HDC

Hoy veremos los famosos archivos **DLL** para los que frecuentan **Windows**.

“A ver, a ver. Ésto es re conocido. Son los famosos archivitos que no se tocan.”

A menos que... quieras ser un **hacker** e intentar tocar todo. Pero ¿**qué** cuernos es este **misterioso** archivito y qué **contiene**? Bueno, para mi también es un misterio así que aquí termina la clase.
Es broma, es broma :). Vamos a intentar de que hoy salgamos de esta clase aprendiendo estas cosas.

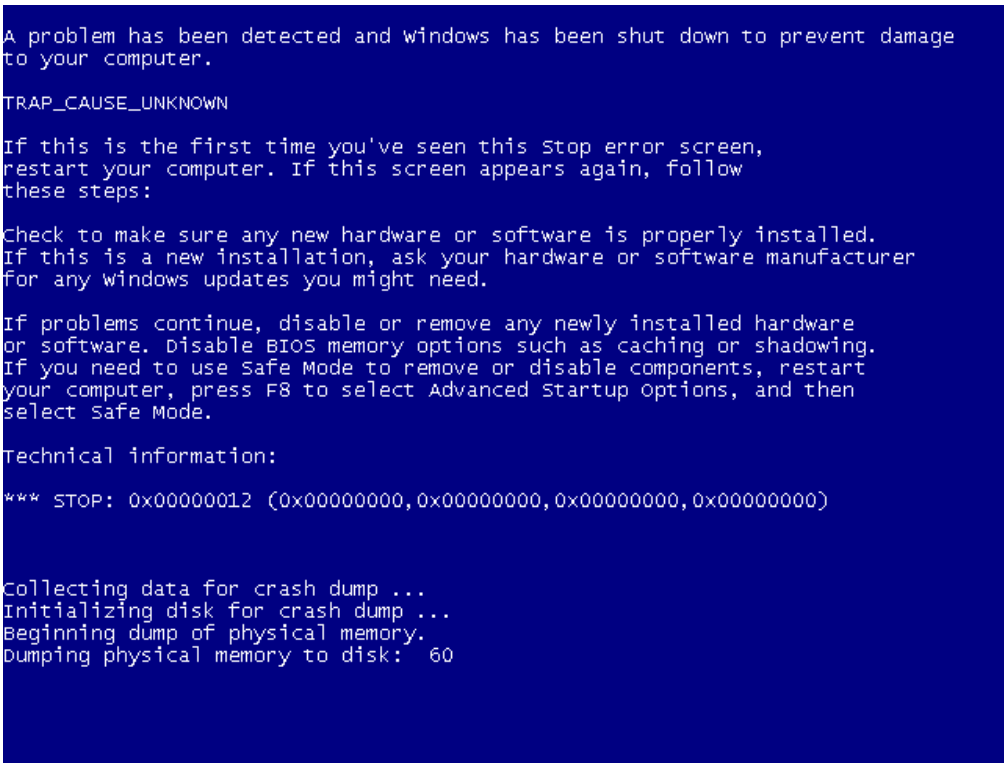
DLL es tambíen **biblioteca de enlace dinámico** y están compuestas de **código ejecutable**. Digamos, en sí esos archivos **no se ejecutan solos**, sino que otros ejecutables pueden usar este código para ejecutarlos pudiendo salvar espacio. Si yo desarrollo software y sé que va a ser ejecutado en un entorno de S.O. Windows. Ésto me da la posibilidad de **llamar** a las DLL que sé que tienen **funciones** que me sirven. Imaginen que lo puedo **compartir** con todos los programas que yo quiera y sigo teniendo **una sola biblioteca** sin que se multiplique el código por cada uno de los programas que necesiten usarlo. Además, para los desarrolladores, el trabajo **modular** hace todo más **eficaz**.

Pero no todo es color de rosa. Hay algo llamado **DLL hell** (sí, infierno de las DLL) que en realidad aunque su nombre lo llama así, yo en particular no lo veo como algo tan terrible. Aunque en sistemas críticos los problemas son siempre multiplicados.

"Estos pequeños no pueden hacer nada contra el gran Windows y su genial administrador:

Manolo."

Ay, Manolo. Los problemas pueden ser densos porque muchas veces terminan en la famosa **BSOD** o **Pantalla Azul de la Muerte**.



En sí, estos problemas son causados por dos razones. Como siempre, cuando instalamos programas más y más nuevos, van actualizando muchos archivos como las DLL del sistema. Éstas pueden hacer que otros programas queden incompatibles. Y del otro lado, si desinstalamos un programa y quita DLL's compartidas que otros softwares utilizan para su buen funcionamiento.
Es algo tan simple como modificar código de programa. Los programadores y aquellos que estuvieron practicando C, sabrán que no es para nada divertido.
Generalmente el primer problema está resuelto si usamos un instalador MSI.

"¿Y qué es lo que tiene dentro una DLL? ¿Magia?"

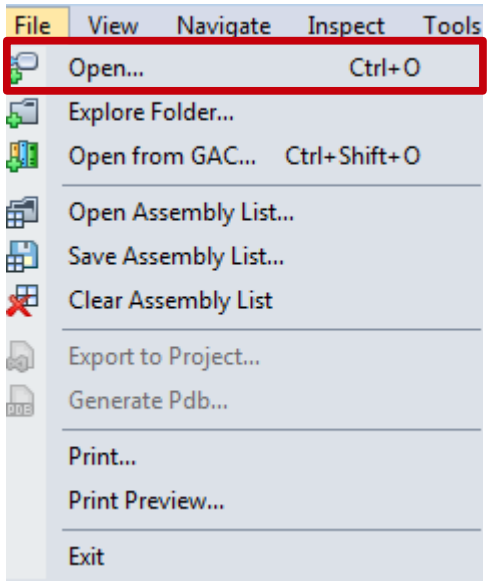
Pues no. Magia no. Las DLL se separan en 3 partes:

1. Definiciones de variables y funciones.
2. DllEntryPoint: función encargada de cargar y descargar en memoria.
3. Funciones.

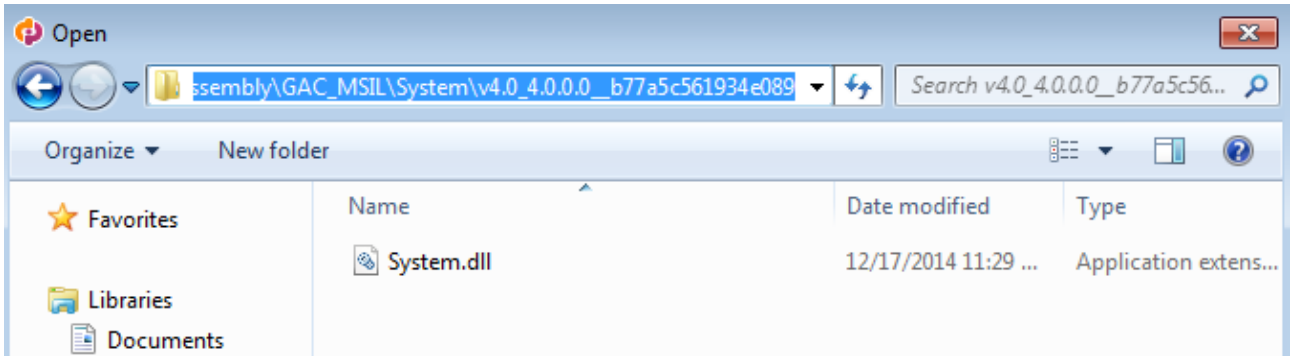
```
using System;
using System.Diagnostics;
using System.Reflection;
using System.Resources;
using System.Runtime;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Security;
using System.Security.Permissions;

// Assembly System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c56
// MVID: E04F7B31-1547-4B53-A83D-8704C8566261
// References: mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77
// References: System.Configuration, Version=4.0.0.0, Culture=neutral, Public
// References: System.Xml, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b

[assembly: AssemblyDelaySign(true)]
[assembly: ComCompatibleVersion(1, 0, 3300, 0)]
[assembly: SatelliteContractVersion("4.0.0.0")]
[assembly: AssemblyKeyFile("f:\\dd\\tools\\devdiv\\EcmaPublicKey.snk")]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: AssemblyInformationalVersion("4.0.30319.1")]
[assembly: DefaultDependency(LoadHint.Always)]
[assembly: TypeLibVersion(2, 4)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSeq
[assembly: NeutralResourcesLanguage("en-US")]
[assembly: CompilationRelaxations(8)]
[assembly: StringFreezing]
```

Veamos cómo **editar** una de éstas con la herramienta **Dot Peek**. Lo podemos descargar de la página **https://www.jetbrains.com/decompiler/**. Corresponde a un **decompilador**. Es decir que hace lo contrario a un **compilador** (como vimos en **C**). El segundo agarra al código y lo traduce a lenguaje máquina mientras el decompiler toma lo que nosotros no podemos leer a un lenguaje un poco más humano. La idea es poder **leerlo**, **editarlo** y poder **recompilarlo**.

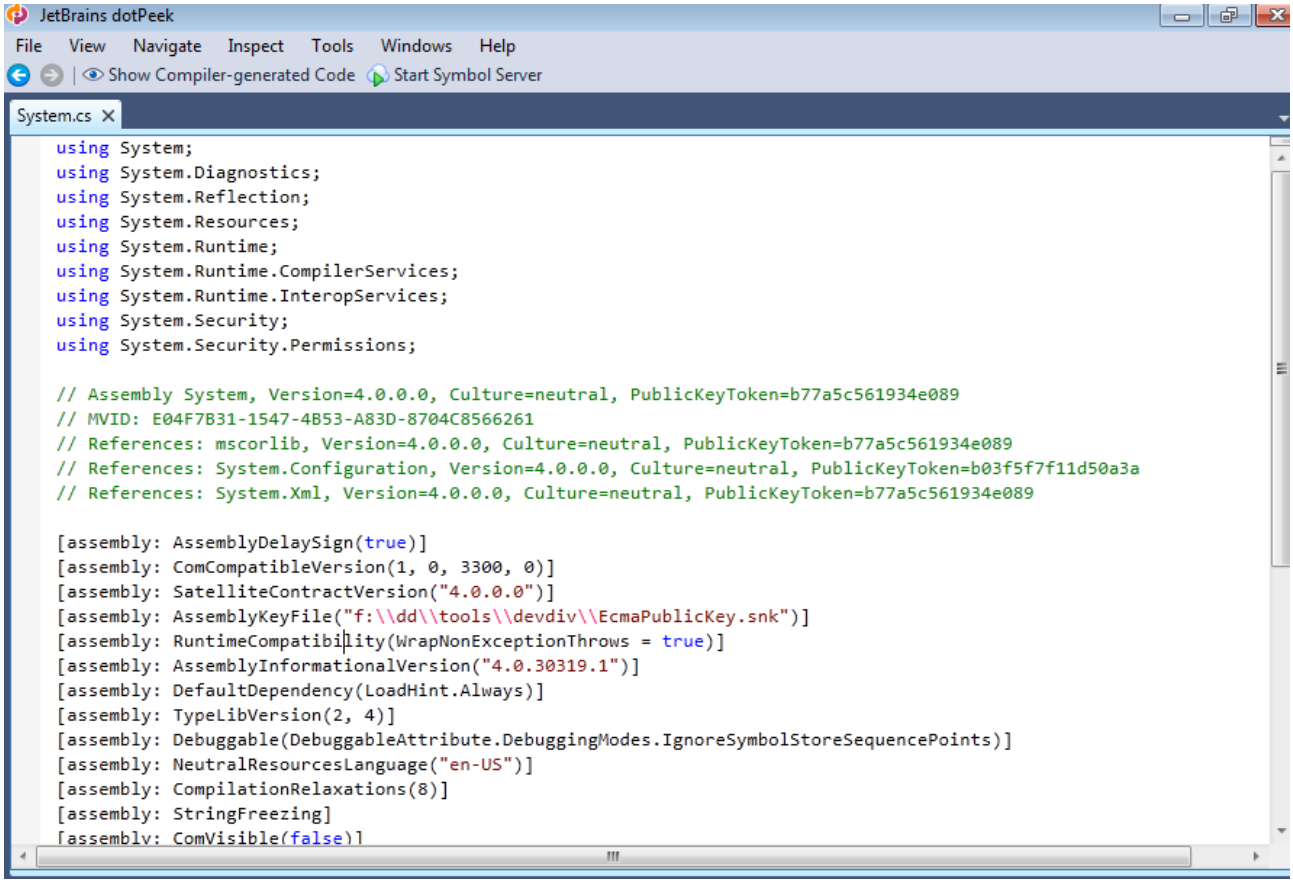
Luego de instalarlo, vamos a **File -> Open**. Allí nos habre una ventana.



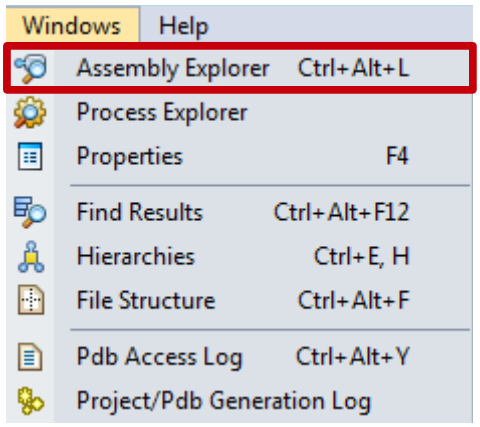
Vamos a la ruta
C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System\v4.0_4.0.0.0__b77a5c561934e089

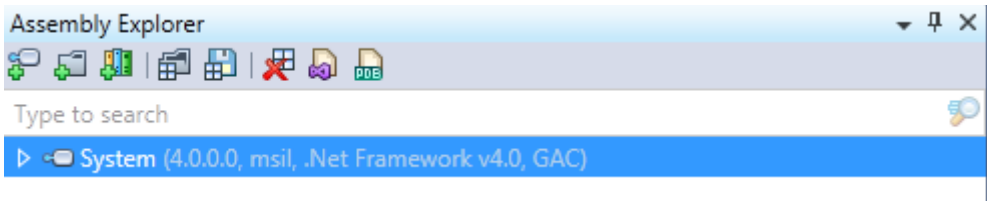


Y abrimos la DLL que se encuentra allí. Por favor no toquen nada porque liarla aquí no es nada agradable, y obviamente para mayor protección usen su **máquina virtual**.

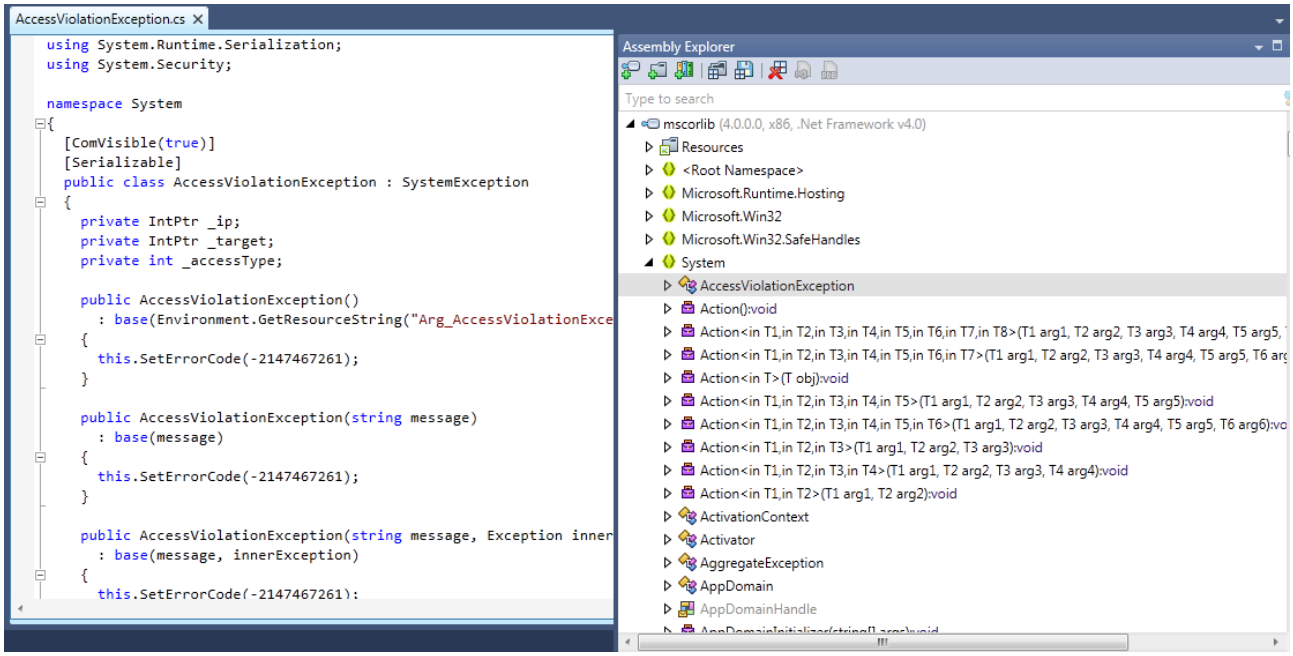


Verán, lo que vemos en esta ventana, son los metadatos del archivo. Vayamos a **Windows** -> **Assembly explorer.**





Ahora aquí le vamos dando a las **flechas** que **expanden** el **árbol** para poder ver más archivos y abrimos uno de éstos. Aquí tenemos código pero es **C#** (.NET y orientado a objetos) y nosotros no sabemos más que algunas cosillas en C.

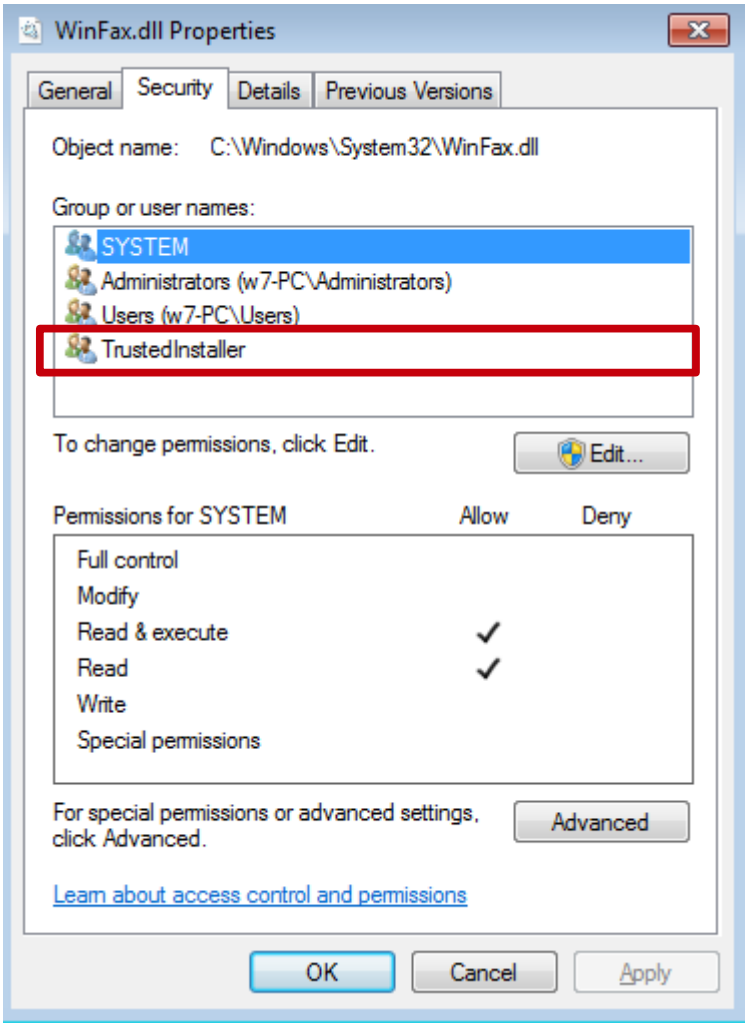


Es cierto, pero quizás podemos encontrar algo bueno. Por ejemplo, el S.O. cada vez que arranca, **carga** algunas **DLL**. Es decir que si encontramos cuáles son y les **inyectamos código** que haga que cargue algo más en memoria cada vez que comience, quizás podamos encontrar algo divertido:)

Claro, si intentamos editar una DLL tan crítica (como por ejemplo kernel32.dll que contiene código para la administración de servicios o de memoria) Windows nos dice que no podemos, que necesitamos **permiso**.

¿Por qué pasa esto? Si lo hago con una cuenta de privilegios de **administrador** también pasa. Bueno la verdad es que existe un **servicio** llamado **Windows Resource Protection (WRP)** que se encarga de hacer que ciertos archivos o carpetas **críticos no** se **modifiquen** y si alguno llega a ser reemplazado de manera imprevista, este servicio mismo se encargará de **reemplazar** el archivo con una **copia original** alocada en **C:/Windows/winsxs/Backup**. En realidad, el permiso es únicamente del sistema y puede desbloquear la protección de manera temporal para editar esos archivos.

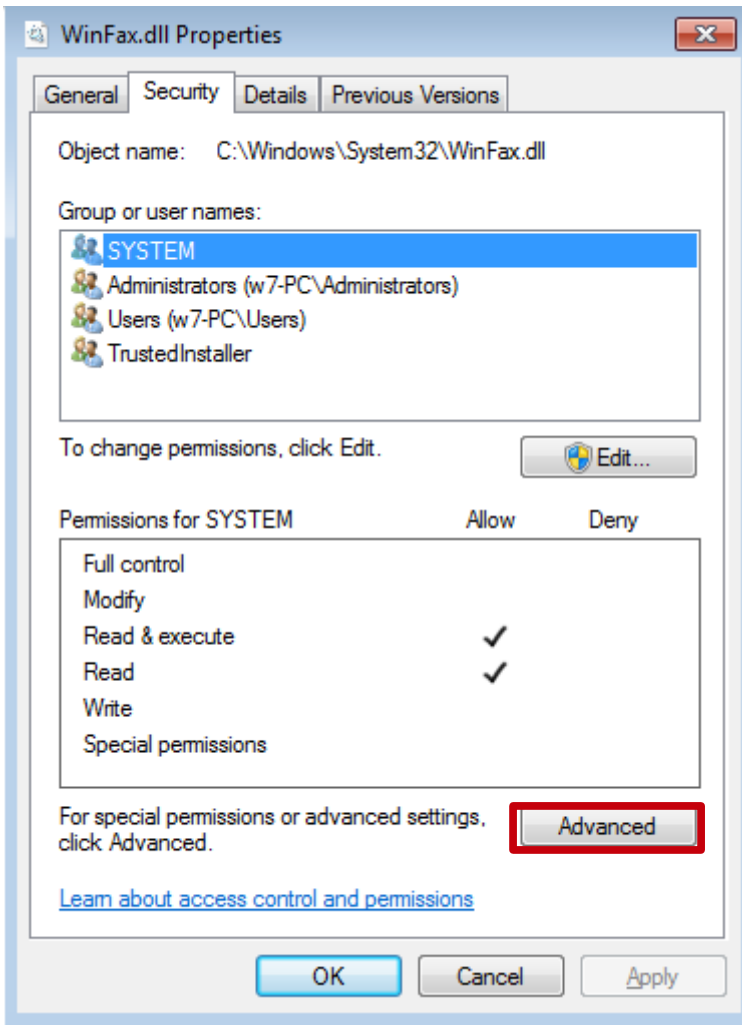
Si vamos al **WinFax.dll** y hacemos **Click Derecho -> Properties -> pestaña Security**, veremos un usuario que parece no ser nuestro.

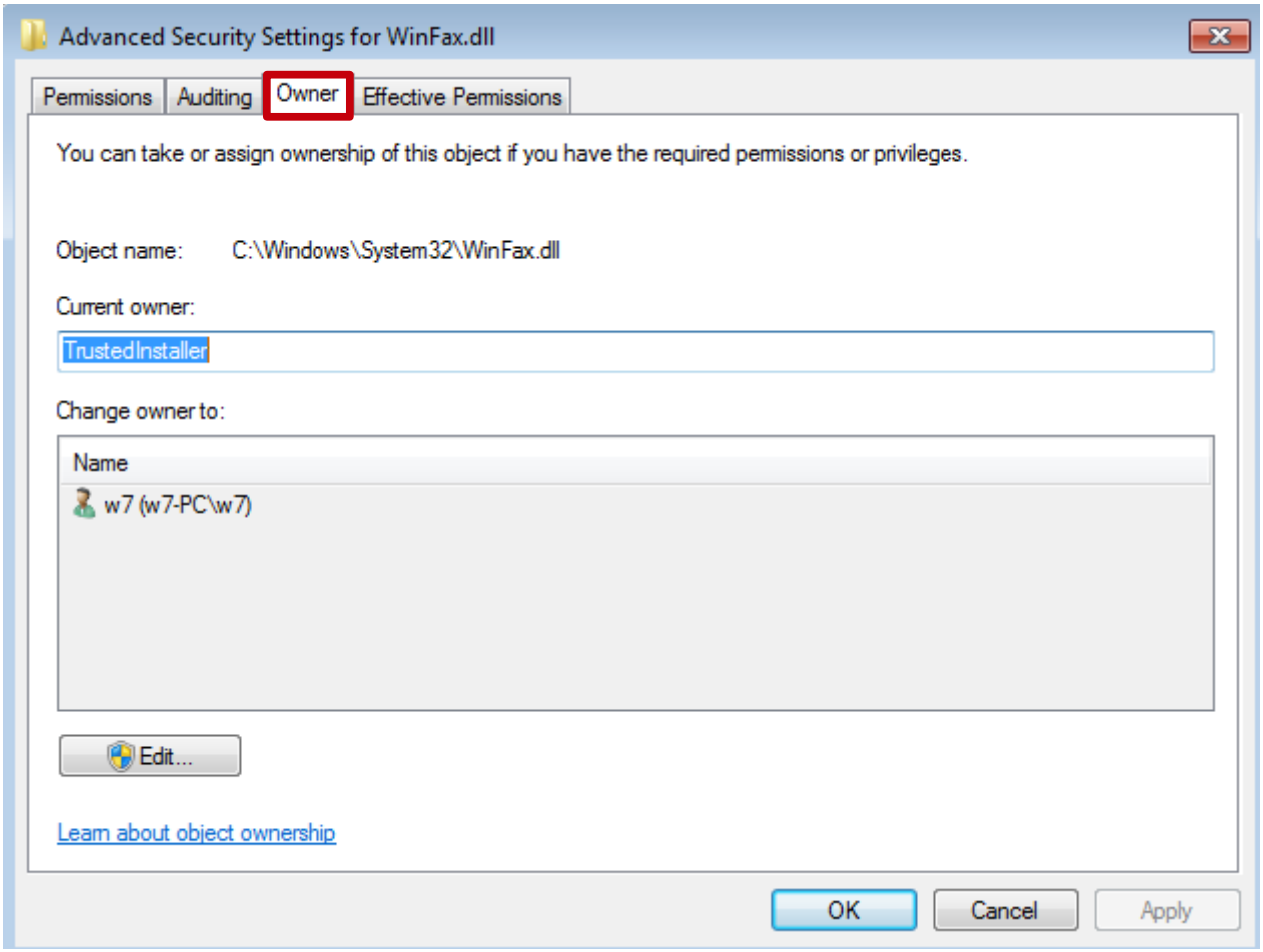


"¿Qué hace allí? ¡Es un virus! ¡Nos están espiando!"

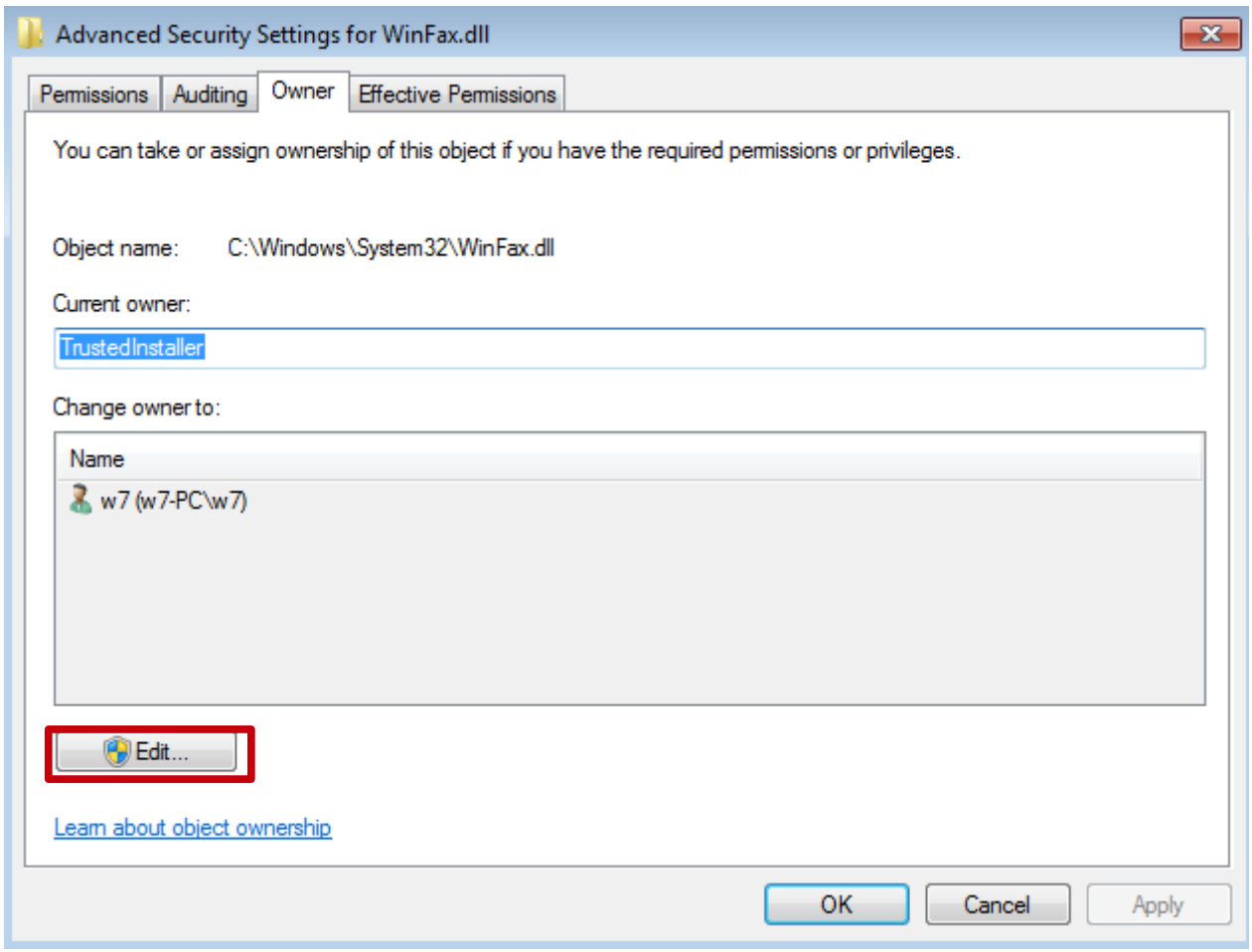
Manolo baja de la palmera que no es necesario que atentemos a las conspiraciones. El usuario **TrustedInstaller** es justamente aquel que le da **permiso** a **Windows** para **modificar** este tipo de archivos **críticos**.

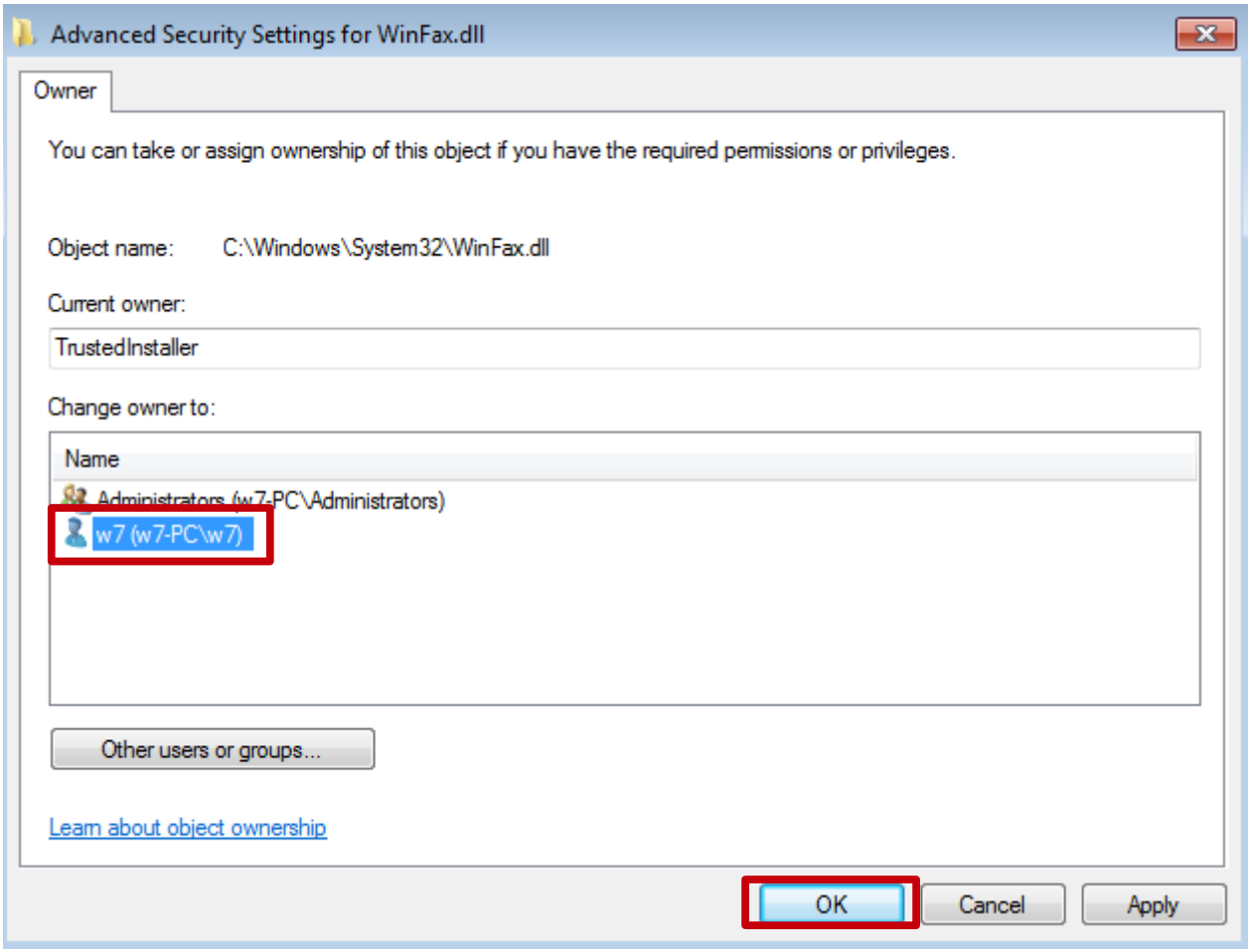
Vamos a intentar que nos de, a nosotros, esos **permisos**. Primero en **Advanced** nos abre una nueva ventana donde nos posicionaremos en la pestaña **Owner**.


WinFax.dll Properties
General
Security
Details
Previous Versions
Object name: C:\Windows\System32\WinFax.dll
Group or user names:
SYSTEM
Administrators (w7-PC\Administrators)
Users (w7-PC\Users)
TrustedInstaller
To change permissions, click Edit.
Edit...
Permissions for SYSTEM
Allow
Deny
Full control
Modify
Read & execute
Read
Write
Special permissions
For special permissions or advanced settings, click Advanced.
Advanced
Learn about access control and permissions
OK
Cancel
Apply

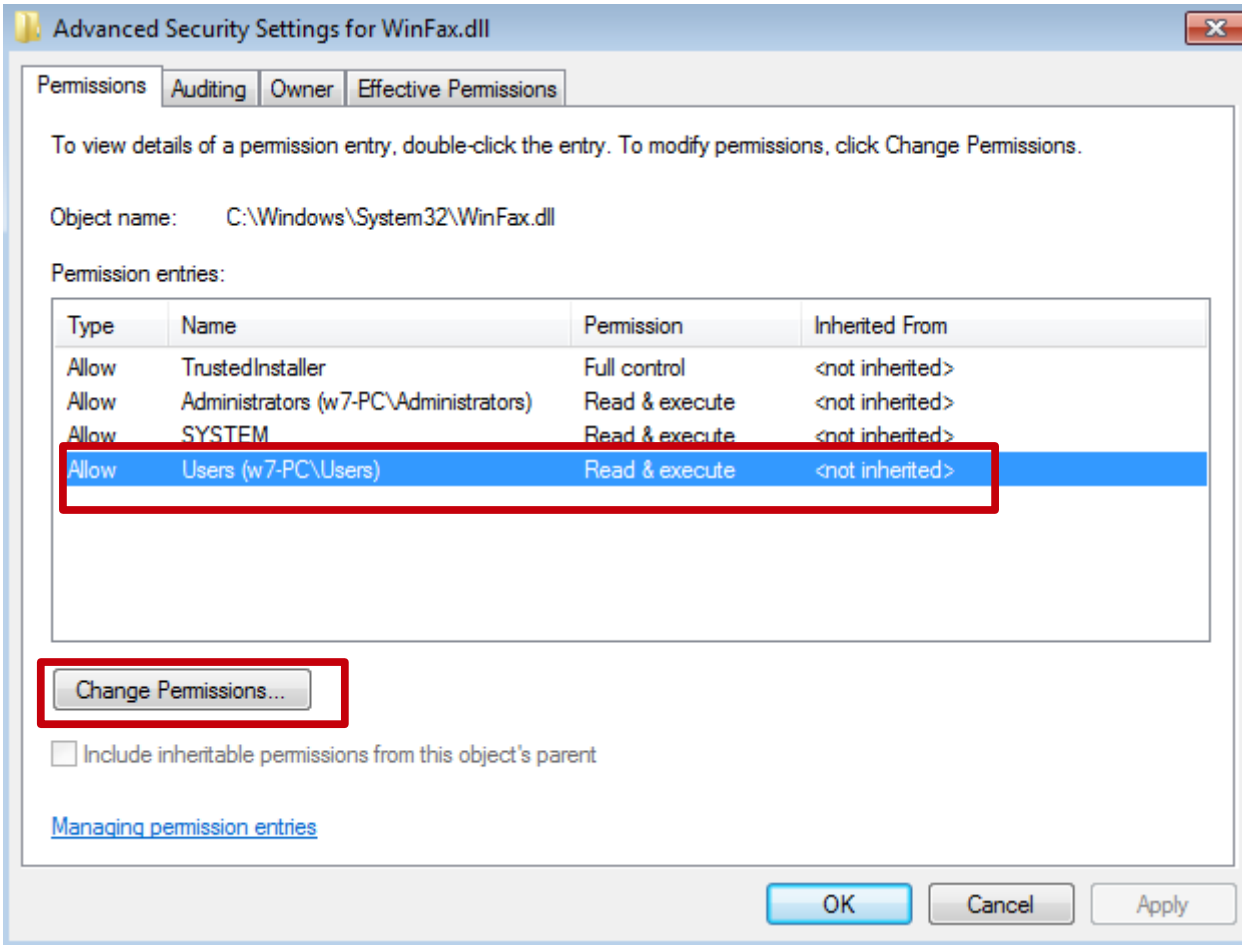


Allí abajo podemos **editar** el **Owner** y le ponemos nuestro **usuario** (en mi caso w7).

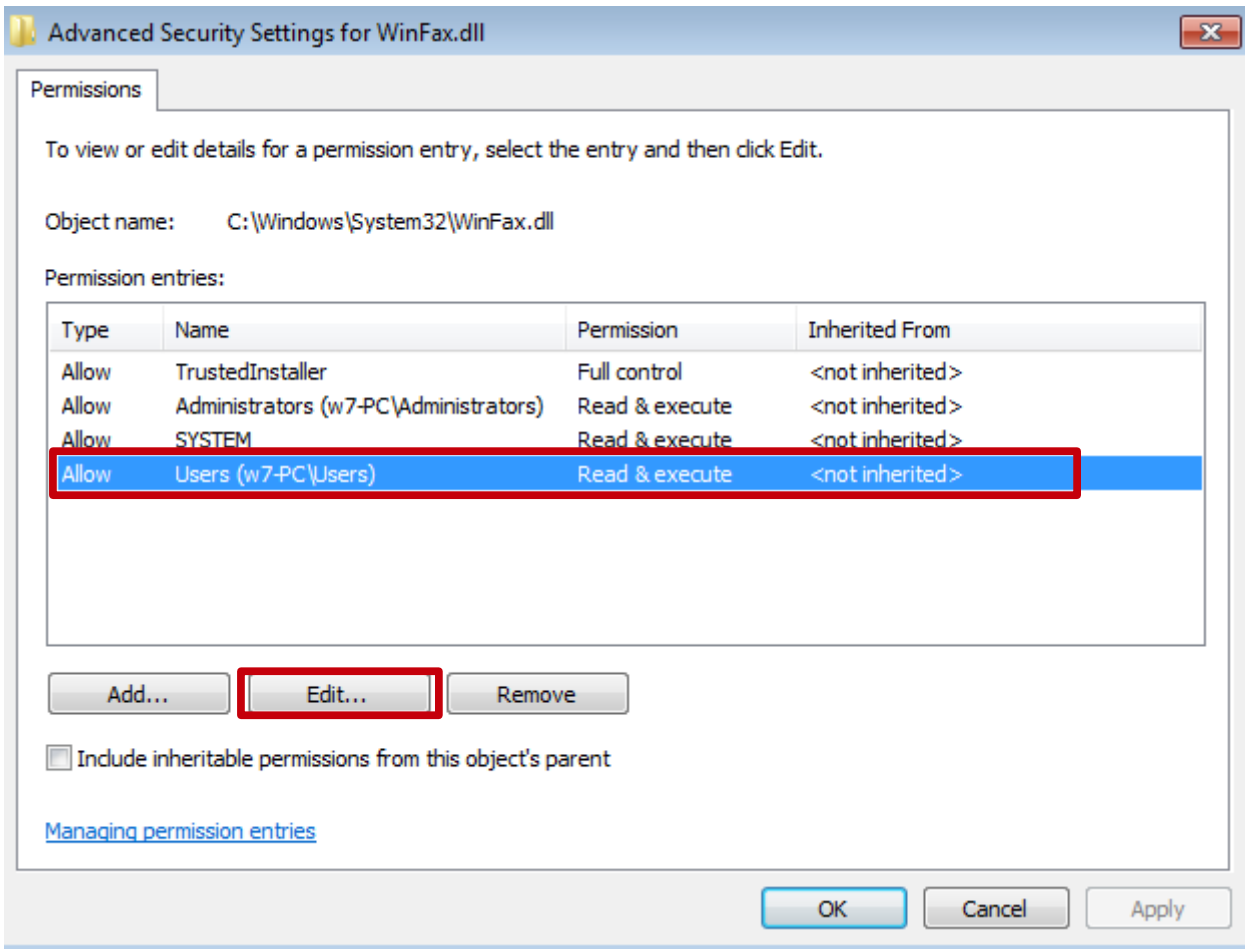

Advanced Security Settings for WinFax.dll
Permissions
Auditing
Owner
Effective Permissions
You can take or assign ownership of this object if you have the required permissions or privileges.
Object name: C:\Windows\System32\WinFax.dll
Current owner:
TrustedInstaller
Change owner to:
Name
w7 (w7-PC\w7)
Edit...
Learn about object ownership
OK
Cancel
Apply

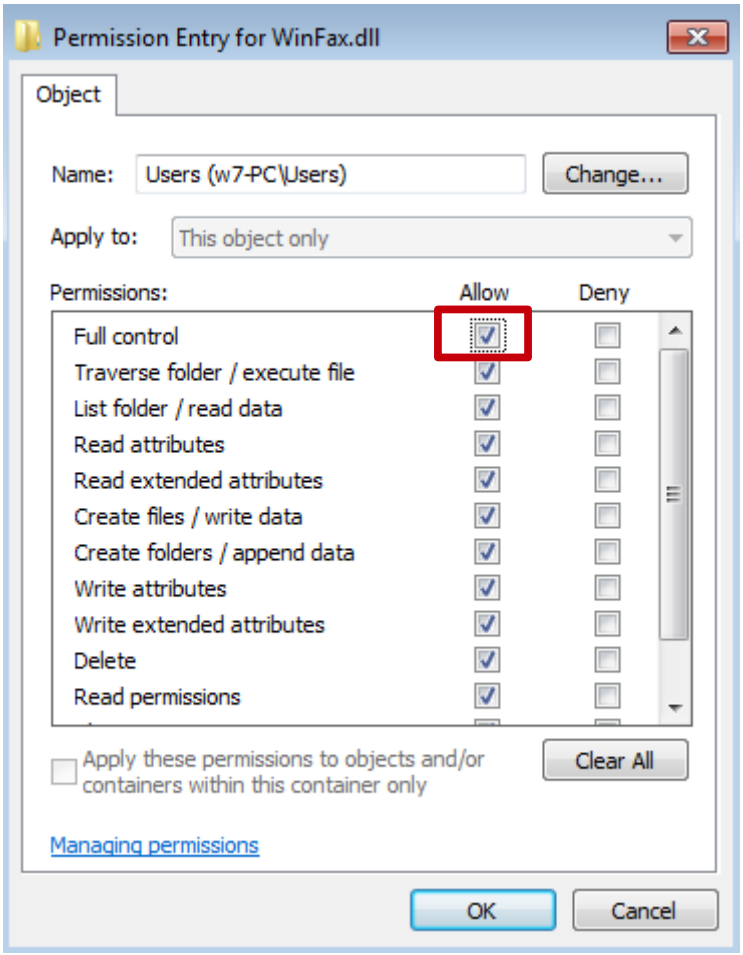


Aceptamos todo y volvemos a darle **click derecho**, **properties** y **Advanced**, dentro de la pestaña **Security**. Pero ahora le damos a **Change Permissions**. Aquí elegimos nuestro usuario y le damos **Full Permissions**.



Y ya tenemos suficiente para poder editarlo (aunque no con el DotPeek). Nos hicimos con su protección para editarlo de la manera que queremos. Ojo con qué le van a editar porque pueden armarse un lío que se van a querer colgar de una torre.


Advanced Security Settings for WinFax.dll
Permissions
To view or edit details for a permission entry, select the entry and then click Edit.
Object name: C:\Windows\System32\WinFax.dll
Permission entries:
Type
Name
Permission
Inherited From
Allow
TrustedInstaller
Full control
<not inherited>
Allow
Administrators (w7-PC\Administrators)
Read & execute
<not inherited>
Allow
SYSTEM
Read & execute
<not inherited>
Allow
Users (w7-PC\Users)
Read & execute
<not inherited>
Add...
Edit...
Remove
Include inheritable permissions from this object's parent
Managing permission entries
OK
Cancel
Apply


Permission Entry for WinFax.dll
Object
Name: Users (w7-PC\Users)
Change...
Apply to: This object only
Permissions:
Allow
Deny
Full control
Traverse folder / execute file
List folder / read data
Read attributes
Read extended attributes
Create files / write data
Create folders / append data
Write attributes
Write extended attributes
Delete
Read permissions
Apply these permissions to objects and/or containers within this container only
Clear All
Managing permissions
OK
Cancel

Hoy no vamos a darle demasiada énfasis al tema porque no es necesario. Está bien entender el sistema de protección que usa el sistema. Aquí terminamos con la parte de DLL's. Espero que se metan un poco más a fondo con el conocimiento de Windows. Y ya estamos por empezar los laboratorios:D.

Pueden seguirme en Twitter: @RoaddHDC

Cualquier cosa pueden mandarme mail a: r0add@hotmail.com

Para donaciones, pueden hacerlo en bitcoin en la dirección siguiente: 1HqpPJbbWJ9H2hAZTmpXnVuoLKkP7RFSvw

Roadd.
