

Target : D4ph1's crackme
Level : 1/10
Lang. : asm32
Tools : OllyDBG
Cracker : allko

Introduction :

حسننا هذا ثاني درس لي ! lol ارجوا ان ينال رضاكم...وكما الدرس الاول ففي هذا الدرس سيكون مفصلا.
يرجى ملاحظة ان هذه الوثيقة تتبع ال open resources وانها تنشر تحت اتفاقية GNU
Let's crack it !!

Here we go :

- 000

قم بتشغيل الملف (ال crackme) ودقق النظر ...يبدو انه لا يقبل باي اسم اقل من 4 احرف ...ادخل أي رقم واضغط على check...لاحظ الرسالة التي تظهر عند ادخال سيريال خطأ...
اضغط على help لتقراء القواعد ruez المطلوب عمل keygen بالاضافة الى كشف العبارة السرية secret string .

-001

قم بفحص الملف باستخدام PEID لمعرفة ما اذا كان مضغوطا packed ...جيد فهو neither packed nor coded
by any HLL وكما تعرفون فحيث انه غير مبرمج بلغات المستوى العلوي HLL أي انه بالاسمبلي فهذا يسهل العمل...

The screenshot displays the OllyDBG interface with three main panes:

- Assembly Pane (Left):** Shows assembly instructions for the function `D4ph1_<ModuleEntryPoint>`. Instructions include `PUSH 0`, `CALL <JMP.&kernel32.GetModuleHandleA>`, `MOV DWORD PTR DS:[403344],EAX`, `CALL <JMP.&kernel32.GetCommandLineA>`, `MOV DWORD PTR DS:[403348],EAX`, `PUSH 0`, `PUSH 0`, `PUSH 66`, `PUSH DWORD PTR DS:[403344]`, `CALL <JMP.&user32.DialogBoxParamA>`, `PUSH EAX`, `CALL <JMP.&kernel32.ExitProcess>`, `MOV EBX,17`, `MOV EAX,20`, `XOR EAX,0C`, `SUB EAX,17`, `SUB BYTE PTR DS:[EBX+4031C6],AL`, `DEC EBX`, `JNZ SHORT D4ph1_<ModuleEntryPoint>+00401042`, `SUB BYTE PTR DS:[EBX+4031C6],AL`, `RETN`, `PUSH EBP`, `MOV EBP,ESP`, `CMP DWORD PTR SS:[EBP+C],110`, `JNZ SHORT D4ph1_<ModuleEntryPoint>+004010A0`, `PUSH 3EA`, `PUSH DWORD PTR SS:[EBP+8]`, `CALL <JMP.&user32.GetDlgItem>`, `PUSH ECX`, `PUSH DWORD PTR SS:[EBP+8]`, `CALL <JMP.&user32.GetDlgItem>`, `PUSH 68`, `CALL <JMP.&user32.LoadIconA>`, `PUSH EAX`, `PUSH 0`.
- Registers Pane (Right):** Shows the state of CPU registers. `EAX` is `00000000`, `ECX` is `0012FFB0`, `EDX` is `7C90EB94` (ntdll.KiFa), `EBX` is `7FFD6000`, `ESP` is `0012FFC4`, `EBP` is `0012FFF0`, `ESI` is `FFFFFFFF`, `EDI` is `7C910738` (ntdll.7C91), `EIP` is `00401000` (D4ph1_<ModuleEntryPoint>). The `EFL` register shows `00000246` (NO,NB,E,B).
- Memory Dump Pane (Bottom):** Shows a hex dump and ASCII representation of memory starting at address `00403000`. The hex dump shows `43 72 61 63 68 60 65 23 31 20 42 79 20 44 34 70` and the ASCII representation shows `Crackme#1`, `hi.Great w`, `te!You pas`, `l the test`, `atching is`, `llowed. Ma`.

جيد قم بفحص البرنامج جيدا بدءا بـ OEP (original entry point) أي 401000 للأسفل...لاحظ جانب ال calls
comments وابحث عن أي شيء قد يثير الاهتمام هناك دالتان من نوع `SendMessageA` وكما هو
واضح فانهما ترسلان ما ندخله في المربعين (مربع اليوزر ومربع السيريال) الى البرنامج...في المثال الاول
استخدم ذلك المبرمج دالة `SendWindowTextA` لتؤدي نفس الوظيفة...حسننا فلنتابع...
ملاحظة : بإمكانك البدء بـ all referenced text strings → search for لكن حيث ان البرنامج مكتوب بالاسمبلي
فهو واضح والأسطر البرمجية قليلة فلا نحتاج الى البحث الاوتوماتيكي فيمكننا البحث يدويا بكل سهولة...

ها قد وصلنا...لعلك لاحظت وجود اربع دوال من نوع message box كل منها يحتوي نصا مختلفا عن الآخر...

004011C1	. ^76 E6	JBE SHORT D4ph1_-.004011A9	
004011C3	. 6A 00	PUSH 0	Style = MB_OK;MB_APPLMODAL
004011C5	. 68 00304000	PUSH D4ph1_-.00403000	Title = "Crackme#1 By D4ph1"
004011CA	. 68 00324000	PUSH D4ph1_-.00403280	Text = "♦0**F"
004011CF	. 6A 00	PUSH 0	hOwner = NULL
004011D1	. E8 CA000000	CALL <JMP.&user32.MessageBoxA>	MessageBoxA
004011D6	. 6A 00	PUSH 0	Style = MB_OK;MB_APPLMODAL
004011D8	. 68 00304000	PUSH D4ph1_-.00403000	Title = "Crackme#1 By D4ph1"
004011DD	. 68 13304000	PUSH D4ph1_-.00403013	Text = "Great work mate!You pa
004011E2	. 6A 00	PUSH 0	hOwner = NULL
004011E4	. E8 B7000000	CALL <JMP.&user32.MessageBoxA>	MessageBoxA
004011E9	. EB 2A	JMP SHORT D4ph1_-.00401215	
004011EB	. 6A 00	PUSH 0	Style = MB_OK;MB_APPLMODAL
004011ED	. 68 00304000	PUSH D4ph1_-.00403000	Title = "Crackme#1 By D4ph1"
004011F2	. 68 C8324000	PUSH D4ph1_-.004032C8	Text = "You did it!But the sec
004011F7	. 6A 00	PUSH 0	hOwner = NULL
004011F9	. E8 A2000000	CALL <JMP.&user32.MessageBoxA>	MessageBoxA
004011FE	. EB 15	JMP SHORT D4ph1_-.00401215	
00401202	. > 6A 00	PUSH 0	Style = MB_OK;MB_ICONHAND;MB_A
00401204	. 68 00304000	PUSH D4ph1_-.00403000	Title = "Crackme#1 By D4ph1"
00401207	. 68 20324000	PUSH D4ph1_-.00403220	Text = "The Serial is not corr
0040120C	. 6A 00	PUSH 0	hOwner = NULL
0040120E	. E8 8D000000	CALL <JMP.&user32.MessageBoxA>	MessageBoxA

الأولى تحتوي نصا مبهما (هل تعتقد انه هو العبارة السرية ؟ لقد حاولت وكانت محاولة فاشلة !!) تفحص الثانية والثالثة والرابعة...

العبارة التي ظهرت لنا هي الرابعة اذن بإمكانك وضع BreakPoint عليها لكن لن تستفيد كثيرا...دعنا نحلل البرنامج...نريد ان نعرف اين هو امر القفز الذي يؤدي الى هذه الدالة...

ملاحظة : تجد بجانب بعض الاوامر علامة مثل هذه > (وضعت دائرة حولها انظر الصورة) هذا يعني ان هذا الامر هناك jump او call تؤدي إليه...

قد تجد نفس العلامة لكن للأعلى ^ او للأسفل ...هاتان العلامتان توجدان بجانب القفزات فقط وهما تدلان على اتجاه القفزة أي هل تؤدي الى أمر بالأعلى او بالأسفل...

اذن اضغط على الامر المقابل لأول باراميتير في الدالة الرابعة أي امر push 10 عند عنوان 401200 فتلاحظ بقسم التعليقات بالاسفل جملة jump from 401194 حسنا لنذهب الى 401194 لنرى ما يوجد هناك..

حسننا انها JNZ ...والان انتظر قليلا لنفكر...يوجد فوق هذا الامر مجموعة اوامر ثم دالة SendDlgItemMessageA ثم مجموعة اوامر اخرى ثم دالة SendDlgItemMessageA...اذن؟

يقوم البرنامج باستقبال اليوزرنيم ويقوم بعمل بعض الحسابات عليه ثم يدخل السيريال ويعمل بعض الحسابات ومن ثم عملية مقارنة ومن ثم قفزة...

هذه هي الإجرائية العامة لكثير من اجرائيات التحقق protection procedures اذن علينا ان نبدا مباشرة بعد دالة SendDlgItemMessageA الاولى...اذذهب العنوان 40111F وضع BreakPoint وذلك بالضغط على F2 ...

00401109	. 68 4C334000	PUSH D4ph1_-.0040334C	
0040110E	. 6A 40	PUSH 40	lParam = 40334C
00401110	. 6A 00	PUSH 0	wParam = 40
00401112	. 68 EA030000	PUSH 3EA	Message = WM_GETTEXT
00401117	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	ControlID = 3EA
0040111A	. E8 8D010000	CALL <JMP.&user32.SendDlgItemMessageA>	hWnd
0040111F	. A2 0D344000	MOV BYTE PTR DS:[40340D],AL	SendDlgItemMessageA
00401124	. 3C 04	CMP AL,4	
00401126	. JBE D4ph1_-.0040124D	JBE D4ph1_-.0040124D	
0040112C	. 3C 06	CMP AL,6	
0040112E	. JNZ SHORT D4ph1_-.00401137	JNZ SHORT D4ph1_-.00401137	
00401130	. MOV BYTE PTR DS:[40340C],1	MOV BYTE PTR DS:[40340C],1	
00401137	. MOV EDX,EAX	MOV EDX,EAX	
00401139	. XOR EBX,EBX	XOR EBX,EBX	
0040113B	. XOR ESI,ESI	XOR ESI,ESI	
0040113D	. MOV AL,BYTE PTR DS:[ESI+40334C]	MOV AL,BYTE PTR DS:[ESI+40334C]	

-011

فنبداً بتتبع البرنامج ...

شغل البرنامج ولنقم بادخال أي اسم...allko...مثلا وكلمة السر 987654321 (افضل ادخالها معكوسة أي ابدأ ب9...اختر ما تريد لا فرق...)

واضغط على زر ها قد عدنا الى Olly عند نقطة التوقف...لاحظ امر mov وانظر الى اسفل لتجد ان قيمة ال ستنتقل الى ذلك العنوان...

00401109: . 30FB		JMP EL	
AL=05			
DS:[0040340D]=00			
Address	Hex	dump	
00403000	43 72 61 63 6B 6D 65 23		

وكما ترى فان قيمة al تساوي 5 ... لكن الاسم الذي ادخلناه مكون من 5 احرف....اذن هذه ال 5 هي عدد احرف اليوزرنيم...

ملاحظة : لا مانع من التأكد وذلك باغلاق البرنامج ثم اعادة فتحه وادخال يوزرنيم اخر اطول او اقصر(لكن 4 احرف عالاقل) للتأكد من ان تلك الـ 5 هي عدد الاحرف وليست قيمة ثابتة
اضغط F8 تنتقل الى امر المقارنة التالي حيث يقارن محتوى al بالقيمة 4 ثم بعده هناك امر قفز JBE أي اذا كان عدد الاحرف 4 او اقل فسقفز الى 40124D حسنا فلنذهب الى هناك لنرى ما يحدث...
ملاحظة : قد لا يهمننا الذهاب الى هناك كثيرا ففتلك الاجرائيات مخصصة في حال كانت اليوزرنيم اقل من 4 أي اقصد انه لا توجد أي خوارزمية تشفير هناك لكننا نذهب لكي نكتشف اكثر فهذه الامور ان لم تقندا اليوم ستفيدنا غدا...يجب عليك ان تستكشف كل امر في البرنامج الضحية حتى تصبح Cracker حقيقي...

0040124D	> 0AC0	OR AL,AL	<pre> Style = MB_OK;MB_ICONHAND;MB_A Title = "Crackme#1 By D4ph1" Text = "You must enter a Name" hOwner = NULL MessageBoxA </pre>
0040124F	.v75 15	JNZ SHORT D4ph1_-.00401266	
00401251	. 6A 10	PUSH 10	
00401253	. 68 00304000	PUSH D4ph1_-.00403000	
00401258	. 68 D4304000	PUSH D4ph1_-.00403004	
0040125D	. 6A 00	PUSH 0	
0040125F	.E8 3C000000	CALL <JMP.&user32.MessageBoxA>	
00401264	.vEB 13	JMP SHORT D4ph1_-.00401279	
00401266	> 6A 10	PUSH 10	
00401268	. 68 00304000	PUSH D4ph1_-.00403000	
0040126D	. 68 AF304000	PUSH D4ph1_-.004030AF	<pre> Style = MB_OK;MB_ICONHAND;MB_A Title = "Crackme#1 By D4ph1" Text = "The Name you entered i hOwner = NULL MessageBoxA </pre>
00401272	. 6A 00	PUSH 0	
00401274	.E8 27000000	CALL <JMP.&user32.MessageBoxA>	

كما ترى هناك امر OR ثم JNZ وهناك دالتي MessageBoxA ...اقراء الاثنتين لتستنج ان الاولى تظهر في حال لم تدخل أي حرف كيوزرنيم والآخرى

يمكننا الاستنتاج انها تظهر اذا كان اليوزرنيم اقل من 4...
ان الامر OR يقوم بعملية منطقية حسب ال truth table التالي ...

1 st operand	2 nd operand	result
0	0	0
0	1	1
1	0	1
1	1	1

كما ترى فالنتيجة لا تكون صفرا الا اذا كان كلا المتغيرين صفرا... اذا طبقنا الجدول على الامر or في حالتنا وكما تلاحظ ان المتغيرين متساويين (al,al) ففي حالة كان al يحتوي

على قيمة ما سواء 1 او 2 او 3 فان عملية ال or لن تنتج صفرا...وبالتالي فان ال zero flag

(انظر الى قسم ال registers) ستكون قيمته صفر ...اذا فالامر Jump if Not Zero سينفذ ويذهب البرنامج الى المسج الثانية التي تخبرك بضرورة ان يكون اليوزرنيم 4 احرف على الاقل اما اذا لم تدخل أي حرف فان al ستكون قيمته 0 وبالتالي فانا امر or نتيجه صفر مما يجعل قيمة zero flag واحد (أي ان هذه الرابية سترفع) وبالتالي فان امر JNZ لن ينفذ وسيذهب البرنامج الى المسج التي تخبرك بضرورة ان تدخل اسما ما...تاكد من ذلك...اغلق ثم اعد التشغيل  واترك مربع الاسم فارغا

0040124D	> 0AC0	OR AL,AL	<pre> Style = MB_OK;MB_ICONHAND;MB_A Title = "Crackme#1 By D4ph1" Text = "You must enter a Name" hOwner = NULL MessageBoxA </pre>
0040124F	.v75 15	JNZ SHORT D4ph1_-.00401266	
00401251	. 6A 10	PUSH 10	
00401253	. 68 00304000	PUSH D4ph1_-.00403000	
00401258	. 68 D4304000	PUSH D4ph1_-.00403004	
0040125D	. 6A 00	PUSH 0	
0040125F	.E8 3C000000	CALL <JMP.&user32.MessageBoxA>	

هل تلاحظ قيمة zero flag ؟ جيد...والان دعنا نعد الى العمل...اعد التشغيل  وادخل allko و 987654321 ولنعد من حيث كنا...

انتهينا من معرفة امر jbe وملحقاته...الامر التالي هو مقارنة بين al (تذكر ان al يحوي عدد الأحرف المدخلة) والقيمة 6...ثم من بعده هناك قفزة JNZ (لاحظ انه يمكن اعتبار jnz ك jne أي اقفز اذا لم يتساوا) قصيرة بمقدار تعليمة واحدة...اي اذا كان الاسم 6 احرف فان التعليمة mov عند 401130 ستنفذ والا فلن تنفذ...اعتقد انها عديمة النفع(يمكنك استنتاج ذلك بعد رؤية ال loop بالاسفل...شخصيا وقعت بالفخ وظننت ان اليوزرنيم يجب ان يكون 6 احرف لكن تبين لي بعدها خطأ اعتقادي) . الامر التالي هو XOR EBX,EBX ولمن لا يعلم هذا الامر فهو باختصار اذا كانت القيمتان متشابهتان فالنتيجة 0 اذا كلما رايت هذا الامر وبجانبه Operands متشابهين فاعلم انه يقوم بتصفيرهما وللتأكد انظر الى قسم المسجلات...وبعده هناك امر اخر يصفّر ESI ومن ثم هناك loop (تذكر ما قلته بالدرس الماضي ، يوجد خط راسي بجانب اوامر ال loop كما بالصورة) واعتقد اننا وصلنا ...

100-cryptography algorithm

لم استطع استنتاج القاعدة العامة لهذه ال loop الا بعد تنفيذها عدة مرات...لذلك لا تقلق فلا يفترض بك ان تستنج هذه الاستنتاجات من اول مرة. في البداية يتل نقل قيمة من [ESI+40334C] وقد شاهدت للتو كيف قمنا بتصفير ESI أي ان قيمته الان صفر...
اضغط F8 وانظر الى اسفل الى قسم التعليقات...هذا العنوان [ESI+40334C] يحتوي على القيمة 61 أي a ...اذن فهنا يخزن الاسم الذي ادخلناه وبالتالي يمكن وضع المخطط التالي :

The letter is stored at	algorithm	The result is stored at
40334C	a=61 XOR 5 = 64	40338C
40334D	l=6C XOR 4 = 68	40338D
40334E	l=6C XOR 3 = 6F	40338E
40334F	k=6B XOR 2 = 69	40338F
403350	o=6F XOR 1 = 6E	403390

0040113D	> 8A86 4C334000	MOV AL, BYTE PTR DS:[ESI+40334C]
00401143	. 3086 80324000	XOR BYTE PTR DS:[ESI+403280], AL
00401149	. 33DB	XOR EBX, EBX
0040114B	. 03D8	ADD EBX, EAX
0040114D	. 33DA	XOR EBX, EDX
0040114F	. 899E 8C334000	MOV DWORD PTR DS:[ESI+40338C], EBX
00401155	. 46	INC ESI
00401156	. 4A	DEC EDX
00401157	. ^75 E4	JNZ SHORT D4ph1_-, 0040113D

الامر التالي يقوم بعمل XOR لقيمة a مع العنوان الموضح... لقد اعتقدت انه شيء مهم لكن تبين لي في النهاية انه مجرد trap لايقع الـ crackers المبتدئين أمثالي!! الامر التالي xor ebx,ebx للتصغير... ثم ينقل اول حرف من eax الى ebx وذلك بامر ADD الامر التالي يبدا انه مهم فهو يقوم بعمل XOR EBX,EDX ثم ينقل النتيجة من EBX الى [ESI+40338C] لاحظ ان EDX يحتوي على عدد الاحرف أي 5 (انظر 401137) ثم هناك امر INC ESI و DEC EDX بالنسبة لامر الزيادة فذلك لتغيير العنوان... الا تلاحظ ان اول امر في الـ loop يحتوي على [ESI+40334C] وبالتالي بداية كانت ESI صفر اذن فان هذه الصيغة تشير الى عنوان اول حرف أي 40334C لكن كلما زاد ESI سيتغير العنوان وفي نهاية الـ loop ستشير تلك الصيغة الى 40334C+4 أي 403350 أي الى الحرف الاخير 0. اما امر النقصان فيليه مباشرة امر JNZ وهذان الامران مترابطان غالبا في كل loop فقيمة EDX كانت في البداية 5 ثم ستتناقص الى ان تصل الى 0 عندها فان القفزة لن تنفذ وسنخرج من الـ loop (أي انه بمثابة عداد)

ملاحظة : انظر الى عنوان اول بارامتر في دالة SendDlgItemTextA على عنوان 401109... ما قيمته؟ صحيح... فالمكان الذي سيخزن فيه الاسم كان محددا في اول بارامتر في الدالة وهذه ملاحظة مهمة.

فلنعد الى امر XOR EBX,EDX... هنا تكمن خوارزمية التشفير!! نعم يتم عمل عملية XOR بين قيمة اول حرف وعدد الاحرف ثم بين قيمة ثاني حرف وعدد الاحرف ناقص واحد ثم... كما هو واضح في الجدول... حسنا افتح ملف ASCII الخاص بك او اذا لم تكن تملك واحدا فقم بفتح الجدول المصغر الذي ارفقته مع الشرح... كما تلاحظ فان عملية الحساب وكما هو مبين في الجدول انتجت لنا كلمة السر التالية dhoin...جرب...رائع!! لقد نجحنا!!

لكن بقيت كلمة السر...دعك منها الان ولنكمل البرنامج لنرى اين هي اجرائية المقارنة... بعد تلك الـ loop هناك الاوامر التالية مباشرة :

00401159	. C686 8C334000	MOV BYTE PTR DS:[ESI+40338C], 2D	[lParam = 4033CC wParam = 41 Message = WM_GETTEXT ControlID = 3EB (1003.) hWnd SendDlgItemMessageA
00401160	. 68 CC334000	PUSH D4ph1_-, 004033CC	
00401165	. 6A 41	PUSH 41	
00401167	. 6A 0D	PUSH 0D	
00401169	. 68 EB030000	PUSH 3EB	
0040116E	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	
00401171	. E8 36010000	CALL <JMP.&user32.SendDlgItemMessageA>	
00401176	. 33D2	XOR EDX, EDX	
00401178	> 0FBEBB CC3340	MOVSBX EDI, BYTE PTR DS:[EDX+4033CC]	
0040117F	. 0FBEB9 8C3340	MOVSBX EBX, BYTE PTR DS:[EDX+40338C]	
00401186	. 42	INC EDX	
00401187	. 3BFB	CMP EDI, EBX	
00401189	. ^74 ED	JE SHORT D4ph1_-, 00401178	

اضغط F8 والان نحن عند اول امر بالصورة وهو امر MOV... القيمة 0x2D (تعني 2D hex) سوف تنقل الى عنوان في الذاكرة لكن هذا العنوان لم نره من قبل... حسنا قد يكون هذا الامر عديم الاهمية على الاقل بالنسبة لنا... تابع الى الامر التالي وهو اور بارامتر في الدالة SendDlgItemMessageA... كما شاهدنا في الدالة السابقة فان اول بارامتر أي lParam يحتوي على العنوان الذي ستخزن فيه المعطيات المقروئة وفي هذه الحالة السيريال نمبر اذن السيريال يبدأ عند العنوان 4033CC... تجاوز بارامترات الدالة وانتقل (متتبعا بـ F8) الى امر XOR يتم الان تصفير EDX وتتوقع انه سيستخدم في الـ loop التالية... اول امر في الـ loop هو لنقل اول خانة في السيريال الى EDI والامر التالي هو لنقل اول خانة في اليوزرنيم بعد التشفير الى EBX ومن ثم هناك امر زيادة لـ EDX أي للانتقال الى الخانة الثانية في كل من السيريال واليوزرنيم المشفر

(الم تستوعب هذه النقطة ؟ لاحظ ان العنوان في امري الـ mov يحتوي على EDX مجموعا الى قيمة ثابتة ، اذن كلما زادت قيمة EDX فان الصيغة ستشير الى العنوان التالي مباشرة أي الى الخانة التالية من السيريال او اليوزر المشفر)

وبالطبع هناك مقارنة ثم قفزة ولا اعتقد ان هناك داع للشرح فالامر واضح اذا تساوى المتغيران اللذان تتم مقارنتهما فستستمر الـ loop وسيتم الانتقال الى الخانتين التاليتين وهكذا الى ان يتم الانتهاء من جميع الاحرف ثم الخروج

من ال loop اما اذا اختلفت احدى المقارنات فسيتم فوراً الخروج من ال loop...ما هذا ؟ هل تقصد انه في الحالتين فاننا سنذهب الى نفس العنوان؟ أي ان المقارنة كانت مجرد خدعة؟ لا...انتظر لترى!!!
الامر التالي مباشرة يقوم بانقاص قيمة EDX (يمكنك ان تعيد تشغيل البرنامج وتدخل السيرال الصحيح وتتبع الى ان تصل هنا) اذا كان السيرال صحيح ففي حالتنا ستكون قيمة EDX تساوي 6 أي تزيد واحدا على عدد الاحرف (لماذا؟ هكذا هي هذه ال loop ..تتبع وانظر بنفسك)

0040118B	4A	DEC EDX
0040118C	83F2 45	XOR EDX,45
0040118F	83F6 45	XOR ESI,45
00401192	2B06	SUB EDX,ESI
00401194	75 6A	JNZ SHORT D4ph1_-.00401200
00401196	E8 97FEFFFF	CALL D4ph1_-.00401032
0040119B	803D 0C344000	CMP BYTE PTR DS:[40340C],1
004011A2	75 47	JNZ SHORT D4ph1_-.004011EB
004011A4	BB 47000000	MOV EBX,47

ثم هناك عملية XOR EDX,45 وبعدها XOR ESI,45 الم اقل لك!! ان قيمة EDX الان تساوي 5 وكذلك ESI (كانت قيمته 5 من ال loop الماضية...) وبغض النظر عن النتيجة لكل عملية على حدة الا ان النتيجة يجب ان تكونان متساويتان اذا ادخلنا السيرال الصحيح...لماذا؟
ان عدد الاحرف تم تخزينه في ESI من ال loop السابقة أي ان قيمته 5...وفي هذه ال 100 اذا ادخلنا سيرال خطأ (او لنقل ان جميع الحروف صحيحة عدا الحرف الاخير...مثلا) فان قيمة EDX ستساوي 4 لان اخر حرف لم يكن صحيحا وبالتالي فاننا خرجنا من ال loop مبكرا...وكما ترى فان قيمة XOR 5,45 لن تساوي XOR 4,45 وبالتالي فان عملية الطرح التي تلي مباشرة SUB EDX,ESI لن تكون قيمتها صفرا ولذلك فان القفزة التالية ستنفذ وستذهب بك الى الجحيم عفواً اقصد الى رسالة bad boy على عنوان 401200 ☺...

الى هنا نكون قد انتهينا بكل ما يتعلق بالسيرال وقد شرحت لك كل شيء بالتفصيل الممل على امل ان اكون قد وفقت في ايصال الفكرة...
اما بخصوص عمل keygen فهذا يعتمد على معرفتك باحدى لغات البرمجة فكل المطلوب ان تستقبل اسما من المستخدم وتحوّل كل حرف من ASCII الى القيمة الرقمية المقابلة ثم تعمل عملية XOR التي وضحتها بالاعلى...

101- the secret message

ملاحظة : لم استطع ان احل هذا الجزء بنفسني رغم انه ليس اصعب من السابق...فلجأت الى احد الحلول ثم اكملت بنفسني لذا وجب التنبيه.
يبدأ هذا القسم من عند انتهاء القسم السابق أي من العنوان 401196 عند دالة call ...

ملاحظة : لا تقم بالضغط على F8 مباشرة اثناء تتبعك لل calls لان ذلك يعني تنفيذ كل الاوامر التابعة لل call ثم العودة والانتقال الى الامر التالي عوضا عن ذلك اضغط انتز للتتبع الى call او ابحث عن العنوان بنفسك...وفورما تصل الى هناك ضع نقطة توقف عن اول امر ثم اكمل تتبعك باستخدام F8 كالمعتاد .
بعد تتبع ال call سنصل الى اعلى البرنامج...تحديدا الى العنوان 401032

00401032	BB 17000000	MOV EBX,17
00401037	BB 2D000000	MOV EAX,2D
0040103C	83F0 0C	XOR EAX,0C
0040103F	83E8 17	SUB EAX,17
00401042	2883 C6314000	SUB BYTE PTR DS:[EBX+4031C6],AL
00401048	4B	DEC EBX
00401049	75 F7	JNZ SHORT D4ph1_-.00401042
0040104B	2883 C6314000	SUB BYTE PTR DS:[EBX+4031C6],AL
00401051	C3	RETN

حسنا ساوفر عليك عناء التعب...اول اربعة اوامر دعك منهم...بعدها هناك loop بعدد EBX يبدأ من 17 حتى صفر...تتبعها وفي كل مرة كنت رى بالاسفل في قسم ال comments رمزا من رموز ASCII فسجلت ال 17 رمز ووظنت انهم الكلمة السرية لكن كانت محاولة فاشلة!!
اذن لا تتعب نفسك...بعد ان تنتهي من ال loop تتبع حتى تصل امر RETN الذي سيعيدنا من حيث بدءنا ...

0040119B	803D 0C344000	CMP BYTE PTR DS:[40340C],1
004011A2	74 47	JE SHORT D4ph1_-.004011EB
004011A4	BB 47000000	MOV EBX,47
004011A9	0FBEB2 CC3340	MOVX EDI,BYTE PTR DS:[EDX+4033CC]
004011B0	0FBEB2 3B3340	MOVX ESI,BYTE PTR DS:[EDX+40333B]
004011B7	03DE	ADD EBX,ESI
004011B9	3BFB	CMP EDI,EBX
004011BB	74 2E	JE SHORT D4ph1_-.004011EB
004011BD	42	INC EDX
004011BE	83FA 05	CMP EDX,5
004011C1	76 F6	JBE SHORT D4ph1_-.004011A9

كما ترى هناك مقارنة تتأكد ما اذا كانت الحسابات في تلك ال loop التي عدنا منها للتو صحيحة ام لا...حسنا انها غير صحيحة ويمكنك معرفة ذلك من قسم ال comments بالاسفل فالمتغيرات ليسا متساويين وبالتالي فان القفزة

JNZ على عنوان 4011A2 ستذهب بنا الى الهاوية...لأذلك وكى تتمكن من تتبع البرنامج غير هذا الامر الى JE بشكل مؤقت أي لا تقم بحفظ التغييرات...والان الامر التالي MOV EBX,47 ينقل القيمة 47 الى EBX...لا بأس...والان ها قد دخلنا الى داخل ال loop ان اول امر فيها ياخذ اول خانة من السيرال تذكر ان السيرال مخزن في 4033CC (يضعها في EDI . الامر التالي ياخذ قيمة من الذاكرة ويضعها في ESI ثم يجمعها الى EBX ويقارن بينها وبين السيرال الذي ادخلناه... (قم بتغيير امر JNZ الثاني كما هو واضح في الصورة) .
عندما تكون عند امر cmp فانظر الى قسم التعليقات لترى ما الذي سيقارن السيرال الخاص بنا معه (أي قيمة EBX)....وبعمليات تتبع حتى النهاية وبملاحظة قسم التعليقات كل مرة ستصل الى التالي

EBX	47	69	66	66	70	75
In ASCII	G	i	f	f	p	u

أي ان السيرال يقارن مع Giffpu
فاذا كان السيرال هكذا فان الـ yوزنيم يمكن استنتاجه بعكس العلاقات الرياضية السابقة...

S1 XOR 6 = 47

S1 = 47 XOR 6 = 41 → A

S2= 69 XOR 5 = 6C → I

S2= 66 XOR 4 = 62 → b

S2= 66 XOR 3 = 65 → e

S2= 70 XOR 2 = 72 → r

S2= 75 XOR 1 = 74 → t

اذن الكلمة السرية هي **Albert**

Do you know?

الـ Domain Name System (DNS) في عالم الاتصالات الحاسوبية هو عبارة عن طريقة لترجمة عناوين الانترنت بحيث يمكن للكمبيوترات المتصلة عبر الانترنت ان تجد بعضها بعضا. يقوم الـ DNS server بترجمة العنوان الرقمي الخاص بكمبيوتر ما (ك 207.46.228.91 مثلا) الى تتابع من الكلمات والعكس صحيح.

Outroduction :

Good , if you , and me , continue with this series we'll finally reach the "medium" level , and that is totally perfect.

You mustn't satisfy yourself with this , rather you should try to solve by your own a similar crackmes . crackmes.de is a good place to start.

Greetings :

Forums : Coding and protection forum , arabteam's coding forum
Special greetings : DiDi

February 2006

There are many crackers , but only one allko
allko34@hotmail.com