

导入符号
用CE寻找切入点
分析算法
写出注册机
校验结果

【软件名称】：Dope2112.2.exe

【软件大小】：171kb

【下载地址】：自行搜索下载

【加壳方式】：无壳

【保护方式】：Name/Serial

【编译语言】：Delphi

【调试环境】：W10 64

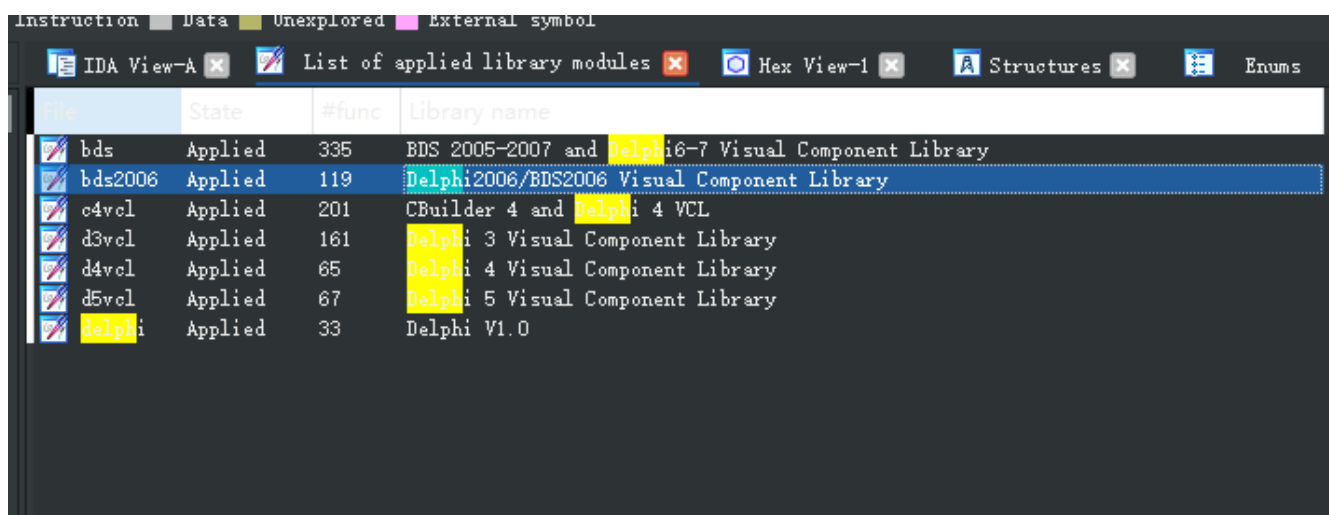
【使用工具】：OD+IDA

【破解日期】：2019-5-28

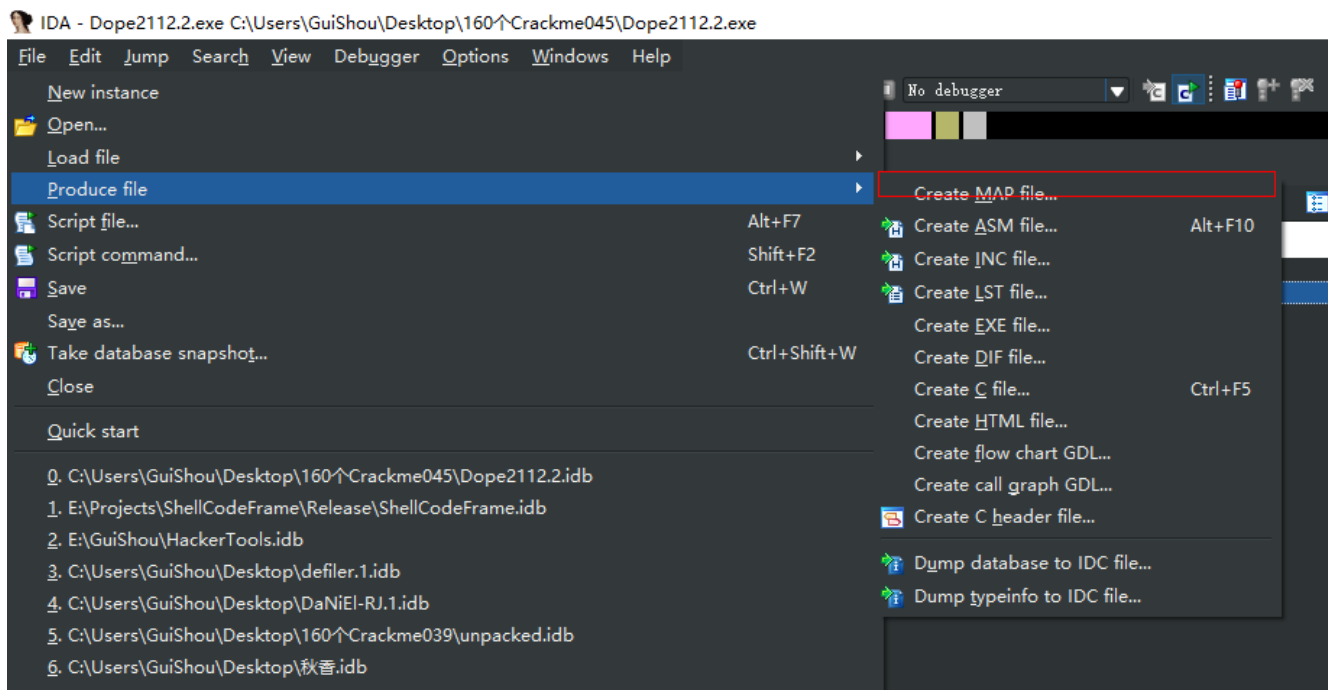
【破解目的】：纯属兴趣

导入符号

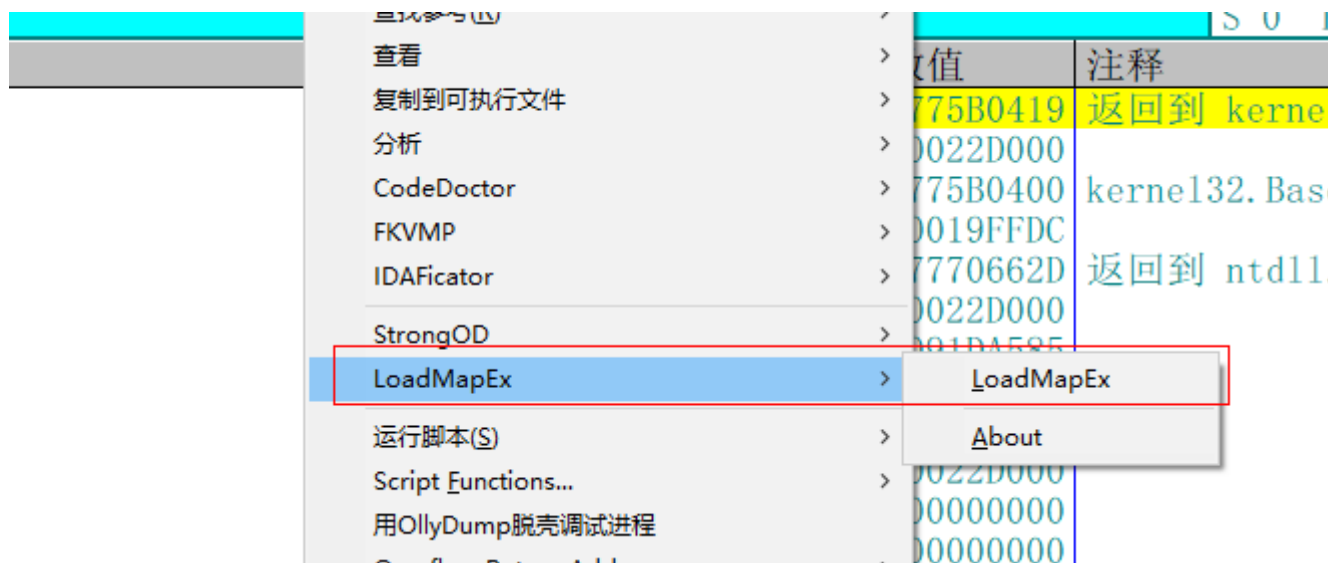
首先将程序载入到IDA，导入所有的Delphi签名



接着生成map文件



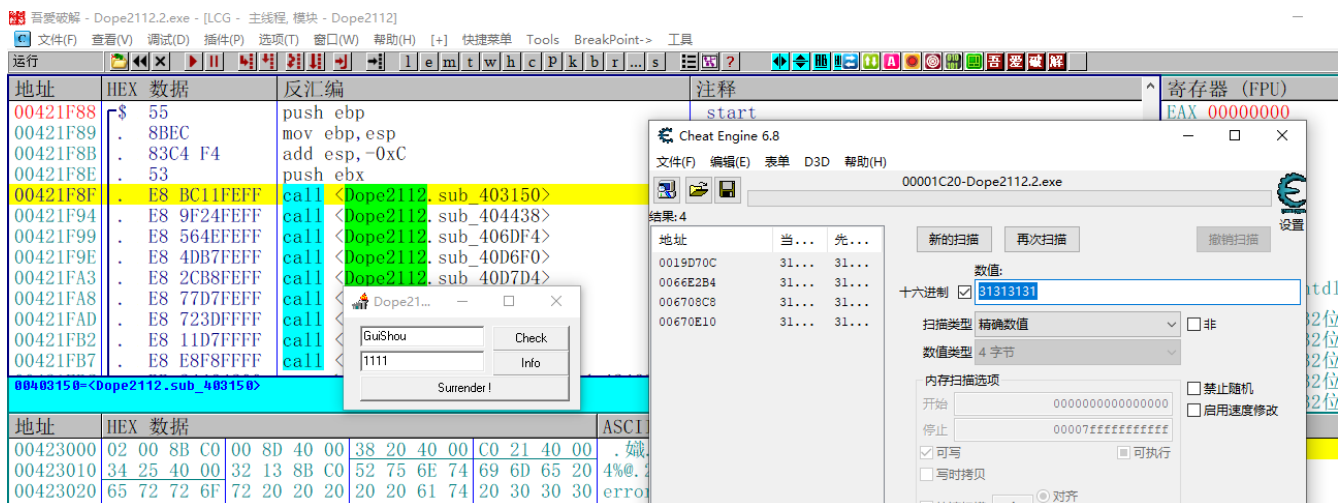
然后在OD中导入



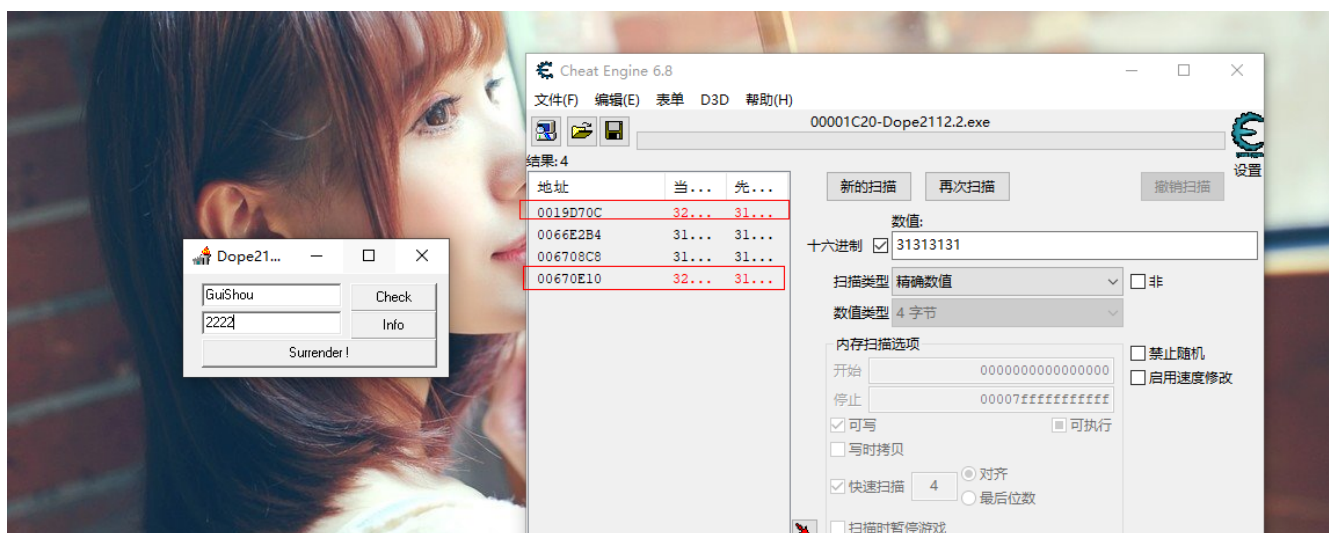
用CE寻找切入点

由于不能用DarkDe，而且无法搜索到字符串，我试了好几个API断点也断不下来，所以只能用CE来搜索地址，首先用OD载入，然后再打开CE，这样可以省去后面再用OD附加的麻烦

由于Delphi中存储的都是ASCII的形式，所以我们直接搜索1111的十六进制ASCII值31313131



接着变换1111为2222



可以很明显的看到我们要的地址就是这两个中的一个了

找到上面两个地址，依次对下面两个地址下一个字节的内存访问断点

地址	HEX 数据	ASCII
0019D70C	32 32 32 32 00 6C 00 00	2222. 1. P?. □..
0019D71C	05 00 00 00 00 00 00 00	□..... 氐
0019D72C	00 00 00 00 18 00 00 00 □.....
0019D73C	18 00 00 00 00 00 00 00	□..... T?. @..
0019D74C	68 D8 19 00 00 10 00 00	h?.. □..... ?..
0019D75C	00 00 00 00 03 00 00 00 □.....
0019D76C	41 C9 F1 7C 90 D7 19 00	A神 悵□. 净Zw柏
0019D77C	A0 50 6E 77 01 00 00 00	嫩nw□.....
0019D78C	D0 D7 19 00 0A 18 DB 75	凶□.. □踰□...
0019D79C	08 59 63 00 D8 07 05 00	□Yc. ?□. 胸□. d.
0019D7AC	F0 D7 19 00 00 B8 3E 00	驢□.. ? . ?a. 櫟□
0019D7BC	98 DD 19 00 98 DD 19 00	櫟□. 櫟□. s 棧□
0019D7CC	FE FF FF FF 8C D8 19 00	? 櫟□. V 庑ux

地址	HEX 数据												ASCII
00670E10	32	32	32	32	35	36	37	00	00	00	00	00	2222567.....
00670E20	00	00	00	00	00	00	00	00	00	00	00	00
00670E30	00	00	00	00	00	00	00	00	00	00	00	44 D. y
00670E40	93	5D	13	D4	91	6A	00	08	28	00	00	00	揮□詰j.□(...□
00670E50	0A	00	00	00	01	00	18	00	00	00	00	00□.□.....?
00670E60	13	0B	00	00	13	0B	00	00	00	00	00	00	□□..□□.....
00670E70	FF	79	30	FF	79	30	FF	79	30	FF	79	30	y0 y0 y0 y
00670E80	79	30	FF	79	30	FF	79	30	FF	79	30	FF	y0 y0 y0 y0
00670E90	30	FF	79	30	FF	79	30	FF	79	30	FF	79	0 y0 y0 y0
00670EA0	FF	79	30	FF	79	30	FF	79	30	FF	79	30	y0 y0 y0 y
00670EB0	79	30	FF	79	30	FF	79	30	FF	79	30	FF	y0 y0 y0 y0
00670EC0	30	FF	79	30	FF	79	30	FF	79	30	FF	79	0 y0 y0 y0
00670ED0	FF	79	30	FF	79	30	FF	79	30	FF	79	30	y0 y0 y0 y

程序断下

吾爱破解 - Dope2112.2.exe - [LCG - 主线程, 模块 - ntdll]

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> 工具

暂停

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
777164A0	8B448E FC	mov eax, dword ptr ds:[esi+ecx*4-0x4]		EAX 00670E14 ASCII "567"
777164A4	89448F FC	mov dword ptr ds:[edi+ecx*4-0x4], eax		ECX 00000001
777164A8	8D048D 00000000	lea eax, dword ptr ds:[ecx*4]		EDX 00000000
777164AF	03F0	add esi, eax		EBX 02195580 ASCII "2222"
777164B1	03F8	add edi, eax		ESP 0019F3F0
777164B3	FF2495 BC647177	jmp dword ptr ds:[edx*4+0x777164BC]	ntdll.777164CC	EBP 0019F3F8
777164BA	8BFF	mov edi, edi		ESI 00670E10 ASCII "2222567"
777164BC	CC	int3		EDI 02195580 ASCII "2222"
777164BD	64:71 77	jmp short 77716537		EIP 777164A0 ntdll.777164A0
777164C0	D4 64	aam 0x64		C 1 ES 002B 32位 0(FFFFFFFF)
777164C2	71 77	jmp short ntdll.7771653B		P 1 CS 0023 32位 0(FFFFFFFF)
777164C4	E0 64	loopdne short ntdll.7771652A		A 1 SS 002B 32位 0(FFFFFFFF)
777164C6	71 77	jmp short ntdll.7771653F		Z 0 DS 002B 32位 0(FFFFFFFF)
777164C8				S 1 FS 0053 32位 3ED000(FF)

循环未实现
ecx=00000001 (十进制 1.)

地址	HEX 数据	ASCII	地址	数值	注释
00670E10	32 32 32 32 35 36 37 00	2222567.....	0019F3F0	00000004	
00670E20	00 00 00 00 00 00 00 00	0019F3F4	006A6B00	
00670E30	00 00 00 00 00 00 00 00 D. y	0019F3F8	0019F418	
00670E40	93 5D 13 D4 91 6A 00 08	揮□詰j.□(...□	0019F3FC	76EB2953	返回到 user32.76EB2953 来自 <jmp.&nt
00670E50	0A 00 00 00 01 00 18 00□.□.....?	0019F400	02195580	ASCII "2222"
00670E60	13 0B 00 00 13 0B 00 00	□□..□□.....	0019F404	00670E10	ASCII "2222567"
00670E70	FF 79 30 FF 79 30 FF 79	y0 y0 y0 y	0019F408	00000004	

接着点击K, 查看调用堆栈

吾爱破解 - Dope2112.2.exe - [调用堆栈: 主线程]

文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> 工具

暂停

地址	堆栈	函数过程 / 参数	调用来自	结构
0019F3FC	76EB2953	<jmp.&ntdll.memcpy>	user32.76EB294E	0019F3F8
0019F400	02195580	dest = 02195580		
0019F404	00670E10	src = 00670E10		
0019F408	00000004	n = 0x4		
0019F41C	76EB1530	user32.76EB290B	user32.76EB152B	0019F418
0019F468	76EB102C	user32.EditWndProc	user32.76EB1027	0019F464
0019F49C	76ED635B	包含user32.76EB102C	user32.76ED6359	0019F498
0019F4C8	76EC729C	user32.76ED6330	user32.76EC7297	0019F4C4
0019F5AC	76EB6C42	user32.76EC6EF0	user32.76EB6C3D	0019F5A8
0019F5E4	0041391D	<Dope2112.CallWindowProcA>	Dope2112.sub_41387C+9C	0019F5E0
0019F5E8	77724570	PrevProc = ntdll.77724570		
0019F5EC	000407A8	hWnd = 000407A8 ('Serial', class='TEdit', parent=000F065)		
0019F5F0	0000000D	Message = WM_GETTEXT		
0019F5F4	00000005	Count = 0x5		
0019F5F8	02195580	Buffer = 02195580		
0019F60C	004213A9	<Dope2112.sub_41387C>	Dope2112.004213A4	
0019F618	001205B8	<Dope2112.System::TObject::Dispatch(void *)>	Dope2112.sub_411FB4+0A2	0019F64C
0019F628	00413873	<Dope2112.sub_411FB4>	Dope2112.sub_413754+11A	0019F64C
0019F648	00411FA7	可能 <Dope2112.sub_413754>	Dope2112.sub_411FB4+20	0019F64C
0019F660	00411848	? <Dope2112.sub_411FB4>	Dope2112.Controls::TControl	0019F64C
0019F668	004118AB	? <Dope2112.Controls::TControl::GetTextBuf(char *,int)>	Dope2112.Controls::TControl	
0019F678	00421DF7	<Dope2112.Controls::TControl::GetText(void)>	Dope2112.sub_421DC8+2A	

可以看到当前所在的是memcpy这个函数，而最下面的是GetText获取用户输入，直接记下这个地址0x421DF7

分析算法

找到0x421DF7函数头的位置0x00421DC8，开始分析整个算法，校验步骤如下

1. 获取序列号

Dope2112.2.exe - [LCG - 主线程, 模块 - Dope2112]					
文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> 工具					
暂停					
地址	HEX	数据	反汇编	注释	寄存器 (FPU)
00421DD9	. 68	861E4200	push <Dope2112. loc_421E86>		EAX 00000004
00421DDE	. 64:FF30		push dword ptr fs:[eax]		ECX 1F6B30F4
00421DE1	. 64:8920		mov dword ptr fs:[eax],esp		EDX 00000000
00421DE4	. BB	37000000	mov ebx,0x37		EBX 00000037
00421DE9	. 8D55 F8		lea edx,[local.2]		ESP 0019F67C
00421DEC	. 8B86 B0010000		mov eax,dword ptr ds:[esi+0x1B0]		EBP 0019F6A0
00421DF2	. E8 89FAFEFF		call <Dope2112.Controls::TControl::GetText(void)>		ESI 02192490 ASCII "dB"
00421DF7	. 8D55 FC		lea edx,[local.1]		EDI 02194E30
00421DFA	. 8B86 AC010000		mov eax,dword ptr ds:[esi+0x1AC]		EIP 00421DF7 Dope2112.0
00421E00	. E8 7BFAFEFF		call <Dope2112.Controls::TControl::GetText(void)>		C 0 ES 002B 32位 0(FFF
00421E05	. 8B45 FC		mov eax,[local.1]		P 0 CS 0023 32位 0(FFF
00421E08	. E8 5715FEFF		call <Dope2112.__linkproc__ LStrLen>		A 0 SS 002B 32位 0(FFF
00421E0D	. 83F8 04		cmp eax,0x4		Z 0 DS 002B 32位 0(FFF
00411880<Dope2112.Controls::TControl::GetText(void)>					
地址	数值	注释	地址	数值	注释
0019F698	02195580	ASCII "2222"	0019F67C	0019F780	指向下一个 SEH 记录的指针
0019F69C	00000000		0019F680	00421E86	SE处理程序

2. 获取用户名

00421DF2	. E8 89FAFEFF	call <Dope2112.Controls::TControl::GetText(void)>			
00421DF7	. 8D55 FC	lea edx,[local.1]			
00421DFA	. 8B86 AC010000	mov eax,dword ptr ds:[esi+0x1AC]			
00421E00	. E8 7BFAFEFF	call <Dope2112.Controls::TControl::GetText(void)>			
00421E05	. 8B45 FC	mov eax,[local.1]			
00421E08	. E8 5715FEFF	call <Dope2112.__linkproc__ LStrLen>			
00421E0D	. 83F8 04	cmp eax,0x4			
堆栈 ss:[0019F69C]=02195870, (ASCII "GuiShou") eax=00000007					
地址	数值	注释	地址	数值	注释
0019F69C	02195870	ASCII "GuiShou"	0019F67C	0019F780	指向下一个
0019F6A0	0019F708		0019F680	00421E86	SE处理程序

3. 比较用户名长度是否大于等于4

Dope2112.2.exe - [LCG - 主线程, 模块 - Dope2112]					
文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> 工具					
暂停					
地址	HEX	数据	反汇编	注释	寄存器 (FPU)
00421DE9	. 8D55 F8		lea edx,[local.2]		EAX 00000007
00421DEC	. 8B86 B0010000		mov eax,dword ptr ds:[esi+0x1B0]		ECX 1F6B30F4
00421DF2	. E8 89FAFEFF		call <Dope2112.Controls::TControl::GetText(void)>		EDX 00000000
00421DF7	. 8D55 FC		lea edx,[local.1]		EBX 00000037
00421DFA	. 8B86 AC010000		mov eax,dword ptr ds:[esi+0x1AC]		ESP 0019F67C
00421E00	. E8 7BFAFEFF		call <Dope2112.Controls::TControl::GetText(void)>		EBP 0019F6A0
00421E05	. 8B45 FC		mov eax,[local.1]		ESI 02192490 ASCII "dB"
00421E08	. E8 5715FEFF		call <Dope2112.__linkproc__ LStrLen>		EDI 02194E30
00421E0D	. 83F8 04		cmp eax,0x4		EIP 00421E0D Dope2112.0
00421E10	. 7D 0C		jge short <Dope2112.loc_421E1E>		C 0 ES 002B 32位 0(FFF
00421E12	. A1 64464200		mov eax,dword ptr ds:[<dword_424664>]		P 0 CS 0023 32位 0(FFF
00421E17	. E8 C8BAFFFF		call <Dope2112.Forms::TCustomForm::Show(void)>		A 0 SS 002B 32位 0(FFF
00421E1C	. EB 4D		jmp short <Dope2112.loc_421E6B>		

4. 再次获取用户名长度

Dope2112.exe - [LCG - 主线程, 模块 - Dope2112]				
文件(F) 查看(V) 调试(D) 插件(P) 选项(T) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint-> 工具				
暂停				
地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00421E10	7D 0C	jge short <Dope2112.loc_421E1E>		EAX 00000007
00421E12	A1 64464200	mov eax,dword ptr ds:[<dword_424664>]		ECX 1F6B30F4
00421E17	E8 C8BAFFFF	call <Dope2112.Forms::TCustomForm::Show(void)>		EDX 00000000
00421E1C	EB 4D	jmp short <Dope2112.loc_421E6B>		EBX 00000037
00421E1E	8B45 FC	mov eax,[local.1]	loc_421E1E	ESP 0019F67C
00421E21	E8 3E15FEFF	call <Dope2112.__linkproc__ LStrLen>		EBP 0019F6A0
00421E26	85C0	test eax,edx		ESI 02192490 ASCII
00421E28	7C 14	jl short <Dope2112.loc_421E3E>		EDI 02194E30
00421E2A	40	inc eax		EIP 00421E26 Dope2
00421E2B	33D2	xor edx,edx		C 0 ES 002B 32位
00421E2D	8B4D FC	mov ecx,[local.1]	loc_421E2D	P 0 CS 0023 32位
00421E30	0FB64C11 FF	movzx ecx,byte ptr ds:[ecx+edx-0x1]		A 0 SS 002B 32位
00421E35	C1E1 09	shl ecx,0x9		Z 0 DS 002B 32位

5. 循环累加用户名的ASCII值右移9位的和

00421E2D	8B4D FC	mov ecx,[local.1]	loc_421E2D
00421E30	0FB64C11 FF	movzx ecx,byte ptr ds:[ecx+edx-0x1]	username[i]
00421E35	C1E1 09	shl ecx,0x9	
00421E38	03D9	add ebx,ecx	
00421E3B	42	inc edx	
00421E3D	48	dec eax	
00421E3F	75 EF	jnz short <Dope2112.loc_421E2D>	
00421E41	8B55 F4	lea edx,[local.3]	loc_421E3E
00421E43	E8 E834FEFF	call <Dope2112.Sysutils::IntToStr(int)>	
00421E48	8B45 F4	mov eax,[local.3]	
00421E4B	8B55 F8	mov edx,[local.2]	
00421E4E	E8 2116FEFF	call <Dope2112.System::__linkproc__ LStrCmp(void)>	
00405330	<Dope2112.Sysutils::IntToStr(int)>		

```

if ( usernameLen >= 0 )
{
    v13 = usernameLen + 1;
    i = 0;
    do
    {
        v12 = *(unsigned __int8 *)(&v26 + i - 1) << 9;
        v6 += v12;
        ++i;
        --v13;
    }
    while ( v13 );

```

6. 将累加的结果转为字符串

00421E30	42	inc edx	
00421E3B	48	dec eax	
00421E3C	75 EF	jnz short <Dope2112.loc_421E2D>	
00421E3E	8D55 F4	lea edx,[local.3]	loc_421E3E
00421E41	8BC3	mov eax,ebx	
00421E43	E8 E834FEFF	call <Dope2112.Sysutils::IntToStr(int)>	
00421E48	8B45 F4	mov eax,[local.3]	
00421E4B	8B55 F8	mov edx,[local.2]	
00421E4E	E8 2116FEFF	call <Dope2112.System::__linkproc__ LStrCmp(void)>	
00405330	<Dope2112.Sysutils::IntToStr(int)>		

地址	数值	注释	地址	数值	注释
0019F694	02195884	ASCII "362551"	0019F67C	0019F780	指向下一个

7. 和序列号进行比较

地址	HEX 数据	反汇编	注释
00421E35	. C1E1 09	shl ecx,0x9	
00421E38	. 03D9	add ebx,ecx	
00421E3A	. 42	inc edx	
00421E3B	. 48	dec eax	
00421E3C	. ^ 75 EF	jmp short <Dope2112.loc_421E2D>	
00421E3E	. > 8D55 F4	lea edx,[local.3]	loc_421E3E
00421E41	. 8BC3	mov eax,ebx	
00421E43	. E8 E834FEFF	call <Dope2112.Sysutils::IntToStr(int)>	
00421E48	. 8B45 F4	mov eax,[local.3]	
00421E4B	. 8B55 F8	mov edx,[local.2]	
00421E4E	. E8 2116FEFF	call <Dope2112.System::__linkproc__LStrCmp(void)>	

下面贴上IDA的伪代码注释

```

3/ v0 = 55;
38 v7 = *(_DWORD *)(a1 + 432);
39 Controls::TControl::GetText(a2); // 获取序列号
40 v8 = *(_DWORD *)(v5 + 428);
41 Controls::TControl::GetText(v9); // 获取用户名
42 if ( __linkproc__ LStrLen() >= 4 ) // 检查用户名长度是否大于等于4
43 {
44     usernameLen = __linkproc__ LStrLen();
45     if ( usernameLen >= 0 )
46     {
47         v13 = usernameLen + 1;
48         i = 0;
49         do
50         {
51             v12 = *(unsigned __int8 *)(v26 + i - 1) << 9; // 循环累加用户名的ASCII值右移9位的和
52             v6 += v12;
53             ++i;
54             --v13;
55         }
56         while ( v13 );
57     }
58     Sysutils::IntToStr(v12); // 将累加的结果转为字符串
59     System::__linkproc__ LStrCmp(v15, v25); // 将累加的结果和序列号进行比较
60     Forms::TCustomForm::Show(v16);

```

写出注册机

注册机代码如下:

```

#define _CRT_SECURE_NO_WARNINGS
#include <windows.h>
#include <stdio.h>

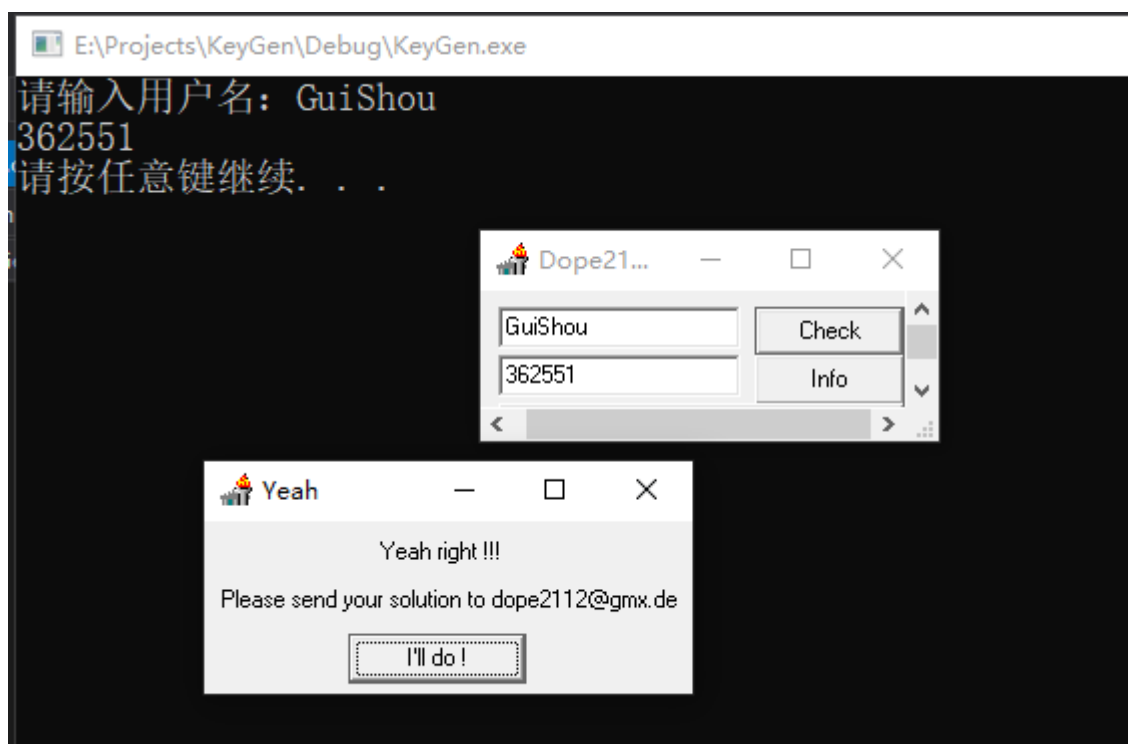
int main()
{
    int result = 0x37;
    char username[20] = { 0 };
    char serial[20] = { 0 };
    printf("请输入用户名: ");
    scanf_s("%s", username, 20);
    int usernameLen = strlen(username);
    if (usernameLen >= 4)
    {
        for (int i = 0; i < usernameLen; i++)

```

```
    {  
        result += username[i] << 9;  
    }  
}  
printf("%d\n", result);  
system("pause");  
return 0;  
}
```

校验结果

输入用户名和计算的序列号



提示成功 破解完成

需要相关文件的可以到我的Github下载: <https://github.com/TonyChen56/160-Crackme>