Co:Z FTP-SSH Proxy

Co:Z FTP-SSH Proxy - User's Guide

V 1.0.4 Edition

Published September, 2015

Table of Contents

Revision History

Co:Z FTP-SSH Proxy 1.0.4 - September 8, 2015

Added support for MVS Modify (F) console commands under z/OS. This may be used to dynamically modify tracing. For example: **F jobname,APPL=TRACE=T**

Co:Z FTP-SSH Proxy 1.0.3 - September 4, 2015

Fixed support for host-specific properties. Updated JSCH to version 0.1.53. This fixes many bugs and potential security problems, but in rare conditions may cause key exchange problems with very old SSHD servers. These can often be worked around by selecting a specific KEX algorithm. See ftpsshproxy.properties for more information and examples.

Co:Z FTP-SSH Proxy 1.0.2 - August 24, 2015

Added the "-t" option for additional JSch (ssh) tracing. Masked passwords in log.

Co:Z FTP-SSH Proxy 1.0.1 - December 21, 2010

Fixed a memory leak in session management.

1. Introduction

Despite its popularity, the venerable FTP protocol can be difficult to implement securely[1]. With traditional FTP, userids, passwords, and data are sent in the clear over the network. FTPS (FTP over SSL or TLS) is one solution, but can be difficult to implement in the presence of some firewalls and NAT routers since FTP creates separate data connections to dynamically determined ports. SFTP, part of the SSH protocol stack, is an alternative but conversion of existing jobs and systems from FTP to SFTP can be prohibitive.

The Co:Z FTP-SSH Proxy is another alternative: it implements FTP over SSH without SFTP[2]. The existing FTP client and server remain unchanged except that the client is configured to use Co:Z FTP-SSH as a SOCKS5 proxy.[3] Co:Z FTP-SSH dynamically proxies FTP connections over SSH to an SSHD server running on the target host. The required client changes can be made without affecting existing FTP jobs or scripts. In addition, since all connections between the client and server are tunneled through a single secure SSH connection, the firewall and NAT router difficulties that commonly plague FTPS (FTP over SSL or TLS) are avoided.

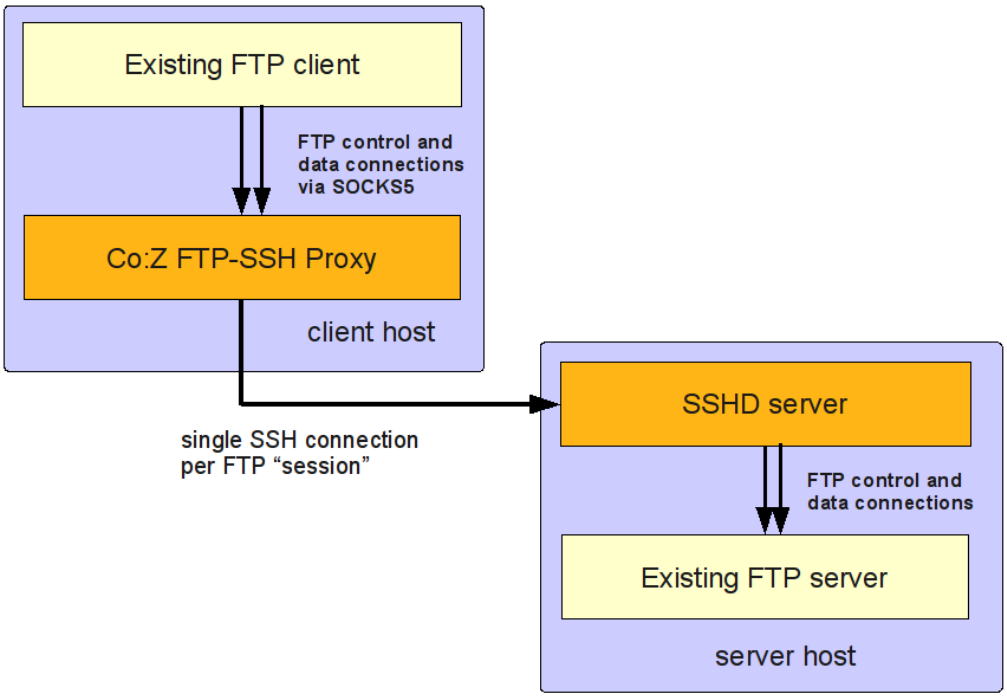

Figure 1.1. Co:Z FTP-SSH Proxy

[1] *Wikipedia: File Transfer Protocol - Criticisms of FTP*
[2] *Wikipedia: File Transfer Protocol - FTP over SSH (Not SFTP)*
[3] *Wikipedia: SOCKS*

1.1 Features

- Works transparently with most FTP clients that support SOCKS5 proxies, including z/OS.
- Works transparently with existing FTP servers. No changes are required on the server host if it is already running SSH.
- Co:Z FTP-SSH Proxy is 100% pure Java and compatible with Java 1.4 or above. It may be run on z/OS or on other Java platforms and on z/OS it exploits zAAP engines.
- Simple to install, configure, and use.
- Available free in under the Apache V2 license.

1.2 System Requirements

- Java SDK version 1.4 or later
- FTP client with SOCKS 5 support

2. Installation and Customization

The following sections describe how to install and customize the Co:Z FTP-SSH Proxy.

2.1 Client Installation and Configuration

The Co:Z FTP-SSH Proxy is a Java application which is typically run on the client host. In order to use it, the FTP client must be configured to use:

- Co:Z FTP-SSH as a SOCKS5 proxy
- passive mode FTP connections

This section will describe how to configure the z/OS Communications Server FTP client. Users of other operating systems should refer to their FTP client documentation for details.

1. Download the `ftpsshproxy-v.r.m.pax` file and upload it it binary to a temporary HFS/zFS file on your z/OS system.
2. Create an installation target directory of your choice and unpack the pax file:

```
mkdir /usr/local/ftpsshproxy
 cd /usr/local/ftpsshproxy
 pax -rf /tmp/ftpsshproxy-v.r.m.pax
```

3. The directory should contain the following files:
   - `ftpsshproxy.jar` - executable Java archive
   - `jsch-0.1.53.jar` - JCraft JSch library
   - `log4j-1.2.14.jar` - Apache Log4J logging framework
   - `ftpsshproxy.sh` - shell script to run under a Unix shell
   - `ftpsshproxy.jcl` - sample JCL to run as z/OS job (task)
   - `license/` - license information
4. Configure the FTP client to use Co:Z FTP-SSH as a SOCKS5 proxy. On z/OS, this can be configured on a system-wide, per user, or per job basis using the z/OS Communications Server FTP configuration files. [1]

   For example, to configure the FTP client for a specific userid to use passive mode and a SOCKS proxy, add the following lines to the dataset `userid.FTP.DATA`

```
FWFRIENDLY TRUE
```

[1] See the IBM publication: "z/OS Communications Server: IP Configuration Reference - File Transfer Protocol (FTP)" for details.

```
SOCKSCONFIGFILE 'userid.FTP.SOCKS.CONF'
```

Next, you must create an FTP SOCKS configuration file that enables the SOCKS proxy for selected destination FTP server ip addresses. For example, to use direct connections for selected local subnets and the proxy for everything else, create a dataset `userid.FTP.SOCKS.CONF` with the following:

```
direct 127.0.0.1/32    ; loopback adapter
direct 10.0.0.0/8      ; private subnet
direct 192.168.0.0/16  ; private subnet
sockd5 @=127.0.0.1 0.0.0.0 0.0.0.0  ; Anything else
```

This assumes that the Co:Z FTP-SSH Proxy will run on the same machine as the FTP client, and listen on the loopback address (127.0.0.1), port 1080. Note that the z/OS Communications Server FTP client can only use SOCKS proxies on port 1080, so if you have another service already running on 127.0.0.1:1080, you will need to define an alternate loopback adapter, say "127.0.0.2" in your z/OS TCP/IP profile dataset.

The z/OS FTP client will select the first line in the socks configuration file that matches the destination address. If you wanted to proxy only selected networks and addresses a sample configuration might be:

```
sockd5 @=127.0.0.1 216.34.181.0/24    ; class C subnet
sockd5 @=127.0.0.1 216.239.120.99/32  ; a single address
direct 0.0.0.0 0.0.0.0                ; Anything else
```

It is also possible to run the Co:Z FTP-SSH Proxy on another machine in the client network, but note that communications between the FTP client and the FTP-SSH Proxy machine are not encrypted.

2.2 Server Configuration

An SSH2 compatible server, such as *OpenSSH* must be installed on the same host as the target FTP server.

- The SSH server must allow "port forwarding" to the FTP server via the loopback interface ("localhost").
- The SSH server must be running on the default SSH port - 22. There is currently no way to configure the Co:Z FTP-SSH proxy to connect to alternate SSHD ports.
- The SSH server must allow logins using the same userid and password that will be used for the FTP server.

Note: The default configuration for OpenSSH supports the above requirements.

3. Operation

The Co:Z FTP-SSH Proxy is a Java application that is started as a "daemon" process, typically on the same machine as the client FTP application. It will listen on a specified TCP/IP interface and port for SOCKS requests from FTP clients. The proxy can support multiple simultaneous users and connections.

To start the Co:Z FTP-SSH Proxy from a Unix shell:

```
> cd /usr/local/ftpsshproxy
> ./ftpsshproxy.sh -h 127.0.0.1:1080
ProxyServer - Co:Z FTP-SSH Proxy - version: 0.1.0
ProxyServer - Copyright (C) Dovetailed Technologies, LLC. 2008. All rights reserved.
ProxyServer - listening on 127.0.0.1:1080
```

When an FTP client connects via the proxy to an FTP server, the Co:Z FTP-SSH Proxy log will show:

```
ProxyConnection[/127.0.0.1:49108] - proxy negotiated to /192.168.0.105:21
FtpControlSession[/127.0.0.1:49108] - client login accepted for userid "kirk"
FtpControlSession[/127.0.0.1:49108] - SSH control channel started to 192.168.0.105:21
FtpControlSession[/127.0.0.1:49108] [/192.168.0.105:21] - server login accepted for userid "kirk"
```

Once connected, the FTP session can be used normally. When the FTP client issues a command that requires a data connection (dir, put, get, etc):

```
FtpControlSession[/127.0.0.1:49108] [/192.168.0.105:21] - server registered passive port: 5101
ProxyConnection[/127.0.0.1:49110] - proxy negotiated to /192.168.0.105:5101
FtpControlSession[/127.0.0.1:49110] - SSH data channel started to 192.168.0.105:5101
```

When the FTP client disconnects:

```
FtpControlSession[/127.0.0.1:49108] [/192.168.0.105:21] - disconnected SSH
ProxyConnection[/127.0.0.1:49108] [/192.168.0.105:21] - control connection closed
```

3.1 Running Co:Z FTP-SSH Proxy under z/OS batch

The JZOS Batch Launcher[1] available with the IBM Java SDKs for z/OS[2] can be used to run the Co:Z FTP-SSH Proxy as a z/OS batch job or started task. Sample JCL, with customization instructions, can be found in the CO:Z FTP-SSH Proxy installation directory in file `ftpsshproxy.jcl`.

[1] *http://www.ibm.com/servers/eserver/zseries/software/java/products/jzos/overview.html*
[2] *http://www.ibm.com/servers/eserver/zseries/software/java/*

Appendix A. Licenses

```
Copyright 2009 Dovetailed Technologies, LLC.

The Co:Z FTP-SSH Proxy is licensed on the following terms:

Licensed under the Apache License, Version 2.0 (the "License").
You may not use the software except in compliance with this license.
You may obtain a copy of the Apache License at:

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, all software
is distributed on an "AS IS" BASIS, WITHOUT REPRESENTATION OR WARRANTIES OF
ANY KIND, either express or implied.  See the Apache License for additional
specific language governing permissions and limitations for the software.

See the NOTICES file for additional copyright and notices related to
third-party components included in this software.
```