عنوان: **تغییر عملکرد و یا ردیابی توابع ویندوز با استفاده از Hookهای دات نتی**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۱۱ ۱۴:۰
آدرس: www.dotnettips.info
گروه‌ها: DateTime, Persian, Windows Hooks

مقدمه

در حالت پیشرفته‌ی تزریق وابستگی‌ها در دات نت، با توجه به اینکه کار وهله سازی کلاس‌ها به یک کتابخانه جانبی به نام IoC Container واگذار می‌شود، امکان یک سری دخل و تصرف نیز در این میان فراهم می‌گردد. برای مثال الان که ما می‌توانیم یک کلاس را توسط IoC container به صورت خودکار وهله سازی کنیم، خوب، چرا اجرای متدهای آن‌را تحت نظر قرار ندهیم. مثلا حاصل آن‌ها را بتوانیم پیش از اینکه به فراخوان بازگشت داده شود، کش کنیم یا کلا تغییر دهیم. به این کار AOP یا Aspect oriented programming نیز گفته می‌شود.

واقعیت این است که یک چنین مفهومی از سال‌های دور به نام Hooking در برنامه‌های WIN32 API خالص نیز وجود داشته است. Hookها یا قلاب‌ها دقیقا کار Interception دنیای AOP را انجام می‌دهند. به این معنا که خودشان را بجای یک متد ثبت کرده و کار ردیابی یا حتی تغییر عملکرد آن متد خاص را می‌توانند انجام دهند. برای مثال اگر برای متد gethostbyname ویندوز یک Hook بنویسیم، برنامه استفاده کننده، تنها متد ما را بجای متد اصلی gethostbyname واقع در Kernel32 ویندوز، مشاهده می‌کند و درخواست‌های DNS خودش را به این متد ویژه ما ارسال خواهد کرد؛ بجای ارسال درخواست‌ها به متد اصلی. در این بین می‌توان درخواست‌های DNS را لاگ کرد و یا حتی تغییر جهت داد.

انجام Interception در دنیای دات نت با استفاده از امکانات Reflection و Reflection.Emit قابل انجام است و یا حتی بازنویسی اسمبلی‌ها و افزودن کدهای IL مورد نیاز به آن‌ها که به آن IL Weaving هم گفته می‌شود. اما در دنیای WIN32 انجام چنین کاری ساده نیست و ترکیبی است از زبان اسمبلی و کتابخانه‌های نوشته شده به زبان C.

برای ساده سازی نوشتن Hookهای ویندوزی، کتابخانه‌ای به نام easy hook ارائه شده است که امکان تزریق Hookهای دات نتی را به درون پروسه برنامه‌های Native ویندوز دارد. این قلاب‌ها که در اینجا متدهای دات نتی هستند، نهایتا بجای توابع اصلی ویندوز جا زده خواهند شد. بنابراین می‌توانند عملیات آن‌ها را ردیابی کنند و یا حتی پارامترهای آن‌ها را دریافت و مقدار دیگری را بجای تابع اصلی، بازگشت دهند. در ادامه قصد داریم اصول و نکات کار با easy hook را در طی یک مثال بررسی کنیم.

صورت مساله

می‌خواهیم کلیه درخواست‌های تاریخ اکسپلورر ویندوز را ردیابی کرده و بجای ارائه تاریخ استاندارد میلادی، تاریخ شمسی را جایگزین آن کنیم.

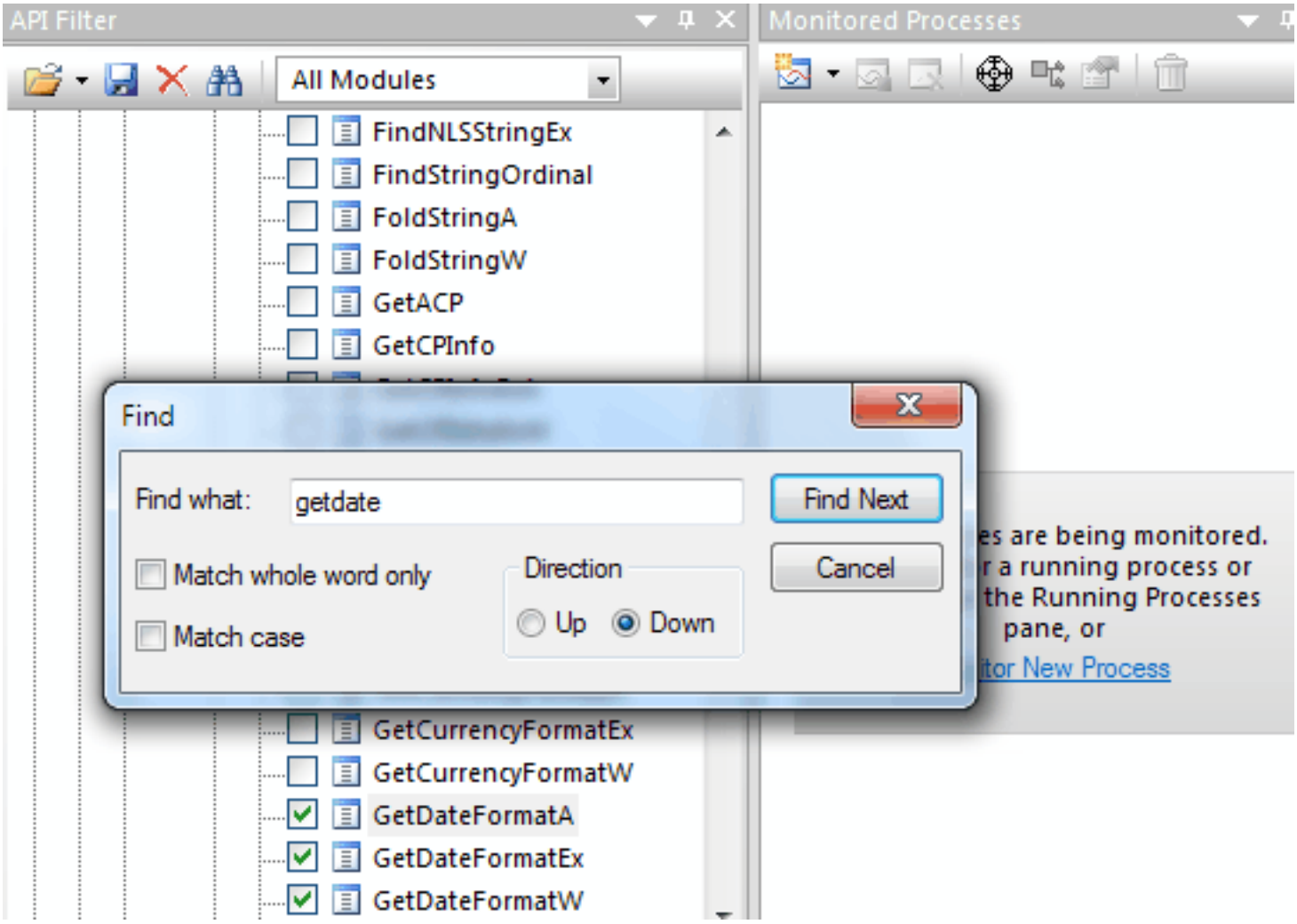
از کجا شروع کنیم؟

ابتدا باید دریابیم که اکسپلورر ویندوز از چه توابع API ایی برای پردازش‌های درخواست‌های تاریخ و ساعت خودش استفاده می‌کند، تا بتوانیم برای آن‌ها Hook بنویسیم. برای این منظور می‌توان از برنامه‌ی بسیار مفیدی به نام API Monitor استفاده کرد. این برنامه‌ی رایگان را از آدرس ذیل می‌توانید دریافت کنید:

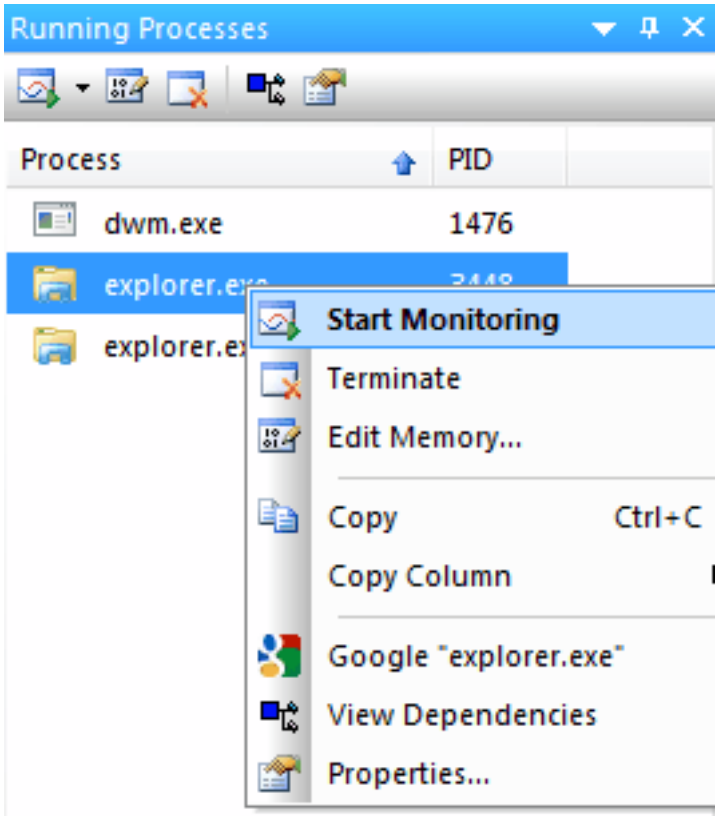
http://www.rohitab.com/apimonitor

اگر علاقمند به ردیابی برنامه‌های 32 بیتی هستید باید apimonitor-x86.exe را اجرا کنید و اگر نیاز به ردیابی برنامه‌های 64 بیتی دارید باید apimonitor-x64.exe را اجرا نمائید. بنابراین اگر پس از اجرای این برنامه، برای مثال فایرفاکس را در لیست پروسه‌های آن مشاهده نکردید، یعنی apimonitor-x64.exe را اجرا کرده‌اید؛ از این جهت که فایرفاکس عمومی تا این تاریخ، نسخه 32 بیتی است و نه 64 بیتی.

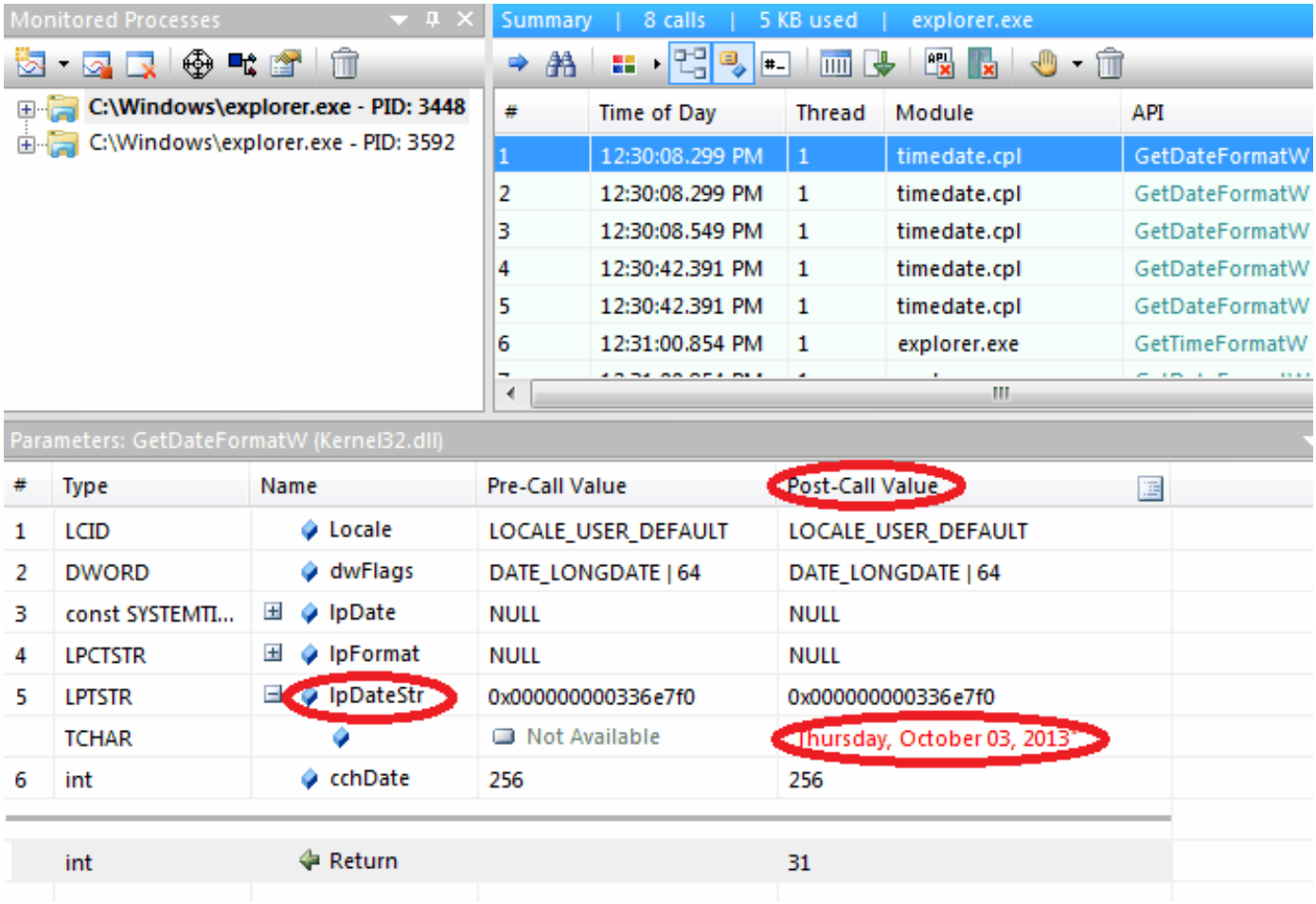
پس از اجرای برنامه API Monitor، در قسمت API Filter آن باید مشخص کنیم که علاقمند به ردیابی کدامیک از توابع API ویندوز هستیم. در اینجا چون نمی‌دانیم دقیقا کدام تابع کار ارائه تاریخ را به اکسپلورر ویندوز عهده دار است، شروع به جستجو می‌کنیم و هر تابعی را که نام date یا time در آن وجود داشت، تیک می‌زنیم تا در کار ردیابی لحاظ شود.



سپس نیاز است بر روی نام اکسپلورر در لیست پروسه‌های این برنامه کلیک راست کرده و گزینه Start monitoring را انتخاب کرد:



اندکی صبر کنید یا یک صفحه جدید اکسپلورر ویندوز را باز کنید تا کار ردیابی شروع شود:



همانطور که مشاهده می‌کنید، ویندوز برای ردیابی تاریخ در اکسپلورر خودش از توابع GetDateFormatW و GetTimeFormatW استفاده می‌کند. ابتدا یک تاریخ را توسط آرگومان lpDate یا lpTime به یکی از توابع یاد شده ارسال کرده و سپس خروجی را از آرگومان lpDateStr یا lpTimeStr دریافت می‌کند.
خوب؛ به نظر شما اگر این خروجی‌ها را با یک ساعت و تاریخ شمسی جایگزین کنیم بهتر نیست؟!

نوشتن Hook برای توابع GetDateFormatW و GetTimeFormatW ویندوز اکسپلورر

ابتدا کتابخانه easy hook را از مخزن کد CodePlex آن دریافت کنید:
https://easyhook.codeplex.com

سپس یک پروژه کنسول ساده را آغاز کنید. همچنین به این Solution، یک پروژه Class library جدید را نیز اضافه نمائید. پروژه کنسول، کار نصب Hook را انجام می‌دهد و پروژه کتابخانه‌ای اضافه شده، کار مدیریت هوک‌ها را انجام خواهد داد. سپس به هر دو پروژه، ارجاعی را به اسمبلی EasyHook.dll اضافه کنید.

الف) ساختار کلی کلاس Hook

کلاس Hook واقع در پروژه Class library باید یک چنین امضایی را داشته باشد:

```
namespace ExplorerPCal.Hooks
{
    public class GetDateTimeFormatInjection : IEntryPoint
    {

        public GetDateTimeFormatInjection(RemoteHooking.IContext context, string channelName)
        {
```

```
            // connect to host...
            _interface = RemoteHooking.IpcConnectClient<MessagesReceiverInterface>(channelName);
            _interface.Ping();
        }

        public void Run(RemoteHooking.IContext context, string channelName)
        {
        }
    }
}
```

یعنی باید اینترفیس IEntryPoint کتابخانه easy hook را پیاده سازی کند. این اینترفیس خالی است و صرفا کار علامتگذاری کلاس Hook را انجام می‌دهد. همچنین این کلاس باید دارای سازنده‌ای با امضایی که ملاحظه می‌کنید و یک متد Run، دقیقا با همین امضای فوق باشد.

ب) نوشتن توابع Hook

کار نوشتن قلاب برای توابع API ویندوز در متد Run انجام می‌شود. سپس باید توسط متد LocalHook.Create کار را شروع کرد. در اینجا مشخص می‌کنیم که نیاز است تابع GetDateFormatW واقع در kernel32.dll ردیابی شود.

```
public void Run(RemoteHooking.IContext context, string channelName)
        {
                GetDateFormatHook = LocalHook.Create(
                                        InTargetProc: LocalHook.GetProcAddress("kernel32.dll",
"GetDateFormatW"),
                                        InNewProc: new GetDateFormatDelegate(getDateFormatInterceptor),
                                        InCallback: this);
```

در ادامه توسط یک delegate، کلیه فراخوانی‌های ویندوز را که قرار است به GetDateFormatW اصلی ارسال شوند، ردیابی کرده و تغییر می‌دهیم.

ج) نحوه مشخص سازی امضای delegateهای Hook

اگر امضای متد GetDateFormatW به نحو ذیل باشد:

```
[DllImport("kernel32.dll", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Auto,
SetLastError = true)]
        public static extern int GetDateFormatW(
                                        uint locale,
                                        uint dwFlags, // NLS_DATE_FLAGS
                                        SystemTime lpDate,
                                        [MarshalAs(UnmanagedType.LPWStr)] string lpFormat,
                                        StringBuilder lpDateStr,
                                        int sbSize);
```

دقیقا delegate متناظر با آن نیز باید ابتدا توسط ویژگی UnmanagedFunctionPointer مزین شده و آن نیز دارای امضایی همانند تابع API اصلی باشد:

```
[UnmanagedFunctionPointer(CallingConvention.StdCall, CharSet = CharSet.Auto, SetLastError = true)]
        private delegate int GetDateFormatDelegate(
                                        uint locale,
                                        uint dwFlags,
                                        SystemTime lpDate,
                                        [MarshalAs(UnmanagedType.LPWStr)] string lpFormat,
                                        StringBuilder lpDateStr,
                                        int sbSize);
```

سپس callback نهایی که کار دریافت پیام‌های ویندوز را انجام خواهد داد نیز، همان امضاء را خواهد داشت:

```
private int getDateFormatInterceptor(
                                        uint locale,
                                        uint dwFlags,
                                        SystemTime lpDate,
```

```
                                        string lpFormat,
                                        StringBuilder lpDateStr,
                                        int sbSize)
        {

        }
```

در اینجا برای تغییر فرمت تاریخ ویندوز تنها کافی است lpDateStr را مقدار دهی کنیم. ویندوز lpDate و سایر پارامترها را به این متد ارسال می‌کند؛ در اینجا فرصت خواهیم داشت تا بر اساس این اطلاعات، lpDateStr صحیحی را تولید و مقدار دهی کنیم.

د) نصب Hook نوشته شده

باید دقت داشت که هر دو برنامه نصاب Hook و همچنین کتابخانه Hook، باید دارای امضای دیجیتال باشند. بنابراین به برگه signing خواص پروژه مراجعه کرده و یک فایل snk را به هر دو پروژه اضافه نمائید.

سپس در برنامه نصاب، یک کلاس را با امضای ذیل تعریف کنید:

```
public class MessagesReceiverInterface : MarshalByRefObject
{
    public void Ping()
    {
    }
}
```

این کلاس با استفاده از امکانات Remoting دات نت، پیام‌های دریافتی از هوک دات نتی تزریق شده به درون یک پروسه دیگر را دریافت می‌کند.

سپس در ابتدای برنامه نصاب، یک کانال Remoting باز شده (که آرگومان جنریک آن دقیقا همین نام کلاس MessagesReceiverInterface فوق را دریافت می‌کند)

```
var channel = RemoteHooking.IpcCreateServer<MessagesReceiverInterface>(ref _channelName,
WellKnownObjectMode.SingleCall);
```

و سپس توسط متد RemoteHooking.Inject کار تزریق ExplorerPCal.Hooks.dll به درون پروسه اکسپلورر ویندوز انجام می‌شود:

```
RemoteHooking.Inject(
  explorer.Id,
  InjectionOptions.Default | InjectionOptions.DoNotRequireStrongName,
  "ExplorerPCal.Hooks.dll", // 32-bit version (the same, because of using AnyCPU)
  "ExplorerPCal.Hooks.dll", // 64-bit version (the same, because of using AnyCPU)
  _channelName
);
```

پارامتر اول متد RemoteHooking.Inject، شماره PID یک پروسه است. این شماره را از طریق متد Process.GetProcesses می‌توان بدست آورد. سپس یک سری پیش فرض مشخص می‌شوند و در ادامه مسیر کامل دو DLL هوک دات نتی باید مشخص شوند. چون تنظیمات پروژه هوک را بر روی Any CPU قرار داده‌ایم، فقط کافی است یک نام DLL را برای هوک‌های 64 بیتی و 32 بیتی ذکر کنیم.

پارامتر و پارامترهای بعدی، اطلاعاتی هستند که به سازنده کلاس هوک ارسال می‌شوند. بنابراین این سازنده می‌تواند تعداد پارامترهای متغیری داشته باشد:

```
.ctor(IContext, %ArgumentList%)
void Run(IContext, %ArgumentList%)
```

چند نکته تکمیلی مهم برای کار با کتابخانه Easy hook

- کتابخانه easy hook فعلا با ویندوز 8 سازگار نیست.

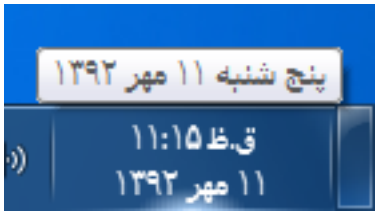
- برای توزیع هوک‌های خود باید تمام فایل‌های همراه کتابخانه easy hook را نیز توزیع کنید و فقط به چند DLL ابتدایی آن بسنده نباید کرد.
- اگر هوک شما بلافاصله سبب کرش پروسه هدف شد، یعنی امضای تابع API شما ایراد دارد و نیاز است چندین و سایت را جهت یافتن امضایی صحیح بررسی کنید. برای مثال در امضای عمومی متد GetDateFormatW، پارامتر SystemTime به صورت struct تعریف شده است؛ درحالیکه ویندوز ممکن است برای دریافت زمان جاری به این پارامتر نال ارسال کند. اما struct دات نت برخلاف struct زبان C یک value type است و نال پذیر نیست. به همین جهت کلیه امضاهای عمومی که در مورد این متد در اینترنت یافت می‌شوند، در عمل غلط هستند و باید SystemTime را یک کلاس دات نتی که Refrence type است، تعریف کرد تا نال پذیر شود و hook کرش نکرده یا اشتباه عمل نکند.
- زمانیکه یک هوک easy hook بر روی پروسه هدف نصب می‌شود، دیگر قابل unload کامل نیست و نیاز است برای کارهای برنامه نویسی و به روز رسانی فایل dll جدید، پروسه هدف را خاتمه داد.
- متد Run هوک باید همیشه در حال اجرا باشد تا توسط CLR بلافاصله خاتمه نیافته و هوک از حافظه خارج نشود. اینکار را توسط روش ذیل انجام دهید:

```
try
            {
                while (true)
                {
                    Thread.Sleep(500);
                    _interface.Ping();
                }
            }
            catch
            {
                _interface = null;
                // .NET Remoting will raise an exception if host is unreachable

            }
```

تا زمانیکه برنامه نصاب هوک که توسط Remoting دات نت، کانالی را به این هوک گشوده است، باز است، حلقه فوق اجرا می‌شود. با بسته شدن برنامه نصاب، متد Ping دیگر قابل دستیابی نبوده و بلافاصله این حلقه خاتمه می‌یابد.
- استفاده همزمان از API Monitor ذکر شده در ابتدای بحث و یک هوک نصب شده، سبب کرش برنامه هدف خواهد شد.

سورس کامل این پروژه را در اینجا می‌توانید دریافت کنید
شمسی ساز تاریخ اکسپلورر ویندوز



نظرات خوانندگان

نویسنده: میثم هوشمند
تاریخ: ۱۳۹۲/۰۷/۱۲ ۰:۴۳

ممکن هست که یک مثال برای لاگ کردن درخواستها ارسالی به یک متد مانند همان متد gethostbyname ارائه نمایید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۱۲ ۱:۰

سورس پروژه قابل دریافت هست و به عنوان یک قالب برای این نوع کارها میشه ازش استفاده کرد.
برای مثال در این قالب جایی که هوک تعریف میشه به این صورت تغییر خواهد کرد:

```
_getHostByNameHook = LocalHook.Create(
                                        InTargetProc: LocalHook.GetProcAddress("ws2_32.dll",
"gethostbyname"),
                                        InNewProc: new GetHostByNameDelegate(getHostByNameHooked),
                                        InCallback: this);
```

delegate آن چنین تعریفی خواهد داشت (توضیحات آن در متن فوق هست؛ قسمت (ج) نوشتن هوک):

```
[UnmanagedFunctionPointer(CallingConvention.StdCall, CharSet = CharSet.Ansi, SetLastError = false)]
        private delegate IntPtr GetHostByNameDelegate(string name);
```

و جایی که نهایتا درخواست‌های DNS دریافت می‌شوند به صورت زیر تعریف خواهد شد:

```
private static IntPtr getHostByNameHooked(string name)
        {
            // redirect ...
            //if (name.StartsWith("www.google.com"))
            //{
            //    return Native.GetHostByName("127.0.0.1");
            //}

            return Native.gethostbyname(name);
        }
```

در اینجا می‌شود درخواست‌های DNS را تغییر جهت داد (مثلا گوگل را بلاک کرد)، یا همان تابع اصلی ویندوز را اجرا کرد و یا name دریافتی را در یک فایل مثلا لاگ کرد.

نویسنده: علی قمشلویی
تاریخ: ۱۳۹۲/۰۷/۱۲ ۱۰:۳۵

با سلام و تشکر
این آموزش یکی ناب‌ترین آموزش‌ها می‌باشد چون من مدتی با easyhook کار کردم و کمبود آموزش‌های فارسی در رابطه با هوک بسیار چشمگیربود .
در رابطه با تغییر تاریخ در شیرپوینت شما چه پیشنهادی دارید؟ چون تحقیقات بنده در این رابطه به جایی نرسید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۱۲ ۱۱:۴۴

من روی تاریخ شیرپوینت وقت نگذاشتم ولی به نظر « OnetUtil.dll » رو باید مدنظر داشته باشید.

نویسنده: محبوبه محمدی

تاریخ: ۱۳۹۲/۰۷/۱۳ ۱۰:۲۳

سلام.ممنون بابت مطلب خوبتون.
Hooking یکی از مفاهیم جالب و خیلی کاربردی هست.یکی از استفاده‌های خوبش زمانیه که یکی از دکمه‌های پر استفاده کیبوردتون خراب شده و امکان تعمیر سخت افزاریش وجود نداره.به جای عوض کردن کیبورد مخصوصا وقتی لپ تاپ باشه،به راحتی می‌تونید کیبوردتون رو Hook کنید و یکی از کلیدهای کم مصرف‌تر رو جایگزین کلید خراب کنید!

نویسنده: شمس
تاریخ: ۱۳۹۲/۰۷/۱۴ ۷:۵۸

بسیار جالب بود.
خاطرات گذشته را زنده کرد.

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۰۷/۱۶ ۹:۱۹

سلام و تشکر
یادمه قبلا گفته بودین توی یکی از مطالبتون اینجا که من از این روش برای دستکاری تقویم استفاده نمی‌کنم و ایراد داره. خواستم بدونم که الان هم همون نظر رو دارین و این یه مثال هست برای کار با هوک، یا اینکه فکر میکنین این روش خوبی برای فارسی سازی تقویم هست و نظرتون عوض شده؟

نویسنده: محمدرضا دستوری
تاریخ: ۱۳۹۲/۰۷/۱۶ ۹:۲۲

تا به حال همچنین توضیح کاملی و خوبی به همراه یک مثال عالی هیچ جا ندیده بودم و میشه گفت روان‌ترین توضیح هوک رو ارائه دادید
من هم علاقه‌مند شدم یک سری از نرم افزارهای روزمره رو به این ترتیب بومی کنم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۱۶ ۱۰:۰

بله. این یک مثال کلی هم هست.
+
- خوب؛ این کار فوق رایگان هست! (و همچنین سورس باز؛ حداقل می‌تونید بررسی کنید که آیا فقط تاریخ پروسه اکسپلورر را هوک کرده یا جاهای دیگری رو هم تحت نظر قرار داده)
- Global hook یا System wide hook نیست. کارهایی که قبلا انجام شدن، کل سیستم رو تحت نظر قرار می‌دن. کار فوق، فقط و فقط پروسه explorer ویندوز رو تحت نظر قرار می‌ده. به همین جهت تاثیر خاصی روی سایر پروسه‌های سیستم نداره. همچنین چون کل سیستم رو مونیتور نمی‌کنه، روی کارآیی کلی آن هم تاثیر منفی نخواهد گذاشت. به علاوه روش فوق ساختارهای درونی تاریخ سیستم رو دستکاری نمی‌کنه. فقط رشته نهایی نمایشی (آرگومان‌های lpDateStr یا lpTimeStr) را شمسی می‌کنه.

نویسنده: هیمن روحانی
تاریخ: ۱۳۹۲/۰۷/۱۶ ۱۲:۲

سلام ممنمون از مطلب خوبتون. شما در مورد انجام این کار در دات نت این رو نوشتید (در دنیای دات نت با استفاده از امکانات Reflection قابل انجام است و یا حتی بازنویسی اسمبلی‌ها و افزودن کدهای IL مورد نیاز به آن‌ها که به آن IL Weaving هم گفته می‌شود.). من می‌خوام اسمبلی اصلی شیرپوینت رو در حین اجرا تغییر بدم ولی Reflection سربار زیادی داره و نمی‌خوام اصل اسمبلی رو تغییر بدم. آیا کتابخانه یا فریم ورکی رو سراغ دارید‌که بشه کدهای دات نت رو در حین اجرا تغییر بده؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۷/۱۶ ۱۳:۴۰

CInject - Code Inject & Runtime Intelligence

Rewrite MSIL Code on the Fly with the .NET Framework Profiling API

نویسنده: هیمن روحانی
تاریخ: ۱۳۹۲/۰۷/۱۶ ۱۴:۲۳

Cinject یک اسبلی جدید براتون می‌سازه. ابزارهای دیگه‌ی هستند که این کارو انجام میدن مثل Reflexil که افزونه Reflector است و ساده‌تر میشه باهاش کار کرد. اکثرا این ابزارها از کتابخانه Mono.Cecil استفاده میکنن که کارش بازنویسی MSIL است.

برای استفاده از .NET Profiling API باید بتونید با c++ اینترفیس ICorProfilerCallback2 رو پیاده سازی کنید و یک پروفایلر بسازید که من نتونستم انجامش بدم. این مورد واقعا عالی و مزایا زیادی داره ولی متاسفانه پیاده سازیش سخته.
دنبال ابزار یا کتابخانه‌ی هستم که از .NET Profiling API استفاده کرده باشه و این روشو ساده کرده باشه.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۷/۱۶ ۱۶:۳۲

برای یک برنامه وب روش‌های ساده‌تری هم برای نمایش یا دستکاری اطلاعات روی صفحه هست. مثلا مانند مطلب تغییر تاریخ وبلاگ‌های بلاگر در همین سایت. یا حتی میشه اینکار رو با افزونه‌های مرورگرها نیز انجام داد. اتفاقا خیلی هم مرسوم است این روش. این هم یک مثال خاصش در مورد شیرپوینت .

نویسنده: هیمن روحانی
تاریخ: ۱۳۹۲/۰۷/۱۷ ۱۰:۶

همه این روش هارو دیدم که بعضی هاشون از جاوااسکریپت هم استفاده میکنند. ولی این روشها مشکلات زیادی دارن. شیرپوینت فقط یک سایت ساده نیست، Object Model داره کلی سرویس داره، فرم اینوپس و... . شیرپوینت میتونه همزمان از چند تا تقویم پشتیبانی بکنه. این روش‌ها خیلی سرسری مشکلو حل میکنن درکل نمیشه با این روشها یک راه کار درست حسابی ارائه داد.
اگه بشه از .NET Profiling API استفاده کنم میتونم تقویم Exchange Outlook Web App رو هم شمسی کنم.

نویسنده: هیمن روحانی
تاریخ: ۱۳۹۲/۰۷/۱۸ ۱۰:۷

خسته نباشد. این API Monitor می‌تونه برنامه‌های دات نت رو مانیتور کنه؟ یا فقط برای کدهای مدیریت نشده کار می‌کنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۱۸ ۱۰:۲۸

با برنامه‌های دات نت هم کار می‌کنه. مثلا متد new WebClient().DownloadData را در برنامه دات نتی خودتون فراخوانی کنید. بعد در برنامه API Monitor تیک قسمت مربوط به شبکه و اینترنت را قرار دهید (تصویر اول مقاله فوق). تمام فراخوانی‌های شبکه برنامه را مونیتور می‌کند.
ضمنا برنامه API Monitor قابل بسط است. یعنی اگر به پوشه API آن مراجعه کنید یک سری فایل XML در آن قرار دارد که تعاریف توابع DLLهای مختلف در آن ارائه شده‌اند. اگر تعریفی یا DLL ایی در آن نیست، قابل افزودن است. یا حتی اگر خودتان نمی‌توانید اینکار را انجام دهید، فایل هدر آن‌را در انجمن این برنامه ارسال کنید تا به شما کمک کنند.

نویسنده: هیمن روحانی
تاریخ: ۱۳۹۲/۰۷/۲۰ ۱۰:۴۸

در مقدمه این مطلب به تزریق AOP اشاره کردید و فرمودید Hookها یا قلاب‌ها دقیقا کار Interception دنیای AOP را انجام می‌دهند ولی در AOP همه چیز از مدیریت وهله سازی شروع میشه واگر امکان مدیریت وهله سازی نباشه نمیشه AOP امکان پذیر نیست. در AOP انجام تغییر در کدها فقط در اسمبلی هایی که در پروسه جاری لود شده اند امکان پذیره. اما در Hooking شما پروسه‌های دیگر رو مانیتور و زمان احتیاج نتیجه برگشتی یک تابعو تغییر میدید. قبلا در همین مطلب این سوالو پرسیدم: *آیا کتابخانه یا فریم ورکی رو سراغ دارید‌که بشه کدهای دات نت رو در حین اجرا تغییر بده*؟ بهتره سوالمو اینجوری اصلاح کنم: آیا کتابخانه یا فریم ورکی رو سراغ دارید‌که بتونه کدهای دات نت رو در پروسه‌ی دیگر تغییر بده؟ آیا با EasyHook میشه همین کارو برای برنامه‌های دات نت در حال اجرا انجام داد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۲۰ ۱۱:۱۱

- با easy hook میشه JIT Compiler دات نت رو هوک کرد . در اینجا فرصت خواهید داشت تا بدنه متد رو با کدهای IL بازنویسی کنید. در مورد JIT Hooking اگر اطلاعات بیشتری خواستید، اینجا
- CLR Injection: Runtime Method Replacer
- .NET CLR Injection: Modify IL Code during Run-time.
- Hawkeye - The .Net Runtime Object Editor
- کار خود مایکروسافت است: Moles - Replace any .NET method with a delegate البته مایکروسافت برای کارهای Native ویندوز هم یک کتابخانه به نام detours دارد. نسخه 32 بیتی آن عمومی است و با C سازگار است.
- Modifying IL at runtime
- WPF Snoop

نویسنده: reza
تاریخ: ۱۳۹۲/۰۸/۲۶ ۱۱:۰

سلام؛ میخوام توی برنامه‌های مختلفی(تمام برنامه ها) که روی ویندوز نصب است وقتی یک کلید میانبری زدم مثل alt+shift+r متنی که در حال تایپش هستم یا متنی که داخل textbox ی تایپ کردم، روش تغییراتی انجام بدم چه کاری باید انجام بدم؟ ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۲۶ ۱۲:۱۲

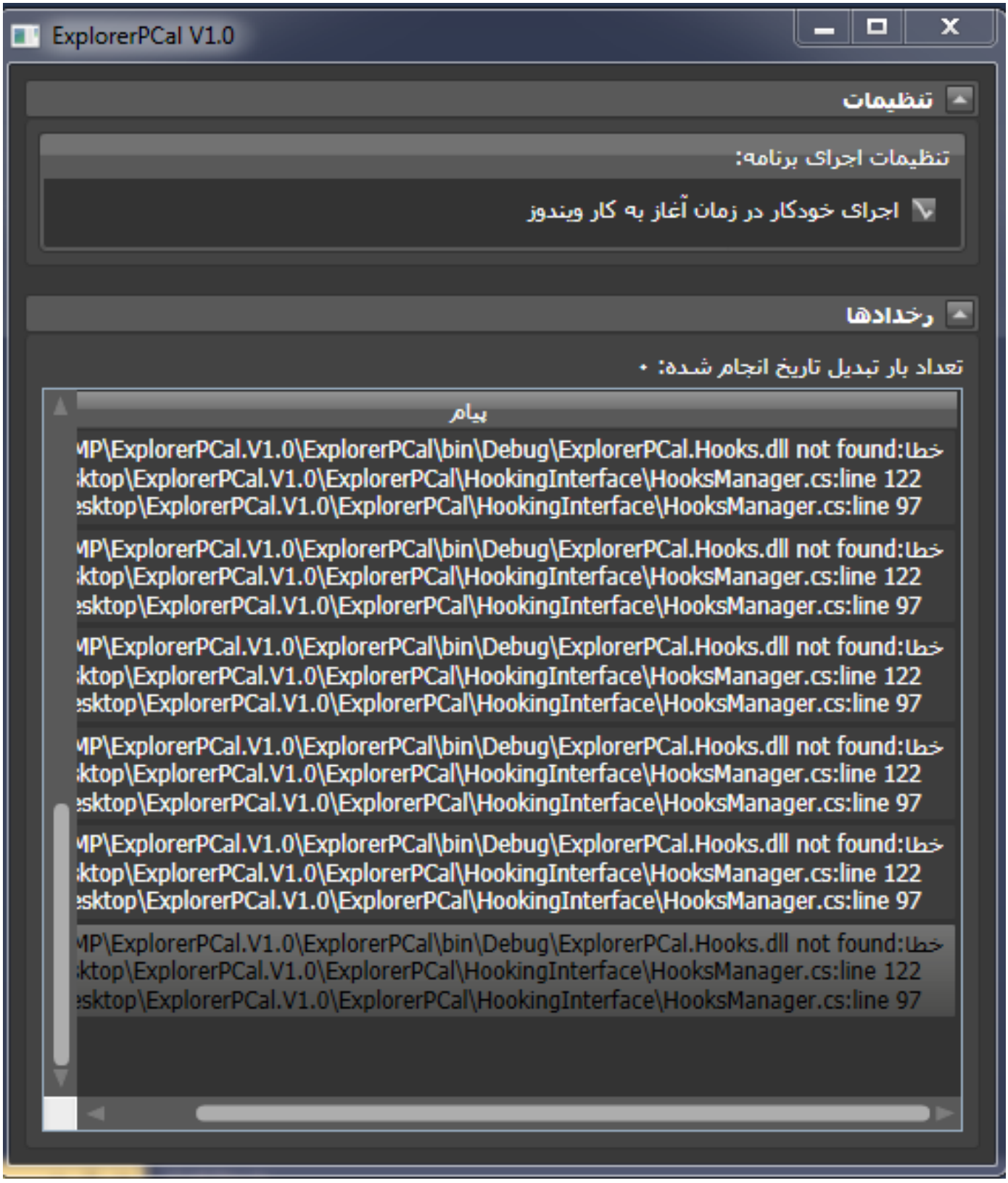
- easyhook کار global hooking را انجام نمی‌دهد؛ به ازای یک یا چند پروسه مشخص کار می‌کند.
- برای تغییر یک متن در حال نمایش در ویندوز عموما از هوک کردن متد ExtTextOutW استفاده می‌شود. خروجی این متد را در ویندوز XP راحت می‌شود تغییر داد. در ویندوز 7 به علت پیشرفته شدن خروجی‌های متنی و پشتیبانی از انواع و اقسام فرهنگ‌های مختلف، خروجی آن به صورت گلیف است و نه متن. یعنی به مقدار پارامتر fuOptions آن باید دقت داشت و اگر مساوی ETO_GLYPH_INDEX بود، تا جایی که تحقیق کردم قابل برگشت به متن یونیکد اصلی آن نیست (یا کار ساده‌ای نیست این پردازش). (قبل از اینکه از متد دریافت تاریخ استفاده کنم، سعی کردم از این روش کمک بگیرم و روی ویندوز XP جواب داد اما روی ویندوز 7 کار نکرد)

نویسنده: علیرضا صبوری
تاریخ: ۱۳۹۲/۱۱/۱۳ ۲:۲۲

سلام
با توجه به اینکه استفاده از easyhook در ویندوز 8 پشتیبانی نمیشه هنوز راهکاری جایگزین برای انجام این اعمال در ویندوزهای جدید معرفی نشده ؟

نویسنده: خیام
تاریخ: ۱۳۹۲/۱۲/۲۲ ۱۵:۱۶

وقتی برنامه رو اجرا میکنم با این خطا مواجه میشم (از ویندوز 7 استفاده میکنم)



نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۲ ۱۶:۱۵

- نگارش 1.1 آن‌را از اینجا دریافت کنید.

- لطفا برای گزارش خطا از قسمت مخصوص بازخوردهای پروژه آن در سایت استفاده کنید.
همچنین در این حالت فایل *ErrosLog.Log* آن‌را هم فراموش نکنید (ریز خطاها در آن ثبت می‌شوند).

اگر پروژه را خودتان کامپایل کرده‌اید (که اینطور به نظر می‌رسد با توجه به پوشه debug)، برنامه اجرا نمی‌شود چون تمام فایل‌های exe و dll همراه easy hook را برای اجرا نیاز دارد و این‌ها باید کنار فایل اجرایی اصلی برنامه همانند بسته‌ای که برای دریافت در سایت قرار گرفته، کپی شوند.
این نکته در متن هم ذکر شده؛ قسمت «چند نکته تکمیلی مهم برای کار با کتابخانه Easy hook» انتهای بحث: «برای توزیع هوک‌های خود باید تمام فایل‌های همراه کتابخانه easy hook را نیز توزیع کنید و فقط به چند DLL ابتدایی آن بسنده نباید کرد»

نویسنده: دریا69
تاریخ: ۱۳۹۳/۰۳/۲۷ ۸:۴۷

با سلام و تشکر بابت آموزش خوبتون
ببخشید من یه سوال در مورد خود api monitor دارم.
اینکه بعد از اجرای یک فایل و بدست آوردن فراخوانی ها، برای اینکه فراخوانی‌ها رو به بردار ویژگی تبدیل کنم و برای کلاس بندی ازشون استفاده کنم باید در فایل xml یا اکسل بریزم ولی وقتی میریزم ساختار سلسله مراتبیش رو دیگه نمایش نمیده. میخواسم ببینم چیکار باید بیکنم که موقع کپی کردن در فایل متنی ساختار سلسله مراتبی و یا اینکه کدوم فراخوانی زیرمجموعه دیگری هست حفظ بشه؟
سوال دیگم اینه که چجوری با استفاده از این نرم افزار و بدون چک کردن تک تک فراخوانی‌ها و پارامتراشون به صورت جداگانه، میتونیم بفهمیم بعد از اجرای یک فایل پارامتر کدام یک از فراخوانیامون دچاد تغییر شدن؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۲۷ ۱۰:۱

اگر این برنامه‌ی ثالث قابلیت خاصی را ندارد، در انجمن پشتیبانی آن مطرح کنید تا اضافه شود.