

- 寻找切入点
- 算法分析
 - 基础校验
 - 第一部分
 - 第二部分
 - 校验部分
- 写出注册机
- 校验结果

【软件名称】：Dope2112.1.exe

【软件大小】：178KB

【下载地址】：自行搜索下载

【加壳方式】：无壳

【保护方式】：无保护

【编译语言】：Delphi

【调试环境】：W7 32

【使用工具】：OD + IDA

【破解日期】：2019年5月21日

【破解目的】：纯属兴趣

寻找切入点

00421D42	> 8B86 C0010000	mov eax,dword ptr ds:[esi+0x1C0]	Hey du hast es geschafft !
00421D48	. BA E41D4200	mov edx,<Dope2112.aHeyDuHastEsGes>	
00421D4D	. E8 D6FAFEFF	call <Dope2112.sub_411828>	
00421D52	> EB 10	jmp short <Dope2112.loc_421D64>	
00421D54	> 8B86 C0010000	mov eax,dword ptr ds:[esi+0x1C0]	loc_421D54
00421D5A	. BA 081E4200	mov edx,<Dope2112.aLeiderNichtVer>	Leider nicht versuchs noch mal !
00421D5F	. E8 C4FAFEFF	call <Dope2112.sub_411828>	
00421D64	> 33C0	xor eax,eax	loc_421D64
00421D66	. 5A	pop edx	kernel32.75E13C45
00421D67	. 59	pop ecx	kernel32.75E13C45
00421D68	. 59	pop ecx	kernel32.75E13C45
00421D69	. 64:8910	mov dword ptr fs:[eax],edx	<Dope2112.start>
00421D6C	. 68 961D4200	push <Dope2112.loc_421D96>	
00421D71	> 8D45 E4	lea eax,[local.7]	loc_421D71
00421D74	. E8 7714FEFF	call <Dope2112.sub_4031F0>	

首先搜索关键字字符串，根据这个地址对应到IDA可以很快找到函数头部00421B84，从这里开始分析整个算法

算法分析



随便输入一个用户名和序列号，开始分析整个注册算法

基础校验

```

51  v25 = __readfsdword(0);
52  __writefsdword(0, (unsigned int)&v25);
53  TotalSum = 0;
54  v6 = *(_DWORD *)(a1 + 428);
55  Controls::TControl::GetText(0);           // 获取用户名
56  Sysutils::LowerCase(username);           // 将用户名全部转为小写
57  v8 = *(_DWORD *)(v4 + 432);
58  Controls::TControl::GetText(Serial);      // 获取序列号
59  HIBYTE(usernameLen) = __linkproc__ LStrLen(); // 获取用户名长度
60  if ( HIBYTE(usernameLen) >= 6u )          // 比较用户名长度是否大于等于6
61  {

```

首先是基础校验部分，在IDA中如图，下面开始详细讲解每一步

1. 获取用户名

00421BA0	. 64:8920	mov dword ptr fs:[eax],esp	
00421BA3	. 33DB	xor ebx,ebx	
00421BA5	. 8D55 E8	lea edx,[local.6]	
00421BA8	. 8B86 AC010000	mov eax,dword ptr ds:[esi+0x1AC]	
00421BAE	. E8 45FCFEFF	call <Dope2112.Controls::TControl::GetText>	获取用户名
00421BB3	. 8B45 E8	mov eax,[local.6]	
00421BB6	. 8D55 FC	lea edx,[local.1]	
00421BB9	. E8 8A36FEFF	call <Dope2112.Sysutils::LowerCase(System::String::ToLower>	将用户名全部转为小写
00421BBE	. 8D55 F8	lea edx,[local.2]	
00421BC1	. 8B86 B0010000	mov eax,dword ptr ds:[esi+0x1B0]	
00421BC7	. E8 2CFCFEFF	call <Dope2112.Controls::TControl::GetText>	获取序列号
00421BCC	. 8B45 FC	mov eax,[local.1]	
00421BCF	. E8 9017FEFF	call <Dope2112.__linkproc__ LStrLen>	获取用户名长度
00421BD4	. 8845 EF	mov byte ptr ss:[ebp-0x11],al	
00421BD7	. 807D EF 06	cmp byte ptr ss:[ebp-0x11],0x6	比较用户名长度是否大于等于6
堆栈 ss:[0012F934]=01154928, (ASCII "GuiShou")			
eax=00000007			
地址	数值	注释	地址
0012F934	01154928	ASCII "GuiShou"	0012F934
0012F938	00000000		0012F938

2. 将用户名全部转为小写

00421BA8	8B86 AC010000	mov eax,dword ptr ds:[esi+0x1AC]	
00421BAE	E8 45FCFEFF	call <Dope2112.Controls::TControl::GetTe	获取用户名
00421BB3	8B45 E8	mov eax,[local.6]	
00421BB6	8D55 FC	lea edx,[local.1]	
00421BB9	E8 8A36FEFF	call <Dope2112.Sysutils::LowerCase(Syste	将用户名全部转为小写
00421BBE	8D55 F8	lea edx,[local.2]	
00421BC1	8B86 B0010000	mov eax,dword ptr ds:[esi+0x1B0]	
00421BC7	E8 2CFCFEFF	call <Dope2112.Controls::TControl::GetTe	获取序列号
00421BCC	8B45 FC	mov eax,[local.1]	
00421BCF	E8 9017FEFF	call <Dope2112.__linkproc__ LStrLen>	获取用户名长度
00421BD4	8845 EF	mov byte ptr ss:[ebp-0x11],al	
00421BD7	807D EF 06	cmp byte ptr ss:[ebp-0x11],0x6	比较用户名长度是否大于等于6
堆栈地址=0012F944 edx=0115492F			
地址	数值	注释	
0012F948	011559C4	ASCII "guishou"	
0012F94C	0012F9B4		
0012F950	0041210A	返回到 Dope2112.sub_4120F0+1A	
0012F954	00420ADD	返回到 Dope2112.00420ADD 来自 <Dope2112.sub_4120F0>	
地址	数值	注释	
0012F918	0012FA2C	指向下	
0012F91C	00421D8F	SE处理	
0012F920	0012F94C		
0012F924	01154CAC		

3. 获取输入的序列号

00421BBE	8D55 F8	lea edx,[local.2]	
00421BC1	8B86 B0010000	mov eax,dword ptr ds:[esi+0x1B0]	
00421BC7	E8 2CFCFEFF	call <Dope2112.Controls::TControl::GetTe	获取序列号
00421BCC	8B45 FC	mov eax,[local.1]	
00421BCF	E8 9017FEFF	call <Dope2112.__linkproc__ LStrLen>	获取用户名长度
00421BD4	8845 EF	mov byte ptr ss:[ebp-0x11],al	
00421BD7	807D EF 06	cmp byte ptr ss:[ebp-0x11],0x6	比较用户名长度是否大于等
堆栈 ss:[0012F948]=011559C4, (ASCII "guishou") eax=00000000			
地址	数值	注释	地址
0012F944	011559D8	ASCII "123456789"	001
0012F948	011559C4	ASCII "guishou"	001
0012F94C	0012F9B4		001

4. 比较用户名长度是否大于等于6

地址	HEX	数据	反汇编	注释	寄存器 (FPU)
00421BAE	E8 45FCFEFF		call <Dope2112.Controls::TControl::GetTe	获取用户名	EAX 00000007
00421BB3	8B45 E8		mov eax,[local.6]		ECX 75EBC5FE user32.75EB
00421BB6	8D55 FC		lea edx,[local.1]		EDX 00000030
00421BB9	E8 8A36FEFF		call <Dope2112.Sysutils::LowerCase(Syste	将用户名全部转为小写	EBX 00000000
00421BBE	8D55 F8		lea edx,[local.2]		ESP 0012F918
00421BC1	8B86 B0010000		mov eax,dword ptr ds:[esi+0x1B0]		EBP 0012F94C
00421BC7	E8 2CFCFEFF		call <Dope2112.Controls::TControl::GetTe	获取序列号	ESI 01151CBC
00421BCC	8B45 FC		mov eax,[local.1]		EDI 01154CAC
00421BCF	E8 9017FEFF		call <Dope2112.__linkproc__ LStrLen>	获取用户名长度	EIP 00421BD7 Dope2112.00
00421BD4	8845 EF		mov byte ptr ss:[ebp-0x11],al		C 0 ES 0023 32位 0(FFFF)
00421BD7	807D EF 06		cmp byte ptr ss:[ebp-0x11],0x6	比较用户名长度是否大于等于6	P 0 CS 001B 32位 0(FFFF)
00421BDB	73 15		jmp short <Dope2112.loc_421BF2>		A 0 SS 0023 32位 0(FFFF)
00421BDD	8B86 C0010000		mov eax,dword ptr ds:[esi+0x1C0]		Z 0 DS 0023 32位 0(FFFF)
00421BE3	BA A81D4200		mov edx,<Dope2112.aDerNameMussMin>	Der Name muss min. 6 Zeichen lang sein	S 0 FS 003B 32位 7FFDE0
00421BE8	E8 3BFCFEFF		call <Dope2112.sub_411828>		T 0 GS 0000 NULL
00421BED	E9 72010000		jmp <Dope2112.loc_421D64>		D 0
00421BF2	33C0		xor eax,eax	loc_421BF2	0 0 LastErr ERROR_SUCCE
00421BF4	33D2		xor edx,edx	loc_421BF4	
00421BF6	9AD0		mov esi,esi	loc_421BF6	

至此基础校验部分结束，开始计算注册码。真正的注册码分为两部分，首先解释第一部分

第一部分

```

{
    i = 0;                                // i初始化为零
    do
    {
        switch ( *(unsigned __int8 *)(username_1 + i - 1) )// 根据username[i]的值给v13赋值
        {
            case 0x61u:
                v13 = 0x18;
                break;
            case 0x62u:
                v13 = 0x25;
                break;
            case 0x63u:
                v13 = 0x42;
                break;
            case 0x64u:
                v13 = 0xC;
                break;
            case 0x65u:
                v13 = 0xD;
                break;
            case 0x66u:
                v13 = 6;
                break;
            case 0x67u:
                v13 = 0x36;
                break;
            case 0x68u:
                v13 = 0x2B;
                break;
            case 0x69u:
                v13 = 0x17;
                break;
            case 0x6Au:
                v13 = 0x2F;
                break;
            case 0x6Bu:
                v13 = 0x13;
                break;
            case 0x6Cu:
                v13 = 0x82u;
                break;
            case 0x6Du:
                v13 = 0x9Bu;
                break;
            case 0x6Eu:
                v13 = 0x92u;
                break;
            case 0x6Fu:
                v13 = 3;
                break;
            case 0x70u:
                break;
            144          break;
            145          default:
            146              v13 = 0x5D;
            147              break;
            148          }
            149          TotalSum += v13;                // 循环累加v13的值
            150          ++i;
            151          }
            152          while ( i != 6 );                // 循环6次

```

第一部分的算法根据用户名的ASCII值循环六次计算出来一个结果，算法和IDA反汇编出来的伪代码是一致的，这里就不多做说明了。

第二部分

接下来是第二部分，这一部分在IDA反汇编的伪代码中是没有的。

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00421CE3	> B2 5D	mov dl,0x5D	loc_421CE3; Default case of switch 00421C00	EAX 00000007
00421CE5	> 02DA	add bl,dl	bl=TotalSum	ECX 011559C4 ASCII "guishou"
00421CE7	. 40	inc eax	i++	EDX 0012F93C
00421CE8	. 3C 06	cmp al,0x6	if(i==6)	EBX 000000F8
00421CEA	. ^ 0F85 04FFFFFF	jnz <Dope2112.loc_421BF4>		ESP 0012F918
00421CF0	. 8D55 F0	lea edx,[local.4]		EBP 0012F94C
00421CF3	. 33C0	xor eax,eax		ESI 01151C8C
00421CF5	. 8A45 EF	mov al,byte ptr ss:[ebp-0x11]	al=usernameLen	EDI 01154CAC
00421CF8	. 69C0 7E4A0000	imul eax,eax,0x4A7E	eax=usernameLen*0x4A7E	EIP 00421CF8 Dope2112.00421C
00421CFE	. E8 7136FEFF	call <Dope2112.Sysutils::IntToStr(int)>	将int转为字符串	C 0 ES 0023 32位 0(FFFFFFFF)
00421D03	. 8D55 E4	lea edx,[local.7]		P 1 CS 001B 32位 0(FFFFFFFF)
00421D06	. 33C0	xor eax,eax		A 0 SS 0023 32位 0(FFFFFFFF)
00421D08	. 8AC3	mov al,bl		7 1 DS 0023 32位 0(FFFFFFFF)
00421D0A	. E8 6536FEFF	call <Dope2112.Sysutils::IntToStr(int)>	将TotalSum转为字符串	
00421D0F	. FF75 E4	push [local.7]		
00421D12	. 68 D81D4200	push <Dope2112.dword_421DD8>	-	
00421D17	. FF75 F0	push [local.4]		
00421D1A	. 8D45 F4	lea eax,[local.3]		
00421D1D	. BA 03000000	mov edx,0x3		
00421D22	. E8 FD16FEFF	call <Dope2112.System::__linkproc__ LStr		
00421D27	. 8D55 E8	lea edx,[local.6]		

首先计算出用户名长度乘以0x4A7E的结果，

00421CF5	. 8A45 EF	mov al,byte ptr ss:[ebp-0x11]	al=usernameLen
00421CF8	. 69C0 7E4A0000	imul eax,eax,0x4A7E	eax=usernameLen*0x4A7E
00421CFE	. E8 7136FEFF	call <Dope2112.Sysutils::IntToStr(int)>	将int转为字符串
00421D03	. 8D55 E4	lea edx,[local.7]	
00421D06	. 33C0	xor eax,eax	
00421D08	. 8AC3	mov al,bl	
00421D0A	. E8 6536FEFF	call <Dope2112.Sysutils::IntToStr(int)>	将TotalSum转为字符串
00421D0F	. FF75 E4	push [local.7]	
00421D12	. 68 D81D4200	push <Dope2112.dword_421DD8>	-
00421D17	. FF75 F0	push [local.4]	
00421D1A	. 8D45 F4	lea eax,[local.3]	
00421D1D	. BA 03000000	mov edx,0x3	
00421D22	. E8 FD16FEFF	call <Dope2112.System::__linkproc__ LStr	
00421D27	. 8D55 E8	lea edx,[local.6]	

堆栈地址=0012F930
dx=00000000

地址	数值	注释	地址	数值	注释
0012F93C	011559F0	ASCII "133490"	0012F918	0012FA2C	指向下一
0012F940	00000000		0012F91C	00421D0F	SE处理程

接着将计算的结果转为字符串

00421D08	. 8AC3	mov al,bl	
00421D0A	. E8 6536FEFF	call <Dope2112.Sysutils::IntToStr(int)>	将TotalSum转为字符串
00421D0F	. FF75 E4	push [local.7]	
00421D12	. 68 D81D4200	push <Dope2112.dword_421DD8>	-
00421D17	. FF75 F0	push [local.4]	
00421D1A	. 8D45 F4	lea eax,[local.3]	
00421D1D	. BA 03000000	mov edx,0x3	
00421D22	. E8 FD16FEFF	call <Dope2112.System::__linkproc__ LStr	
00421D27	. 8D55 E8	lea edx,[local.6]	

堆栈 ss:[0012F930]=01155A04, (ASCII "248")

地址	数值	注释	地址	数值
0012F930	01155A04	ASCII "248"	0012F918	0012FA2C

然后再将第一部分用户名计算的结果转为字符串

校验部分

00421D1A	. 8D45 F4	lea eax,[local.3]		D 0
00421D1D	. BA 03000000	mov edx,0x3		0 0 LastE
00421D22	. E8 FD16FEFF	call <Dope2112.System::__linkproc__ LStrCatN(void)>		EFL 000002
00421D27	. 8D55 E8	lea edx,[local.6]		ST0 empty
00421D2A	. 8B86 B0010000	mov eax,dword ptr ds:[esi+0x1B0]		ST1 empty
00408342H=<Dope2112.System::__linkproc__ LStrCatN(void)>				
0012F930	01155A04	ASCII "248"	0012F90C	011559F0 ASCII "133490"
0012F934	01154928	ASCII "GuiShou"	0012F910	00421DD8 UNICODE "-"
0012F938	07000000		0012F914	01155A04 ASCII "248"

将两个部分的注册码拼接

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
00421C9E	. E8 7136FEFF	call <Dope2112.Sysutils::IntToStr(int)>	将int转为字符串	EAX 01155A14 ASCII "248-133490"
00421D03	. 8D55 E4	lea edx,[local.7]		ECX 75EBC5FE ASCII "23-75EBC5FE"
00421D06	. 33C0	xor eax,eax		EDX 01155A2C ASCII "123456789"
00421D08	. 8AC3	mov al,bl		EBX 000000F8
00421D0A	. E8 6536FEFF	call <Dope2112.Sysutils::IntToStr(int)>	将TotalSum转为字符串	ESP 0012F918
00421D0F	. FF75 E4	push [local.7]		EBP 0012F94C
00421D12	. 68 D81D4200	push <Dope2112.word_421D008>	-	ESI 01151CBC
00421D17	. FF75 F0	push [local.4]		EDI 01154CAC
00421D1A	. 8D45 F4	lea eax,[local.3]		EIP 00421D3B Dope2112.00421D3B
00421D1D	. BA 03000000	mov edx,0x3		C 0 ES 0023 32位 0(FFFFFFFF)
00421D22	. E8 FD16FEFF	call <Dope2112.System::__linkproc__ LStrCatN(void)>		P 1 CS 001B 32位 0(FFFFFFFF)
00421D27	. 8D55 E8	lea edx,[local.6]		A 0 SS 0023 32位 0(FFFFFFFF)
00421D2A	. 8B86 B0010000	mov eax,dword ptr ds:[esi+0x1B0]		Z 0 DS 0023 32位 0(FFFFFFFF)
00421D30	. E8 C3FAFEFF	call <Dope2112.Controls::TControl::GetText(void)>	获取输入的序列号	S 0 FS 003B 32位 7FFDE000(FFF)
00421D35	. 8B55 E8	mov edx,[local.6]		D 0
00421D38	. 8B45 F4	mov eax,[local.3]		T 0 GS 0000 NULL
00421D3B	. E8 3417FEFF	call <Dope2112.System::__linkproc__ LStrCmp(void)>	将输入的序列号和拼接的字符串进行比较	D 0
00421D40	. 75 12	jnz short <Dope2112.loc_421D54>		O 0 LastErr ERROR_SUCCESS (00000000)
00421D42	. 8B86 C0010000	mov eax,dword ptr ds:[esi+0x1C0]		EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)
00421D48	. BA E41D4200	mov edx,<Dope2112.aHeyDuHastEsGes>	Hey du hast es geschafft !	ST0 empty 0.0
00421D4D	. E8 D6FAFEFF	call <Dope2112.sub_411828>		ST1 empty 0.0

接着将输入的序列号和两部分拼接的字符串进行比较，根据比较的结果提示是否注册成功

写出注册机

这个注册机也是好写的不行啊，直接把IDA的代码拷下来稍微改改就行了，代码如下：

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

int main()
{
    char username[20] = { 0 };
    printf("请输入用户名:");
    scanf_s("%s", username, 20);
    int usernameLen = strlen(username);
    if (usernameLen<6)
    {
        printf("用户名长度必须大于等于6");
    }
    //大写转小写
    for (int i=0;i<usernameLen;i++)
    {
        if (username[i]>='A'&&username[i]<='Z')
        {
            username[i] += 32;
        }
    }

    int v13 = 0;
    int TotalSum = 0;
    int i = 0; // i初始化为零
    do
    {
        switch ((username[i- 1]))// 根据username[i]的值给v13赋值
        {
            case 0x61u:
                v13 = 0x18;
                break;
            case 0x62u:
                v13 = 0x25;
```

```
        break;
    case 0x63u:
        v13 = 0x42;
        break;
    case 0x64u:
        v13 = 0xC;
        break;
    case 0x65u:
        v13 = 0xD;
        break;
    case 0x66u:
        v13 = 6;
        break;
    case 0x67u:
        v13 = 0x36;
        break;
    case 0x68u:
        v13 = 0x2B;
        break;
    case 0x69u:
        v13 = 0x17;
        break;
    case 0x6Au:
        v13 = 0x2F;
        break;
    case 0x6Bu:
        v13 = 0x13;
        break;
    case 0x6Cu:
        v13 = 0x82u;
        break;
    case 0x6Du:
        v13 = 0x9Bu;
        break;
    case 0x6Eu:
        v13 = 0x92u;
        break;
    case 0x6Fu:
        v13 = 3;
        break;
    case 0x70u:
        v13 = 0x63;
        break;
    case 0x71u:
        v13 = 0x21;
        break;
    case 0x72u:
        v13 = 0x42;
        break;
    case 0x73u:
        v13 = 0x5C;
        break;
    case 0x74u:
```

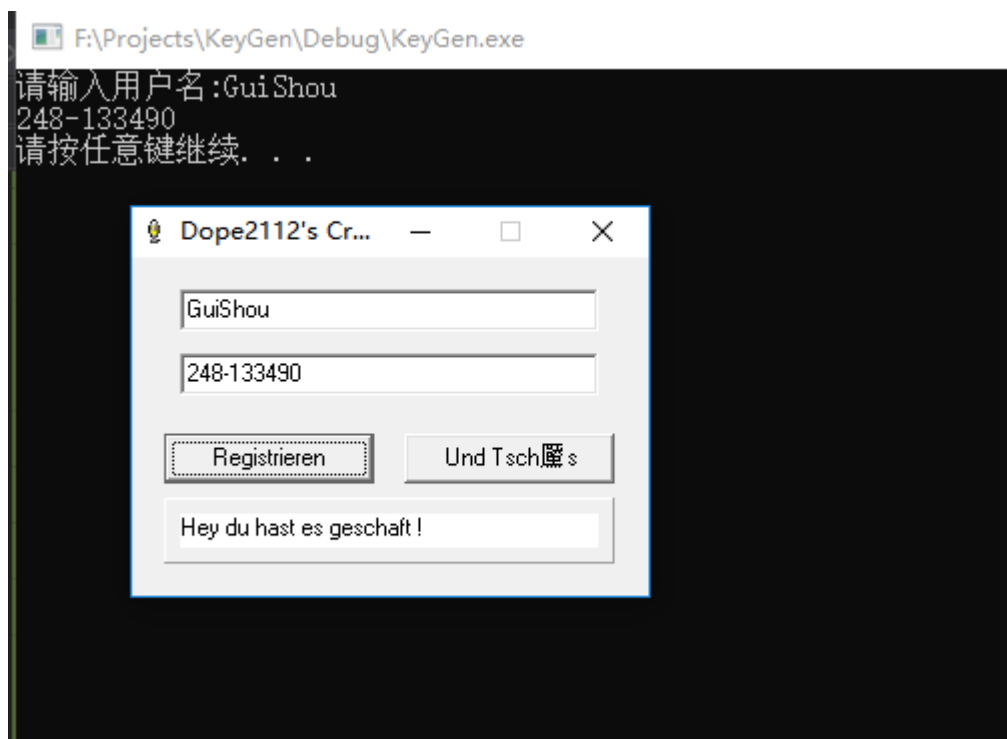
```

        v13 = 0x29;
        break;
    case 0x75u:
        v13 = 0xc7u;
        break;
    case 0x76u:
        v13 = 0x66;
        break;
    case 0x77u:
        v13 = 0x58;
        break;
    case 0x78u:
        v13 = 0xA;
        break;
    case 0x79u:
        v13 = 0x28;
        break;
    case 0x7Au:
        v13 = 0x50;
        break;
    default:
        v13 = 0x5D;
        break;
    }
    TotalSum += v13;                // 循环累加v13的值
    ++i;
    //这里记得超出范围清掉高位
    if (TotalSum>0xFF)
    {
        TotalSum &= 0x00FF;
    }
} while (i != 6);

printf("%d-%d\n", TotalSum, 0x4A7E*usernameLen);
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

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