stunnel使用详解

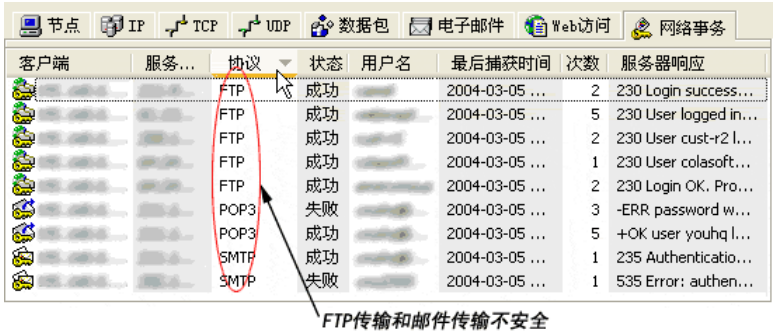
本文链接：https://blog.csdn.net/qq_43667702/article/details/87020490
http://www.stunnel.org/faq/args.html

http://www.colasoft.com.cn/support/monitor_stunnel.php

科来网络分析系统与stunnel结合使用

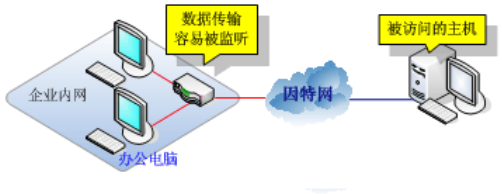
科来网络分析系统是一款强大的网络检测分析工具，可对网络中未加密的数据传输进行检测分析并实时显示分析结果，包括用户的邮件收发、Web访问以及各种网络登录等操作。所以，未经加密的数据传输是不安全的，存在被别人窃听的安全隐患。

图一显示的是科来网络分析系统对网络数据传输捕获的结果，从中可以看出，当前网络中的敏感数据传输都未经过任何的加密保护，存在安全隐患。为加强网络本身的安全性，在使用科来网络分析系统进行网络管理的同时，我们建议用户对重要的数据传输进行加密保护，以达到管理和保护的有效结合。在这种情况下，即使数据被窃取，攻击者也无法分析数据的真实内容，从而保证了数据传输的安全性。



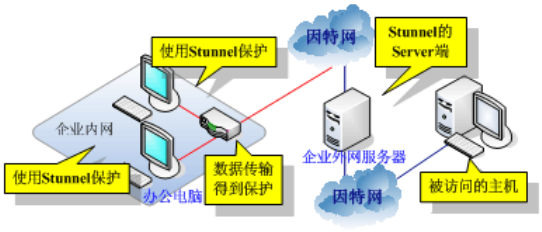
图一，网络事务分析结果

图二所示为常见的网络传输情况，在这种情况下，数据在网络中的传输没有经过任何保护，当网络在遭受黑客攻击或黑客入侵时，重要数据很容易被窃取。为了使我们局域网传输的重要数据都是安全的，我们可以利用Stunnel工具对数据进行加密。



图二，普通网络

Stunnel (http://www.stunnel.org/)是一款可以加密网络数据的TCP连接，并可工作在Unix和Windows平台上，它采用Client/Server模式，将Client端的网络数据采用SSL(Secure Sockets Layer)加密后，安全的传输到指定的Server端再进行解密还原，然后再发送到访问的服务器。在加密传输过程中，可充分确保数据的安全性，我们只要把Server端程序安装在局域网外面的一台服务器上，即可保证传输的数据在局域网内是安全的，如图三所示。



图三，Stunnel加密后的网络

操作过程：

Stunnel是一款免费的工具，可以在这里下载。下面我们介绍一下具体的使用。

1. 下载Stunnel Client端程序，并解压到本机的C:/Program Files目录下。
2. 下载Stunnel Server端程序，并解压后放在外网的服务器上。
3. 分别配置stunnel.conf文件。
4. 更改本机应用程序的网络连接配置。
5. 分别运行stunnel.-4.04.exe执行文件。

说明：

我们使用Stunnel，需要在外网有一台有管理权限的服务器，来运行Stunnel的Server程序。配置好Stunnel.conf后，可将执行文件在启动菜单中建立快捷方式，这样

让每次开机时，能自动运行。Stunnel技术是将传输的信息加密后，通过Server端的服务器进行解密才到达的目的主机，所以在选择Server端服务器的时候，对服务器的带宽速度有一定的要求。

Stunnel的配置：

Client和Server端都包含stunnel.conf配置文件，格式如下表所示：

Client端stunnel.conf文件内容	Server端stunnel.conf文件内容

# Use it for client mode 	
client = yes
#Client-level configuration
[应用服务名称]
accept =本地IP : 目标端口
connect =Server端IP : 指定的端口
	# Use it for server mode
	client = no
	#Server-level configuration
	[应用服务名称]
	accept = 指定的端口
	connect =目标服务器IP : 目标端口

常见应用实例：

Stunnel.conf文件配置比较简单，下面我们介绍一些常见应用配置，其中Client端是放在本机，IP是127.0.0.1，Server端是放在外网的服务器上，IP是202.151.90.28。

1.加密邮件传输：

加密邮件，需要将发送和接收的过程都要进行保护，那么我们就要对POP3和SMTP传送方式进行加密。如果我们有一个xxx@colasoft.com.cn信箱，服务器的IP是202.108.44.153，配置文件stunnel.conf如下：

Client端SMTP和POP3文件内容	Server端SMTP和POP3文件内容
[smtp.colasoft.com.cn] accept = 127.0.0.1:25 connect = 202.151.90.28:125 [pop3.colasoft.com.cn] accept = 127.0.0.1:110 connect = 202.151.90.28:1110	[smtp.colasoft.com.cn] accept = 125 connect = 202.108.44.170:25 [pop3.colasoft.com.cn] accept = 1110 connect = 202.108.44.153:110

如果有多个邮件传输需要加密，则增加相应的POP3和SMTP设置即可。设置好了配置文件，我们还需要将邮件客户端（常见的为Foxmail或Outlook）与其对应，设置如下：

发送的邮件地址改为：127.0.0.1　　端口改为：125

接收的邮件地址改为：127.0.0.1　　端口改为：1110

2.加密FTP传输：

FTP是比较早的文件传输协议，内容都是以明文方式传输，我们利用Stunnel后，也可以让FTP的传输非常安全，现在我们只需要在前面的stunnel.conf内容里面增加以下配置信息：

Client端FTP的配置	Server端FTP的配置
[ftp.net130.com] accept = 127.0.0.1:21 connect = 202.151.90.28:121	[ftp.net130.com] accept = 121 connect = 218.7.9.73:21

FTP软件（如CuteFTP）也要做相应更改：

登录的远程地址改为：127.0.0.1　　端口改为121

3.加密HTTP网站访问传输：

我们不能对所有的网站访问都进行加密，因为太多，但对于很重要的网站，我们也可以用Stunnel来保护访问的内容不受到监听。例如我们要访问www.colasoft.com.cn，网站IP地址是202.108.36.172，HTTP的配置如下：

Client端HTTP的配置	Server端HTTP的配置
[www.colasoft.com.cn] accept = 127.0.0.1:80 connect = 202.151.90.28:8080	[www.colasoft.com.cn] accept = 8080 connect = 202.108.36.172:80

通过结合使用科来网络分析系统与Stunnel，既可以做到对网络的安全检测，并找出网络内的潜在安全隐患，又能从防护的角度出发，保护公司内部网络的重要信息。此方案成本低，不改变当前网络内的结构，容易实施，是一个简单有效的安全管理方案。

Stunnel-4.10 Man Page

- NAME
- SYNOPSIS
- DESCRIPTION
- OPTIONS
- CONFIGURATION FILE
  - GLOBAL OPTIONS
  - SERVICE-LEVEL OPTIONS
- RETURN VALUE
- EXAMPLES
- FILES
- BUGS
- RESTRICTIONS
- NOTES
  - INETD MODE
  - CERTIFICATES
  - RANDOMNESS
- SEE ALSO
- AUTHOR

NAME

stunnel - universal SSL tunnel

SYNOPSIS

Unix:

stunnel [<filename>] | -fd n | -help | -version | -sockets

WIN32:

stunnel [[-install | -uninstall] [-quiet] [<filename>]] | -help | -version | -sockets

DESCRIPTION

The **stunnel** program is designed to work as *SSL* encryption wrapper between remote clients and local (*inetd*-startable) or remote servers. The concept is that having non-SSL aware daemons running on your system you can easily set them up to communicate with clients over secure SSL channels.

stunnel can be used to add SSL functionality to commonly used *Inetd* daemons like POP-2, POP-3, and IMAP servers, to standalone daemons like NNTP, SMTP and HTTP, and in tunneling PPP over network sockets without changes to the source code.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)

OPTIONS

<filename>

Use specified configuration file

-fd n (Unix only)

Read the config file from specified file descriptor

-help

Print **stunnel** help menu

-version

Print **stunnel** version and compile time defaults

-sockets

Print default socket options

-install (NT/2000/XP only)

Install NT Service

-uninstall (NT/2000/XP only)

Uninstall NT Service

-quiet (NT/2000/XP only)

Don't display a message box when successfully installed or uninstalled NT service

---

CONFIGURATION FILE

Each line of the configuration file can be either:

- an empty line (ignored)
- a comment starting with ';' (ignored)
- an 'option_name = option_value' pair
- '[service_name]' indicating a start of a service definition

GLOBAL OPTIONS

CApath = directory

Certificate Authority directory

This is the directory in which **stunnel** will look for certificates when using the *verify*. Note that the certificates in this directory should be named XXXXXXXX.0 where XXXXXXXX is the hash value of the cert.

CApath path is relative to *chroot* directory if specified.

CAfile = certfile

Certificate Authority file

This file contains multiple CA certificates, used with the *verify*.

cert = pemfile

certificate chain PEM file name

A PEM is always needed in server mode. Specifying this flag in client mode will use this certificate chain as a client side certificate chain. Using client side certs is optional. The certificates must be in PEM format and must be sorted starting with the certificate to the highest level (root CA).

chroot = directory (Unix only)

directory to chroot **stunnel** process

chroot keeps **stunnel** in chrooted jail. *CApath*, *CRLpath*, *pid* and *exec* are located inside the jail and the patches have to be relative to the directory specified with **chroot**.

To have libwrap (TCP Wrappers) control effective in a chrooted environment you also have to copy its configuration files (/etc/hosts.allow and /etc/hosts.deny) there.

ciphers = cipherlist

Select permitted SSL ciphers

A colon delimited list of the ciphers to allow in the SSL connection. For example DES-CBC3-SHA:IDEA-CBC-MD5

client = yes | no

client mode (remote service uses SSL)

default: no (server mode)

compression = zlib | rle

select data compression algorithm

default: no compression

CRLpath = directory

Certificate Revocation Lists directory

This is the directory in which **stunnel** will look for CRLs when using the *verify*. Note that the CRLs in this directory should be named XXXXXXXX.0 where XXXXXXXX is the hash value of the CRL.

CRLpath path is relative to *chroot* directory if specified.

CRLfile = certfile

Certificate Revocation Lists file

This file contains multiple CRLs, used with the *verify*.

debug = [facility.]level

debugging level

Level is a one of the syslog level names or numbers emerg (0), alert (1), crit (2), err (3), warning (4), notice (5), info (6), or debug (7). All logs for the specified level and all levels numerically less than it will be shown. Use **debug = debug** or **debug = 7** for greatest debugging output. The default is notice (5).

The syslog facility 'daemon' will be used unless a facility name is supplied. (Facilities are not supported on Win32.)

Case is ignored for both facilities and levels.

EGD = egd path (Unix only)

path to Entropy Gathering Daemon socket

Entropy Gathering Daemon socket to use to feed OpenSSL random number generator. (Available only if compiled with OpenSSL 0.9.5a or higher)

engine = auto | <engine id>

select hardware engine

default: software-only cryptography

foreground = yes | no (Unix only)

foreground mode

Stay in foreground (don't fork) and log to stderr instead of via syslog (unless **output** is specified).

default: background in daemon mode

key = keyfile

private key for certificate specified with *cert* option

Private key is needed to authenticate certificate owner. Since this file should be kept secret it should only be readable to its owner. On Unix systems you can use the following command:

```
chmod 600 keyfile
```

default: value of *cert* option

options = SSL_options

OpenSSL library options

The parameter is the OpenSSL option name as described in the *SSL_CTX_set_options(3ssl)* manual, but without *SSL_OP_* prefix. Several *options* can be used to specify multiple options.

For example for compatibility with erroneous Eudora SSL implementation the following option can be used:

```
options = DONT_INSERT_EMPTY_FRAGMENTS
```

output = file

append log messages to a file instead of using syslog

/dev/stdout device can be used to redirect log messages to the standard output (for example to log them with daemontools splogger).

pid = file (Unix only)

pid file location

If the argument is empty, then no pid file will be created.

pid path is relative to *chroot* directory if specified.

RNDbytes = bytes

bytes to read from random seed files

Number of bytes of data read from random seed files. With SSL versions less than 0.9.5a, also determines how many bytes of data are considered sufficient to seed the PRNG. More recent OpenSSL versions have a builtin function to determine when sufficient randomness is available.

RNDfile = file

path to file with random seed data

The SSL library will use data from this file first to seed the random number generator.

RNDoverwrite = yes | no

overwrite the random seed files with new random data

default: yes

service = servicename

use specified string as the service name

On Unix: *inetd* mode service name for TCP Wrapper library.

On NT/2000/XP: NT service name in the Control Panel.

default: stunnel

session = timeout

session cache timeout

setgid = groupname (Unix only)

`setgid()` to groupname in daemon mode and clears all other groups

setuid = username (Unix only)

`setuid()` to username in daemon mode

socket = a|l|r:option=value[:value]

Set an option on accept/local/remote socket

The values for linger option are l_onof:l_linger. The values for time are tv_sec:tv_usec.

Examples:

```
    socket = l:SO_LINGER=1:60        set one minute timeout for closing local socket    socket =
r:TCP_NODELAY=1        turn off the Nagle algorithm for remote sockets    socket = r:SO_OOBINLINE=1
place out-of-band data directly into the        receive data stream for remote sockets    socket =
a:SO_REUSEADDR=0        disable address reuse (enabled by default)    socket = a:SO_BINDTODEVICE=lo
only accept connections on loopback interface
```

taskbar = yes | no (WIN32 only)

enable the taskbar icon

default: yes

verify = level

verify peer certificate

```
    level 1 - verify peer certificate if present    level 2 - verify peer certificate    level 3 - verify
peer with locally installed certificate    default - no verify
```

SERVICE-LEVEL OPTIONS

Each configuration section begins with service name in square brackets. The service name is used for libwrap (TCP Wrappers) access control and lets you distinguish **stunnel** services in your log files.

Note that if you wish to run **stunnel** in *inetd* mode (where it is provided a network socket by a server such as *inetd*, *xinetd*, or *tcpserver*) then you should read the section entitled *INETD MODE* below.

accept = [host:]port

accept connections on specified host:port

If no host specified, defaults to all IP addresses for the local host.

connect = [host:]port

connect to remote host:port

If no host specified, defaults to localhost.

delay = yes | no

delay DNS lookup for 'connect' option

exec = executable_path (Unix only)

execute local inetd-type program

exec path is relative to *chroot* directory if specified.

execargs = $0 $1 $2 ... (Unix only)

arguments for *exec* including program name ($0)

Quoting is currently not supported. Arguments are separated with arbitrary number of whitespaces.

ident = username

use IDENT (RFC 1413) username checking

local = host

IP of the outgoing interface is used as source for remote connections. Use this option to bind a static local IP address, instead.

protocol = proto

Negotiate SSL with specified protocol

currently supported: cifs, nntp, pop3, smtp

pty = yes | no (Unix only)

allocate pseudo terminal for 'exec' option

TIMEOUTbusy = seconds

time to wait for expected data

TIMEOUTclose = seconds

time to wait for close_notify (set to 0 for buggy MSIE)

TIMEOUTconnect = seconds

time to wait to connect a remote host

TIMEOUTidle = seconds

time to keep an idle connection

transparent = yes | no (Unix only)

transparent proxy mode

Re-write address to appear as if wrapped daemon is connecting from the SSL client machine instead of the machine running **stunnel**. This option is only available in local mode (*exec* option) by LD_PRELOADing env.so shared library or in remote mode (*connect* option) on Linux 2.2 kernel compiled with *transparent proxy* option and then only in server mode. Note that this option will not combine with proxy mode (*connect*) unless the client's default route to the target machine lies through the host running **stunnel**, which cannot be localhost.

RETURN VALUE

stunnel returns zero on success, non-zero on error.

EXAMPLES

In order to provide SSL encapsulation to your local *imapd* service, use

```
[imapd]    accept = 993    exec = /usr/sbin/imapd    execargs = imapd
```

If you want to provide tunneling to your *pppd* daemon on port 2020, use something like

```
[vpn]    accept = 2020    exec = /usr/sbin/pppd    execargs = pppd local    pty = yes
```

If you want to use **stunnel** in *inetd* mode to launch your imapd process, you'd use this *stunnel.conf*. Note there must be no *[service_name]* section.

```
exec = /usr/sbin/imapd    execargs = imapd
```

FILES

stunnel.conf

stunnel configuration file

stunnel.pem

stunnel certificate and private key

BUGS

Option *execargs* does not support quoting.

RESTRICTIONS

stunnel cannot be used for the FTP daemon because of the nature of the FTP protocol which utilizes multiple ports for data transfers. There are available SSL enabled versions of FTP and telnet daemons, however.

NOTES

INETD MODE

The most common use of **stunnel** is to listen on a network port and establish communication with either a new port via the connect option, or a new program via the *exec* option. However there is a special case when you wish to have some other program accept incoming connections and launch **stunnel**, for example with *inetd*, *xinetd*, or *tcpserver*.

For example, if you have the following line in *inetd.conf*:

```
imaps stream tcp nowait root /usr/sbin/stunnel stunnel /etc/stunnel/imaps.conf
```

In these cases, the *inetd*-style program is responsible for binding a network socket (*imaps* above) and handing it to **stunnel** when a connection is received. Thus you do not want **stunnel** to have any *accept* option. All the *Service Level Options* should be placed in the global options section, and no *[service_name]* section will be present. See the *EXAMPLES* section for example configurations.

CERTIFICATES

Each SSL enabled daemon needs to present a valid X.509 certificate to the peer. It also needs a private key to decrypt the incoming data. The easiest way to obtain a certificate and a key is to generate them with the free *OpenSSL* package. You can find more information on certificates generation on pages listed below.

Two things are important when generating certificate-key pairs for **stunnel**. The private key cannot be encrypted, because the server has no way to obtain the password from the user. To produce an unencrypted key add the *-nodes* option when running the **req** command from the *OpenSSL* kit.

The order of contents of the *.pem* file is also important. It should contain the unencrypted private key first, then a signed certificate (not certificate request). There should be also empty lines after certificate and private key. Plaintext certificate information appended on the top of generated certificate should be discarded. So the file should look like this:

```
    -----BEGIN RSA PRIVATE KEY-----    [encoded key]    -----END RSA PRIVATE KEY-----    [empty line]    -----BEGIN
CERTIFICATE-----    [encoded certificate]    -----END CERTIFICATE-----    [empty line]
```

RANDOMNESS

stunnel needs to seed the PRNG (pseudo random number generator) in order for SSL to use good randomness. The following sources are loaded in order until sufficient random data has been gathered:

- The file specified with the *RNDfile* flag.
- The file specified by the RANDFILE environment variable, if set.
- The file .rnd in your home directory, if RANDFILE not set.
- The file specified with '--with-random' at compile time.
- The contents of the screen if running on Windows.
- The egd socket specified with the *EGD* flag.

- The egd socket specified with '--with-egd-sock' at compile time.
- The /dev/urandom device.

With recent (>=OpenSSL 0.9.5a) version of SSL it will stop loading random data automatically when sufficient entropy has been gathered. With previous versions it will continue to gather from all the above sources since no SSL function exists to tell when enough data is available.

Note that on Windows machines that do not have console user interaction (mouse movements, creating windows, etc) the screen contents are not variable enough to be sufficient, and you should provide a random file for use with the *RNDfile* flag.

Note that the file specified with the *RNDfile* flag should contain random data -- that means it should contain different information each time **stunnel** is run. This is handled automatically unless the *RNDoverwrite* flag is used. If you wish to update this file manually, the *openssl rand* command in recent versions of OpenSSL, would be useful.

One important note -- if /dev/urandom is available, OpenSSL has a habit of seeding the PRNG with it even when checking the random state, so on systems with /dev/urandom you're likely to use it even though it's listed at the very bottom of the list above. This isn't **stunnel's** behaviour, it's OpenSSLs.

SEE ALSO

tcpd(8)
access control facility for internet services

inetd(8)
internet 'super-server'

http://stunnel.mirt.net/
stunnel homepage

http://www.stunnel.org/
stunnel Frequently Asked Questions

http://www.openssl.org/
OpenSSL project website

AUTHOR

Michal Trojnara

<*Michal.Trojnara@mirt.net*>

The Ubiquitous Miscellaneous Section

Those things that don't have a proper home yet may end up here. Or they may stay here. I've heard that many answers prefer to be in the miscellaneous sections. Maybe they're just shy.

What are the offical SSL ports?

These are the officially 'registered' ports for various SSL-ified protocols. note that listing them here doesn't mean that they can be used with stunnel.

```
    nsiiops      261/tcp   # IIOP Name Service over TLS/SSL    https        443/tcp   # http protocol over TLS/SSL
smtps        465/tcp   # smtp protocol over TLS/SSL (was ssmtp)    nntps        563/tcp   # nntp protocol over TLS/SSL
(was snntp)    imap4-ssl    585/tcp   # IMAP4+SSL (use 993 instead)    sshell       614/tcp   # SSLshell
ldaps        636/tcp   # ldap protocol over TLS/SSL (was sldap)    ftps-data    989/tcp   # ftp protocol, data, over
TLS/SSL    ftps         990/tcp   # ftp protocol, control, over TLS/SSL    telnets      992/tcp   # telnet protocol
```

```
over TLS/SSL    imaps        993/tcp   # imap4 protocol over TLS/SSL    ircs         994/tcp   # irc protocol over
TLS/SSL    pop3s        995/tcp   # pop3 protocol over TLS/SSL (was spop3)    msft-gc-ssl  3269/tcp  # Microsoft Global
Catalog with LDAP/SSL
```

How do I know which encryption ciphers are available?

The ciphers that are availabre to stunnel (and usable by the '-C' flag) are determined by your OpenSSL library. To list the available ciphers, run the following:

```
openssl ciphers -v
```

How can I delay DNS lookups until connect time?

If you're using Stunnel-4.0 or later, add the following to your Stunnel configuration file:

```
delay = yes
```

If you are using older versions, there are several different patches available for this in the patches directory on this site you may try.

Another option is to launch redir (a TCP redirector) dynamically instead of using the " `-r host:port` " option, like this:

```
-l /usr/bin/redir -- redir --inetd --caddr host --cport port
```

How can I convert a certificate from der format (.cer) to PEM format?

Some institutions that supply certificates will send them to you in der format instead of PEM format. You can use the openssl command line tool to convert from one to the other:

```
openssl x509 -in file.cer -inform d -out file.pem
```

再分享一下我老师大神的人工智能教程吧。零基础！通俗易懂！风趣幽默！还带黄段子！希望你也加入到我们人工智能的队伍中来！
https://blog.csdn.net/jiangjunshow