عنوان: **ارسال ترافیک وب یک برنامه‌ی خاص به یک پروکسی سرور به کمک FiddlerCore**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۲۶ ۱۲:۱۰
آدرس: www.dotnettips.info
گروه‌ها: Fiddler, FiddlerCore

خیلی از برنامه‌ها به صورت پیش‌فرض تنظیمات پروکسی خاصی را درنظر نگرفته‌اند. در شبکه‌های داخلی شرکت‌ها هم معمولا اینترنت از طریق پروکسی سرورهایی مانند ISA Server ویندوزی و یا Squid لینوکسی، بین کاربران توزیع می‌شود.

**سؤال: چطور می‌شود برنامه‌ای را که تنظیمات پروکسی ندارد، پروکسی خور کرد؟!**

روشی که با سطح دسترسی معمولی و بدون نیاز به درایورهای خاص بررسی پکت‌های TCP و UDP سیستم و همچنین توسط دات نت فریم ورک قابل استفاده باشد، استفاده از کتابخانه‌ی معظم FiddlerCore است. برنامه‌ی Fiddler توسط یکی از کارکنان سابق مایکروسافت و عضو پیشین تیم IE تهیه شده‌است. کار اصلی این برنامه، دیباگ درخواست‌های HTTP/HTTPS، FTP و امثال آن است. هسته‌ی اصلی آن نیز به صورت یک کتابخانه‌ی مجزا به نام FiddlerCore در اختیار برنامه نویس‌های دات نت است. این برنامه اخیرا توسط شرکت تلریک پشتیبانی و تملک شده‌است.
کتابخانه‌ی FiddlerCore و برنامه‌ی Fiddler را از اینجا می‌توانید دریافت کنید. (اگر سایت آن باز نمی‌شود به این علت است که هاستینگ شرکت تلریک IPهای ایرانی را بسته است)

**اسکلت اصلی یک برنامه‌ی مبتنی بر FiddlerCore**

```
using System;
using System.Net;
using System.Threading;
using Fiddler;
using System.Net.Security;

namespace FiddlerTest
{
    class Program
    {
        static void beforeRequest(Session oSession)
        {
        }

        static void Main(string[] args)
        {
            try
            {
                startFiddlerApplication();

                Console.WriteLine("FiddlerCore started on port " +
FiddlerApplication.oProxy.ListenPort);
                Console.WriteLine("Press any key to exit");
                Console.ReadKey();
            }
            finally
            {
                shutdownFiddlerApplication();
            }
        }

        static void onLogString(object sender, LogEventArgs e)
        {
            Console.WriteLine("** LogString: " + e.LogString);
        }

        static void onNotification(object sender, NotificationEventArgs e)
        {
            Console.WriteLine("** NotifyUser: " + e.NotifyString);
        }

        static void onValidateServerCertificate(object sender, ValidateServerCertificateEventArgs e)
        {
            if (SslPolicyErrors.None == e.CertificatePolicyErrors)
                return;
```

```
            Console.WriteLine("invalid certificate: {0}", e.ServerCertificate.Subject);

            e.ValidityState = CertificateValidity.ForceValid;
        }

        static void shutdownFiddlerApplication()
        {
            FiddlerApplication.OnNotification -= onNotification;
            FiddlerApplication.Log.OnLogString -= onLogString;
            FiddlerApplication.BeforeRequest -= beforeRequest;
            FiddlerApplication.OnValidateServerCertificate -= onValidateServerCertificate;

            FiddlerApplication.oProxy.Detach();
            FiddlerApplication.Shutdown();

            Thread.Sleep(500);
        }

        private static void startFiddlerApplication()
        {
            FiddlerApplication.OnNotification += onNotification;
            FiddlerApplication.Log.OnLogString += onLogString;
            FiddlerApplication.BeforeRequest += beforeRequest;
            FiddlerApplication.OnValidateServerCertificate += onValidateServerCertificate;

            FiddlerApplication.Startup(5656,
                FiddlerCoreStartupFlags.RegisterAsSystemProxy |
                FiddlerCoreStartupFlags.MonitorAllConnections |
                FiddlerCoreStartupFlags.CaptureFTP); // proxy server on 5656
        }
    }
}
```

اسکلت کلی یک برنامه‌ی مبتنی بر FiddlerCore را در اینجامشاهده می‌کنید. در متد startFiddlerApplication کار برپایی پروکسی آن صورت می‌گیرد. همچنین یک سری Callback نیز در اینجا قابل تنظیم هستند. برای مثال پیام‌ها و اخطارهای داخلی FiddlerCore را می‌توان دریافت کرد و یا توسط روال رخدادگردان BeforeRequest می‌توان کار تحت کنترل قرار دادن یک درخواست را انجام داد. به همین جهت است که به این برنامه و کتابخانه، Web debugger نیز گفته می‌شود. متد BeforeRequest دقیقا جایی است که می‌توانید روی یک درخواست صادر شده توسط مرورگر، break point قرار دهید.
در متد FiddlerApplication.Startup روی پورتی مشخص، کار تنظیم پروکسی فیدلر انجام می‌شود. سپس مشخص می‌کنیم که چه مواردی را باید تحت نظر قرار دهد. با تنظیمات RegisterAsSystemProxy و MonitorAllConnections فیدلر قادر خواهد بود ترافیک وب اکثر برنامه‌های ویندوزی را مونیتور و دیباگ کند.
در متد shutdownFiddlerApplication نیز روال‌های رخدادگردان، آزاد شده و پروکسی آن خاموش می‌شود.

### هدایت درخواست‌های وب کلیه‌ی برنامه‌ها به یک پروکسی مشخص

```
static void beforeRequest(Session oSession)
{
  //send each request to the next proxy
  oSession["X-OverrideGateway"] = "socks=" + IPAddress.Loopback + ":" + 2002; //socks on 2002
}
```

در اینجا شیء oSession، حاوی اطلاعات کامل درخواست در حال بررسی است. توسط آن می‌توان با استفاده از تنظیم خاصی به نام X-OverrideGateway، به فیدلر اعلام کرد که درخواست رسیده را به پروکسی سرور دیگری منتقل کن. تنها کاری که باید صورت گیرد ذکر IP و پورت این پروکسی سرور است. اگر نوع آن سرور، ساکس باشد به ابتدای رشته یاد شده باید یک =socks، نیز اضافه شود.

### هدایت درخواست‌های تنها یک برنامه‌ی خاص به یک پروکسی مشخص

در متد beforeRequest، متغیر oSession.LocalProcessID مشخص کننده‌ی مقدار PID پروسه‌ای است که درخواست وب آن در حال بررسی است. برای بدست آوردن این PIDها در دات نت می‌توان از متد Process.GetProcesses استفاده کرد. Id هر پروسه،

همان LocalProcessID فیدلر است. بر این اساس می‌توان تنها یک پروسه‌ی مشخص را تحت نظر قرار داد و نه کل سیستم را.

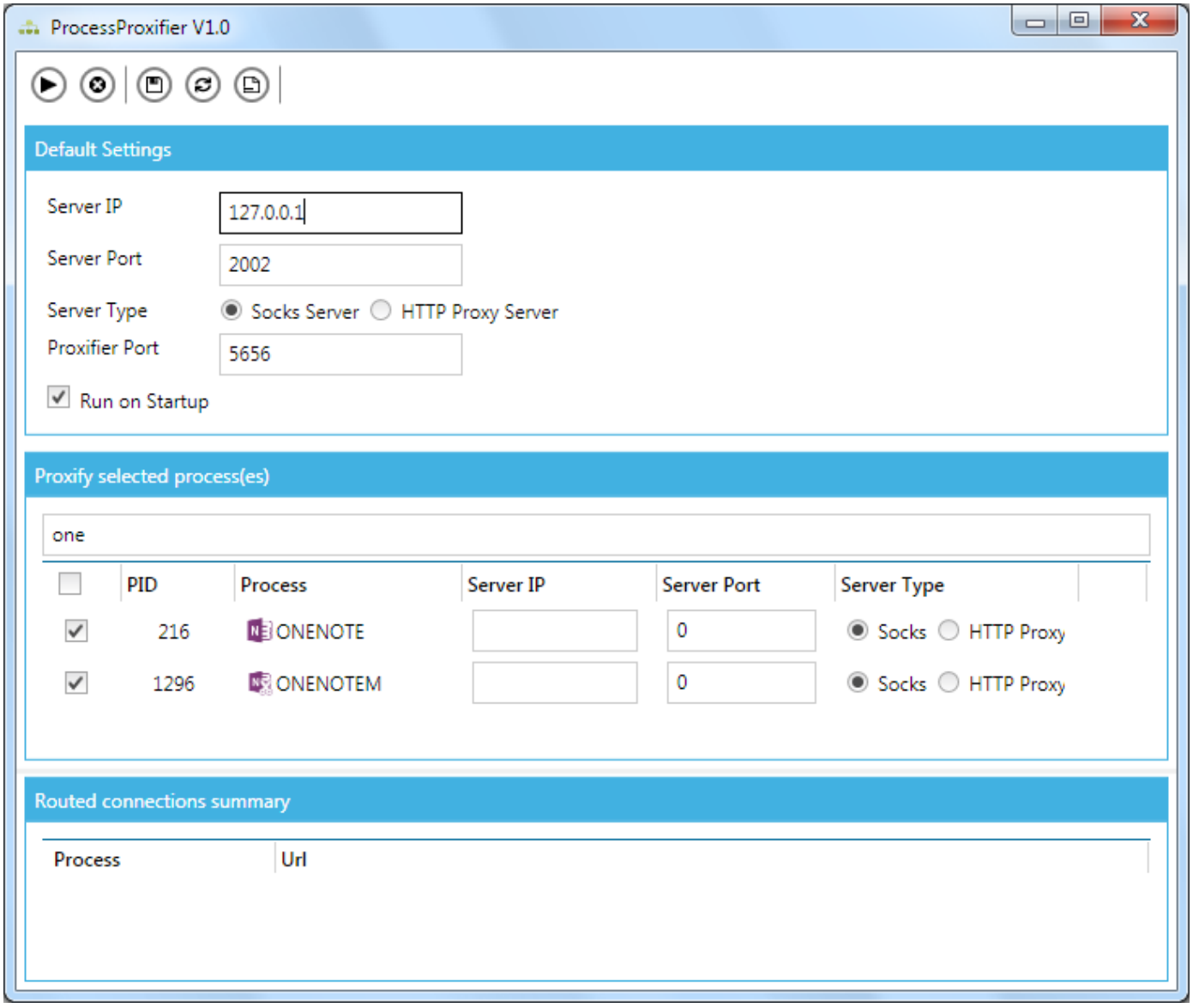
**کاربردها**

- فرض کنید برنامه‌ای تنظیمات پروکسی ندارد. با استفاده از روش فوق می‌توان برای آن پروکسی تعریف کرد.
- فرض کنید برنامه‌ای تنظیمات HTTP پروکسی دارد، اما پروکسی سرور شما از نوع ساکس است و نمی‌توان از این پروکسی سرور در برنامه‌ی مورد نظر استفاده کرد. X-OverrideGateway ذکر شده با هر دو نوع پروکسی‌های HTTP و Socks کار می‌کند.

اگر علاقمند به مطالعه‌ی اطلاعات بیشتری در مورد این کتابخانه هستید، کتاب 316 صفحه‌ای Debugging with Fiddler نویسنده‌ی اصلی آن، Eric Lawrence توصیه می‌شود.

**معرفی برنامه‌ی Process Proxifier**

اگر اطلاعات فوق را کنار هم قرار دهیم و یک GUI نیز برای آن طراحی کنیم، به برنامه‌ی Process Proxifier خواهیم رسید:




کار کردن با آن نیز بسیار ساده‌است. در قسمت تنظیمات پیش فرض برنامه، آدرس IP و پورت پروکسی سرور خود را وارد کنید.

نوع آن‌را نیز مشخص نمائید که Socks است یا از نوع HTTP Proxy.
سپس در لیست پروسه‌ها، مواردی را که لازم است از این پروکسی عبور کنند تیک بزنید. در اینجا می‌شود یا از تنظیمات پیش فرض استفاده کرد، یا می‌توان به ازای هر پروسه، از یک پروکسی مجزا با تنظیماتی که ذکر می‌کنید، کمک گرفت. اگر صرفا یک پروسه را انتخاب کنید و اطلاعاتی را وارد ننمائید، از اطلاعات پروکسی پیش فرض استفاده خواهد شد.

**دریافت سورس + باینری**

ProcessProxifier_V1.0.rar

## نظرات خوانندگان

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۲/۱۲/۰۸ ۱۷:۳۹

با سلام برای استفاده بعنوان سرور هم ممکنه راهنمایی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۸ ۱۷:۴۶

برای استفاده از FiddlerCore به عنوان سرور راه دور (نصب شده در یک سیستم دیگر) فقط کافی است به FiddlerCoreStartupFlags مورد FiddlerCoreStartupFlags.AllowRemoteClients را هم اضافه کنید.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۳/۰۶/۲۴ ۹:۲۰

بنظرم بخشی رو هم باید بابت درج نام کاربری و پسورد سرور پروکسی قرار داد چون در بسیاری از این شرکتها برای ارائه اینترنت داخل سازمان از نام کاربری و کلمه عبور استفاده میکنند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۲۴ ۹:۵۸

برای داخل دومین به این صورت قابل تنظیم است:

```
// OnBeforeRequest
oSession.oFlags["x-AutoAuth"] = @"domain\user:password";
```