# 简介

本系统主要的目的在于，在两台主机之间通过Socks5协议构建一个安全可靠、具有伪装机制的连接。在公司中信息安全是第一要素，因为新员工的身份HR很难完全掌握，万一其是其他公司派来的二五仔，则一些机密的通讯很可能会被侦听，也有可能对通讯工具进行攻击。因此，若将通讯工具的信道进行一定的伪装，则会很大程度上干扰二五仔的工作，并且若能提供类似于数据防火墙的功能，也是很吼的。

# 主要的通讯结构



其中，App是指使用Socks5协议的用本系统来进行通讯的程序，Client是指本系统运行在客户主机内的程序，Server是指运行在远端主机内的服务程序，Internet是指App要连接的目的主机。

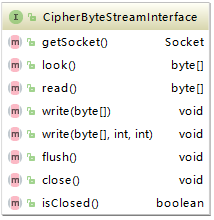
App通过Socks5协议来构建连接。App发出Socks5协议包；Client只负责将App发出的数据包进行加密，并发送给Server；Server解密数据包，并解析Socks5协议数据包。

# IO结构



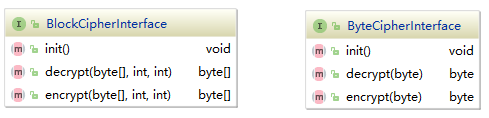
如图所示，假设Client监听的地址为localhost:2080，服务器监听的地址为1.2.3.4:2090，并且EncryptStream是socket.getOutputStream()的封装，DecryptStream是socket.getInputStream()的封装，二者一个用于加密，一个用于解密，上图的意思为Server中加密和解密可以用不同的算法和密钥。App将Socks5协议数据发送给localhost:2010，Client将数据进行加密后，发送给1.2.3.4:2020。Server收到数据后解密，并通过其EncryptedStream进行发送。最后，Server通过解析Socks5协议内容，构建与Internet的连接。

上图中EncryptStream和DecryptStream为逻辑上的IOStream，系统通过一个CipherByteStreamInterface的接口提供抽象。



其中look()是指解密之前的数据，用以监测是否是其他协议或者嗅探程序的接入。read()是指解密后的数据，两个write()是指加密数据。

本系统的加密和解密由2个接口提供抽象：

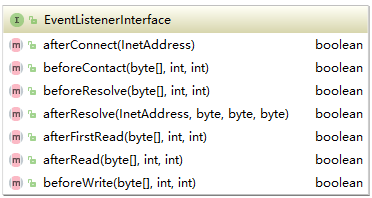


其中BlockCipherInterface负责App发送的数据的加密和解密，而ByteCipherInterface是负责控制字节的加密和解密，因为控制字段需要在App数据解密之前解密，而且需要固定的长度（不然则需要控制字段的控制字段），因此采用了按照字节加密的方式。因此，当CipherStream使用CipherInterface进行加密和解密时，可以将client的read()与server的write()设定为配对的密钥，write()与server的read()设定为配对的密钥。值得注意的是，系统传入的readKey和writeKey对于BlockCipher和ByteCipher是一样的，即他们拿到的writeKey是同一个，readKey也是同一个（因为我觉得没必要设置4个key），而是否是对称加密，取决于具体实现的方式。

# 伪装机制

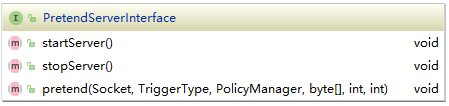
为了防止“别有用心”之人进行嗅探，本系统提供伪装机制。通过使用CipherStream中的look()方法，可以在进行解密之前查看收到的数据。以HTTP举例，正常收到的数据是无规则的数据（因为加密过了），所以当look()返回的数据是以“GET /index.html HTTP/1.1”开头的数据，则表示有人进行了HTTP的嗅探，因此需要将本系统伪装成HTTP服务器。

本系统通过以下几个类和接口实现该功能。



事件监听器，其用来监听所发生的事件和所收到的数据，比如HTTP&HTTPS可以在BeforeContact()中进行监听（已经有对应的默认实现，HttpListener和HttpsListener类），其他的某些事件与Socks5协议的握手方式有关：beforeResolve()是指解析Socks5请求之前，afterResolve()是指解析Socks5请求之后，其第一个参数是指App所请求的Internet地址。

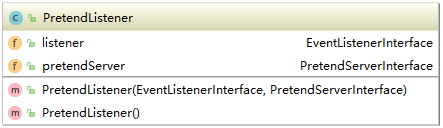
当事件监听器的方法返回值为true时，表示连接是非法的，需要伪装服务器，为其提供虚假的服务。



此为伪装服务器的接口，主要的方法是pretend()，参数的意义在接口注解中有详细的说明。连接被转到伪装服务器后，则由相应的服务器负责，一旦pretend()返回，则server直接关闭连接。

其中的PolicyManager是来负责全局的IP接入控制。如果我们想要对192.168.233.\*的IP段进行伪装，则可以通过其来完成。同时，因为pretend()方法中也有该对象，可以将当前pretend的IP也放入manager中，下次则不会被正常的服务解析。

上述两者构成了一个PretendListener。



系统通过一个List<PretendListener>来读取配置信息。

值得注意的是

1. 因为该服务是公共的（若放入PolicyManager中），需要保证线程安全。
2. 暂时没有实现伪装服务器的关闭机制。
3. 目前添加PretendListener的方式为在MainService中用pretendListenerList.add(…)；

# 其他

1. 为了简化伪装服务器的安装配置，该系统使用了CodeStory / fluent-http(<https://github.com/CodeStory/fluent-http>)作为服务器的配置库。
2. 目前没有GUI，请用IDE的run/debug的方式来运行，或者请大佬添加GUI。