



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH

KHOA: CÔNG NGHỆ THÔNG TIN

BỘ MÔN: KIẾN TRÚC MÁY TÍNH & HỢP NGỮ

BÁO CÁO ĐỒ ÁN

ĐỒ ÁN 3: CRACK PHẦN MỀM

Mục lục

Tổng quan	2
Thông tin đồ án.....	2
Phân công	2
Kiến thức cần có để giải quyết.....	2
1.1.....	3
Manh mối	3
Đoạn mã phát sinh khóa của chương trình.....	7
Test case	10
Phân tích mã assemble.....	7
1.2.....	11
Manh mối	11
Phân tích mã assemble.....	11
1.3.....	12
Manh mối	14
Phân tích mã assemble.....	18
1.5.....	19
Manh mối	19

Phân tích mã assemble.....	20
Thuật toán phát sinh key	30
Tổng kết.....	30
Mức độ hoàn thành	31
Tham khảo	31

I. Tổng quan

1. Phân công

Tên	MSSV	Công việc
Huỳnh Lâm Tứ	1712864	-1.1 -1.3 -1.5 - Làm báo cáo
Hồng Quang Tú	1712855	-1.2 - Làm báo cáo
Đoàn Nhật Trường	1712851	- Hỗ trợ - Làm báo cáo

- Công việc gồm: viết báo cáo và tạo keygen (xxx.cpp, xxx.exe) (nếu có) của bài crackme được phân công.

2. Kiến thức cần có để giải quyết

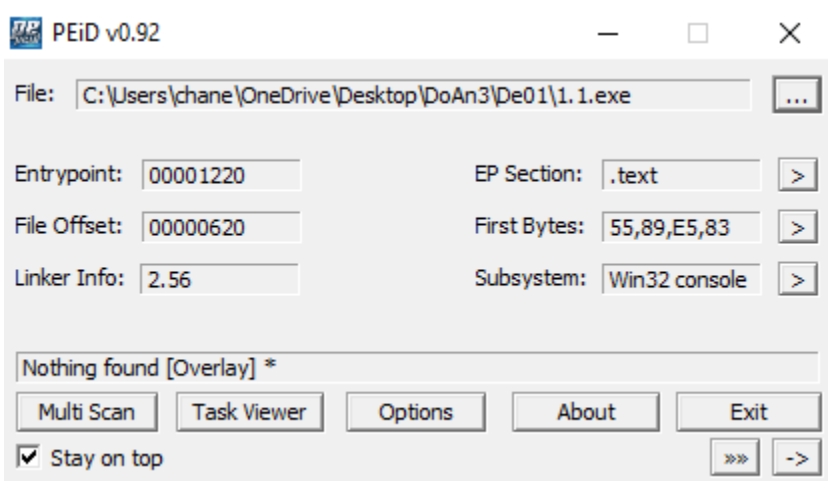
1. Kỹ năng kiểm tra bằng **PEiD**: kiểm tra file có pack không, kiểm tra có dấu vết của mã hóa trong file thực thi không, unpack nếu cần.
2. Kỹ năng debug trong **OllyDbg**: đặt breakpoint F2, F9 để bắt đầu debug, F8 để chạy từng lệnh, F7 để đi vào bên trong hàm chạy từng lệnh; tìm chuỗi với Search for > All referenced strings; truy cập đến mọi vùng nhớ bằng Memory map; kỹ năng xử lý địa chỉ của vùng nhớ stack, dump.
3. Đọc hiểu các lệnh hợp ngữ: nhóm lệnh tính toán (ADD, SUB, IMUL, XOR, OR, AND), nhóm lệnh so sánh (TEST, CMP), nhóm lệnh nhảy (JE, JAE, JNE, JNZ...), nhóm lệnh gán (MOV-gán giá trị, LEA-gán địa chỉ).

1.1

Manh mối

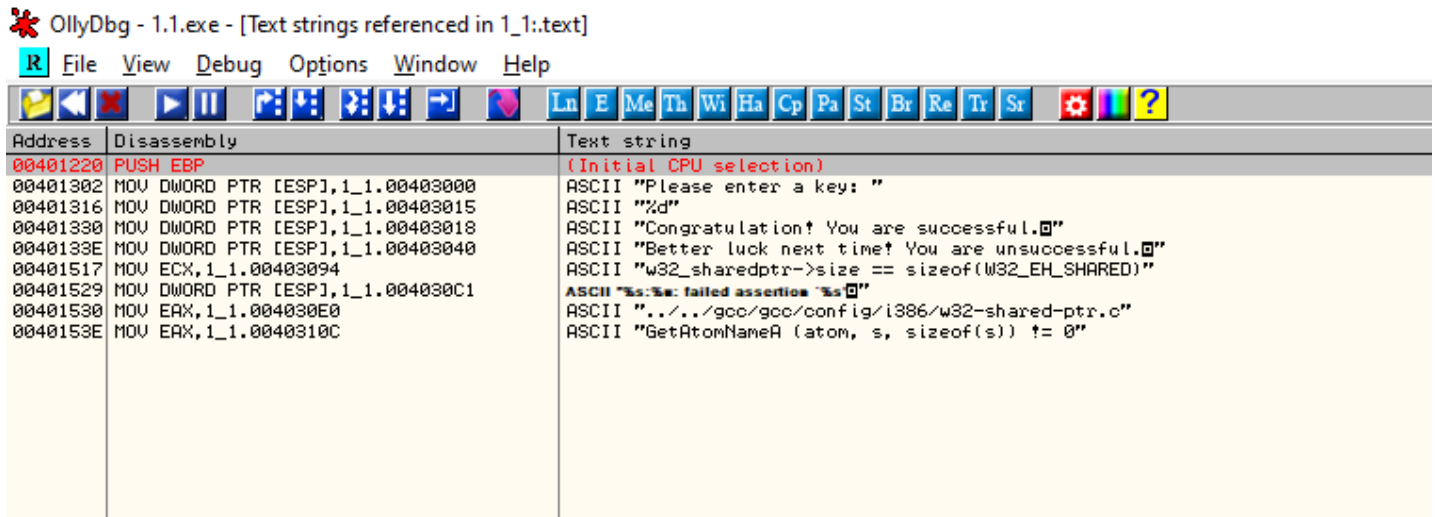
Trước hết, phân tích mã hợp ngữ phải trên file thực thi không bị pack. Pack là thao tác của người lập trình để nén chương trình. Việc này ngăn chặn những nỗ lực đọc hiểu mã nguồn hợp ngữ của các cracker. Nếu chủ quan không kiểm tra xem file bị pack hay không mà trực tiếp phân tích, kết quả phân tích sẽ không chính xác, thậm chí không phân tích được do nhiều dữ kiện quan trọng bị ẩn mất.

- Đầu tiên, **Scan with PEiD 1.1.exe** để kiểm tra tệp thực tin này đã pack chưa.

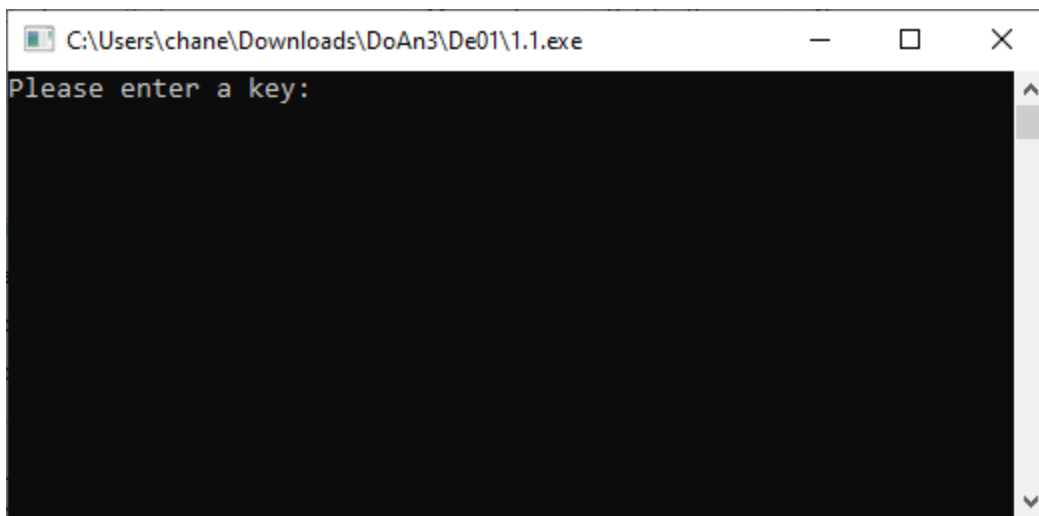


- Kết quả là **Nothing found ***, **1.1.exe** không bị pack, tiến hành phân tích mã hợp ngữ bằng **OllyDbg**
- Chạy **OllyDbg**, mở 1.1.exe

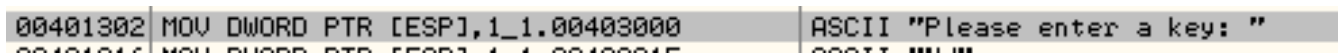
- Một cửa sổ mới xuất hiện, chứa nhiều chuỗi, nhưng những chuỗi được hiển thị trong hình cần chú ý:



- Xét với khi chạy chương trình 1.1.exe:



- Ta thấy chương trình bắt đầu chạy từ dòng lệnh 00401302:



* OllyDbg - 1.1.exe - [Text strings referenced in 1_1:text]

Address	Disassembly	Text string
00401220	PUSH EBP	(Initial CPU selection)
00401302	MOV DWORD PTR [ESP],1_1.00403000	ASCII "Please enter a key: "
00401316	MOV DWORD PTR [ESP],1_1.00403015	ASCII "%d"
00401330	MOV DWORD PTR [ESP],1_1.00403018	ASCII "Congratulation! You are successful.☐"
0040133E	MOV DWORD PTR [ESP],1_1.00403040	ASCII "Better luck next time! You are unsuccessful.☐"
00401517	MOV ECX,1_1.00403094	ASCII "w32_sharedptr->size == sizeof(W32_EH_SHARED)"
00401529	MOV DWORD PTR [ESP],1_1.004030C1	ASCII "%s:%s: failed assertion '%s'☐"
00401530	MOV EAX,1_1.004030E0	ASCII ".../..../gcc/gcc/config/i386/w32-shared-ptr.c"
0040153E	MOV EAX,1_1.0040310C	ASCII "GetAtomNameA (atom, s, sizeof(s)) != 0"

- Nháy đúp chuột để truy xuất vào dòng lệnh trên trong chương trình:

* OllyDbg - 1.1.exe - [CPU - main thread, module 1_1]

Address	Disassembly	Comment
0040120B	83EC 18 SUB ESP,18	
004012DE	83E4 F0 AND ESP,FFFFFFF0	
004012E1	B8 00000000 MOV EAX,0	
004012E6	83C0 0F ADD EAX,0F	
004012E9	83C0 0F ADD EAX,0F	
004012EC	C1E8 04 SHR EAX,4	
004012EF	C1E8 04 SHL EAX,4	
004012F2	8945 FC MOV DWORD PTR [EBP-4],EAX	
004012F5	8B45 FC MOV EAX,DWORD PTR [EBP-4]	
004012F8	E8 A3040000 CALL 1_1.004017A0	
004012FD	E8 3E010000 CALL 1_1.00401440	
00401302	C70424 00304 MOV DWORD PTR [ESP],1_1.00403000	ASCII "Please enter a key: "
00401309	E8 A2050000 CALL <JMP.&msvcrt.printf>	printf
0040130E	C74424 04 10 MOV DWORD PTR [ESP+4],1_1.00404010	
00401316	C70424 15304 MOV DWORD PTR [ESP],1_1.00403015	ASCII "%d"
0040131D	E8 7E050000 CALL <JMP.&msvcrt scanf>	scanf
00401322	E8 69FFFFFF CALL 1_1.00401290	
00401327	833D 1040400 CMP DWORD PTR [404010],1	
0040132E	75 0E JNZ SHORT 1_1.0040133E	
00401330	C70424 18304 MOV DWORD PTR [ESP],1_1.00403018	ASCII "Congratulation! You are successful.☐"
00401337	E8 74050000 CALL <JMP.&msvcrt.printf>	printf
0040133C	EB 0C JMP SHORT 1_1.0040134A	
0040133E	C70424 40304 MOV DWORD PTR [ESP],1_1.00403040	ASCII "Better luck next time! You are unsuccessful.☐"
00401345	E8 66050000 CALL <JMP.&msvcrt.printf>	printf
0040134A	E8 D1040000 CALL <JMP.&msvcrt._getch>	_getch
0040134F	B8 00000000 MOV EAX,0	
00401354	C9 LEAVE	
00401355	C3 RET	
00401356	90 NOP	
00401357	90 NOP	
00401358	90 NOP	
00401359	90 NOP	
0040135A	90 NOP	
0040135B	90 NOP	

- Ta thấy rằng sau lệnh Scanf để nhập input từ người dùng vào, chương trình nhảy tới dòng lệnh có địa chỉ 00401290

Address	Disassembly	Comment
004012F8	E8 A3040000 CALL 1_1.004017A0	
004012FD	E8 3E010000 CALL 1_1.00401440	
00401302	C70424 00304 MOV DWORD PTR [ESP],1_1.00403000	ASCII "Please enter a key: "
00401309	E8 A2050000 CALL <JMP.&msvcrt.printf>	printf
0040130E	C74424 04 10 MOV DWORD PTR [ESP+4],1_1.00404010	
00401316	C70424 15304 MOV DWORD PTR [ESP],1_1.00403015	ASCII "%d"
0040131D	E8 7E050000 CALL <JMP.&msvcrt scanf>	scanf
00401322	E8 69FFFFFF CALL 1_1.00401290	
00401327	833D 1040400 CMP DWORD PTR [404010],1	
0040132E	75 0E JNZ SHORT 1_1.0040133E	
00401330	C70424 18304 MOV DWORD PTR [ESP],1_1.00403018	ASCII "Congratulation! You are successful.☐"
00401337	E8 74050000 CALL <JMP.&msvcrt.printf>	printf
0040133C	EB 0C JMP SHORT 1_1.0040134A	
0040133E	C70424 40304 MOV DWORD PTR [ESP],1_1.00403040	ASCII "Better luck next time! You are unsuccessful.☐"
00401345	E8 66050000 CALL <JMP.&msvcrt.printf>	printf
0040134A	E8 D1040000 CALL <JMP.&msvcrt._getch>	_getch

- Truy cập vào dòng lệnh địa chỉ 00401290 ta có thể bước đầu đoán đây là đoạn mã tạo key cho chương trình:

OllyDbg - 1.1.exe - [CPU - main thread, module 1_1]

File View Debug Options Window Help

Ln E Me Th Ws Ha Cp Pa St Br Re Tr Sx

```

00401284  E9 D7020000 JMP 1_1.00401560
00401289  90             NOP
0040128A  90             NOP
0040128B  90             NOP
0040128C  90             NOP
0040128D  90             NOP
0040128E  90             NOP
0040128F  90             NOP
00401290  55             PUSH EBP
00401291  . 89E5         MOV EBP,ESP
00401293  . 83EC 04      SUB ESP,4
00401296  . C745 FC C800 MOV DWORD PTR [EBP-4],0C8
0040129D  . 8B55 FC      MOV EDX,DWORD PTR [EBP-4]
004012A0  . 89D0         MOV EAX,EDX
004012A2  . C1E0 02      SHL EAX,2
004012A5  . 01D0         ADD EAX,EDX
004012A7  . 8945 FC      MOV DWORD PTR [EBP-4],EAX
004012AA  . 8D45 FC      LEA EAX,DWORD PTR [EBP-4]
004012AD  . 8330 64      XOR DWORD PTR [EAX],64
004012B0  . 8D45 FC      LEA EAX,DWORD PTR [EBP-4]
004012B3  . F710         NOT DWORD PTR [EAX]
004012B5  . A1 10404000 MOV EAX,DWORD PTR [404010]
004012BA  . 3B45 FC      CMP EAX,DWORD PTR [EBP-4]
004012BD  . 75 0C        JNZ SHORT 1_1.004012CB
004012BF  . C705 10404000 MOV DWORD PTR [404010],1
004012C9  . EB 0A        JMP SHORT 1_1.004012D5
004012CB  > C705 10404000 MOV DWORD PTR [404010],0
004012D5  . C9           LEAVE
004012D6  . C3           RET
004012D7  90             NOP
004012D8  55             PUSH EBP
004012D9  . 89E5         MOV EBP,ESP
004012DB  . 83EC 18      SUB ESP,18
004012DE  . 8B55 FC      MOV EDX,DWORD PTR [EBP-4]

```

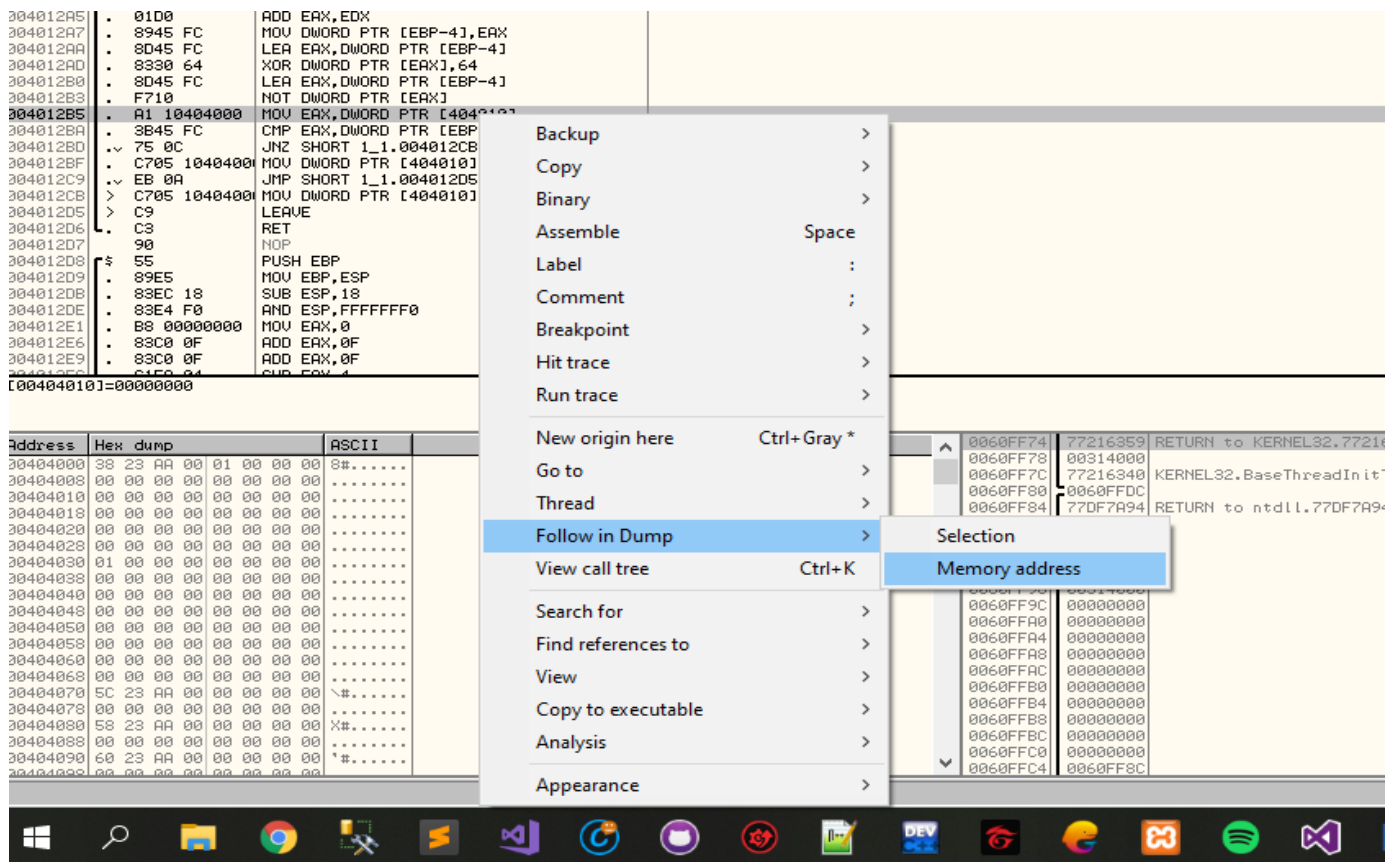
Local call from 00401322

- Ta đi vào phân tích đoạn mã trên:

00401290	55	PUSH EBP	Truy cập vào con trỏ stack EBP
00401291	. 89E5	MOV EBP,ESP	Trỏ EBP đến stack ESP
00401293	. 83EC 04	SUB ESP,4	Dịch chuyển con trỏ ESP
00401296	. C745 FC C800	MOV DWORD PTR [EBP-4],0C8	Gán giá trị tại stack [EBP-4] = 0C8
0040129D	. 8B55 FC	MOV EDX,DWORD PTR [EBP-4]	Gán EDX = [EBP-4], khi đó EDX = 0C8
004012A0	. 89D0	MOV EAX,EDX	Gán EAX = EDX, khi đó EAX = 0C8
004012A2	. C1E0 02	SHL EAX,2	Dịch trái EAX 2 đơn vị => EAX = 320
004012A5	. 01D0	ADD EAX,EDX	EAX += EDX = 3E8
004012A7	. 8945 FC	MOV DWORD PTR [EBP-4],EAX	[EBP-4] = 3E8
004012AA	. 8D45 FC	LEA EAX,DWORD PTR [EBP-4]	*EAX = *[EBP - 4], trỏ [EBP - 4] đến EAX, khi EAX bị thay đổi giá trị thì [EBP - 4] cũng sẽ thay đổi sang giá trị tương ứng với EBP
004012AD	. 8330 64	XOR DWORD PTR [EAX],64	EAX xor 64 = 38C => [EBP - 4] = 38C
004012B0	. 8D45 FC	LEA EAX,DWORD PTR [EBP-4]	*EAX = *[EBP - 4], trỏ [EBP - 4] đến EAX, khi EAX bị thay đổi giá trị thì [EBP - 4] cũng sẽ thay đổi sang giá trị tương ứng với EBP
004012B3	. F710	NOT DWORD PTR [EAX]	Not EAX = FFFFC73 => [EBP - 4] = FFFFC73

- Ta xét dòng lệnh tiếp theo 004012B5 . A1 10404000 MOV EAX,DWORD PTR [404010]

- Ở đây ta không biết địa chỉ nhớ [404010] chứa gì, ta thử truy cập địa chỉ nhớ [404010] để xem bằng cách nhấn chuột phải vào dòng lệnh -> Follow in Dump -> Memory address



- Không thấy thông tin gì từ địa chỉ này

Address	Hex dump	ASCII
00404000	38 23 AA 00 01 00 00 00	8#.....
00404008	00 00 00 00 00 00 00 00
00404010	00 00 00 00 00 00 00 00
00404018	00 00 00 00 00 00 00 00
00404020	00 00 00 00 00 00 00 00
00404028	00 00 00 00 00 00 00 00
00404030	01 00 00 00 00 00 00 00
00404038	00 00 00 00 00 00 00 00
00404040	00 00 00 00 00 00 00 00
00404048	00 00 00 00 00 00 00 00
00404050	00 00 00 00 00 00 00 00
00404058	00 00 00 00 00 00 00 00
00404060	00 00 00 00 00 00 00 00
00404068	00 00 00 00 00 00 00 00
00404070	5C 23 AA 00 00 00 00 00	\#.....
00404078	00 00 00 00 00 00 00 00
00404080	58 23 AA 00 00 00 00 00	X#.....
00404088	00 00 00 00 00 00 00 00
00404090	60 23 AA 00 00 00 00 00	*#.....
00404098	00 00 00 00 00 00 00 00

- Dự đoán rằng có thể đây là địa chỉ lưu thông tin key nhập vào và đã bị xóa trước khi chương trình kết thúc => Ta tiến hành kiểm tra dự đoán trên:
- Thực hiện tạo break point bằng phím F2 tại dòng lệnh
004012B5 | . A1 10404000 MOV EAX,DWORD PTR [404010]
- Sau đó nhấn F9 để chạy chương trình, nhập thử key là **123**

Snipping mode

```

0040128C . 90 NOP
0040128D . 90 NOP
0040128E . 90 NOP
0040128F . 55 PUSH EBP
00401290 . 89E5 MOV EBP,ESP
00401291 . 83EC 04 SUB ESP,4
00401292 . C745 FC C800 MOV DWORD PTR [EBP-4],0C8
00401293 . 8B55 FC MOV EDX,DWORD PTR [EBP-4]
00401294 . 89D0 MOV EAX,EDX
00401295 . C1E0 02 SHL EAX,2
00401296 . 01D0 ADD EAX,EDX
00401297 . 8945 FC MOV DWORD PTR [EBP-4],EAX
00401298 . 8D45 FC LEA EAX,DWORD PTR [EBP-4]
00401299 . 8330 64 XOR DWORD PTR [EAX],64
0040129A . 8D45 FC LEA EAX,DWORD PTR [EBP-4]
0040129B . F710 NOT DWORD PTR [EAX]
0040129C . A1 10404000 MOV EAX,DWORD PTR [404010]
0040129D . 3B45 FC CMP EAX,DWORD PTR [EBP-4]
0040129E . 75 0C JNZ SHORT 1_1.004012CB
0040129F . C705 10404000 MOV DWORD PTR [404010],1
004012A0 . EB 0A JMP SHORT 1_1.004012D5
004012A1 . C705 10404000 MOV DWORD PTR [404010],0
004012A2 . C9 LEAVE
004012A3 . C3 RET
004012A4 . 90 NOP
004012A5 . 55 PUSH EBP
004012A6 . 89E5 MOV EBP,ESP
004012A7 . 83EC 18 SUB ESP,18
004012A8 . 83E4 F0 AND ESP,FFFFFFF0
004012A9 . B8 00000000 MOV EAX,0
004012AA . 8B00 MOV ECX,0

```

DS:[00404010]=0000007B
EAX=0060FEF4

Address	Hex dump	ASCII
00402000	FF FF FF FF 00 00 00 00
00402008	00 00 00 00 00 00 00 00
00402010	00 40 00 00 00 00 00 00	..@.....
00402018	00 00 00 00 00 00 00 00
00402020	00 19 40 00 00 00 00 00	..@.....
00402028	00 00 00 00 00 00 00 00
00402030	00 00 00 00 FF FF FF FF
00402038	00 00 00 00 FF FF FF FF
00402040	00 00 00 00 00 00 00 00
00402048	00 00 00 00 00 00 00 00
00402050	00 00 00 00 00 00 00 00
00402058	00 00 00 00 00 00 00 00
00402060	00 00 00 00 00 00 00 00
00402068	00 00 00 00 00 00 00 00
00402070	00 00 00 00 00 00 00 00

0060FEF4 FFFFC73
0060FEF8 00401327 RETURN to 1_1.00401327 from 1_1.004
0060FF00 00403015 ASCII "%d"
0060FF04 00404010 1_1.00404010
0060FF08 00761898
0060FF0C 004012FD RETURN to 1_1.004012FD from 1_1.004
0060FF10 0060FF58
0060FF14 00000035
0060FF18 00000008
0060FF1C 0000002D
0060FF20 0000002D
0060FF24 00000010
0060FF28 0060FF60
0060FF2C 004011E7 RETURN to 1_1.004011E7 from 1_1.004
0060FF30 00000001
0060FF34 00000000

- Sau đó truy cập địa chỉ nhớ **[404010]** để xem bằng cách nhấn chuột phải vào dòng lệnh -> Follow in Dump -> Memory address

```

004012A2 . C1E0 02 SHL EAX,2
004012A5 . 01D0 ADD EAX,EDX
004012A7 . 8945 FC MOV DWORD PTR [EBP-4],EAX
004012A8 . 8D45 FC LEA EAX,DWORD PTR [EBP-4]
004012A9 . 8330 64 XOR DWORD PTR [EAX],64
0040129A . 8D45 FC LEA EAX,DWORD PTR [EBP-4]
0040129B . F710 NOT DWORD PTR [EAX]
0040129C . A1 10404000 MOV EAX,DWORD PTR [404010]
0040129D . 3B45 FC CMP EAX,DWORD PTR [EBP-4]
0040129E . 75 0C JNZ SHORT 1_1.004012CB
0040129F . C705 10404000 MOV DWORD PTR [404010],1
004012A0 . EB 0A JMP SHORT 1_1.004012D5
004012A1 . C705 10404000 MOV DWORD PTR [404010],0
004012A2 . C9 LEAVE
004012A3 . C3 RET
004012A4 . 90 NOP

```

DS:[00404010]=0000007B
EAX=0060FEF4

Address	Hex dump	ASCII
00404000	38 23 75 00 01 00 00 00	8#u....
00404008	00 00 00 00 00 00 00 00
00404010	7B 00 00 00 00 00 00 00
00404018	00 00 00 00 00 00 00 00
00404020	00 00 00 00 00 00 00 00
00404028	00 00 00 00 00 00 00 00
00404030	01 00 00 00 00 00 00 00
00404038	00 00 00 00 00 00 00 00
00404040	00 00 00 00 00 00 00 00
00404048	00 00 00 00 00 00 00 00
00404050	00 00 00 00 00 00 00 00
00404058	00 00 00 00 00 00 00 00
00404060	00 00 00 00 00 00 00 00

C:\Users\chane\Downloads\DoAn3\De01\1.1.exe
Please enter a key: 123

- Ta thấy địa chỉ nhớ **[404010]** lưu giá trị **7B**, đúng bằng giá trị của key **123** vừa nhập trong hệ 16.

- Suy ra địa chỉ nhớ [404010] chính là để lưu giá trị key nhập vào.
- Dòng lệnh [004012B5] . A1 10404000 MOV EAX,DWORD PTR [404010] có nghĩa là lưu giá trị của key vừa nhập vào EAX
- Tiếp tục xét các dòng lệnh tiếp theo

[004012BA] . 3B45 FC CMP EAX,DWORD PTR [EBP-4]	So sánh EAX với [EBP-4], ở đây EAX là key vừa nhập và [EBP-4] đang mang giá trị FFFFFFF3
[004012BD] . 75 0C JNZ SHORT 1_1.004012CB	Nếu EAX = [EBP-4] thì chuyển đến dòng lệnh 004012CB
[004012BF] . C705 10404000 MOV DWORD PTR [404010],1	Gán [404010] = 1
[004012C9] . EB 0A JMP SHORT 1_1.004012D5	Nhảy đến dòng lệnh 004012D5
[004012CB] > C705 10404000 MOV DWORD PTR [404010],0	Gán [404010] = 0
[004012D5] > C9 LEAVE	Thoái khỏi hàm con này
[004012D6] L C3 RET	Trở về hàm chính

- Ta xét trường hợp nếu key nhập vào bằng [EBP-4] đang mang giá trị FFFFFFF3, khi đó sẽ gán [404010] = 1 và chương trình quay về nhảy đến dòng lệnh [00401327]

004012FD . E8 3E010000 CALL 1_1.00401440	
00401302 . C70424 00304 MOV DWORD PTR [ESP],1_1.00403000	ASCII "Please enter a key: "
00401309 . E8 A2050000 CALL <JMP.&msvrt.printf>	printf
0040130E . C74424 04 10 MOV DWORD PTR [ESP+4],1_1.00404010	ASCII "%d"
00401316 . C70424 15304 MOV DWORD PTR [ESP],1_1.00403015	scanf
0040131D . E8 7E050000 CALL <JMP.&msvrt.scanf>	
00401322 . E8 69FFFFFF CALL 1_1.00401290	
00401327 . 833D 10404000 CMP DWORD PTR [404010],1	
0040132E . 75 0E JNZ SHORT 1_1.0040133E	
00401330 . C70424 18304 MOV DWORD PTR [ESP],1_1.00403018	ASCII "Congratulation! You are successful."
00401337 . E8 74050000 CALL <JMP.&msvrt.printf>	printf
0040133C . EB 0C JMP SHORT 1_1.0040134A	
0040133E . C70424 40304 MOV DWORD PTR [ESP],1_1.00403040	ASCII "Better luck next time! You are unsuccessful."
00401345 . E8 66050000 CALL <JMP.&msvrt.printf>	printf
0040134A . E8 D1040000 CALL <JMP.&msvrt._getch>	_getch
0040134F . B8 00000000 MOV EAX,0	
00401354 . C9 LEAVE	
00401355 . C3 RET	
00401356 . 90 NOP	

- Thực hiện so sánh [404010] với 1, nghĩa là nếu thấy [404010] = 1 thì key vừa nhập sẽ bằng biến [EBP-4] đang mang giá trị FFFFFFF3, khi đó theo như câu lệnh tiếp theo ở dòng [0040132E], chương trình sẽ nhảy đến thực hiện in ra màn hình "Congratulation!" báo hiệu nhập đúng key
- Kết luận: giá trị phát sinh cho biến [EBP-4] chính là key của chương trình, bằng FFFFFFF3 trong hệ 16 và bằng -909 trong hệ 10. **Key là -909**
- Kiểm tra...

```

C:\Users\chane\Downloads\DoAn3\De01\1.1.exe
Please enter a key: -909
Congratulation! You are successful.

```

- Thành công!!

1.2

Manh mối

- Đầu tiên ta mở chương trình OllyDbg lên và mở file 1.2.
- Nhấn chuột phải chọn **Search for -> All referenced text strings**

```
(Initial CPU selection)
ASCII "About"
ASCII "Musical Crackme"Rules: - No debugging - No patching - Make a keygenwt@vrenr@gmail.com"
```

- Ta thấy có dòng chữ "Make a keygen..."=> có vẻ như chương trình có keygen.
- Quay lại chương trình, nhấn chuột phải chọn Search for -> All Intermodular calls, tìm đến **GetDlgItemInt**

00401121	. 6A 01	PUSH 1	IsSigned = TRUE
00401123	. 6A 00	PUSH 0	pSuccess = NULL
00401125	. 68 EC030000	PUSH 3EC	ControlID = 3EC (1004.)
0040112A	. FF75 08	PUSH DWORD PTR [EBP+8]	hWnd
0040112D	. E8 D8000000	CALL <JMP.&user32.GetDlgItemInt>	GetDlgItemInt
00401132	. 8BD8	MOV EBX,EAX	
00401134	. 33C0	XOR EAX,EAX	
00401136	. 6A 01	PUSH 1	IsSigned = TRUE
00401138	. 6A 00	PUSH 0	pSuccess = NULL
0040113A	. 68 ED030000	PUSH 3ED	ControlID = 3ED (1005.)
0040113F	. FF75 08	PUSH DWORD PTR [EBP+8]	hWnd
00401142	. E8 C3000000	CALL <JMP.&user32.GetDlgItemInt>	GetDlgItemInt
00401147	. 8BC8	MOV ECX,EAX	
00401149	. E8 4DFFFFFF	CALL 1_2.0040109B	
0040114E	. 3BC3	CMP EAX,EBX	
00401150	. 74 0C	JE SHORT 1_2.0040115E	
00401152	. BB 00000000	MOV EBX,0	
00401157	. BA 00000000	MOV EDI,0	

- Ta thấy ở dòng lệnh **00401142** khi nhấn đúp chuột vào sẽ đưa ta quay lại màn hình chính nơi mà **GetDlgItemInt** được gọi
- Tiến hành tạo Break Point tại đây

00401132	. 8BD8	MOV EBX,EAX	
00401134	. 33C0	XOR EAX,EAX	
00401136	. 6A 01	PUSH 1	IsSigned = TRUE
00401138	. 6A 00	PUSH 0	pSuccess = NULL
0040113A	. 68 ED030000	PUSH 3ED	ControlID = 3ED (1005.)
0040113F	. FF75 08	PUSH DWORD PTR [EBP+8]	hWnd
00401142	. E8 C3000000	CALL <JMP.&user32.GetDlgItemInt>	GetDlgItemInt
00401147	. 8BC8	MOV ECX,EAX	
00401149	. E8 4DFFFFFF	CALL 1_2.0040109B	
0040114E	. 3BC3	CMP EAX,EBX	
00401150	~ 74 0C	JE SHORT 1_2.0040115E	
00401152	. BB 00000000	MOV EBX,0	
00401157	. BA 00000000	MOV EDX,0	
0040115C	~ EB 53	JMP SHORT 1_2.004011B1	
0040115E	> 68 EB030000	PUSH 3EB	ControlID = 3EB (1003.)
00401163	. FF75 08	PUSH DWORD PTR [EBP+8]	hWnd

- Nhập chạy thử



- Thấy ID được lưu vào thanh ghi EBX, Serial được lưu vào thanh ghi EAX và ECX

Registers (FPU)	
EAX	0000007B
ECX	0000007B
EDX	0019F836
EBX	00000064
ESP	0019F90C
EBP	0019F918
ESI	008605F6
EDI	8006010

- Để ý kĩ hơn ta thấy rằng ở dưới có dòng lệnh 00401149 gọi tới một dòng lệnh khác

00401149 . E8 4DFFFFFF CALL 1_2.0040109B

- Truy cập đến dòng lệnh này:

0040109B	. 83C3 4C	ADD EBX,4C
0040109E	. 83C2 03	ADD EDX,3
004010A1	. 43	INC EBX
004010A2	. 81C3 8B030000	ADD EBX,38B
004010A8	. 03DB	ADD EBX,EBX
004010AA	. 0FAFDA	IMUL EBX,EDX
004010AD	. 4B	DEC EBX
004010AE	. C3	RET

- Ta nghi ngờ rằng đây là đoạn mã phát sinh khóa
- Tiến hành phân tích đoạn mã trên:

ADD EBX, 4C: EBD += 76

ADD EDX, 3: $EDX += 3$

INC EBX: $EBX++$

ADD EBX, 38B: $EBX += 907$

ADD EBX, EBX: $EBX += EBX$

IMUL EBX, EDX: $EBX = EBX * EDX$

DEC EBX: $EBX--$

- Kết quả cho thấy $EBX = (EBX + 76 + 1 + 907) * 2 * 3 - 1$
- Để xem kết quả này có liên quan với ID không ta chạy thử với ID = 1234



Kết quả

```

EAX 0000162E
ECX 0000162E
EDX 00000003
EBX 0000051E
ESP 0018FB70
```

- Ta thấy rằng $EBX = 51E$, tức là $EBX = 1310 = 1234 + 76$
- Có nghĩa là EBX lúc đầu chính là ID
- Thử kết quả với Serial là EBX vừa tìm được ta thấy chương trình hoàn toàn thực hiện được



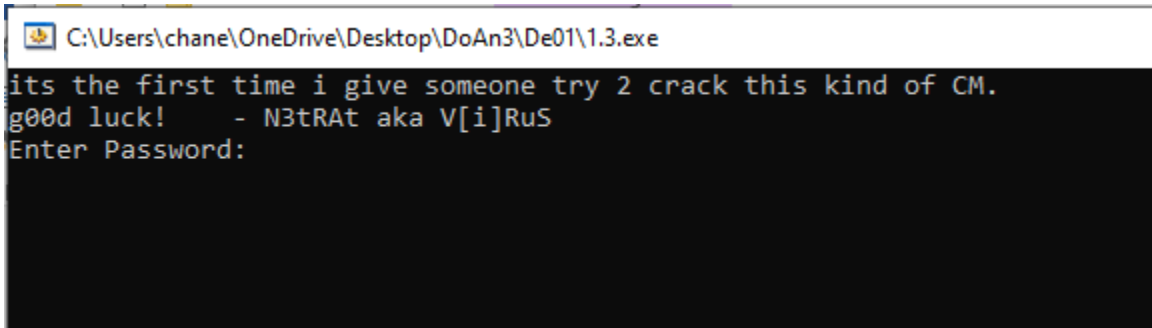
⇒ Thành công!!!!

Manh mối

- Chuỗi in ra trong console không xuất hiện trong vùng nhớ chương trình, đúng hơn là vùng nhớ không thể truy xuất khi chưa chạy chương trình.

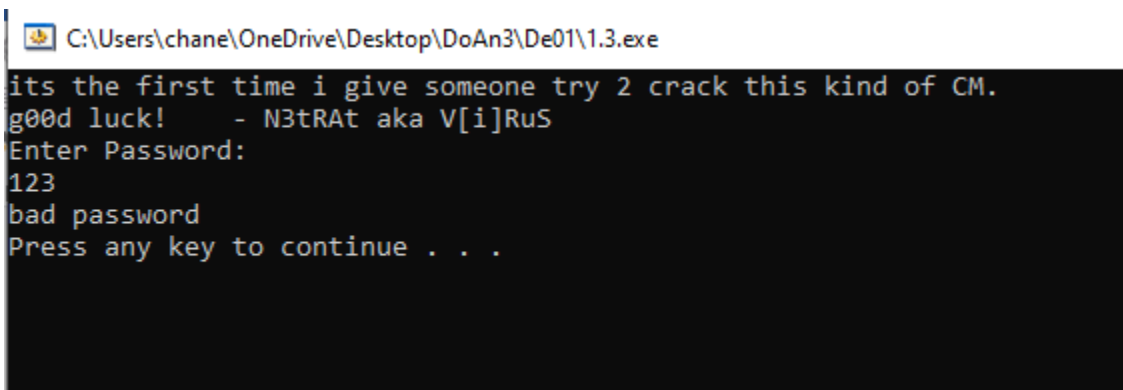
• `004195F3 . E8 98CBFEFF CALL <JMP.&kernel32.WaitForSingleObject> | LWaitForSingleObject`

- Sau khi chạy dòng lệnh 004195F3, màn hình console xuất một lượt như sau:



```
C:\Users\chane\OneDrive\Desktop\DoAn3\De01\1.3.exe
its the first time i give someone try 2 crack this kind of CM.
good luck! - N3tRat aka V[i]RuS
Enter Password:
```

- Nhập từ bàn phím password thử, kết quả là:



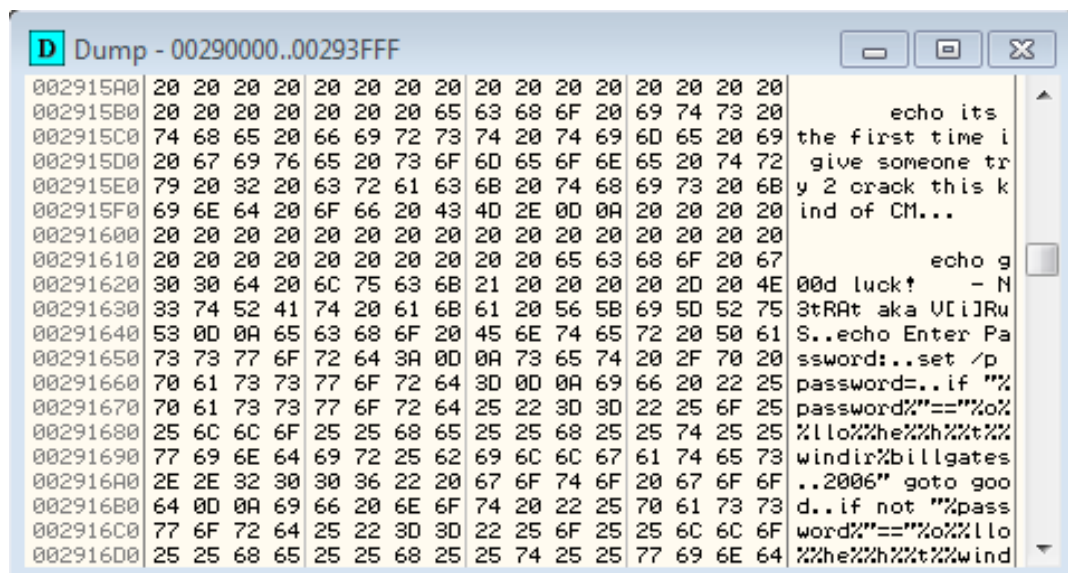
```
C:\Users\chane\OneDrive\Desktop\DoAn3\De01\1.3.exe
its the first time i give someone try 2 crack this kind of CM.
good luck! - N3tRat aka V[i]RuS
Enter Password:
123
bad password
Press any key to continue . . .
```

- Bằng cách nhấn **chuột phải tại Disassembler Window > Search for > All referenced text strings**, các string được tìm thấy không hề chứa những string được hiển thị từ console.

- Ta tiến hành tìm kiếm từng vùng, phát hiện vùng nhớ này có điều đặc biệt:

00290000	00004000			Priv 00021004	RW	
----------	----------	--	--	---------------	----	--

- Chuỗi hiện thị ở console đã được tìm thấy:



- Tuy nhiên, khi **Restart (Ctrl + F12)**, rồi tìm đến **Memmmory map** thì vùng nhớ đặc biệt này đã không thể truy cập tới nữa.

00220000	00001000			Priv 00021004	RW	RW	
002A0000	00007000			Priv 00021004	RW	RW	

- Chọn lân cận với vùng nhớ đặc biệt nhưng cũng không tìm thấy vùng nhớ đặc biệt.
- Trong quá trình debug, khi dò tìm ta phát hiện có một stack đã lưu địa chỉ của vùng nhớ dump, nơi đang lưu chuỗi: C:\Users\chane\AppData\Local\Temp\

EBP-180	00000234	hObject = 00000234 (window)
EBP-17C	FFFFFFFF	Timeout = INFINITE
EBP-178	0019FE04	Pointer to next SEH record
EBP-174	004196ED	SE handler
EBP-170	0019FF70	
EBP-16C	0019FF64	Pointer to next SEH record
EBP-168	0041971F	SE handler
EBP-164	0019FF70	
EBP-160	002B9000	
EBP-15C	00000000	
EBP-158	00000000	
EBP-154	00000000	
EBP-150	021E0A38	ASCII "C:\Users\chane\AppData\Local\Temp\"
EBP-14C	3A43222D	
EBP-140	4C79EEEC	

- Một Stack khác lưu chuỗi: C:\Users\chane\AppData\Local\Temp\bt.7273.bat

EBP-1B0	0041E8C0	1_3.0041E8C0
EBP-1AC	00000000	
EBP-1A8	0019FF70	
EBP-1A4	004195E7	1_3.004195E7
EBP-1A0	00000000	
EBP-19C	021E154C	ASCII "cmd.exe /c C:\Users\chane\AppData\Local\Temp\bt7273.bat "C:\Users\chane\Downloads\DoAn3\De01\1.3.exe""
EBP-198	00000000	
EBP-194	00000000	
EBP-190	00000000	
EBP-18C	00000020	
EBP-188	00000000	

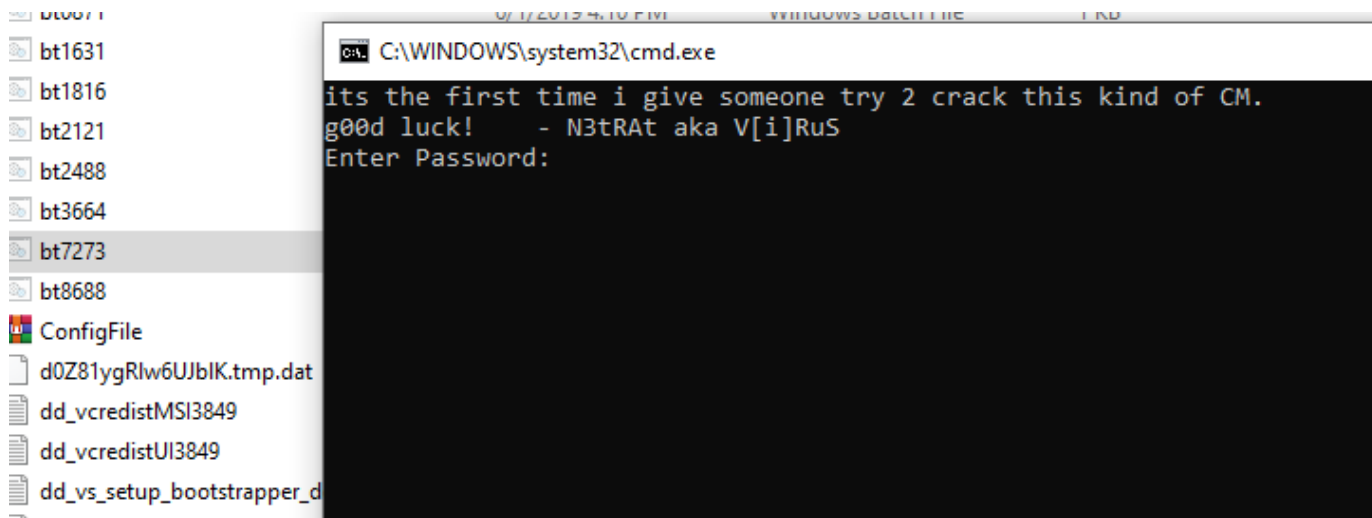
- Follow vào hàm <JMP.&kernel32.WaitForSingleObject> (sau khi chạy hàm này console in ra chuỗi), có nhiều hàm đáng nghi: **KERNEL32.WriteFile**, **user32.LoadString** [2]

MOV EAX,EAX	
JMP DWORD PTR [&kernel32.WaitForSingleObject]	KERNEL32.WaitForSingleObject
MOV EAX,EAX	
JMP DWORD PTR [&kernel32.WriteFile]	KERNEL32.WriteFile
MOV EAX,EAX	
JMP DWORD PTR [&user32.CharNextA]	user32.CharNextA
MOV EAX,EAX	
JMP DWORD PTR [&user32.CharPrevA]	user32.CharPrevA
MOV EAX,EAX	
JMP DWORD PTR [&user32.CharToOemA]	user32.CharToOemA
MOV EAX,EAX	
JMP DWORD PTR [&user32.CharUpperBuffA]	user32.CharUpperBuffA
MOV EAX,EAX	

- Ta có thể bước đầu suy ra, chuỗi in ra trong console có thể liên quan đến nội dung của file: C:\Users\chane\AppData\Local\Temp\bt.7273.bat
- Tìm đến file path này, bt7273.bat thực sự tồn tại:

<div> <div>← → ↕ ↑</div> <div> <div> <div></div> <div>This PC</div> </div> <div> <div>Local Disk (C:)</div> <div>Users</div> <div>chane</div> <div>AppData</div> <div>Local</div> <div>Temp</div> </div> </div> </div>				
Name	Date modified	Type	Size	
asc12_startupBlack	6/11/2018 3:57 PM	Data Base File	9 KB	
asc12_startupWhite	1/14/2019 10:42 AM	Data Base File	30 KB	
BdDSi#KYH)2)sex9.tmp.dat	6/5/2019 3:53 PM	DAT File	1 KB	
bt0871	6/1/2019 4:10 PM	Windows Batch File	1 KB	
bt1631	6/6/2019 12:27 PM	Windows Batch File	1 KB	
bt1816	6/5/2019 4:33 PM	Windows Batch File	1 KB	
bt2121	6/8/2019 10:40 PM	Windows Batch File	1 KB	
bt2488	6/8/2019 10:33 PM	Windows Batch File	1 KB	
bt3664	6/5/2019 4:04 PM	Windows Batch File	1 KB	
bt7273	6/8/2019 10:40 PM	Windows Batch File	1 KB	
bt8688	6/6/2019 12:57 PM	Windows Batch File	1 KB	
ConfigFile	6/6/2019 4:56 PM	WinRAR ZIP archive	303 KB	
d0Z81ygRlw6UJbIk.tmp.dat	6/8/2019 6:11 PM	DAT File	1 KB	

- Khi mở file này, command line hiện ra giống khi chạy chương trình 1.3.exe:



- Đến lúc này ta có thể suy đoán File bat này chính là được tạo từ chương trình 1.2.exe, và nó xử lý những gì chúng ta cho là 1.3.exe đang làm:
- Thử mở code của file bat:

```

C:\Users\chane\AppData\Local\Temp\bt7273.bat - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
change.log httpd-ssl.conf wp-config.php blacklist.conf bt0871.bat bt7273.bat
1  @shift 1
2  ECHO OFF
3  cls
4  REM title crackme - Batch or not?
5  set r=o
6  set o=t
7  set llo=he
8  set t=y
9  set h=u
10 set j=w
11 set he=llo
12
13          echo its the first time i give someone try 2 crack this kind of CM.
14          echo g00d luck! - N3tRat aka V[i]RuS
15 echo Enter Password:
16 set /p password=
17 if "%password%"=="%o%%llo%he%%h%%t%%windir%billgates..2006" goto good
18 if not "%password%"=="%o%%llo%he%%h%%t%%windir%billgates..2006" goto bad
19 :good
20 echo good password
21 pause
22 exit
23 :bad
24 echo bad password
25 pause
26 exit
27

```

- Ta thấy có một đoạn kiểm tra password, tiến hành phân tích đoạn code này
- Ta thấy file này trong 1.3.exe file này có thể chạy bằng cmd, nên có thể thử xử lý để tìm ra chuỗi khớp với password bằng cmd.

- Dán đoạn code sau vào cmd > nhấn Enter

```
set r=o
set o=t
set llo=he
set t=y
set h=u
set j=w
set he=llo
echo %o%%llo%%he%%h%%t%%windir%billgates..2006
```

- Được kết quả: thellouyC:\WINDOWSbillgates..2006

C:\WINDOWS\system32\cmd.exe


```
Microsoft Windows [Version 10.0.18362.116]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\chane>set r=o
C:\Users\chane>set o=t
C:\Users\chane>set llo=he
C:\Users\chane>set t=y
C:\Users\chane>set h=u
C:\Users\chane>set j=w
C:\Users\chane>set he=llo

C:\Users\chane>echo %o%%llo%%he%%h%%t%%windir%billgates..2006
thellouyC:\WINDOWSbillgates..2006

C:\Users\chane>
```

- Thử nhập kết quả này vào chương trình 1.3.exe => thành công!!

 C:\Users\chane\Downloads\DoAn3\De01\1.3.exe

```
its the first time i give someone try 2 crack this kind of CM.  
g00d luck!      - N3tRAt aka V[i]RuS  
Enter Password:  
thellouyC:\WINDOWSbillgates..2006  
good password  
Press any key to continue . . .
```

1.5

Manh mối

- Trước tiên, tìm kiếm những chuỗi được 1.5.exe hiển thị.
- **Nhấn chuột phải tại Disassembler Window > Search for > All referenced text strings**
- Ta thấy có dòng 004014E2 hiển thị thông báo khi người dùng nhập chuỗi rỗng, truy cập vào dòng này...

OlllyDbg - 1.5.exe - [Text strings referenced in 1_5:.text]

File View Debug Options Window Help

Address	Disassembly	Text string
0040107A	PUSH 1_5.00403291	ASCII "A1"
00401080	ASCII "Ribbere 1.42 (Re"	
0040108D	ASCII "gistered)",0	
004010C9	ASCII "Ribbere 1.42 (Un"	
004010D9	ASCII "registered)",0	
004010E5	PUSH 1_5.004010C9	ASCII "Ribbere 1.42 (Unregistered)"
0040116A	ASCII "Enter the about "	
0040117A	ASCII "box.",0	
0040117F	PUSH 1_5.0040116A	ASCII "Enter the about box."
0040119A	ASCII "Exit please",0	
004011A6	PUSH 1_5.0040119A	ASCII "Exit please"
004011E8	PUSH 1_5.00403068	ASCII "Tooltips_class32"
00401233	ASCII "About",0	
00401239	PUSH 1_5.00401233	ASCII "About"
00401254	ASCII "Exit",0	
00401259	PUSH 1_5.00401254	ASCII "Exit"
0040127C	PUSH 1_5.00403294	ASCII "IDD_BUTTONDLG"
004012E3	PUSH 1_5.004032AC	ASCII "(Unregistered)"
00401403	PUSH 1_5.004032A2	ASCII "IDD_ABOUT"
004014A0	CALL <JMP.&user32.GetDlgItemTextA>	(Initial CPU selection)
004014AE	ASCII "Your name must b"	
004014BE	ASCII "e at least one b"	
004014CE	ASCII "yter",0	
004014D5	ASCII "Error",0	
004014DD	PUSH 1_5.004014D5	ASCII "Error"
004014E2	PUSH 1_5.004014AE	ASCII "Your name must be at least one byte!"
00401643	PUSH 1_5.00403079	ASCII " Welcome to a Crackme "
0040165F	PUSH 1_5.0040309D	ASCII " Ribbere 1.42 "
0040167B	PUSH 1_5.004030C4	ASCII " your host: Bswap "
00401697	PUSH 1_5.004030EC	ASCII " Written in 100% ASM "
004016B3	PUSH 1_5.0040310E	ASCII " Some greetings go to: "
004016CF	PUSH 1_5.00403130	ASCII " All "
004016EB	PUSH 1_5.00403165	ASCII " Members "
00401707	PUSH 1_5.0040319A	ASCII " of "
00401723	PUSH 1_5.004031C6	ASCII " the "
00401758	PUSH 1_5.004031F2	ASCII " crackmes.de TEAM "
0040178D	PUSH 1_5.00403216	ASCII " Tools used "
004017A9	PUSH 1_5.00403242	ASCII " RadASM and Masm 32 U10 "
004017C5	PUSH 1_5.00403269	ASCII " Made in Holland "

- Ta thấy phía trên xuất hiện dòng lệnh GetDlgItemTextA -> đây có thể là hàm nhập input của chương trình

0040148C	. 75 E5	JNZ SHORT 1_5.00401473	
0040148E	. 6A 1E	PUSH 1E	
00401490	. 68 1C334000	PUSH 1_5.0040331C	
00401495	. 68 C9000000	PUSH 0C9	
0040149A	. FF35 00304000	PUSH DWORD PTR [403000]	
004014A0	. E8 59080000	CALL <JMP.&user32.GetDlgItemTextA>	GetDlgItemTextA
004014A5	. 83F8 01	CMPL EAX,1	
004014A8	. 7D 56	JGE SHORT 1_5.00401500	
004014AA	. EB 2F	JMP SHORT 1_5.004014DB	
004014AC	. EB 25	JMP SHORT 1_5.004014D3	
004014AE	. 59 6F 75 72	ASCII "Your name must b"	
004014BE	. 65 20 61 74	ASCII "e at least one b"	
004014CE	. 79 74 65 21	ASCII "yter",0	
004014D3	. EB 06	JMP SHORT 1_5.004014DB	
004014D5	. 45 72 72 6F	ASCII "Error",0	
004014DB	. 6A 40	PUSH 40	
004014DD	. 68 D5144000	PUSH 1_5.004014D5	
004014E2	. 68 AE144000	PUSH 1_5.004014AE	
004014E7	. 6A 00	PUSH 0	
004014E9	. E8 3A080000	CALL <JMP.&user32.MessageBoxA>	MessageBoxA
004014EE	. 6A 00	PUSH 0	
004014F0	. FF35 60304000	PUSH DWORD PTR [403060]	

Count = 1E (30.)
 Buffer = 1_5.0040331C
 ControlID = C9 (201.)
 hWnd = NULL

Style = MB_OK|MB_ICONASTERISK|MB_APPLMODAL
 Title = "Error"
 Text = "Your name must be at least one byte!"
 hOwner = NULL
 Enable = FALSE
 hWnd = NULL

- Tạo break point tại đây sau đó nhấn F9 để chạy chương trình và nhập thử key test là "1234" -> nhấn Register...

- Ta thấy có một vòng lệnh lặp Loop để gọi lại lệnh 0040197F

00401957	. C9	LEAVE
00401958	. C2 1000	RET 10
0040195B	\$ 55	PUSH EBP
0040195C	. 8BEC	MOV EBP,ESP
0040195E	. BE 1C334000	MOV ESI,1_5.0040331C
00401963	. BF 3B334000	MOV EDI,1_5.0040333B
00401968	. B9 10000000	MOV ECX,10
0040196D	> 0FB606	MOVZX EAX,BYTE PTR [ESI]
00401970	. 51	PUSH ECX
00401971	. 50	PUSH EAX
00401972	. E8 08000000	CALL 1_5.0040197F
00401977	. 8907	MOV DWORD PTR [EDI],EAX
00401979	. 47	INC EDI
0040197A	. 46	INC ESI
0040197B	^ E2 F0	LOOPD SHORT 1_5.0040196D
0040197D	. C9	LEAVE
0040197E	. C3	RET
0040197F	\$ 55	PUSH EBP

- Truy cập vào địa chỉ 0040197F: ta thấy dòng code từ 4019C5 đến 4019E0 là đoạn code “có ý nghĩa” trong hàm con 0040197F

004019C2	. 83C0 01	ADD EAX,1
004019C5	. 8B45 08	MOV EAX,DWORD PTR [EBP+8]
004019C8	. 03C1	ADD EAX,ECX
004019CA	. 83F8 21	CMP EAX,21
004019CD	~ 73 03	JNB SHORT 1_5.004019D2
004019CF	. 83C0 21	ADD EAX,21
004019D2	> 83F8 7B	CMP EAX,7B
004019D5	~ 7E 02	JLE SHORT 1_5.004019D9
004019D7	. D1E8	SHR EAX,1
004019D9	> 8945 08	MOV DWORD PTR [EBP+8],EAX
004019DC	. 8B45 08	MOV EAX,DWORD PTR [EBP+8]
004019DF	. C9	LEAVE
004019E0	. C3	RET
004019E1	\$ 8D3D E4324000	LEA EDI,DWORD PTR [4032E4]
004019F7	60 00	PUSH 0

- Giải thích đoạn mã trên:
- ECX là biến đếm được gán ban đầu bằng 10, khi tới lệnh LOOPD lần đầu tiên ECX sẽ được chuyển sang giá trị F và giảm dần 1 đơn vị qua từng vòng lặp LOOPD, vòng lặp sẽ thoát khi biến đếm sử dụng ECX bằng 0.
- Mã ascii trong hệ Hex của mỗi kí tự của key nhập vào (“1234”) được nạp vào EAX, giá trị của EAX sau đó được cộng thêm với ECX:
 - + Nếu EAX > 21, nhảy tới lệnh CC, tiếp đó EAX được so với 7B, ngược lại EAX được cộng thêm 21 để tăng lên giá trị lớn hơn 21.
 - + Nếu EAX < 7B thì bỏ qua lệnh SHR, lưu EAX vào [EBP + 8], ngược lại EAX bị dịch phải 1 để giảm xuống giá trị nhỏ hơn 7B.
- Dựa vào sự thay đổi giá trị của thang ghi ESI và EDI tại dòng lệnh 0040197A và 0040197B

00401977	. 8907	MOV DWORD PTR [EDI],EAX
00401979	. 47	INC EDI
0040197A	. 46	INC ESI
0040197B	^ E2 F0	LOOPD SHORT 1_5.0040196D
0040197D	. C9	LEAVE

- Ta có thể xem được kết quả của vòng lặp này như sau:

Address	Hex dump	ASCII
004032E0	73 65 2E 00 00 00 00 00	se.....
004032E8	00 00 00 00 00 00 00 00
004032F0	00 00 00 00 60 04 07 00'...
004032F8	0F 11 05 1A 74 07 3C 00t.<.
00403300	00 00 00 00 00 00 00 00
00403308	00 00 00 00 00 00 00 00
00403310	00 00 00 00 00 00 00 00
00403318	00 00 00 00 31 32 33 341234
00403320	00 00 00 00 00 00 00 00
00403328	00 00 00 00 00 00 00 00
00403330	00 00 00 00 00 00 00 00
00403338	00 00 0A 41 41 41 41 2D	...AAAA-
00403340	2C 2B 2A 29 28 27 26 25	,+*)('%%
00403348	24 23 22 00 00 00 00 00	\$#?.....
00403350	00 00 00 00 00 00 00 00
00403358	00 00 00 00 00 00 00 00
00403360	00 00 00 00 00 00 00 00
00403368	00 00 00 00 00 00 00 00
00403370	00 00 00 00 00 00 00 00

- Ở dòng địa chỉ 04003318 là các mã Hex dump của 1 kí tự 1 2 3 4 vừa nhập
- Ở dòng địa chỉ từ 0400333B đến 0400334A là 16 mã Hex tương ứng với vòng lặp 16 lần của lệnh LOOPD với biến đếm ECX từ giá trị 10, chuyển sang giá trị F (trong mã Hex) và giảm dần đến 0
-
- Sau khi vòng lặp được thực hiện ta chuyển đến dòng lệnh:

00401500	> 50	PUSH EHX
00401501	. E8 55040000	CALL 1_5.0040195B
00401506	. A1 3D334000	MOV EAX,DWORD PTR [40333D]
0040150B	. 8B1D 41334000	MOV EBX,DWORD PTR [403341]
00401511	. A3 E4324000	MOV DWORD PTR [4032E4],EAX
00401516	. 891D E8324000	MOV DWORD PTR [4032E8],EBX
0040151C	. E8 C0040000	CALL 1_5.004019E1

- Chạy từng dòng lệnh, khi kết thúc dòng lệnh 00401516, ta được kết quả tương ứng như sau ở thanh ghi địa chỉ và giá trị của EAX và EBX

REGISTERS (FPU)
EAX 2C2D4141
ECX 00000000
EDX 00001215
EBX 28292A2B

004032D0	75 72 20 73	6F 6C 75 74	ur solut
004032D8	69 6F 6E 20	70 6C 65 61	ion plea
004032E0	73 65 2E 00	41 41 2D 2C	se..AA-,
004032E8	2B 2A 29 28	00 00 00 00	+*)(....
004032F0	00 00 00 00	60 04 07 00'...
004032F8	0F 11 05 1A	74 07 3C 00t.<.

- ⇒ 4 byte (bắt đầu từ byte thứ 3) của các kí tự chuyển đổi ở dòng địa chỉ từ 0400333B đến 0400334A của chuỗi được mã hóa được lưu trữ bằng đoạn mã này
- Tiếp theo chương trình nhảy tới dòng lệnh 0040151C

004019E1	8D3D E432400	LEA EDI,DWORD PTR [4032E4]
004019E7	6A 00	PUSH 0
004019E9	6A 00	PUSH 0
004019EB	50	PUSH EAX
004019EC	DF2C24	FILD QWORD PTR [ESP]
004019EF	DF3424	FBSTP TBYTE PTR [ESP]
004019F2	59	POP ECX
004019F3	58	POP EAX
004019F4	8BD1	MOV EDX,ECX
004019F6	8BD8	MOV EBX,EAX
004019F8	C1E9 04	SHR ECX,4
004019FB	C1E8 04	SHR EAX,4
004019FE	83E3 0F	AND EBX,0F
00401A01	81E2 0F0F0F0F	AND EDX,0F0F0F0F
00401A07	81E1 0F0F0F0F	AND ECX,0F0F0F0F
00401A0D	81C2 30303030	ADD EDX,30303030
00401A13	81C1 30303030	ADD ECX,30303030
00401A19	83C0 30	ADD EAX,30
00401A1C	83C3 30	ADD EBX,30
00401A1F	8B07	MOV BYTE PTR [EDI],AL
00401A21	8B5F 01	MOV BYTE PTR [EDI+1],BL
00401A24	8B4F 08	MOV BYTE PTR [EDI+8],CL
00401A27	8B57 09	MOV BYTE PTR [EDI+9],DL
00401A2A	8B6F 06	MOV BYTE PTR [EDI+6],CH
00401A2D	8B77 07	MOV BYTE PTR [EDI+7],DH
00401A30	0FC9	BSWAP ECX
00401A32	0FCA	BSWAP EDX
00401A34	8B4F 02	MOV BYTE PTR [EDI+2],CL
00401A37	8B57 03	MOV BYTE PTR [EDI+3],DL
00401A3A	8B6F 04	MOV BYTE PTR [EDI+4],CH
00401A3D	8B77 05	MOV BYTE PTR [EDI+5],DH
00401A40	58	POP EAX
00401A41	C3	RET
00401A42	FF	PUSH EBP

- Mã tại 004019EF chuyển đổi 4 byte được lưu trữ gần đây (41412D2C) của chuỗi được mã hóa thành giá trị thập phân của nó.

EBP	0019F628
ESI	0040332C 1_5.0040332C
EDI	004032E4 ASCII "AA-,+*)"("
EIP	004019F2 1_5.004019F2
C 1	ES 002B 32bit 0(FFFFFFFF)
P 0	CS 0023 32bit 0(FFFFFFFF)
A 0	SS 002B 32bit 0(FFFFFFFF)
Z 0	DS 002B 32bit 0(FFFFFFFF)
S 0	FS 0053 32bit 3B6000(FFF)
T 0	GS 002B 32bit 0(FFFFFFFF)
D 0	
O 0	LastErr ERROR_SUCCESS (00000000)
EFL	00000203 (NO,B,NE,BE,NS,PO,GE,G
ST0	empty 0.0
ST1	empty 0.0
ST2	empty 0.0
ST3	empty 0.0
ST4	empty 0.0
ST5	empty 1.000000000000000000
ST6	empty 16.000000000000000000
ST7	empty 741163329.0000000000

- Tiếp theo, các mã còn lại định dạng phần thập phân đó theo ví dụ sau: nếu 123456789 là giá trị thập phân thì 2 số cuối sẽ được tách ra và thêm ký tự '3' vào -> '39' và '38' được lưu trữ sau khi định dạng.

Address	Hex dump	ASCII
004032D0	75 72 20 73 6F 6C 75 74	ur solut
004032D8	69 6F 6E 20 70 6C 65 61	ion plea
004032E0	73 65 2E 00 30 37 34 31	se..0741
004032E8	31 36 33 33 32 39 00 00	163329..
004032F0	00 00 00 00 60 04 07 00'...
004032F8	0F 11 05 1A 74 07 3C 00t.<.

- Sau đó chương trình nhảy đến địa chỉ 00401525

00401516	. 891D E8324000	MOV DWORD PTR [4032E8],EBX	
0040151C	. E8 C0040000	CALL 1_5.004019E1	
00401521	. 33C0	XOR EAX,EAX	
00401523	. 33DB	XOR EBX,EBX	
00401525	. 66:A1 3B3340	MOV AX,WORD PTR [40333B]	
0040152B	. 66:8B5F 08	MOV BX,WORD PTR [EDI+8]	
0040152F	. 66:2BC3	SUB AX,BX	
00401532	. 35 3F1B0000	XOR EAX,1B3F	
00401537	. 2D 23010000	SUB EAX,123	
0040153C	.v EB 03	JMP SHORT 1_5.00401541	
0040153E	01	DB 01	
0040153F	44	DB 44	CHAR 'D'
00401540	00	DB 00	
00401541	> 0BC0	OR EAX,EAX	
00401543	.v 75 54	JNZ SHORT 1_5.00401599	
00401545	.v EB 03	JMP SHORT 1_5.0040154A	
00401547	01	DB 01	

- Sau đó, tiến hành nạp 2 byte đầu của kết quả chuyển đổi trong vòng lặp vào thanh ghi AX (giá trị 4141)

00403330	00 00 00 00 00 00 00 00
00403338	00 00 00 41 41 41 41 2D	...AAAA-
00403340	2C 2B 2A 29 28 27 26 25	,+*)('&%
00403348	24 23 22 00 00 00 00 00	\$#".....
00403350	00 00 00 00 00 00 00 00
00403358	00 00 00 00 00 00 00 00

- Và nạp 2 giá trị vừa định dạng ở phần trước đó vào thanh ghi BX (giá trị 3932)

004032E0	73 65 2E 00 30 37 34 31	se..0741
004032E8	31 36 33 33 32 39 00 00	163329..
004032F0	00 00 00 00 60 04 07 00'....
004032F8	0F 11 05 1A 74 07 3C 00t.<.

Registers (FPU)	
EAX	00004141
ECX	32333134
EDX	39333631
EBX	00003932
ESP	0019F5D4

- Ta lưu ý các phép toán ở đoạn chương trình này...

0040152F	. 66:2BC3	SUB AX,BX	
00401532	. 35 3F1B0000	XOR EAX,1B3F	
00401537	. 2D 23010000	SUB EAX,123	
0040153C	.v EB 03	JMP SHORT 1_5.00401541	
0040153E	01	DB 01	
0040153F	44	DB 44	CHAR 'D'
00401540	00	DB 00	
00401541	> 0BC0	OR EAX,EAX	
00401543	.v 75 54	JNZ SHORT 1_5.00401599	
00401545	.v EB 03	JMP SHORT 1_5.0040154A	

- Kết quả cuối cùng EAX = 120D

Registers (FPU)	
EAX	0000120D
ECX	32333134
EDX	39333631
EBX	00003932
ESP	0019F5D4
EBP	0019F628
ESI	0040332C 1_5.0040332C
EDI	004032E4 ASCII "0741163329"
EIP	0040153C 1_5.0040153C

- Ở dòng lệnh 00401543 chương trình thực hiện so sánh EAX với 0 để thực hiện bước tiếp theo

00401541	> 0BC0	OR EAX,EAX
00401543	75 54	JNZ SHORT 1_5.00401599
00401545	EB 03	JMP SHORT 1_5.0040154A
00401547	01	DB 01

- Trường hợp này EAX khác 0, ta tiếp tục nhấn F8 để thực thi các lệnh tiếp theo nhưng chương trình nhảy vào các lệnh hệ thống và chạy rất lâu

76DDBA95	8B89 28010000	MOV ECX,DWORD PTR [ECX+128]
76DDBA98	8B41 08	MOV EAX,DWORD PTR [ECX+8]
76DDBA9E	0B41 0C	OR EAX,DWORD PTR [ECX+C]
76DDBAA1	0F84 10010000	JE USER32.76DDBBB7
76DDBAA7	8B41 08	MOV EAX,DWORD PTR [ECX+8]
76DDBAAA	F640 18 01	TEST BYTE PTR [EAX+18],1
76DDBAAE	0F85 03010000	JNZ USER32.76DDBBB7
76DDBAB4	6A 01	PUSH 1
76DDBAB6	56	PUSH ESI
76DDBAB7	56	PUSH ESI
76DDBAB8	56	PUSH ESI
76DDBAB9	8D45 D4	LEA EAX,DWORD PTR [EBP-2C]
76DDBABC	50	PUSH EAX
76DDBABD	E8 BE08FFFF	CALL USER32.PeekMessageW
76DDBAC2	85C0	TEST EAX,EAX
76DDBAC4	75 64	JNZ SHORT USER32.76DDBB2A
76DDBAC6	8B45 F8	MOV EAX,DWORD PTR [EBP-8]
76DDBAC9	85C0	TEST EAX,EAX
76DDBACB	0F84 43010000	JE USER32.76DDBC14
76DDBAD1	85FF	TEST EDI,EDI
76DDBAD3	74 17	JE SHORT USER32.76DDBAEC
76DDBAD5	57	PUSH EDI
76DDBAD6	E8 0564FEFF	CALL USER32.IsWindow
76DDBADB	F7D8	NEG EAX
76DDBADD	1BC0	SBB EAX,EAX
76DDBADF	23F8	AND EDI,EAX
76DDBAE1	74 09	JE SHORT USER32.76DDBAEC
76DDBAE3	3975 F4	CMP DWORD PTR [EBP-C],ESI
76DDBAE6	0F84 AA000000	JE USER32.76DDBB96
76DDBAEC	8BCB	MOV ECX,EBX
76DDBAEE	E8 E7B8FEFF	CALL USER32.76DC73DA
76DDBAF3	85C0	TEST EAX,EAX
76DDBAF5	0F84 BC000000	JE USER32.76DDBBB7
76DDBAFD	8D45 F8	LEA EAX,DWORD PTR [EBP-8]

- Ta quay lại dòng lệnh xét EAX với 0

00401541	> 0BC0	OR EAX,EAX
00401543	75 54	JNZ SHORT 1_5.00401599
00401545	EB 03	JMP SHORT 1_5.0040154A
00401547	01	DB 01

- Ở trường hợp EAX bằng 0, ta thấy chương trình nhảy tới dòng lệnh 0040154A

00401543	75 54	JNZ SHORT 1_5.00401599
00401545	EB 03	JMP SHORT 1_5.0040154A
00401547	01	DB 01
00401548	44	DB 44
00401549	00	DB 00
0040154A	EB 02	JMP SHORT 1_5.0040154E
0040154C	2300	AND EAX,DWORD PTR [EAX]
0040154E	6A 01	PUSH 1
00401550	FF35 60304000	PUSH DWORD PTR [403060]
00401556	E8 73070000	CALL <JMP.&user32.EnableWindow>
0040155B	6A 00	PUSH 0
0040155D	FF35 64304000	PUSH DWORD PTR [403064]
00401563	E8 66070000	CALL <JMP.&user32.EnableWindow>
00401568	8D05 C9104000	LEA EAX,DWORD PTR [4010C9]
0040156F	8BF8 1C	MOV EAX,1C

CHAR 'D'

```

[Enable = TRUE
hwnd = 000A0482 (class='Edit',parent=003C0774)
EnableWindow
Enable = FALSE
hwnd = 000C04A2 (class='Edit',parent=003C0774)
EnableWindow

```

- Tiếp đó, chương trình gửi một thông điệp MessageA, ta có thể bước đầu kết luận rằng đây là Good boy của chương trình...

0040154C	. 2300	AND EAX,DWORD PTR [EAX]	
0040154E	> 6A 01	PUSH 1	Enable = TRUE
00401550	. FF35 60304000	PUSH DWORD PTR [403060]	hWnd = NULL
00401556	. E8 73070000	CALL <JMP.&user32.EnableWindow>	EnableWindow
00401558	. 6A 00	PUSH 0	Enable = FALSE
0040155D	. FF35 64304000	PUSH DWORD PTR [403064]	hWnd = NULL
00401563	. E8 66070000	CALL <JMP.&user32.EnableWindow>	EnableWindow
00401568	. 8D05 C9104000	LEA EAX,DWORD PTR [4010C9]	
0040156E	. 83E8 1C	SUB EAX,1C	
00401571	. 50	PUSH EAX	
00401572	. 6A 00	PUSH 0	lParam => 4010AD
00401574	. 6A 0C	PUSH 0C	wParam = 0
00401576	. FF75 08	PUSH DWORD PTR [EBP+8]	Message = WM_SETTEXT
00401579	. E8 B6070000	CALL <JMP.&user32.SendMessageA>	SendMessageA
0040157E	. 8D05 AC324000	LEA EAX,DWORD PTR [4032AC]	
00401584	. 83C0 0F	ADD EAX,0F	
00401587	. 50	PUSH EAX	lParam => 4032BB
00401588	. 6A 00	PUSH 0	wParam = 0
0040158A	. 6A 0C	PUSH 0C	Message = WM_SETTEXT
0040158C	. 68 CA000000	PUSH 0CA	ControlID = CA (202.)
00401591	. FF75 08	PUSH DWORD PTR [EBP+8]	hWnd
00401594	. E8 95070000	CALL <JMP.&user32.SendDlgItemMessageA>	SendDlgItemMessageA
00401599	> 817D 10 2F01	CMP DWORD PTR [EBP+10],12F	

- Vậy chương trình sẽ đi đến Good boy khi EAX = 0 sau các dòng lệnh từ 0040152F đến 00401537:

0040152F	. 66:2BC3	SUB AX,BX	
00401532	. 35 3F1B0000	XOR EAX,1B3F	
00401537	. 2D 23010000	SUB EAX,123	
0040153C	.. EB 03	JMP SHORT 1_5.00401541	
0040153E	. 01	DB 01	CHAR 'D'
0040153F	. 44	DB 44	
00401540	. 00	DB 00	
00401541	> 0BC0	OR EAX,EAX	
00401543	.. 75 54	JNZ SHORT 1_5.00401599	
00401545	.. EB 03	JMP SHORT 1_5.0040154A	

- Xét ngược lại từ chương trình, ta sẽ tìm cách tạo một Keygen để phát sinh key cho chương trình 1.5.exe này:
- Ta thấy để EAX bằng 0, xét tại dòng 00401537, EAX sẽ bằng 123. Trước đó, để XOR với 1B3F bằng 123, EAX sẽ bằng 1A1C.
- Vậy hiệu giữa AX – BX sẽ là 1A1C. [1]
- Ta thấy, chương trình thực hiện một quá trình chuyển đổi sang giá trị thập phân sau khi xử lý key nhập

004032D0	75 72 20 73	6F 6C 75 74	ur solut
004032D8	69 6F 6E 20	70 6C 65 61	ion plea
004032E0	73 65 2E 00	41 41 2D 2C	se..AA-,
004032E8	2B 2A 29 28	00 00 00 00	+) (....
004032F0	00 00 00 00	60 04 07 00'...
004032F8	0F 11 05 1A	74 07 3C 00t.<.



Address	Hex dump	ASCII
004032D0	75 72 20 73 6F 6C 75 74	ur solut
004032D8	69 6F 6E 20 70 6C 65 61	ion plea
004032E0	73 65 2E 00 30 37 34 31	se..0741
004032E8	31 36 33 33 32 39 00 00	163329..
004032F0	00 00 00 00 60 04 07 00'...
004032F8	0F 11 05 1A 74 07 3C 00t.<.

- Vì vậy ta cần chắc rằng key nhập sẽ không có những trường bị thay đổi không tính toán tới, nghĩa là ở đoạn code này:

004019C2	. 83C0 01	ADD EAX,1
004019C5	. 8B45 08	MOV EAX,DWORD PTR [EBP+8]
004019C8	. 03C1	ADD EAX,ECX
004019CA	. 83F8 21	CMP EAX,21
004019CD	~ 73 03	JNB SHORT 1_5.004019D2
004019CF	. 83C0 21	ADD EAX,21
004019D2	> 83F8 7B	CMP EAX,7B
004019D5	~ 7E 02	JLE SHORT 1_5.004019D9
004019D7	. D1E8	SHR EAX,1
004019D9	> 8945 08	MOV DWORD PTR [EBP+8],EAX
004019DC	. 8B45 08	MOV EAX,DWORD PTR [EBP+8]
004019DF	. C9	LEAVE
004019E0	. C3	RET
004019E1	\$ 8D3D E4324000	LEA EDI,DWORD PTR [4032E4]
004019F7	60 00	PUSH 0

Ta sẽ cần phát sinh những số nằm trong khoảng 21 đến 7B, và trừ đi ECX biến đếm tương ứng, để chắc rằng giá trị EAX + ECX luôn nằm trong đoạn 21 đến 7B để chương trình không nhảy vào các trường hợp không mong đợi bị thay đổi giá trị (dòng lệnh 004019CF và dòng lệnh 004019D7) [2]

- Xét kĩ lại vòng lặp trong đoạn chương trình sau:

00401957	. C9	LEAVE
00401958	. C2 1000	RET 10
0040195B	\$ 55	PUSH EBP
0040195C	. 8BEC	MOV EBP,ESP
0040195E	. BE 1C334000	MOV ESI,1_5.0040331C
00401963	. BF 3B334000	MOV EDI,1_5.0040333B
00401968	. B9 10000000	MOV ECX,10
0040196D	> 0FB606	MOVZX EAX,BYTE PTR [ESI]
00401970	. 51	PUSH ECX
00401971	. 50	PUSH EAX
00401972	. E8 08000000	CALL 1_5.0040197F
00401977	. 8907	MOV DWORD PTR [EDI],EAX
00401979	. 47	INC EDI
0040197A	. 46	INC ESI
0040197B	^ E2 F0	LOOPD SHORT 1_5.0040196D
0040197D	. C9	LEAVE
0040197E	. C3	RET
0040197F	\$ 55	PUSH EBP

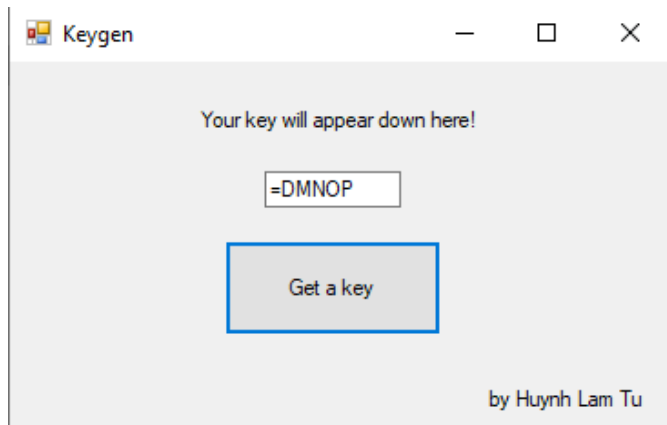


004019C2	. 83C0 01	ADD EAX,1
004019C5	. 8B45 08	MOV EAX,DWORD PTR [EBP+8]
004019C8	. 03C1	ADD EAX,ECX
004019CA	. 83F8 21	CMP EAX,21
004019CD	~ 73 03	JNB SHORT 1_5.004019D2
004019CF	. 83C0 21	ADD EAX,21
004019D2	> 83F8 7B	CMP EAX,7B
004019D5	~ 7E 02	JLE SHORT 1_5.004019D9
004019D7	. D1E8	SHR EAX,1
004019D9	> 8945 08	MOV DWORD PTR [EBP+8],EAX
004019DC	. 8B45 08	MOV EAX,DWORD PTR [EBP+8]
004019DF	. C9	LEAVE
004019E0	. C3	RET
004019E1	\$ 8D3D E4324000	LEA EDI,DWORD PTR [4032E4]
004019F7	60 00	PUSH 0

- Ta thấy rằng, ECX sẽ tương ứng với lần lượt qua mỗi vòng lặp những giá trị sau: 10, F, E, D, C, B, A, 9, 8,...,1
- Khi kết thúc dòng lệnh 00401516, 4 byte (bắt đầu từ byte thứ 3) của các kí tự chuyển đổi ở dòng địa chỉ từ 0400333B đến 0400334A của chuỗi được đem đi thực hiện quá trình: mã hóa sang giá trị thập phân, xử lí với hai số cuối XY của giá trị thập phân này thành dạng 3Y3X và sau đó dòng lệnh từ 00401525 đến 00401537 tiến hành so sánh hiệu 2 byte đầu của dãy kí tự được chuyển đổi ban đầu với 3Y3X
- Ta suy ra được key nhập phải gồm có 6 kí tự. [3]

Thuật toán phát sinh key

- Từ [1], [2] và [3] tiến hành phân tích ngược chương trình ta suy ra được thuật toán phát sinh key như sau:
 - + Gọi key đúng là ABCDEF
 - + Ta sẽ phát sinh một số ngẫu nhiên trong đoạn 21 đến 7B, (giá trị này bằng EAX + ECX, trong vòng lặp sẽ không bị thay đổi trong những phép so sánh), gọi số này là X
 - + Sau đó gán XXXX là giá trị của 4 byte cuối của key sau khi chuyển đổi trong vòng lặp LOOPD
 - + Gọi M là giá trị thập phân của XXXX và PQ là 2 chữ số cuối của M
 - + Khi đó ta tạo được số 3P3Q chính là BX
 - + AX khi đó sẽ bằng BX: 3P3Q cộng với giá trị tạo key đúng là **1A1C**, ta được số Hexa mới chắc chắn có 4 kí tự đặt là ERTY
 - + Dựa vào bước 1, ta trừ tương ứng các kí tự kết quả với ECX trong thứ tự của vòng lặp, khi đó các kí tự của key đúng sẽ là:
 - A = ER - 10 (vòng lặp 1 ECX = 10)
 - B = TY - F (vòng lặp 2)
 - C = X - E (vì X = EAX + ECX, ECX đang tại vòng lặp 3)
 - D = X - E + 1 (để bảo toàn rằng sau xử lí 4 kí tự cuối đề bằng nhau và bằng X)
 - E = X - E + 2
 - F = X - E + 3
- Tạo một chương trình phát sinh key ngẫu nhiên theo thuật toán đã trình bày, bằng C#
- Phát sinh một key là: **=DMNOP**



- Copy key này chạy thử:



⇒ Thành công!!!

Tổng kết

Mức độ hoàn thành

1.1	100%
1.2	100%
1.3	100%
1.4	0%
1.5	100%

Tham khảo

1. Cộng đồng crackme: <https://crackmes.one/>
2. Tài liệu hướng dẫn lập trình C# trên Windows Forms App: <https://www.youtube.com/watch?v=JwLn2dISK6Q>

HẾT!!
NHÓM XIN CẢM ƠN THẦY CÔ ĐÃ XEM! CHÚC THẦY CÔ MỘT NGÀY TỐT LÀNH!!