OllyDbgによる実践ソフトクラック 初級編(その1)

1. ソフトプロテクトの種類

概ね, ソフトプロテクトには次の3種類の型があります.

1) シリアルナンハ<sup>、</sup>-認証型 2) 試用期限型 3) ID・ハ<sup>°</sup>スワート<sup>、</sup>型

2. ソフトフ<sup>°</sup>ロテクトクラック概要

1) シリアルナンバー認証型

比較的に容易なもので、チェックルーチンが解読できれば簡単にCrackできます. この構造・動作にはいくつものヒントとなる要素が含まれていて、その要素に 着目することでCrackの端緒を見いだします.

- \* 一般的なシリアルナンバーのチェックルーチン
- a. ユーサーが入力したシリアルナンハー(文字列)を取得する
- b. シリアルナンバーの妥当性をチェックする(ここが特定すべき個所)
- c. チェックの結果に応じてメッセージを表示する

エディットボックスから文字列を取得するにはWindowsが提供するAPI関数を 用いています.

\* 文字列取得: API(Application Program Interface)関数

GetWindowText, GetDlgItemText, GetDlgItemInt の3種類です

これらのAPI関数が呼び出される瞬間を捉えることにより、チェックルーチンの個所を 特定します.これを実現するのがブレークポイントの設定です. OllyDbgでは実行ファイルが使用しているAPI関数の一覧を表示することが出来ます.

具体的な操作は後述します.

- \* チェックルーチンのトレースの第一歩
- a. OllyDbgでソフトの実行ファイルを開きます
- b. F9キーを押して、シリアルナンバー入力のエディットホックスが出たら、フェイクシリアルナンバーを入れておきます([OK]、[登録]等のホックンは押しません)
- c. 文字列取得のAPI関数Iこブレークポイントを設定します(実例を後述します)
- d. ここで, [OK], [登録]等のボタンを押します
- e. API関数が実行される直前で停止(ブレーク)します
- f. この先にチェックルーチンが存在するはずです
- g. これからは、1命令ずつ実行(F8キーを押す)して精査を行っていきます

アプリケーションによっては簡単に解明するものもあれば、この先複雑なルーチンを 仕掛けているものもあります.俗に言う『倍直』は実行ファイルにシリアルナンバーを 直接書き込んでいるもので、いとも簡単にCrackできます.

それでは"CrackMe0x01. exe"を解析してみましょう.



> OllyDbgで ソフトの"CrackMe0x01. exe"を開きます.

F9キーを1回押します.

00401027 00401029 0040102E 0040102F 00401034 00401039 00401038 00401038 00401038	. 8800 . E8 6E110000 . F8 04050000 . E8 67110000 . E8 C410000 . E8 C410000 . E8 C410000	mov         edx, eax           call         0040219C           pop         edx           call         00401538           call         00401538           call         004021A0           push         0           call         00402504	Car	s1 = 00000000 ackike0.00402504	STO empty STI empty SI2 empty ST3 empty ST4 empty ST5 empty ST6 empty ST7 empty FST 00000 FCW 027F	-UNDERN EDEC 01050104 0000000 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	(GT)
71'17	Hex9">7"		ASCII	· 71'17	値	JX>F	^
00409000	00 00 2C 2D 40 00 00	00 CO 2F 40 00 00 05 AC 37	a JIEA //8. 147	0012FF0	4 7C816FD7	RETURN先: kernel32.7C816FD7	
00409020	D0 4B 40 00 00 0A EC	46 40 00 00 0A B0 50 40 00	3K8. 8P8.	0012FFC	C FFFFFFFF	11011.70050750	
00409030	00 01 BC 57 40 00 00	01 30 58 40 00 00 03 00 50	. r?#8r<[8]	0012FFD	0 7FFDB000		
00409040	40 00 00 02 7C 62 40		e., 1 be. Dde	0012FFU	4 8054C6ED		
00408050	A8 6A 40 00 00 00 1C	00 68 38 40 00 00 01 81 26 40 00	1/2 ble L	0012FF0	C 22668C68		
00409070	40 00 00 02 B4 62 40	00 00 03 C4 62 40 00 0B C9	8. 11b8. 11b8. 21	0012FFE	0 FFFFFFFF	SEH連鎖のポインタ	120
00409080	74 4D 83 3D 42 6F 72	6C 61 6E 64 20 43 28 28 20	tM Borland C++	0012FFE 0012FFE	4 7C839AA8	SEH71- UZ	~
分析データ(	FrackMeO: 209 の判例したプロン	/-プャ, 78 calls: 既知の, 314 calls:	推測される関数コール			傳上中	



# > シリアルナンバー入力のエディットホックスが出たら、 フェイクシリアルナンバーを入れておきます。 ([OK], [登録]等のホタンは押しません)



🐮 OllyDbg –	GrackMe0	d01.exe			
JP11KE)	表示( <u>V</u> )	解析(D)	フラグイン オブション()	) ウィンドウ( <u>W</u> )	ノモン(王)
		NTI -	- LEMTWHO	KBRS	E 2
CPU - X	インスレット、	モジュール C	rackMe0	- 🗆 🗙	
7 N 711	ネーム Crack	MeO			
71 71 07	セクション	917	2-6		<b>^</b>
00409110	G data	Export	CPPdebusHook	ndle	
00 0040E0D0	C idata	Import	KERNEL 32 Createl	ileA .	
0040E190	idata	Import	USER32. EndDialog	Farana	
00000E1A4	.idata	Import	USER32.EnumThrea	difindows	
0040E0E4	4 .idata	Import	(KERNEL 32 Get ACP	cess	
0040E0E0	C.idata	Import	KERNEL 32 Get Comm	andLineA	
0040E0F0	0.idata	Import	(KERNEL 32, GetCurr	ro entThreadId	
00 0040E1A8	idata .	Import	USER32.GetDigite	m	
004020F4	9 .idata 9 .text	Export	GetExceptDLL in	fo	
0040E0F8	8 .idata	Import	KERNEL32.GetFile	Туре	
0040E0F0	l idata	Import	(KERNEL 32. GetLast (KERNEL 32. GetLoca	LTime	
0040E104	4 . idata	Import	KERNEL32 Get Modu	leFileNameA	
0040E100	C.idata	Import	KERNEL 32. Get OEMC	P	-

🔆 OllyDbg – CrackMe0x01.exe		
ファイル(E) 表示(V) 解析(D)	) プラグイン オプション( <u>T</u> ) ウィンドウ( <u>W</u> ) ^	ルプ(日)
	J → LEMTWHC/KBRS	≡∎?
CPU – メイン スレット・, モジューノ	V CrackMe0	
ፖドレス  Hexダンプ	ディスアセンブル 🗛	
00401000 \$~ EB 10	imp short 00401012	
00401002 00 0040100 N Find: GETWINDOW	ТЕХТА	
0040101 アト・レス セクション	\$17° 12-4	
004010 0040E10C .idata	Import (KERNEL32.GetOEMCP	
00401000040E110 .idata	Import (KERNEL32.GetProcAddress	
0040100 0040E118 . idata	Import (KERNEL32.GetStartupInfoA	
004010 0040E11C .idata	Import (KERNEL32.GetStdHandle	
0040100 0040E120 . idata	Import (KEKNEL32.GetVersion	
0040100 0040E128 . idata	Import (KERNEL32.GetVersionExA	
004010 0040E1AC .idata	Import USER32.GetWindowTextA	
004010 0040E12C .idata	Import (KERNEL32.GlobalMemoryStatus	
004010 0040E130 .idata	Import (KEKNEL32.HeapAlloc	
004010 0040F1B0 idata	Import (USER32.LoadIconA	
28.1918 0040E138 .idata	Import (KERNEL32.LoadLibraryA	
0040E1B4 .idata	Import (USER32.MessageBoxA	
UU4UE13C .idata	Import (KEKNEL32, KaiseException	
0040E140 .Idata	Import (NERNELSZ.Rtiunwind)	
0040E144 . idata	Import (KERNEL32.SetConsoleCtrlHand	er 🚽
0040E148 .idata	Import (KERNEL32.SetFilePointer	× .
モンショール C:¥WINDOWS¥system32¥Secur)	32.dll	
		) )20131

€9'\_\_\_& C:¥WINDOWS¥system32¥Secur32.d

🐮 OllyDbg – GrackMe0x01.exe – [	CPU - メイン スレット・、モジュール GrackMe0]		
C ファイル(E) 表示(Y) 解析(D)	フラグイン オブション(D ウィンドウ(M) ヘルン(	В)	_ 6 ×
	LEMIWHC/KBR.S	laster i	
00401105 B8 76750000	call (imp &USER32 GetWindowLext A)	177L	
004011CA = \$2 4C\$14000 004011CF = \$055 00111FF 00401105 = \$2 00100000 00401108 = \$3C4 108 00401184 = \$3C4 08 00401184 = \$3C4 08 00401184 = \$3C4 08 00401184 = \$3C4 08 = \$505 00401184 = \$3C4 08 = \$505 00401184 = \$3C4 08 = \$505 00401184 = \$3C4 08 = \$505 00401184 = \$3C4 08 = \$50514000 00401184 = \$6 3.5914000 00401184 = \$6 3.5914000 00401184 = \$6 3.5914000 00401184 = \$6 3.5914000 00401185 = \$6 3.5914000 00401184 = \$6 3.5914000 00401174 = \$6 3.00 00401174 = \$6 3.00 00401204 = \$6 3.00 00401210 = \$6 3.00 00401218 = \$2 1000 00401218 = \$2 1000 00401210 = \$2 1000 00408748=<1 mp. &USER32.GetWind	call churchester push 0040912C lea cax, Leword ss:ebp=100] push 0040912C teall 00402185 push 00402185 push 00409150 push 00409150 push 00409150 push 10409150 push 10409150 push 00409150 push 00409150 push 00409150 push 00409150 push 00409151 push 00409150 push 00409152 push 104001218 push 00409152 push 10409152 push 00409152 push 00409152 push 00409152 push 00409152 push 00409152 push 00409152 push 104001218 push 00409152 push 00409152 push 104001218 push 00409152 push 00409154 push 00409154 push 00409154 push 00409154 push 00409154 push 00409155 push 00409154 push 00409155 push 00409155 push 00409154 push 00409155 push 005 push 005	ASCII "TRONWHIP" Style = MB_OK MB_A Title = "CrackMeOx Text = "SIR&SUCK hOwner MessangeBoxA Style = MB_OKIMB I Title = "CrackMeOx Text = "2"U7)U7 MessangeBoxA Result = 0; Case 1 hInd ASCII "TRONWHIP"	Characterization         Characterization           CONTRACT         CrackMe0.00401150           END 001101         CrackMe0.00401150           END 001102         CrackMe0.00401150           END 001102         CrackMe0.00401150           END 001102         CrackMe0.00401150           END 001102         CrackMe0.00401155           C 0 FS 0012 32bit 0(FFFFFFFF)         N           A 0 SS 0023 32bit 0(FFFFFFFFF)         N           A 0 SS 0023 32bit 0(FFFFFFFFF)         N           S 0 FS 0038 32bit 0(FFFFFFFFF)         N           A 0 SS 0000 NULL         O           D 0         774 115.           EFL 00000246 (NO.NE.E.BE.NS.PE.0E.LE)           S10 empty 272 FFF 004646 Control           S11 empty 272 FFF 004664 Control           S12 empty 272 FFF 004664 Control
71'12 Hexy'27*	ASCII	<u>~</u> 71'bZ	値 コメント ^
00409000 00 00 2C 20 40 00 01 00409010 40 00 00 40 83 44 00409020 00 48 40 00 00 0A E 00409030 00 01 18 57 40 00 00409030 40 00 18 57 40 00 00409040 40 00 00 27 7C 82 41 00409080 00 00 18 27 40 00 01 00409080 00 00 18 27 40 00 00 00409080 00 00 18 27 70 77 97 00409080 00 00 00 18 27 70 77 97 018 20 27 20 20 43 8E 70 00 107	0 00 (10) 21 40 00 00 15 AC 37         •         <		A         00010322         (rhmd)         00010322         (class)         Ldit         parent           3         0017500         Buffer         = 0017500         Buffer         = 0017500           3         0010000F         Count         = FF (255.)         0         0         000000F           4         000000F         0

> Ctrl+Nキーを押します.

常行中

> ここでキーホート、で「GETWINDOWTEXTA」とタイプします.

GetWindowTextA を選択した状態で右クリック→ [全ての参照にブレークポイントをセット]を選択すると, GetWindowTextA を呼び出している命令行の全てに ブレークポイントが設定されます.

この時点で[Find]画面は消してもOK.

[CPU]画面を拡大します.

- 【注記】: API関数名の末尾の"A"はANSIコード形式の 文字列に対応した関数を表します. 因みに、"W"はUnicode形式を表します.
- > API関数が実行される直前で停止(ブレーク)しました.

この call命令でタイプした″98765432″の文字列を取得して いるようです.

🐮 OllyDbg – GrackMe0x01.exe – [G	PU ー メイン スレット・、モン・ュール CrackMe0]		
C ファイル(E) 表示(Y) 解析(D)	フラゲイン オブション(I) ウィンドウ(W) ヘルブ(H)	6—.	_ # ×
	LEMTWHC/KBRS		
71" Hex9">7"	5° (ZPE)7° N		・ レジスタ (FPU) く く
00408746         \$= F25         AC: 14000           00408746         \$= F75         BE: 14000           00408754         \$= F75         BE: 14000           00408754         \$= F75         BE: 14000           00408754         \$= F75         BE: 14000           00408765         CC         00408766           00408766         00         00408766           00408766         00         00408766           00408766         00         00408766           00408766         00         00408766           00408766         00         00408766           004087760         00         00408776           00408776         00         00408777           00408777         00         00408777           00408776         00         00408777           00408777         00         00408776           00408777         00         00408776           00408778         00         00408776           004048778         00         00408778           004048778         00         00408778           004048778         00         00408778           004048778         00         00           <	Imp         near         deword         des:<&USER32.0edHpind           imp         near         deword         des:<&USER32.0edHpind	USER82: Cetty indewTe USER82: Acad ConA USER82: Acad ConA USER82: Acad ConA USER82: Send Acad Con USER82: Acad Cond USER82: Acad Cond USER82: Acad Cond USER82: Cond USER82: Cond Cond Cond Cond Cond Cond Cond Cond	EAX 00000000 EXX 00000000 EXX 00000000 ESY 00000000 ESY 00000000 ESY 00000000 ESY 00000000 ESY 0000000 ESY 0000000 ESY 0040150 CrackMe0.00401150 EDY 01400748 (jmp.&USER32.GetWindowTextA) C 0 ES 0023 32bit 0(FFFFFFF) A 0 SS 0023 32bit 0(FFFFFFFF) A 0 SS 0023 32bit 0(FFFFFFFF) A 0 SS 0023 32bit 0(FFFFFFFF) C 0 ES 0023 32bit 0(FFFFFFFF) A 0 SS 0023 32bit 0(FFFFFFFF) C 0 SO 000 NULL D 0 C 0 774-17-: ERROR_SUCCESS (0000000) EFL 00000248 (NO.NB.E.BE.NS.PE.GE.LE) S10 empty -222 FFFF 00FF00FF 00FF00FF S12 empty -222 FFFF 00FF00FF 00FF00FF S14 empty -222 FFFF 00FF00FF 00FF00FF S14 empty -222 FFFF 000F000F 000F000F0 S16 empty -222 FFFF 00000000 S16 empty -222 FFFF 000000000 S16 empty -222 FFFF 00000000 S16 empty -220 FFFF 00000000 S16 empty -200 FFFFFFFFFFF
u-加のCALL元: 00401105		FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT) FCW 1372 Prec NEAR.64 Mask 1 1 0 0 1 0	

### > F8キ−を1回押します.

次のAPI関数が実行される直前で再度停止(ブレーク) しました.

#### call命令の内部でまた停止したようです.



004011C5 E8 7E750000 <jmp. &USER32. GetWindowTex> call 004011CA 68 2C914000 0040912C ;ASCII "IRONWHIP" push eax, [dword ss:ebp-100] 004011CF 8D85 00FFFFF lea 004011D5 50 push eax 004011D6 E8 DD100000 004022B8 call

004011CA の push命令で メモリー0040912C に文字列"IRONWHIP"を書き込んでいます. 004011CF の lea命令(実効アドレスの取得)で文字列"98765432"のアドレスを eaxレジスタに書き込んでいます 004011D5 の push命令でレジスタ eax に文字列"98765432"を書き込んでいます. 004011D6 の call命令で比較をしているようです.

コメン

0322 h#nd = 00010322 (class='Edit 500 Buffer = 00125900 00FF -Count = FF (255.)

RETURN先: USER32.77D0F8D6 from GD13

ここまで解析すれば、チェックルーチンを回避するパッチも作成できるかもしれません、



CrackMe0x01 区 シリアルナンバーが違います。 OK > F9キーを1回押します.

call命令を出ました.

> F8キーを3回押して 00411D6 まで進めてみます.

スタックウィント・ウに注目します.

0012F9C8 0012F9D0 ASCII "98765432" 0012F9CC 0040912C ASCII "IRONWHIP"

"IRONWHIP"がシリアルナンバーのようです.

> F9キーを押してみます. 間違いないようです.



> OllyDbg を終了します.

アプリケーションを再起動して正しいシリアルナンバーを入力します. 的中でした!!

次回は 2) 試用期限型 です 印刷時のページレイアウトは考慮していませんので悪しからず.

以上

OllyDbgによる実践ソフトクラック 初級編(その2)

#### 2) 試用期限型

概ね,その試用期限の取り方で2種類があります.

- + 初回起動日から一定日数まで試用可能なもの
- + 試用期限を過ぎると試用できなくなるもの
- \* 一般的な試用期限のチェックルーチン
- a. 現在日時を取得する
- b. 試用期限の妥当性をチェックする(ここが特定すべき個所)
- c. チェックの結果に応じてメッセージを表示する

現在日時を取得するにはWindowsが提供するAPI関数を 用いています。

\* 現在日時取得: API(Application Program Interface)関数

GetLocalTime, GetSystemTime の2種類です

- これらのAPI関数が呼び出された後のチェックルーチンの個所を特定します.
- \* チェックルーチンのトレースの第一歩
- a. OllyDbgでソフトの実行ファイルを開きます
- b. 現在日時取得のAPI関数にブレークポイントを設定します
- c. ここで, [OK], [登録]等のボタンを押します
- d. API関数が実行される直前で停止(フレーク)します
- e. この先にチェックルーチンが存在するはずです

アプリケーションによってはPCの日時設定を操作すると使用期限内で試用可能なものもありますが、これはPCにいろんな障害を起こす恐れがあります. ただし、インターネットサバーから現在日時を取得している場合は通用しません.

それでは"CrackMe0x02. exe"を解析してみましょう.



🐮 OllyDbg – GrackMe0x02.exe	e – [CPU – メイン スレッド , モジュール GrackMe0]		E 2 3
C ファイル(E) 表示(Y) 解析	(D) 7.5ゲイン オブション(D) ウィンドウ(M) ヘルブ(H)		- 0 ×
	TI TI LEMIWHC/KBK	[nth	1 37 4 (EDII)
$\begin{array}{rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr$	7       AP(2) *         [eal]       Call       Cimp. MKIRNEL32. GetLocalline>         [ea]       eax, [dword ss:ebp-10]         [ush       eax, [dword ss:ebp-10]         [ush       eax, [ak]         [call       0.0401236         [push       eax, [ak]         [ush       0.0409151         [push       0.0409151         [push       0.0409152         [mord       s:ebp+8]         [call <imord< td="">         [call       <imord< td="">         [call       <imord< td="">         [mord       s:ebp+8]         [call       <imord< td="">         [call       <imord< td="">         [call       <imord< td="">         [call       <imord< td="">         [mord       s:ebp+8]         [call       <imord< td="">         [mord       s:ebp+8]         [call       <imord< td="">         [mord       s:ebp+8]         [call       <imord< td="">         [mord       s:ebp-</imord<></imord<></imord<></imord<></imord<></imord<></imord<></imord<></imord<></imord<></imord<></imord<></imord<></imord<></imord<>	CArg1 Card cost 1 inc Carg Cost 1 inc Style = MB_OKIMB_IC Title = CrackMeDx0 Text = "2005410711 bOwner = 0 Wessauce = WA_CLOSE Ind Secult = 0; Case 10 bMed Inclassor Case 1 = 0; Case 10 bMed Inclassor Default case of swi	L2X3         (FPO)         < <
00406070- Cjillp. akelule 22. C			3 2 1 0         E S P U 0 Z D I           FST 4000         Cond I 0 0 0         Err 0 0 0 0 0 0 0 0           FCW 1372         Prec NLAR, 64         Mask         1 1 0 0 1 0
TF UA         Hex3         277           00409000         00         00         20         20         40         0           00409000         00	ASCI1         ASCI1           15         40         00         00         5         A0.37	A         7F-VA         III           0012FA40         0012FA44         00012FA45           0012FA45         0012FA46         0012FA46           0012FA45         0012FA46         00012FA45           0012FA45         00100000         0012FA45           0012FA45         00100000         0012FA45           0012FA45         0012FA45         10012FA35           10012FA55         0012FA45         10012FA35           0012FA55         0012FA45         0000000           0012FA55         10012FA35         10012FA35           0012FA54         0000000         0000000           0012FA54         0000000         0000000           0012FA54         00000000         0000000           0012FA54         00000000         0000000           0012FA54         00000000         00000000           0012FA54         00000000         00000000	1月2日 1月2

> OllyDbgで ソフトの"CrackMe0x01. exe"を開きます.

試用期限のチェックは起動直後に行われるので、 F9キーを押す前にAPI関数(GetLocalTime)に ブレークポイントを設定します.

F9キーを1回押します.

> "004011F5" でブレークしました.

この "GetLocalTime"を実行する前に,不要な ブレークホペントを一時的に無効化すると,思わぬ個所で ブレークしないので,スムース、なトレースが出来るようになります.

¥	OllyDbg -	CrackMe0	0x02.exe					
	ファイル(E)	表示(⊻)	解析( <u>D</u> )	フ・ラグイン	オプション(エ)	ウィンドウ( <u>W</u> )	ヘルプ(円)	
P	∍≪× ►	11 4 4	전 11 - 기	→ L E	мтwнс	/ K B R S		
	_							
	CPU -	メイン スレット	`, モジュール	/ CrackMe	0			
	アト・レス	Hexy >7	D	ディスアセン	7*₩	^		
	004011F5	. E8 8	32740000	call	<pre><jmp &kernei<="" pre=""></jmp></pre>	_32.GetL		
	004011FA	. 8U4	o FU	lea	eax, Ldword	ss:ebp-		
	004011FF	F8 3	3B000000	call	0040123F			

#### >Alt+Bキーを押します.

ハイライトしている"004011F5"以外を一時的に無効化します.

	рор есх	
B ブレークホ・イント		
アドレス モジュール アクティブ		ディスアセンブル ^
004011F5 004058E0 0040867C CrackMe0 設定済 設定済		call <jmp.&kernel32.getlocaltime call <jmp.&kernel32.getlocaltime jmp near [dword ds:&lt;&amp;KERNEL32.G</jmp.&kernel32.getlocaltime </jmp.&kernel32.getlocaltime 
0040121E         6A         00           00401220         6A         10           00401222         FF75         08           00401225         E8         18750000           0040122A         -         EB         0A           0040122A         -         EB         0A           0040122A         -         EB         0A	push 0 push 10 push Edword s call <jmp.&u jmp short 0</jmp.&u 	ss:ebp+8] SER32.SendMe J401236
フ <sup>°</sup> レークホ°イント: CrackMe0.004011F5		停止中

🔆 OllyDbg – Crac	ckMe0x02.exe		
ファイル(E) 表示	ヾ <u>(V) 解析(D</u> ) プラグイン	オプション(エ) ウィンドウ(W) ヘルプ(H)	
	₩## ¥## ₩ <b>L</b> E	M T W H C / K B R S 📰 🏋 ?	
<mark>C</mark> CPU - メイン	スレット゛, モジュール CrackMe	e0 🔳 🗖 💟	
アドレス Hex	タ`ンプ テ`ィスアセン	/ブ <b>ル</b> <mark>^</mark>	
004011F5	E8 82740000 call	<pre><jmp.&kernel32.getl< pre=""></jmp.&kernel32.getl<></pre>	
004011FA   .	50 ru Tea	eax, Ldword SS;edp-	
004011FE	̧ 3B000000 call	0040123E	
00401203    .	59 pop	ecx	
B ブレークホ・イント			
7ドレス モジュ	1/ アクティフ *	<b>ディスアセンブル</b>	^
004011F5 Crac	kMe0 設定済	call <jmp.&kernel32.g< td=""><td>etLocalTime</td></jmp.&kernel32.g<>	etLocalTime
004038E0 Crac	kme0 屏际/F	imp near dword ds:	KERNEL32.G
			×
0040121Ĕ	6A 00 push	Ŏ	
00401220	6A_10 push	10	
00401222	FF75-08 push F8-18750000 call	Ldword ss:ebp+8] Kimp &USER32 SendMa	
0040122A	EB 0A jmp	short 00401236	
フ <sup>・</sup> レーク林 <sup>®</sup> イント: Crack	kMe0.004011F5		停止中

004011F4 51 push ecx ; pLocaltime 004011F5 E8 82740000 call <jmp.&KERNEL32.GetLocalTime> 004011FA 8D45 F0 lea eax, [dword ss:ebp-10]

0012FA40 0012FA44 ¥pLocaltime = 0012FA44

(この値は環境によって変わります)



CONVD12 - CrackMe0x02.exe - [CPU - オンスレッド、モンコール CrackMe0]
 C ファイルとD 表示(x) 解析(x) アンライン オブション() ウィルドウ(x) ヘルン(D)
 Set (x) 解析(x) アンライン オブション() ウィルドウ(x) ヘルン(D)
 Set (x) 解析(x) アンライン オブション() ウィルドウ(x) ヘルン(D)
 Set (x) Null (x) オジョン() ウィルドウ(x) ヘルン(D)
 Set (x) Null (x) オジョン(x) ウィルン(D)
 Set (x) Null (x) オジョン(x) クィルン(D)
 Set (x) Null (x) オジョン(x) クィルン(D)
 Set (x) Null (x)

>スペースキーで解除済みにします.

(DELキーを使用するとブレークポイントが削除されます)

[B]画面は消してもOK.

[C]画面を拡大します.

- > ディスアセンブル欄 ラベル pLocaltime の内容を引数として call命令に 渡しています.
- > スタックウィント・ウを見れば、その引数が "0012FA44"で あることが判ります、 "GetLocalTime"が実行されると 現在日時がこの領域に格納されます。
- >この領域をダンプウィンドウに表示させます.

スタックウィント・ウの "0012FA44"の個所を右クリックします.

[ダンプ画面へ]をクリックします.

> ダンプウィンドウが Hex/ASCII(16バイト)表示になって いたので, Short/符号あり 10進表示に変更します.

ダンプウィント・ウで右クリック→[Short]→[符号あり 10進] をクリックします.

7F 1/2         18-bit short         7/1/2         16-bit short         7/1/2         16/2         1/	004001276 004001277 004001274 004001274 004001274 1 F75 005 004001274 1 G mp.AUSER32.MessageBoxA> 004001275 004001275 1 G mp.AUSER32.MessageBoxA> 004001275 1 G mp.AUSER32.MessageBoxA> 0 G mp.AUSER3	Test = -2005%T077:       EIP 0040115 CrackMe0.0040115         bDmccr       C ES 0023 32bit 0 (FFFFFFF)         HDaram = 0       C ES 0023 32bit 0 (FFFFFFF)         W2aram = 0       X 0 SS 0023 32bit 0 (FFFFFFF)         Wearam = 0       X 0 SS 0023 32bit 0 (FFFFFFFF)         Wearam = 0       X 0 SS 0023 32bit 0 (FFFFFFFF)         Wearam = 0       X 0 SS 0023 32bit 0 (FFFFFFFF)         Wearam = 0       X 0 SS 0020 32bit 0 (FFFFFFFF)         Wearam = 0       X 0 SS 0020 32bit 0 (FFFFFFFF)         Wearam = 0       X 0 SS 0020 32bit 0 (FFFFFFFF)         Wearam = 0       0 0 7 Abi7-: HF00000 (FFF)         Wmd       StoreMassanceA         StoreMassanceA       0 0 0 7 Abi7-: HF00000 (FFF)         Pault = 0; Case 10       FFL 00000246 (N0.N.B.E.BE.NS.PE.GE.LE)         Storematical aloas       Storematy -222 FFFF 0000078 00580005         Storematy -222 FFFF 0000078 00580005       Storematy -222 FFFF 00000078 00580005         Storematy -222 FFFF 00000078 00580005       Storematy -222 FFFF 00000078 00580005         Storematy - 00000000000000000000000000000000000
0012FA54 0012FA	71/1/2 16-bit short	▲ 71 <sup>1</sup> レス 値 コメント ▲
	0012FA54 -1408 18 -30324 30679 728 68 2.72 0 0012FA54 12345 0 184 -30424 30679 728 68 2.72 0 0012FA54 12345 0 -1848 18 4471 64 -21555 -9030 0012FA54 12345 30672 4471 64 720 68 2.73 8 0012FA54 12345 30672 4471 64 720 68 2.73 8 0012FA54 12345 106 2.0 0 7 0 720 8 0012FA54 -24216 106 2.0 0 7 0 0 0 0 0 0012FA54 -24216 106 2.0 0 0 0 0 0 0 0012FA54 -1380 18 -10912 30671 -1068 18 1111 30674	0012FA44 006A168 0012FA44 006A168 0012FA46 006A1788 0012FA46 006A1788 0012FA50 006000001 0012FA50 006000001 0012FA54 -0012FA80 0012FA54 77CF374 RETURN大: USER32.77CF3734 0012FA54 000000333 ~ 0012FA64 00000333



> F8キーを押して、 "004011F5" のcall命令を実行します.



> ダンプウィンドウから



この配列は、C言語のSYSTEMTIME構造体の構成で決められています.

004011F5	E8 82740000	call	<jmp.&kernel32.getlocaltime></jmp.&kernel32.getlocaltime>
004011FA	8D45 F0	lea	eax, [dword ss:ebp−10]
004011FD	50	push	eax
004011FE	E8 3B000000	call	0040123E
00401203	59	рор	ecx
00401204	85C0	test	eax, eax
00401206	75 2E	jnz	short 00401236
00401208	6A 30	push	30
0040120A	68 51914000	push	00409151
0040120F	68 2C914000	push	0040912C
	;Text = "2	2005年	10月12日以降は起動できません。"
00401214	FF75 08	push	[dword ss:ebp+8]
00401217	E8 20750000	call	<jmp.&user32.messageboxa></jmp.&user32.messageboxa>



> チェックルーチンのトレースを進めます.

call命令直後に比較・分岐命令がある場合はcall命令の 内部にチェックルーチンが潜んでいる可能性が高く,次の call命令 "call 0040123E"の内部を精査する必要が あります.

そこで, F8キーを2回押します.

"004011FE"まで進みました.

> この call命令を調べるために, 詳細ステップ実行します.

F7キー(詳細ステップ実行)を押します.

"0040123E"に制御が移りました.

> "call 0040123E"の内部を精査します.

現在年のチ	·ェッ <b>ク</b> (1)			> 現在年のチェック(1)
0040123E	55	push	ebp	
0040123F	8BEC	mov	ebp, esp	
00401241	8B45 08	mov	eax, [dword ss:ebp+8]	
00401244	66:8138 D507	cmp	[word ds:eax], 7D5	+ 現在年>2005(7D5h)の時,
00401249	73 07	jnb	short 00401252	+ "00401252"にジャンプします.
0040124B	B8 01000000	mov	eax, 1	+ 現在年≦2005(7D5h)の時, 戻り値 1 が返ります.
00401250	5D	рор	ebp	
00401251	C3	retn		+ (試用期限内)
現在年のチ	·ェッ <b>ク</b> (2)			> 現在年のチェック(2)
00401252	66:8138 D507	cmp	[word ds:eax], 7D5	+ 現在年≦2005(7D5h)の時,
00401257	76 04	jbe	short 0040125D	+ "0040125D" にジャンプします.
00401259	33C0	xor	eax, eax	+ 現在年>2005(7D5h)の時, 戻り値 0 が返ります.
0040125B	5D	рор	ebp	
0040125C	C3	retn		+ (試用期限切れ)
現在月のチ	יביי <b>ל</b> (1)			> 現在月のチェック(1)
0040125D	66:8378 02 0A	\ cmp	[word ds:eax+2], 0A	+ 現在月>10(0Ah)の時,
00401262	73 07	jnb	short 0040126B	+ "0040126B"にジャンプします.
00401264	B8 01000000	mov	eax, 1	+ 現在年≦10(0Ah)の時, 戻り値 1 が返ります.
00401269	5D	рор	ebp	
0040126A	C3	retn		+ (試用期限内)
現在月のチ	·ェック(2)			> 現在月のチェック(2)
0040126B	66:8378 02 0A	\ cmp	[word ds:eax+2], 0A	+ 現在月≦10(0Ah)の時,
00401270	76 04	jbe	short 00401276	+ "00401276"にジャンプします.
00401272	33C0	xor	eax, eax	+ 現在月>10(0Ah)の時, 戻り値 0 が返ります.
00401274	5D	рор	ebp	
00401275	C3	retn		+ (試用期限切れ)

現在日のチ	ביי <b>ל</b> (1)		
00401276	66:8378 06 0C	cmp	[word ds:eax+6], 0C
0040127B	77 07	ja	short 00401284
0040127D	B8 01000000	mov	eax, 1
00401282	5D	рор	ebp
00401283	C3	retn	
現在日のチ	ェッ <b>ク</b> (2)		
00401284	33C0	xor	eax, eax
00401286	5D	рор	ebp
00401287	C3	retn	

現在年のチ	ェッ <b>ク</b> (1)		
0040123E	55	push	ebp
0040123F	8BEC	mov	ebp, esp
00401241	8B45 08	mov	eax, [dword ss:ebp+8]
00401244	66:8138 D507	cmp	[word ds:eax], <mark>7D5</mark>
00401249	73 07	jnb	short 00401252
0040124B	B8 01000000	mov	eax, 1
00401250	5D	рор	ebp
00401251	C3	retn	
現在年のチ	ביש <b>ל</b> (1)		

	· · · · · · · · · · · · · · · · · · ·			
0040123E	55	push	ebp	
0040123F	8BEC	mov	ebp, esp	
00401241	8B45 08	mov	eax, [dword ss:ebp+8]	
00401244	66:8138 D507	cmp	[word ds:eax], 7D5	
00401249	73 07	jnb	short 00401252	=
0040124B	B8 01000000	mov	eax, 1	
00401250	5D	рор	ebp	
00401251	C3	retn		

- > <mark>現在日のチェック(1)</mark> + 現在日>12(0Ch)の時,
- + "00401284"にジャンプします.
- + 現在日≦12(0Ch)の時, 戻り値1 が返ります.
- + (試用期限内)
- > 現在日のチェック(2)
- + 現在月>10(0Ah)の時, 戻り値 0 が返ります.
- + (試用期限切れ)

> クラックパッチの作成(1)

⇒ チェックルーチンの最初の比較命令のオペランドを書き換えると 試用期限が延長されるようです.

7D5h→0FFFFh に変更します.

] > クラックパッチの作成(2)

⇒ チェックルーチンの最初の分岐命令のオペランドを書き換えると 常に試用期限内と判断されるようです.

00401252→0040124B に変更します.

> では、OllyDbgでの実際の書き換えを行います.

# > クラックパッチの作成(1)

書き換えたい行 "00401244"をダブルクリックします.

(同行をクリックしてSpaceキーを押してもOK)



アセンブル: 00401244	
cmp [word ds:eax], OFFFF	•
☞ 余ったバイトをNOPで埋める	アセンフル キャンセル

オプション(1) ウィンドウ(10)

表示(V) 解析(D)

17" 177477"

> 7D5→0FFFF に変更します.

16進数の表記で最初がA~Fで始まるときは"0"を頭に 付けます。 (末尾の"h"は不要)

[アセンブル]ボタンを押します.

> メモリー内の書き換えが完了しました.

00401235 55	push ebp	EAX 0012FA44
0040123F . 8BEC	mov ebp, esp	ECX 000001D4
00401241   . 8645 08	mov eax, [dword ss:ebp*8]	EDX 00000029
00401244 00-0130 111	iph chart 00401252	ESP BALLER
0040124B B8 01000000	mov eax. 1	FBP 0012FA54
00401250   . 50	pop ebp	ESI 00401177 CrackWe0.00401177
00401251 . 03	retn	EDI 0012FABC
00401252 > 466:8138 D507	cmp [word ds:eax], 705	E1P 00(0122F CrackWay 0040122F
00401257   76 04	jbe short 00401250	
00401259 . 3300	xor eax, eax	C = 0 = ES = 0023 = 326 + 1 = 0 (FFFFFFFFF)
00401256	pop ebp	
00401250 S 66-9379 02 0A	cmp [word_detext2]_0A	7 1 DS 0023 32Dit 0(FFFFFFF)
00401262 73 07	inb short 0040126B	S 0 FS 003B 32bit 7FFDE000(FFF)
00401264 B8 01000000	mov eax, 1	T 0 G\$ 0000 NULL
00401269 5D	pop ebp	D 0
0040126A . C3	retn retn	0 0 5XFi5-: ERROR_SUCCESS (00000000)
00401268    > 66:8378 UZ UA	cmp Lword ds:eax+2], UA	EFL 00000248 (NO.NB.E.BE.NS.PE.GE.LE)
00401270	Jbe short 00401276	ST0
00401272 - 3360	xor eax, eax	ST1 empty -222 EFEE 00500060 00700070
00401275	retn	S12 empty -222 FFFF 00000038 00480065
00401276 > 66:8378 06 0C	cmp [word ds:eax+6], 0C	ST3 empty -??? FFFF 00000078 009800D9
0040127B 77 07	ja short 00401284	SI4 empty -??? FFFF 0F7FA3E6 8F92ACE9
0040127D . B8 01000000	mov eax. I	ST5 empty 1.0000000000000000000
00401252-00401252	Linan - Ilin - I	S15 empty 1.00000000000000000
7h117 18-hit short	The subscription of the su	A 71/17 (# 11/04
0012EAAA 2007 9	1 3 15 41 1 468	0012EA3C 00401203 RETURN先, CrackWall 00401203 from CrackWall
0012FA54 -1408 18 -3092	24 30671 658 12 273 0	0012FA40 0012FA44
0012FA64 12345 0	0 0 4471 64 -21555 -9030	0012FA44 000907D7
0012FA74 0 0 -134	48 18 4471 64 -1300 18	0012FA48 00030001
0012FA84 14149 30672 447	71 64 658 12 273 0	0012FA4C 0029000F
0012FA94 12345 0	0 0 2/3 0 658 12	0012FA50 01D40001
0012FAR4 -3064 104 2		0012FA34 F0012FA80 0012EAE9 770E9724 DETUDM#+ USED22 770E9724
0012FACA 0 0		0012FA30 770F0734 RETORIZE: USER32.770F0734
0012FAD4 -1380 18 -1091	12 30671 -1068 18 1111 30674	0012FA80 00000111
0012FAE4 14184 30672	0 0 -1228 18 13713 30672	0012FA64 00003039
0012FAF4 0 0 447	71 64 658 12 273 0	0012FA68 0000000
0012FB04 12345 0	0 0 -3644 104 -1220 18	0012FA6C 00401177 CrackMe0.00401177
001218141 1096 18 1684		I UUI ZEAZUH DUBAABODI

ヘルプ(日)

1ddr

5 1 7



逆アセンブルウィンドウで右クリック [実行ファイルlこコピー]→[全ての変更箇所]をクリックします.



> 出てきたダイアログの[全てのコピー]をクリックします.



🔆 OllyDbg - CrackMe0x02.exe	
ファイル(E) 表示(V) 解析(D)	ブラグイン オブション(エ) ウィンドウ(W) ヘルプ(Ŀ)
CPU – ንተን አሁቃኑ , ቺንጌ–ዙ C	rackMe0
アドレス Hexダンプ	<u> </u> ディスアセンブ <b>ル</b>
0040123E 0040123F 00401241 8BEC 00401241 8B45 08	push ebp, esp mov ebp, esp mov eax, [dword ss:eb
00401244 66:8138 FFFF 00401249 ~ 73 07 00401248 . B8 01000000 . B8 01000000	inb short 00401252 mov eax, 1
00401250 - 5D 00401251 - C3 00401252 > 66:8138 D507	pop epp retn cmp [word_ds:eax], 7D
0 D ファイル D:¥_DVD_tagawa¥_実践	ジフトクラック初級編¥CrackMe0x02.exe
00000844   66:8138 FFFF	cmp [word ds:eax], 0FFF
00000849 ~ 73 07	inb short_00000852
00000848 88 0100000	mov eax, l
0 00000851 C3 0 00000852 66: 771ル変更済	
9         00000857         76           9         00000859         331         2         7           0000085B         5D         0         2         7           0000085C         C3         7         7           0000085D         66:         7         7	ァイル 'D.¥_DVD_tagawa¥ 実践ソフトクラックヤフ級編¥CrackMe0x02.exe' のダンプはオリジナルと異なりま ち。変更済みファイルを保存しますか? もし答えが「はい」ならば、ファイル保存名を指定して下さい。「いい こ」ならば、既に行われた総ての変更が取り消されます。
00000864 B8 00000869 5D 0000086A C3	はい(Y) いいえ(N) キャンセル
00000808 06:0370 02 0A 00000870 76 04 00000872 33C0 00000874 5D 00000875 C2	ibe short 00000876 xor eax, eax pop ebp
00000073 00	

~として保存				? 🛛
保存する場所①:	🗀 _実践ソフトクラックネフル級編	•	+ 🗈 💣 🎞 🕇	
した 最近使ったファイル	■CrackMe0x01.exe CrackMe0x02.exe CrackMe0x02.exe CrackMe0x03.exe			
デスクトップ				
ک جر انجع				
ער בארד אב ארבארד				
द्व र्न २०१७-७				
	ファイル名(N): CrackMe0x02_crack.exe ファイルの種類(T): 実行可能ファイル(*.exe)		•	保存(S) キャンセル



> [D]画面の[×]を押すと, [ファイル変更済み]のダイアログが 出てきますので, [はい]ボタンを押します.

> リネームして[保存]を押します.

(上書き保存をしても, OllyDbgではバックアップファイルを 作成します)

これで、OllyDbgによるクラックは完了です.

> 次に quickbe でクラックパッチファイルを作成します.

設定をして, [作成]を押します.

> パッチファイル作成が完了しました.



\* FileSize: 48640 bytes \* LastMod.: 2005/03/12 22:41:44

00000847: D5 FF 00000848: 07 FF

# > クラックパッチの作成(2)

こちらもクラックハッチの作成(1)と同様の方法で行います.

結果は以下の通りでした.

00000849: 73 EB 0000084A: 07 00

次回は 3) ID・パスワード型 です

印刷時のページレイアウトは考慮していませんので悪しからず.

以上

3) ID・パスワード型

アプリケーションによっては、登録の際にパスワードの他にユーザーIDなど、 ユーザー固有の情報の入力を求めるものがあります。 これらのプロテクションはユーザーID毎に正当なパスワードを生成し、入力 されたものと比較するように設計・実装されています。

- \* 一般的なチェックルーチン
- a. ユーザーIDとユーザーが入力したパスワート(文字列)を取得する
- b. ユーザーIDからパスワートを生成し、入力されたパスワートとの妥当性を チェックする(ここが特定すべき個所)
- c. チェックの結果に応じてメッセージを表示する

エディットボックスから文字列を取得するにはWindowsが提供するAPI関数を 用いています。

\* 文字列取得: API(Application Program Interface) 関数

GetWindowText, GetDlgItemText, GetDlgItemIntの3種類です

これらのAPI関数が呼び出される瞬間を捉えることにより, チェックルーチンの個所を 特定します.

- \* チェックルーチンのトレースの第一歩
- a. OllyDbgでソフトの実行ファイルを開きます
- b. F9キーを押して、ユーザーIDとパスワート、入力のエディットホックスが出たら、 ユーザーIDとフェイクハ<sup>°</sup>スワート、を入れておきます ([OK], [登録]等のホックンは押しません)
- c. 文字列取得のAPI関数Iこブレークポイントを設定します
- d. ここで, [OK], [登録]等のボタンを押します
- e. API関数が実行される直前で停止(ブレーク)します
- f. この先にチェックルーチンが存在するはずです
- g. これからは、1命令ずつ実行(F8キーを押す)して精査を行っていきます

このパスワードを生成するルーチンを解析すれば, 条件を満たすパスワードを作り出す Key Generator を作成できるということです. ただし, これには アセンブリ言語と C言語の高度な知識と, C言語のコンパイラが必要になります.

それでは"CrackMe0x03. exe"を解析してみましょう.



> OllyDbgで ソフトの"CrackMe0x03. exe"を開きます.

F9キーを1回押します.

ューサーID:s\_tagawa シリアルナンハー:98765432 を入力します (ここではパスワードではなくシリアルナンハーとなっています)

([OK], [登録]等のボタンは押しません)

#### 🍓 OllyDbg - CrackMe0x03.exe - [CPU - パンスレット', モジュール CrackMe0] 💽 ファイル(E) 表示(V) 解析(D) ブラグイン オプション(D) ウィントウ(W) ヘルプ(E)

アト・レス	Hexダンプ	ティスアセンフッル	<u>م</u>	レジスタ (FPU) く
004011FA	- E8 DF750000	call <jmp.&user32.getdlgitemtexta></jmp.&user32.getdlgitemtexta>	LGetDlgItemTextA	EAX 0000000
004011FF 00401202 00401204 00401209 0040120F 00401210 00401215 00401218	. 83F8 08 75 64 68 FF000000 8085 00FEFFI 50 68 EE030000 FF75 08 E8 C1750000	cmup eax,8 jnz short 00401268 push 0FF lea eax,[dword ss:ebp-200] push eax push eax push [dword ss:ebp+8] call < jmp,&USER32.GetDlgItemTextA>	Count = FF (255.) Buffer ControlID = 3EE (1006.) hMmd ListHightemInstA	ECX 001271300 EDX 001200400 EBX 00121340 EBP 00127140 EBP 00127140 ESI 004217140 EDI 001271338 EDI 001271338
0040121D 00401227 00401222 00401224 0040122A 00401228 00401230 00401230 00401231 00401239 00401239 00401235 0040123F 00401240 00401240 00401244 00401248 00401248	. 8BD8 . 83FB 08 . 75 44 . 8D85 00FFFFI . 50 . 88D85 00FFFFI . 50 . 88D8 . 8D85 00FFFFI . 50 . 88D8 . 75 24 . 6A 00 . 68 47914000 . 68 479140000 . 68 479140000 . 68 479140000 . 68 479140000 . 68 479140000 . 68 479140000 . 68 4791400000 . 68 4791400000000000000000000000000000000000	mov         ebx, eax           cmp         ebx, 8           inz         short 00401268           lea         eax, [dword ss:ebp-200]           push         eax           call         00403090           pop         ecx           mov         ebx, eax           lea         eax, [dword ss:ebp-100]           push         eax           call         0040150           pop         ecx           call         00401150           pop         ecx           cmp         ebx, eax           push         00408147           push         004081262           push         004081272           push         004081272	L <sup>Arg1</sup> Crash Mad. 00403080 C <sup>Arg1</sup> Had. 00401150 Style = MB_OKIMB_APPLMODAL Title = CrackMedx08 Text = ご言語なりかとうございます Homore at	CT 0 ES 0023 32bit 0(FFFFFFF) P 1 CS 001B 32bit 0(FFFFFFFF) Z 1 DS 0023 32bit 0(FFFFFFFF) Z 1 DS 0023 32bit 0(FFFFFFFF) S 0 FS 0038 32bit 7FFD0000(FFF) T 0 GS 0000 NULL 0 0 0 7Zh17-: FROM SUCCESS (00000000) FFL 00000246 (NO.NB.E.BE.NS.PE.GE.LE) S10 empty -222 FFF 001 C0F 001 001 S11 empty -222 FFF 001 C0F 001 001 S13 empty -222 FFF 001 C0F 001 001 S13 empty -222 FFF 001 C0F 001 001 S13 empty -222 FFF 001 C0F 001 001 S14 empty -222 FFF 001 C0F 001 001 S15 empty -222 FFF 001 C0F 001 001 S15 empty -222 FFF 001 C0F 001 001 S15 empty -222 FFF 001 001 S15 empty -222 FFF 001 001 S15 empty -222 FFF 001 001 S16 empty -222 FFF 001 001 S16 empty -222 FFF 001 001 S16 empty -222 FFF 001 0000000
00401255	(imp &USER32 Get	DigitemTextA	-messageboxa *	
		BISTOMOATA		FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 FCW 1372 Prec NEAR,64 Mask 1 1 0 0 1
71*12	Hex\$``\7°	ASCII	<u>^ アドレス</u>	値 コメント 🔨
00409000 00409020 00409030 00409030 00409040 00409060 00409060 00409070 00409080 00409080	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	10 00 00 06 <u>F4 2F 40 00</u> 00 05 <u>D0 37</u> P- e 55 40 00 00 06 <u>54 44 40 00</u> 00 06 M e • e. 15 88 47 40 00 00 0A <u>46 51 40 00</u> ILe • e 10 00 01 <u>18 55 40 00</u> 00 03 <u>96 50</u> • F 13 40 <u>00</u> 00 03 <u>E0 64 40 00</u> 00 00 <u>e.</u> . • t ce. 10 00 00 <u>86 38 40 00</u> 00 03 <u>24 55</u> • e. 13 40 <u>00</u> 00 <u>03 66 63 40 00</u> <u>18 26 50 00</u> <u>ILE • e.</u> 15 72 6C 61 6E 64 20 43 28 28 20 <u>14</u> • Borla 16 72 6C 61 6E 63 20 43 20 20 <u>10</u> • C	• e., 1\$7 • TDe •De 0012F820 eLoe. 0012F820 0000 0000 0000 0000 0000 0000 0000 0000 0000	2 000102C8   hHmd = 000102C8 (°crackme 0) 000003E0   Control 10 = 8E0 (1005.) 0012F9D0   Buffer = 0012F9D0 000000F4   Count = FF (255.) 00000004 00000004 0012F9A0 0012F9A0   S010 - 22FD0260   ♥

> 全てのAPI関数(GetDlgTextA)にブレークホペイントを設定します. (操作の詳細は省略します) 今回はAPI関数が "GetDlgTextA"なのに注意して 下さい.

アプリケーションのエディットボックスの[登録]ボタンを押します.

API関数が実行される直前(~004011FA~)でブレーク しました.

#### このAPI関数によって何が取得されるのでしょうか?

0012F8C0	000003EE	ControlID = 3EE (1006.)
0012F8C4	0012F8D0	Buffer = 0012F8D0
0012F8C8	00000FF	Count = FF (255.)



> スタックウィンドウに注目します.

Bufferが示すアドレス "0012F8D0"に取得した文字列が 格納されます.

この行を選択状態にして、右クリックします.

[ダンプ画面へ]をクリックします.

> API関数(GetDlgTextA)を実行します.

[登録]ボタンを押します.

F9キーを2回押します.

次のAPI関数(GetDlgTextA)が実行される直前 ("00401218")でブレークしました.

ダンプウィンドウのASCII欄を見てみます.

> "s\_tagawa"の文字列が表示されています.

> ブレーク前後のコートを見てみましょう.

ューサーID取	得·文字列長の	)比較		> ユーザーID取得・文字列長の比較
004011FA	E8 DF750000	call	<pre><jmp.&user32.getdlgitemtexta></jmp.&user32.getdlgitemtexta></pre>	+ ユーサ <sup>*</sup> ーID取得
004011FF	83F8 08	cmp	eax, 8	+ 文字列長が8文字以外は登録失敗
00401202	75 64	jnz	short 00401268	+ 登録失敗処理へ
シリアルナンバ・	ーの取得・文字の	列の比	較	> シリアルナンバーの取得・文字列の比較
00401204	68 FF000000	push	0FF ;Count = FF (255.)	
00401209	8D85 00FEFFF	F lea	eax, [dword ss:ebp-200]	
0040120F	50	push	eax ;Buffer	
00401210	68 EE030000	push	3EE ;ControlID = 3EE (1006.)	
00401215	FF75 08	push	[dword ss:ebp+8] ;hWnd	
00401218	E8 C1750000	call	<pre><jmp.&user32.getdlgitemtexta></jmp.&user32.getdlgitemtexta></pre>	+ シリアルナンバー取得(トレースの <mark>現在位置</mark> )
0040121D	8BD8	mov	ebx, eax	
0040121F	83FB 08	cmp	ebx, 8	+ 文字列長が8文字以外は登録失敗
00401222	75 44	jnz	short 00401268	+ 登録失敗処理へ
入力シリアルフ	ナンバーを数値に	変換		> 入力シリアルナンバーを数値に変換
00401224	8D85 00FEFFF	FF lea	eax, [dword ss:ebp-200]	
0040122A	50	push	eax ; /Arg1	+ 入力シリアルナンバーのアト゛レス
0040122B	E8 602B0000	call	00403D90	+ 入力シリアルナンバーを数値に変換
00401230	59	рор	ecx	
00401231	8BD8	mov	ebx, eax	+ 戻り値を ebxレジスタlこコピー



ここで "401230"にブレークポイントを仕掛けます. (左クリック+F2キー)

F9キーを2回押します.

**"00401218"でブレークしました.** 

0012F9D0 s\_tagawa....h•.

7 k b2         Hext* 27*         ASCII         7 k b2         164         322 k         ASCII           0012F900 73 5F 74 61 67 61 77 61 00 00 00 68 FA 12 00 5-tarawah*.         0012F800 73 5F 74 61 67 61 77 61 07 01 50 48 00 5A 00 04 FA 12 00 5-tarawah*.         0012F800 73 5F 74 61 77 61 07 01 50 48 00 5A 00 04 FA 12 00 5-tarawah*.         0012F800 73 5F 74 61 77 61 07 01 50 48 00 5A 00 04 FA 12 00 5-tarawah*.         0012F800 73 5F 74 61 76 10 70 150 48 00 5A 00 04 FA 12 00 5-tarawah*.         0012F800 70 12F800 0000000           0012F800 10 00 00 00 00 00 00 00 00 00 00 00 0	00401926         - EB 00430000 (241)           0040126A         - 66.30           0040126A         - 68.3014000 (241)           0040126A         - 68.5314000 (241)           0040126A         - 68.5314000 (241)           0040126A         - 68.55314000 (241)           0040127C         - 18.85750000 (241)           0040127C         - 18.86750000 (241)           0040127C         - 18.8750000 (241)           0040127C         - 58.60           0040127C         - 58.40           00401280         - F75.08           00401280         - 12.870000 (241)           75.7         10012F8C31-0012F800 (0012)           ecx=00000032	Comp. Mod. Cor. and Commercial geno short 10041288 00409180 00409183 [dword ss:ebor8] Comp.&USER32.MessageBoxA> short 00401288 0 [dword ss:ebor8] Cimp.&USER32.EredDialog> F8D0). ASCII 98765432*	Testvie = MB_OK MB_ICONEXCLAMATT Title = "CrackMe0x03" Test = ユーヴーIDまた(はシリア) Homor MessanaBoxA Result = 0; Case 10 (WM_CLOSE) HWnd 	EFL 000000 STO empty STI empty ST3 empty ST3 empty ST4 empty ST5 empty ST5 empty ST6 empty ST7 empty EST 0000 FCW 1372	22         FFFF 00FF00FF         00F00FF           -222         FFFF 00FF00FF         00F00FF           -222         FFFF 00FF00FF         00F00FF           -222         FFFF 000F000 0000000         00000000           -222         FFFF 0000000 0000000         00000000           -222         FFFF 00000000 000000000         00000000           -222         FFFF 00000000 000000000         000000000           -222         FFFF 00000000 0000000000         000000000           -222         FFFF 00000000 0000000000         000000000           -222         FFFF 00000000 000000000         000000000           -222         FFFF 00000000 000000000         000000000           -222         FFFF 00000000 000000000         000000000           -222         FFFF 00000000 0000000000         000000000           -222         FFFF 00000000 000000000         000000000           -222         FFFF 00000000 00000000000         000000000           -222         FFFF 00000000 0000000000         000000000           -222         FFFF 000000000 0000000000         000000000           -222         FFFF 000000000 0000000000000         000000000000000000000000000000000000	Z D 0 0 0 1
0012F800 [73 5F 74 61 67 61 77 61 00 00 00 00 8F FA 12 00 5 trazwar.ht. 0012F800 [06 F3 00 77 61 07 01 50 43 00 54 00 05 FA 12 00 5 trazwar.ht. 0012F800 [00 00 FD 77 00 00 00 00 70 40 66 00 01 00 00 00 0] = trazwar.ht. 0012F800 [00 00 FD 77 00 00 00 00 40 00 00 00 0] = trazwar.ht. 0012F800 [00 00 7D 76 00 00 00 40 00 00 00 0] = trazwar.ht. 0012F800 [00 00 7D 76 00 00 00 40 00 00 00 0] = trazwar.ht. 00 12F800 [00 00 7D 76 00 00 00 40 00 00 00 0] = trazwar.ht. 00 12F800 [00 00 7D 76 00 00 00 40 00 00 00 0] = trazwar.ht. 00 12F800 [00 00 7D 76 00 00 00 40 00 00 00 0] = trazwar.ht. 00 12F800 [00 00 7D 70 0] = trazwar.ht. 00 12F800 [00 00 7D 70 0] = trazwar.ht. 00 12F800 [00 00 0] = trazwar.ht. 00 12F800 [00	71'17 Hex9'27"	ASCII	~ 7F'VZ	値	1771	^
	$\begin{array}{c} 00127900 \ (73 \ 55^{-} 74 \ 61 \ 67 \ 81 \ 77 \ 61 \\ 00127960 \ (06 \ F8 \ 007 \ 77 \ 61 \ 07 \ 01 \ 51 \\ 00127400 \ (16 \ 00 \ 00 \ F0 \ 77 \ 61 \ 00 \ 00 \ 00 \ 00 \ 00 \ 00 \ 00$	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		0012F8D0 00000000 36373839 32333435 0012F900 00000004 73FD0260 00000005 73FD0490	ASCII "98765432" USP10.73FD0260 USP10.73FD0490	

## レシ、スタウィント、ウのEAXレシ、スタの値が

EAX 05E30A78 となっています.

文字列 "98765432"が数値に変換されているはずです.

EAX の数値の部分をダブルクリックします.

> やはり, 数値になっています.

Windows の電卓で計算して確認しても同様でした.

EAX を修正	
16進	05E30A78
符号あり	98765432
符号なし	98765432
文字 ¥×05 ¥×E3	X
OK	キャンセル

00401230	59	рор	ecx	; 0012F8D0
00401231	8BD8	mov	ebx, eax	
00401233	8D85 00FFFFF	F lea	eax, [dword ss	:ebp-100]
00401239	50	push	eax	; /Arg1
0040123A	E8 11FFFFFF	call	00401150	
0040123F	59	рор	ecx	
00401240	3BD8	cmp	ebx, eax	
00401242	75 24	jnz	short 00401268	
00401244	6A 00	push	0	
00401246	68 47914000	push	00409147	
0040124B	68 2C914000	push	0040912C	
00401250	FF75 08	push	[dword ss:ebp <sup>.</sup>	+8] ;hOwner
00401253	E8 92750000	call	<jmp.&user32.i< td=""><td>MessageBoxA&gt;</td></jmp.&user32.i<>	MessageBoxA>



EAX を	修正			×
16)進			01 AD8 BD7	
符号あり	J		28150743	
符号なし	,		28150743	
文字	¥×01	¥×AD	¥x8B ¥xD7	
	ок		キャンセル	

00401242	75 24	jnz	short 00401268
00401244	6A 00	push	0
			$\downarrow$
00401242	EB 00	jmp	short 00401244
00401244	6A 00	push	0

00401242	75 24	jnz short 00401268
00401244	6A 00	push 0
		$\downarrow$
00401242	90	nop
00401243	90	nop
00401244	6A 00	push 0

> 以降のコードを見てみましょう.

- + (トレースの<mark>現在位</mark>置)
- + 入力されたユーサーIDの格納アドレスを eaxレジスタへコピー. + ユーサーIDから正規シリアルナンバーを生成します.
- + スタックから値を取り出して ecxレジスタに書き込みます.
- + シリアルナンバーの入力値と生成値を比較しています.
- + 登録失敗のメッセージを表示へジャンプします.

+;Text = "登録ありがとうございます。"

+ 登録成功のメッセージを表示します.

 > 以上のことから、 "call 00401150" 命令の戻り値 (eaxレシズタに返されます)を調べれば、正規シリアル ナンバーが判明します。

そこで, "004123F" にプレークポイントをさらに仕掛けます. (左クリック+F2キー)

F9キーを1回押します.

″0040123F″でブレークしました.

EAXレシ、スタの値"01AD8BD7"を10進数に変換すれば, 正規シリアルナンバーが得られます.

> EAX の数値の部分をダブルクリックします.

ユーサ<sup>\*</sup>ーID: s\_tagawa

シリアルナンバー: 28150743

Windows の電卓で計算して確認しても同様でした.

# > クラックパッチの作成(1)

+ 登録失敗のメッセージを表示へジャンプします.

この条件ジャンプを次の行へ無条件ジャンプさせます.

# > クラックパッチの作成(2)

+ 登録失敗のメッセージを表示へジャンプします. この条件ジャンプを何もしないに変更します.

00401242	75 24	jnz short 00401268
00401244	6A 00	push 0
		$\downarrow$
00401242	<mark>74</mark> 24	je short 00401268
00401244	6A 00	push 0

+ 登録失敗のメッセージを表示へジャンプします.

この条件ジャンプを等しければ分岐に変更します.

> 左記のクラッチパッチが出来ました.

ュ−ザ−ID: 任意の 8 桁の文字列

シリアルナンバー:任意の8桁の文字列

を入力するとが条件です.

ただし、クラックハッチ(3)の場合は入力シリアルナンバーと

生成シリアルナンバーが不幸にして一致すると登録失敗に

> **クラックハ<sup>°</sup>ッチ**(1)

00000842: 75 EB 00000843: 24 00

<sup>+ ;</sup>Title="CrackMe0x03"

FILENAME CrackMe0x03.exe \* FileSize: 49152 bytes \* LastMod.: 2007/09/05 00:00:00 \*: 00000842: 75 90 00000843: 24 90 FILENAME CrackMe0x03.exe \* FileSize: 49152 bytes \* LastMod.: 2007/09/05 00:00:00 00000842: 75 74

0040123A E8 11FFFFFF call 00401150

終わります.

クラックパッチは書き換えるバイト数をできだけ

少なくするのが美しいといわれています.

> さて, 解析を少し前に戻します.

+ ユーサーIDから正規シリアルナンバーを生成します.

> ここに注目すると、この call命令の先に Key Generator のルーチンが隠されているようです.

この部分のコートを分析して(勿論, アセンブリ言語を), 『C言語』等の高級言語で記述し、コンパイラで実行ファイルを 作成すれば Keygen が出来ます.

印刷時のページレイアウトは考慮していませんので悪しからず.

以上