



<http://www.arabteam2000-forum.com>

الكاتب: Xacker

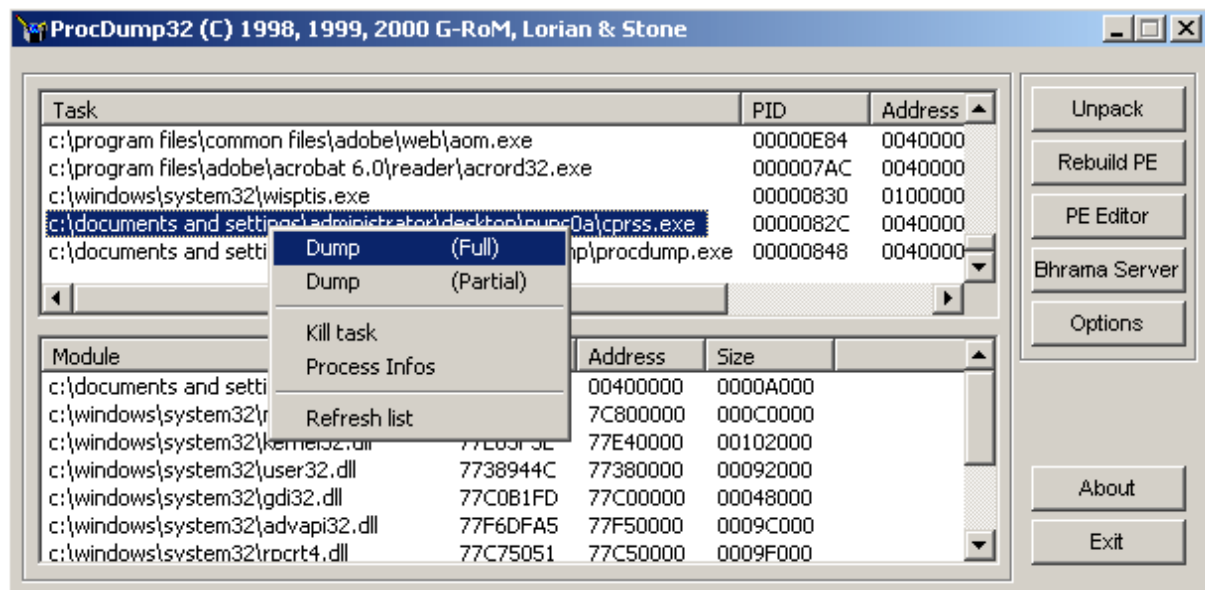
تشفير وفك تشفير الملفات التنفيذية – الجزء الثاني

هذا المقال متابعة لمقال الاخ JAAS الذي كتبه بتاريخ 2004/9/8 حول فك تشفير الملفات التنفيذية المحمية بواسطة برامج حماية. طبعاً المقال لا يحوي اي امر غير قانوني فالتطبيق الذي سنقوم بالتعامل معه هنا هو تطبيق كتبته انا ولا امانع ان قام احد ما بنشره طويلاً او عرضاً!

لمزيد من المعلومات حول بنية الملف التنفيذي ومفهوم حماية البرامج راجع مقال الاخ JAAS, في البداية سنقوم بتوضيح مقال الاخ JAAS باتباع نفس الطرق التي قام بتنفيذها ثم ساعرض كيفية التخلص من Aspack 2.12 بطرق اخرى.

في البداية نقوم باستخدام برنامج ProcDump او LordPE كي نحصل على نسخة شبه مفكوكة الحماية من الذاكرة, هذه الخطوة ليست ضرورية دائماً لكنها مفيدة جداً خصوصاً مع البرامج التي تحوي على resources لان الـ resources عندما تصبح في الذاكرة تكون بصيغتها القياسية وليست المشفرة.

قم بتشغيل البرنامج, افتح ProcDump قم بتحديد البرنامج من القائمة واختر Dump full



حجم الملف الناتج اكبر لانه تمت اعادة جزء كبير من البيانات المحمية الى وضعها الطبيعي بدون حماية.

افتح Olly, قم بتشغيل البرنامج المحمي, قم بالتالي: File > Attach واختر البرنامج المحمي.

الان قم بالنقر على alt+m لتقوم باستعراض محتويات الذاكرة وستظهر لك الاقسام المؤلفة منها البرامج الموجودة في الذاكرة حالياً, ما يهمنا هي الاقسام الخاصة بالبرنامج

بكل بساطة سنقوم باخذ هذه التعليمات الغير محمية وسنقوم باستبدال التعليمات المحمية بها. ما نريده الان هو الحصول على 3000 بايت من هذه القطعة لنقوم باستبدال الجزء المحمي في النسخة الثانية التي قمنا بتحريرها بواسطة اوللي بالجزء غير المحمي, لكن 3000 بايت, لماذا؟ في الواقع هو حجم القطعة text. الطاهرة في الصورة قبل السابقة, لكن لن تحتاج الى كل هذه البايتات فهو الحجم الافتراضي الممنوح من قبل المفسر وليس بالضرورة ان يتم استخدامه بالكامل.

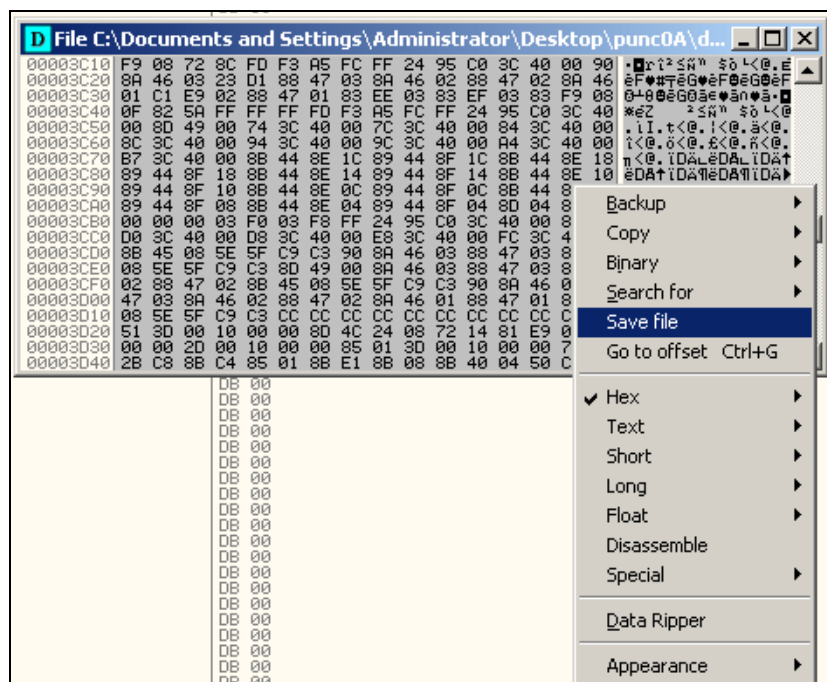
في نافذة CPU قم بالنقر بالزر اليمين واختر selection > Follow in dump لاول تعليمة. سيتم اظهار التمثيل الست عشري للتعليمات في نافذة dump في الاسفل, علينا الان ان نقوم باختيار التعليمات الى نهاية القطعة تقريبا حيث تبدأ الاصفار بالظهور دليل انتهاء الجزء المستخدم من القطعة, البايتات المطلوبة موجودة عند العنوان 1000 وحتى 403DAD كما في الصورة

Address	Hex dump	ASCII
00403D50	8B 54 24 0C 8B 4C 24 04 85 D2 74 47 33 C0 8A 44	iT\$.iL\$+amtG3+eC
00403D60	24 08 57 8B F9 83 FA 04 72 2D F7 D9 83 E1 03 74	\$Wli-ā-r-žāp+
00403D70	08 2B D1 88 07 47 49 75 FA 8B C8 C1 E0 08 03 C1	□+tē+Giū. iLāo□
00403D80	8B C8 C1 E0 10 03 C1 8B CA 83 E2 03 C1 E9 02 74	iLα+β+ iLēf+□□
00403D90	06 F3 AB 85 D2 74 06 88 07 47 4A 75 FA 8B 44 24	α\$+amt+ē+Glu· iDs
00403DA0	08 5F C3 8B 44 24 04 C3 FF 25 38 40 40 00 00 00	□_ iD\$+† %8@e...
00403DB0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403DC0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403DD0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403DE0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403DF0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

الان قم بعمل نقرة يمين فوق هذا الجزء المحدد واختر binary > Binary copy.

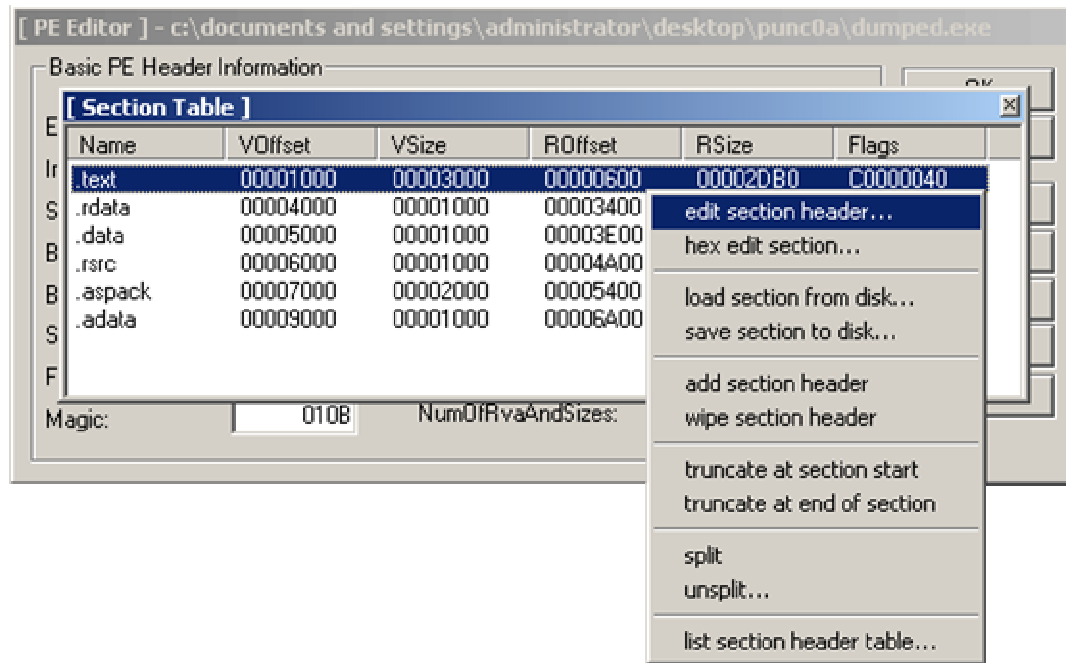
في نفس النسخة من olly انقر File > View واختر الملف الذي حصلنا عليه من الذاكرة وليكن اسمه dumped.exe. انقر ctrl+g وادخل القيمة 1000 وهي بداية القطعة التي قمنا بنسخ البيانات غير المحمية منها.

الان قم بتحديد البايتات من العنوان x1000 حتى x3DAD وقم بعمل نقرة يمين واختر Binary > binary paste.



قم بتحديد نفس الملف او اي ملف اخر تريد, سيتم انشاء نسخة احتياطية منه باي حال.

قم الان بفتح ProcDump او LordPE او اي برنامج تحرير, انا استخدم LordPE, توجه الى نافذة القطع الموجودة وقم بالنقر على text. واختر



سنقوم الان بتصحيح قيمة الحجم الفعلي لبيانات القطعة, كما يظهر في الصورة فان القيمة سلفا صحيحة لدي, لا ادري ما السبب الذي جعلها تختلف لدى JAAS, باي حال اصبحت تعلم ان ال RawSize يجب ان يكون 2DB0.

لكن يبقى علينا تعديل ال Flags الخاصة بالقطعة وذلك من خلال edit section header > flags ندخل القيمة E0000020 (Executable/Readable/Writable) لتنفاذي اي مشكلة من قراءة او كتابة. القيمة الافتراضية كما في الصورة تخبرنا بان القطعة لا يوجد فيها سماحية التنفيذ وهذا امر خطأ لان البيانات فيها الان اصبحت بيانات تنفيذية صحيحة.

ناتي الان الى اصلاح قسم rdata, هذا القسم تتوضع فيه ال RVAs (relative virtual address) التي تشير الى الدوال المراد استيرادها. كيف نعلم هذا؟ انظر الصورة

<pre> MOV ESI,DWORD PTR SS:[ESI+724] PUSH EDI MOV EDI,DWORD PTR DS:[4040DC] PUSH 64 PUSH Cprss.0040551C PUSH 67 PUSH ESI CALL EDI PUSH 64 PUSH Cprss.004054B8 PUSH 6D PUSH ESI CALL EDI PUSH ESI CALL Cprss.004010C0 </pre>	<pre> user32.LoadStringA Count = 64 (100.) Buffer = Cprss.0040551C RsrcID = STRING "DBG32" hInst = 00000004 LoadStringA Count = 64 (100.) Buffer = Cprss.004054B8 RsrcID = STRING "DBG32" hInst = 00000004 LoadStringA </pre>
---	---

هذه الصورة هي لبداية قسم ال text. الصورة تظهر طلب البرنامج لدالة LoadString, لاحظ كيف ان الدالة يتم طلبها باستدعاء القيمة الموجودة في EDI, انظر بضع اسطر للاعلى ستلاحظ ان EDI تاخذ قيمتها من السطر

mov edi, dword ptr [4040DC]

هذا يعني ان القيمة الموجودة في [4040DC] هي عنوان تلك الدالة. قم بالنقر على ذلك السطر بالزر الايمن, اختر Follow in dump > memory address وسترى التالي في نافذة الـ dump

Address	Hex dump	LoadString virtual addr	ASCII
004040DC	EC 82 38 77 3D 6F 3A 77 FA 2E 39 77 3E 9D 3A 77		w8w=o:w.9w>:w
004040EC	5D CA 33 77 F7 E3 38 77 00 00 00 00 FF FF FF FF		9w%T8w....
004040FC	07 16 40 00 1B 16 40 00 72 75 6E 74 69 6D 65 20		.@.+@.runtime

السطر يقوم بتحميل العنوان الموجود هنا الى EDI كي يتم طلب الدالة. لكن كيف وصلت هذه القيم هنا؟

هذا الامر مسؤول عنه الـ loader الخاص بالويندوز فعند تشغيل برنامج يقوم الـ loader بقراءة جدول الوارد ImportTable التي تحوي على كتل متراصة حجم كل كتلة 20 بايت تشير الى معلومات مطلوبة من مكاتب الربط. للتوضيح اكثر انظر الشكل التالي:

D6 20 00 00 E2 20 00 00 F0 20 00 00 02 21 00 00	0 ..â ..ð ...!..
0E 21 00 00 1C 21 00 00 00 00 00 00 68 20 00 00	.!...!.....h ..
00 00 00 00 00 00 00 00 B6 20 00 00 00 20 00 00	9I ...
74 20 00 00 00 00 00 00 00 00 00 00 2C 21 00 00	t/!..
0C 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00 00 00 00 00 00 00 00 A2 20 00 00 94 20 00 00	< .." ..
00 00 00 00 00 00 00 00 C4 20 00 00 D6 20 00 00 E2 20 00 00	ä ..ä ..

هذه الصورة تبين كتلة الـ ImportTable, لاحظ كل اطار يمثل بيانات خاصة بمكتبة من مكاتب الربط المستخدمة في البرنامج. الاطار باللون الاحمر يحمل 20 بايت وكذلك بقية الاطارات. هذه البايتات تقسم الى 5 اقسام كل منها يحمل 4 بايتات وهي عبارة عن dw. الكتلة الاولى باللون الاحمر تمثل كما يلي:

OriginalFirstThunk: (dw) x00002068
 TimeDateStamp: (dw) x00000000
 ForwarderChain: (dw) x00000000
 Name: (dw) x000020B6
 FirstThunk: (dw) x00002000

كيف حصلت على هذه البيانات؟ ببساطة كما قلت في الاعلى الـ ImportTable تقسم الى كتل تمثل كل منها وتكون بطول 20 بايت معلومات ضرورية تتعلق بمكاتب الربط المستخدمة. القيمة الخاصة بالـ Name هنا هي عبارة عن RVA يشير الى المنطقة التي توجد فيها اسم المكتبة المطلوبة.

006B0:	64 6C 65 41 00 00 6B 65 72 6E 65 6C 33 32 2E 64	dleA.	kernel32.d
006C0:	6C 6C 00 00 8A 00 44 69 61 6C 6F 67 42 6F 78 50	ll..š.	DialogBoxP

اما بخصوص الـ OriginalFirstThunk و FirstThunk فالـ loader يقوم بقراءة القيم الخاصة بهم, هاتين القيمتين تمثلان جدولاً تتوضع فيه عناوين تشير الى الوظائف المراد طلبها من هذه المكتبة. الـ OriginalFirstThunk هو ما يعتمد الـ loader ويكون الـ FirstThunk مخصصاً لتصحيح اي مشكلة موجودة في الـ OriginalFirstThunk بمعنى انه نوع من الـ backup يقوم الـ loader بعمل double check على الـ OriginalFirstThunk بواسطته. لننظر الى محتويات العنوان x2068 على سبيل المثال

Address	Hex dump	ASCII
00404000	10 3B 2B 2A 19 90 00 00 3A 37 B9 82 2A 51 42 E8	!;+*+E...:7i!e*QB
00404010	8B A1 41 31 A2 5D 0D 19 65 8D 6C B1 12 13 26 4C	i(A16].+e!l!&L
00404020	9F C1 CC 93 1C CC 92 2A 20 9D 90 EF B9 99 09 7B	f+!f6]fE* n!i0+C
00404030	9D EF 4E F7 DE D9 FC 14 41 1D 0E 8A 22 88 C2 AA	nN% "qA*8e"eT
00404040	31 44 75 54 54 2A 08 25 0B 69 44 74 5D 88 97 52	1DuTT*%?iDtJeuF
00404050	0A 52 8E A4 44 10 57 42 0E 72 51 70 A2 E8 90 74	.R&D!WB#rQp03eT
00404060	55 28 2B B2 3A E2 8E 88 35 D1 DC 96 59 20 A3 5E	U(+&:rA&5T-uY uT
00404070	C8 B6 DC 8A 26 76 E3 A2 F6 44 1A CA A0 96 89 53	!l!e&vT!6+D+&=aue3
00404080	2C 85 84 95 A8 63 09 A3 11 14 90 29 4A 0F DF FE	,a&0&c.u!q!E)J&#
00404090	FF F0 47 9F B9 C3 9E F3 D3 DE 79 A7 D0 DE 79 E7	=Gf!i!<u!q!l!vY
004040A0	EE 79 E7 9C EC 5B 88 55 D4 76 8A A0 E9 C2 7C 33	eyY&e[EUv&e&8T!3
004040B0	A1 98 09 60 E4 03 D2 A2 85 E6 06 B4 35 81 A8 0D	iY.*2#r&8p+!5u!z
004040C0	38 69 43 BC 0A F0 EB C2 B8 2A 83 A5 09 C0 CC 84	8iC!.=3T*8&N.4!f&
004040D0	BD 0C 29 E0 7A 0F 80 81 D6 85 78 69 03 C5 0F 20	".)α&2C!u!r&8i!+*&
004040E0	3D 00 CB DF 0D 13 11 B2 96 B3 67 F2 94 10 B5 DE	=.T!&!!E!g&2&0!&
004040F0	FB C7 0F 10 3C 20 D2 05 80 68 83 AA 0A A0 EA 01	J!f!&*< T*Ch&8.â&0

سنقوم بالانتقال الى نهاية القسم حيث تبدو البيانات المشفرة وسنقوم بوضع نقطة توقف عند الكتابة بواسطة اوللي بحيث نضمن انه تم فك التشفير عن هذا القسم.

الان نقر f9 لتنفيذ البرنامج وسيتوقف التنفيذ هنا عند محاولة الكتابة الى تلك الكلمة (4 bytes)

00407180	F3:A5	REP MOVSD, DWORD PTR ES:[EDI], DWORD PTR DS:[ESI]
00407182	8BC8	MOV ECX, EAX
00407184	83E1 03	AND ECX, 3
00407187	F3:A4	REP MOVSB, BYTE PTR ES:[EDI], BYTE PTR DS:[ESI]
00407189	5E	POP ESI
0040718A	68 00800000	PUSH 8000
0040718F	6A 00	PUSH 0

سنقوم بتنفيذ السطر الثاني REP الظاهر بواسطة F8 وستلاحظ اثناء هذا ان هناك بيانات قد تغيرت بعد المنطقة التي قمنا بوضع نقطة توقف عليها, يبدو اننا اخترنا نهاية خاطئة لكن باي حال حصلنا على ما نريد (:

بداية القسم:

04040000	90 45 00 00	7A 49 00 00	6A 49 00 00	5A 49 00 00	EE..zI..jI..ZI..
04040010	44 49 00 00	34 49 00 00	22 49 00 00	14 49 00 00	DI..4I..*I..*I..
04040020	04 49 00 00	F8 48 00 00	EC 48 00 00	E2 48 00 00	*I..*H..*H..*H..
04040030	D6 48 00 00	CA 48 00 00	BE 48 00 00	B2 48 00 00	*H..*H..*H..*H..
04040040	A4 48 00 00	96 48 00 00	88 48 00 00	7A 48 00 00	*H..*H..*H..*H..
04040050	6A 48 00 00	58 48 00 00	3E 48 00 00	26 48 00 00	jH..XH..>H..&H..
04040060	2E 47 00 00	42 47 00 00	54 47 00 00	66 47 00 00	.G..BG..TG..FG..
04040070	74 47 00 00	82 47 00 00	96 47 00 00	AA 47 00 00	tG..eG..uG..oG..
04040080	C6 47 00 00	DC 47 00 00	F6 47 00 00	10 48 00 00	fG..mG..oG..pG..
04040090	8C 49 00 00	00 00 00 00	14 47 00 00	02 47 00 00	iI.....*IG..*IG..
040400A0	F6 46 00 00	E4 46 00 00	D4 46 00 00	C4 46 00 00	*F..*F..*F..*F..
040400B0	B6 46 00 00	AA 46 00 00	9E 46 00 00	8C 46 00 00	*F..*F..*F..*F..
040400C0	7C 46 00 00	6A 46 00 00	5C 46 00 00	4C 46 00 00	*F..*F..*F..*F..
040400D0	40 46 00 00	32 46 00 00	1E 46 00 00	10 46 00 00	*F..*F..*F..*F..
040400E0	FC 45 00 00	EE 45 00 00	D6 45 00 00	C2 45 00 00	*E..*E..*E..*E..
040400F0	AE 45 00 00	00 00 00 00	FF FF FF FF	07 16 40 00	<<E.....*..

وفيه تظهر عناوين ال RVAS المطلوبة ☺ والتي تشير الى عناوين مثل x4590..

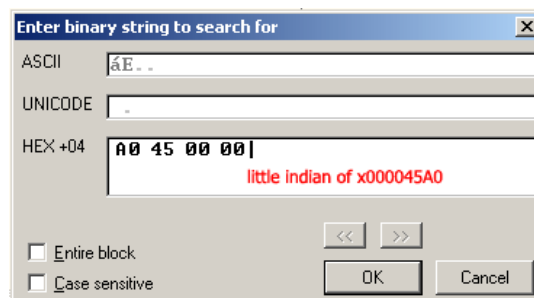
404570	1E 46 00 00	10 46 00 00	FC 45 00 00	EE 45 00 00	*F..*F..*E..*E..
404580	D6 45 00 00	C2 45 00 00	AE 45 00 00	00 00 00 00	*E..*E..*E.....
404590	1A 01 47 65	74 4C 61 73	74 45 72 72	6F 72 00 00	*0GetLastError..
4045A0	4B 45 52 4E	45 4C 33 32	2E 64 6C 6C	00 00 95 00	KERNEL32.dll..o.
4045B0	44 69 73 70	61 74 63 68	4D 65 73 73	61 67 65 41	DispatchMessageA
4045C0	00 00 82 02	54 72 61 6E	73 6C 61 74	65 4D 65 73	..e0TranslateMes
4045D0	73 61 67 65	00 00 7F 02	54 72 61 6E	73 6C 61 74	sage..d0Translat
4045E0	65 41 63 63	65 6C 65 72	61 74 6F 72	41 00 2A 01	eAcceleratorA.*0
4045F0	47 65 74 4D	65 73 73 61	67 65 41 00	96 01 4C 6F	GetMessageA.u0Lo
404600	61 64 41 63	63 65 6C 65	72 61 74 6F	72 73 41 00	adAcceleratorsA.
404610	AB 01 4C 6F	61 64 53 74	72 69 6E 67	41 00 F3 01	%0LoadStringA.%0
404620	52 65 67 69	73 74 65 72	43 6C 61 73	73 45 78 41	RegisterClassExA
404630	00 00 9A 01	4C 6F 61 64	43 75 72 73	6F 72 41 00	..u0LoadCursorA.
404640	9E 01 4C 6F	61 64 49 63	6F 6E 41 00	91 02 55 70	0LoadIconA.*0Up
404650	64 61 74 65	57 69 6E 64	6F 77 00 00	6A 02 53 68	dateWindow..j0Sh
404660	6F 77 57 69	6E 64 6F 77	00 00 59 00	43 72 65 61	owWindow..V.Crea
404670	74 65 57 69	6E 64 6F 77	45 78 41 00	8E 00 44 65	telWindowExA.*De
404680	73 74 72 6F	79 57 69 6E	64 6F 77 00	84 00 44 65	stroyWindow.*De
404690	66 57 69 6E	64 6F 77 50	72 6F 63 41	00 00 BB 00	fWindowProcA..il.
4046A0	45 6E 64 50	61 69 6E 74	00 00 AF 00	44 72 61 77	EndPaint...Draw
4046B0	54 65 78 74	41 00 0C 00	42 65 67 69	6E 50 61 69	TextA...BeginPal
4046C0	6E 74 00 00	14 02 53 65	6E 64 4D 65	73 73 61 67	nt..*0SendMessag
4046D0	65 41 00 00	F0 00 47 65	74 43 6C 69	65 6E 74 52	eA...*GetClientR
4046E0	65 63 74 00	E0 01 50 6F	73 74 51 75	69 74 4D 65	ect.*0PostQuitMe
4046F0	73 73 61 67	65 00 B9 00	45 6E 64 44	69 61 6C 6F	ssage.il.EndDialo
404700	67 00 93 00	44 69 61 6C	6F 67 42 6F	78 50 61 72	a.*DialogBoxEx

وهي فعلا بداية توضع تعليمة GetLastError اذا تاكدنا من حديثي السابق حول ال importable ومكوناته من ان البيانات في الصورتين صحيحة. لنقم الان بتحديد هذه البيانات بنفس الطريقة السابقة ولنقم بوضعها في ملف dumped.exe الذي عدلناه من قبل عند تصحيح ال text. وبنفس الطريقة نقوم بحفظ الملف. لا تنسى ان تقوم بتعديل الحجم ايضا الى 15A0. الان قمنا بتصحيح قسم text. و rdata. لكن لم ينته الامر هنا.

ما زال الملف dumped.exe يشير في معلومات ال ImportTable الى قيم خاطئة. نعود الى الشرح اعلاه، ال IT تحوي على كتل مؤلفة من 20 بايت لكل كتلة تشير الى معلومات حول المكاتب والوظائف المراد جلب عناوينها من هذه المكاتب.

كيف بإمكاننا الوصول من خلال هذه الفكرة الى ال IT الصحيحة بعد ان قمنا بازالة الحماية عنها؟

انظر الصورة اعلاه، هل ترى عبارة kernel32.dll، هذا اسم مكتبة الربط الذي من المفترض ان يكون ال RVA الذي يشير اليه موجودا ضمن ال IT. ما هو عنوانه؟ x45A0 جيد، لنقم بالعودة الى بداية القسم ولننقر ctrl+b لفتح نافذة البحث



ناتج البحث عن هذه القيمة:

0404408	47 65 74 41 63 74 69 76 65 57 69 6E 64 6F 77 00	GetActiveWindow
0404418	40 65 73 73 61 67 65 42 6F 78 41 00 75 73 65 72	MessageBoxA.use
0404428	33 32 2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00	32.dll.....
0404438	FF FF FF FF 2E 37 40 00 32 37 40 00 FF FF FF FF	.70.270.....
0404448	E2 37 40 00 E6 37 40 00 FF FF FF FF 66 39 40 00	70.70..... f90
0404458	6A 39 40 00 98 44 00 00 00 00 00 00 00 00 00 00	j90.70.....
0404468	A0 45 00 00 00 40 00 00 30 45 00 00 00 00 00 00	æ...e...0E.....
0404478	00 00 00 00 22 47 00 00 98 40 00 00 00 00 00 00	...G...70.....
0404488	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0404498	90 45 00 00 7A 49 00 00 6A 49 00 00 5A 49 00 00	æ...zI...jI...ZI...
04044A8	44 49 00 00 34 49 00 00 22 49 00 00 14 49 00 00	DI...4I...I...I...
04044B8	04 49 00 00 F8 48 00 00 EC 48 00 00 E2 48 00 00	♦I...°H...H...H...
04044C8	D6 48 00 00 CA 48 00 00 BE 48 00 00 B2 48 00 00	πH...°H...°H...H...

وهي بداية ال IT ☺

ما نحتاجه من معلومات الان هو عنوان بدايتها وهو: x445C (offset) وحجمها وهو:

3 كتل * 20 بايت = 60 = 3Ch

لماذا 3 كتل؟ انظر الصورة مرة اخرى, 20 بايت خاص ب kernel32.dll, 20 بايت اخرى خاصة ب user32.dll واخيرا 20 بايت كلها اصفار تحدد نهاية ال ImportTable.

قم بفتح الملف dumped.exe الان بواسطة LordPE او ProcDump وقم بتعديل بيانات ال ImportTable من خلال Directories في LordPE, انتبه الى ان القيم يجب ان تكون عبارة عن RVA وليس offset بالتالي 445C تصبح 565C

ExportTable:	00000000	00000000	...	L	H
ImportTable:	0000565C	0000003C	...	L	H
Resource:	00006000	000009D8	...	L	H

اصح القيم وقم بالحفظ.

من المفترض الان انه عند النقر على L فانك ستري بيانات ال IT بصورة واضحة ومفهومة لكن ما نراه هو التالي

```

->Import Table
1. ImageImportDescriptor:
  OriginalFirstThunk: 0x00004498
  TimeDateStamp: 0x00000000 (GMT: Thu Jan 01 00:00:00 1970)
  ForwarderChain: 0x00000000
  Name: 0x000045A0 ("ال 32De<Md%")
  FirstThunk: 0x00004000

  Ordinal/Hint API name
  -----
  0xDB33
  Bad RVA in thunk !
  Bad RVA in thunk !
  Bad RVA in thunk !
  Bad RVA in thunk !
  Bad RVA in thunk !
  0x0040
  
```

ما هذا!!! ما الذي حصل؟

يا سيدي الكريم ما حصل هو ان العناوين الظاهرة هي عناوين offset وليست RVA وهذا خاطئ. نحتاج الى تعديل هذه القيم لتصبح بصورة RVA ويمكن هذا من بالنقر على FLC في LordPE للحصول على القيم المقابلة

Addresses

VA: 004057A0

RVA: 000057A0

Offset: 000045A0

Additional Information

Section: .data

Bytes: 4B 45 52 4E 45 4C 33 32 2E 64

Hex Edit

إذا نحتاج الى تصحيح العناوين الظاهرة في الـ ImportTable الى القيم التالية:

```

FF FF FF FF 66 39 40 00 6A 39 40 00 98 56 00 00  yyyyyf9@.j9@."V.
00 00 00 00 00 00 00 00 00 00 AD 57 00 00 00 52 00 00  ..... W...R.
30 57 00 00 00 00 00 00 00 00 00 00 00 22 59 00 00  0W....."Y.
98 52 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  "R_.....
00 00 00 00 00 00 00 00 00 00 90 45 00 00 7A 49 00 00  ..... R...zT.

```

ثم لدينا تصحيح بوابات الـ RVAs لتشير الى الوظائف بصورة صحيحة في OFT:

```

00 00 00 00 00 00 00 00 90 57 00 00 7A 55 00 00
6A 55 00 00 5A 55 00 00 44 55 00 00 34 55 00 00
22 55 00 00 14 55 00 00 04 55 00 00 F8 54 00 00
EC 54 00 00 E2 54 00 00 D6 54 00 00 CA 54 00 00
BE 54 00 00 B2 54 00 00 A4 54 00 00 96 54 00 00
88 54 00 00 7A 54 00 00 6A 54 00 00 58 54 00 00
3E 54 00 00 26 54 00 00 2E 53 00 00 42 53 00 00
54 53 00 00 66 53 00 00 74 53 00 00 82 53 00 00
96 53 00 00 AA 53 00 00 C6 53 00 00 DC 53 00 00
F6 53 00 00 10 54 00 00 8C 55 00 00 00 00 00 00
14 53 00 00 02 53 00 00 F6 52 00 00 E4 52 00 00
D4 52 00 00 C4 52 00 00 B6 52 00 00 AA 52 00 00
9E 52 00 00 8C 52 00 00 7C 52 00 00 6A 52 00 00
5C 52 00 00 4C 52 00 00 40 52 00 00 32 52 00 00
1E 52 00 00 10 52 00 00 FC 51 00 00 EE 51 00 00
D6 51 00 00 C2 51 00 00 AE 51 00 00 00 00 00 00

```

وفي FT (نسخة ويندوز 2000 تتطلب ان يكون المقطع الثاني صحيح على الاقل)

```

81 FE 00 00 00 00 00 00 40 FE 00 00 00 00 00 00
90 57 00 00 7A 55 00 00 6A 55 00 00 5A 55 00 00
44 55 00 00 34 55 00 00 22 55 00 00 14 55 00 00
04 55 00 00 F8 54 00 00 EC 54 00 00 E2 54 00 00
D6 54 00 00 CA 54 00 00 BE 54 00 00 B2 54 00 00
A4 54 00 00 96 54 00 00 88 54 00 00 7A 54 00 00
6A 54 00 00 58 54 00 00 3E 54 00 00 26 54 00 00
2E 53 00 00 42 53 00 00 54 53 00 00 66 53 00 00
74 53 00 00 82 53 00 00 96 53 00 00 AA 53 00 00
C6 53 00 00 DC 53 00 00 F6 53 00 00 10 54 00 00
8C 55 00 00 00 00 00 00 14 53 00 00 02 53 00 00
F6 52 00 00 E4 52 00 00 D4 52 00 00 C4 52 00 00
B6 52 00 00 AA 52 00 00 9E 52 00 00 8C 52 00 00
7C 52 00 00 6A 52 00 00 5C 52 00 00 4C 52 00 00
40 52 00 00 32 52 00 00 1E 52 00 00 10 52 00 00
FC 51 00 00 EE 51 00 00 D6 51 00 00 C2 51 00 00
AE 51 00 00 00 00 00 00 FF FF FF FF 07 16 40 00

```

والنتيجة ستكون:

```
->Import Table
1. ImageImportDescriptor:
  OriginalFirstThunk: 0x00005698
  TimeDateStamp:      0x00000000 (GMT: Thu Jan 01 00:00:00 1970)
  ForwarderChain:      0x00000000
  Name:                0x000057A0 ("KERNEL32.dll")
  FirstThunk:          0x00005200

Ordinal/Hint  API name
-----
0x011A       "GetLastError"
0x0153       "GetStringTypeA"
0x01C0       "LCMapStringW"
0x01BF       "LCMapStringA"
0x01E4       "MultiByteToWideChar"
0x01C2       "LoadLibraryA"
0x013E       "GetProcAddress"
```

😊

يمكن الان متابعة الخطوات الباقية المتعلقة بـ data و rsrsc. من خلال مقالة الـ JAAS.

اه، هناك ملاحظة بخصوص التغييرات التي حصلت بين المقالين، لكل اسلوبه، ولربما ظهرت معي مشاكل لم تظهر معه. اهم ما في الامر ان تفهم ما تقرا وان تطبق على برامجك الخاصة لانك ستكون ملما بها اكثر من غيرك.

لم لا نتعرف على طريقة اخرى للحصول على الملف بدون حماية بيضع نقرات؟

قم بتحميل EncTest.exe المرفق بواسطة olly، انت الان عند بداية الملف وعند تعليمة PUSHAD

00405001	60	PUSHAD
00405002	E8 03000000	CALL EncTest.0040500A
00405007	- E9 EB045D45	JMP 459D54F7
0040500C	55	PUSH EBP
0040500D	C3	RETN
0040500E	E8 01000000	CALL EncTest.00405014
00405013	✓ EB 5D	JMP SHORT EncTest.00405072
00405015	BB EDFFFFFF	MOV EBX, -13
0040501A	03D0	ADD EBX, EBP
0040501C	81EB 00500000	SUB EBX, 5000
00405022	83BD 22040000 00	CMP DWORD PTR SS:[EBP+422], 0
00405029	89D0 22040000	MOV DWORD PTR SS:[EBP+422], EBX
0040502F	✓ 0F85 65030000	JNZ EncTest.0040539A
00405035	8D85 2E040000	LEA EAX, DWORD PTR SS:[EBP+42E]
0040503B	50	PUSH EAX

هذه التعليمة تستخدم في حفظ جميع قيم المسجلات.. لم يقوم Aspack بذلك؟ عند تنفيذ هذه التعليمة فان المعالج يقوم بحفظ قيم المسجلات جميعها في فضاء المكس، لاحظ نافذة الـ dump قبل وبعد عند العنوان الذي يشير اليه ESP

address	Hex dump	ASCII
0013FFC4	CD 23 E5 77 00 00 00 00 00 00 00 00 00 E0 FD 7F	=#σw.....α²
0013FFD4	18 78 5A 82 C8 FF 13 00 00 00 00 00 FF FF FF FF	↑xZε! !!.....
0013FFE4	98 B7 E6 77 D8 23 E5 77 00 00 00 00 00 00 00 00	γπρwτ#σw.....
0013FFF4	00 00 00 00 01 50 40 00 00 00 00 00	0P@.....

address	Hex dump	ASCII
0013FFA4	00 00 00 00 00 00 00 00 F0 FF 13 00 C4 FF 13 00	= !!.- !!
0013FFB4	00 E0 FD 7F 54 ED 82 7C B0 FF 13 00 00 00 00 00	.α²ΔTφε! !!.....
0013FFC4	CD 23 E5 77 00 00 00 00 00 00 00 00 00 E0 FD 7F	=#σw.....α²
0013FFD4	18 78 5A 82 C8 FF 13 00 00 00 00 00 FF FF FF FF	↑xZε! !!.....
0013FFE4	98 B7 E6 77 D8 23 E5 77 00 00 00 00 00 00 00 00	γπρwτ#σw.....
0013FFF4	00 00 00 00 01 50 40 00 00 00 00 00	0P@.....

هل ترى القيم المحصورة بين العنوانين, هذه هي قيم المسجلات محفوظة بالترتيب.
لاستعادة هذه القيم يتم استخدام التعليمة POPAD.

قم أولا بتنفيذ PUSHAD بواسطة F8, الان اختر ESP وقم باختيار View in dump, قم بتحديد
الاربع بايتات الظاهرة في الصورة الثانية في البداية وانقر نقرة يمين واختر > Breakpoint
Hardware on access > dword - هكذا وضعنا نقطة توقف يتم تفعيلها عندما يحاول البرنامج
استعادة القيم وغالبا في برامج الحماية ما تكون التعليمات بين PUSHAD و POPAD خاصة بفك
الحماية.

الان قم بتنفيذ البرنامج بواسطة F9 وستقف عند الصورة التالية

004053B0	75 08	JNZ SHORT EncTest.004053BA
004053B2	B8 01000000	MOV EAX,1
004053B7	C2 0C00	RETN 0C
004053BA	68 00104000	PUSH EncTest.00401000
004053BF	C3	RETN RETN used like a JMP
004053C0	8B85 26040000	MOV EAX,DWORD PTR SS:[EBP+426]
004053C6	8D8D 3B040000	LEA ECX,DWORD PTR SS:[EBP+43B]
004053CC	51	PUSH ECX
004053CD	50	PUSH EAX
004053CE	FF95 490F0000	CALL DWORD PTR SS:[EBP+F49]
004053D4	8985 55050000	MOV DWORD PTR SS:[EBP+555],EAX
004053DA	8D85 47040000	LEA EAX,DWORD PTR SS:[EBP+447]
004053E0	50	PUSH EAX
004053F1	FF95 51050000	CALL DWORD PTR SS:[EBP+555]

لاحظ ان JNZ دائما تؤخذ وهذه مجرد حيلة من المبرمج. ايضا لاحظ انه يتم دفع العنوان الخاص
ببداية قسم text. الى المكسدس ومن ثم يتم تنفيذ تعليمة RETN مما يجعل هذه التعليمة
تتصرف وكأنه تم تنفيذ السطر: JMP 401000, ايضا كي لا يتم التعرف اليها مباشرة.

00401000	6A	DB 6A
00401001	00	DB 00
00401002	E8	DB E8
00401003	BF	DB BF
00401004	0100	ADD DWORD PTR DS:[EAX],EAX
00401006	00	DB 00
00401007	A3	DB A3
00401008	E4	DB E4
00401009	30	DB 30

تابع بواسطة F8 الى ان تصل الى الصورة السابقة, انقر ctrl+a عندها ليتم تحليل الكود:

00401000	6A 00	PUSH 0	[pModule = NULL GetModuleHandleA
00401002	E8 BF010000	CALL EncTest.004011C6	
00401007	A3 E4304000	MOV DWORD PTR DS:[4030E4],EAX	
0040100C	6A 00	PUSH 0	[lParam = NULL DlgProc = EncTes
0040100E	68 29104000	PUSH EncTest.00401029	hOwner = NULL hTemplate = 65
00401013	6A 00	PUSH 0	hInst = NULL
00401015	6A 65	PUSH 65	DialogBoxParamA
00401017	FF35 E4304000	PUSH DWORD PTR DS:[4030E4]	ExitCode = 0
0040101D	E8 AA010000	CALL EncTest.004011CC	ExitProcess
00401022	6A 00	PUSH 0	
00401024	E8 97010000	CALL EncTest.004011C0	
00401029	55	PUSH EBP	
0040102A	8BEC	MOV EBP,ESP	
0040102C	8B45 0C	MOV EAX,[ARG.2]	
0040102F	3D 10010000	CMP EAX,110	
00401034	75 49	JNZ SHORT EncTest.0040107F	
00401036	68 057F0000	PUSH 7F05	[RsrcName = IDI_0
00401038	6A 00	PUSH 0	hInst = NULL
0040103D	E8 A2010000	CALL EncTest.004011E4	LoadIconA
00401042	68 EB030000	PUSH 3EB	ControlID = 3EB
00401047	FF75 08	PUSH [ARG.1]	hWnd = 00405001
0040104A	E8 89010000	CALL EncTest.004011D8	GetDlgItem
0040104F	6A 00	PUSH 0	lParam = 0
00401051	6A 14	PUSH 14	wParam = 14
00401053	68 C5000000	PUSH 0C5	Message = EM_LIN
00401058	50	PUSH EAX	hWnd = NULL
00401059	E8 92010000	CALL EncTest.004011F0	SendMessageA
0040105E	68 EC030000	PUSH 3EC	ControlID = 3EC
00401063	FF75 08	PUSH [ARG.1]	hWnd = 00405001
00401066	E8 6D010000	CALL EncTest.004011D8	GetDlgItem
0040106B	6A 00	PUSH 0	lParam = 0
0040106D	6A 28	PUSH 28	wParam = 28

البرنامج بدون حماية! يمكنك استخدام اي plugin في Olly لتحصل على نسخة منه بدون حماية وباستخدام ImpRec تستطيع تصحيح العناوين (لكن لا اعتقد انه سيكون هناك اي عنوان خاطئ لانه تطبيق صغير).

طريقة اخرى, كما ذكرت في الاعلى Aspack سيقوم بتصحيح القيم جميعها في نفس العناوين ونعلم ان هذا التطبيق سيعود في النهاية الى بداية الـ text. اذن لنضع نقطة توقف هناك! لكن المشكلة ان اي تعديل سيزيل نقطة التوقف لذا سنقوم بوضع نقطة توقف على بدايته بتحديد اول بايت ونقر نقرة يمين واختيار Breakpoint > memory on write. يلزمنا النقر على F9 ثلاث مرات فقط وسترى بان البداية اصبحت تشير الى تعليمات صحيحة, يمكنك عندها الغاء نقطة التوقف على الكتابة ووضع نقطة توقف بواسطة F2 وتنفيذه F9 وستكون عند بدايته وهو بدون حماية. بنفس الخطوات تحصل على الملف بدون حماية!

طريقة ثالثة, يمكنك من خلال الـ dump الذي حصلت عليه في بداية المقال ان تحدد نوعا ما اللغة التي تمت كتابة البرنامج بها. برنامجنا مكتوب بـ masn هذا ما يجعله ينفذ في البداية وظيفه GetModuleHandleA ليحصل على مقبض البرنامج. لولا ان البرنامج قياسي لما نجحت هذه الطريقة فانتبه. توجه الى ملف kernel32.dll بعد النقر على alt+e, ادخل اليه, انقر ctrl+n واكتب GetModuleHandle, ضع نقطة توقف عليها. الان نفذ البرنامج وستتوقف في kernel32.dll انقر alt+f9, هل عدت الى كتل البرنامج الاصلية؟ لا, كرر النقر على f9 وفي النهاية ستجد نفسك عند العنوان الظاهر في الصورة اعلاه 401007, يتضح عندها لك بداية الملف الصحيحة تستطيع تغيير الـ EP الى بدايته وعمل dump.

هل تحتاج الى طريقة رابعة؟ manual unpacking/decompressing ☺

خامسة؟ Aspack stripper, CASPR, Aspackdie1.41, ...

اتمنى في الختام ان اكون قد وفقت في ايصال الفكرة الصحيحة, نسال الله لكم التوفيق.