VC8编译器在编译可能发生缓冲区溢出的函数时，会定义一个很特别局部变量用来最为缓冲区溢出的判断。这个局部变量叫做SECURITY\_COOKIE，编译器增加的这个局部变量时紧挨着EBP的存放地址的。顺序就是：VAR COOKIE EBP RET ARGS。下面看一下最初的值：

#ifdef \_WIN64  
#define DEFAULT\_SECURITY\_COOKIE 0x00002B992DDFA232  
#else

#define DEFAULT\_SECURITY\_COOKIE 0xBB40E64E  
#endif

所以从Export的Start进去就是：

**int start()**

**{**

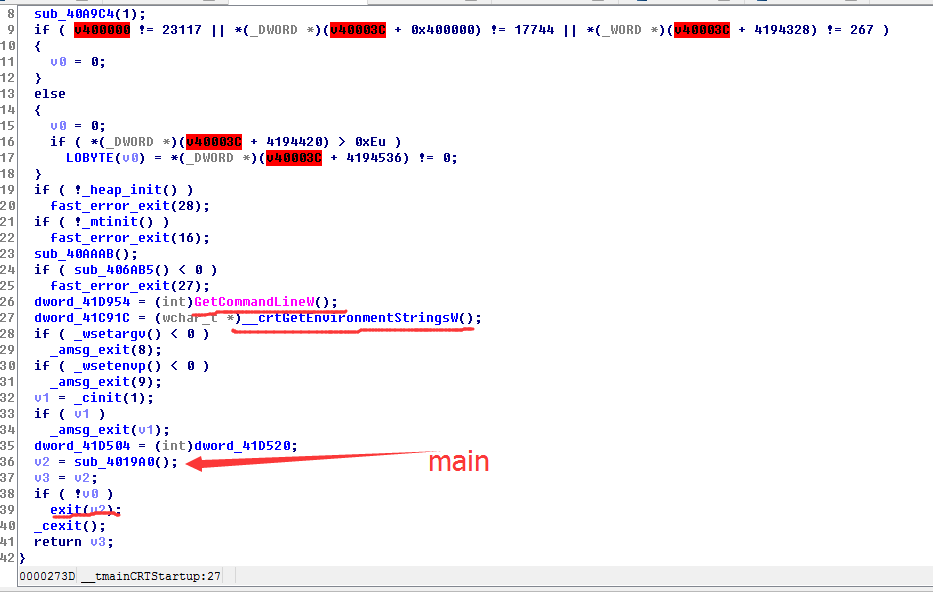
**\_\_security\_init\_cookie();**

**return \_\_tmainCRTStartup();**

**}**

**\_\_security\_init\_cookie**中可以看出来，为了取得好的随机性，先是取出时间，异或之，然后是分别跟其他一些列具有随机性的数据（进程ID，线程ID，TickCount和性能计数器）进行异或运算。VC编译器的版本不同，release文件中的mainCRTSartup函数也可能会有所不同，IDA中将mainCRTSartup函数命名为\_\_\_tmainCRTStartup。

main函数有三个参数，分别为命令行参数个数、命令行参数信息和环境变量信息。根据main函数调用的特征会将3个参数压入栈内作为函数的参数。所以查找用户入口main()前必然会有3个push指令。所以进入\_\_\_tmainCRTStartup函数以后，结合下面的void \_\_noreturn start()函数，找到应该是main的函数，可能会有参数传递吧（是IDA的解析问题）：



Microsoft Visual C++ 5.0[调试] 这种在IDA中F5以后会自动帮你定位到main函数，或者从Export标签页的start函数进入：

void \_\_noreturn start()

{

unsigned int v0; // eax@1

int v1; // eax@4

v0 = GetVersion();

dword\_441CD0 = (unsigned \_\_int16)v0 >> 8;

dword\_441CCC = (unsigned \_\_int8)v0;

dword\_441CC8 = dword\_441CD0 + ((unsigned \_\_int8)v0 << 8);

dword\_441CC4 = v0 >> 16;

if ( !sub\_414FD0(0) )

fast\_error\_exit(28);

\_ioinit();

dword\_442330 = (int)GetCommandLineA();

dword\_441CAC = \_\_crtGetEnvironmentStringsA();

\_setargv();

\_setenvp();

\_cinit();

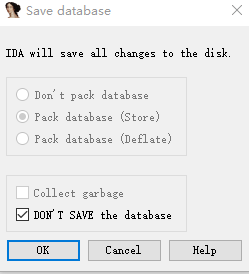
dword\_441CE4 = (int)envp;

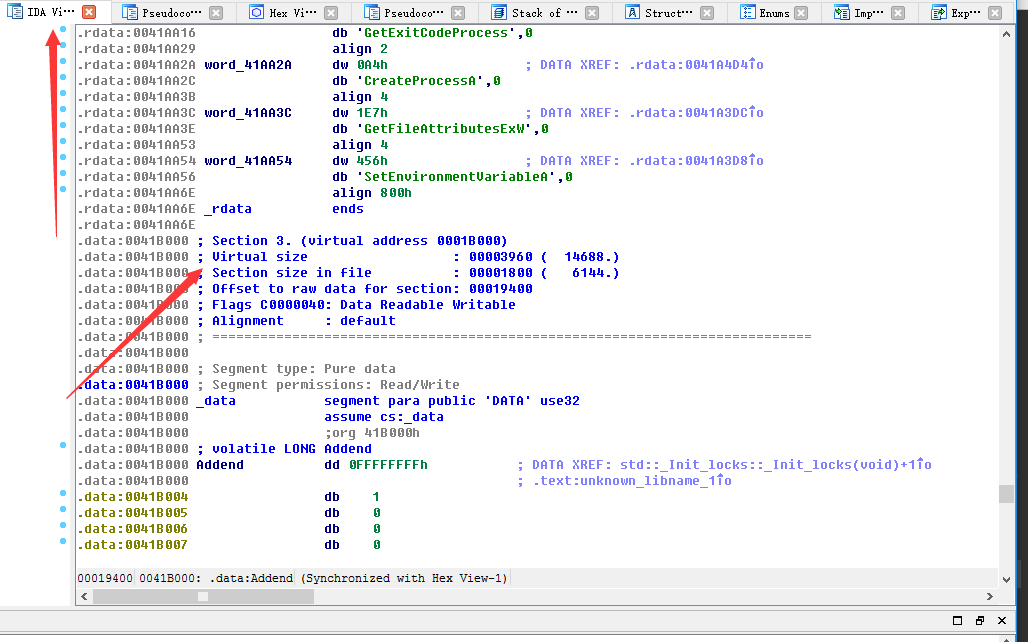
v1 = main(argc, (const char \*\*)argv, (const char \*\*)envp);

exit(v1);

}

如果要想最后什么都不保存的话：





如果使用Cygwin，那么export标签页中的函数有TlsCallback\_0, TlsCallback\_1, \_mainCRTStartup。

void \_\_noreturn mainCRTStartup()

{

\_\_\_set\_app\_type(1);

\_\_mingw\_CRTStartup();

}

void \_\_noreturn \_\_mingw\_CRTStartup()

{

const char \*\*\*v0; // eax@5

int v1; // ebx@5

char v2; // [sp+28h] [bp-14h]@3

int v3; // [sp+2Ch] [bp-10h]@3

if ( \_\_dyn\_tls\_init\_callback )

\_\_dyn\_tls\_init\_callback(0, 2, 0);

SetUnhandledExceptionFilter(\_gnu\_exception\_handler);

\_\_cpu\_features\_init();

fpreset();

v3 = 0;

\_\_getmainargs(&\_argc, &\_argv, &v2, \_CRT\_glob, &v3);

if ( \_CRT\_fmode )

{

\_fmode = \_CRT\_fmode;

\_setmode(\_\_iob[0].\_file, \_CRT\_fmode);

\_setmode(\_\_iob[1].\_file, \_CRT\_fmode);

\_setmode(\_\_iob[2].\_file, \_CRT\_fmode);

}

\*(\_DWORD \*)\_\_p\_\_fmode() = \_fmode;

\_pei386\_runtime\_relocator();

\_\_main();

v0 = (const char \*\*\*)\_\_p\_\_environ();

v1 = main(\_argc, (const char \*\*)\_argv, \*v0);

\_cexit();

ExitProcess(v1);

}

进入到main发现自己编写的函数名字都是在的….没有一点混淆… 这是因为处于debug模式算是，加载了DWARF information.

int \_\_cdecl main(int argc, const char \*\*argv, const char \*\*envp)

{

int v3; // eax@1

\_\_main();

v3 = getLocalMac();

printf("%d", v3);

return 0;

}

QT:

int WinMainCRTStartup()

{

mingw\_app\_type = 1;

\_\_security\_init\_cookie();

return \_\_tmainCRTStartup();

}

int \_\_tmainCRTStartup()

{

char \*v0; // edx@1

int v1; // ecx@1

void \*v2; // esp@1

void \*v3; // esp@1

unsigned int v4; // eax@1

signed \_\_int32 v5; // ebx@3

signed \_\_int32 v6; // eax@6

signed \_\_int32 v7; // ebx@7

CHAR \*v8; // eax@14

char v9; // cl@15

CHAR v10; // dl@20

signed int v11; // eax@28

int v12; // esi@31

int v13; // ebx@32

char \*\*v14; // esi@32

size\_t v15; // eax@33

size\_t v16; // edi@33

char \*v17; // eax@33

char \*v18; // ecx@33

int v19; // eax@34

char \*\*v20; // esi@35

int result; // eax@35

size\_t v22; // [sp+14h] [bp-74h]@31

int v23; // [sp+18h] [bp-70h]@1

char \*\*v24; // [sp+1Ch] [bp-6Ch]@31

char v25; // [sp+2Ch] [bp-5Ch]@1

char v26; // [sp+58h] [bp-30h]@28

unsigned \_\_int16 v27; // [sp+5Ch] [bp-2Ch]@29

v0 = &v25;

memset(&v25, 0, 0x44u);

v1 = 0;

v2 = alloca(48);

v3 = alloca(48);

v4 = ((unsigned int)&v23 + 3) & 0xFFFFFFF0;

\*(\_DWORD \*)v4 = -858993460;

\*(\_DWORD \*)(v4 + 4) = -858993460;

\*(\_DWORD \*)(v4 + 8) = -858993460;

\*(\_DWORD \*)(v4 + 12) = -858993460;

\*(\_DWORD \*)(v4 + 16) = -858993460;

\*(\_DWORD \*)(v4 + 20) = -858993460;

\*(\_DWORD \*)(v4 + 24) = -858993460;

\*(\_DWORD \*)(v4 + 28) = -858993460;

if ( mingw\_app\_type )

GetStartupInfoA((LPSTARTUPINFOA)&v25);

v5 = \*(\_DWORD \*)(\_\_readfsdword(24) + 4);

while ( 1 )

{

v6 = \_InterlockedCompareExchange((volatile signed \_\_int32 \*)&\_\_native\_startup\_lock, v5, 0);

if ( !v6 )

{

v7 = 0;

if ( \_\_native\_startup\_state == 1 )

goto LABEL\_40;

goto LABEL\_8;

}

if ( v6 == v5 )

break;

Sleep(0x3E8u);

}

v7 = 1;

if ( \_\_native\_startup\_state == 1 )

{

LABEL\_40:

\_amsg\_exit(v1, v0);

if ( \_\_native\_startup\_state == 1 )

goto LABEL\_41;

LABEL\_11:

if ( v7 )

goto LABEL\_12;

goto LABEL\_42;

}

LABEL\_8:

if ( \_\_native\_startup\_state )

{

has\_cctor = 1;

}

else

{

\_\_native\_startup\_state = 1;

\_initterm(v1, v0);

}

if ( \_\_native\_startup\_state != 1 )

goto LABEL\_11;

LABEL\_41:

\_initterm(v1, v0);

\_\_native\_startup\_state = 2;

if ( v7 )

goto LABEL\_12;

LABEL\_42:

\_InterlockedExchange((volatile signed \_\_int32 \*)&\_\_native\_startup\_lock, v7);

LABEL\_12:

if ( \_\_dyn\_tls\_init\_callback )

\_\_dyn\_tls\_init\_callback(v1, v0, 0, 2, 0);

\_pei386\_runtime\_relocator(v1, v0);

\_\_mingw\_oldexcpt\_handler = (int)SetUnhandledExceptionFilter(\_gnu\_exception\_handler);

\_set\_invalid\_parameter\_handler(\_\_mingw\_invalidParameterHandler);

fpreset();

\_\_mingw\_winmain\_hInstance = (HINSTANCE)0x400000;

v8 = (CHAR \*)\_\_acmdln;

if ( !\_\_acmdln )

goto LABEL\_27;

v9 = 0;

while ( 1 )

{

v10 = \*v8;

if ( \*v8 <= 32 )

break;

if ( v10 == 34 )

v9 ^= 1u;

LABEL\_19:

++v8;

}

if ( !v10 )

goto LABEL\_26;

if ( v9 & 1 )

{

v9 = 1;

goto LABEL\_19;

}

do

++v8;

while ( \*v8 <= 32 && \*v8 );

LABEL\_26:

\_\_mingw\_winmain\_lpCmdLine = v8;

LABEL\_27:

if ( mingw\_app\_type )

{

v11 = 10;

if ( v26 & 1 )

v11 = v27;

\_\_mingw\_winmain\_nShowCmd = v11;

}

v23 = argc;

v12 = argc;

v22 = 4 \* argc + 4;

v24 = (char \*\*)malloc(v22);

if ( v12 <= 0 )

{

v19 = 0;

}

else

{

v13 = 0;

v14 = argv;

do

{

v15 = strlen(v14[v13]);

v16 = v15 + 1;

v17 = (char \*)malloc(v15 + 1);

v24[v13] = v17;

v18 = v14[v13++];

memcpy(v17, v18, v16);

}

while ( v13 != v23 );

v19 = v22 - 4;

}

v20 = v24;

\*(char \*\*)((char \*)v24 + v19) = 0;

argv = v20;

\_\_main();

\_\_\_initenv = envp;

result = main(argc, (const char \*\*)argv, (const char \*\*)envp);

mainret = result;

if ( !managedapp )

exit(result);

if ( !has\_cctor )

{

\_cexit();

result = mainret;

}

return result;

}

int \_\_cdecl main(int argc, const char \*\*argv, const char \*\*envp)

{

\_\_main();

return WinMain(\_\_mingw\_winmain\_hInstance, 0, \_\_mingw\_winmain\_lpCmdLine, \_\_mingw\_winmain\_nShowCmd);

}