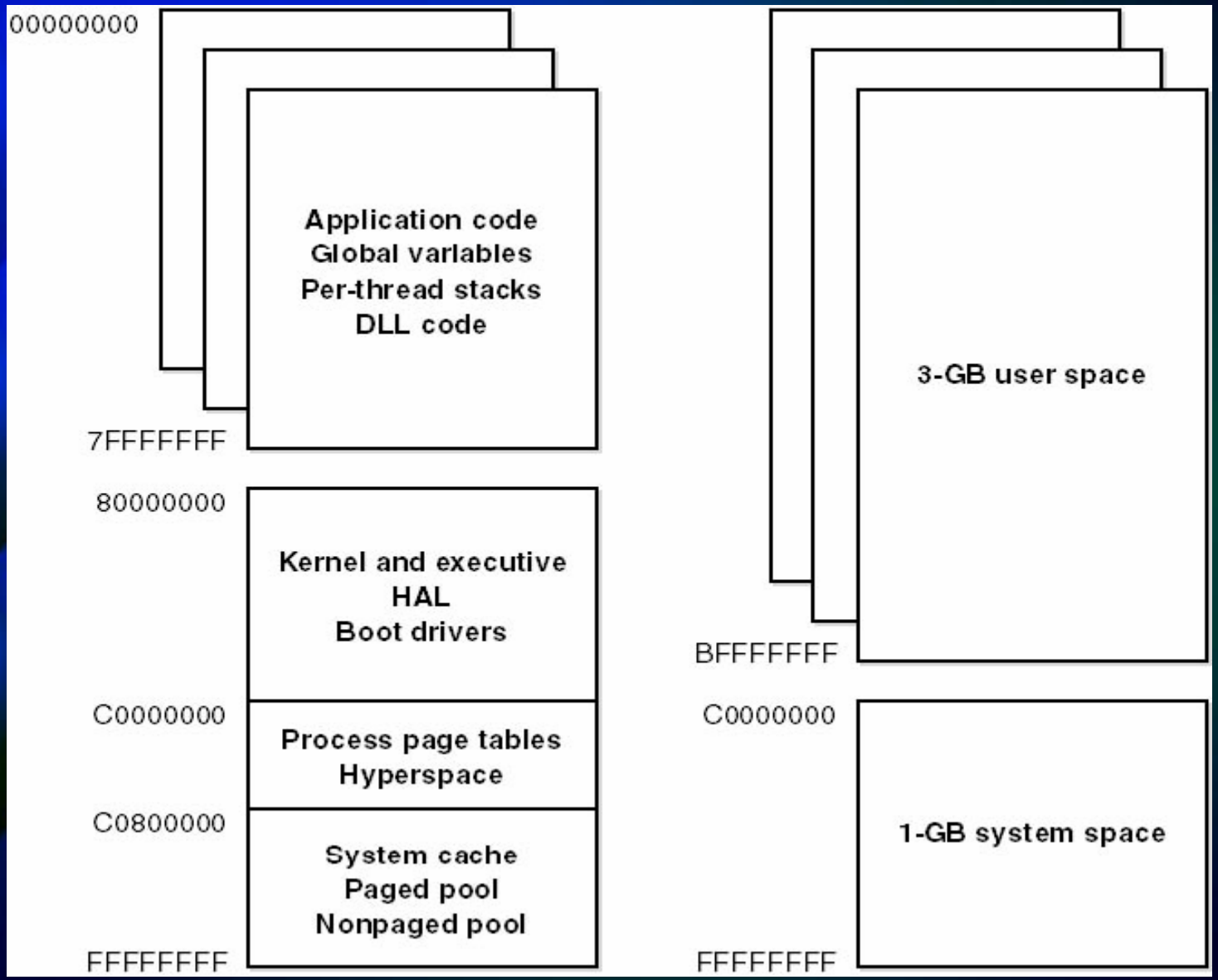
- db "WSAStartup",0;edi-14h
- db "socket",0;edi-10h
- db "connect",0;edi-ch
- db "send",0;edi-8
- db "recv",0;edi-4

Windows_Kernel_shellcode

- patch EPROCESS Token
- APC inject

- Kernel Memery
- Kernel struct
- IRQL
- Patch EPROCESS Token
- APC_Inject

Kernel Memery



Kernel struct

dt nt!_EPROCESS

dt nt!_KPROCESS

dt nt!_ETHREAD

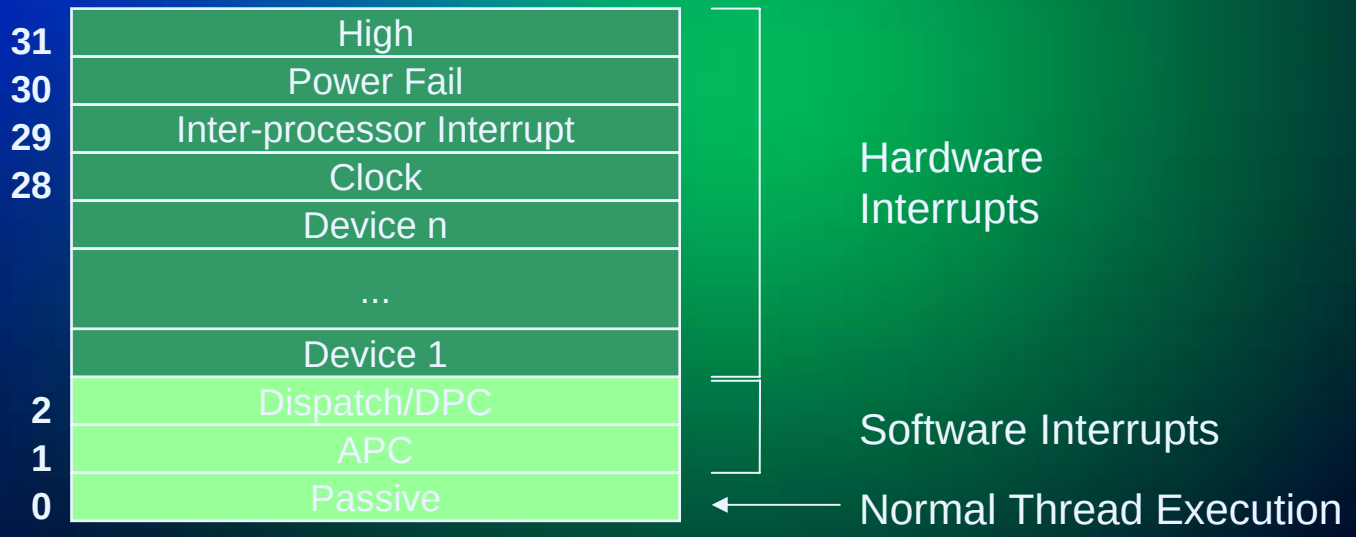
dt nt!_KTHREAD

dt nt!_HANDLE_TABLE

- EPROCESS
- KPROCESS
- ETHREAD
- KTHREAD
- HANDLE TABLE

IRQL

IRQL = Interrupt Request Level (0 to 31)



Patch EPROCESS Token

```
+0x000 Pcb              : _KPROCESS
+0x084 UniqueProcessId  : Ptr32 Void
+0x088 ActiveProcessLinks : _LIST_ENTRY
+0x0c4 ObjectTable      : Ptr32 _HANDLE_TABLE
+0x0c8 Token            : _EX_FAST_REF
+0x190 ThreadListHead   : _LIST_ENTRY
```

APC_Inject

- Kernel ffdf0000 = user 7ffe0000
- lkd> dt nt!_KTHREAD
- +0x02d State : UChar

- lkd> dt nt!_KAPC_STATE
- +0x000 ApcListHead : [2] _LIST_ENTRY
- +0x010 Process : Ptr32 _KPROCESS
- +0x014 KernelApcInProgress : UChar
- +0x015 KernelApcPending : UChar
- +0x016 UserApcPending : UChar

void KeInitializeApc(struct _KAPC *Apc, PKTHREAD thread, unsigned char state_index, PKKERNEL_ROUTINE ker_routine, PKRUNDOWN_ROUTINE rd_routine, PKNORMAL_ROUTINE nor_routine, unsigned char mode, void *context);

void KeInsertQueueApc(struct _KAPC *APC, void *SysArg1, void *SysArg2, unsigned char arg4);

CreateRemoteThread backdoor

- backdoor byshell
- monitor thread&process

backdoor byshell

- **HANDLE OpenProcess(DWORD *dwDesiredAccess*, BOOL *bInheritHandle*, DWORD *dwProcessId*);**
- **LPVOID VirtualAllocEx(HANDLE *hProcess*, LPVOID *lpAddress*, SIZE_T *dwSize*, DWORD *flAllocationType*, DWORD *flProtect*);**
- **BOOL WriteProcessMemory(HANDLE *hProcess*, LPVOID *lpBaseAddress*, LPCVOID *lpBuffer*, SIZE_T *nSize*, SIZE_T* *lpNumberOfBytesWritten*);**
- **HANDLE CreateRemoteThread(HANDLE *hProcess*, LPSECURITY_ATTRIBUTES *lpThreadAttributes*, SIZE_T *dwStackSize*, LPTHREAD_START_ROUTINE *lpStartAddress*, LPVOID *lpParameter*, DWORD *dwCreationFlags*, LPDWORD *lpThreadId*);**

monitor thread&process

- PsSetCreateProcessNotifyRoutine(ProcessCreateMon, FALSE)
- PsLookupProcessByProcessId((ULONG)PId, &EProcess);
- PsSetCreateThreadNotifyRoutine(ThreadCreateMon);
- PsLookupThreadByThreadId((PVOID)4, &Thread);

Windows Kernel shellcode Exploit

- Ring3->Ring0
- Exploit Apc inject

Ring3->Ring0

- OpenPhysicalMemory()
- MapPhysicalMemory( )

Exploit Apc inject Demo

Exploit Demo

- Microsoft Jet Database Engine DB File Buffer Overflow Exploit
- Microsoft Exchange Server Remote Code Execution Exploit (MS05-021)
- Microsoft Internet Explorer "javaprxy.dll" Command Execution Exploit

Q&A

Reference

- **Remote Windows Kernel Exploitation - Step Into the Ring 0 (pdf)**
  http://www.eeye.com/~data/publish/whitepapers/research/OT20050205.FILE.pdf
- ring3->ring0 code by http://zzzevazzz.blogchina.com/427939.html
- monitor thread&process
  http://www.xfocus.net/articles/200503/788.html
- Inside Microsoft Windows 2000
- Byshell backdoor
  http://www.xfocus.net/tools/200412/943.html

  MSDN

謝謝各位