Activación en Línea. Análisis y Emulación

Análisis del proceso de registro y activación del software Advanced SystemCare 13

Escrito por: ThunderCls
https://github.com/ThunderCls/

CONTENIDO

PREFACIO

Este escrito ha surgido gracias a la idea de un amigo que me propuso algo interesante, me propuso como reto hacer una revisión de mi primer software comercial jamás analizado por allá por el año 2008[1], justo cuando empezaba en el mundo de la ingeniería inversa. El fin era ver cuanto había cambiado la aplicación durante los años con nuevas versiones y de esta forma también hacer especie de un homenaje en forma de escrito a todos estos años de reversing junto a la lista CracksLatinoS. Al final la curiosidad y la nostalgia se unieron y aunque hace mucho tiempo desde la última vez que analice un software en general o incluso que estoy activo en estos temas, supongo que lo que bien se aprende nunca se olvida. Este escrito va dedicado a todos esos años de aprendizaje junto a esta comunidad mundial de ingeniería inversa.

Este escrito se suponía seria del primer software comercial que analice, el tema es que al descargar la última versión disponible en su web oficial y dar los primeros vistazos, quede muy defraudado ya que el sistema de registro no ha cambiado en nada, literalmente, por 11 años. Debido a lo anterior y siguiendo el orden de mis primeros tutoriales, el próximo seria uno que publique en los primeros días del año 2009[2] y de esta forma este fue el seleccionado.

Antes de empezar el escrito debo aclarar que este trabajo no supone ser un tutorial paso a paso de cómo obtener ilegalmente un software, no está incluso dirigido siquiera a principiantes o iniciados en la ingeniería inversa, más bien es simplemente mis apuntes del análisis del sistema de registro usado por esta aplicación a nivel de información general y en ningún momento promuevo practicas indeseables, si deseas el software cómpralo y apoya a los programadores, si por otra parte solo te interesa el conocimiento, supongo que eso es lo que encontraras en estas páginas.

---

[1] "ID USB Lock Key v1.2", http://ricardonarvaja.info/WEB/CURSO%20NUEVO/TEORIAS%20NUMERADAS/1001-1100/1088-ID_USB_Lock_Key_v1.2_Tute%2Bby%2BThunder.rar

[2] "Advanced WinCare Pro v2", http://ricardonarvaja.info/WEB/CURSO%20NUEVO/TEORIAS%20NUMERADAS/1001-1100/1090-Advanced_WinCare_Pro_v2_By%2BThunder.rar

"Todo individuo tiene derecho a la libertad de opinión y de expresión;
este derecho incluye el de no ser molestado a causa de sus opiniones,
el de investigar y recibir informaciones y opiniones, y el de difundirlas,
sin limitación de fronteras, por cualquier medio de expresión."

Artículo 19. "Declaración Universal de Derechos Humanos"

INTRODUCCIÓN

Los sistemas de activación en línea han sido y son muy utilizados por infinidad de aplicaciones hoy en día. El funcionamiento básico consistiría en tener un software instalado en la PC cliente que se encargara de hacer la validación de una licencia, la cual, una vez cumple los requisitos establecidos, es enviada a los servidores oficiales con el fin de saber si dicha licencia consta en la base de datos como ya pagada y por consiguiente proceder a dejar el software correctamente activado[3]. Este sería un funcionamiento bien simple y evidentemente existirán variaciones a dicho esquema dependiendo de cómo los programadores lo hayan querido implementar.

De aquí se pueden sacar varios factores que intervienen en el registro en línea de una aplicación determinada. El algoritmo de verificación de la licencia (que puede estar en el software cliente o directamente en el servidor oficial), peticiones a través de internet para la comunicación entre el software cliente y el servidor (puede ser utilizadas varias tecnologías para enviar y recibir mensajes, las más comunes son las peticiones HTTP) y por ultimo el proceso de activación llevado a cabo en la aplicación cliente al recibir la confirmación del servidor o simplemente el proceso donde la aplicación crea ficheros de registro o configuraciones para quedar registrada en futuras ejecuciones, casi siempre valiéndose tanto de ficheros en disco como en el registro de Windows.

Teniendo todo esto en cuenta y dependiendo de la implementación usada por los programadores, puede ser posible llegar a emular dicho proceso de activación sin modificar de ninguna forma el software cliente y sin usar los servidores oficiales de activación y de eso específicamente trata este escrito.

[3] "Online Software Activation", http://bastioninfotech.com/india/solution/software-protection/online-software-activation.jsp
"What is online license activation?", https://soraco.co/2014/05/14/online-license/

ALGORITMO DE REGISTRO

El software analizado será el Advanced SystemCare 13, siendo la última versión disponible en la web oficial en la fecha de realizado este escrito la versión 13.1.0.184. Una vez instalada e iniciada la aplicación se pueden ver que está en modo "*Free*" con las limitaciones establecidas

Imagen 1 - Aplicación en modo free

El proceso de registro no es llevado a cabo por el ejecutable principal, sino que al proceder a registrarse se hace uso de un segundo ejecutable llamado "*Register.exe*" que se encuentra en la misma carpeta que el ejecutable principal. La aplicación esta creada en Borland Delphi y no posee ningún tipo de compresor o protector así que esto facilita un poco el análisis.

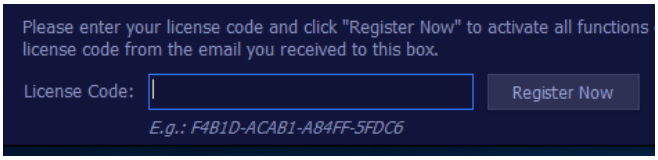

Imagen 2 - Ventana de registro

Mirando el código del botón se puede ver que todo el proceso de registro será documentado a lo largo del código en un fichero llamado "*register.log*"

```
mov     [esi+444h], eax
mov     edx, offset aRegisterLog_5 ; "register.log"
mov     eax, offset aActivateclickS ; "ActivateClick Start"
call    LogWrite
cmp     byte ptr [esi+464h], 0
```

Imagen 3 - Funcion LogWrite

Todo el algoritmo de registro consiste en la interacción entre el ejecutable "*Register.exe*" y la librería "*OFCommon.dll*", la cual es la encargada de comprobar finalmente que el código de licencia pase todas las verificaciones y pueda pasar a la segunda etapa de activación. En las primeras líneas del código del botón "*Register Now*" y justo antes de empezar a verificar el código introducido se hace una llamada a la función exportada "*GetTaskTypeEx*". En esta parte la función básicamente busca ficheros de licencia en disco y en el registro de Windows para comprobar si ha sido activado previamente el producto.

```
OFCommon.GetTaskTypeEx
0048EAD4        push        ebp
0048EAD5        mov         ebp,esp
0048EAD7        add         esp,0FFFFFFF8
0048EADA        push        ebx
0048EADB        xor         eax,eax
0048EADD        mov         dword ptr [ebp-8],eax
0048EAE0        xor         eax,eax
0048EAE2        push        ebp
0048EAE3        push        48EB41
0048EAE8        push        dword ptr fs:[eax]
0048EAEB        mov         dword ptr fs:[eax],esp
0048EAEE        mov         dl,1
0048EAF0        mov         eax,[483250];TLicenseInfo
0048EAF5        call        TLicenseInfo.Create
0048EAFA        mov         dword ptr [ebp-4],eax
0048EAFD        lea         edx,[ebp-8]
0048EB00        mov         eax,dword ptr [ebp-4]
0048EB03        mov         eax,dword ptr [eax+8]
0048EB06        call        Trim
0048EB0B        cmp         dword ptr [ebp-8],0
0048EB0F>       je          0048EB19
0048EB11        lea         eax,[ebp-4]
0048EB14        call        TLicenseInfo.Load
>0048EB19       mov         eax,dword ptr [ebp-4]
0048EB1C        call        TLicenseInfo.GetType
0048EB21        mov         ebx,eax
0048EB23        mov         eax,dword ptr [ebp-4]
0048EB26        call        TObject.Free
0048EB2B        xor         eax,eax
0048EB2D        pop         edx
0048EB2E        pop         ecx
0048EB2F        pop         ecx
0048EB30        mov         dword ptr fs:[eax],edx
0048EB33        push        48EB48
>0048EB38       lea         eax,[ebp-8]
0048EB3B        call        @UStrClr
0048EB40        ret
0048EB41>       jmp         004045CC
0048EB46>       jmp         0048EB38
0048EB48        mov         eax,ebx
0048EB4A        pop         ebx
0048EB4B        pop         ecx
0048EB4C        pop         ecx
0048EB4D        pop         ebp
0048EB4E        ret         4
```

Imagen 4 - Función exportada de la librería OFCommon.dll

Adicionalmente, esta función intenta restaurar cualquier licencia de respaldo que encuentre y además en caso de encontrar previos ficheros validos cargara toda la información de la activación (tipo de licencia, fecha de expiración, etc).

Volviendo al código inicial, se procede entonces a trabajar con el serial introducido y aplicarle todas las validaciones antes de pasar a la segunda parte del proceso de activación en línea. Las primeras operaciones aplicadas serán las siguientes

```
xor         ecx,ecx
mov         edx,5198C0;'-'
mov         eax,dword ptr [ebp-4]
call        StringReplace
mov         eax,dword ptr [ebp-10]
lea         edx,[ebp-8]
call        Trim
lea         edx,[ebp-14]
mov         eax,dword ptr [ebp-8]
call        UpperCase
mov         edx,dword ptr [ebp-14]
lea         eax,[ebp-8]
call        @UStrLAsg
lea         eax,[ebp-8]
call        @EnsureUnicodeString
call        @UStrLen
cmp         eax,14
jne         00519817
```

Imagen 5 - Verificación de longitud de cadena

Como se aprecia la longitud de la licencia excluyendo el carácter comodín "-" debe ser igual a 20 caracteres, aunque ya esto se conocía desde la pantalla de entrada del serial.

LICENCIAS INVALIDAS

Las siguientes comprobaciones son llevadas a cabo buscando los primeros 18 caracteres de la licencia introducida dentro de un listado de seriales incluidos en el software que son considerados no validos o lo que comúnmente se conoce como puestos en lista negra. Dicho listado está comprendido por 176 licencias que puede ser que estén revocadas, vencidas, pirateadas, etc.

```
mov     edx, offset aF2c74c2fe45fbd ; "F2C74C2FE45FBD61FB"
mov     eax, edi
mov     ecx, [eax]
call    dword ptr [ecx+38h]
mov     edx, offset a647881155f82b3 ; "647881155F82B3973B"
mov     eax, edi
mov     ecx, [eax]
call    dword ptr [ecx+38h]
mov     edx, offset a231c4fd8d5af9c ; "231C4FD8D5AF9CD69B"
mov     eax, edi
mov     ecx, [eax]
call    dword ptr [ecx+38h]
mov     edx, offset aA5608ce580cde2 ; "A5608CE580CDE2D1CB"
mov     eax, edi
```

Imagen 6 - Parte del listado de licencias invalidas

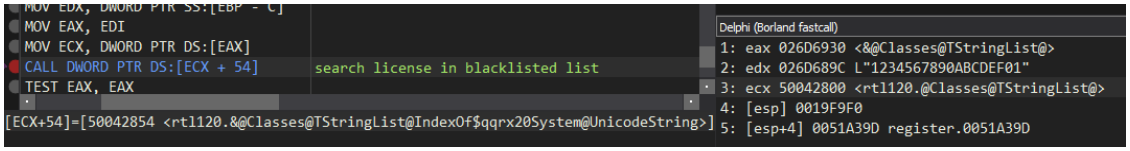

Imagen 7 - Buscando licencia introducida en lista negra

En el caso que la licencia introducida forme parte del listado negro no se realizaran mas comprobaciones y directamente será catalogada como invalida. Lo que continua luego es un segundo bloque de comprobaciones para licencias invalidas en el que se comprueba si varios caracteres forman parte de la licencia que se introdujo. Ambas comprobaciones verifican si las cadenas "*41E2*" y "*8-4A6*" forman parte de la licencia falsa. En el caso que ambas comprobaciones sean verdaderas, se pasa a comprobar el ultimo carácter el cual es verificado por "*4*" o "*8*" o "*C*" o "*F*". Siguiendo esta lógica, y en todo caso si la licencia introducida hubiera sido:

41E25-67890-ABCD8-4A624

La licencia es tratada automáticamente como invalida dando la siguiente información. "*This license code cannnot be activated manually because it has been pre-installed in ASC PRO version. Re-registering the license online will return it back to free version*"

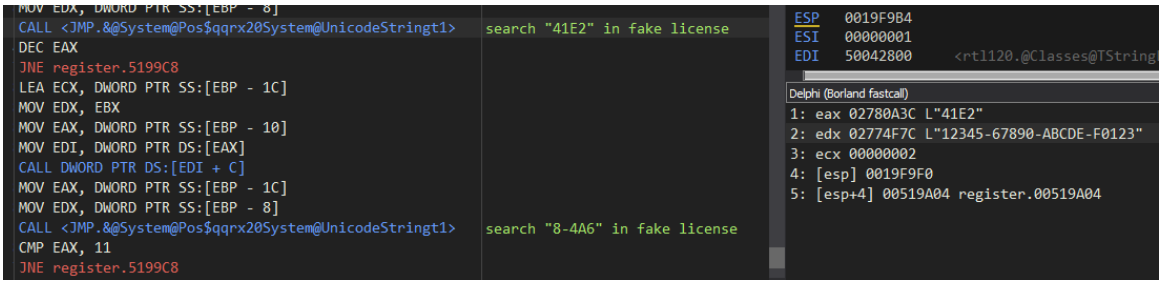

Imagen 8 - Verificando instancia de cadenas invalidas

Imagen 9 - Mensaje de licencia preinstalada

LICENCIA VALIDA

El algoritmo de comprobación de licencias validas se encuentra en la librería que se comentó anteriormente llamada "*OFCommon.dll*" y específicamente en la función exportada

"*IsTaskValidEx*". El algoritmo utilizado es muy sencillo y fácilmente reproducible, en este caso la función verifica la licencia introducida para dos versiones distintas de la aplicación. La primera verificación comprueba que la licencia sea para la versión Pro que es la que se esta analizando y la segunda para la versión Ultimate.

Comprobando la licencia Pro se utilizan varias verificaciones y operaciones, la longitud requerida continúa siendo de 20 caracteres excluyendo los guiones "-" usados como separadores, en otras palabras, se necesitan 4 grupos de 5 caracteres. Posteriormente se limpian caracteres introducidos por error por su carácter correcto, en este caso se cambian los "o" por "0", "i" por "1" y "l" por "1" para finalmente dividir la licencia en varios grupos

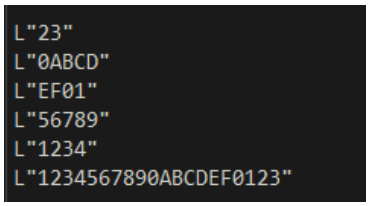

Imagen 10 - Grupos de caracteres creados

Cada grupo será usado para una verificación u operación específica. Las primeras operaciones serán sobre los grupos "*EF01*" y "*0ABCD*", de los cuales se obtiene un hash MD5[4], en este caso se obtiene lo siguiente:

MD5("*EF01*") = *405925253375E4CFF8255CD2D57B8C92*
MD5("*0ABCD*") = *C34B3EE22447089065B0A79B6AD397E8*

Seguidamente se compara el próximo grupo de caracteres extraídos de la licencia falsa, en este caso "*1234*" (primeros 4 caracteres del serial falso) con los últimos 4 caracteres del hash MD5 del grupo de caracteres "*EF01*".

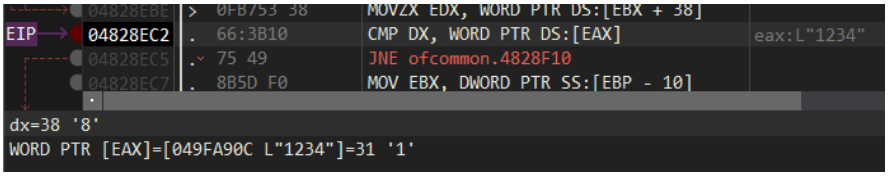

Imagen 11 - Comparando primeros caracteres de la licencia falsa

[4] "MD5", https://es.wikipedia.org/wiki/MD5

El mismo procedimiento es aplicado al grupo de caracteres "*56789*", los cuales son comparados a los primeros 5 caracteres del hash MD5 de la cadena "*0ABCD*". Por último, varios chequeos son dirigidos al grupo "*23*" (los últimos dos caracteres de la licencia falsa) los cuales identifican la versión del software para la cual la licencia introducida seria valida, además del tipo de licencia que seria. Una comprobación en común es que el ultimo carácter "3" no puede ser igual a "0". Existen muchos tipos de combinaciones que el software comprueba para de esta forma decidir que tipo de licencia se introdujo, un resumen de estas comprobaciones y sus resultados se encuentra en Table 1

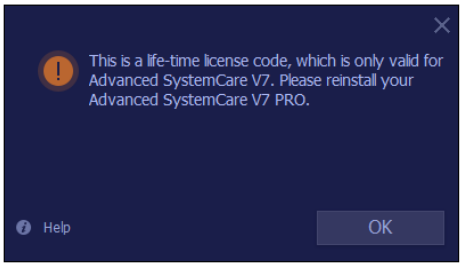

Imagen 12 - Mensaje de error usando la licencia *8C92C-34B30-ABCDE-F01B9*

1er carácter ("2")	2do carácter ("3")	Resultado
"4"	0-9A-Z	Licencia para versión 13 (válida)
0-9A-Z	NOT ("1", "4", "5", "8", "9", "C", "D", "H", "K", "M", "N")	
	"H", "K", "M", "N"	Licencia de sorteo (no valida)
"F"	"A", "B"	Licencia para versión 4
"E", "A", "F"	"5", "9", "D"	
"D"		Licencia para versión 5
"C"		Licencia para versión 6
"B"		Licencia para versión 7
"9"		Licencia para versión 8
"8"		Licencia para versión 9
"7"		Licencia para versión 10
"6"		Licencia para versión 11
"5"		Licencia para versión 12

Table 1 - Resumen de las combinaciones de los dos últimos caracteres de la licencia

Teniendo en cuenta lo anterior se puede conformar una licencia modificada para pasar estas comprobaciones como la siguiente:

8C92C-34B30-ABCDE-F0143

Luego de pasar correctamente las verificaciones se puede confirmar en el código que la licencia es aceptada como correcta por el algoritmo en la aplicación cliente, por lo que se procede al segundo paso de la activación en línea, la petición al servidor.

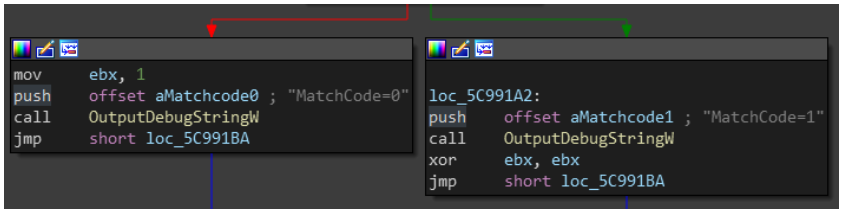

Imagen 13 - Confirmación de licencia valida al tomar la ruta derecha

Como se ha comentado anteriormente, la función "*IsTaskValidEx*" también verificara si la licencia introducida es para la versión Ultimate de la aplicación. En caso de no poseer las características antes discutidas que la harían válida para una versión Pro, también se verifica si en cambio es válida para la versión superior del software, en este caso la versión Ultimate. Aquí no se entrarán en detalles de este tipo de licencias ya que no es el objetivo de este escrito, solo comentar algunos aspectos o verificaciones que se llevan a cabo. La primera diferencia es que todos los caracteres en las licencias Ultimate son tratados en minúsculas, la longitud del serial seria idéntica al Pro, las funciones hash son aplicadas a diferentes grupos de caracteres ("*DEF01*", "*5678*") y comparadas con diferentes grupos de caracteres ("*90ABC*", "*1234*") que los usados en las licencias Pro.

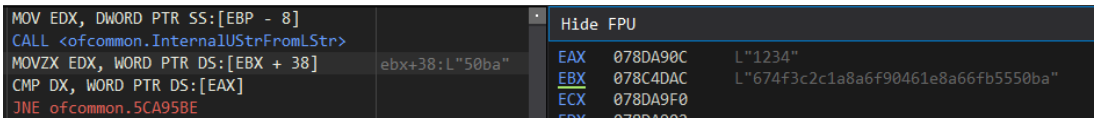

Imagen 14 - Verificando licencia Ultimate

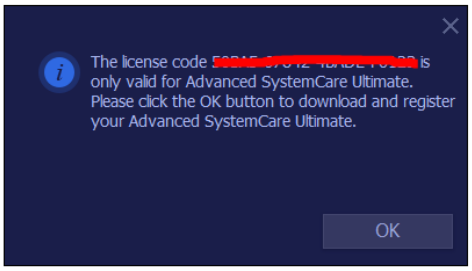

Imagen 15 - Mensaje de error al usar una licencia Ultimate

COMUNICACIÓN CLIENTE-SERVIDOR

Regresando al código principal del ejecutable "*Register.exe*" y una vez pasadas las comprobaciones de la licencia introducida, se procede al siguiente paso en la activación y la aplicación cliente enviara la petición necesaria al servidor para poder activarse correctamente. El mecanismo usado en esta ocasión se basa en una petición HTTP con información de hardware en las variables enviadas, potencialmente relevante para el proceso en el lado del servidor.

```
mov         dl,1
mov         eax,[516E10];TCheckOnlineThread
call        TThread@Create;TCheckOnlineThread.Create
mov         dword ptr [ebp-0E4],eax
```

Imagen 16 - Hilo responsable de la verificación en línea

Se comienza creando un hilo que contendrá la función que construirá la petición HTTP, la enviará y devolverá la respuesta obtenida del servidor, para proceder a su evaluación y por consiguiente activar o no el software.

CHECK HOSTS CHEAT

Antes de enviar la petición al servidor de activación el software realiza algo interesante y es que verifica que no se encuentre una redirección activa en el fichero "*hosts*" del equipo cliente con el dominio de activación "*asc.iobit.com*"

```
PUSH register.4B8E74                4B8E74:L"CheckHostsCheat"
CALL <JMP.&OutputDebugStringW>
```

Imagen 17 – Inicio de función de protección anti-redireccion local

El procedimiento que realiza se basa en verificar que no existe una entrada en el fichero "*hosts*" que redireccione su dominio. Esta función la realizara con un par de métodos, el primero consiste en realizar la búsqueda en todo el contenido del fichero

```
MOV EDX, DWORD PTR SS:[EBP - 20]
MOV EAX, DWORD PTR SS:[EBP - 8]
CALL <JMP.&@System@Pos$qqrx20System@UnicodeStringt1>
TEST EAX, EAX
JLE register.4B82CC
```

```
EIP   004B8265   register.004B8265
Delphi (Borland fastcall)
1: eax 00518C18 L"asc.iobit.com"
2: edx 040D4114 L"# Copyright (c) 1993-2009 Microsoft Corp."
3: ecx 00000002
4: [esp] 06F9FBB8
```

Imagen 18 - Búsqueda de dominio en contenido de fichero hosts

En caso de encontrar el dominio redirigido procede a eliminar dicha entrada completamente del fichero. El segundo método es más interesante y se usa en caso de fallar el primero, para ello el software hace ping a la URL "*asc.iobit.com*" y verifica que el nombre resuelto no sea otro que su propio nombre "*iobit-license.us-east-1.elasticbeanstalk.com*". Para ello crea un proceso "*cmd.exe*" del cual lee su buffer de salida con un Pipe[5].

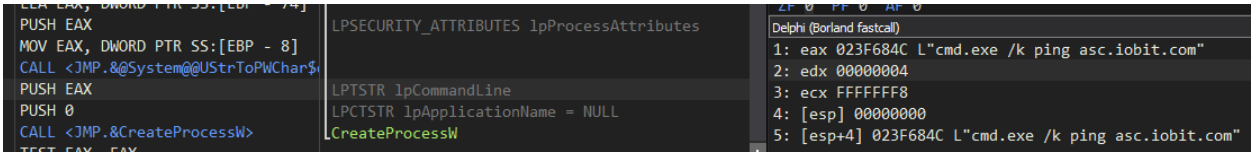

Imagen 19 - Creando proceso cmd.exe

PETICIÓN DE ACTIVACIÓN

Al terminar las comprobaciones para evitar posibles redirecciones y activaciones offline, el software finalmente obtiene varios datos del hardware instalado en la PC y crea con ellos el contenido de la petición que se enviara a los servidores de activación.

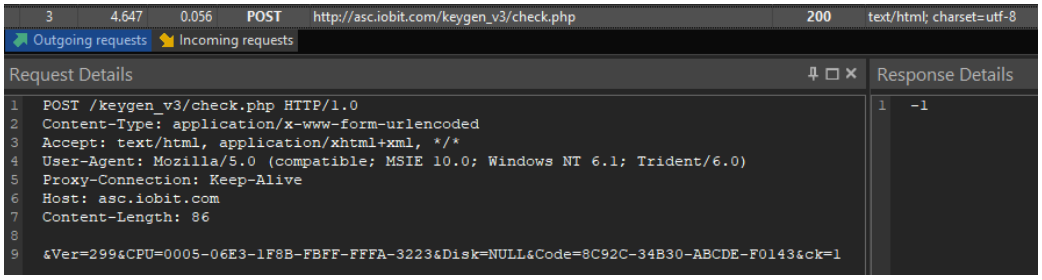

Imagen 20 - Petición de activación

El contenido de la petición incluye la licencia introducida y otras variables que no se verán en detalles ya que lo que se intenta es una activación sin los servidores oficiales, lo que hace que dicho contenido no sea relevante y por consiguiente se omiten en este escrito. La atención en este caso seria para la respuesta del servidor que por lo que se aprecia es -1.

[5] "CreatePipe function", https://docs.microsoft.com/en-us/windows/win32/api/namedpipeapi/nf-namedpipeapi-createpipe

RESPUESTA DEL SERVIDOR

Una vez obtenida la respuesta del servidor se procede al siguiente paso en el proceso de activación, donde la aplicación cliente actúa conforme al contenido recibido. El contenido de una respuesta correcta del servidor es desconocido pues evidentemente la licencia usada no es una licencia registrada y de ahí que se esté recibiendo un "-1" como respuesta. Lo siguiente es descifrar como sería una respuesta correcta y que es lo que espera la aplicación cliente para hacer dicha evaluación. Justo después de regresar del hilo encargado de hacer la petición el software hace un "Split[6]" de la cadena recibida usando el carácter "&" como separador, por lo que se puede deducir que una respuesta correcta estaría compuesta por dos o mas cadenas enlazadas con el carácter "&".

```
call    TStringStream@GetDataString
mov     eax,dword ptr [ebp-12C]
lea     edx,[ebp-1C]
call    Trim
lea     ecx,[ebp-104]
mov     edx,518F3C;'&'
mov     eax,dword ptr [ebp-1C]
call    SplitString
```

Imagen 21 - Separando la respuesta del servidor en un arreglo de cadenas

Seguidamente se toma la primera cadena del arreglo de cadenas creado y se comparara con una serie de constantes, las cuales potencialmente se traducen en el tipo de respuesta del servidor para la petición de activación enviada. Un resumen de las diferentes respuestas posibles seria las siguientes

Respuesta	Resultado
-1	Licencias no registradas
-2	Licencias vencidas o Licencias de sorteo no validas (Si penúltimo caracter = "4" y ultimo caracter = "8", "9", "A", "B")
-3	Licencias agotadas
-5 o -6	Error de conexión
0	Licencia registrada

Table 2 - Posibles respuestas de los servidores de activacion

[6] "System.StrUtils.SplitString", http://docwiki.embarcadero.com/Libraries/Rio/en/System.StrUtils.SplitString

DESCIFRANDO LA RESPUESTA CORRECTA

La primera cadena que forma la respuesta del servidor es el código de operación visto anteriormente y el resto de la respuesta es una o más cadenas adicionales. Anteriormente se vio que se creaba un nuevo array con cadenas usadas en la respuesta HTTP, luego a lo largo de la función dicho array es accedido y su contenido es manipulado.

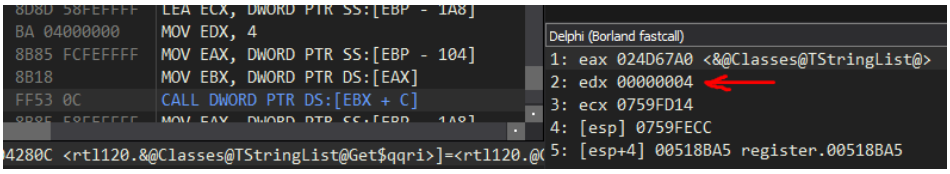

Imagen 22 - Tomando el contenido de la 5ta posición del array

Siguiendo estas pistas se puede deducir una longitud mínima necesaria para el arreglo de 5 posiciones ya que no se accede a un índice mayor posteriormente. Lo siguiente a descifrar es el tipo de contenido que se espera de cada grupo de cadenas.

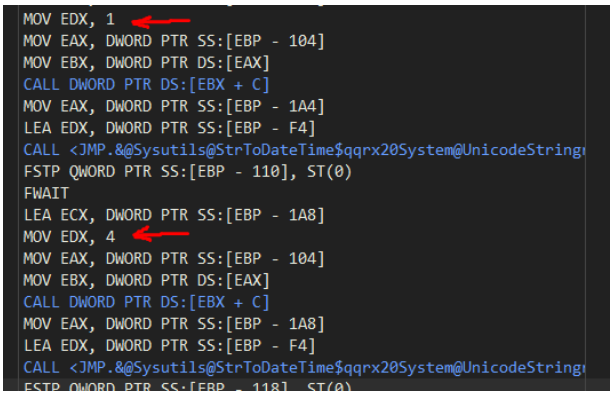

Imagen 23 - Convirtiendo el contenido del 2do y 5to grupo de caracteres a formato de fecha

Junto a código similar al visto en la Imagen 23 se puede finalmente conformar una básica pero posible respuesta del servidor de activación de la siguiente forma

Índice	Cadena	Significado
1	0	Licencia registrada
2	2030-01-02	Fecha de expiración
3	N/A	N/A
4	N/A	N/A
5	2020-01-02	Fecha actual
Respuesta Final		**0&2030-01-02&&&2020-01-02**

Table 3 - Descripción de los componentes de la respuesta de activación

Con una respuesta potencialmente satisfactoria ya generada se puede entonces proceder a interceptar la petición HTTP que realiza la aplicación "*Register.exe*" y modificarla por la anterior para continuar el análisis.

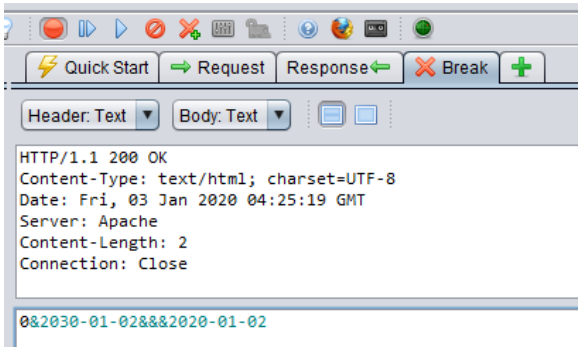

Imagen 24 - Modificando la respuesta HTTP del servidor de activación

Habiendo pasado el resto de verificaciones al usar una respuesta manipulada se procede a llamar otra función en la librería "*OFCommon.dll*", en este caso será la función exportada "*SetTaskInfo*", la cual se encarga de crear todos los ficheros de registro y/o activación en disco y en el registro de Windows. Los ficheros creados en el registro se encuentran bajo la clave

```
HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\IObit\ASC (x64)
HKEY_LOCAL_MACHINE\SOFTWARE\\IObit\ASC (x86)
```

Los ficheros de registro creados en el sistema de archivos se encuentran en la ruta

```
C:\Program Files (x86)\Common Files\IObit\Advanced SystemCare (x64)
C:\Program Files\Common Files\IObit\Advanced SystemCare (x86)
```

Finalmente se hace una última operación antes de concluir todo el proceso de registro. Se llama a una tercera aplicación

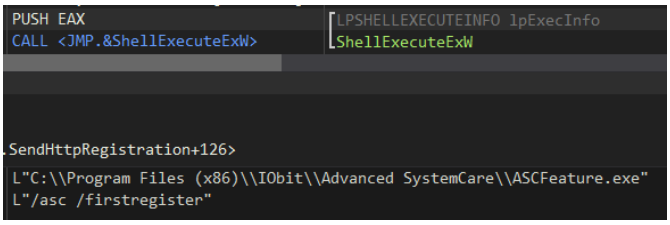

Imagen 25 - Ejecución de una tercera aplicación antes de completar la activación

Esta próxima aplicación hace un ultimo envío de información a los servidores de activación, lo cual parece ser una potencial petición de registro. Para ello incluye cierta información cifrada de nuestro hardware, así como algunos otros datos.

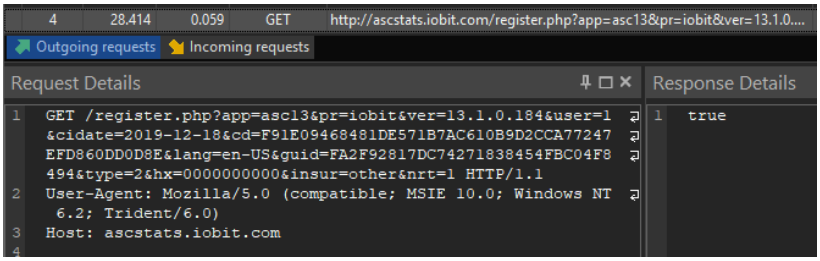

Imagen 26 - Petición GET a servidores de registro

El proceso de activación en línea es concluido con esta última operación y queda confirmado al obtener un mensaje como el siguiente y poder usar todas las opciones y características de la aplicación que una vez estuvieron bloqueadas

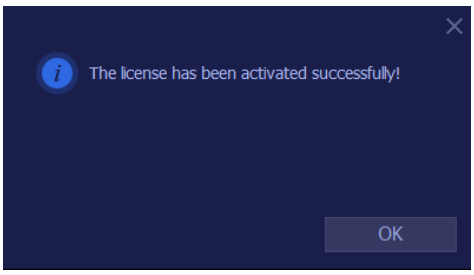

Imagen 27 - Activación completada con éxito

EMULANDO ACTIVACIÓN EN LÍNEA

Teniendo en cuenta todo lo analizado anteriormente y los procedimientos usados por el software para completar su registro y activación, se hace relativamente fácil escribir un servidor web que emule el comportamiento de los servidores originales de activación y a través del cual se pueda activar el software con cualquier licencia generada que cumpla los requisitos de validez sin estar registrada en las bases de datos propietarias.

Lo primero es añadir una entrada en el fichero "hosts" de la PC cliente con el fin de redireccionar peticiones al dominio "*asc.iobit.com*" a "*localhost*"[7]. Lo próximo es escribir un servidor web que escuche en "localhost" en el puerto "80" y responda de acuerdo a lo que el software espera recibir para su correcta activación[8].

```
static void Main(string[] args)
{
    const string serverUri = "http://127.0.0.1:80/keygen_v3/";

    // start web server
    var ws = new WebServer(SendResponse, serverUri);
    ws.Run();
}

1 reference
private static string SendResponse(HttpListenerRequest request)
{
    const int licenseYears = 30;
    const string activationResponse = "0";

    string expirationDate = DateTime.Now.AddYears(licenseYears).ToString(format: "yyyy-MM-dd");
    string currentDate = DateTime.Now.ToString(format: "yyyy-MM-dd");

    // send activation response
    return $"{activationResponse}&{expirationDate}&&&{currentDate}";
}
```

Imagen 28 - Servidor de activación

De lo otro que se necesita ocupar el servidor web de activación seria de la protección del fichero hosts de la PC cliente que realiza el software. Como se analizo anteriormente, con el fin de evitar posibles validaciones offline como la que se intenta hacer, el software chequea que no exista una redirección de su dominio. Una posible y sencilla solución a este tipo de protección seria la de bloquear el fichero "hosts" mientras se activa el software para evitar cualquier modificación en el

[7] "Tips and Tricks Using the Windows Hosts File", http://techgenix.com/tips-and-tricks-using-windows-hosts-file/
[8] "Creating a web server in C#", https://www.technical-recipes.com/2016/creating-a-web-server-in-c/

mismo. Un método para lograr esto seria obtener un manejador (HANDLE) exclusivo e impedir posteriores lecturas o escrituras sobre el mismo por segundas aplicaciones.

Se puede obtener el HANDLE de un archivo (CreateFile()) con el parámetro *dwShareMode* igual a cero, lo que impide que otros procesos abran dicho archivo o dispositivo si solicita acceso de eliminación, lectura o escritura[9]. Una vez obtenido el HANDLE en modo exclusivo, ningún otro proceso podrá abrirlo, eliminarlo, etc, hasta que este sea liberado.

El generador de licencias podría ser algo como el siguiente

```
private static string GenerateLicense()
{
    const int firstRandHexSize = 4;
    const int secondRandHexSize = 5;
    const int posSecondChunk = 11;
    const int posThirdChunk = 17;
    const string licenseDiv = "-";

    string firstRandHex = GetRandomHexNumber(firstRandHexSize);
    string secondRandHex = GetRandomHexNumber(secondRandHexSize);
    int lastRandNumber = Rnd.Next(1, 10);

    // keygen algo
    string firstMd5Hash = CalculateMD5Hash(firstRandHex);
    string secondMd5Hash = CalculateMD5Hash(secondRandHex);
    string firstMd5Chunk = firstMd5Hash.Substring( startIndex: firstMd5Hash.Length - firstRandHexSize,  length: firstRandHexSize);
    string secondMd5Chunk = secondMd5Hash.Substring( startIndex: 0,  length: secondRandHexSize);
    string license = $"{firstMd5Chunk}{secondMd5Chunk}{secondRandHex}{firstRandHex}4{lastRandNumber}";

    return license.Insert(secondRandHexSize, licenseDiv).
                   Insert(posSecondChunk, licenseDiv).
                   Insert(posThirdChunk, licenseDiv);
}
```

Imagen 29 - Generador de licencias

Luego de juntar todo lo anterior e intentar realizar una nueva activación con el servidor web de activación emulado se puede observar que todo funciona como debería y se obtiene un registro y activación satisfactoria.

[9] "CreateFile", "https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilea"

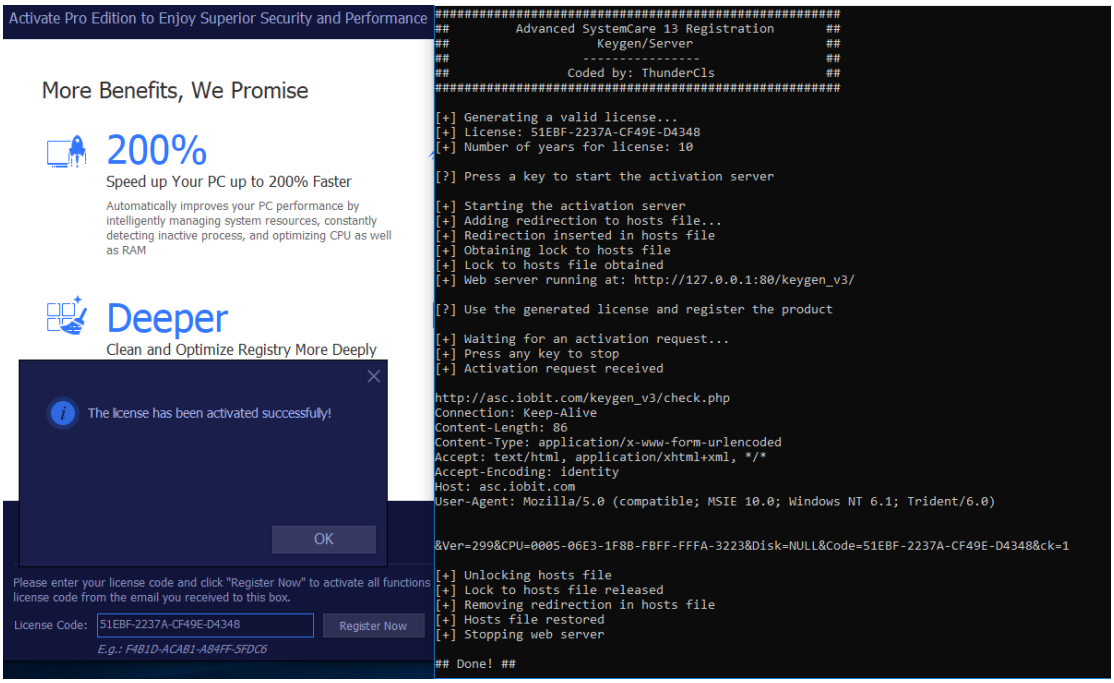

Imagen 30 – Emulando activación online

HERRAMIENTAS UTILIZADAS

ExeinfoPe: Packer, compressor detector.

(http://www.exeinfo.byethost18.com/?i=1)

x64dbg: An open-source x64/x32 debugger for windows.

(https://x64dbg.com/#start)

Interactive Delphi Reconstructor: A decompiler of executable files (EXE) and dynamic libraries (DLL), written in Delphi

(https://github.com/crypto2011/IDR)

IDA Pro: The Interactive Disassembler is a disassembler for computer software which generates assembly language source code from machine-executable code.

(https://www.hex-rays.com/products/ida/)

ProxyCap: Enables you to redirect your computer's network connections through proxy servers.

(http://www.proxycap.com/)

HTTP Debugger Pro: Debug HTTP API calls to a back-end and between back-ends

(https://www.httpdebugger.com/)

OWASP ZAP: The OWASP Zed Attack Proxy (ZAP) is one of the world's most popular free security tools with a built-in HTPP debugger

(https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project)

Microsoft Visual Studio: Visual Studio dev tools & services make app development easy for any platform & language.

(https://visualstudio.microsoft.com/)