

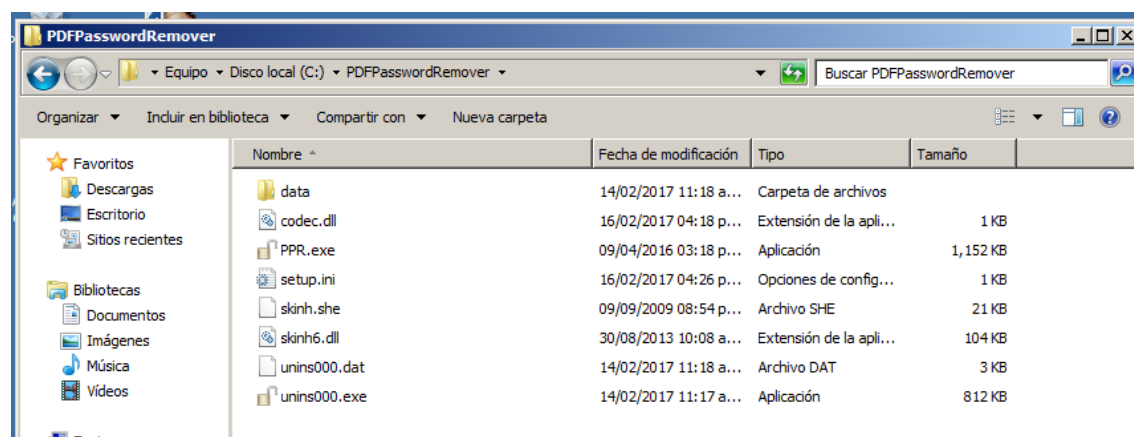
PDF Password Remover	
Version	1.5
URL:	http://www.pdfpasswordremover.com/
Proteccion :	Ninguna
Dificultad :	Newbie
Herramientas :	RDG Packer Detector v0.7.6 , Ollydbg y Mucho animos.
Objetivo :	Registrarlo + Parchar
Reverser :	L1oR
Grupo :	Peru Crackeando
<p>Gracias a Ricardo Narvaja, por sus manuales de Ollydbg, donde fue la base de empezar en el Reverser, y a Davicorn por sus aportes constantes subidos al grupo, a Abel, Nox, K3n4n (Se encuentra secuestrado) y SoftDat.</p>	

INFORMACION DE LA VICTIMA :

PDF Password Remover permite a los usuarios eliminar las restricciones de formato PDF, para que tenga la capacidad de imprimir y modificar, el costo del software es de unos \$29.95 XD.

ATACANDO :

Lo primero que realizo es encontrar la ruta de instalacion del software, para luego crear un Backup, por medidas de seguridad.



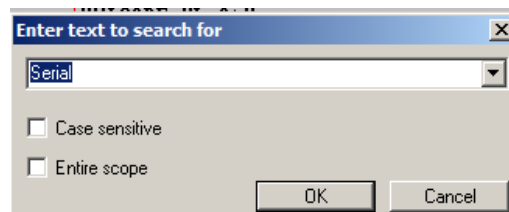
Luego de obtener un backup, procederemos a ver en que lenguaje se desarrollo dicha aplicación, como vemos esta desarrollado Visual Basic 6.0 y no tiene medida de seguridad, estamos en un buen camino.



Uno de los detalles es recolectar informacion, por ejemplo el BAD BOY, del software, en este caso nos muestra la venta "Invalid Serial Code".



Con el dato obtenido del BAD BOY, buscaremos el STRING, buscaremos la palabra "Serial".



En este caso, encontramos el STRING “Invalid Serial Code”, en la direccion de memoria 004FC7E5.

```
Address Disassembly Text string
004F7E5 mov dword ptr ss:[ebp-0x4C],PPR.004ED35C UNICODE "PDF Files(*.PDF)!*.pdf"
004F7E7 mov edx,PPR.004ED33C UNICODE "Open Files"
004F7E9 push PPR.004EC8A0 UNICODE "c:\tmpremove"
004F7EB push PPR.004EC8BC UNICODE "."
004F7ED mov edx,PPR.004EB590 UNICODE "%s.%s"
004F7EF push PPR.004ECEFA UNICODE "L"
004F7F1 push PPR.004ECEFA UNICODE "L"
004F7F3 push PPR.004ECEFA UNICODE "L"
004F7F5 push PPR.004ECEFA UNICODE "L"
004F7F7 push PPR.004ECEFA UNICODE "L"
004F7F9 push PPR.004ECEFA UNICODE "Left"
004F7FB push PPR.004ECDE4 UNICODE "Left"
004F7FD push PPR.004EC8E4 UNICODE "Left"
004F7FF push PPR.004ED934 UNICODE "Top"
004F801 push PPR.004EC8F0 UNICODE "Top"
004F803 push PPR.004ED934 UNICODE "Width"
004F805 push PPR.004ED934 UNICODE "Width"
004F807 push PPR.004ED940 UNICODE "Height"
004F809 push PPR.004ED940 UNICODE "Height"
004F80B push PPR.004EC8F0 UNICODE "FontSize"
004F80D push PPR.004EC8F8 UNICODE "FontSize"
004F80F push PPR.004ED954 UNICODE "https://secure.avangate.com/order/checkout.php?PRODS=4556898&QTY=1&CARD=1&CARD=2"
004F811 push PPR.004EC930 UNICODE "open"
004F813 push PPR.004ED9F8 UNICODE "Picture"
004F815 push PPR.004ED9F8 UNICODE "Picture"
004F817 push PPR.004EC95C UNICODE "\codec.dll"
004F819 push PPR.004ED900 UNICODE "asdfsdfasdfs"
004F81B push PPR.004EC878 UNICODE "kk"
004F81D mov dword ptr ss:[ebp-0xA0],PPR.004ED8A0 UNICODE "Successful"
004F81F mov dword ptr ss:[ebp-0x98],PPR.004ED8A2 UNICODE "Register Successfully! Please restart program."
004F821 push PPR.004ED2B0 UNICODE "http://www.pdfpasswordrecovery.com"
004F823 push PPR.004EC930 UNICODE "open"
004F825 mov dword ptr ss:[ebp-0xA8],PPR.004ED8A0 UNICODE "Invalid"
004F827 mov dword ptr ss:[ebp-0x98],PPR.004ED8A0 UNICODE "Invalid Serial Code."
004F829 push PPR.004ED954 UNICODE "https://secure.avangate.com/order/checkout.php?PRODS=4556898&QTY=1&CARD=1&CARD=2"
004F82B push PPR.004EC930 UNICODE "open"
004F82D push PPR.004ECEFA UNICODE "L"
004F82F push PPR.004ECEFA UNICODE "L"
004F831 push PPR.004ECEFA UNICODE "L"
004F833 push PPR.004ECEFA UNICODE "L"
004F835 push PPR.004ECEFA UNICODE "L"
004F837 push PPR.004EB590 UNICODE "%s.%s"
004F839 mov edx,PPR.004EDFCC UNICODE "Save"
004F83B push PPR.004ECEFA UNICODE "L"
004F83D push PPR.004ECEFA UNICODE "L"
004F83F push PPR.004ECEFA UNICODE "L"
004F841 push PPR.004ECEFA UNICODE "L"
004F843 push PPR.004ECEFA UNICODE "L"
004F845 mov dword ptr ss:[ebp-0xEC],PPR.004EE03C UNICODE "Command Line empty"
004F847 push PPR.004EE064 UNICODE "CreatePipe failed. Error: "
004F849 mov dword ptr ss:[ebp-0xEC],PPR.004EE0B8 UNICODE "File or command not found"
004F84B mov dword ptr ss:[ebp-0xA8],PPR.004EE354 UNICODE "NMS"
```

Con este datos iremos a dicha direccion de memoria 004FC7C3, si observamos en la direccion 004FC79A nos muestra una linea de codigo que fue llamado por medio de un SALTO.

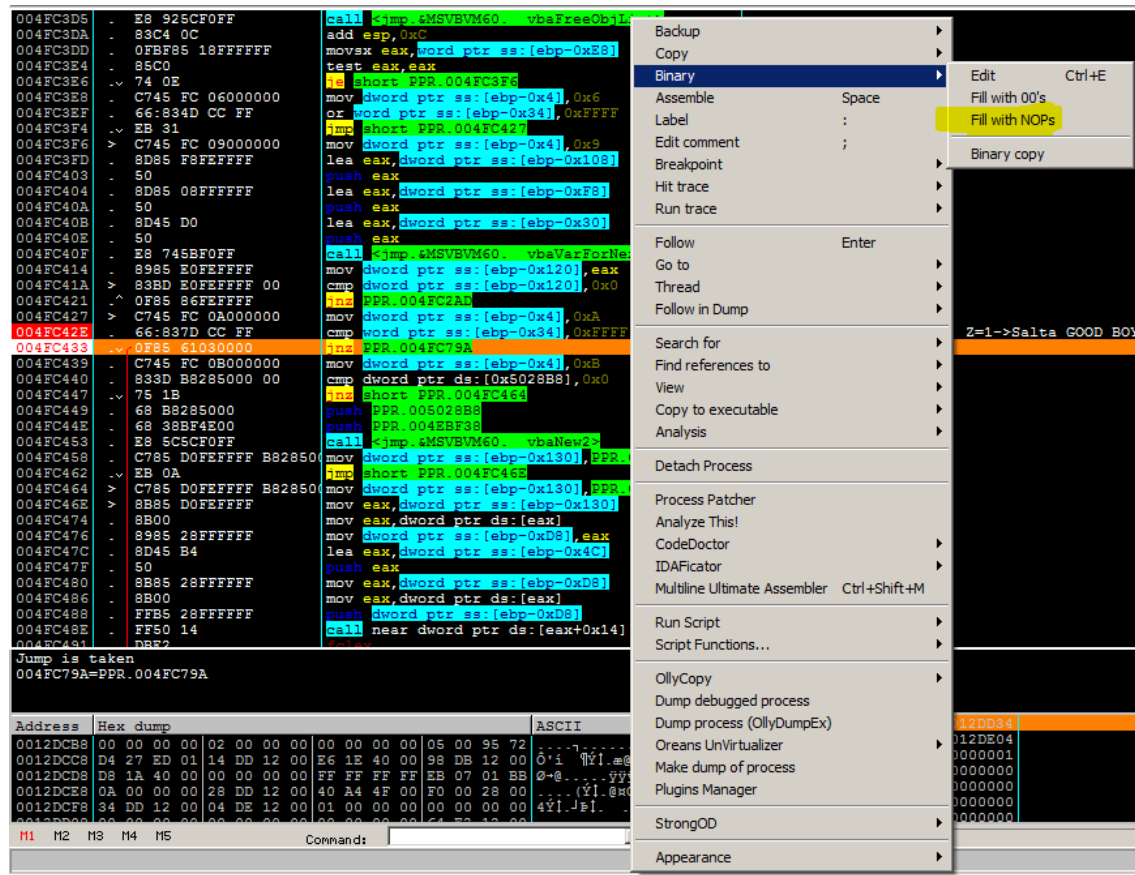
```
Address Disassembly Text string
004FC75C - 50 push eax
004FC75D - E8 BC58F0FF call <jmp.<MSVBVM60... vbaStrToAnsi>
004FC762 - 50 push eax
004FC763 - 6A 00 push 0x0
004FC765 - E8 1AF7FEFF call PPR.004EBE84
004FC76A - E8 C359F0FF call <jmp.<MSVBVM60... vbaSetSystemError>
004FC76F - 8B 45 B8 lea eax,dword ptr ss:[ebp-0x48]
004FC772 - 50 push eax
004FC773 - 8B 45 BC lea eax,dword ptr ss:[ebp-0x44]
004FC776 - 50 push eax
004FC777 - 8B 45 C0 lea eax,dword ptr ss:[ebp-0x40]
004FC77A - 50 push eax
004FC77B - 8B 45 C4 lea eax,dword ptr ss:[ebp-0x3C]
004FC77E - 50 push eax
004FC77F - 6A 04 push 0x4
004FC781 - E8 0459F0FF call <jmp.<MSVBVM60... vbaFreeStrList>
004FC786 - 83C4 14 add esp,0x14
004FC789 - C745 FC 11000000 mov dword ptr ss:[ebp-0x4],0x11
004FC790 - E8 1959F0FF call <jmp.<MSVBVM60... vbaEnd>
004FC795 - E9 45010000 jmp PPR.004FC8D1
004FC79A - C745 FC 13000000 mov dword ptr ss:[ebp-0x4],0x13
004FC7A1 - C785 78FFFFFF 04000020 mov dword ptr ss:[ebp-0x88],0x80020004
004FC7AB - C785 70FFFFFF 0A000000 mov dword ptr ss:[ebp-0x90],0xA
004FC7B5 - C745 88 040000280 mov dword ptr ss:[ebp-0x78],0x80020004
004FC7BC - C745 80 0A0000000 mov dword ptr ss:[ebp-0x80],0xA
004FC7C3 - C785 S8FFFFFF D8DA4E mov dword ptr ss:[ebp-0xA8],PPR.004EDAD8 UNICODE "Invalid"
004FC7CD - C785 S0FFFFFF 08000000 mov dword ptr ss:[ebp-0x80],0x8
004FC7D7 - 8D95 S0FFFFFF lea edx,dword ptr ss:[ebp-0x80]
004FC7DD - 8D4D 90 lea ecx,dword ptr ss:[ebp-0x70]
004FC7E0 - E8 2758F0FF call <jmp.<MSVBVM60... vbaVarDup>
004FC7E5 - C785 68FFFFFF A8DA4E mov dword ptr ss:[ebp-0x98],PPR.004EDAA8 UNICODE "Invalid Serial Code."
004FC7EF - C785 60FFFFFF 08000000 mov dword ptr ss:[ebp-0xA0],0x8
004FC7F9 - 8D95 60FFFFFF lea edx,dword ptr ss:[ebp-0xA0]
004FC7FF - 8D4D A0 lea ecx,dword ptr ss:[ebp-0x60]
004FC802 - E8 0558F0FF call <jmp.<MSVBVM60... vbaVarDup>
004FC807 - 8B85 70FFFFFF lea eax,dword ptr ss:[ebp-0x90]
004FC80D - 50 push eax
004FC80E - 8B 45 80 lea eax,dword ptr ss:[ebp-0x80]
004FC811 - 50 push eax
004FC812 - 8B 45 90 lea eax,dword ptr ss:[ebp-0x70]
004FC815 - 50 push eax
```

The screenshot displays the OllyDbg interface with the Disassembly window active. A right-click context menu is open over the assembly code, listing various actions. The 'Go to' option is selected. The assembly code shows instructions like `push eax`, `call <msvbm60_vbaStrToAnsi>`, `Backup`, `Copy`, `Binary`, `Assemble`, `Label`, `Comment`, `Breakpoint`, `Hit trace`, `Run trace`, `New origin here`, `Go to`, `Follow in Dump`, `Search for`, `Find references to`, `View`, `Copy to executable`, `Analysis`, `Detach Process`, `Process Patcher`, `Analyze This!`, `CodeDoctor`, `IDAFicator`, `Multiline Ultimate Assembler`, `Run Script`, `Script Functions...`, `OllyCopy`, `Dump debugged process`, `Dump process (OllyDumpEx)`, `Oceans UnVirtualizer`, `Make dump of process`, `Plugins Manager`, `StrongOD`, and `Appearance`. The assembly code in the background includes instructions for pushing and calling functions, as well as backup and copy operations. The Hex dump window at the bottom shows memory addresses from 00502000 to 00502050.

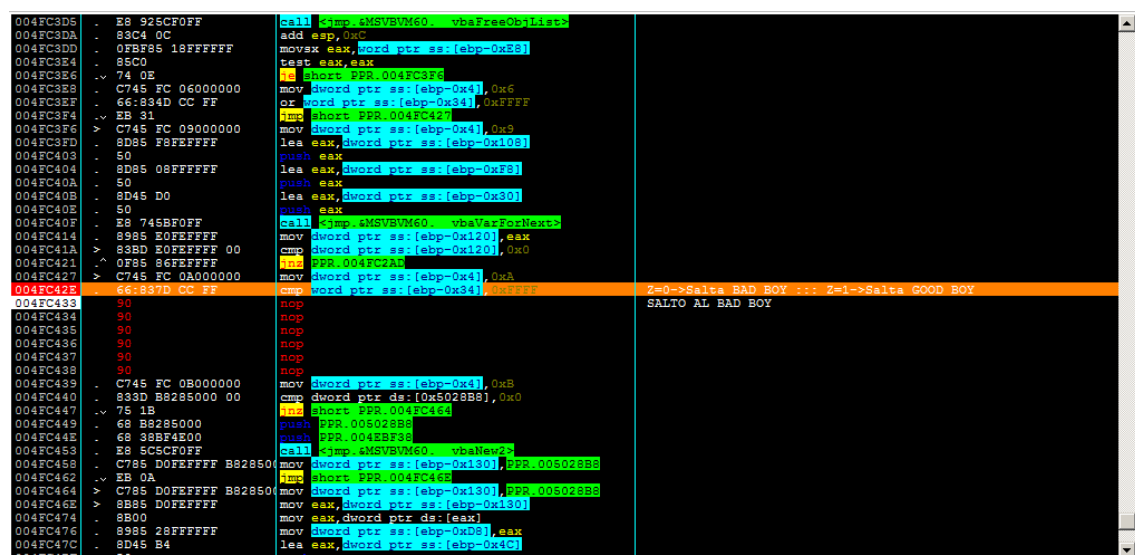
- 1) Modificar el codigo y que sea : CMP 0xffff, 0xffff
- 2) Modificar agregando NOP en la direccion de memoria: 004FC433.

004FC421	>	8B8D EBF7FFFF 00	hnd	dword ptr ss:[ebp-0x120],0xA	
004FC421	>	OF85 86FFFFFF	hnd	FPR.004FC2AD	
004FC427	>	C745 FC 0A000000 00	mov	dword ptr ss:[ebp-0x4],0xA	
004FC42E	>	55-3A7D CC FF	mov	word ptr ss:[ebp-0x34],0xFFFF	Z=0->Salta BAD BOY ; Z=1->Salta GOOD BOY
004FC433	>	OF85 61030000	hnd	FPR.004FC79A	
004FC438	>	C745 FC 0B000000 00	mov	dword ptr ss:[ebp-0x4],0xB	SALTO AL BAD BOY
004FC440	>	83BD B8285000 00	cmp	dword ptr ds:[0x5028B8],0x0	
004FC447	>	75 1B	hnd	short FPR.004FC464	
004FC449	>	68 B8285000	push	FPR.005028B8	
004FC44E	>	68 3BBF4E00	push	FPR.004EBF38	
004FC453	>	E8 5C5CF0FF	call	kjmp.4MSVBVM60_vbaNew2>	
004FC458	>	C7B5 D0FFFFFF B82850	mov	dword ptr ss:[ebp-0x130],FPR.005028B8	
004FC462	>	EB 0A	hnd	short FPR.004FC46E	
004FC464	>	C7B5 D0FFFFFF B82850	mov	dword ptr ss:[ebp-0x130],FPR.005028B8	
004FC46E	>	8B85 D0FFFFFF	mov	eax,dword ptr ss:[ebp-0x130]	
004FC474	>	8B00	mov	eax,dword ptr ds:[eax]	
004FC476	>	8B85 28FFFFFF	mov	dword ptr ss:[ebp-0xD8],eax	

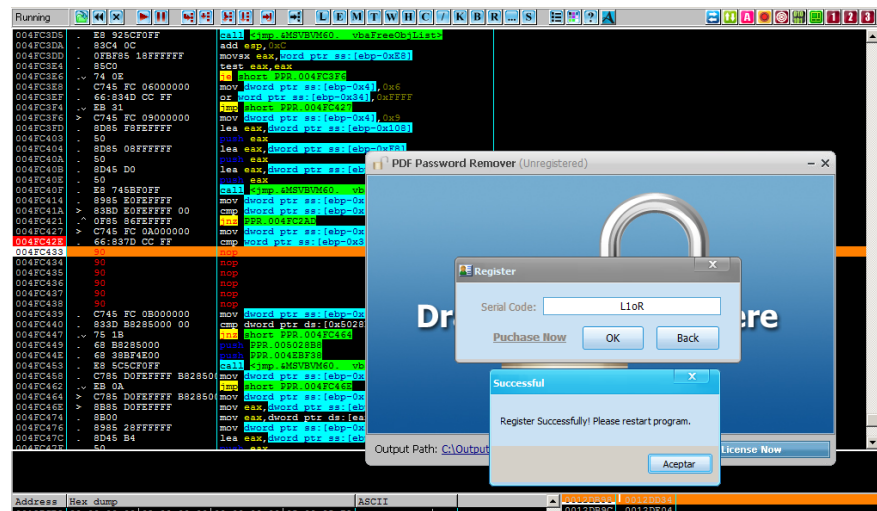
En mi caso utilize la segunda opcion, convertir el salto JNZ con direccion de memoria 004FC433 a NOPs, para que no salte y asi se vaya al GOOD BOY.



Como veras al final queda con 6 NOPs, como se comento esto realizar que no se haga ningun salto por eso le dice : NO OPERATION (No operación).

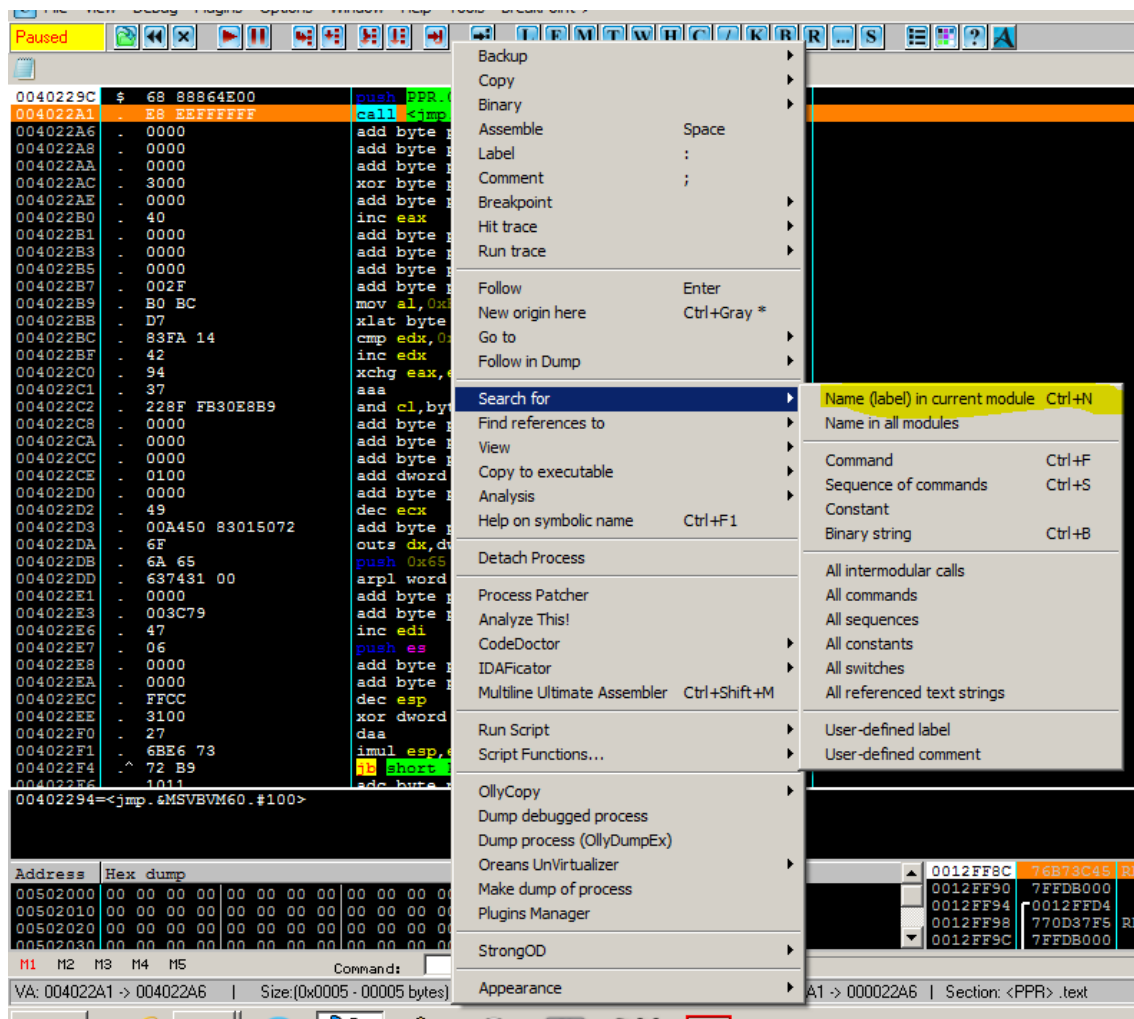


Para validar que nuestra modificacion es la correcta, ejecutamos el software con F9, y vemos que nos aparece que el GOOD BOY, reto vencido, veremos otro metodo, asi que no se aburran y procedamos.



SEGUNDO METODO.

Primero comenzamos a depurar el software, y luego vamos a buscar un API.



En este caso buscamos este API llamado VBASTRCMP que se encuentra en la direccion de memoria 004010F4 , lo que realiza es comparar dos string,

Address	Section	Type	Name
00401064	.text	Import	MSUBVM60.vbaLsetFixstr
00401138	.text	Import	MSUBVM60.vbaNew
004011C8	.text	Import	MSUBVM60.vbaNew2
004010D8	.text	Import	MSUBVM60.vbaNextEachCollobj
004010A0	.text	Import	MSUBVM60.vbaObjSet
004010B0	.text	Import	MSUBVM60.vbaObjSetAddrRef
0040110C	.text	Import	MSUBVM60.vbaObjVar
0040109C	.text	Import	MSUBVM60.vbaOnError
00401158	.text	Import	MSUBVM60.vbaPrintFile
004011C4	.text	Import	MSUBVM60.vbaR8Str
00401040	.text	Import	MSUBVM60.vbaRaiseEvent
00401050	.text	Import	MSUBVM60.vbaRecAnsiToUni
0040126C	.text	Import	MSUBVM60.vbaRecAssign
00401070	.text	Import	MSUBVM60.vbaRecDestruct
00401238	.text	Import	MSUBVM60.vbaRecDestructAnsi
00401130	.text	Import	MSUBVM60.vbaRecUniToAnsi
0040106C	.text	Import	MSUBVM60.vbaSetSystemError
00401060	.text	Import	MSUBVM60.vbaStrCat
004010F4	.text	Import	MSUBVM60.vbaStrCmp
004011E0	.text	Import	MSUBVM60.vbaStrCopy

Hacemos clic derecho en 004010F4 -> Follow import in Disassembler, nos iremos a ver el codigo en el area de desemsanblado.

0040106C	.text	Import	MSUBVM60.vbaSetSystemError
00401060	.text	Import	MSUBVM60.vbaStrCat
004010F4	.text	Import	MSUBVM60.vbaStrCmp
004011E0	.text	Import	MSUBVM60.vbaStrCopy
004010BC	.text	Import	MSUBVM60.vbaStrFixstr
00401008	.text	Import	MSUBVM60.vbaStrI2
00401014	.text	Import	MSUBVM60.vbaStrI4
00401248	.text	Import	MSUBVM60.vbaStrMove
0040112C	.text	Import	MSUBVM60.vbaStrR8
00401214	.text	Import	MSUBVM60.vbaStrToAnsi
00401160	.text	Import	MSUBVM60.vbaStrToUnicode
0040124C	.text	Import	MSUBVM60.vbaStrVarCopy
00401024	.text	Import	MSUBVM60.vbaStrVarMove
00401198	.text	Import	MSUBVM60.vbaStrVarVal
00401194	.text	Import	MSUBVM60.vbaUbound
0040113C	.text	Import	MSUBVM60.vbaUII2
0040119C	.text	Import	MSUBVM60.vbaVarCat
00401228	.text	Import	MSUBVM60.vbaVarCopy
00401174	.text	Import	MSUBVM60.vbaVarDiv
00401218	.text	Import	MSUBVM60.vbaVarDup
00401098	.text	Import	MSUBVM60.vbaVarForInit
00401264	.text	Import	MSUBVM60.vbaVarForNext
004011D0	.text	Import	MSUBVM60.vbaVarLateMenCal
00401118	.text	Import	MSUBVM60.vbaVarLateMenSt

Actualize
 Follow in Disassembler Enter
 Follow import in Disassembler
 Follow in Dump
 Find references
 View call tree
 Help on symbolic name Ctrl+F1
 Toggle breakpoint F2
 Conditional breakpoint Shift+F2
 Conditional log breakpoint Shift+F4
 Set breakpoint on every reference
 Set log breakpoint on every reference
 Remove all breakpoints
 Copy to clipboard
 Sort by
 Appearance

Veremos en la direccion de memoria 72A29596 y 72A2959A.

72A29596	FF7424 08	push dword ptr ss:[esp+0x8]
72A2959A	FF7424 08	push dword ptr ss:[esp+0x8]
72A2959E	6A 00	push 0x0
72A295A0	E8 44E6FFFF	call MSVBVM60.vbaStrComp
72A295A5	C2 0800	retn 0x8
72A295A8	FF7424 08	push dword ptr ss:[esp+0x8]
72A295AC	FF7424 08	push dword ptr ss:[esp+0x8]
72A295B0	6A 01	push 0x1
72A295B2	E8 32E6FFFF	call MSVBVM60.vbaStrComp
72A295B7	C2 0800	retn 0x8
72A295BA	FF7424 08	push dword ptr ss:[esp+0x8]
72A295BE	FF7424 08	push dword ptr ss:[esp+0x8]
72A295C2	6A 00	push 0x0
72A295C4	E8 41EEFFFF	call MSVBVM60.72A2840A
72A295C9	0FBFC0	movsx eax,ax
72A295CC	C2 0800	retn 0x8
72A295CF	FF7424 08	push dword ptr ss:[esp+0x8]
72A295D3	FF7424 08	push dword ptr ss:[esp+0x8]
72A295D7	6A 01	push 0x1
72A295D9	E8 2CEEFFFF	call MSVBVM60.72A2840A
72A295DE	0FBFC0	movsx eax,ax
72A295E1	C2 0800	retn 0x8

Hacemos clic en PLAY o F9, para poder ver la clave en este caso escribi "123456789", en el primer PUSH con direccion de memoria 72A29596 y me muestra el dato FLA885301308611047, si seguimos dando PLAY (F9), en el primer PUSH, nos mostrara la clave.

```

72A29594 ^ EB F8 jmp short MSVBVM60.72A2958E
72A29596 FF7424 08 push dword ptr ss:[esp+0x8]
72A2959A FF7424 08 push dword ptr ss:[esp+0x8]
72A2959E 6A 00 push 0x0
72A295A0 E8 44E6FFFF call MSVBVM60.vbaStrComp
Stack ss:[0012E68C]=0020A51C, (UNICODE "FLA885301308611047")

```

```

72A29594 ^ EB F8 jmp short MSVBVM60.72A2958E
72A29596 FF7424 08 push dword ptr ss:[esp+0x8]
72A2959A FF7424 08 push dword ptr ss:[esp+0x8]
72A2959E 6A 00 push 0x0
72A295A0 E8 44E6FFFF call MSVBVM60.vbaStrComp
Stack ss:[0012E688]=002093CC, (UNICODE "123456789")

```

```

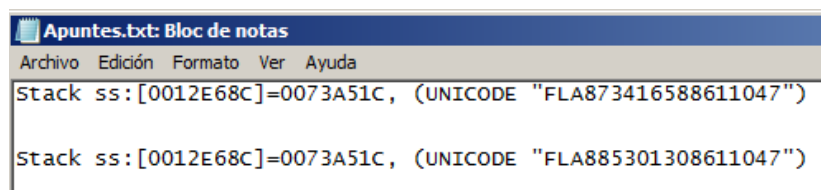
72A29594 ^ EB F8 jmp short MSVBVM60.72A2958E
72A29596 FF7424 08 push dword ptr ss:[esp+0x8]
72A2959A FF7424 08 push dword ptr ss:[esp+0x8]
72A2959E 6A 00 push 0x0
72A295A0 E8 44E6FFFF call MSVBVM60.vbaStrComp
72A295A6 C3 0000 ret 0
Stack ss:[0012E68C]=0020A51C, (UNICODE "FLA837900198611047")

```

```

72A29592 93C0 xor eax, eax
72A29594 ^ EB F8 jmp short MSVBVM60.72A2958E
72A29596 FF7424 08 push dword ptr ss:[esp+0x8]
72A2959A FF7424 08 push dword ptr ss:[esp+0x8]
72A2959E 6A 00 push 0x0
72A295A0 E8 44E6FFFF call MSVBVM60.vbaStrComp
72A295A6 C3 0000 ret 0
Stack ss:[0012E68C]=0020A51C, (UNICODE "FLA850510518611047")

```



Valimos copiando una de las claves para poder visualizar si nos aparece el GOOD BOY.



Graciass por su apoyo constante y un saludos. HAPPY CRACKING.