

# Crackeando acunetix v8



Alejandro torres  
(torrescrack)

researcher

Torrescrack.blogspot.com

- **VICTIMA: ACUNETIX V8 2012**
- **PROTECCION: SERIAL NAME**
- **URL DESCARGA:**  
[http://www.acunetix.com/download/fullver8/2012\\_11\\_13\\_01\\_webvulnscan8.exe](http://www.acunetix.com/download/fullver8/2012_11_13_01_webvulnscan8.exe)



<http://www.facebook.com/yo.torrescrack>



<https://twitter.com/TorresCrack248>



[www.torrescrack.blogspot.com](http://www.torrescrack.blogspot.com)

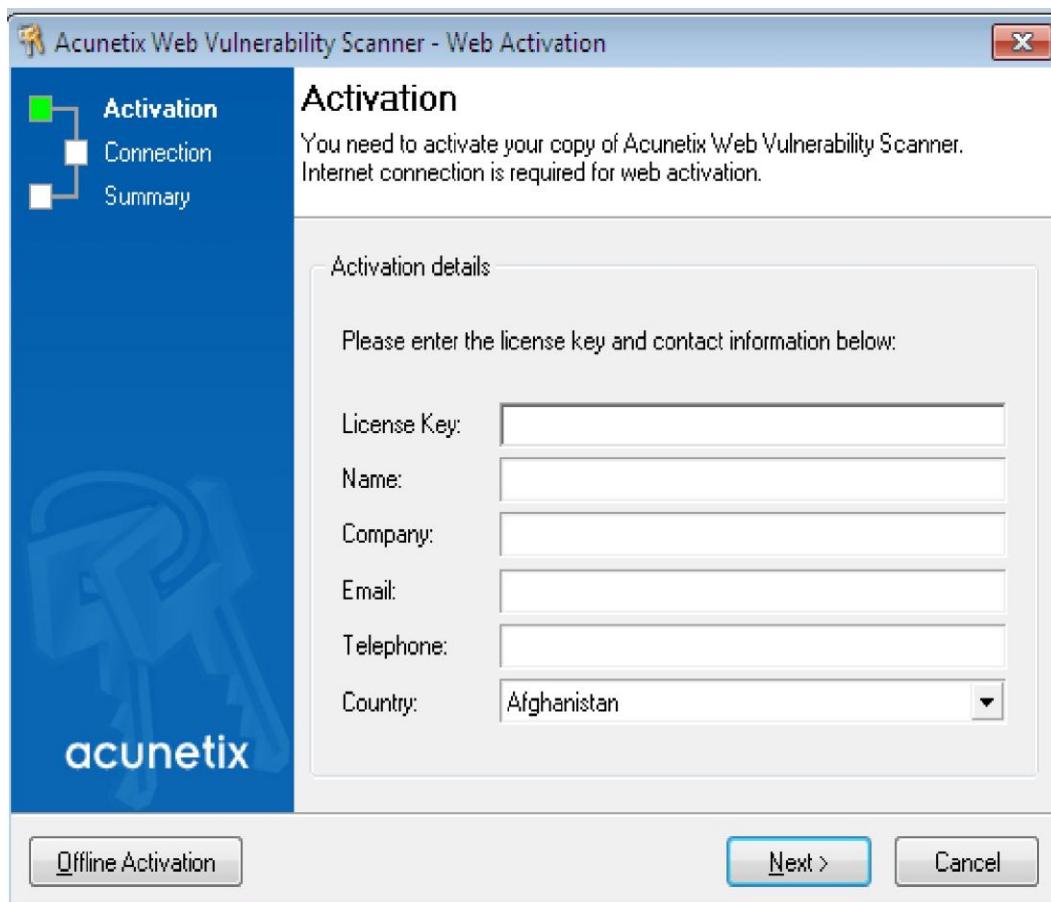
## INTRODUCCION:

***Este tutorial es únicamente con fines educativos y no me hago responsable por el mal uso de su contenido.***

Bien , este tutorial lo hice ya que no existe algun tute en la red de como atacar este software , y ademas de esto, resulta que yo estaba buscando por la web algun escaneador de vulnerabilidades web reciente y bueno , lei en muchos sitios que este software acunetix es uno de los mejores pero el problema es que yo queria probarlo y en la pagina tienen solo una version free que no escanea completamente como deberia y la otra version completa tiene un costo , en la red hay un crack o parche pero no me anime a bajarlo , asi que me puse a darle un vistazo para obtener mi propia version liberada y demostrar que este software especializado en buscar vulnerabilidades web es muy vulnerable.

## EXPLORANDO AL ENEMIGO:

Primero Instalamos y se me ocurrio correr el software para ver que tipo de proteccion o limitacion utilizaba , veamos.



Uhmm vemos que antes de iniciar el soft me pide datos de la licencia para activar online y si no ingresas nada o algo erroneo no entras al soft , aparte de caro \$\$ ni siquiera te dejan probar el software por unos dias en su version completa , vayamos a la carpeta donde se instalo el software para ver que mas tiene

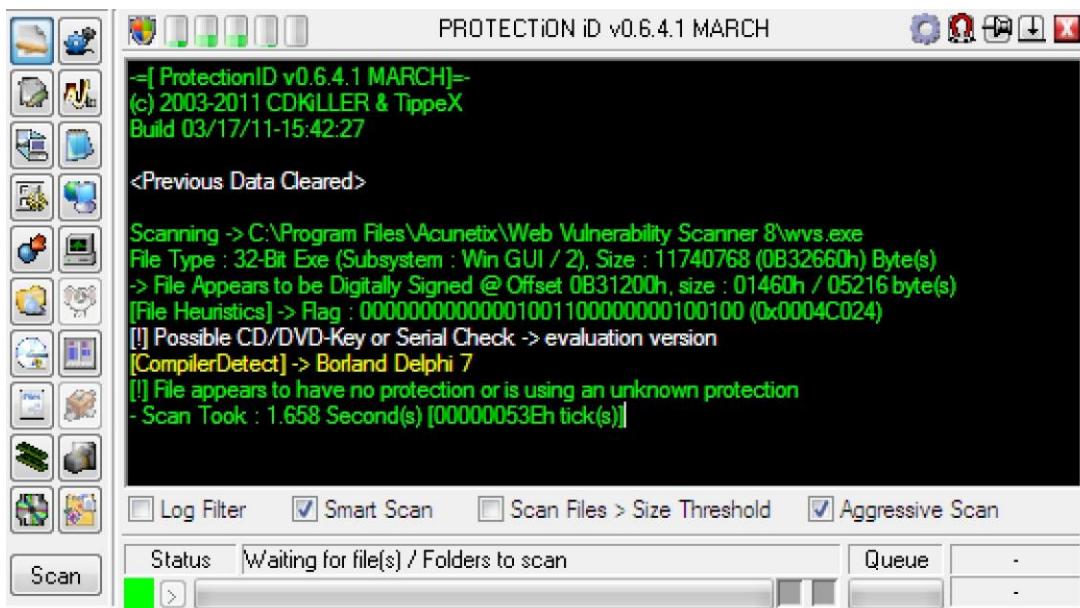
 Activation	13/11/2012 03:10 .
 DelZip190.dll	19/01/2012 02:36 .
 ffacuscan.xpi	10/04/2012 02:41 .
 libeay32.dll	27/07/2011 11:09 a
 license	14/02/2012 10:46 a
 lsl	13/11/2012 03:10 .
 pcre.dll	27/07/2011 11:09 a
 Reporter	13/11/2012 03:10 .
 reporter_console	13/11/2012 03:10 .
 SciLexer.dll	27/07/2011 11:09 a
 ssleay32.dll	27/07/2011 11:09 a
 unins000.dat	25/11/2012 09:40 .
 unins000	25/11/2012 09:39 .
 UnInstall	13/11/2012 03:10 .
 ve	13/11/2012 03:10 .
 wvs	13/11/2012 03:10 .
 wvs	25/11/2012 09:40 .
 wvs_console	13/11/2012 03:10 .
 WVSScheduler	13/11/2012 03:10 .

Bien no tiene muchas cosas , pero hay se encuentra un ejecutable llamado “Activation.exe” que al correrlo nos damos cuenta que es la misma pantalla que vimos anteriormente y el ejecutable principal es “wvs.exe”.

Bien , entonces ya sabemos que el programa principal es

"wvs.exe" entonces empezamos por hay, ya que de alguna forma el programa compara si esta registrado o licenciado y de lo contrario carga el otro ejecutable para iniciar con el registro o activacion.

Vamos a examinarlo con un escaneador en este caso el "Protection ID" para saber si el programa esta empacado y ver en que esta compilado , veamos.



Bien no usa algun packer/protector , solo detecto un posible deteccion de CD KEY o serial check , que en este caso es lo segundo.

Abramos nuestro debugger y empezemos .

The screenshot shows the Immunity Debugger interface with assembly code. The assembly window displays the following code:

```
00ABF0F0 55      PUSH EBP
00ABF0F1 8BEC    MOU EBP,ESP
00ABF0F3 B9 2E000000 MOU ECX,2E
00ABF0F8 6A 00    PUSH 0
00ABF0FA 6A 00    PUSH 0
00ABF0FC 49      DEC ECX
00ABF0FD ^ 75 F9 JNZ SHORT wws.00ABF0F8
00ABF0FF 53      PUSH EBX
00ABF100 56      PUSH ESI
00ABF101 57      PUSH EDI
00ABF102 B8 1BD6AB00 MOU EAX,wws.00ABD618
00ABF107 E8 989194FF CALL wws.004082A4
00ABF10C 33C0    XOR EAX,EAX
00ABF10E 55      PUSH EBP
00ABF10F 68 3107AC00 PUSH wws.00AC0731
00ABF114 64:FF30 PUSH DWORD PTR FS:[EAX]
00ABF117 64:8920 MOU DWORD PTR FS:[EAX],ESP
00ABF118 E8 ED1A95FF CALL wws.00410C0C
00ABF11F 83C4 F8 ADD ESP,-8
00ABF122 DD1C24 RETP QWORD PTR SS:[ESP]
```

Al iniciar lo primero que se me ocurrio con la info obtenida anteriormente fue buscar en las strings el nombre del proceso que ejecuta , en este caso “Activation.exe” veamos lo que encontramos

The screenshot shows the Immunity Debugger strings search results. The found strings include:

- ASCII "User Objects: %d"
- ASCII "\0\0"
- ASCII "FastMM4: %d Kb"
- ASCII "Debugger detected! The application will now terminate.\nPlease close the debugger and restart the application.\n"
- ASCII "\Activation.exe"
- ASCII "Acunetix Ltd."
- ASCII "Signature error"
- ASCII "This application has been tampered with!\nThe activation signature doesn't match!\nYou may be a victim of softw.
- ASCII "Parse settings ..."
- ASCII "Rewrite log files ..."
- ASCII "Init logging ..."

El software tiene algun metodo antidebug que detecta si esta siendo debuggado o si hay algun parche , todavia no sabemos como lo hace pero sigamos buscando mas strings haber si hay otras llamadas y alrato regresamos :P , colocamos en el inicio de esa rutina un BreakPoint y sigamos..

The screenshot shows the assembly view of the Immunity Debugger. The assembly code is as follows:

```

002C3B8 $ 55 PUSH EBP
002C3B9 . 8BEC MOV EBP,ESP
002C3B9 . B9 09000000 MOU ECX,9
002C3B0 > 6A 00 PUSH 0
002C3B2 . 6A 00 PUSH 0
002C3B4 . 49 DEC ECX
002C3B5 .^ 75 F9 JNZ SHORT wvs.00A2C3B0
002C3B7 . 51 PUSH ECX
002C3B8 . 53 PUSH EBX
002C3B9 . 56 PUSH ESI
002C3B9 . 57 PUSH EDI
002C3BB . 8BD8 MOU EBX,EAX
002C3BD . 8B35 143DAEEF MOU ES,IWORD PTR DS:[AE3D14]
002C3C3 . 33C0 XOR EAX,EAX
002C3C5 . 55 PUSH EBP
002C3C6 . 68 2BCAA200 PUSH wvs.00A2CA2B
002C3CB . 64:FF30 PUSH DWORD PTR FS:[EAX]
002C3CE . 64:8920 MOV DWORD PTR FS:[EAX],ESP
002C3D1 . C683 9CB50000 MOU BYTE PTR DS:[EBX+59C1],0
002C3D8 . C683 B4040000 MOU BYTE PTR DS:[EBX+4B41],0
002C3DF . E8 58E1FFFF CALL wvs.00A2A53C
002C3E4 . 84C0 TEST AL,AL
002C3E6 .. 74 26 JE SHORT wvs.00A2C40E
002C3E8 . 6A 00 PUSH 0
002C3E9 . 66:BB0D 3CC0A1 MOU CX,WORD PTR DS:[A2CA3C1]
002C3F1 . 33D2 XOR EDX,EDX
002C3F3 . B8 48CAA200 MOU EAX,wvs.00A2CA48
002C3F8 . E8 CB6796FF CALL wvs.00492BC8
002C3FD . A1 6435AE00 MOU EAX,DWORD PTR DS:[AE3564]
002C402 . 8B00 MOU EAX,DWORD PTR DS:[EAX]
002C404 . E8 3ECE0AC0E CALL wvs.004F9348

```

In the bottom right corner, there is a yellow box containing the following text:

Arg1 = 00000000  
 ASCII "Debugger detected! The application will now terminate"  
 wvs.00492BC8

Seguimos buscando y aqui tenemos algo interesante si ven la imagen tenemos la zona mas sopechosa de todas pues aqui

vemos cuando decide si expira o no si esta activado o no , vayamos a esa zona haber que veemos

```

ASCII  "/loginseq"
ASCII  "/?"
ASCII  "/scan"
ASCII  "/scanwsdl"
ASCII  "/scanfromcrawl"
ASCII  "acunetix_root_ca.key"
ASCII  "acunetix_root_ca.cer"
ASCII  "certificate_cache.dat"
ASCII  "The Acunetix WVS root certificate was not found. Please
ASCII  "ScanOutput.wvs"
ASCII  "CrawlOutput.cwl"
ASCII  "dd-mm-yyyy_hh-nn-ss"
ASCII  "Invalid password"
ASCII  "Application is not activated!"
ASCII  "Your Acunetix WVS license will expire in "
ASCII  " days!"
ASCII  "Your Acunetix WVS license will expire tomorrow!"
ASCII  "Activation.exe"

3947 calls to known, 9143 calls to guessed functions

```

<pre> &gt; A1 6435AE00 MOU EAX,DWORD PTR DS:[AE35641] . 8B00 MOU EAX,DWORD PTR DS:[EAX] . E8 4D8EA3FF CALL wvs.004F91C4 . A1 383EAE00 MOU EAX,DWORD PTR DS:[AE3E381] . 8B00 MOU EAX,DWORD PTR DS:[EAX] . 8078 04 00 CMP BYTE PTR DS:[EAX+4],0 . . 75 58 JNZ SHORT wvs.00AC03DC . . 6A 01 PUSH 1 . . 68 700CAC00 PUSH wvs.00AC0C78 . . 8D95 98FEFFFF LEA EDX,DWORD PTR SS:[EBP-168] . . A1 6435AE00 MOU EAX,DWORD PTR DS:[AE35641] . . 8B00 MOU EAX,DWORD PTR DS:[EAX] . . E8 7F95A3FF CALL wvs.004F991C . . 8B85 98FEFFFF MOU EAX,DWORD PTR SS:[EBP-168] . . 8D95 9CPEPPPF LEA EDX,DWORD PTR SS:[EBP-164] . . E8 DEE894FF CALL wvs.0040EC8C . . 8B85 9CPEFFFF MOU EAX,DWORD PTR SS:[EBP-164] . . E8 A35A94FF CALL wvs.00405E5C . . 50 PUSH EAX . . 68 7C0CAC00 PUSH wvs.00AC0C7C . . 68 8C0CAC00 PUSH wvs.00AC0C8C . . 6A 00 PUSH 0 . . E8 05CE97FF CALL &lt;JMP.&amp;shell32.ShellExecuteA&gt; . . A1 6435AE00 MOU EAX,DWORD PTR DS:[AE35641] . . 8B00 MOU EAX,DWORD PTR DS:[EAX] . . E8 718FA3FF CALL wvs.004F9348 . . E9 D5020000 JMP wvs.00AC06B1 </pre>	Parameters FileName = "Activation.exe" Operation = "open" hWnd = NULL ShellExecuteA
--	---

Que bien!! , si ven hay arriba tenemos una direccion de memoria que dentro de ella se encuentra un buffer y compara el contenido de esa direccion con 0 y si el valor es diferente ya no llama al "activation.exe" supongo que al haber licencia debe valer 1 el contenido del buffer.

En mi experiencia en diferentes software comercial he visto que utilizan comparaciones en diferentes partes o ubicaciones del programa para ver si el soft esta registrado, ocupa las mismas direcciones de memoria y hace las mismas comparaciones por otras partes del software , pensando que es un metodo anti-cracking por si alguien se le ocurre parchear unicamente ese salto condicional , o bien tambien en algun momento debe meterle contenido a ese buffer ,veamos si mis sospechas son ciertas busquemos si alguien mas usa esa direccion de memoria , hagamos click derecho y "find references to: address constant" y veamos.

Disassembly	Comment
39E4 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
39FE MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
3H14 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
3727 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
1732 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
1779 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
1B0E MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
3530 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
3567 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
3C28 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
3C35 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
3C5A MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
3C8E MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
HEC9 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
IF94 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:1EA MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:1F9 MOU EDX, DWORD PTR DS:[AE3E381]	wvs..00AEA76C
:229 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:316 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:325 MOU EDX, DWORD PTR DS:[AE3E381]	wvs..00AEA76C
:355 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:3FF3 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:602 MOU EDX, DWORD PTR DS:[AE3E381]	wvs..00AEA76C
:632 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:172 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:17F MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:18C MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:491 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:56F MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:5ED MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:76C MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:78A MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:797 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:7A9 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:7C9 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:D68 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:D75 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:D87 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:4B6 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:503 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:511 MOU EDX, DWORD PTR DS:[AE3E381]	wvs..00AEA76C
:51F MOU EDX, DWORD PTR DS:[AE3E381]	wvs..00AEA76C
:52D MOU EDX, DWORD PTR DS:[AE3E381]	wvs..00AEA76C
:5DE MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:5ED MOU EDX, DWORD PTR DS:[AE3E381]	wvs..00AEA76C
:C8F MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:1HA MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:1B9 MOU EDX, DWORD PTR DS:[AE3E381]	wvs..00AEA76C
:1F7 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:FB2 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:FC1 MOU EDX, DWORD PTR DS:[AE3E381]	wvs..00AEA76C
:FE9 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:88D MOU EDX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:89C MOU EDX, DWORD PTR DS:[AE3E381]	wvs..00AEA76C
:8AE MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:574 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:117 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:13F MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:14C MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:158 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:166 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:17B MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:1AF MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
1877 MOU EAX, DWORD PTR DS:[AE3E381]	Initial CPU selection
:9DC MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:9ED MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:9F9 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C
:A07 MOU EAX, DWORD PTR DS:[AE3E381]	[00AE3E381]=00AEA76C

Como les decia mis sospechas eran ciertas pues el software utiliza esa direccion de memoria en muchisimos otros lugares , posiblemente para hacer chequeos de la licencia , pongamos un Break Point en todos ellos y ahora si ejecutemos el Olly debugger y veamos en donde parara , veamos.

```

MOV EAX,DWORD PTR DS:[AE3E38]
MOV EAX,DWORD PTR DS:[EAX]
CALL wws.006A9FB8
CALL wws.00ABBE34
TEST AL,AL
JNZ SHORT wws.00ABF5C4
MOV EAX,DWORD PTR DS:[AE2A9C]
CMP BYTE PTR DS:[EAX],0
JE SHORT wws.00ABF59F
MOV EAX,wws.00AC08AC
CALL wws.006290B4
JMP SHORT wws.00ABF5B4
PUSH B
MOU CX,WORD PTR DS:[AC08E4]
MOU DL,1
MOU EAX,wws.00AC08AC
CALL wws.00492BC8
MOU EAX,DWORD PTR DS:[AE2344]
MOU DWORD PTR DS:[EAX],309
JMP wws.00AC06B1
XOR EAX,EAX
PUSH EBP

```

ASCII "This version does not permits running more instances.  
Arg1 = 00000000  
ASCII "This version does not permits running more instances.  
wws.00492BC8

Ok como vemos que hace unas comparaciones y de hay verifica si tu version es valida para correr varias instacias en el mismo pc , creo que la version mas completa te permite hasta 10 instancias corriendo, bien sigamos viendo donde mas detiene la ejecucion y ver que otras comparaciones hace.

00AC0377	. A1 383EAE00	MOU EAX,DWORD PTR DS:[AE3E38]
00AC037C	. 8B00	MOU EAX,DWORD PTR DS:[EAX]
00AC037E	. 8078 04 00	CMP BYTE PTR DS:[EAX+4],0
00AC0382	. 75 58	JNZ SHORT wws.00AC03DC
00AC0384	. 6A 01	PUSH 1
00AC0386	. 68 780CAC00	PUSH wws.00AC0C78
00AC038B	. 8D95 98FFFFF1	LEA EDX,DWORD PTR SS:[EBP-168]
00AC0391	. A1 6435AE00	MOU EAX,DWORD PTR DS:[AE3564]
00AC0396	. 8B00	MOU EAX,DWORD PTR DS:[EAX]

Vemos que paro donde saca de la direccion de memoria 0xAE3E38 , un buffer y compara el contenido de ese buffer+4 si es igual a 0 y de caso contrario ya no ejecuta el "activation"

Ahora bien ya tenemos el dato que en esa direccion de memoria debe valer a 1 , cambiemosle el byte a 1 .

Address	Hex dump
0212ECCC	01 00 00 00 00 00 00 00
0212ECD4	00 00 00 00 02 00 00 00
0212ECDC	00 00 00 00 00 00 00 00
0212ECE4	00 00 00 00 00 00 00 00
0212ECEC	00 00 00 00 00 00 00 00
0212ECD1	00 00 00 00 00 00 00 00

Sigamos la ejecucion del software y veamos en donde mas ocupa esa direccion de memoria , sigamos

03D2	. E8 718FH3FF	CALL wvs.004F9348	
03D7	. v E9 D5020000	JMP wvs.00AC06B1	
03DC	> A1 383EAE00	MOV EAX,DWORD PTR DS:[AE3E38]	
03E1	. 8B00	MOV EAX,DWORD PTR DS:[EAX]	
03E3	. 8978 48 00	CMP BYTE PTR DS:[EAX+48],0	
03E7	. v 0F84 8C000000	JE wvs.00AC0479	
03ED	. A1 383EAE00	MOV EAX,DWORD PTR DS:[AE3E38]	
03F2	. 8B00	MOV EAX,DWORD PTR DS:[EAX]	
03F4	. E8 A793BEFF	CALL wvs.006A97A0	
03F9	. A1 383EAE00	MOV EAX,DWORD PTR DS:[AE3E38]	
03FB	. 8B00	MOV EAX,DWORD PTR DS:[EAX]	
0400	. 8B58 3C	MOV EBX,DWORD PTR DS:[EAX+3C]	
0403	. 85DB	TEST EBX,EBX	
0405	. v 7E 54	JLE SHORT wvs.00AC045B	
0407	. A1 383EAE00	MOV EAX,DWORD PTR DS:[AE3E38]	
040C	. 83FB 07	CMP EBX,7	
040F	. v 7D 68	JGE SHORT wvs.00AC0479	
0411	. 6A 00	PUSH 0	
0413	. 68 040CAC00	PUSH wvs.00AC0C04	
0418	. 8D95 90FEFFFF	LEA EDX,DWORD PTR SS:[EBP-170]	
041E	. A1 383EAE00	MOV EAX,DWORD PTR DS:[AE3E38]	
0423	. 8BC3	MOV EAX,EBX	
0425	. E8 A2DC94FF	CALL wvs.0040E0CC	
042A	. FFB5 90FEFFFF	PUSH DWORD PTR SS:[EBP-170]	
0430	. 68 380CAC00	PUSH wvs.00AC0C38	
0435	. 8D85 94FEFFFF	LEA EAX,DWORD PTR SS:[EBP-16C]	
0438	. B8 03000000	MOU EDX,3	

ASCII "Your Acunetix WVS license will expire in "

ASCII " days!"

vemos que compara el mismo buffer + 48 si es igual a 0 y de ser correcto salta el mensaje que nos diria cuando caduca la licencia, en este caso el buffer+48 vale 0 asi que no nos preocupamos pero hay que tenerlo en cuenta, seguimos la ejecucion haber que otra sorpresa nos tiene el software.

. 64:8920	MOU DWORD PTR FS:[EAX],ESP	
. A1 383EAEE00	MOU EAX,DWORD PTR DS:[AE3E38]	
. 8B00	MOU EAX,DWORD PTR DS:[EAX]	
. 8078 04 00	CMP BYTE PTR DS:[EAX+4],0	
.~ 0F84 17010000	JE wws.00988A20	
. E8 FE82A8FF	CALL wws.00410C0C	
. 83C4 F8	ADD ESP,-8	
. DD1C24	FSTP QWORD PTR SS:[ESP]	
. 9B	WAIT	
. 6A 00	PUSH 0	Arg2 <8-byte>
. 33C9	XOR ECX,ECX	Arg1 = 00000000
. 33D2	XOR EDX,EDX	
. 33C0	XOR EAX,EAX	
. E8 B208C6FF	CALL wws.005E91D4	wws.005E91D4
. DD5D F0	FSTP QWORD PTR SS:[EBP-10]	
. 9B	WAIT	
. A1 383EAEE00	MOU EAX,DWORD PTR DS:[AE3E38]	
. 8B00	MOU EAX,DWORD PTR DS:[EAX]	
. 8B50 68	MOU EDX,DWORD PTR DS:[EAX+68]	
. 8955 F8	MOU DWORD PTR SS:[EBP-8],EDX	
. 8B50 6C	MOU EDX,DWORD PTR DS:[EAX+6C]	
. 8955 FC	MOU DWORD PTR SS:[EBP-4],EDX	
. DD45 F8	FLD QWORD PTR SS:[EBP-8]	
. D81D 488A9800	FCOMP DWORD PTR DS:[988A48]	
. DFE0	FSTSW AX	
. 9E	SAHF	
.~ 0F86 C9000000	JBE wws.00988A14	
. FF75 F4	PUSH DWORD PTR SS:[EBP-C]	Arg4
. FF75 F0	PUSH DWORD PTR SS:[EBP-10]	Arg3
. FF75 FC	PUSH DWORD PTR SS:[EBP-4]	Arg2
. FF75 F8	PUSH DWORD PTR SS:[EBP-8]	Arg1
. E8 6406C6FF	CALL wws.005E8FC0	wws.005E8FC0
. 8BD8	MOU EBX,EAX	
. FF75 F4	PUSH DWORD PTR SS:[EBP-C]	Arg4

Bien paro la ejecucion en otra comparacion al mismo buffer+4 con 0, pero en este caso esta valiendo 1 (true) y de ser asi vemos mas abajo que nos saltea este pedazo de codigo

The screenshot shows a debugger interface with two main panes. The left pane displays assembly code in green font, and the right pane shows memory dumps in blue font. The assembly code is as follows:

```
0000000000401201 MOU EDX,DWORD PTR DS:[EDX]
0000000000401202 LEA EAX,DWORD PTR SS:[EBP-14]
0000000000401203 MOU ECX,wvs.00988A54
0000000000401204 CALL wvs.00405CA8
0000000000401205 MOU EDX,DWORD PTR SS:[EBP-14]
0000000000401206 MOU EAX,DWORD PTR DS:[AEDBBC]
0000000000401207 CALL wvs.00987C94
0000000000401208 MOU EAX,DWORD PTR DS:[AE3E38]
0000000000401209 MOU EAX,DWORD PTR DS:[EAX]
000000000040120A MOU EAX,DWORD PTR DS:[EAX+24]
000000000040120B PUSH EAX
000000000040120C MOU EAX,DWORD PTR DS:[AE3E38]
000000000040120D MOU EAX,DWORD PTR DS:[EAX]
000000000040120E MOU EAX,DWORD PTR DS:[EAX+30]
000000000040120F PUSH EAX
0000000000401210 MOU EAX,DWORD PTR DS:[AE3E38]
0000000000401211 MOU EAX,DWORD PTR DS:[EAX]
0000000000401212 MOU EAX,DWORD PTR DS:[EAX+2C]
0000000000401213 PUSH EAX
0000000000401214 LEA EDX,DWORD PTR SS:[EBP-18]
0000000000401215 MOU EAX,DWORD PTR DS:[AE3E38]
0000000000401216 MOU EAX,DWORD PTR DS:[EAX]
0000000000401217 MOU EAX,DWORD PTR DS:[EAX+34]
0000000000401218 CALL wvs.006A54D4
0000000000401219 MOU EAX,DWORD PTR SS:[EBP-18]
000000000040121A PUSH EAX
000000000040121B MOU EAX,DWORD PTR DS:[AEDBC0]
000000000040121C PUSH EAX
000000000040121D PUSH wvs.009896D8
000000000040121E MOU EAX,DWORD PTR DS:[AE3E38]
000000000040121F MOU EAX,DWORD PTR DS:[EAX]
0000000000401220 MOU EDX,DWORD PTR DS:[EAX+28]
0000000000401221 LEA EAX,DWORD PTR SS:[EBP-1C]
0000000000401222 MOU ECX,wvs.00988A6C
0000000000401223 CALL wvs.00405CA8
0000000000401224 MOU ECX,DWORD PTR SS:[EBP-1C]
```

The right pane shows memory dump segments:

- Segment 1: ASCII "Activation.xml"
- Segment 2: ASCII " (AutoActivation)"

Bien , si recuerdan el contenido del buffer se quedo valiendo 1 por lo tanto tenemos que entrar en la rutina , pero no nos pondremos a investigar , seria muy extenso el tutorial y no creo que haga alguna otro chequeo , asi que demos run (f9) que es lo que hace que sigamos la ejecucion

Sigamos la ejecucion de codigo y veamos si existen otros chequeos , sigamos...

B	. E8 3194BFFF	CALL wws.006278D0	
B	. 837D F8 00	CMP DWORD PTR SS:[EBP-8],0	
B	.~ 0F84 09010000	JE wws.00A2E5B2	
B	. 80BE 9C050000	CMP BYTE PTR DS:[ESI+59C],0	
B	.~ 0F84 FC000000	JE wws.00A2E5B2	
B	5 . A1 383EAEE00	MOU EAX,DWORD PTR DS:[AE3E38]	
B	. 8B00	MOU EAX,DWORD PTR DS:[EAX]	
D	. 8378 10 64	CMP DWORD PTR DS:[EAX+10],64	
L	.~ 0F84 81000000	JE wws.00A2E548	
P	. A1 A432AE00	MOU EAX,DWORD PTR DS:[AE32A4]	
C	. 8B00	MOU EAX,DWORD PTR DS:[EAX]	
E	. BA E0E5A200	MOU EDX,wws.00A2E5E0	ASCII "yes"
B	. E8 D0789DFF	CALL wws.00405DA8	
B	.~ 74 09	JE SHORT wws.00A2E4E3	
A	. 80BE 94050000	CMP BYTE PTR DS:[ESI+594],0	
L	.~ 74 58	JE SHORT wws.00A2E53B	
B	> B9 ECE5A200	MOU ECX,wws.00A2E5EC	ASCII "no"
B	. BA F8E5A200	MOU EDX,wws.00A2E5F8	ASCII "IsFirstTime"
D	. 33C0	XOR EAX,EAX	
F	. E8 B4A4C7FF	CALL wws.006A89A8	
A	. A1 A432AE00	MOU EAX,DWORD PTR DS:[AE32A4]	
P	. BA ECE5A200	MOU EDX,wws.00A2E5EC	ASCII "no"
E	. E8 DD749DFF	CALL wws.004059E0	
B	3 . A1 383EAEE00	MOU EAX,DWORD PTR DS:[AE3E38]	
B	. 8B00	MOU EAX,DWORD PTR DS:[EAX]	
D	. 8A40 48	MOU AL,BYTE PTR DS:[EAX+48]	
F	. 84C0	TEST AL,AL	
F	.~ 74 1C	JE SHORT wws.00A2E52D	
B	1 . 8B15 383EAEE00	MOU EDX,DWORD PTR DS:[AE3E38]	wws.00A2E76C
D	OP12	MOU EDX,DWORD PTR DS:[EDX]	

Bien paro en este lugar donde hace otro chequeo pero esta vez al buffer+10 y lo compara con 64 (hexa) , al parecer es donde chequea , si es la primera vez que lo abrimos el software (eso creo) , la rutina es muy grande y no me pondre a investigarla pues a mi solo me interesa enfocarme en donde hace chequeos a esa direccion de memoria , asi que seguimos la ejecucion...

D491	- A1 383EAE00	MOU EAX,DWORD PTR DS:[AE3E38]	
D496	- 8B00	MOU EAX,DWORD PTR DS:[EAX]	
D498	- 8B40 10	MOU EAX,DWORD PTR DS:[EAX+10]	
D49B	- 83F8 03	CMP EAX,3	
D49E	~ 7F 19	JG SHORT wvs.00A2D4B9	Switch (cases 0..64)
D4A0	~ 0F84 86010000	JE wvs.00A2D62C	
D4A6	- 83E8 01	SUB EAX,1	
D4A9	~ 0F82 FF000000	JB wvs.00A2D5AE	
D4AF	~ 74 ?F	JE SHORT wvs.00A2D530	
D4B1	- 48	DEC EAX	
D4B2	~ 74 23	JE SHORT wvs.00A2D4D7	
D4B4	~ E9 A9020000	JMP wvs.00A2D762	
D4B9	> 93E8 04	SUB EAX,4	
D4BC	~ 0F84 C3010000	JE wvs.00A2D685	
D4C2	- 48	DEC EAX	
D4C3	~ 0F84 15020000	JE wvs.00A2D6DE	
D4C9	- 83E8 5F	SUB EAX,5F	
D4CC	~ 0F84 62020000	JE wvs.00A2D734	
D4D2	~ E9 8B020000	JMP wvs.00A2D762	
D4D7	> 8D55 FC	LEA EDX,DWORD PTR SS:[EBP-4]	Case 2 of switch 00A2D49B
D4DA	- 8BC3	MOU EAX,EBX	
D4DC	- E8 F797AFF	CALL wvs.004D6CD8	
D4E1	- 8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]	
D4E4	- BA 4CDFA200	MOU EDX,wvs.00A2DF4C	ASCII " (Free Edition)"
D4E9	- E8 76879DFF	CALL wvs.00405C64	
D4EE	- 8B55 FC	MOU EDX,DWORD PTR SS:[EBP-4]	
D4F1	- 8BC3	MOU EAX,EBX	
D4F3	- E8 1098AFF	CALL wvs.004D6D08	
D4F8	- A1 C02EAE00	MOU EAX,DWORD PTR DS:[AE2EC0]	
D4FD	- BA 64DFA200	MOU EDX,wvs.00A2DF64	ASCII "Evaluation"
D502	- E8 D9849DFF	CALL wvs.004059E0	
D507	- A1 BC2EAE00	MOU EAX,DWORD PTR DS:[AE2EBC]	
D50C	- BA 78DFA200	MOU EDX,wvs.00A2DF78	ASCII "Free Edition"
D511	- E8 CA849DFF	CALL wvs.004059E0	
D516	- 8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
D519	- E8 32FCFFFF	CALL wvs.00A2D150	
D51E	- 8B55 F8	MOU EDX,DWORD PTR SS:[EBP-8]	
D521	- A1 6833AE00	MOU EAX,DWORD PTR DS:[AE3368]	
D526	- E8 B5849DFF	CALL wvs.004059E0	

2D521 . A1 6833AE00 MOU EDX, DWORD PTR DS:[AE3368]	
2D526 . E8 B5849DFF CALL wvs.00A2D49B	Case 1 of switch 00A2D49B
2D52B .~ E9 32020000 JMP wvs.00A2D762	
2D530 > 8D55 F4 LEA EDX, DWORD PTR SS:[EBP-C]	
2D533 . 8BC3 MOU EAX, EBX	
2D535 . E8 9E97AAFF CALL wvs.004D6CD8	ASCII " <Consultant Edition>"
2D53A . 8D45 F4 LEA EAX, DWORD PTR SS:[EBP-C]	
2D53D . BA 90DF0200 MOU EDX,wvs.00A2DF90	
2D542 . E8 1D879DFF CALL wvs.00405C64	
2D547 . 8B55 F4 MOU EDX, DWORD PTR SS:[EBP-C]	
2D54A . 8BC3 MOU EAX, EBX	
2D54C . E8 B797AAFF CALL wvs.004D6D08	
2D551 . A1 C02EEAE0 MOU EAX, DWORD PTR DS:[AE2EC0]	
2D556 . BA B0DFA200 MOU EDX,wvs.00A2DFB0	
2D558 . E8 80849DFF CALL wvs.004059E0	
2D560 . A1 BC2EEAE0 MOU EAX, DWORD PTR DS:[AE2EBC]	
2D565 . BA C4DFA200 MOU EDX,wvs.00A2DFC4	
2D56A . E8 71849DFF CALL wvs.004059E0	
<b>2D56F . A1 383EEAE0 MOU EAX, DWORD PTR DS:[AE3E38]</b>	
2D574 . 8B00 MOU EAX, DWORD PTR DS:[EAX]	
2D576 . 8B40 14 MOU EAX, DWORD PTR DS:[EAX+14]	
2D579 . BA E0DFA200 MOU EDX,wvs.00A2DFE0	
2D57E . E8 25889DFF CALL wvs.00405DA8	
2D583 .~ 75 0F JNZ SHORT wvs.00A2D594	
2D585 . A1 BC2EEAE0 MOU EAX, DWORD PTR DS:[AE2EBC]	
2D58A . BA F0DFA200 MOU EDX,wvs.00A2DFF0	
2D58F . E8 4C849DFF CALL wvs.004059E0	
> 8D45 F0 LEA EAX, DWORD PTR SS:[EBP-10]	
2D597 . E8 B4FBFFFF CALL wvs.00A2D150	
2D59C . 8B55 F0 MOU EDX, DWORD PTR SS:[EBP-10]	
2D59F . A1 6833AE00 MOU EAX, DWORD PTR DS:[AE3368]	
2D5A4 . E8 37849DFF CALL wvs.004059E0	
2D5A9 .~ E9 B4010000 JMP wvs.00A2D762	Case 0 of switch 00A2D49B
2D5AE > 8D55 EC LEA EDX, DWORD PTR SS:[EBP-14]	
2D5B1 . 8BC3 MOU EAX, EBX	
2D5B3 . E8 2097AAFF CALL wvs.004D6CD8	
2D5B8 . 8D45 EC LEA EAX, DWORD PTR SS:[EBP-14]	
2D5B9 . BA 1CE0A200 MOU EDX,wvs.00A2E01C	
2D5C0 . E8 9FB69DFF CALL wvs.00405C64	ASCII " <Enterprise Edition>"

Uhmm Bingo , llegamos a una zona muy importante aqui , aqui vuelve hacer un chequeo al mismo buffer pero ahora lo hace al buffer+10 en esa direccion , entonces si analizamos la rutina , como veremos hay tenemos la version "consultant edition" si vemos cual de esos saltos condicionales es el que lleva a esa direccion nos encontramos con este salto.

1A0 .~ 7F 17	JG SHORT wvs.00A2D4D7	
1A0 .~ 0F84 86010000	JE wvs.00A2D62C	
1A6 . 83E8 01	SUB EAX,1	
1A9 .~ 0F82 FF000000	JB wvs.00A2D5AE	
<b>1AF .~ 74 7F</b>	<b>JE SHORT wvs.00A2D530</b>	
1B1 . 48	DEC EAX	
1B2 .~ 74 23	JE SHORT wvs.00A2D4D7	
1B4 .~ E9 A9020000	JMP wvs.00A2D762	
1B9 > 83E8 04	SUB EAX,4	
1BC .~ 0F84 C3010000	JE wvs.00A2D685	
1C2 . 48	DEC EAX	

Este es el salto indicado , entonces para lograr hacer que tome ese salto , vemos mas arriba lo siguiente:

- (1) compara el registro EAX con 3 y de ser igual o mayor no manda a otro lugar , entonces debe ser menos a 3 el contenido del buffer para poder acercarnos a dicho salto.
- (2) Resta el contenido de eax con 1 y despues verifica con un salto JB ese salto es tomado si la bandera C (carry) esta activa.

Por lo tanto , no puede ser mayor a tres y tampoco puede valer 2 , por lo tanto tiene que vale a 1

Entonces le damos el valor a esa direccion de memoria , y hacemos un recuento de los valores que hemos cambiado que son:

Buffer+4=1

Buffer+48=0

Buffer+10=1

Uff que larga tarea y mas cuando tienes que escribir y al mismo tiempo analizar el soft , seguimos la ejecucion y paramos en ...

565	.	BA C4DFA200	MOU EDX,wss.00A2DFC4	ASCII "Consultant Edition"
568	.	E8 71849DF	CALL wss.00A2DFC4	
56F	.	A1 303EAE00	MOU EAX, DWORD PTR DS:[AE3E38]	
574	.	8B00	MOU EAX, DWORD PTR DS:[EAX]	
576	.	8B40 14	MOU EAX, DWORD PTR DS:[EAX+14]	
579	.	BA E0DFA200	MOU EDX,wss.00A2DFE0	
57E	.	E8 25889DF	CALL wss.00A2DFE0	
583	.	75 0F	JNZ SHORT wss.00A2D594	ASCII "WUSC10"
585	.	A1 BC2EAE00	MOU EAX, DWORD PTR DS:[AE2EBC]	
588	.	BA F0DFA200	MOU EDX,wss.00A2DFF0	ASCII "Consultant Edition <10 instances"
58F	.	E8 4C849DF	CALL wss.00A2DFF0	
594	>	8D45 F0	LEA EAX,DWORD PTR SS:[EBP-10]	
597	.	E8 B4FBFFF	CALL wss.00A2D150	
59C	.	8B55 F0	MOU EDX,DWORD PTR SS:[EBP-10]	
59F	.	A1 6833AE00	MOU EAX,DWORD PTR DS:[AE3368]	
5A4	.	E8 37849DF	CALL wss.00A2D594	
5A9	.	75 0F	JMP wss.00A2D762	Case 0 of switch 00A2D49B
5AE	>	8D55 EC	LEA EDX,DWORD PTR SS:[EBP-14]	
5B1	.	8BC3	MOU EAX,EBX	
5B3	.	E8 2097AAFF	CALL wss.00A2D6CD8	
5B8	.	8D45 EC	LEA EAX,DWORD PTR SS:[EBP-14]	
5BB	.	BA 1CE0A200	MOU EDX,wss.00A2E01C	ASCII "<Enterprise Edition>"
5C0	.	E8 9FB69DF	CALL wss.00A2E01C	
5C5	.	8B55 EC	MOU EDX,DWORD PTR SS:[EBP-14]	
5C8	.	8BC3	MOU EAX,EBX	
5CA	.	E8 3997AAFF	CALL wss.00A2E03C	
5CF	.	A1 C02EAE00	MOU EAX,DWORD PTR DS:[AE2EC0]	
5D4	.	BA B0DFA200	MOU EDX,wss.00A2DFB0	ASCII "Licensed"
5D9	.	E8 02849DF	CALL wss.00A2DFB0	
5DE	.	A1 BC2EAE00	MOU EAX,DWORD PTR DS:[AE2EBC]	
5E3	.	BA 3CE0A200	MOU EDX,wss.00A2E03C	ASCII "Enterprise Edition <2 instances"
5E8	.	E8 F3839DF	CALL wss.00A2E068	
5ED	.	A1 383EAE00	MOU EAX,DWORD PTR DS:[AE3E38]	
5F2	.	8B00	MOU EAX,DWORD PTR DS:[EAX]	
5F4	.	8B40 14	MOU EAX,DWORD PTR DS:[EAX+14]	
5F7	.	BA 68E0A200	MOU EDX,wss.00A2E068	ASCII "WUSE10"
5F8	.	CC 00000000	CALL wss.00A2E068	

Aqui vemos que compara el buffer+14 y mas abajo entra en un call y saliendo hace unas comparaciones para saber que version va a correr si la que corre únicamente dos instancias o la que corre las 10 instancias que creo esa es la mas completa por ahora ignoramos esta parte y seguimos la ejecucion , ya que esto lo resolveremos al final , recuerdan al principio ? Paramos en un primer breakpoint donde chequeaba las intancias , entonces no nos preocupemos ahora por eso , seguimos la ejecucion y vemos que mas encontramos.

```

0D172: . A1 383EAE00 MOU EAX,DWORD PTR DS:[AE3E38]
0D177: . 8B00 MOU EAX,DWORD PTR DS:[EAX]
0D179: . 8078 58 00 CMP BYTE PTR DS:[EAX+58],0
0D17D: . 74 41 JE SHORT vws.00A2D1C0
0D17E: . A1 383EAE00 MOU EAX,DWORD PTR DS:[AE3E38]
0D184: . 8B00 MOU EAX,DWORD PTR DS:[EAX]
0D186: . 8078 59 00 CMP BYTE PTR DS:[EAX+59],0
0D188: . 75 26 JNZ SHORT vws.00A2D1B2
0D18C: . A1 383EAE00 MOU EAX,DWORD PTR DS:[AE3E38]
0D191: . 8B00 MOU EAX,DWORD PTR DS:[EAX]
0D193: . FF70 64 PUSH DWORD PTR DS:[EAX+64]
0D196: . FF70 60 PUSH DWORD PTR DS:[EAX+60]
0D199: . 8D45 FC LEA EAX,DWORD PTR SS:[EBP-4]
0D19C: . E8 FF469EFF CALL vws.004118A0
0D1A1: . 8B4D FC MOU ECX,DWORD PTR SS:[EBP-4]
0D1A4: . 8BC3 MOU EAX,EBX
0D1A6: . BA 04D2A200 MOU EDX,vws.00A2D204
0D1AB: . E8 F88A9DFF CALL vws.00405CAB
0D1B0: . EB 1A JMP SHORT vws.00A2D1CC
0D1B2: > 8BC3 MOU EAX,EBX
0D1B4: . BA 38D2A200 MOU EDX,vws.00A2D238
0D1B9: . E8 22889DFF CALL vws.004059E0
0D1BE: . EB 0C JMP SHORT vws.00A2D1CC
0D1C0: > 8BC3 MOU EAX,EBX
0D1C2: . BA F8D1A200 MOU EDX,vws.00A2D1F8
0D1C7: . E8 14889DFF CALL vws.004059E0
0D1CC: > 33C0 XOR EAX,EAX
0D1CE: . 5A POP EDX
0D1CF: . 59 POP ECX
0D1D0: . 59 POP ECX
0D1D1: . 64:8910 MOU DWORD PTR FS:[EAX],EDX
0D1D4: . 68 E9D1A200 PUSH vws.00A2D1E9
0D1D9: > 8D45 FC LEA EAX,DWORD PTR SS:[EBP-4]

```

Uff , hace otro chequeo en el buffer+58 y lo compara con 0 , al parecer chequea si ya caduco el periodo de mantenimiento , en este caso el buffer vale 0 , pero igual hay que tomarlo en cuenta , seguimos

D76C	. A1 383EAE00	MOU EAX,DWORD PTR DS:[AE3E38]	
D771	. 8B00	MOU EAX,DWORD PTR DS:[EAX]	
D773	. 8078 49 00	CMP BYTE PTR DS:[EAX+49],0	
D777	.> 74 11	JE SHORT wvs.00A2D78A	
D779	. A1 A03DAE00	MOU EAX,DWORD PTR DS:[AE3DA0]	
D77E	. BA 8CE1A200	MOU EDX,wvs.00A2E18C	
D783	. EB 58829DFF	CALL wvs.004059E0	
D788	. EB 76	JMP SHORT wvs.00A2D800	
D78A	> A1 383EAE00	MOU EAX,DWORD PTR DS:[AE3E38]	
D78F	. 8B00	MOU EAX,DWORD PTR DS:[EAX]	
D791	. 8078 48 00	CMP BYTE PTR DS:[EAX+48],0	
D795	.> 74 69	JE SHORT wvs.00A2D800	
D797	. A1 383EAE00	MOU EAX,DWORD PTR DS:[AE3E38]	
D79C	. 8B00	MOU EAX,DWORD PTR DS:[EAX]	
D79E	. 8378 3C 00	CMP DWORD PTR DS:[EAX+3C],0	
D7A2	.> 7C 4D	JL SHORT wvs.00A2D7F1	
D7A4	. 68 B4E1A200	PUSH wvs.00A2E1B4	
D7A9	. A1 383EAE00	MOU EAX,DWORD PTR DS:[AE3E38]	
D7AE	. 8B00	MOU EAX,DWORD PTR DS:[EAX]	
D7B0	. FF70 54	PUSH DWORD PTR DS:[EAX+54]	
D7B3	. FF70 50	PUSH DWORD PTR DS:[EAX+50]	
D7B6	. 8D45 C8	LEA EAX,DWORD PTR SS:[EBP-38]	
D7B9	. E8 E2409EFF	CALL wvs.004118A0	
D7BE	. FF75 C8	PUSH DWORD PTR SS:[EBP-38]	
D7C1	. 68 DCE1A200	PUSH wvs.00A2E1DC	
D7C6	. 8D55 C4	LEA EDX,DWORD PTR SS:[EBP-3C]	
D7C9	. A1 383EAE00	MOU EAX,DWORD PTR DS:[AE3E38]	
D7CE	. 8B00	MOU EAX,DWORD PTR DS:[EAX]	
D7D0	. 8B40 3C	MOU EAX,DWORD PTR DS:[EAX+3C]	
D7D3	. E8 F4089EFF	CALL wvs.0040E0CC	
D7D8	. FF75 C4	PUSH DWORD PTR SS:[EBP-3C]	
D7DB	. 68 E8E1A200	PUSH wvs.00A2E1B8	
D7E0	. A1 A03DAE00	MOU EAX,DWORD PTR DS:[AE3DA0]	
D7E5	. BA 95000000	MOU EDX,5	
D7EA	. E8 2D859DFF	CALL wvs.00405D1C	
D7EF	.> EB 0F	JMP SHORT wvs.00A2D800	
D7F1	> A1 A03DAE00	MOU EAX,DWORD PTR DS:[AE3DA0]	
D7F6	. BA FCE1A200	MOU EDX,wvs.00A2E1FC	

Bien hizo otra parada , uff son muchisimos chequeos , aqui esta verificando que la direccion de memoria buffer+49 sea diferente a 0 para no lanzarnos el mensaje que ya caduco nuestra licencia y un poco mas abajo hace lo mismo en el buffer+48 y verifica de lo contrario el contenido sea 0 ó de lo contrario nos lanzaria un mensaje que esta caducando nuestra licencia , pero por el momento el contenido del buffer esta a nuestro favor , pero tambien debemos tenerlo en cuenta . Seguimos la ejecucion haber si hace alguno otro de sus 500 chequeos XD jaja , seguimos...

despues de unas cuantas paradas sin hacer chequeos importante llegamos hasta aqui , donde al parecer hace otro chequeo haber cuantos escaneos paralelos puede hacer

```

F | . 64:0720 MOU DWORD PTR FS:[EBX+1],ESP
2 | . 8BD3 MOU EDX,EBX
4 | . 8BC3 MOU EAX,EBX
6 | . E8 E9FFFF CALL wws.009BB414
B | . 68 C4B59B00 PUSH wws.009BB5C4
0 | . A1 383EAE00 MOU EAX,DWORD PTR DS:[AE3E38]
5 | . 8B00 MOU EAX,DWORD PTR DS:[EAX]
7 | . E8 7CEACEFF CALL wws.006A9FB8
C | . 8D55 F8 LEA EDX,DWORD PTR SS:[EBP-8]
F | . E8 882BA5FF CALL wws.0040E0CC
4 | . FF75 F8 PUSH DWORD PTR SS:[EBP-8]
7 | . 68 E4B59B00 PUSH wws.009BB5E4
C | . 8D45 FC LEA EAX,DWORD PTR SS:[EBP-4]
F | . BA 03000000 MOU EDX,3
4 | . E8 C3A7A4FF CALL wws.00405D1C
0 | . 00E8 EC MOU EDX,DWORD PTR SS:[EBP-4]

```

ASCII "Parallel scans (max "

Entramos en la call con direccion 0x9bb537 y vemos dentro de la llamada , esto:

```

B | $ 53 PUSH EBX
9 | . 8BD8 MOU EBX,EAX
B | . BA F49F6A00 MOU EDX,wws.006A9FF4
0 | . 8B43 14 MOU EAX,DWORD PTR DS:[EBX+14]
3 | . E8 DC38D6FF CALL wws.0040D8A4
8 | . 84C0 TEST AL,AL
A | . 75 11 JNZ SHORT wws.006A9FDD
C | . BA 04A06A00 MOU EDX,wws.006AA004
1 | . 8B43 14 MOU EAX,DWORD PTR DS:[EBX+14]
4 | . E8 CB38D6FF CALL wws.0040D8A4
9 | . 84C0 TEST AL,AL
B | . 74 07 JE SHORT wws.006A9FE4
D | > B8 0A000000 MOU EAX,0A
2 | . 5B POP EBX
3 | . C3 RETN
4 | > B8 02000000 MOU EAX,2
9 | . 5B POP EBX
A | . C3 RETN
B | . 00 DB 00
C | . FFFFFFFF DD FFFFFFFF
D | . 0600000000 DD 0000000006

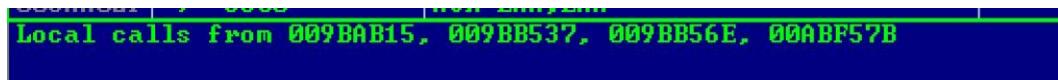
```

ASCII "WUSE10"

ASCII "WUSC10"

hace un chequeo en el buffer+14 y entra a una call donde hara todo un procedimiento que la verdad no tengo ni ganas de mirar , pero veo que esos saltos condicionales te redireccionan a diferentes retornos uno regresa dandole a eax= 2 y el otro valiendo eax=0A (10 en decimal) ,y tambien doy cuenta que esta rutina es llamada de otros lugares,

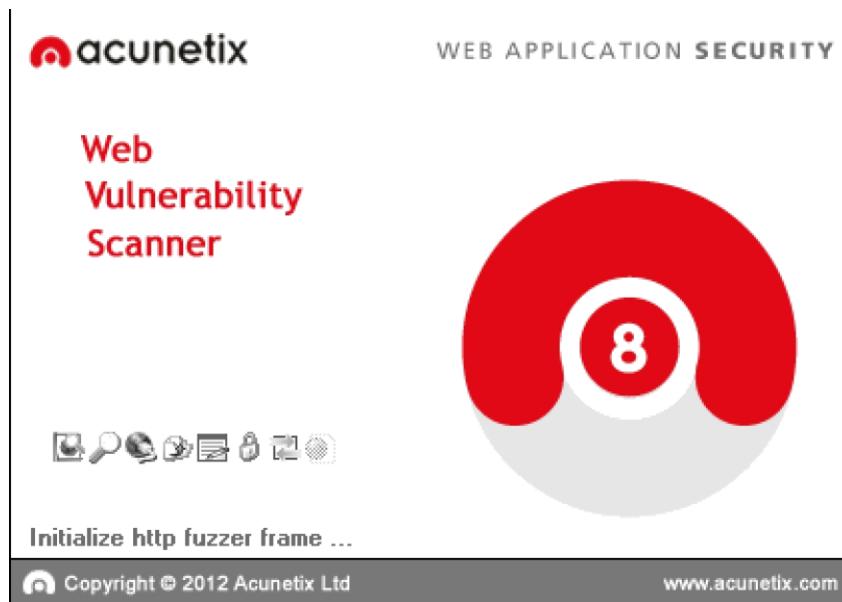
entonces si modifco algo, surtira efecto en varios lugares donde quiera entrar a la rutina.



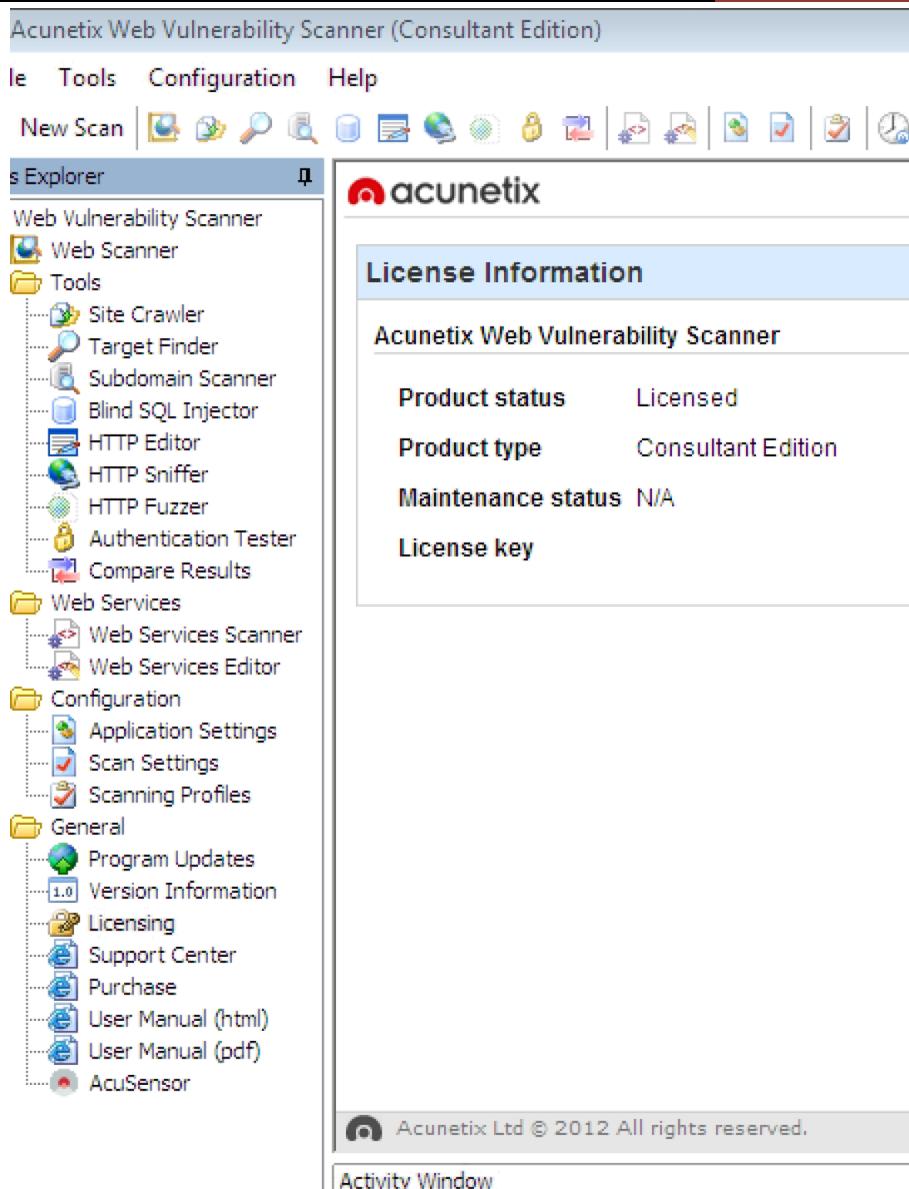
Asi que lo que hago es nopear esos saltos condicionales de tal forma que pase por el retorno que devuelve a eax=A que es muy logico es para la version mas completa que permitia analizar 10 sitios webs al mismo tiempo , entonces hago los cambios y quedaria algo como esto

F49F6A00	MOU EDX,wvs.006A9FF4
I3 14	MOU EAX,DWORD PTR DS:[EBX+
DC38D6FF	CALL wvs.0040D8A4
C0	TEST AL,AL
	NOP
	NOP
04A06A00	MOU EDX,wvs.006AA004
I3 14	MOU EAX,DWORD PTR DS:[EBX+
CB38D6FF	CALL wvs.0040D8A4
C0	TEST AL,AL
	NOP
	NOP
0A000000	MOU EAX,0A
	POP EBX
02000000	RETN
	MOU EAX,2
	POP EBX
	RETN

ahora sigo la ejecucion y vemos que otras sorpresas tiene este soft , seguimos...



El soft sigue corriendo y vuelve a parar la ejecución en un BP donde hace un chequeo normal en el buffer+4 y en el buffer+10 que eran los que contenían datos donde decidía si estaba registrado y que tipo de licencia contenía , pero recuerden que teníamos ya datos en el buffer para seguir el funcionamiento como si tuviéramos la versión “consultant edition” , por lo tanto seguimos ejecutando...



Por fin...

Se ejecuto el software correctamente y en versión "consultant edition" , sin problema alguno ,ya tenemos los datos con lo que vamos a realizar un parcheo eficiente

Bien recordemos lo valores que deben contener las direcciones del buffer:

Buffer+4=1

Buffer+48=0

Buffer+10=1

Buffer+58=0

Buffer+49=1

Ahora debemos meter esos valores al buffer y se me ocurre una idea , veamos si funciona , recuerdan el primer breakpoint? Donde detectaba si estabas ejecutando las instancias de lo que te permitia cierta licencia , vayamos hay

```
MOV EAX,DWORD PTR DS:[AE3E38]
MOV EAX,DWORD PTR DS:[EAX]
CALL wvs.006A9FB8
CALL wvs.00ABBE34
TEST AL,AL
JNZ SHORT wvs.00ABF5C4
MOV EAX,DWORD PTR DS:[AE2A9C]
CMP BYTE PTR DS:[EAX],0
JE SHORT wvs.00ABF59F
MOV EAX,wvs.00AC08AC
CALL wvs.006290B4
JMP SHORT wvs.00ABP5B4
PUSH B
MOV CX,WORD PTR DS:[AC08E4]
MOV DL,1
MOV EAX,wvs.00AC08AC
CALL wvs.00492BC8
MOV EAX,DWORD PTR DS:[AE2344]
MOV DWORD PTR DS:[EAX],309
JMP wvs.00AC06B1
XOR EAX,EAX
PUSH EBP
```

ASCII "This version does not permits running more instances.  
wvs.00492BC8

Arg1 = 00000000

ASCII "This version does not permits running more instances.  
wvs.00AC06B1

En este lugar hay un espacio perfecto para hacer un parche y mas porque es el primer chequeo que hace en el buffer entonces es perfecto , como vemos el programa jamás deberá pasar donde están esas strings que nos hacen sacar

un mensaje malo , por lo tanto vamos a nopear ese pedacito de codigo y escribimos nuestro parche que mete los valores correctos a las direcciones de memoria para poder bypassar todo correctamente como si tuviéramos una licencia completísima :D , el parche será el siguiente veamos

0ABF57B	.	E8 38AHBEFF	CALL wws.006A9FB8
0ABF580	.	E8 AFC8FFFF	CALL wws.00ABBE34
0ABF585	50		PUSH EAX
0ABF586	A1 383EAEO0		MOU EAX,DWORD PTR DS:[IAE3E38]
0ABF58B	8B00		MOU EAX,DWORD PTR DS:[EAX]
0ABF58D	C640 04 01		MOU BYTE PTR DS:[EAX+4],1
0ABF591	C640 10 01		MOU BYTE PTR DS:[EAX+10],1
0ABF595	C640 48 00		MOU BYTE PTR DS:[EAX+48],0
0ABF599	C640 49 00		MOU BYTE PTR DS:[EAX+49],0
0ABF59D	C640 58 00		MOU BYTE PTR DS:[EAX+58],0
0ABF5A1	58		POP EAX
0ABF5A2	EB 20		JMP SHORT wws.00ABP5C4
0ABF5A4	90		NOP
0ABF5A5	90		NOP
0ABF5A6	90		NOP
0ABF5A7	90		NOP
0ABF5A8	90		NOP
0ABF5A9	90		NOP
0ABF5AA	90		NOP
0ABF5AB	90		NOP
0ABF5AC	90		NOP
0ABF5AD	90		NOP
0ABF5AE	90		NOP
0ABF5AF	90		NOP

Lo que hacemos primero es hacer un push a eax para guardar el contenido en el stack (la pila) y no dañar nada , despues metemos los valores , 1,1,0,0,0 a las direcciones que comentabamos haya arriba que eran:

Buffer+4=1

Buffer+48=0

Buffer+10=1

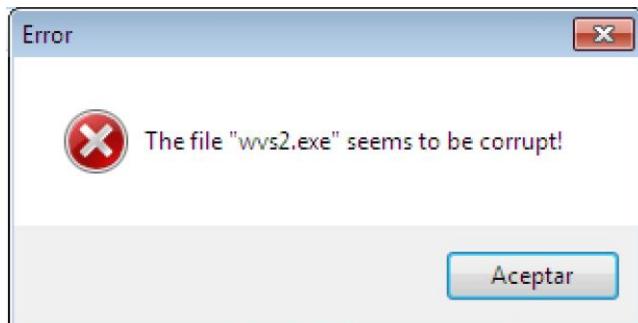
Buffer+58=0

Buffer+49=1

ahora bien , regresamos al registro eax el contenido que tenia y hacemos un salto directo a donde deberia seguir ejecutandose el programa y listo .. :) , click derecho “copy tu executable – all modifications”



le damos en copy all , le ponemos un nombre al archivo y LISTO YA DEBERIA DE FUNCIONA NUESTRO PARCHE :D :D :D  
EJECUTAMOS ...



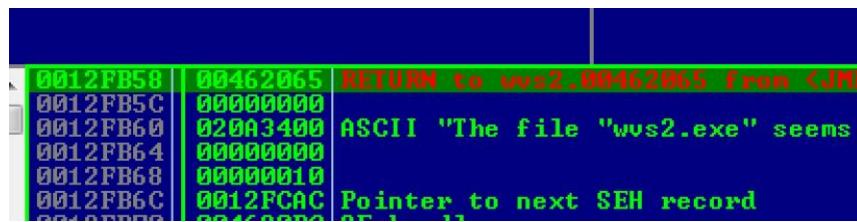
NOO!! , al parecer el acunetix se da cuenta que ha sido parcheado , pero no nos rendiremos , abrimos el olly en el parcheado y hacemos que corra hasta aparecer este MessageBox y presionamos "PAUSE" (f12) , nos vamos al call stack y vemos esto

```

2E78C user32.7742E38F           user32.774
2E836 user32.MessageBoxTimeoutW user32.774
2E9E4 ? user32.MessageBoxTimeoutA user32.774
2EA56 ? user32.MessageBoxExA    user32.774
20000 hOwner = NULL
33400 Text = "The file \"wvs2.exe\" seems to be corrupt"
20000 Title = NULL
20010 Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
20000 LanguageID = 0 (LANG_NEUTRAL)
52065 ? <JMP.&user32.MessageBoxA> wvs2.00462
20000 hOwner = NULL
33400 Text = "The file \"wvs2.exe\" seems to be corrupt"
20000 Title = NULL
20010 Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
52560 ? wvs2.00461F2C          wvs2.00462
5390E ? wvs2.0046249C          wvs2.00463
25603 Includes wvs2.004639DE   wvs2.00465
25668 wvs2.004055BC          wvs2.0040E
282E3 wvs2.0040562C          wvs2.0040E
3F10C wvs2.004082A4          wvs2.00ABF
:

```

Vemos dos referencias a MessageBox le damos click derecho a la ultima de esas dos ( la que tengo seleccionada en la imagen) y le damos click derecho y "folow address in stack" y veremos la direccion en el stack



si le damos click derecho a esa direccion "FOLLOW IN DISASSEMBLER" llegamos a esta parte

```

F  PUSH EAX
F  PUSH 0
CALL <JMP.&user32.MessageBoxA>
F  PUSH 0
CALL <JMP.&kernel32.GetModuleHandleA>
E00  CMP EAX,DWORD PTR DS:[AE4668]
JE SHORT wws2.00462094
0  MOV EAX,wws2.000AC53F0
F  CALL wws2.00407ADC
MOV ECX,DWORD PTR SS:[EBP-14]
MOV DL,1
0  MOV EAX,DWORD PTR DS:[419E4C]
F  CALL wws2.00419EB8
F  CALL wws2.00405300
JMP SHORT wws2.0046209B
PUSH 0
CALL <JMP.&kernel32.ExitProcess>
xor EAX EAX

```

Hay esta la zona encargada de todo si subimos un poco mas vemos la api CreateFileA , esa API normalmente se usa en este tipo de protecciones para leer el archivo existente y devolver el tamaño del mismo , por lo tanto le damos un Hardware Brakpoint On execution , reiniciamos y corremos el debugger y vemos cuando pare ahí

```

6H 03  PUSH 3
68 00000080  PUSH 80000000
8D85 E7FEFFFF  LEA EAX,DWORD PTR SS:[EBP-119]
50  PUSH EAX
E8 B469FAFF  CALL <JMP.&kernel32.CreateFileA>
8BF0  MOU ESI,EAX
83FE FF  CMP ESI,-1
0F84 AC000000  JE wws2.00462037
6A 00  PUSH 0
6A 00  PUSH 0
6A 00  PUSH 0
6A 02  PUSH 2
6A 00  PUSH 0

```

vemos unas 2 instrucciones mas abajo que hay un salto condicional , debemos tomarlo para no pasar por toda el código de abajo que lo que hace es crear un map del ejecutable en memoria toma el tamaño del archivo y mucha tonteria mas, pues digo tonteria porque con ese salto me burlo todo eso , asi que pongamos un JMP que eso hara que

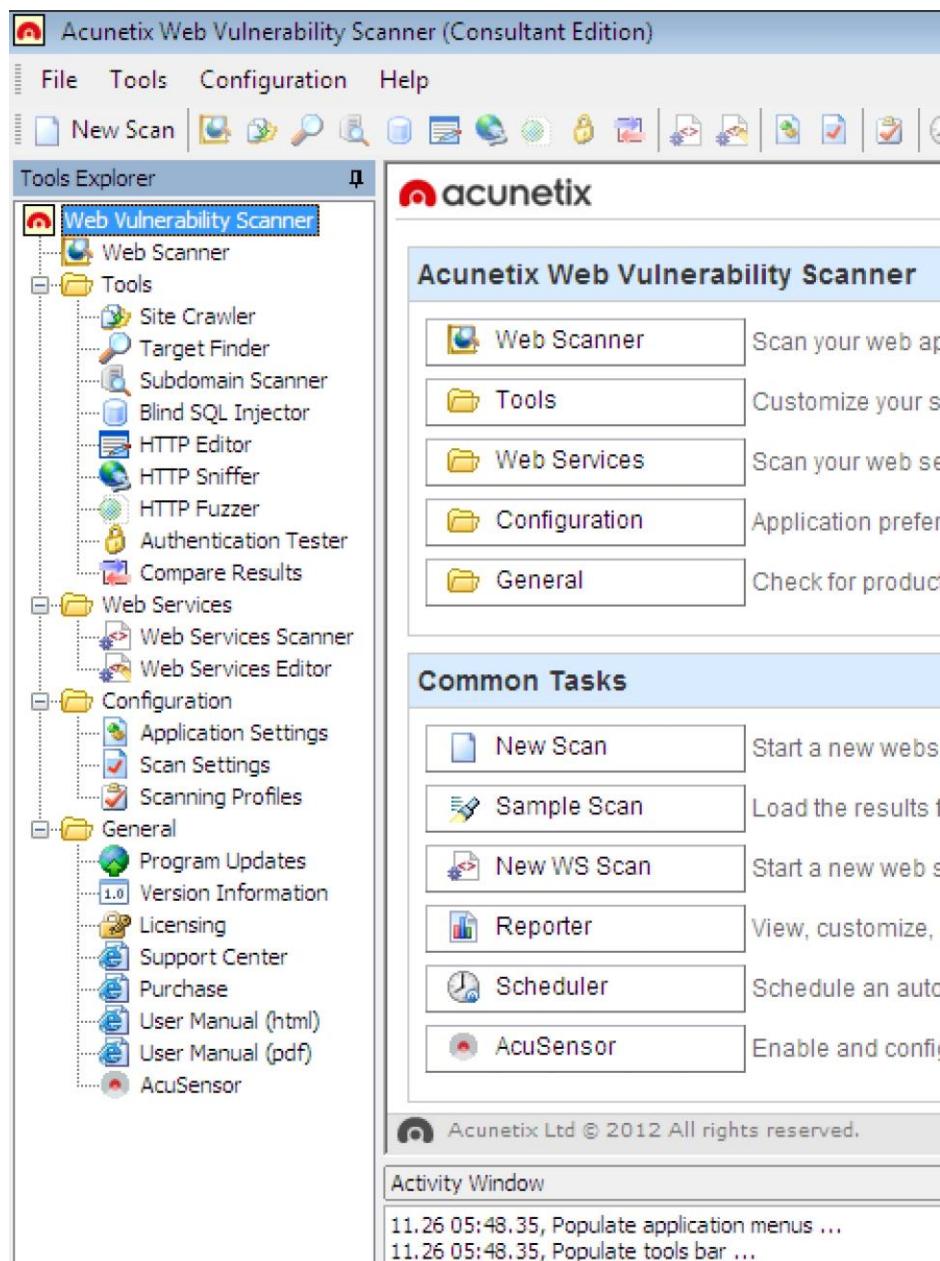
salte siempre pase lo que pase siempre que pase por hay tiene que tomar el salto

80	8BF0	MOU ESI,EAX
82	83FE FF	CMP ESI,-1
85	E9 AD880000	JMP <a href="#">ws2.00462037</a>
8A	90	NOP
8B	6A 00	PUSH 0
8D	6A 00	PUSH 0
8F	6A 00	PUSH 0
91	6A 02	PUSH 2
93	6A 00	PUSH 0
9C	56	PUSH ECX

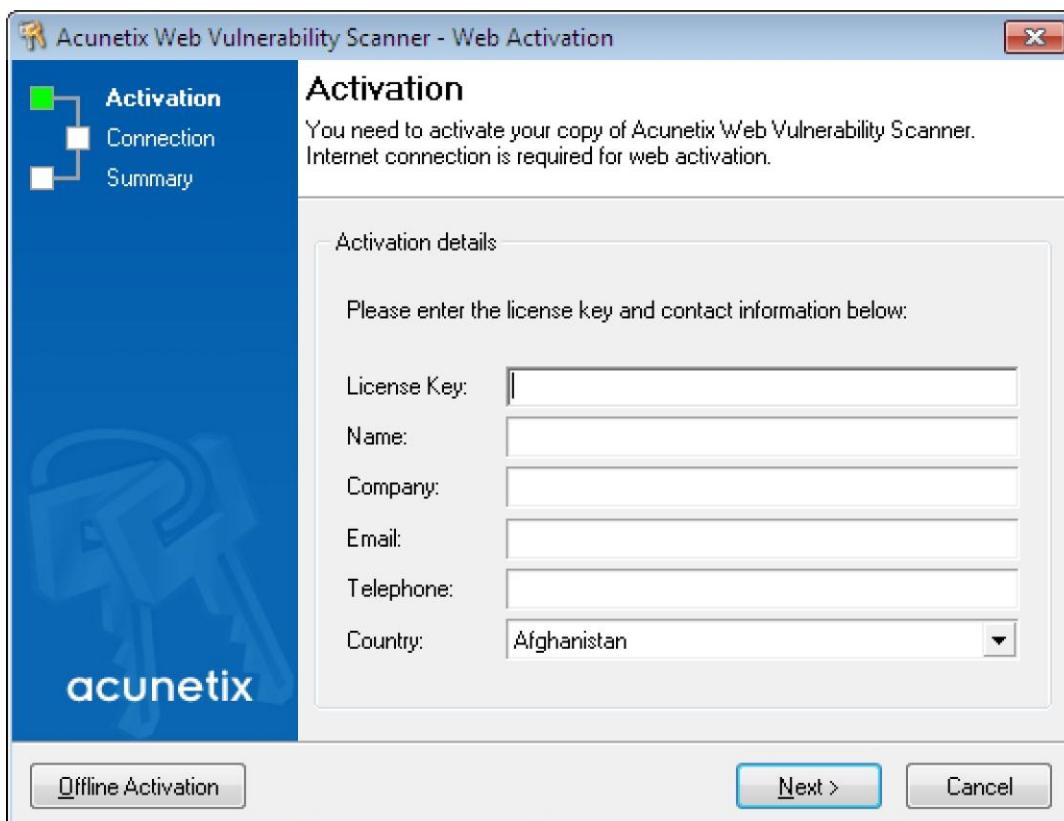
seguimos y nos encontramos con otro salto que igual le pondremos un JMP para que igual siempre tome ese salto y nos saque de la rutina verificadora

202C	E8 40000000	CALL <JMP.&kernel32.CloseHandle>
2031	56	PUSH ESI
2032	E8 A568FAFF	CALL <JMP.&kernel32.CloseHandle>
2037	84DB	TEST BL,BL
2039	EB 60	JMP SHORT <a href="#">ws2.0046209B</a>
203B	6A 10	PUSH 10
203D	6A 00	PUSH 0
203F	8D95 E0FEFFFF	LEA EDX,DWORD PTR SS:[EBP-120]
2045	A1 F053AC00	MOV EAX,DWORD PTR DS:[AC53F0]
204A	8B08	MOV ECX,DWORD PTR DS:[EAX]
204C	FF91 48020000	CALL DWORD PTR DS:[ECX+248]
2052	8B85 E0FEFFFF	MOV EAX,DWORD PTR SS:[EBP-120]
2058	E8 FF3DFAFF	CALL <a href="#">ws2.00405E5C</a>

y listo ya con eso es todo , ahora si volvamos a guardar los cambios , listo ahora si probemos , cerramos todo y vemos haber como funciona el parche..



Ahora bien hay que probar sus opciones haber si tiene algun otro chequeo , vamos a crear un reporte y cuando presionamos , veemos esto ...



Uhmm Al parecer tiene otro chequeo y no es con el mismo buffer pues no paro en ningun Breakpoint, veamos que volvio a correr el "activation.exe" , pero recuerden que en la carpeta donde se encuentra el software hay unos ejecutables activation y reporter , el "reporter.exe" es el que debemos enfocarnos ahora , vamos a debuggear ese ejecutable , lo cargamos y buscamos la string "activation.exe" veamos....

```
rter.008E65H0      ASCII "wlet_data.dat"
rter.008E65C4      ASCII "File is missing or corrupt (error #1).Please
rter.008E6928      ASCII "Invalid password!"
rter.008E7E38      ASCII "Write"
rter.008E7E48      ASCII "Data\Graphics\Reporter\acu logo.png"
rter.008E7E74      ASCII "Data\Graphics\Reporter\acuwash.png"
rter.008E7EA0      ASCII "reporter_?????????????.csv"
rter.008E7EC8      ASCII "csv"
rter.008E7ED4      ASCII "reporter_"
porter.008E7AF5    (Initial CPU selection)
rter.008E7EE8      ASCII "Application must be activated!"
r.008E7F0C          ASCII "Activation.exe"
r.008E7F1C          ASCII "open"
rter.008E7F2C      ASCII "Application has been expired!"
rter.008E7F2C      ASCII "Application has been expired!"
rter.008E7F58      ASCII "Invalid password!"
rter.008E7F74      ASCII "WWS Reporter"
```

Bien encontramos eso , si vemos mas arriba pareciera que habla de las imagenes o logos del reporte y una de los beneficios que se obtienen al ser una versión “consultant edition” es que podemos modificar ese logo sin problemas , eso quiere decir que estamos cerca de eso , vayamos a esa dirección donde esta el “activation.exe”

```

CALL Reporter.008E11E0
MOU EAX,DWORD PTR DS:[8FAEA4]
MOU EAX,DWORD PTR DS:[EAX]
CMP BYTE PTR DS:[EAX+4],0
JNZ SHORT Reporter.008E7AF5
MOU EAX,DWORD PTR DS:[8FA038]
MOU EAX,DWORD PTR DS:[EAX]
CMP BYTE PTR DS:[EAX+30],0
JE SHORT Reporter.008E7AB5
MOU EAX,DWORD PTR DS:[8FA038]
MOU EAX,DWORD PTR DS:[EAX]
MOU EDX,Reporter.008E7EE8
CALL Reporter.008E2234
JMP Reporter.008E7DFB
PUSH 1
PUSH Reporter.008E7F08
LEA EDX,DWORD PTR SS:[EBP-1C]
MOU EAX,DWORD PTR DS:[8FA9B8]
MOU EAX,DWORD PTR DS:[EAX]
CALL Reporter.005035E8
MOU EAX,DWORD PTR SS:[EBP-1C]
LEA EDX,DWORD PTR SS:[EBP-18]
CALL Reporter.004594A0
MOU EAX,DWORD PTR SS:[EBP-18]
CALL Reporter.004059D8
PUSH EAX
PUSH Reporter.008E7F0C
PUSH Reporter.008E7F1C
PUSH 0
CALL <JMP.&shell32.ShellExecuteA>
JMP Reporter.008E7DFB
MOU EAX,DWORD PTR DS:[8FAEA4]

```

ASCII "Application must be .

ASCII "Activation.exe"

ASCII "open"

Veamos que es casi idéntica la rutina al otro ejecutable , en el offset 0x8E7A85 , intenta sacar el contenido de una dirección de memoria y de sacar sacar un buffer+4 y chequear que sea distinto a 0 o si no nos manda la ventanita de activación que ya nos tiene locos , entonces coloquemos el cambio del valor al contenido del buffer

Address	Hex dump
01F98BBC	01 00 00 00
01F98BC4	00 00 00 00
01F98BC6	00 00 00 00

Seguimos...

0	PUSH 0
06FB4FF	CALL <JMP.&she1132.ShellExecuteA>
6030000	JMP Reporter.008E7DFB
4AE8F00	MOU EAX,DWORD PTR DS:[8FAEA4]
48 00	MOU EAX,DWORD PTR DS:[EAX]
6	CMP BYTE PTR DS:[EAX+48],0
4AE8F00	JE SHORT Reporter.008E7B58
659C9FF	MOU EAX,DWORD PTR DS:[8FAEA4]
4AE8F00	MOU EAX,DWORD PTR DS:[EAX]
CALL Reporter.0057D4C4	CALL Reporter.0057D4C4
MOU EAX,DWORD PTR DS:[8FAEA4]	MOU EAX,DWORD PTR DS:[EAX]
3C 00	CMP DWORD PTR DS:[EAX+3C],0
D	JE SHORT Reporter.008E7B58

Volvio a parar y hace una comparacion al buffer+48 sea igual a cero de caso contrario nos mandara mensajes de expiracion de la licencia , como vemos es casi identica la rutina de chequeo , seguimos.

8BC6	MOU EAX,ESI
E8 56E0CDFF	CALL Reporter.004045D8
A1 A4AE8F00	MOU EAX,DWORD PTR DS:[8FAEA4]
8B00	MOU EAX,DWORD PTR DS:[EAX]
8B40 10	MOU EAX,DWORD PTR DS:[EAX+10]
83C0 FC	ADD EAX,-4
83E8 02	SUB EAX,2
0F83 1B010000	JNB Reporter.007266B3
8B43 30	MOU EAX,DWORD PTR DS:[EBX+30]
E8 10C2EDFF	CALL Reporter.006027B0
8BF0	MOU ESI,EAX
4E	DEC ESI

Aqui veamos hace un chequeo al buffer+10 y si recordamos en los chequeos del soft principal en el buffer+10 se encontraba el tipo de licencia y aqui tambien compara el tipo de licencia :) , entonces el contenido del buffer debe valer 1, puff ni se molestaron en hacernos mas dificil el trabajo , entonces ya sabemos los datos del parcheo:

buffer+4=1

buffer+48=0

buffer+10=1

bien los parches la forma de parchear se me ocurrio que en el primer chequeo hacer lo mismo que con el parche anterior recuerdan? meter los datos que debe ir en cada direccion de memoria jejeje :D veamos:

parche:

ORIGINAL:

The screenshot shows a debugger interface with two main panes. The left pane displays assembly code, and the right pane shows memory dump data.

**Assembly Code (Left Pane):**

```
CALL Reporter.008E11E0
MOU EAX,DWORD PTR DS:[8FAEA4]
MOU EAX,DWORD PTR DS:[EAX]
CMP BYTE PTR DS:[EAX+4],0
JNZ SHORT Reporter.008E7AF5
MOU EAX,DWORD PTR DS:[8FA038]
MOU EAX,DWORD PTR DS:[EAX]
CMP BYTE PTR DS:[EAX+30],0
JE SHORT Reporter.008E7AB5
MOU EAX,DWORD PTR DS:[8FA038]
MOU EAX,DWORD PTR DS:[EAX]
MOU EDX,Reporter.008E7EE8
CALL Reporter.008E2234
JMP Reporter.008E7DFB
PUSH 1
PUSH Reporter.008E7F08
LEA EDX,DWORD PTR SS:[EBP-1C]
MOU EAX,DWORD PTR DS:[8FA9B8]
MOU EAX,DWORD PTR DS:[EAX]
CALL Reporter.005035E8
MOU EAX,DWORD PTR SS:[EBP-1C]
LEA EDX,DWORD PTR SS:[EBP-18]
CALL Reporter.004594A0
MOU EAX,DWORD PTR SS:[EBP-18]
CALL Reporter.004059D8
PUSH EAX
PUSH Reporter.008E7F0C
PUSH Reporter.008E7F1C
PUSH 0
CALL <JMP.&shell32.ShellExecuteA>
JMP Reporter.008E7DFB
MOU EAX,DWORD PTR DS:[8FAEA4]
```

**Memory Dump (Right Pane):**

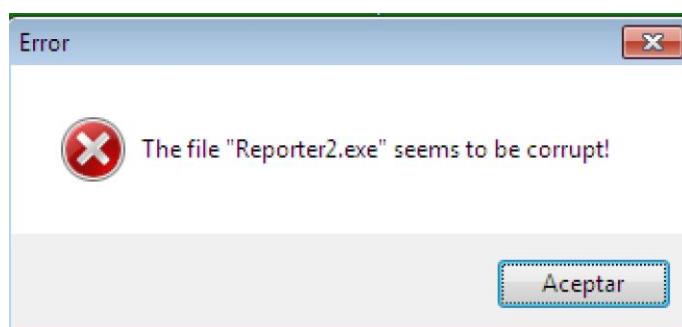
- Address 8FAEA4: ASCII "Application must be..."
- Address 8FA9B8: ASCII "Activation.exe"
- Address EBP-18: ASCII "open"

## PARCHEADO:

R1 R48E8F00	MOU BX,DWORD PTR DS:[8FAE84]
8B00	MOU EAX,DWORD PTR DS:[EAX]
C640 04 01	MOU BYTE PTR DS:[EAX+4],1
C640 48 00	MOU BYTE PTR DS:[EAX+48],0
C640 10 01	MOU BYTE PTR DS:[EAX+10],1
75 5B	JNZ SHORT Reporter.008E7AF5
78 30	JS SHORT Reporter.008E7ACC
007416 A1	ADD BYTE PTR DS:[ESI+EDX-5F],DH
38A0 8F008B00	CMP BYTE PTR DS:[EAX+8B008F],AH
BA E87E8E00	MOU EDX,Reporter.008E7EE8
E8 84A7FFFF	CALL Reporter.008E2234
E9 46030000	JMP Reporter.008E7DFB
6A 01	pushl 1

ASCI

Lo mismo que como lo hicimos anteriormente, que metiera los datos a las direcciones de memoria y retornara por donde tiene que seguir su camino , comparan del original al parcheado y veraz exactamente la modificacion, **OJO en ese ultimo salto debe ser un JMP no un JNZ como muestra la imagen** , pues de lo contrario no tomara el salto , ahora guardamos los cambios y cargamos otra vez con el olly hasta ver este mensaje:



Presionamos pause (f12) y ya sabemos lo que tenemos que hacer , vamos a esa direccion de memoria que nos muestra el call stack

```

00453AC0 user32.MessageBoxTimeoutW
7742E836 user32.MessageBoxTimeoutW
: 7742E9E4 ? user32.MessageBoxTimeoutA
: 7742EA56 ? user32.MessageBoxExA
00000000 hOwner = NULL
01FFA688 Text = "The file \"Reporter2.exe\""
: 00000000 Title = NULL
: 00000010 Style = MB_OK|MB_ICONHAND|MB_APPLI
: 00000000 LanguageID = 0 (LANG_NEUTRAL)
00453AC5 ? <JMP.&user32.MessageBoxA> Reporter.00453AC0
: 00000000 hOwner = NULL
01FFA688 Text = "The file \"Reporter2.exe\""
: 00000000 Title = NULL
: 00000010 Style = MB_OK|MB_ICONHAND|MB_APPLI
00453FC0 ? Reporter.0045398C Reporter.00453FBB
0045543E ? Reporter.00453E9C Reporter.00455439
0040517F Includes Reporter.0045543E Reporter.0040517C
: 004051E7 Reporter.0040513B Reporter.004051E2
: 00407F6B Reporter.004051A8 Reporter.00407F66
: 008E79AD Reporter.00407F2C Reporter.008E79A8

```

Ya que seguimos la direccion en el disassembler , subimos y vemos la misma rutina que verifica el tamaño y demas cosas para saber si ha sido modificado el ejecutable , hacemos los mismos cambios que hicimos anteriormente , veamos

```

00000000 PUSH 0000000000
FFFFFFFFFF LEA EAX,DWORD PTR SS:[EBP-119]
PUSH EAX
FBFF CALL <JMP.&kernel32.CreateFileA>
MOU ESI,EAX
CMP ESI,-1
0000 JMP Reporter.00453A97
NOP
PUSH R
84DB TEST BL,BL
EB 60 JMP SHORT Reporter.00453AFB
6A 10 PUSH 10
6A 00 PUSH 0
8D95 E0FFFFFF LEA EDX,DWORD PTR SS:[EBP-120]
A1 DCBB8E00 MOU EAX,DWORD PTR DS:[8EBBDC]
8B08 MOU ECX,DWORD PTR DS:[EAX]
FF91 48020000 CALL DWORD PTR DS:[ECX+248]

```

OK , ahora guardamos los cambios y probamos el ejecutable "reporter.exe"

The screenshot shows the Acunetix WVS Reporter application window. On the left, there's a navigation sidebar with options like 'Affected Items', 'Developer Report', 'Executive Summary', etc., and 'Report Preview' which is currently selected. The main area has three sections: 'Threat level' (Level 3: High), 'Alerts distribution' (with a bar chart showing 35 High, 38 Medium, 5 Low, and 9 Informational alerts), and 'Alerts summary' (listing 'Blind SQL Injection' under 'Affects').

**Threat level**

**acunetix threat level**  
Level 3: High

**Acunetix Threat Level 3**  
One or more high-severity type vulnerabilities can be exploited by a malicious user to compromise your website.

**Alerts distribution**

Severity	Count
High	35
Medium	38
Low	5
Informational	9

**Alerts summary**

Blind SQL Injection  
Affects

LISTO , PERFECTO!!!, Quedo completamente funcional, y puedo generar reportes sin problema y todo lo que yo quiera :) , ahora si porfin puedo utilizar el software completamente full , sin tener que bajar activadores de la red , que ni siquiera se si son seguros.

Este texto se alargo muchisimo debido a que me encontre muchas verificaciones que hacia el software , algunos trucos anti-cracking que usa y es muy complicado escribir y debuggear al mismo tiempo , pero bueno hemos aprendido que no fue tan dificil atacar este software y eso que se dedican a la "seguridad" , por lo menos no fue tan dificil como me imagine al principio , espero que hayan llegado hasta aqui , aun no he probado al 100% el funcionamiento del software , pero si alguien descubre algo que se me haya pasado de largo , le agradeceria me lo hiciera saber , aunque intente que no se me fuera algun byte que despues causara problemas , espero les haya servido de mucho este escrito pues me ha costado no dormir en esta noche lo cual ahora me pondre a escanear unos cuantos sitios y listo.

Agradecimientos: A todos los que alguna vez me apoyaron en seguir adelante y los que colaboraron conmigo en diferentes proyectos , familia ,amigos y a ti por leer este escrito.

**Alejandro Torres (torrescrack)**

**e-mail: [tora\\_248@hotmail.com](mailto:tora_248@hotmail.com)**

