百万人のMutt
～Mutt活用講座～

滝澤隆史
taki@cyber.email.ne.jp

第12回　通信路の暗号化

普段私たちがメールの送受信を行う際、特に対策を立てていなければ、ネットワーク上には生（平文）のメッセージが流れています。そのため、第三者にメッセージが盗み見られる可能性があります。

これを防ぐために、クライアント側で実施できる対策としては、第三者に見られたくないメッセージを「PGPやS/MIMEで暗号化する」という方法があります。しかし、メールをやりとりするすべての人がPGPなどを使っているわけではありませんし、さらにメールサーバが平文認証しかサポートしていない場合には、認証のセッションにおいてパスワードが盗み見られる危険性があります。そのため、ネットワークの環境や利用形態によっては別の対策が必要になってきます。この対策とは「通信路自体を暗号化する」方法です。

通信路を暗号化する方法には、PPTP、L2TP、IPsecなど、レイヤ3（ネットワーク層）以下のものや、SSH、SSL/TLSなどレイヤ4（トランスポート層）レベル以上のものがあります。このうち、レイヤ3以下のものはMuttとは関係のない話になるので、ここでは無視します。Muttでは、SSHとSSL/TLSに対応しています。そこで今回は通信路の暗号化をテーマとしてMuttでSSHとSSL/TLSを使う方法について紹介します。

本題に入る前にMuttの最新状況をお知らせします。執筆時点での最新版は1.3.28iです。これは1.4の最終ベータ版（予定）です。

SSH

●Muttとssh

メールサーバにSSHでログインできる環境であれば、SSHのポート転送を使って、POP3やIMAPのセッションを暗号化することができます。SSHでクライアントとサーバ間に暗号化したトンネルを確立して、その上にPOP3やIMAPのセッションを載せればよいので、実はMuttが直接SSHに対応している必要はないのです。では「MuttのSSH対応とは何か」ということになりますが、これは「POP3やIMAP4のセッションを開始する前に、sshを自動的に実行して暗号化されたトンネルを確立できる機能が用意されている」ということです。この機能がなければ、Muttの起動前にsshを手動で起動させ、Muttの終了後にもsshを手動で終了させなければならなくなります。

なおこのMuttのSSH対応は、ハードコーディングされているわけではなく、設定変数で指定できるので、SSHと同じような動きをするプログラムなら何でも使えるようになっています。

本記事では、SSHとしてOpenSSH（記事末のResource［1］、［2］を参照）を使うことと、皆さんがSSHを使用したことがあることを前提として説明を行います。

●手動接続

まず、手動でSSHのポート転送を行ってみます。初めてサーバに接続するのであれば、サーバの公開鍵を受け取って保存する作業があるので、必ず手動での接続を一度行ってください。

表1に示す情報をもとに実行例1のコマンドを実行し、リモートサーバにログインします。「-L」はポート転送を指定するオプションで、この例では、localhostの8143番のポートに接続すると、imap.example.orgの143番ポートに転送されるようになります。そのため、IMAPサーバのメールボックスにアクセスするためのIMAP URLは次のようになります。

```
imap://localhost:8143/
```

ここで実行例2のようにしてIMAPフォルダを開いてみます。ログイン名は「foo」とします。

【実行例1】SSHでリモートサーバにログインする

```
$ ssh -L 8143:imap.example.org:143 imap.example.org
```

【実行例2】IMAPフォルダを手動で開く

```
$ mutt -f imap://foo@localhost:8143/INBOX
```

【表1】ポート転送の情報

設定する情報	値
リモートホスト	imap.example.org
リモートポート	143
ローカルホスト	localhost
ローカルポート	8143

接続に成功したらMuttを終了し、sshも終了させます。

●$preconnect

先ほど説明したように、SSHのポート転送を使った接続では、POP3やIMAPの接続前にsshを起動し、切断後にsshを終了させる必要があります。この作業をサーバに接続するたびに行うのは面倒です。そこでMuttに用意されている$preconnectという設定変数を使用します。この設定変数が設定されている場合は「サーバに接続を試みても、接続できなければ$preconnectで指定したコマンドを実行する」という動作を行います。従って、サーバに接続しようとしたときにsshの接続がなければサーバに接続できないため、sshの接続を自動的に行ってくれるようになります。

手動接続のときと同じ情報（表1）で$preconnectを設定する例をリスト1に示します。バックグラウンドで実行するために「-f」オプションを、メッセージの表示を抑止するために「-q」オプションを指定し、実行するリモートのコマンドには「sleep」を指定して自動的に切断されるようにします。

また、sshはMuttのバックグラウンドで実行されるため、秘密鍵のパスフレーズは入力できません。そこで、秘密鍵のパスフレーズの入力を省略するためにssh-agentの設定を行う必要があります。詳しくはコラム「ssh-agent」をご覧ください。

以上の設定が終わったら、実行例2のようにして再度IMAPフォルダを開いてみます。接続

【リスト1】$preconnectの設定例

```
set preconnect="ssh -f -q -L 8143:imap.example.org:143 \
      imap.example.org sleep 20 > /dev/null"
```

に成功したらMuttを終了します。その後、sleepで指定した秒数が経過したらsshが終了するかどうかをpsコマンドで確認してください。特に問題がなければ、リスト2のようにIMAPサーバをlocalhostとして、ポート番号をポート転送で指定した「8143」としてIMAPの設定を行ってください。

●$tunnel

inetdやtcpserverから起動するタイプのPOP3サーバやIMAPサーバは、標準入力からデータを読み取り、標準出力にデータを書き出します*1。これらのサーバの中には、一般ユーザーの権限で起動すると認証が行われたものと見なし、認証なしにそのユーザーのメールボックスにアクセスできるものがあります。そのようなサーバでは実行例3のように、端末上でIMAPのコマンドを入力することでメールボックスにアクセスすることができます。ちなみにこの例ではUW IMAP Server*2を使っています。

一方Muttには、ネットワーク経由でサーバに接続してメールボックスにアクセスする代わりに、先ほど示したサーバなどのコマンドを直接起動して、そのコマンドに対するパイプ経由でメールボックスにアクセスする機能があります。この機能を利用するためには、設定変数$tunnelを設定する必要があります。ローカルにIMAPのメールボックスがある場合は、リスト3のように設定します。リモートにIMAPのメールボックスがあり、SSH経由でアクセスするのであればリスト4のように設定します。このとき、このIMAPのメールボックスにアクセスするには、IMAPサーバのホスト名として、実際のIMAPサーバではなく「localhost」を指定します。そのため、リスト5に示すIMAP URLを使用した設定になります。

このように設定してMuttを起動すると、実行例3のようにIMAPのメールボックスにアクセスすることができます。

【リスト2】IMAPフォルダの設定

```
set spoolfile=imap://foo@localhost:8143/INBOX
set folder=imap://foo@localhost:8143/
```

【リスト3】ローカルのIMAPサーバへアクセスする

```
set tunnel="/usr/local/sbin/imapd"
```

【リスト4】リモートのIMAPサーバへSSH経由でアクセスする

```
set tunnel="ssh -q imap.example.org \
      /usr/local/sbin/imapd"
```

【リスト5】IMAP URLの指定

```
set spoolfile=imap://localhost/INBOX
set folder=imap://localhost/
```

【リスト6】乱数の種となるファイルをMuttで指定する

```
set entropy_file=~/.mutt/random_seed
```

【実行例3】端末から直接サーバを起動し、メールボックスへアクセスする

```
$ /usr/local/sbin/imapd
* PREAUTH [CAPABILITY IMAP4 IMAP4REV1 STARTTLS NAMESPACE IDLE MAILBOX-REFERRALS
SCAN SORT THREAD=REFERENCES THREAD=ORDEREDSUBJECT MULTIAPPEND] imap.example.org
 IMAP4rev1 2000.284 at Sat, 23 Mar 2002 13:52:04 +0900 (JST)
a0001 LIST "" "~/%"
* LIST (\NoSelect) "/" ~/
* LIST (\NoInferiors \Marked) "/" ~/newfolder
* LIST (\NoInferiors \Marked) "/" ~/Trash
* LIST (\NoInferiors \UnMarked) "/" ~/&ZeVnLIqe-
* LIST (\NoSelect) "/" ~/test
* LIST (\NoInferiors \UnMarked) "/" ~/mbox
a0001 OK LIST completed
a0002 LOGOUT
* BYE imap.example.org IMAP4rev1 server terminating connection
a0002 OK LOGOUT completed
```

【実行例4】疑似乱数の種

```
$ perl -e 'for (1..20) {system("ps auwx|openssl sha1")}' > ~/.mutt/random_seed
$ chmod 600 ~/.mutt/random_seed
$ cat ~/.mutt/random_seed
0c4b4d60e9a66a7fc50e82c972d43fbaa021626b
b2c8ce9f9c5f5d989069dc63520458e6f9eb1469
c12cc5595be98d72ce361725b652fd72c87be887
b83f2528982b26d350641b244f5b41223fb6eed8
b8002753a53daec0408679b462d3c3f608f62ee0
          ⋮
```

SSL/TLS

●準備

現在使っているMuttがSSLに対応しているかどうかは次のコマンドで確認できます。

```
$ mutt -v | grep SSL
```

SSLに対応している場合には「+USE_SSL」と表示されます。「-USE_SSL」と表示されたらSSLには対応していないため、Muttのコンパイル時のconfigureスクリプトに、--with-sslオプションを付けてコンパイルとインストールを行ってください。なお、OpenSSL（[4]、[5]）が必要ですので、もしインストールされていないようでしたら、インストールしてください。

●乱数の種

SSLのライブラリ関数を初期化する際には乱数の種が必要です。Linux 2.0以降の場合には、乱数生成器である/dev/urandomデバイスがあるため特に設定は必要ありません。/dev/urandomを持っていない場合には、The Entropy Gathering Daemon（EGD、[6]）を利用してください。また、EGDを導入できない場合には、リスト6に示すように、Muttの設定変数$entropy_fileに乱数の種となるファイルを指定します。

擬似的な乱数の種としては、実行例4のように、ある程度の乱雑さが含まれるものを集めてください。ただし、このようにして生成された疑似乱数の種は、乱雑さとして保証されるわけではないので、定期的に作り直すようにしてください。

●STARTTLS

SSLを使う場合は、表2のように、通常POP3やIMAPで使用するポート番号とは異なるポート番号を使用します。一方、STARTTLSコマンドと共にTLSを使う場合は、先ほど示したSSLとは異なり、通常使用するポート番号のまま使用します。

【表2】プロトコルとポート番号

プロトコル	SSLなし	SSLあり
POP3	110（pop）	995（pops）
IMAP	143（imap）	993（imaps）

注）括弧内はURL表記でのプロトコル名

＊1　inetdやtcpserverがネットワークから読み取ったデータをサーバの標準入力へ送り、サーバの標準出力から受け取ったデータをネットワークへ書き出します。
＊2　UW IMAP Serverについて詳しくは、「IMAP Information Center」（[3]）を参照してください。

さてここで、「TLS」と「STARTTLS」というあまり馴染みのない言葉が出てきましたので、簡単に説明しておきましょう。TLSとは「Transport Layer Security」の略で、SSL 3.0のプロトコルの仕様をベースにして作られたプロトコルです。現在のバージョンは1.0です。STARTTLSは、SMTP、POP3、IMAP、ACAPなどで拡張された機能で、TLSでセッションの暗号化を開始するためのコマンドです。SSLでは暗号化されたセッションの上にPOP3やIMAPのセッションが載るのに対して、STARTTLSの場合は、通常のPOP3やIMAPのセッションの途中でSTARTTLSコマンドによって暗号化を開始します。そのため、使用するポート番号は通常のままになります。

以上のことから、STARTTLSが使える場合と使えない場合とでは使用するポート番号が異なるため、IMAP/POP URLの表記が変わります。そのため、サーバ側でSTARTTLSが使えるかどうかを確認します。

まず、通常通りのPOP3/IMAPの設定を行ってください。POP/IMAP URLのプロトコル名には、通常のpopやimapを記述します。パスワードは記述しないでください。この状態でサーバに接続したときにSSL証明書検査の画面（画面1）が表示されたらSTARTTLSは使用できます。逆に、何も表示されずにパスワードが要求されたら、STARTTLSは使用できません。

STARTTLSが使えなかったら、POP/IMAP URLのプロトコル名として、POP3/IMAP over SSLを示すpopsやimapsを記述します。この状態でサーバに接続できればSSLが使用できます。逆に、接続できなかったらSSLは使用できません。

以上の結果から、STARTTLSへの対応状況に応じてIMAP/POP URLの表記を変えてください。なお、SSL/TLSのどのプロトコルを使うかは、表3のような設定変数がありますが、すべてデフォルトで有効になっているので、特に気にせずそのまま使ってよいでしょう。

●証明書の確認

サーバに接続すると、SSL証明書検査の画面が表示され、次のように尋ねられます。

```
(r)拒否 (o)今回のみ承認
```

SSL証明書の内容を確認して、証明書が正しいと判断できれば「o」（小文字のオー）を入力します。これはサーバに接続するたびに尋ねられますが、毎回証明書を確認するのは大変です。そこで、承認した証明書を保存して、次回以降は、受け取った証明書と保存した証明書が同じであれば、自動的に承認するようにできます。これを行うためには、設定変数`$certificate_file`に証明書を保存するファイル名を指定します。設定例を**リスト7**に示します。

この設定を行ってから、Muttを再起動すると、今度は次のように尋ねられます。

```
(r)拒否 (o)今回のみ承認 (a)いつでも承認
```

ここで「a」と入力すると、証明書が保存され、次回以降は尋ねられなくなります。なお、あらかじめ証明書が入手できるようでしたら、入手した証明書を設定変数で指定したファイルに追加することもできます。

このように証明書を保存した状態で、もしSSL証明書検査の画面が出るようでしたら、中間一致攻撃（man-in-the-middle attack：なりすまし）が行われ、偽の証明書が送られてきている可能性があります。この場合は、その証明書が正しいことを確認できない限り拒否してください。

また、STARTTLSを使うように設定していても、CAPABILITYコマンド（対応している機能を表示させるコマンド）に対して、「サーバがSTARTTLSに対応していない」と返答した場合、MuttはTLSで暗号化せずにログインしようとします。このときTLSを使わないため、証明書の検査も行われません。

そのため、中間一致攻撃が行われていて、意図的に「STARTTLSに対応していない」と返答のある場合は、メッセージが盗まれるだけでなく、平文認証を行っていれば、パスワードまで盗まれることになります。

このような中間一致攻撃を防ぐためには、設定変数`$imap_force_ssl`を設定して、証明書の検査を確実に行う必要があります。この設定変数は、デフォルトでは無効になっているため、次のようにして有効にしてください。

```
set imap_force_ssl=yes
```

この設定変数は、IMAPのみ適応されます。

POP3の場合、この`$imap_force_ssl`設定変数に該当するものがないので、中間一致攻撃を受ける危険性があります。回避方法としては、TLSを使わず、SSLのみを使うことくらいしかありません。そのため、POP URLのプロトコル名として「`pops`」を使ってください。

以上で設定は完了です。これでSSL/TLSのを使うことができます。

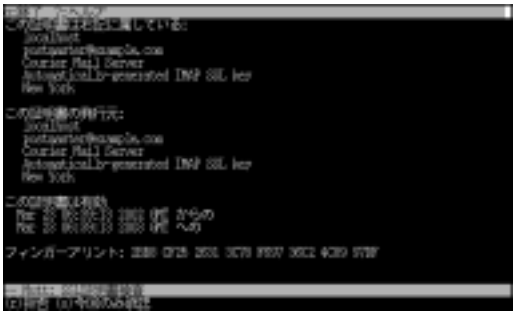
fetchmail

fetchmail（**[7]**）も、Muttと同じようにSSHやSSLに対応しています。Muttとは直接は関係ありませんが、利用している人も多いと思いますのでここで紹介します。

●SSH

fetchmailでSSHを使うには、設定ファイルに記述するユーザーオプションとして`preconnect`を使用します。Muttの設定変数

【画面1】SSL証明書検査の画面



【表3】SSL/TLSのプロトコルの設定変数

設定変数	プロトコル等	デフォルト値
ssl_use_sslv2	SSLv2	yes
ssl_use_sslv3	SSLv3	yes
ssl_use_tlsv1	TLSv1	yes
ssl_starttls	STARTTLSコマンド	yes

【リスト7】承認の自動実行

```
set certificate_file=~/.mutt/certificates
```

【リスト8】.fetchmailrcの設定例（SSH）

```
poll pop.example.org via localhost port 8110 with proto pop3
      user foo
      preconnect "ssh -f -L 8110:pop.example.org:110 pop.example.org sleep 20 >/dev/null";
      mda "maildrop ~/.mailfilter/foo"
```

$preconnectとまったく同じ役割をするので、コマンドも同じものを記述します。

.fetchmailrcの設定例をリスト8に示します。サーバオプションとして、localhostの8110番ポートに接続するように記述してあることに注意してください。

● SSL

fetchmailでSSLを使うには、fetchmailのコンパイル時のconfigureスクリプトに、オプション「--with-ssl=＜DIR＞」を付けてコンパイルとインストールを行ってください。＜DIR＞には、OpenSSLがインストールされているディレクトリを指定します。例えば、/usrにインストールされているのであれば「--with-ssl=/usr」を付けます。RPMなどのパッケージシステムでインストールされているfetchmailには、SSLを使うようにコンパイルされていないものもあるので注意してください。

SSLを使うためには設定ファイルにユーザーオプションとして「ssl」と記述するだけです。ポート番号の指定は特に必要なく、POP3であれば995番に、IMAPであれば993番に接続されます。

.fetchmailrcの設定例をリスト9に示します。この例では、ユーザーオプションとしてsslfingerprintも設定しています。このオプションは、サーバの公開鍵のフィンガープリントを指定することによって、中間一致攻撃を防ぐことができます。

また、証明書を検査するためのオプションとしてsslcertckとsslcertpathがあります。このオプションは、サーバの証明書が信用された認証局から署名されたものであるかどうかを検査します。

SSLのオプションに関しては、マニュアルやヘルプには記述されていないものがあるので、fetchmailの付属文書のREADME.SSLを読んで調べてください。なお、執筆時点での安定バージョン5.9.0ではSTARTTLSには対応していないようです。

最後に

SSHは、サーバへのログインアカウントを実際に持っていないと使えません。最近のメールサーバでは、仮想メールボックスを使用しているケースが多いので、SSHを使いたくても使えないケースが多いでしょう。

そのため、一般的に「実用性が高いのはSSLかな」と思います。SSLはサーバの実際のログインアカウントの有無に関わらず使用できるからです。POP3サーバやIMAPサーバで使用しているプログラムが直接SSLに対応していなくても、サーバ側でstoneやstunnelのようなプログラムと組み合わせればSSL対応のサーバが簡単に構築できます。サーバがSSLに対応していない場合は、サーバの管理者と親しければ、突っついてみるのもよいでしょう。

【リスト9】.fetchmailrcの設定例（SSL）

```
poll pop.example.org proto pop3
      user foo
      ssl
      sslfingerprint "2B:B8:CF:25:26:31:3C:78:F6:97:36:C2:4C:89:97:DF"
      mda "maildrop ~/.mailfilter/foo"
```

Column ssh-agent

SSHで、公開鍵認証方式（RSA/DSA）を用いて接続する際には、秘密鍵のパスフレーズの入力が必要です。しかし、Muttからsshをバックグラウンドで呼び出すような場合は、パスフレーズの入力ができません。そのため、代わりにパスフレーズを入力してくれるプログラムとしてssh-agentを使います。

【実行例A】ssh-agentの起動

```
$ eval `ssh-agent`
Agent pid 8848
```

ssh-agentは、単独で起動するのではなく、実行例Aのようにして起動させます。なお、使用しているLinuxのディストリビューションによっては、ログイン時に自動で起動するようにあらかじめ設定されているものもあります。

次に、利用する秘密鍵のパスフレーズをssh-addで登録します。SSHプロトコルv2用のRSAの秘密鍵を登録する例を実行例Bと実行例Cに示します。ssh-askpassがインストールされていて、かつX Window上で使用している場合は、実行例Cのように実行してください。GUIの入力画面が表示されたら、パスフレーズを入力してください。（滝澤隆史）

【実行例B】端末のみの環境の場合

```
$ ssh-add $HOME/.ssh/id_rsa
Enter passphrase for /home/foo/.ssh/id_rsa:
Identity added: /home/foo/.ssh/id_rsa (/home/foo/.ssh/id_rsa)
```

【実行例C】X Window環境の場合

```
$ ssh-add < /dev/null
Identity added: /home/foo/.ssh/id_rsa (/home/foo/.ssh/id_rsa)
```

Resource

[1] OpenSSH
http://www.openssh.org/ja/

[2] OpenSSH Download（ミラー）
ftp://ring.gr.jp/pub/OpenBSD/OpenSSH/portable/

[3] IMAP Information Center
http://www.washington.edu/imap/

[4] OpenSSL
http://www.openssl.org/

[5] OpenSSL Download（ミラー）
ftp://ring.gr.jp/pub/net/openssl/

[6] EGD: The Entropy Gathering Daemon
http://egd.sourceforge.net/

[7] Fetchmail Home Page
http://www.tuxedo.org/~esr/fetchmail/