# Setjmp，longjump分析

## Setjmp

拷贝所有寄存器的值到第一个参数里面

69722EE0 mov edx,dword ptr [esp+4] 第一个参数地址

69722EE4 mov dword ptr [edx],ebp 拷贝ebp

69722EE6 mov dword ptr [edx+4],ebx

69722EE9 mov dword ptr [edx+8],edi

69722EEC mov dword ptr [edx+0Ch],esi

69722EEF mov dword ptr [edx+10h],esp

69722EF2 mov eax,dword ptr [esp] 栈顶内容（好像就是要跳出的eip）

69722EF5 mov dword ptr [edx+14h],eax

69722EF8 mov dword ptr [edx+20h],56433230h ？？可能是所谓的检查吧

69722EFF mov dword ptr [edx+24h],0

69722F06 mov eax,dword ptr fs:[00000000h] 异常？？

69722F0C mov dword ptr [edx+18h],eax

69722F0F cmp eax,0FFFFFFFFh

69722F12 jne \_\_setjmp3+3Dh (69722F1Dh) ？不知道是做什么，但第一次的确跳到\_s3\_get\_count了

69722F14 mov dword ptr [edx+1Ch],0FFFFFFFFh

69722F1B jmp \_\_setjmp3+78h (69722F58h)

\_s3\_get\_count:

69722F1D mov ecx,dword ptr [esp+8]

69722F21 or ecx,ecx

69722F23 je \_\_setjmp3+4Fh (69722F2Fh) 不知道做什么，但第一次的确跳\_s3\_default\_trylevel了

69722F25 mov eax,dword ptr [esp+0Ch]

69722F29 mov dword ptr [edx+24h],eax

69722F2C dec ecx

69722F2D jne \_\_setjmp3+57h (69722F37h)

\_s3\_default\_trylevel:

69722F2F mov eax,dword ptr [eax+0Ch]

69722F32 mov dword ptr [edx+1Ch],eax

69722F35 jmp \_\_setjmp3+78h (69722F58h) 不知道做什么，但第一次的确跳到结尾\_s3\_done了

\_s3\_save\_trylevel:

69722F37 mov eax,dword ptr [esp+10h]

69722F3B mov dword ptr [edx+1Ch],eax

69722F3E dec ecx

69722F3F je \_\_setjmp3+78h (69722F58h)

69722F41 push esi

69722F42 push edi

69722F43 lea esi,[esp+1Ch]

69722F47 lea edi,[edx+28h]

69722F4A cmp ecx,6

69722F4D jbe \_\_setjmp3+74h (69722F54h)

69722F4F mov ecx,6

\_s3\_save\_data:

69722F54 rep movs dword ptr es:[edi],dword ptr [esi]

69722F56 pop edi

69722F57 pop esi

\_s3\_done:

69722F58 sub eax,eax

69722F5A ret

## Longjmp

DECLSPEC\_GUARD\_SUPPRESS

void

longjmp (

\_In\_ jmp\_buf JumpBuffer,

\_In\_ int ReturnValue

)

/\*++

Routine Description:

This function validates a jump buffer and performs a long jump to the

context specified by the jump buffer.

Arguments:

JumpBuffer - Supplies the address of a jump buffer.

ReturnValue - Supplies the value that is to be returned to the caller of

set jump.

Return Value:

None (however the return value is loaded).

--\*/

{

69725860 push ebp

69725861 mov ebp,esp

//

// Validate the contents of the jump buffer and execute long jump.

//

#if defined(NTOS\_KERNEL\_RUNTIME) && defined(\_AMD64\_)

\_JUMP\_BUFFER \*jmpBuf;

jmpBuf = (\_JUMP\_BUFFER \*)JumpBuffer;

KeCheckStackAndTargetAddress(jmpBuf->Rip, jmpBuf->Rsp);

#else

\_\_except\_validate\_jump\_buffer(JumpBuffer);

69725863 mov eax,dword ptr [JumpBuffer]

69725866 push eax

69725867 call \_\_except\_validate\_jump\_buffer (697257E0h)+0

6972586C add esp,4

#endif

\_\_longjmp\_internal(JumpBuffer, ReturnValue);这是关键

6972586F mov ecx,dword ptr [ReturnValue]

69725872 push ecx

69725873 mov edx,dword ptr [JumpBuffer]

69725876 push edx

69725877 call \_\_\_longjmp\_internal (69722DE0h)+0

6972587C add esp,8

}

6972587F pop ebp

69725880 ret

### \_\_except\_validate\_jump\_buffer

void

\_\_except\_validate\_jump\_buffer (

\_In\_ jmp\_buf JumpBuffer

)

/\*++

Routine Description:

This function validates a jump buffer for exception handling support.

Arguments:

JumpBuffer - Supplies a pointer to the jump buffer to validate.

Return Value:

None. If the jump buffer was not valid, a fast fail event is raised if

CFG was enforced.

--\*/

{

697257E0 push ebp

697257E1 mov ebp,esp

697257E3 sub esp,8

PVOID StackPointer;

PNT\_TIB Tib;

//

// If guard ICall checks are enforced, then validate the stack extents of

// the jump buffer and raise a fast fail exception if the extents are

// invalid. If checks are not enforced or the jump buffer was valid, then

// return.

//

if (\_guard\_icall\_checks\_enforced()) {

697257E6 call \_guard\_icall\_checks\_enforced (69725830h)

697257EB test eax,eax

697257ED je \_\_except\_validate\_jump\_buffer+3Dh (6972581Dh)

Tib = (PNT\_TIB)NtCurrentTeb();

697257EF call NtCurrentTeb (69725760h)+0

697257F4 mov dword ptr [Tib],eax

#pragma prefast(suppress:26007, "JumpBuffer is really a \_JUMP\_BUFFER in disguise and not a jmp\_buf. The code is correct.")

StackPointer = (PVOID)JUMP\_BUFFER\_TO\_STACK\_POINTER((\_JUMP\_BUFFER\*)JumpBuffer);

697257F7 mov eax,dword ptr [JumpBuffer]

697257FA mov ecx,dword ptr [eax+10h]

697257FD mov dword ptr [StackPointer],ecx

if ((StackPointer < Tib->StackLimit) ||

69725800 mov edx,dword ptr [Tib]

69725803 mov eax,dword ptr [StackPointer]

69725806 cmp eax,dword ptr [edx+8]

69725809 jb \_\_except\_validate\_jump\_buffer+36h (69725816h)

6972580B mov ecx,dword ptr [Tib]

6972580E mov edx,dword ptr [StackPointer]

69725811 cmp edx,dword ptr [ecx+4]

69725814 jbe \_\_except\_validate\_jump\_buffer+3Dh (6972581Dh)

(StackPointer > Tib->StackBase)) {

\_\_fastfail(FAST\_FAIL\_INVALID\_SET\_OF\_CONTEXT);

69725816 mov ecx,0Dh

6972581B int 29h

}

#ifndef JUMP\_BUFFER\_NO\_FRAME

if (((\_JUMP\_BUFFER\*)JumpBuffer)->Frame == 0) {

\_\_fastfail(FAST\_FAIL\_INVALID\_SET\_OF\_CONTEXT);

}

#endif

}

}

6972581D mov esp,ebp

}

#ifndef JUMP\_BUFFER\_NO\_FRAME

if (((\_JUMP\_BUFFER\*)JumpBuffer)->Frame == 0) {

\_\_fastfail(FAST\_FAIL\_INVALID\_SET\_OF\_CONTEXT);

}

#endif

}

}

6972581F pop ebp

69725820 ret

### \_\_\_longjmp\_internal(void)关键代码

69722DE0 push ebp

69722DE1 mov ebp,esp

69722DE3 sub esp,50h 给局部变量腾出空间

69722DE6 mov ebx,dword ptr [esp+58h] 第一个参数，jumpbuffer地址

69722DEA mov dword ptr [ebp-50h],80000026h 清零

69722DF1 mov dword ptr [ebp-4Ch],0

69722DF8 mov dword ptr [ebp-48h],0

69722DFF mov dword ptr [ebp-44h],0

69722E06 mov dword ptr [ebp-40h],0

69722E0D lea eax,[ebp-50h]

69722E10 mov ebp,dword ptr [ebx] 回复寄存器ebp

69722E12 mov esi,dword ptr [ebx+18h] 检查异常？第一次会跑到\_lj\_local\_unwind

69722E15 cmp esi,dword ptr fs:[0]

69722E1C je \_\_\_longjmp\_internal+50h (69722E30h)

69722E1E push ebx

69722E1F push esi

69722E20 push 0

69722E22 push eax

69722E23 push 69722E2Eh

69722E28 push esi

69722E29 call \_RtlUnwind@16 (69735016h)

\_lj\_return:

69722E2E pop esi

69722E2F pop ebx

\_lj\_local\_unwind:

69722E30 cmp esi,0

69722E33 je \_\_\_longjmp\_internal+85h (69722E65h)

69722E35 lea eax,[ebx+20h]

69722E38 push eax

69722E39 call \_rt\_probe\_read4 (697239E0h) 可能是检查所谓的cookie吧

69722E3E or eax,eax

69722E40 je \_\_\_longjmp\_internal+78h (69722E58h)

69722E42 mov eax,dword ptr [ebx+20h] 检查[20]位置的数字

69722E45 cmp eax,56433230h

69722E4A jne \_\_\_longjmp\_internal+78h (69722E58h)

69722E4C mov eax,dword ptr [ebx+24h]

69722E4F or eax,eax

69722E51 je \_\_\_longjmp\_internal+85h (69722E65h)

69722E53 push ebx

69722E54 call eax

69722E56 jmp \_\_\_longjmp\_internal+85h (69722E65h) 正常情况会跳到\_lj\_no\_unwind

\_lj\_old\_unwind:

69722E58 mov eax,dword ptr [ebx+1Ch]

69722E5B push eax

69722E5C push esi

69722E5D call \_\_local\_unwind2 (69722FD5h)

69722E62 add esp,8

\_lj\_no\_unwind:

69722E65 push 0

69722E67 mov eax,dword ptr [ebx+14h]

69722E6A call \_NLG\_Notify (69723085h)

69722E6F mov edx,ebx

69722E71 mov ebx,dword ptr [edx+4]

69722E74 mov edi,dword ptr [edx+8]

69722E77 mov esi,dword ptr [edx+0Ch]

69722E7A mov eax,dword ptr [esp+5Ch]

69722E7E cmp eax,1

69722E81 adc eax,0

69722E84 mov esp,dword ptr [edx+10h]

69722E87 add esp,4

69722E8A jmp dword ptr [edx+14h] 跳到原来记下来的地址。经调试，发现是调用call \_\_setjmp3的下一句，即add esp,8

69722E8D mov esp,ebp

69722E8F pop ebp

69722E90 ret