目录

1 大概思路

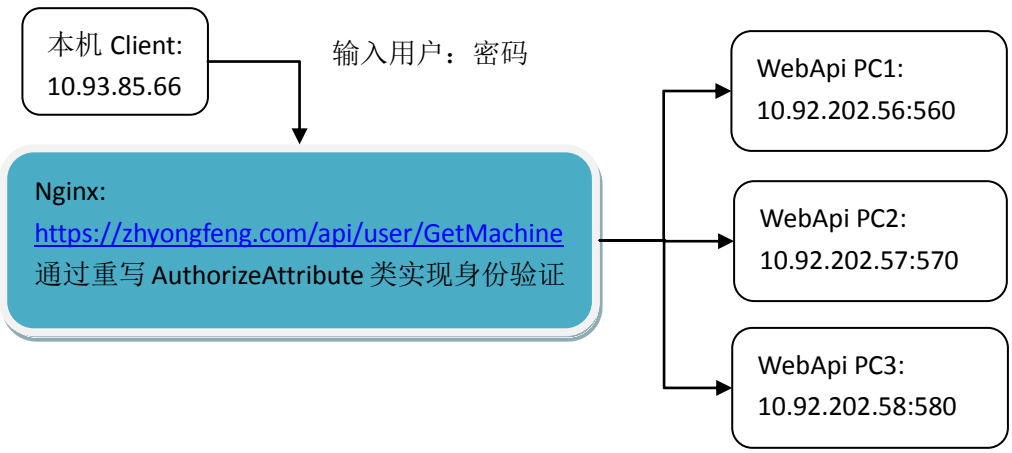
- Nginx 集群之 SSL 证书的 WebApi 身份验证
- AuthorizeAttribute 类
- Openssl 生成 SSL 证书
- 编写.NET WebApi
- 部署 WebApi 到局域网内 3 台 PC 机
- Nginx 集群配置搭建
- 运行结果
- 总结

2 Nginx 集群之 SSL 证书的 WebApi 身份验证

Nginx 集群可以实现基于 Http Basic 身份验证，通过输入用户、密码，经过 SSL 协议的 HTTPS，从而实现有效的身份验证并访问相应的 WebApi。当然，访问的方式不仅仅基于 Http Basic 一种，还可以通过令牌 token 的方式进行访问，又或者基于 redis 实现单点登录的访问，本文主要讲述的是基于 Http Basic 身份验证，并在 HTTPS 安全的通信下，实现简单集群身份验证。

以下是本文讲述的主要结构图：

客户端输入用户名密码，访问 Nginx 的 URL：https://zhyongfeng.com/api/user/GetMachine，然后 Nginx 进行负载均衡，返回 https 的响应。Nginx 集群之 SSL 证书的 WebApi 身份验证架构，如下图所示：



BASIC 认证的缺点

HTTP 基本认证的目标是提供简单的用户验证功能，其认证过程简单明了，适合于对安全性要求不高的系统或设备中，如大家所用路由器的配置页面的认证，几乎都采取了这种方式。其缺点是没有灵活可靠的认证策略，如无法提供域（domain 或 realm）认证功能，另外，BASE64 的加密强度非常低。当然，HTTP 基本认证系统也可以与 SSL 或者 Kerberos 结合，实现安全性能较高（相对）的认证系统。

3 AuthorizeAttribute 类

当你使用 AuthorizeAttribute 标记某个操作方法时，对该操作方法的访问将限于已经过身份验证且获得授权的用户。

如果使用该特性标记某个控制器，则该控制器中的所有操作方法均将受到限制。

Authorize 特性允许你指示将授权限于预定义角色或个别用户。这使你可以对谁有权查看站点上的任何页面进行严格控制。

如果未经授权的用户尝试访问使用 Authorize 特性标记的方法，MVC 框架将返回 401 HTTP 状态代码。

如果站点配置为使用 ASP.NET 窗体身份验证，则 401 状态代码会导致浏览器将用户重定向到登录页。

从 AuthorizeAttribute 派生，如果要从 AuthorizeAttribute 类派生，则派生的类型必须是线程安全的。

因此，不要在类型实例本身中（例如，在实例字段中）存储状态（除非该状态要应用于所有请求），而应在 Items 属性中按请求存储状态，该属性可通过传递给 AuthorizeAttribute 的上下文对象进行访问。

具体特性可以访问：

https://msdn.microsoft.com/zh-cn/library/system.web.mvc.authorizeattribute.aspx

4 Openssl 生成 SSL 证书

请参照《Nginx 集群之 SSL 证书的 WebApi 微服务》

http://www.cnblogs.com/yongfeng/p/7921905.html

5 编写.NET WebApi

服务器端：

CustomAuthorizeAttribute.cs

```
using System.Web.Http;
using System.Web.Http.Controllers;

namespace SSLWebApi.Controllers
{
    public class CustomAuthorizeAttribute : AuthorizeAttribute
    {
        public override void OnAuthorization(HttpActionContext actionContext)
        {
            //判断用户是否登录
            if (actionContext.Request.Headers.Authorization != null)
            {
```

```
                string userInfo = System.Text.Encoding.Default.GetString(System.Convert.FromBase64String(actionContext.Request.Headers.Authorization.Parameter));
                //用户验证逻辑
                if (string.Equals(userInfo, string.Format("{0}:{1}", "zhyongfeng", "123456")))
                {
                    IsAuthorized(actionContext);
                }
                else
                {
                    HandleUnauthorizedRequest(actionContext);
                }
            }
            else
            {
                HandleUnauthorizedRequest(actionContext);
            }
        }
        protected override void HandleUnauthorizedRequest(System.Web.Http.Controllers.HttpActionContext actionContext)
        {
            var challengeMessage = new System.Net.Http.HttpResponseMessage(System.Net.HttpStatusCode.Unauthorized);
            challengeMessage.Headers.Add("WWW-Authenticate", "Basic");
            throw new System.Web.Http.HttpResponseException(challengeMessage);
        }
    }
}
```

BaseController.cs

```
using System.Web.Http;

namespace SSLWebApi.Controllers
{
    /// <summary>
    /// BaseController 继承 BaseController 则需要身份验证
    /// </summary>
    [CustomAuthorize]
    public class BaseController : ApiController
    {
    }
}
```

UserController.cs

```
using System.Net;
using System.Web.Http;
```

```
namespace SSLWebApi.Controllers
{
    [RoutePrefix("api/User")]
    public class UserController : BaseController
    {
        /// <summary>
        ///  获取当前用户信息
        /// </summary>
        /// <param name="msg"></param>
        /// <returns></returns>
        [HttpPost]
        [Route("PostMessage")]
        public string PostMessage([FromBody]string msg)
        {
            return string.Format("当前输入的消息是:{0}", msg);
        }

        [Route("GetMachine")]
        public string GetMachine()
        {
            string AddressIP = string.Empty;
            foreach (IPAddress _IPAddress in Dns.GetHostEntry(Dns.GetHostName()).AddressList)
            {
                if (_IPAddress.AddressFamily.ToString() == "InterNetwork")
                {
                    AddressIP = _IPAddress.ToString();
                }
            }
            return string.Format("当前 WebApi 部署的 IP 是：{0}", AddressIP);
        }
    }
}
```

客户端：

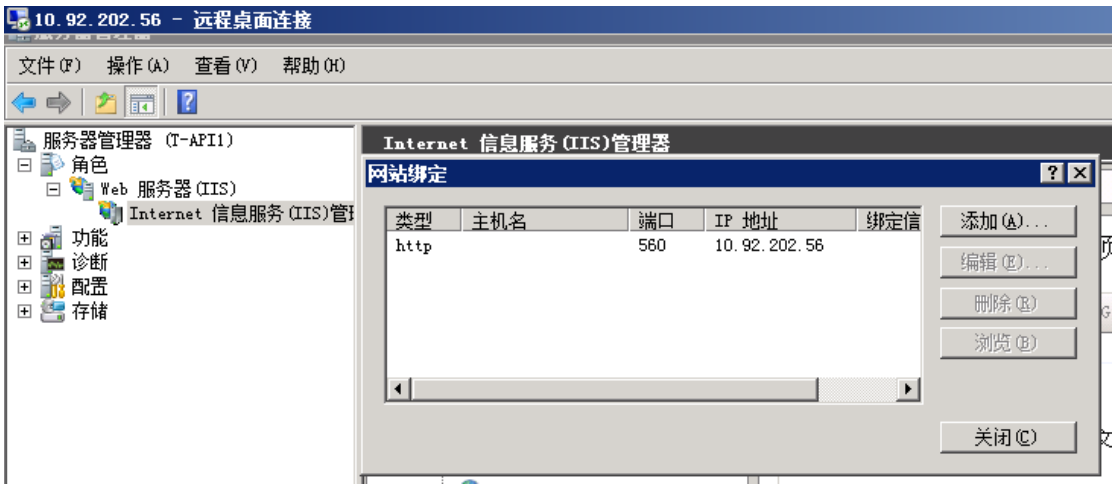
Program.cs

```
using System;
using System.IO;
using System.Net;
using System.Text;

namespace SSLWebApiClient
{
```

```
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 0; i < 10; i++)
            {
                //https 协议基本认证 Authorization
                string url = "https://zhyongfeng.com/api/user/GetMachine";
                ServicePointManager.ServerCertificateValidationCallback = delegate { return
true; };
                HttpWebRequest req = (HttpWebRequest)WebRequest.Create(url);
                NetworkCredential credential = new NetworkCredential("zhyongfeng",
"123456");
                req.Credentials = credential;
                HttpWebResponse response = (HttpWebResponse)req.GetResponse();
                Stream responseStream = response.GetResponseStream();
                StreamReader streamReader = new StreamReader(responseStream,
Encoding.UTF8);
                string html = streamReader.ReadToEnd();
                Console.WriteLine(html);
            }
            Console.Read();
        }
    }
}
```

6 部署 WebApi 到局域网内 3 台 PC 机

将 WebApi 部署到以下 10.92.202.56 的 3 台 PC 机



7 Nginx 集群配置搭建

通过自主义域名 zhyongfeng.com：80 端口进行负载均衡集群访问，则访问 C:\Windows\System32\drivers\etc\hosts，添加下列“本机 IP 自定义的域名”：

```
10.93.85.66    zhyongfeng.com
```

Nginx 的集群配置：

```
#user  nobody;
worker_processes  1;
events {
    worker_connections  1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;
    #server {
    #    listen       80;
    #    server_name  localhost;
    #    location / {
    #        root   html;
    #        index  index.html index.htm;
    #    }
    #    error_page   500 502 503 504  /50x.html;
    #    location = /50x.html {
    #        root   html;
    #    }
    #}

    upstream zhyongfeng.com {
        server    10.92.202.56:560;
        server    10.92.202.57:570;
        server    10.92.202.58:580;
    }
    server {
        listen       80;
        server_name  zhyongfeng.com;
        rewrite ^(.*)$  https://$host$1 permanent;
    }
    # HTTPS server
    #
    server {
```

```
        listen          443 ssl;
        server_name    zhyongfeng.com;
        ssl_certificate          server.crt;
        ssl_certificate_key    server_nopass.key;
    #       ssl_session_cache       shared:SSL:1m;
    #       ssl_session_timeout    5m;
    #       ssl_ciphers    HIGH:!aNULL:!MD5;
    #       ssl_prefer_server_ciphers    on;
        location / {
            proxy_pass      http://zhyongfeng.com;
        }
    }
}
```

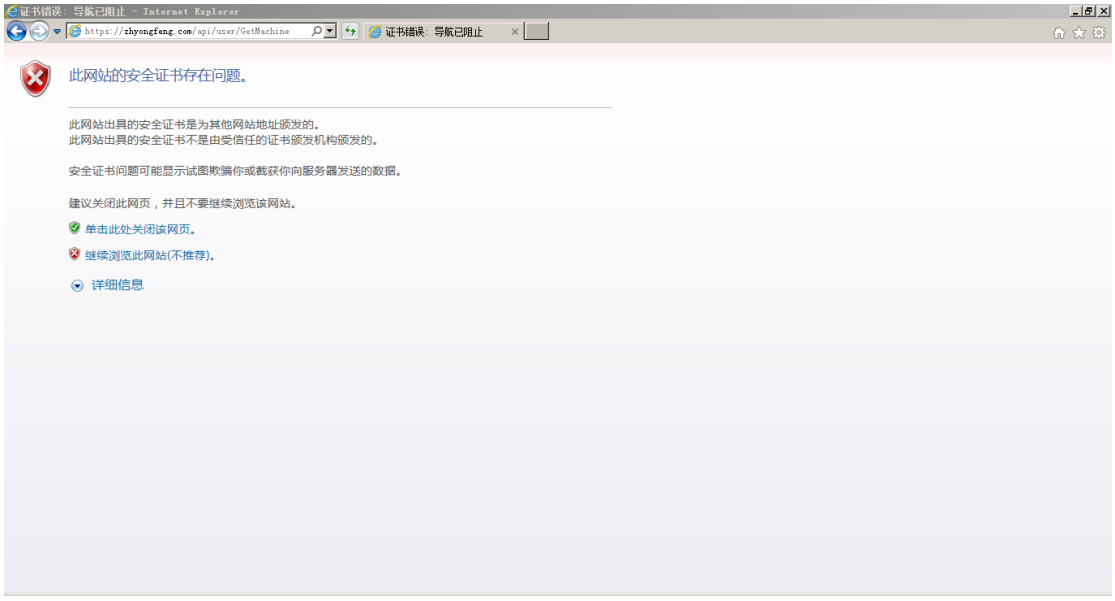
运行 CMD：

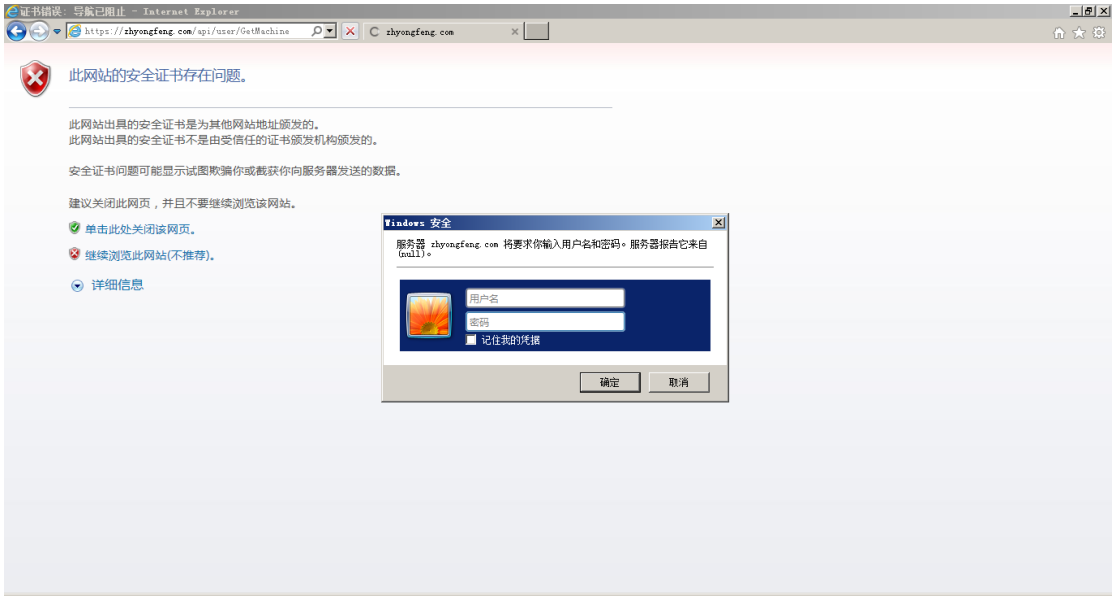
```
D:\DTLDownLoads\nginx-1.10.2>start nginx

D:\DTLDownLoads\nginx-1.10.2>nginx -s reload
```

8 运行结果

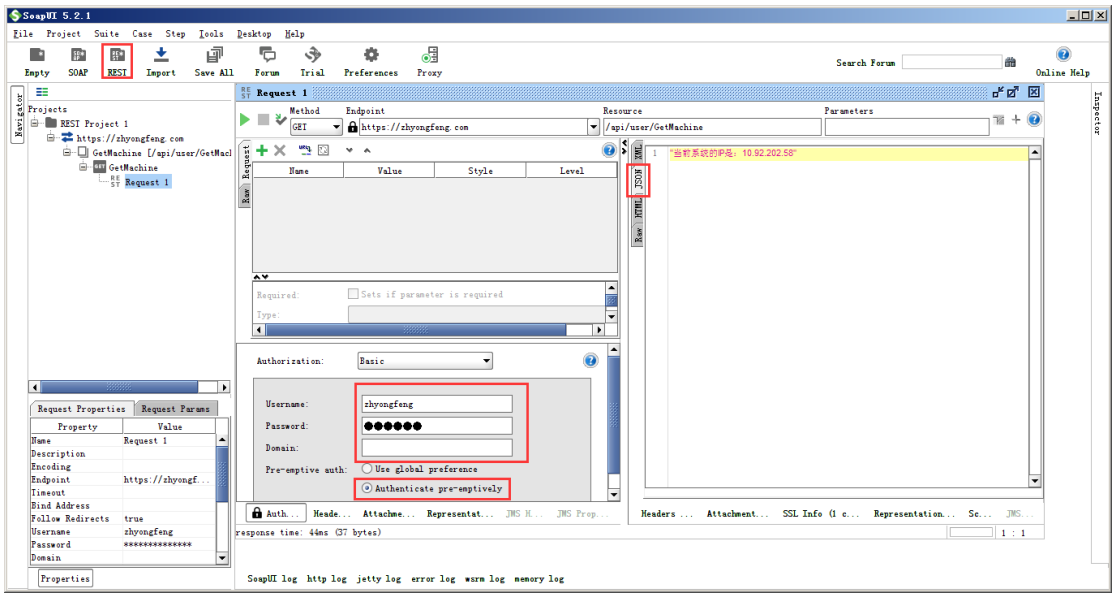
直接访问域名运行结果：

Hello, welcome to web api!

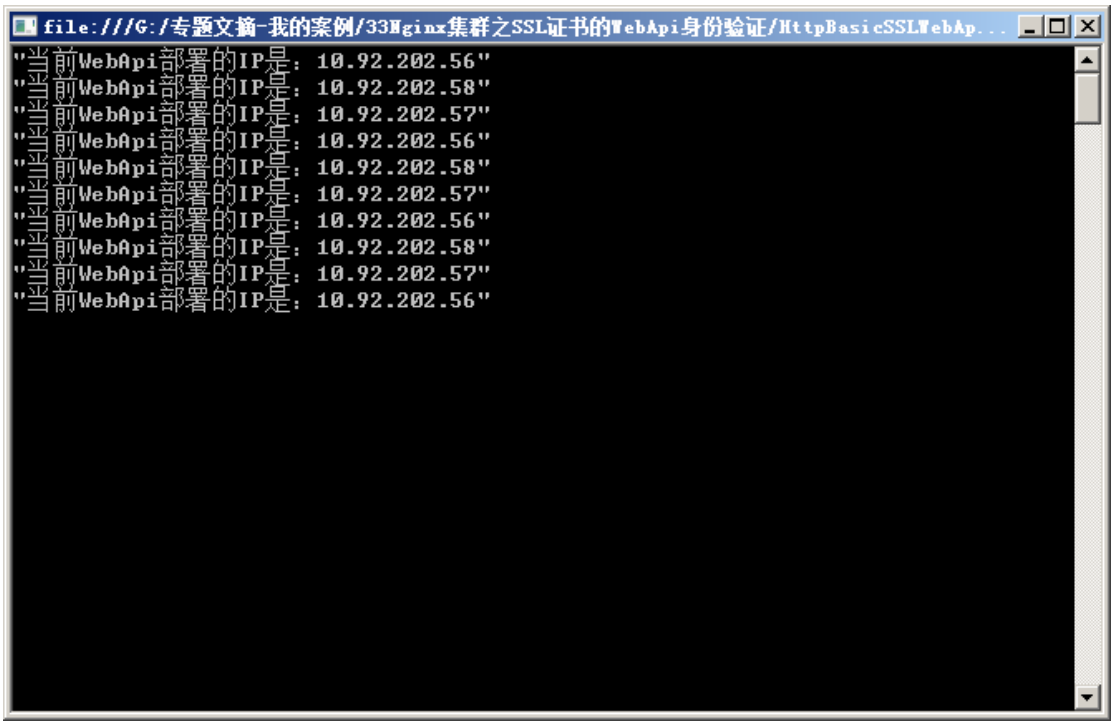




SOAP UI 调用运行结果：



启动客户端，通过身份验证运行结果如下：



9 总结

在 HTTP 协议进行通信的过程中，HTTP 协议定义了基本认证过程以允许 HTTP 服务器对 WEB 浏览器进行用户身份证的方法，当一个客户端向 HTTP 服务 器进行数据请求时，如果客户端未被认证，则 HTTP 服务器将通过基本认证过程对客户端的用户名及密码进行验证，以决定用户是否合法。客户端在接收到 HTTP 服务器的身份认证要求后，会提示用户输入用户名及密码，然后将用户名及密码以 BASE64 加密。

Nginx 基于 SSL 协议下，利用 http basic 身份验证，可以实现简单访问 WebApi，达到集群负载均衡的效果。通过简单的设计，在局域网上应用还是够用的。当然，身份认证方式有很多种，使用 redis、token 都是可以的。WebApi 基于 SSL 协议数据传输加密，结合 http basic 在一定上保证了通信的安全性。

源代码下载：
http://download.csdn.net/download/ruby_matlab/10141134

PDF 下载：
Nginx 集群之 SSL 证书的 WebApi 身份验证.pdf