The FTP-Proxy White Paper

The SuSE Proxy-Suite Core Development Team $Revision: 1.7 $ ($Date: 2002/05/02 13:05:28 $)

FTP-Proxy is SuSE Proxy-Suite's application-level proxy for FTP connections. It provides support for both active and passive transfers and for user-specific configuration directives. Configuration data can be obtained either from a local configuration file or from an LDAP directory.

Contents

1 Introduction

*** Note: this document is work in progress and by no means complete. It is included just for the sake of "Release early and often." ***

FTP-Proxy is a transparent, application-level proxy server for FTP connections, designed to protect FTP servers against attacks based on the FTP protocol.

Due to the dual TCP connection nature of the FTP protocol, special handling is required to allow secure transfers. Data connections are usually opened by the FTP server (the FTP protocol specification calls this "active" transfers) or optionally by the FTP client ("passive" transfers). The data ports to be used are determined dynamically and announced as part of the protocol. This makes it very difficult for static packet filters to handle FTP correctly.

Most routers and firewalls know about this procedure when NAT is performed. E.g. Linux provides a special kernel module for FTP (and other protocols). However such code usually only works in "outbound" direction, allowing FTP access from the inner network to FTP servers on the outer network, but not in the other direction.

This is a design restriction: TCP/IP packets originating from the inner network can be marked with special information that allows for correct routing of response packets from the outer network. For instance, if a WWW client on host 192.168.168.27 on your internal network requests a web page from Web server 194.112.123.200, the router maintains internal information when it rewrites the headers of the request packets. If a response packet arrives (with a "From" address of 194.112.123.200), the router is then able to replace the public IP address in the "To" field with the private IP address of the appropriate client, in this case 192.168.168.27.

Of course, if a connection originates from the outer network, the router does not have this additional information and it can't forward incoming IP packets since it has no idea which host on the inner network is to be addressed. One solution could be to enable port forwarding on the router (if supported), so that FTP clients on the outer network could connect to port X on the router to reach server Y on the inner network, however this fails because of the restrictions in the FTP protocol outlined above.

A proxy server is an elegant way to circumvent these restrictions, but there are other benefits of using FTP-Proxy as well:

- Simple forwarding of all traffic destined for the inner FTP server leaves that server nearly as vulnerable as if it had a "direct" connection to the outer network (without the firewall in front of it). In that case the FTP server would still be open to Denial-of-Service and similar attacks on the IP level.

  As FTP-Proxy runs on the application level, it doesn't forward the original IP packets but only maintains the information within. For instance, a packet containing not only a valid FTP command but also information to exploit an FTP server's bugs will never reach the server because only the FTP command itself is forwarded, within a completely new IP packet.

  Of course, FTP-Proxy could be subject to an attack itself. However, FTP-Proxy is much less complex than any current FTP server, has been designed with great care and performs chroot(), setuid() and setgid() to avoid such vulnerability.

- Using a circuit-level proxy server like SOCKS requires that all FTP clients support SOCKS. This seems to be too restrictive. FTP-Proxy, on the other hand, is completely transparent to FTP clients because it is fully RFC 959 compliant.

2 Installation

The following discussion assumes that you have obtained the tarball distribution and want to compile and install the FTP-Proxy package from ground up. Let's assume you have received a file named `proxy-suite-x.y.tar.gz`, where 'x' is the major and 'y' is the minor version number.

1. Decide where to install and compile the source code. Say you want to install it in `/usr/src/proxy-suite-x.y`. Then do the following to unpack the source (assuming the tarball is available in the current directory when calling gunzip):

   ```
   cd /usr/src
   gunzip -c proxy-suite-x.y.tar.gz | tar xvf -
   ```

   If you do not have a copy of GZIP, look at the *GNU* archive or one of its numerous mirrors. It's free.

2. After unpacking the source you need to run the `configure` shell script which performs the customization for you operating system and local environment. This script is based upon the GNU AutoConf package. Change your current directory to be the root of the Proxy-Suite:

```
cd proxy-suite-x.y
```

The `configure` script understands a number of options which control the customization process. To see a complete list of all options simply run:

```
./configure --help
```

The following options are most likely to be involved:

`--prefix=<directory>`

This is the root directory for the installation. If none of the other directory related directives further down is also given, all files will be installed in subdirectories under this one. The default is `/usr/local/proxy-suite/`.

`--exec-prefix=<directory>`

A second top level installation directory, this one is used as the basis for all executables. Defaults to the same as PREFIX.

`--mandir=<directory>`

The FTP-Proxy comes with two manual pages: *ftp-proxy(8)* for the program and *ftp-proxy.conf(5)* for the configuration file format. These will be installed under the `man5` and `man8` subdirectories of the given one, or under `PREFIX/man` if not specified.

`--sbindir=<directory>`

The next directory specifies the location where the `ftp-proxy` executable itself will be installed. The default is `EXEC-PREFIX/sbin` if this option does not request otherwise.

`--sysconfdir=<directory>`

The last directory related option (others may be given, but have no effect) deals with the `proxy-suite/ftp-proxy.conf` configuration file. Again there is a default if the option is not provided, `PREFIX/etc`. Please note that the location of the config file can be selected at run time with the *-f* flag, but if the file is installed at the default location, this flag is redundant.

`--enable-debug`

This option allows the generation of debugging output and is mainly included for evaluation (and development) purposes. It is recommended not to specify this option for deployment of the FTP-Proxy into production environments.

`--enable-warnings`

If compiling with the GNU GCC compiler, this flag increases the warning level. This is useful to detect programming errors or incompatibilities with the target platform. Our recommendation is to enable it anyway.

`--enable-so-linger`

It is strongly recommended to enable this option for all systems. Only if the target does have problems with *SO_LINGER* handling, it should be disabled.

`--with-regex[=PATH]`

This option is mainly needed to provide the location of the POSIX 1002.3 regular expression library if it is not included in the standard C library. Regular expressions are used to scan the arguments of the various FTP commands received from clients. If your system does not include the required support, you can download the latest version from the *GNU* archive. Again, it's free. GNU RegEx is known to work with FTP-Proxy. If the option is not given, regular expression support is not installed at all and arguments will not be investigated any further.

`--with-libwrap[=PATH]`

If running as a standalone background daemon, FTP-Proxy can be instructed to make use of the *TCP Wrapper* library. If this is compiled in and enabled in the configuration file, the `/etc/hosts.allow` and `/etc/hosts.deny` files will be consulted to decide if a client will be served.

`--with-libldap[=PATH]`

If you want to supply the user specific configuration values (not the basic ones) via an LDAP directory, compile in LDAP support. The `configure` script can determine without further manual intervention whether to use the *OpenLDAP*, the *University of Michigan* or the *Netscape* version of the API. *OpenLDAP* is the preferred API.

`--with-crypt[=PATH]`

This option enables support for crypted passwords in user authentication (curently ldap based only). See also the `LDAPAuthPWType` configuration file option.

3. After `configure` has completed successfully, it is now time to run `make` (GNU make). It will read the `Makefile` generated by `configure` in the previous step and perform the compilation process. The program should not display any warnings or errors.

4. When the executables have been generated, a final call to `make install` will copy the FTP-Proxy files into their appropriate places. Usually root privileges are required in order to install the files in their proper places.

5. Currently, you'll have to modify your system manually so that ftp-proxy is run at system bootup or from inetd (depending on which mode you choose in the configuration file or command line). We provide an sample run level script `rc.script` for (SuSE) Linux; see also description in `rc.script.txt`.

6. The last step is the configuration of the installed FTP-Proxy; see following 3 (Configuration) section.

3 Configuration

The following section describes the fundamental configuration of the FTP-Proxy. It assumes, that you have successfully compiled and installed the FTP-Proxy.

The proxy uses a configuration file *ftp-proxy.conf* you have to edit before the proxy can be used - it is needed to configure at least the variables described in this section. For a complete list of valid configuration options consult the *ftp-proxy.conf(5)* manual page and the sample configuration file *ftp-proxy.conf.sample* file which comes with the distribution.

First, you have to choose, if you wand to start the proxy as a stand-alone daemon or via "(x)inetd" super daemon using the `ServerType` configuration option (or the start arguments). If nothing specified, "inetd" mode is assumed. See also section 4 (Local System Security). In stand-alone daemon mode, the proxy defaults to listen on the FTP port 21 and all interfaces. You can override this using the `Port` and `Listen` configuration options.

The next step you have to do, is to choose, if the Proxy should run in "inbound" or in "outbound" mode. See the section 5 (Inbound and Outbound FTP Traffic).

Now, the proxy should be able to run and serve the client requests. The proxy follows the FTP transfer mode the client has choosed.

You can override this behaviour setting the option `DestinationTransferMode` to either *active* or *passive* instead of the default *client*, and the proxy will be forced to use the specified FTP transfer mode for all transfers it does between the proxy and the ftp server. Most common usage is to force the proxy to use *passive* transfers only.

4 Local System Security

FTP-Proxy comes with several configuration features that help to increase local system security, namely `ServerRoot`, `User` and `Group`.

The way FTP-Proxy is being called needs to be considered. One possible way is via the system's inetd (or xinetd) Internet Super Daemon. In this case FTP-Proxy will not fork or become a daemon. It will serve the client and terminate itself after delivery. When configuring (x)inetd to include the ftp-proxy executable, `ServerRoot` (chroot) should be used. The `User` and `Group` need not be given if they are specified in the inetd configuration itself.

The `User` and `Group` options should actually be considered for standalone operations. In this case the ftp-proxy will bind the listening socket to the port number set using the `Port` and `Listen` options, preform the chroot operation if `ServerRoot` is used, drop privileges to the UID/GID set with `User` and `Group` options and open log.

It might be a good idea to create a new user (e.g. "ftpproxy") as well as a group (e.g. "ftpproxy") in order to reach a better granularity for the user administration.

When using `ServerRoot`, please note that usually other files needs to be installed into the runtime environment as well, e.g. the /dev/null device, system databases like /etc/services, /etc/hosts, libraries like libc and possibly other (e.g. libcrypt under AIX 4.3, a resolver library like libresolv or libnss libraries on systems using the NameServiceSwitch - see also nsswitch.conf(5)). If you are using the `User` and `Group` options, you may also need the /etc/passwd and /etc/group files.

The sample run level script `rc.script` for (SuSE) Linux supports the preparation of a chroot runtime environment - see description in `rc.script.txt`.

5 Inbound and Outbound FTP Traffic

The most common use of FTP-Proxy will probably be "inbound" FTP traffic. This means that clients from an "outside" world seek access to a "local" FTP server specified in the `DestinationAddress` variable.

Nevertheless, FTP-Proxy supports also an "outbound" mode, where clients have more control over the FTP connections. The FTP-Proxy implements two concepts of "outbound" traffic.

- *MagicUser* The first one is based on the increased level of trust that users enjoy. When setting the `AllowMagicUser` config option to "yes" and allowing the "@" (see also `UseMagicChar` option) and ":" characters as part of the USER command argument, users can determine the destination server's address and port with the USER command. All they have to do is to append the host name, separated by the "@" sign (or other set using `UseMagicChar` option), optionally followed by a colon and the port. These components will be removed from the name and evaluated as destination.

- *Transparent-Proxy* The second one is based on IP-NAT packet redirections, commonly called *transparent proxying*. This method is currently implemented for Linux ipchains (2.2 kernel) and iptables (2.4 kernel), as well as for BSD ipnat, tested on OpenBSD 2.9 and FreeBSD 4.4. A description how to setup the redirections is provided in the `TransProxy-Mini-Howto.txt` file.

  When setting the `AllowTransProxy` config option to "yes", the proxy will evaluate the original destination address and port the client wanted connect as destination. If `AllowMagicUser` is enabled as well, the users are still able to provide a different destination using the USER command argument.

If `AllowTransProxy` and `AllowMagicUser` are not used, the FTP-Proxy runs in the "inbound" mode and the `DestinationAddress` is mandatory.

In "outbound" mode, `DestinationAddress` is used as default or fallback destination, that will be used if no other destination is found using *Transparent-Proxy* or *Magic-User* mechanisms.

6 Command Restriction

The FTP-Proxy allows to define a space separated list of allowed FTP commands using the `ValidCommands` configuration variable in global and in a per user context. If this variable is not used (default), no command restriction will take place and all of the following commands are allowed:

```
ABOR ACCT ALLO APPE CDUP CWD
DELE HELP LIST MAIL MDTM MKD
MLFL MODE MRCP MRSQ MSAM MSND
MSOM NLST NOOP PASS PASV PORT
PWD  QUIT REIN REST RETR RMD
RNFR RNTO SITE SIZE SMNT STAT
STOR STOU STRU SYST TYPE USER
XCUP XCWD XMKD XPWD XRMD
```

Otherwise, only commands included in the list are allowed and all other denied.

Further, if the FTP-Proxy is compiled with regular expression support (see 2 (–with-regex) switch), each command may be followed by an optional equal sign and a *POSIX 1003.2 Extended Regular Expression (RE)* that describes the valid argument(s) for the command.

If the whole string is to be matched, the pattern has to start with a caret (^) and end with a dollar ($). If no pattern follows a command, its arguments are not checked.

An example for a name would be the pattern `^[a-zA-Z0-9]{1,16}$`, i.e. as expresion for the USER command:

```
USER=^[a-zA-Z0-9]{1,16}$
```

This definition specifies, that the argument is mandatory and may consist of up to 16 letters or digits only.

A command that does not allow any arguments can also easily be represented, i.e:

```
QUIT=^$
```

Please note that the regular expression is "pre-processed". This means that a pattern in the form `%xx` will be interpreted as a hexadecimal constant and will be replaced by the value of that constant. This looks a bit like HTML and helps to include characters that might not be handled as expected, like `%20` for space or `%5c` (equivalent to `%5C`) for backslash. The space is especially important because it is the separator for the commands within the list itself.

7 User Authentication

Since proxy-suite Version 1.9 the ftp-proxy supports user authentication. To enable it, the `UserAuthType` configuration option have to be set to the name of the mechanism used, i.e. *ldap* (currently the only one).

For more information on LDAP based authentication, see also the 8 (Using an LDAP Directory) section.

Per default, the normal "ftpuser" and "ftppass" from *USER* and *PASS* FTP commands are used for the authentication. This may be usefull especially in "inbound" mode of the proxy.

In "outbound" mode, it may be usefull to use an extended enconding of an "authuser" and "authpass" additionally to the normal "ftpuser" and "ftppass" using the `UserAuthMagic` option supported by some FTP clients, i.e. *"@auth" for NcFTP, type 5*.

The `UserAuthMagic` configuration variable can be set to either *auth@* or *@auth*, where "@" is an encoding separator character and can also be set to an different one, i.e. to ":", using *auth:* or *:auth*.

If the "auth" keyword is prepended by the separator character, the USER command will be parsed as "ftpuser@authuser" or in combination with the `AllowMagicUser` as "ftpuser@authuser@host:port" and the PASS FTP command as "ftppass@authpass".

If the "auth" keyword is followed by the separator character, the parsing is done in the different order as "authuser@ftpuser".

8 Using an LDAP Directory

The main option to use an LDAP directory is the `LDAPServer` configuration option. If given, it specifies the hostname of the directory server (optionally followed by port number separated by a colon).

The program will bind the directory using `LDAPBindDN` (and `LDAPBindPW`) and retrive the values having an object class of `LDAPObjectClass` and identified by the `LDAPIdentifier`.

The *LDAPBindDN* and *LDAPBindPW* option defines the distinguished name and credentials (password) needed to access the data in the directory service. It is allowed to include one %s in *LDAPBindDN* - it will be replaced with the user name. If *UserAuthMagic* is used, the special "authuser" and "authpass" are used, otherwise normal "ftpuser" and "ftppass" from USER and PASS ftp commands. If no *LDAPBindDN* specified, a annonymous bind will be used.

Additionally the directory tree root should be specified using the `LDAPBaseDN` or `LDAPAuthDN` option. You can also use both options set to differen root's if your profile data is stored in a different tree than the authentication data. One of both options is mandatory.

- *LDAP User-Profiles* In order to gain more flexibiliy for the user management, the user dependent parts of the configuration can also be supplied with an LDAP directory.

  The following configuration options will be tryed to retrieve from the tree root specified by the `LDAPBaseDN` option:

`DestinationAddress, DestinationPort, DestinationTransferMode, DestinationMinPort, DestinationMaxPort, ActiveMinDataPort, ActiveMaxDataPort, PassiveMinDataPort, PassiveMaxDataPort, SameAddress, TimeOut, ValidCommands.`

For example, if your user (USERNAME) dependent configuration is stored as *uid=USERNAME,ou=FTPProxy,dc=domain,dc=top* in the directory, the setup may be as follows:

```
LDAPServer      ldap.domain.top:389
LDAPBaseDN      ou=FTPProxy,dc=domain,dc=top
LDAPIdentifier  uid
```

If a non-anonymous bind is needed to access the tree, a *LDAPBind* can be specified either to an specific user, i.e. "proxyuser":

```
LDAPBindDN      uid=proxyuser,ou=FTPProxy,dc=domain,dc=top
```

or also the user who want to login (ftp-user or auth-user name depending on UserAuthMagic):

```
LDAPBindDN      uid=%s,ou=FTPProxy,dc=domain,dc=top
```

- *User-Authentication* Since proxy-suite Version 1.9 the ftp-proxy supports also LDAP based user authentication.

  To activate it, you have to set the `UserAuthType` configuration option to *ldap* and define the authentication tree root using the `LDAPAuthDN` configuration option. If no `LDAPAuthDN` option is set, `LDAPBaseDN` is used instead.

  If you only want to check, if an user is allowed to use the ftp-proxy service or not, you can define the `LDAPAuthOKFlag` option to an attribute name and its value, separated with an equal character. The program will check, if the value for the given attribute exists - the attribute may contain multiple values. Example:

  ```
  LDAPAuthOKFlag   AllowedService=ftpProxy
  ```

  .

  Further it is also possible to preform an password authentication using the `LDAPAuthPWAttr` and `LDAPAuthPWType` options. The `LDAPAuthPWAttr` defines the name of the password attribute. A common name for this attribute is *userPassword*.

  The `LDAPAuthPWType` option defined the type of the password stored in the directory service. Supported password types are *plain* for plain-text passwords, *crypt* for crypted passwords and {*crypt*} for crypted passwords prefixed with a {*crypt*} string (a scheme specification). The type may be followed by the number 0-9 of minimal allowed passord length, i.e.

  ```
  LDAPAuthPWType   plain
  ```

  This definition means, the directory contains plain-text password with the default minimal length of at least 5 characters.

9 Logging and Auditing

All possible log messages with their exact wording have been collected and listed in the file *SYSLOG* which is part of the FTP-Proxy distribution. You can find it in the *ftp-proxy* subdirectory.

The FTP-Proxy logs can be sent to a file, to a pipe, or to the *syslogd(8)* daemon for further evaluation and handling. It is strongly recommended to use syslog because it is the only way to send logging messages to another machine as they are generated.

This off shore logging is especially important for FTP-Proxy servers located in exposed areas, like the Demilitarized Zone of a firewall. Anything that is stored on the computer itself can easily be manipulated in the case of a hostile takeover. And when the auditor stops by on his weekly audit tour, all traces have long since been wiped out. In order to implement remote logging the *syslog.conf(5)* file has to include an entry for the selected facility using a target starting with the letter '@' followed by the DNS name or IP address of the external log host. Usually this loghost may be not part of the DMZ, but be located on the intranet or dedicated administration network. The UDP syslog traffic is then allowed to pass through the internal firewall router. Note that the syslogd on the log host usually has to be started with the "*-r*" option to enable the reception of logging information from another machine.

The FTP-Proxy syslog messages have been created in a way that should ease the task of scanning and evaluating. Each message contains a tag that can easily be recognized. It consists of a prefix that is either *TECH* or *USER*, followed by a hyphen and one of *DBG*, *INF*, *WRN*, *ERR*, or *FTL*.

Using the `LogLevel` configuration option, you can skip message levels not interesting for you. For example,

```
LogLevel  WRN
```

will cause skipping of all *DBG* and *INF* messages and displaying only the *WRN*, *ERR* and *FLT* level messages. Default level is *INF* - no *DBG* messages will get logged.

The possible combinations can be interpreted as follows:

`TECH-DBG`

Messages with this tag carry some technical informations usefull for diagnostics purposes with no need to react to them.

`TECH-INF`

Messages with this tag carry some technical information with no need to react to them. They might display the contents of the configuration file or the start or stop of the program. These messages are generated with the *INFO* syslog severity.

`TECH-WRN`

Messages with this tag display some mild technical problem or inconsistency. E.g. a configuration variable with no value given or a temporary resource shortage would be sent in this context. Also if an FTP server closed a connection without being instructed by the client or proxy would generate this kind of message. These messages are generated with the *WARNING* syslog severity.

`TECH-ERR`

Messages with this tag usually mean a severe error condition that will lead to the termination of the emitting process. Problems that fall under this category include the inability to open the configuration file, internal security handling, network problems like ports already in use by other processes, input/output errors or other communication faults. It is strongly advised that all such messages shall be investigated and further action is taken. These messages are generated with the *ERROR* syslog severity.

`TECH-FTL`

Messages with this tag should be very rare. They are reserved for really weird situations which reveal bugs in the internal FTP-Proxy programming. Please report any such occurrance to the Proxy-Suite development team, it requires the programmer's attention. These messages are generated with the *CRIT* syslog severity.

`USER-DBG`

Messages with this tag are currently not being used.

`USER-INF`

Messages with this tag will be the majority of all messages. They are generated for all regular user actions like logging in, sending commands and transfer statistics. Their main purpose is to provide a complete audit trail for every user interaction. These messages are generated with the *INFO* syslog severity.

`USER-WRN`

Messages with this tag indicate some sort of unforeseen user action, but need not be taken too seriously. Only if there is a pattern of regular or frequent messages of this kind, you might want to look a bit closer. These messages are generated with the *WARNING* syslog severity.

`USER-ERR`

Messages with this tag mean that the user has been rejected. Currently no other message has been defined with this tag, although this might be changed in the future. These messages are generated with the *ERROR* syslog severity.

`USER-FTL`

Messages with this tag are currently not being used.

10 Copyright and License

Free Software Foundation, Inc. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

11 Authors

FTP-Proxy is part of the SuSE Proxy-Suite project and was written by

Volker Wiegand (programming, *wiegand@suse.de*),

Jens-Gero Boehm (project management, *Jens-Gero.Boehm@suse.de*),

Pieter Hollants (documentation, *Pieter.Hollants@suse.de*), and

Marius Tomaschewski (maintainance, programming, *mt@suse.de*).

A Mailing List has been installed for the discussion of SuSE Proxy-Suite. To subscribe, send an empty E-Mail to *proxy-suite-subscribe@suse.com*.

We have also installed an announcement mailing list. Here you will be informed about updates and other major issues. To subscribe, send an empty E-Mail to *proxy-suite-announce-subscribe@suse.com*.

Last but not least you may send bug reports to the following address: *proxy-suite-bugs@suse.com*.

Please visit the SuSE Proxy-Suite Homepage at *http://proxy-suite.suse.de/*.