

查壳

寻找突破口

分析算法

第一部分

第二部分

写出注册机

校验结果

【软件名称】：damn.exe

【软件大小】：30KB

【下载地址】：<https://github.com/TonyChen56/160-Crackme>

【加壳方式】：Aspack

【保护方式】：Name/Serial

【编译语言】：汇编

【调试环境】：W10 64

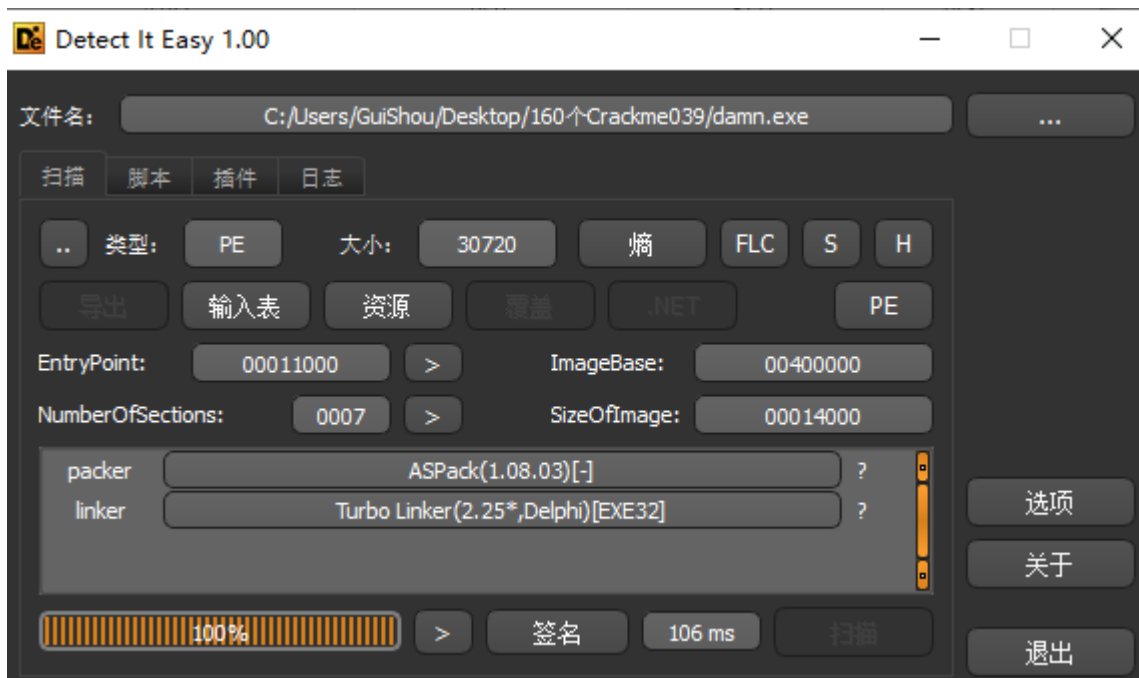
【使用工具】：OD

【破解日期】：2019-05-04

【破解目的】：纯属兴趣

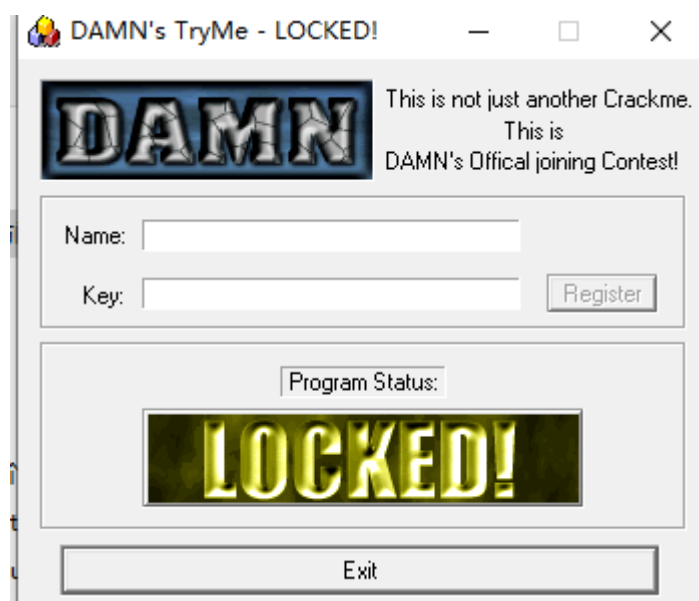
## 查壳

---



这个目标程序加了个Aspack的壳，用脱壳机或者ESP定律手动脱都行。

## 寻找突破口



这个程序按钮是被禁用的，所以不太好找突破口。

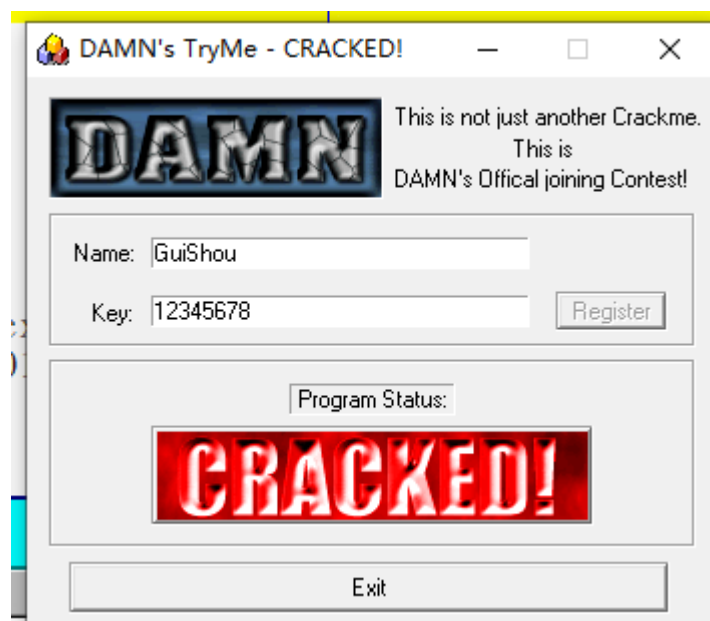
地址	HEX 数据	反汇编	注释
00401000	6A 00	push 0x0	pModule = NULL
00401002	E8 28040000	call <jmp.&kernel32.#OpenSemaphoreW_632>	GetModuleHandleA
00401007	BF 0B234000	mov edi, unpacked.0040230B	
0040100C	85C0	test eax, eax	
0040100E	74 1E	je short unpacked.0040102E	
00401010	A3 95234000	mov dword ptr ds:[0x402395], eax	
00401015	8307 01	add dword ptr ds:[edi], 0x1	
00401018	33C0	xor eax, eax	
0040101A	50	push eax	lParam = 000DFFCC
0040101B	68 45104000	push unpacked.00401045	DlgProc = unpacked.00401045
00401020	50	push eax	hOwner = 000DFFCC
00401021	6A 73	push 0x73	pTemplate = 0x73
00401023	FF35 95234000	push dword ptr ds:[0x402395]	hInst = NULL
00401029	E8 B3030000	call <jmp.&user32.#1692>	DialogBoxParamA
0040102E	6A 00	push 0x0	ExitCode = 0x0
00401030	E8 F4030000	call <jmp.&kernel32.#GetFirmwareEnvironm>	ExitProcess
00401035	20	db 20	CHAR ' '
00401036	20	db 20	CHAR ' '

直接载入程序，发现是汇编写的。汇编写的程序不会有编译器生成的无用代码，所以可以一直往下拉，找到关键代码处，当然也可以在GetDlgItemTextA函数下断。

地址	HEX 数据	反汇编	注释
00401129	68 EB030000	push 0x3EB	ControlID = 3EB (1003.)
0040112E	FF35 91234000	push dword ptr ds:[0x402391]	hWnd = NULL
00401134	E8 C6020000	call <jmp.&user32.#1838>	GetDlgItemTextA
00401139	A3 89234000	mov dword ptr ds:[0x402389], eax	获取用户名及输入的长度
0040113E	33C0	xor eax, eax	
00401140	5E	pop esi	kernel32.74F70179
00401141	5F	pop edi	kernel32.74F70179
00401142	5B	pop ebx	kernel32.74F70179
00401143	C9	leave	
00401144	C2 1000	ret 0x10	
00401147	6A 22	push 0x22	Count = 22 (34.)
00401149	68 21234000	push unpacked.00402321	Buffer = unpacked.00402321
0040114E	68 EC030000	push 0x3EC	ControlID = 3EC (1004.)
00401153	FF35 91234000	push dword ptr ds:[0x402391]	hWnd = NULL
00401159	E8 A1020000	call <jmp.&user32.#1838>	GetDlgItemTextA
0040115E	A3 8D234000	mov dword ptr ds:[0x40238D], eax	获取输入的序列号和长度
00401163	E8 8B010000	call unpacked.004012F3	关键算法
00401168	50	push eax	Enable = TRUE

一直往下拉，你会发现有两个获取用户输入的函数，分别是获取用户名和序列号。下面那个函数就是关键的算法函数了。

## 分析算法



随便输入一个用户名，开始分析算法

## 第一部分

地址	HEX	数据	反汇编	注释
004012F3	90		nop	
004012F4	8B0D	89234000	mov ecx,dword ptr ds:[0x402389]	ecx用户名长度
004012FA	85C9		test ecx,ecx	unpacked. <ModuleEntryPoint>
004012FC	74 71		jg short unpacked.0040136F	用户名不为空
004012FE	49		dec ecx	长度-1
004012FF	8BF1		mov esi,ecx	unpacked. <ModuleEntryPoint>
00401301	BF 53234000		mov edi,unpacked.00402353	edi=username
00401306	BB 4E4D4144		mov ebx,0x44414D4E	ebx=0x44414D4E
0040130B	33D2		xor edx,edx	unpacked. <ModuleEntryPoint>
0040130D	8BCA		mov ecx,edx	unpacked. <ModuleEntryPoint>
0040130F	33C0		xor eax,eax	
00401311	8A040F		mov al,byte ptr ds:[edi+ecx]	al=username[i]
00401314	03D0		add edx,eax	用户名ASCII值累加
00401316	D1CB		ror ebx,1	ebx循环右移1位
00401318	D3CB		ror ebx,cl	ebx循环右移i位
0040131A	33DA		xor ebx,edx	ebx异或username[i]
0040131C	3BCE		cmp ecx,esi	比较循环次数
0040131E	74 03		jg short unpacked.00401323	

这个算法分为两个部分，第一个部分是对用户名进行处理，算出一个数。这个部分没什么好说的，直接看注释，无非是将用户名的ASCII值经过各种运算计算出来一个数而已。

## 第二部分

00401320	41	inc ecx	i++
00401321	EB EC	jmp short unpacked.0040130F	
00401323	81CB 10101010	or ebx,0x10101010	ebx^0x10101010
00401329	87DA	xchg edx,ebx	edx=用户名计算的结果
0040132B	BF 21234000	mov edi,unpacked.00402321	edi=序列号
00401330	8B0D 8D234000	mov ecx,dword ptr ds:[0x40238D]	ecx=序列号长度
00401336	83F9 08	cmp ecx,0x8	序列号长度必须为8位
00401339	75 34	jnz short unpacked.0040136F	
0040133B	33C9	xor ecx,ecx	ecx清零
0040133D	33C0	xor eax,eax	eax清零
0040133F	C1C2 08	rol edx,0x8	用户名计算的结果循环右移8位
00401342	8AC2	mov al,dl	
00401344	8AD8	mov bl,al	

程序首先会将第一部分用户名计算的结果和0x10101010进行异或，然后比较序列号的长度是否为8位，不是则报错。

接着将用户名计算的结果循环右移8位，然后拆分为高位和低位，

文件(F) 查看(V) 调试(T) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools Breakpoint->					
暂停					
l e m t w h c p k b r j ... s					
地址 HEX 数据 反汇编 注释					
00401329	>	81CB 10101010	or ebx,0x10101010	ebx 0x10101010	EAX 000000D5
00401329	.	87DA	xchg edx,ebx	edx=用户名计算的结果	ECX 00000000
0040132B	.	BF 21234000	mov edi,unpacked.00402321	edi=序列号	EDX 5E14D4D5
00401330	.	8B0D 8D234000	mov ecx,dword ptr ds:[0x40238D]	ecx=序列号长度	EBX 000002D5
00401336	.	83F9 08	cmp ecx,0x8	序列号长度必须为8位	ESP 000DF490
00401339	.	75 34	jnz short unpacked.0040136F		EBP 000DF4A0
0040133B	.	33C9	xor ecx,ecx	ecx清零	ESI 00000006
0040133D	>	33C0	xor eax,eax	eax清零	EDI 00402321 ASCII "12
0040133F	.	C1C2 08	rol edx,0x8	用户名计算的结果循环右移8位	EIP 00401346 unpacked.
00401342	.	8AC2	mov al,dl		C 1 ES 002B 32位 0(FF
00401344	.	8AD8	mov bl,al		P 1 CS 0023 32位 0(FF
00401346	.	24 0F	and al,0xF	将al的第二位清零 只保留个位	A 0 SS 002B 32位 0(FF
00401348	.	C0EB 04	shr bl,0x4		Z 1 DS 002B 32位 0(FF
0040134B	.	80E3 0F	and bl,0xF	以下四步为转十六进制为ASCII	S 0 FS 0053 32位 38B0
0040134E	.	3C 0A	cmp al,0xA		T 0 GS 002B 32位 0(FF
00401350	.	1C 69	sbb al,0x69		D 0
00401352	.	2F	das		0 0 LastErr ERROR_SUC
a1=05					
地址 HEX 数据 ASCII 地址 数值 注释					

HEX	数据	反汇编	注释	寄存器 (FPU)
>	81CB 10101010	or ebx,0x10101010	ebx 0x10101010	EAX 00000005
.	87DA	xchg edx,ebx	edx=用户名计算的结果	ECX 00000000
.	BF 21234000	mov edi,unpacked.00402321	edi=序列号	EDX 5E14D4D5
.	8B0D 8D234000	mov ecx,dword ptr ds:[0x40238D]	ecx=序列号长度	EBX 000002D5
.	83F9 08	cmp ecx,0x8	序列号长度必须为8位	ESP 000DF490
√	75 34	jnz short unpacked.0040136F		EBP 000DF4A0
.	33C9	xor ecx,ecx	ecx清零	ESI 00000006
>	33C0	xor eax,eax	eax清零	EDI 00402321 ASCII "123"
.	C1C2 08	rol edx,0x8	用户名计算的结果循环右移8位	EIP 00401348 unpacked.0
.	8AC2	mov al,dl		C 0 ES 002B 32位 0(FFF)
.	8AD8	mov bl,al		P 1 CS 0023 32位 0(FFF)
.	24 0F	and al,0xF	将al的第二位清零 只保留个位	A 0 SS 002B 32位 0(FFF)
.	C0EB 04	shr bl,0x4		Z 0 DS 002B 32位 0(FFF)
.	80E3 0F	and bl,0xF	以下四步为转十六进制为ASCII	S 0 FS 0053 32位 38B00
.	3C 0A	cmp al,0xA		T 0 GS 002B 32位 0(FFF)
.	1C 69	sbb al,0x69		D 0
.	2F	das		0 0 LastErr ERROR_SUCC

然后将al和0xF进行与运算，清掉十位，保留个位->5

调试器 - unpacked.exe - [x86] - 主线程, 调试 - unpacked.exe					
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-»					
[Icons] [					

将保留位转成ASCII值->35



```

void ROR(DWORD * myd, unsigned char n)
{
    __asm
    {
        MOV EBX, myd;
        MOV eax, [EBX];
        MOV cl, n;
        ROR EAX, cl;
        MOV[EBX], eax;
    }
}

int main()
{
    char *username;
    unsigned long serial;
    serial = 0x44414D4E;
    unsigned long nTemp = 0;
    username = new char[260];
    memset(username, 0, 260);
    printf("请输入用户名: ");
    scanf_s("%s", username, 260);
    for (int i = 0; i < strlen(username); i++)
    {
        nTemp += username[i];
        ROR(&serial, 1);
        ROR(&serial, i);
        serial ^= nTemp;
    }

    serial |= 0x10101010;
    delete[] username;

    printf("%x\n", serial);
    system("pause");
    return 0;
}

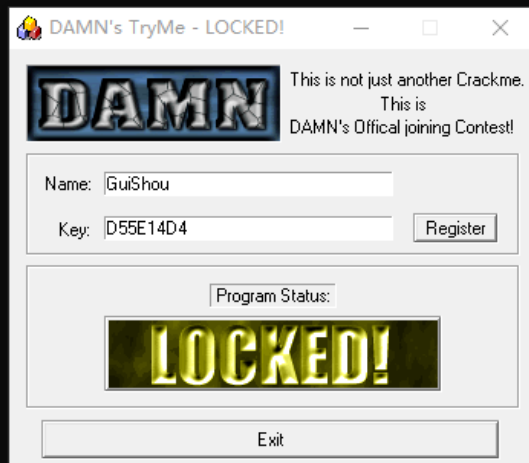
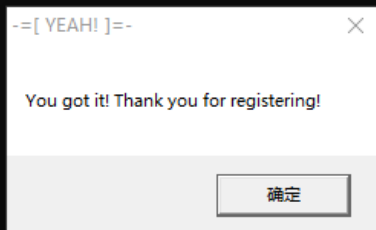
```

## 校验结果

输入用户名和计算的序列号，提示注册成功

E:\Projects\KeyGen\Debug\KeyGen.exe

请输入用户名: GuiShou  
d55e14d4  
请按任意键继续. . .



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>