XFOCUS TEAM
from internet,for internet

Windows interrupt context kernel overflow exploits

FlashSky@xfocus.org
FangXing@venustech.com.cn

X'CON 2004

- **Special Thanks**

  KeJi(KeJi@venustech.com.cn)'s great help on Windows kernel

- **Thanks**

  Thanks ALERT7 and or Xfocus members

  Thanks all guys in Venustech AD-LAB **J**

First view of Windows kernel exploiting

- What we are talking about:
  - This is a kernel driver overflow, not via int2e.
  - Exploiting the local/remote kernel driver overflow
  - This occurs in the interrupt context not in the process context which not uses DeviceIoControl.

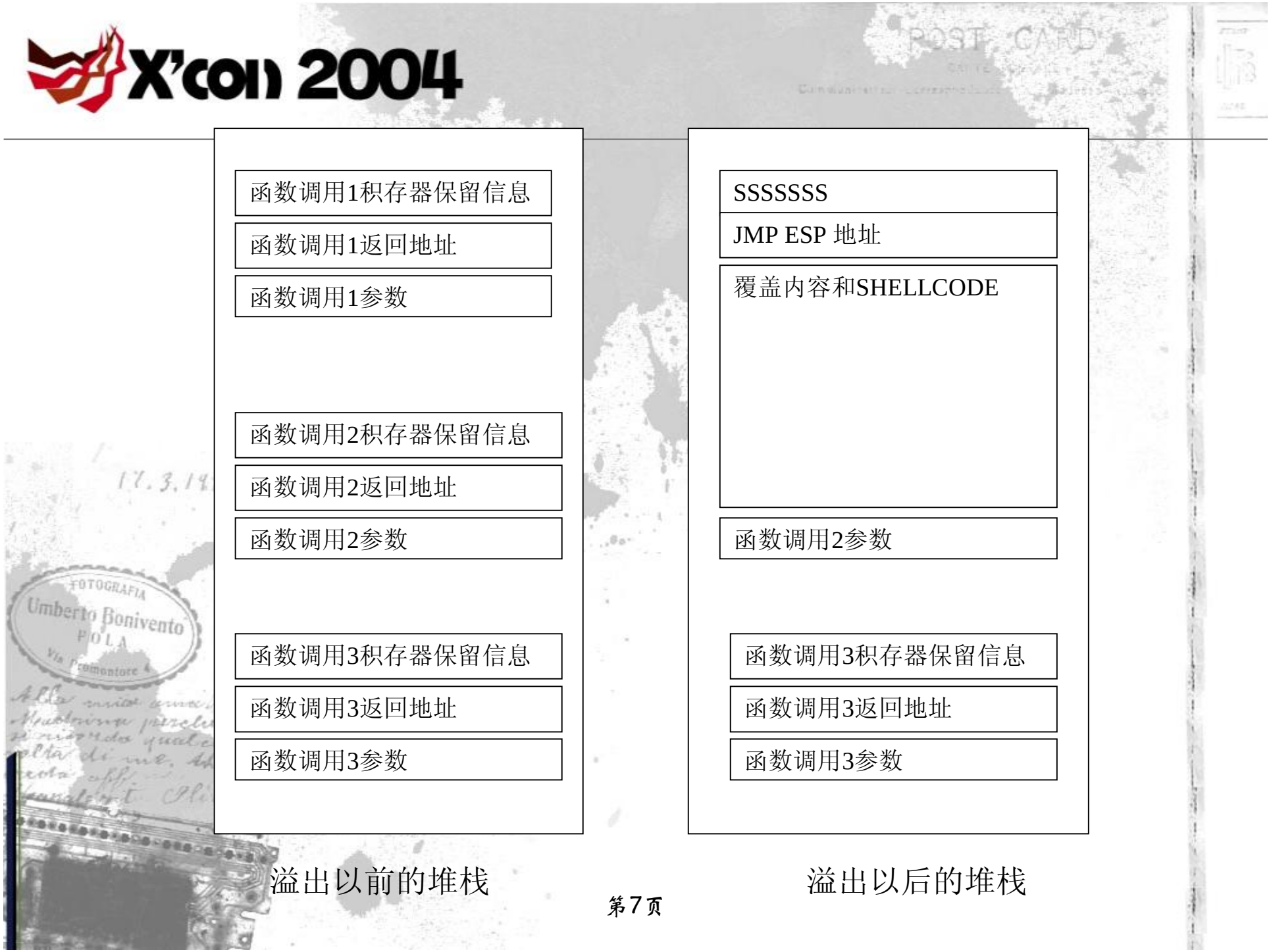
- Related tech
  - Using security-return/unsecurity-return.
  - Some undocument structions and information in Windows kernel.
  - Special drivers such as pid 0.

- **Features of Windows kernel driver**
  - ü Running in pid 0 or 8 process which not map user space(KPEB of pid 0 is special)
  - ü Pid 0 or 8 process is used for process schedule and DPC schedule
  - ü Running in DPC level interrupt disabled and IRQL above 2 will not infect page mapping
  - ü Kernel space stack features
    - ü Usable data space
    - ü Can't use some return value
    - ü ESP ADD instruction in Shellcode
  - ü Special features of kernel stack operations
  - ü Length of kernel stack

- **Features of WINDOWS kernel driver exploiting**
  - ü Shellcode should be in the user space
    - ü Write kernel shellcode need more instructions
    - ü Need handling more details such as page mapping in kernel space
    - ü We can't locate and use kernel functions
    - ü Need handling thread and DPC schedule by ourself
  - ü Overflow occurs in the DPC interrupt level so it need user space shellcode execute without DPC, Thread locked or crash

Kernel safe return method

- **Principle of safe return in kernel space**
  - In the un-overflowed stack area we can find enough date make our process safe return
  - Overflow occurs in function1 and destroyed the stack of function1 and 2 but we can return to function3 when the shellcode finished running .



溢出以前的堆栈

溢出以后的堆栈

- **Kernel buffer overflow under LINUX**
  - ü XFOCUS member Alert7 gave a presentation on Linux kernel overflow at XCON 2002
    - ü Modify the Dispatch List in interrupt context the make specific service/interrupt point to our code, and save the old entry address
    - ü Safe return in overflowed process
    - ü Enter our hooked code and copy the shellcode to this process space then change the uid and restore the entry address.
    - ü Return to user space and run the shellcode.

- **How can we do this? Porting Linux method?**
  - **ü The problems of porting Linux method to Windows**
    - **ü** The related structures in Windows kernel is undocument
    - **ü** Need kernel operation twice
    - **ü** Need extra HOOK code (Can't put directly on stack).
  - **ü Our method notices**
    - **ü** Using multiple LDT switching method to copy our Shellcode directly to the specific process just one time.
    - **ü** Manipulate the specific process and run our shellcode actively.

- **What need consider of the safe return**
  - ü Obtain the KPEB/KTEB of all processes and threads
  - ü Can switch to specific process
  - ü Pages status of SHELLCODE
  - ü Can control EIP of specific process/thread

- **Limitations of safe return**
  - ü Sometimes can't safe return

The Shellcode for Safe return in Kernel

- **Kernel Shellcode**

- **User Shellcode**

Unsafe return 1

- **Why can't safe return**
  - ü Stack space is so limited can't store all information for safe return
  - ü Need correct return value in some structures
  - ü Can't be used on all system
  - ü Safe return may 'cause system locked
  - ü Can't store register

- **Unsafe return?**
  - ü Can we just use iret return to user space?
- The problems we need to face
  - ü Can't schedule and fork, signal operation may lead system locked
  - ü User space code shouldn't trigger DPC operations, such as write file operations may lead system locked
  - ü User space DLL not mapped can't use usual API
  - ü If the pointer in KTEB not correct will lead kernel crash

- **Our plan**
  - ü Can't use the Int2e cause we can't load user space DLL.
  - ü Using not network related, not DPC schedule APIs create a .bat file under the RUN folder.
  - ü Quickly reboot the system and our .bat file will be run.
  - ü Download a .exe file and execute it.
- **Limitations**
  - ü Need reboot
  - ü The shellcode functions is so limited and use lots of hard code
  - ü Can't load DLL APIs directly

X'con 2004

The Shellcode for Unsafe return method 1

- **Kernel Shellcode**

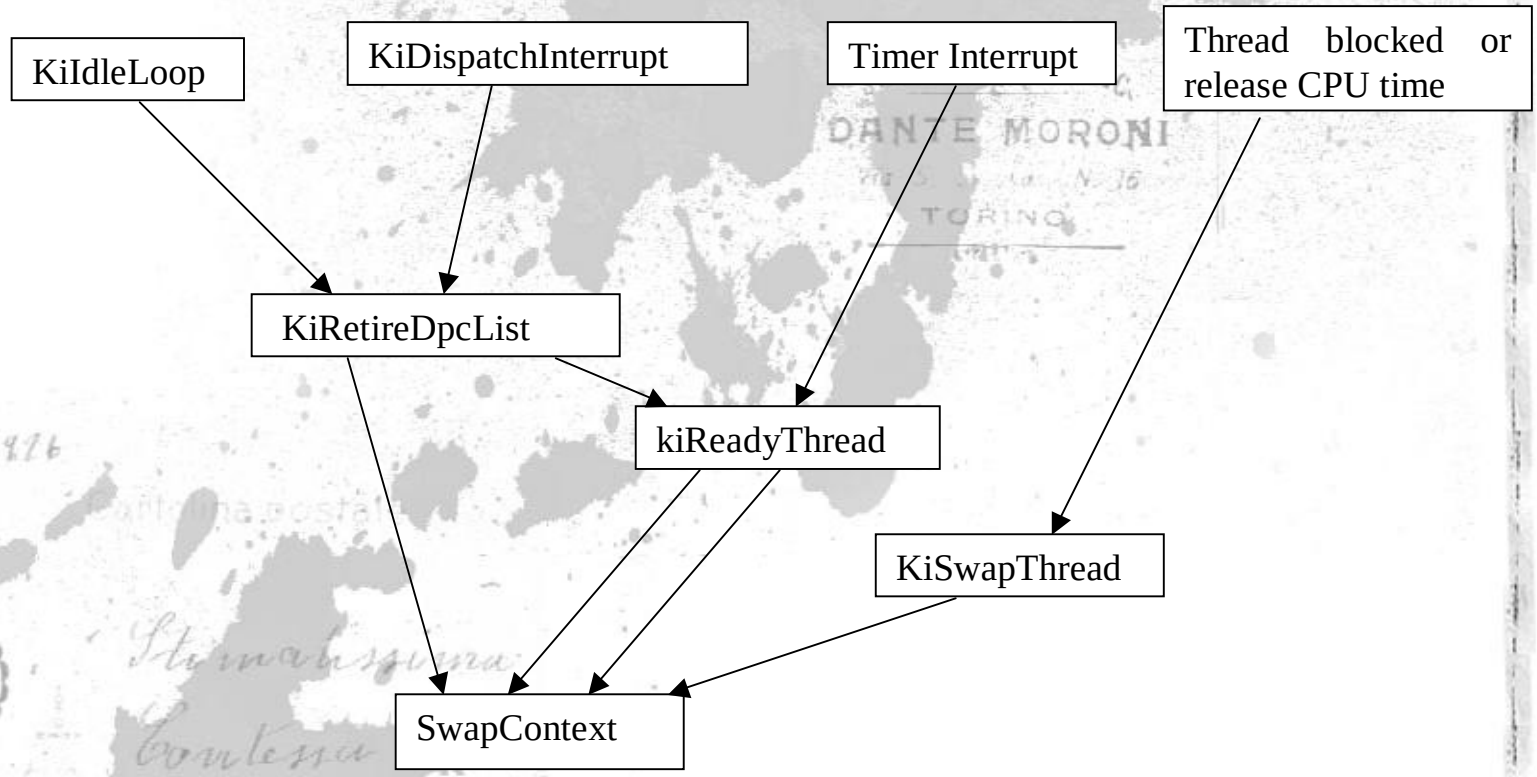
- **User Shellcode**

Unsafe return method 2

- **Focus：Restore the schedule of Thread and DPC**
  - ü ‘Cause limitations of method 1 we can’ write functional Shellcode even write the register.
  - ü Method 1 too complex
  - ü We need more about the Windows DPC and Thread schedule

- **Schedule progress of Thread and DPC**
  - ü Hardware interrupt and APC
  - ü DPC and timer interrupt
  - ü Thread schedule
    - ü Thread blocked and release CPU
    - ü Time slice
    - ü Schedule of IDLE process

- **Key functions of Thread and DPC schedule**
  - ü Function KiIdleLoop
  - ü Function KiDispatchInterrupt
  - ü Function KiRetireDpcList
  - ü Function KeReadythread
  - ü Function KiSwapthread
  - ü Function SwapContext
  - ü TIMER scheduler



调用大致序列

- **Thread and DPC schedule related structures**
  - ü Thread schedule related structures
  - ü DPCLISTHEAD structures of DPC request
  - ü Structures DPC request
  - ü Segment FS and schedule related structures
  - ü KTEB and schedule related
  - ü PID 0 restore function
  - ü Thread kernel stack mapping list

- **Unsafe return method 2 plan**
  - ü Overflowed in KiRetireDpcList
  - ü Restore the status before KiRetireDpcList and restore the DPC schedule.

- **Preparations for unsafe return method 2**
  - ü Select a proper user space Thread to schedule to
  - ü Use new switched process to return
  - ü Page fault handle
  - ü Restore the last interrupt when reenter the DPC and Thread schedule

- **Problems and limitations of unsafe return method 2**
  - ü If the data too long and overflowed the context saved on the stack, the unsafe return method 2 will fail
  - ü Avoid system locked but driver not valid

The Shellcode for Unsafe return method 2

- **Kernel Shellcode**

- **User Shellcode**

- **Differences between method 1 and 2**

More in-depth topic

- **Shellcode running in the kernel real mode**
  - ü Mapping between real and virtual mode
  - ü Need to know NTFS deep
  - ü Can't use DOS interrupts in Windows
  - ü Safe return is better and functional
  - ü Using unsafe return method 2 when stack and schedule related structures are not destroyed.

DEMO

- **SYMANTEC DNS Kernel Overflow**
  - ü **About this vuln**
  - ü **Features of this kernel driver**
  - ü **Vuln details and exploiting**
    - ü **Vuln details**
    - ü **Limited of exploiting**
    - ü **Other usable features**

- safe return in Windows XP
- safe return in Windows 2000
- unsafe return method 1 in Windows 2000
- unsafe return method 2 in Windows 2000

Q/A

Thanks !