

Target : monkey keygen #2
Lang. : C++
Tools : OllyDBG , PEiD , LordPE .
Cracker : allko

Learn Reverse Engineering Series- lesson nine Introduction to code injection III

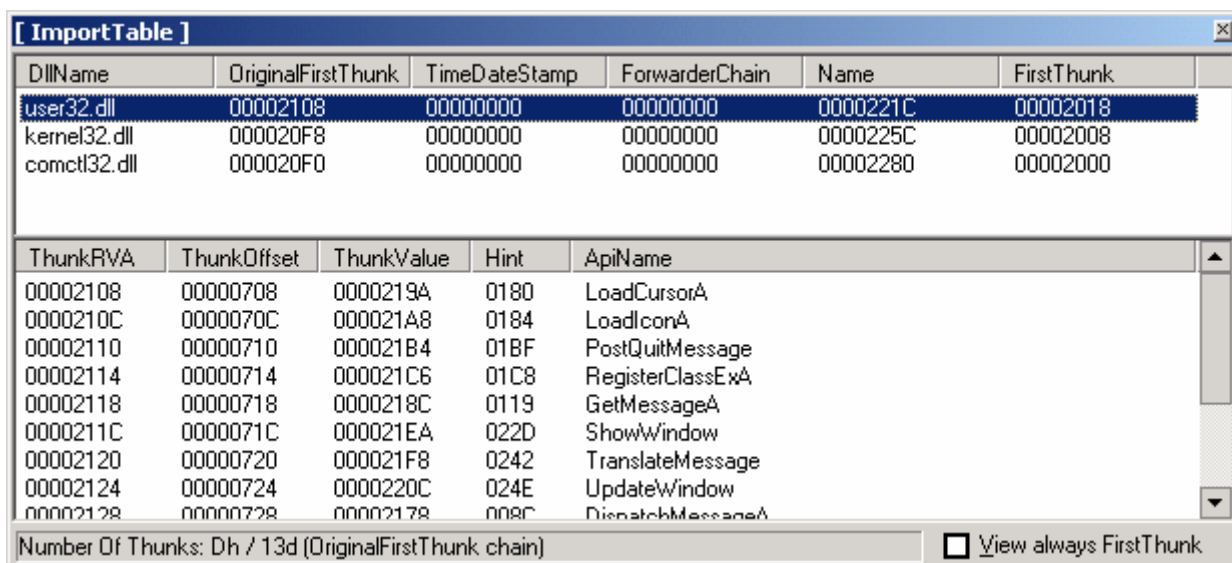
Introduction :

في الدرسين الماضيين تحدثنا عن كيفية عمل code injection للبرنامج بحيث نتخطى الحماية الخاصة به (بعدما كسرناها) واليوم نتعلم شيئا جديدا في هذا المجال حيث سنقوم بإضافة message box للبرنامج. يمكن أن يكون العمل أسهل لو كانت الدالة MessageBoxA مستخدمة في البرنامج ، لكنني فضلت أن أجعل الموضوع أصعب حيث أن هذا البرنامج لا يستورد هذه الدالة مما يستلزم منا أن نقوم بإضافتها يدويا. أود القول أنه من الواجب عليك أن تكون قد قرأت الثلاث دروس السابقة (السادس والسابع والثامن) وإلا فأنك ستواجه صعوبة في فهم هذا الدرس. الاختصارات التي سترد في هذا الدرس هي كالتالي :

IT	Import Table
OFT	Original First Thunk
FT	First Thunk
IAT	Import Address Table
RVA	Relative Virtual Address
IID	Image Import Descriptor
DLL	Dynamic Library Link

Analysis :

سنقوم بتحليل الـ IT الخاص بالبرنامج. ولكن بداية سنقوم بتحليل برنامج آخر - لغاية في نفس يعقوب سأذكرها لاحقا - وهذا البرنامج تجده مرفقا باسم CuTeMessage وللأمانة العلمية فهذا البرنامج من برمجة Cute Devil وحصلت عليه من احد الدروس للأخ Xacker ، فلهما كل الشكر .
قم بفحص البرنامج باستخدام LordPE ثم اختر Directories ثم اضغط زر ... بجانب الـ IT ليظهر التالي.



DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk
user32.dll	00002108	00000000	00000000	0000221C	00002018
kernel32.dll	000020F8	00000000	00000000	0000225C	00002008
comctl32.dll	000020F0	00000000	00000000	00002280	00002000

ThunkRVA	ThunkOffset	ThunkValue	Hint	ApiName
00002108	00000708	0000219A	0180	LoadCursorA
0000210C	0000070C	000021A8	0184	LoadIconA
00002110	00000710	000021B4	018F	PostQuitMessage
00002114	00000714	000021C6	01C8	RegisterClassExA
00002118	00000718	0000218C	0119	GetMessageA
0000211C	0000071C	000021EA	022D	ShowWindow
00002120	00000720	000021F8	0242	TranslateMessage
00002124	00000724	0000220C	024E	UpdateWindow
00002128	00000728	00002178	008C	DispatchMessageA

Number Of Thunks: Dh / 13d (OriginalFirstThunk chain) ☒ View always FirstThunk

انظر إلى أول دالة في user32.dll ولاحظ أن $RVA = 2108$ و $OFT's = 708$ والآن تمعن في المعادلة التالية :

$$\text{Offset} = (\text{Virtual Address} - \text{Image Base} - \text{Virtual Offset}) + \text{Raw Offset}$$

وحيث أن $\text{Virtual Address} - \text{Image Base} = \text{RVA}$ فالمعادلة السابقة تصبح كالتالي : (عليك حفظها وفهمها)

$$\text{offset} = \text{RVA} - \text{Virtual Offset} + \text{Row Offset}$$
$$= 2108 - 2000 + 600$$
$$= 708$$

لكن كيف عرفت قيم الـ Virtual Offset و Row Offset ؟
اذهب إلى sections .

[Section Table]					
Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00001E0	00000400	00000200	60000020
.rdata	00002000	0000028E	00000600	00000400	40000040
.data	00003000	0000000C	00000000	00000000	C0000040
.rsrc	00004000	00002B30	00000A00	00002C00	40000040

لاحظ أن ما نريده (2108) يقع في القسم الثاني .rdata (انظر إلى ثاني عمود من اليسار) . من هنا نحصل على كل القيم اللازمة لإكمال المعادلة السابقة . يمكنك أيضا استخدام المحول الخاص بـ LordPE وذلك بالضغط على زر FLC كما بالصورة

VA:	00402108
RVA:	00002108
Offset:	00000708

عموما ، الآن لا يلزمنا تنفيذ أية حسابات لكنني أردت أن يكون كل شيء واضح.
والآن لنعد إلى الصورة قبل السابقة . هل تمعنت في تلك القيم جيدا؟ إذن كاستنتاج أولي فان الـ IT سوف يحوي (وكما تعلمنا في الدرس الماضي) 5 قيم لكل ملف DLL نستورد منه الدوال لكن حيث أن اثنتان منها غير مستخدمتان (تحديدا ، ForwarderChain & TimeDateStamp فكما ترى فان قيمتهما تساوي الصفر) إذن الـ IT سيحوي 3 قيم لكل ملف DLL أي بمجموع 9 قيم. إذن انتقل إلى الصورة التالية أي عد واضغط زر H بجانب IT .

000006A0:	08 21 00 00 00 00 00 00 00 00 00 00 1c 22 00 00	!....."
000006B0:	18 20 00 00 F8 20 00 00 00 00 00 00 00 00 00 00	...ø...allko...
000006C0:	5c 22 00 00 08 20 00 00 F0 20 00 00 00 00 00 00	\"...ø"
000006D0:	00 00 00 00 80 22 00 00 00 20 00 00 00 00 00 00	...€".....
000006E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

الجزء المظلل هو الـ IT ... لن أقول لك "كما قلنا بالدرس السابق..." بل سأشرح لك عمليا... تمعن الصورتين:
هناك مستطيلات باللون الوردي... هذه هي الـ OFT ومستطيلات باللون الأزرق هذه هي الـ Name والدوائر الصفراء هي الـ FT .

وبالفعل لو نظرنا إلى المستطيل الأول الوردي مثلا فان قيمته 2108 (لاحظ أنها معكوسة، هكذا : 08 21) أي أن الـ offset تساوي 708 ارجع للصورة الأولى لتجد أن قيمة الـ OFT هذه تعود لدالة LoadCurserA .
والآن إذا ذهبت إلى ذلك الـ Offset أي 708 ستجد الـ OFT (هو مجرد قيم... لن تفهم منها شيئا معينا)
وبالمثل لو نظرت إلى المستطيل الوردي الثاني 20F8 لوجدت أنه يمثل الـ OFT لأول دالة في ملف kernel.dll وهي GetCommandLineA ونفس الشيء للمستطيل الثالث.

الخلاصة :

الـ IT يحوي عناوين OFT لأول دالة في كل ملف DLL يتم استيراد دوال منه.
الـ IT يحوي عناوين FT لأول دالة في كل ملف DLL يتم استيراد دوال منه.
الـ IT يحوي عناوين names لكل ملف DLL يتم استيراد دوال منه.
وكما تتذكر فان بنية الـ IMPORT_DESCRIPTOR تكون كالتالي

DWORD	OriginalFirstThunk
DWORD	TimeDateStamp
DWORD	ForwarderChain
DWORD	Name
DWORD	FirstThunk

أي أن هناك (20=5*4) بايت... وهذا يطابق ما نراه... لاحظ أن TimeDateStamp و ForwarderChain تساويان 0 ، يمكنك اعتبار هذه قاعدة بشكل عام (لكن طبعا هناك العديد من التطبيقات لن تكون هاتين القيمتين 0 لكن هذا لا يهمنا) .

هذه النتيجة هامة .

ملاحظة : نفس الشيء ينطبق على ضحيتنا اذهب إلى sections ستلاحظ أن Virtual Offset يساوي الـ Row Offset وبالتالي بالعودة إلى المعادلة $offset = RVA - Virtual Offset + Row Offset$ ستلاحظ أن $offset = RVA$. هل علمت الآن لم تطبق الـ analysis على ملف آخر؟ لو طبقته على ضحيتنا لما فهمت المعادلة ولظننت انه دائما ما يكون $offset = RVA$ على الرغم من أن هذه ليست قاعدة إنما هي مجرد مصادفة .

الآن حمل ملفنا في LordPE . لاحظ أن نفس الفكرة تنطبق فهذا هو الـ IT الخاص بملفنا

000044A0:	10 45 00 00 00 00 00 00 00 00 00 00 E8 46 00 00	.E.....èF..
000044B0:	0C 40 00 00 80 46 00 00 00 00 00 00 00 00 00 00	.@...€F.....
000044C0:	1E 47 00 00 7C 41 00 00 04 45 00 00 00 00 00 00	.G... A...E.....
000044D0:	00 00 00 00 0A 48 00 00 00 40 00 00 C8 46 00 00	H...@...èF..
000044E0:	00 00 00 00 00 00 00 00 80 48 00 00 C4 41 00 00	€H...ÅA..
000044F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	allko.....
00004500:	00 00 00 00 E4 47 00 00 F8 47 00 00 00 00 00 00	äG...øG.....

لاحظ أيضا (بالضغط على زر [...] بجانب الـ IT) أن أول ملف DLL يتم استدعاء دواله باستخدام الـ ordinal وليس باستخدام الاسم. وحيث أن الـ $offset = 0$ لذا فعملية البحث ستكون أسهل...

Adding The functions :

والآن ماذا لدينا...لاحظ الملف من أوله إلى آخره ...

ما الذي نريده؟

نريد أن نضيف دالتي LoadLibraryA و GetProcAddress إلى ملف kernel32.dll لأنه بواسطة هاتين الدالتين يمكننا استدعاء (call) أي دالة أخرى .

ملاحظة : من الممكن إضافة MessageBoxA مباشرة لكن إذا ما أردنا إضافة دالة أخرى سنضطر لإعادة العمل من جديد وهذا سيستهلك الكثير والكثير من الوقت. لكن انظر الصورة التالية:

000047D0:	6C 65 72 33 00 00 B7 00 5F 63 6F 6E 74 72 6F 6C	ler3... control
000047E0:	66 70 00 00 26 01 47 65 74 4D 6F 64 75 6C 65 48	fp...&.GetModuleH
000047F0:	61 6E 64 6C 65 41 00 00 50 01 47 65 74 53 74 61	andleA...P.GetSta
00004800:	72 74 75 70 49 6E 66 6F 41 00 4B 45 52 4E 45 4C	rtupInfoA.KERNEL
00004810:	33 32 2E 64 6C 6C 00 00 9E 01 4C 6F 61 64 49 63	32.dll...z.LoadIc
00004820:	6F 6E 41 00 14 02 53 65 6E 64 4D 65 73 73 61 67	onA...SendMessage

حسنا...يمكننا أن نضيفهما إلى ملف kernel32.dll وذلك بكتابتهم بجانب الدالتين المستوردتين لهذا الملف لكن كما تلاحظ لا يوجد مكان كافٍ (ولا حتى بايت واحد فارغ!!) ...حسنا لم لا نضيف IID جديد لملف KERNEL32.dll ؟ صحيح أن kernel32.dll مضاف سلفا للـ IT لكننا سنضيفه بطريقة عادية وكأنه ملف آخر...لكن لحظة... هل تكفي المساحة لعمل ذلك؟ نعم يوجد...إذن اذهب الآن أولا إلى LordPE ثم اضغط H بجانب الـ IT...هل رأيت الاوفست أنها 0x44A0 ولاحظ انه يمتد حتى 0x4503 (أي أن حجمه 0x63) .
والآن قم بعمل نسخة من الملف وافتحها بـ LordPE (سنستخدمها لننظر فقط لتسهيل مهمة نقل الـ IT)
وافتح النسخة الأخرى وانقل الـ IT إلى عنوان آخر وليكن مثلا 4920 (اعرف أن العملية مملة ومرهقة للعين لكن لا حل آخر . حاولت نسخه ولصقه باستخدام HexWorkshop لكن بعد الحفظ وتشغيله يظهر error)
كما قلنا نريد إضافة IID لـ Kernel32.dll وإضافة دالتي لها وهما LoadLibraryA و GetProcAddress حيث يمكن استخدامهما لاستيراد أي دالة أخرى.. الآن علينا كتابة أسماء الدالتين في مكان ما... إذا اذهب إلى 48A0 (اخترت أن أضيفهم أسفل أسماء بقية الدوال) واكتب LoadLibraryA ثم 4 أزواج من الأصفار ثم GetProcAddress

0004880:	55 53 45 52 33 32 2E 64 6C 6C 00 00 AA 01 5F 73	USER32.dll...a. s
0004890:	65 74 6D 62 63 70 00 00 00 00 00 00 00 00 00 00	etmbcp.....
00048A0:	4C 6F 61 6C 72 61 72 79 41 00 00 00 00 00 00 00	LoadLibraryA....
00048B0:	47 65 74 5F 41 64 64 72 65 73 73 00 00 00 00 00	GetProcAddress..

لاحظ أن اسم كل دالة مسبوق بصفرين (زوج واحد من الأصفار). لماذا؟ هذين الصفرين هما ما يسمى بـ hint ويمكننا وضع قيمتهما صفرا...حسنا لقد قلت صفرين ووضعت 4 لماذا؟ بدون سبب...يمكنك وضع صفرين فقط...
والآن لنحسب الـ RVA لهذه العناوين للدوال. اضغط على أول بايت في الدالة الأولى (أي أول بايت في الـ hint) وستجد أن الـ Offset تساوي 489E . والآن اذهب إلى قسم الـ SECTIONS لتر التالي

(كيف عرفت الحجم؟ كان هناك IID 4 + 1 فارغة = 5. تذكر أن أي IT يجب أن ينتهي بـ IID فارغة. ثم أضفنا IID 1 فأصبحوا 6. إذن 6*20=120 أو بالعكس 6*0x14 = 0x78)
والآن اضغط على زر L لتر التالي

```
5. ImageImportDescriptor:
OriginalFirstThunk: 0x000049A0
TimeDateStamp:      0x00000000 (GMT: Thu Jan 01 00:00:00 1970)
ForwarderChain:      0x00000000
Name:                0x0000480A ("KERNEL32.dll")
FirstThunk:          0x000049A0

Ordinal/Hint  API name
-----
0x0000       "LoadLibraryA"
0x0000       "GetProcAddress"
```

allko

(انزل إلى أسفل القائمة). أن ظهر لك كالذي تراه بالصورة فعملك صحيح و الا فراجع الخطوات مرة أخرى.
والآن أن لم تكن قد حفظت التغييرات على الـ IT فاحفظها. أغلق LordPE وشغل الملف لتأكد أن عملنا صحيح.

Adding the Message Box :

الآن نأتي إلى القسم الثاني من الدرس. نريد إضافة Message. وذلك يتم عن طريق استدعاء الدالة المسماة MessageBoxA بداية اذهب إلى موقع www.msdn.com و ابحث عن الدوال الثلاث التي تهمننا و اقرأ بعض المعلومات عنها .

GetProcAddress

The GetProcAddress function retrieves the address of an exported function or variable from the specified DLL.

FARPROC GetProcAddress (HMODULE hModule , LPCSTR lpProcName);

Parameters

hModule

Handle to the DLL module that contains the function or variable. The LoadLibrary or GetModuleHandle function returns this handle

lpProcName

Pointer to a null-terminated string that specifies the function or variable name, or the function's ordinal value. If this parameter is an ordinal value, it must be in the low-order word; the high-order word must be zero

LoadLibrary

The LoadLibrary function maps the specified executable module into the address space of the calling process

HMODULE LoadLibrary(LPCTSTR lpFileName);

Parameters

lpFileName

Pointer to a null-terminated string that names the executable module (either a .dll or .exe file).

إذن سنستخدم LoadLibrary للحصول على مقبض (handle) للملف USER32.DLL (وهو الملف التي توجد فيه دالة MessageBoxA) ومن ثم نستخدم GetProcAddress للحصول على عنوان لدالة MessageBoxA الموجودة في KERNEL32.DLL . لاحظ أن أغلب الدوال ترجع القيم في EAX أي أن مقبض الملف سيكون في EAX . يجدر بالذكر أن الـ null-terminated string تبدأ من أول حرف في اسم الدالة (اقصد أول حرف في MessageBoxA) أي أننا نتخطى الـ hint .

أما MessageBoxA فاعتقد انه لا حاجة للذهاب إلى الموقع للبحث عنها فكثيرا ما تعاملنا معها أثناء الدروس السابقة ولعلك لاحظت أنها تحوي 4 بارامترات كالتالي :

MessageBox (HWND hWnd , LPCTSTR lpText, LPCTSTR lpCaption, UINT uType);

لاحظ أن البارامترات تدفع من اليمين إلى اليسار .
إذن يلزمنا 2 strings واحدة لعنوان المسح والأخرى للنص الذي سيعرض. اكتب ما يحلو لك في أي مكان تريد.
أنا اخترت كتابتهما في 0406A60 و 0406A80 . أيضا يلزمنا كتابة الـ string التالية : "MessageBoxA" أنا اخترت

أن اكتبها في 0406AC2 إضافة إلى ذلك يلزمنا عنوان الـ strings لـ USER32.DLL . إذن افتح الملف باستخدام LordPE وابحث عن USER32.DLL لتجدها في الـ offset رقم 4880 . استخدم المعادلة السابقة أو الآلة الحاسبة الخاصة بـ LordPE للحصول على RVA لتجد بأنه 404880 .

إذن الكود اللازم سيكون كالتالي : (كل دالة مع بارامتراتنا ستجدها بلون مختلف)

PUSH 0404880h	بداية ندفع عنوان الـ RVA لـ USER32.DLL إلى الـ stack
CALL LoadLibraryA	والآن نستدعي دالة LoadLibraryA حيث ستعيد مقبض USER32.DLL في EAX
PUSH 0406AC2h	ندفع الـ RVA لـ MessageBoxA's string (يختلف حسب المكان الذي كتبته فيها)
PUSH EAX	ندفع مقبض ملف USER32.DLL إلى الـ stack كي تستخدمه GetProcAddress
CALL GetProcAddress	نستدعي GetProcAddress حيث ستعيد عنوان MessageBoxA في EAX
PUSH 0h	أول بارامتر لدالة MessageBoxA (غالبا يكون 0)
PUSH 0406A60h	ثاني بارامتر يخص عنوان المسج
PUSH 0406A80h	ثالث بارامتر يخص النص الذي سيعرض في المسج
PUSH 0h	رابع بارامتر غالبا يكون 0
CALL EAX	نستدعي MessageBoxA

والآن يجب أن المكان الذي سنغير من عنده مجرى البرنامج (code flow) . افتح البرنامج باستخدام OllyDBG وانظر قليلا . في الواقع جميع الأوامر تقريبا (عدا البعض ، كأوامر call مثلا) يمكن تغيير البرنامج من عندها. لاحظ أننا نريد عرض المسج قبل تشغيل البرنامج نفسه وليس بعد تنغيله. أن أمر PUSH ESI عند عنوان 402FC1 يبدو مناسباً . في الواقع جميع الأوامر من البداية 402FA0 وحتى 402FCB تبدو مناسبة في حالة أردنا ظهور المسج قبل البرنامج. قبل تغيير أمر PUSH ESI لاحظ أننا يجب أن نحفظ قيم المسجلات حتى إذا ما تغيرت أثناء عمل الكود الخاص بنا استطعنا من بعده إعادتها إلى ما كانت عليه. هذا يتم من خلال أمر pushad الذي يقوم بحفظ جميع المسجلات دفعة واحدة وبعد الانتهاء من الكود نستخدم popad . هناك أمر آخر يجب الانتباه إليه . أن أمر PUSH ESI الذي سنغيره إلى قفزة jmp أو اتصال call كان يؤدي وظيفة ما ، وبتغييره قد تحدث مشكلة ، إذن يجب أن يحتوي الكود الخاص بنا في نهايته على أمر PUSH ESI ليقوم بنفس العمل وكان شيئا لم يكن . سأختار تغيير الفيض (سريان الأوامر) عن طريق قفزة . إذن غير PUSH ESI إلى JMP 004031FF لا داعي لأن أقول أن 4031FF هو عنوان اختياري فأنت أصبحت تعرف هذا. اذهب إلى هناك واكتب الكود ثم في النهاية أضف قفزة لإعادة البرنامج لوضعه السابق. بعد حفظ التغييرات سيبدو الكود بالشكل التالي :

004031FF	> 60	PUSHAD	FileName = "USER32.dll"
00403200	. 68 80484000	PUSH reversed.00404880	LoadLibraryA
00403205	. E8 6DEB3F7C	CALL kernel32.LoadLibraryA	ProcNameOrOrdinal = "MessageBoxA"
0040320A	. 68 C26A4000	PUSH reversed.00406AC2	hModule
0040320F	. 50	PUSH EAX	GetProcAddress
00403210	. E8 11A437	CALL kernel32.GetProcAddress	
00403215	. 6A 00	PUSH 0	ASCII ".:!Reverseengineering!:"
00403217	. 68 606A4000	PUSH reversed.00406A60	ASCII "reversed by : allko"
0040321C	. 68 806A4000	PUSH reversed.00406A80	
00403221	. 6A 00	PUSH 0	
00403223	. FFD0	CALL EAX	
00403225	. 61	POPAD	
00403226	. 56	PUSH ESI	
00403227	. ^E9 9AFDFFFF	JMP reversed.00402FC6	

والآن أغلق olly لكن أن حاولت تشغيل البرنامج فعابا ستصادفك مشكلة. القسم الذي كتبت فيه الكود تنقصه بعض التعديلات.

الحل أن تفتح الملف بـ LordPE ومن الواجهة الرئيسية اختر sections وقم بالضغط على أول قسم باليمين واختر Edit section header ثم اضغط زر ... بجانب Flags وقم باختيار الخيارات التالية.(بعضها قد يكون مختارا أصلا) Executable as code و Writable و Contains executable code ثم OK . لاحظ قيمة الـ Flags أصبحت E0000020 والآن اما أن تكرر نفس الخطوات مع بقية الأقسام أو أن تنسخ تلك القيمة (E0000020) في خانة Flags لكل قسم.

الآن قم بحفظ التغييرات وشغل البرنامج .



مبارك!! لقد نجحنا !!

Do you know?

حقائق عن المجتمعات الغربية التي لا هم لشباب اليوم إلا أن يقلدوهم
(وأخص بالذكر برنامج الحثالة المسمى ستار أكاديمي):

- أظهرت الإحصائيات أن 19 مليون امرأة أمريكية كانت ضحية لعمليات الاغتصاب.
- أجرى الاتحاد الإيطالي للطب النفسي استطلاعاً للرأي اعترف فيه 70% من الرجال أنهم خانوا زوجاتهم.
- في أمريكا مليون طفل كل عام من الزنا ومليون حالة إجهاض.
- في ألمانيا وحدها تغتصب 35 ألف امرأة سنوياً حسب الإحصاءات الرسمية ، أما تقديرات الشرطة الجنائية فتشير إلى خمس أضعاف هذا الرقم.
- في الدانمرك يوم 1999/3/13 عقد وزير الصحة السابق " تورين لوند " - 49 سنة - قرانه على صديقه الشاب " كلاوس لاثروب " بحضور عدد من أهالي الزوجين والأصدقاء وبحضور وزير العدل الدانمركي! علماً بأنه - أي تورين - تزوج قبل ذلك مرتين وله ابنتان . المصدر: جريدة الحياة الجديدة-العدد 1160 - 1999/3/14 .
- في بريطانيا ، أحد وزراء حكومة العمل (سنة 1999) شاذ جنسياً ، والرجل الذي يعيش معه يحضر جلسات وزارة العموم في الغرفة المجاورة على أساس أن له امتياز الزوجة .

Outroduction :

This was really time-consuming , it take me days to complete this , but it's worth fatigue.

I was planning to add the message by clicking a button , but I discovered that it's tough and won't be that useful .

As usual , any criticisms , suggestions or thankfulls are welcomed : allko34[at]hotmail[dot]com.

Greetings :

DiDi.

Xacker.

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