サーバー設定におけるいくつかの留意事項

総合技術センター
情報システム技術分野　木戸　崇博(Takahiro Kido)
計測・制御技術分野　山中　卓也(Takuya Yamanaka)

1.　はじめに

平成23年度にいくつかのサーバ更新・設定変更を行った。この技術報告ではその作業の中から,特筆すべきと思われる以下の3点について記述する。

・パケットフィルタの設定
・postfixのaliases設定
・openssslのコマンドの使い方・認証局作成からサーバの秘密鍵作成

今回のサーバ作業をしたOSはすべてFreeBSDである。また，今回の報告では結果だけではなく，作業の途中で試行錯誤した過程も記載するようにした。

2.　パケットフィルタの設定

セキュリティの観点から，サーバがping, tracerouteといったコマンドに対する応答を無条件で返すのはよくないため，パケットフィルタを用いて規制を行うことにした。

・pingに対しては応答する
・tracert（もしくはtraceroute）には応答しない。

というような規制を入れることを予定していた。

今回のパケットフィルタ設定のためのネットワーク構成は以下の図１のようなものである。

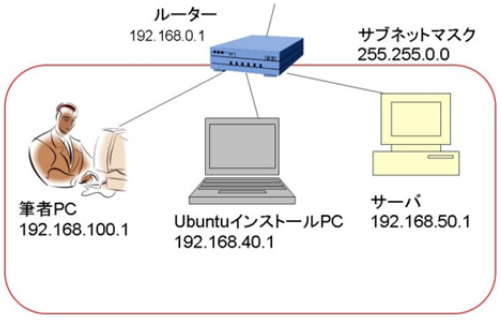

図１　作業時のネットワーク構成

ルータ(192.168.0.1)の下にサーバ(192.168.50.1)を置く。サーバへのアクセス用に Ubuntu をインストールしたノートパソコン（192.168.40.1）と，筆者の Windows パソコン（192.168.100.1）がある。ルータによって構成されたプライベートネットワークのサブネットマスクは 255.255.0.0 である。

2.1 rc.conf の設定

FreeBSD において，パケットフィルタを使用する場合,/etc/rc.d/pf を実行すればよい。サーバ起動時にパケットフィルタを実行するためには/etc/rc.conf に以下の設定を記述する。

```
pf_enable=“YES”
pf_rules=“/etc/pf.conf”
pflog_enable="YES"
pflog_logfile="/var/log/pflog"
```

2.2 pf.conf の設定

パケットフィルタを使用する場合, pf.conf に設定を記述すれば, そのルールが適用される。まず, 筆者の PC から ping だけ通して, tracert はとおさないことを想定し，以下のような pf.conf を記述した。

```
ext_if = "bge0"
block on $ext_if inet proto icmp all
pass on $ext_if inet proto icmp from ¥
 192.168.100.0/24 icmp-type 8 ¥
 code 0 keep state
```

２行目の block は，ICMP の信号をすべて止める。３行目の pass で，筆者 PC（192.168.100.1）からの ICMP の echoreq を通す，という設定をしている。

この設定後，筆者の PC からコマンドを打ってみたが，ping も tracert も両方通ってしまった。

調べてみたところ，筆者の PC から使用した tracert コマンドは，ICMP の echoreq を使う。ping も同様に ICMP の echoreq を使うので，「ping は通すけど，tracert は通さない」という設定は難しいことがわかった。なお，UNIX の traceroute コマンドは UDP のリクエストを送信する。

また，それぞれのコマンドに対して，応答する信号の種類は異なる。tracert は ICMP でリクエストをし，受けたサーバは ICMP でレスポンスを返す。ping も ICMP でリクエストをし，受けたサーバは ICMP でレスポンスを返す。traceroute は UDP でリクエストをし，それを受けたサーバは ICMP でレスポンス返す。これを以下の表 1 に示す

表 1　各コマンドのリクエスト・レスポンス信号の種類

種類	リクエスト	レスポンス
ping	ICMP	ICMP
tracert	ICMP	ICMP
traceroute	UDP	ICMP

この時，Ubuntu の PC から，traceroute コマンドを打ってみたが，こちらは通らない。これは，パケットフィルターで許可されたネットワークが，192.168.100.0/24 であり，Ubuntu の PC の IP アドレス 192.168.40.1 は許可されていないからである。

ping だけ通して tracert は通さない，とするのはあきらめ，限られた IP アドレスからだけ，ping および tracert(traceroute)を通すようにすることにした。

まず，筆者 PC からの ping および tracert は通すが，それ以外は通さない，という設定をしてみる。少し macro などを追加して，pf.conf は以下のように変更した。

```
#macros
ext_if = "bfe0"
allow_nets = "{ 127.0.0.0/8 }"
test_nets = "{ 192.168.100.0/24,  ¥
 192.168.50.1 }"

block on $ext_if inet proto icmp all
pass on $ext_if inet proto icmp from ¥
 $test_nets icmp-type 8 code 0 keep ¥
 state
```

test_nets という macro に筆者 PC の IP アドレスがあり，pass の行で，それが許可される，という設定になっている。これで筆者 PC から ping および tracert を試すと，サーバにまで通った。Ubuntu の PC から ping と traceroute はとおらない。

今度は Ubuntu からの ping と traceroute コマンドは通して，筆者の PC からの ping と tracert は通さない設定にしてみる。

```
#macros
ext_if = "bfe0"
allow_nets = "{ 127.0.0.0/8}"
test_nets = "{ 192.168.40.0/24, ¥
 192.168.50.1 }"

block on $ext_if inet proto icmp all
pass on $ext_if inet proto icmp from ¥
 $test_nets icmp-type 8 code 0 ¥
 keep state
```

これで，筆者の PC から ping および tracert を試す。通らないので設定はうまくできたと思われた。

しかし Ubuntu の PC から試すと，ping は通るのだが，traceroute がなぜか通らなくなってしまう。

表 1 で示したように，tracert は Request に ICMP を使うが，traceroute は UDP を使う。しかし，traceroute は “-I”というオプションを使用すると，tracert と同様，Request に ICMP を使う。Ubuntu の PC からこのコマンドを試してみた。すると今度は，サーバからの応答があった。”traceroute -I”は通る，ということである。

pf.conf に，UDP について記述をしていないのに，なぜか traceroute が通らない。しかし，ICMP の request をする traceroute –I は応答がある。という理解しがたいと思える

現象のため，ここでかなりの時間を費やすこととなった。

そして，ログ解析などの結果，以下のような流れで通信されていることがわかった。

1.ubuntu から traceroute する。

2.UDP-Request がサーバへ行く。

3.ファイアーウォールは UDP 信号は気にせず通す。

4.サーバは ICMP で応答信号を返す。

5.ファイアーウォールは pf.conf の以下の記述により，ICMP はすべてブロックするので，ICMP の応答が ubuntu に返らない。

```
block on $ext_if inet proto icmp all
```

すなわち，サーバから「返される方」の ICMP の信号が，block されていることがわかった。

サーバから返される ICMP の信号が block されるなら，なぜ tracert コマンドや traceroute -I は成功するのだろうか？これは，”keep state”という仕組みのためであることがわかった。

以下で，keep state について簡単に説明をする。

keep state はある信号を許可したとき，それとかかわりのある一連の信号すべてを許可する，という仕組みである。

tracert もしくは traceroute -I では，ICMP の Request 信号がファイアウォールの

```
pass on $ext_if inet proto icmp ¥
 from $test_nets icmp-type 8 code 0 ¥
 keep state
```

の記述で許可されると，それに対する応答の ICMP 信号はかかわりのある一連の信号として許可されるため，以下の記述

```
block on $ext_if inet proto icmp all
```

があっても無視して，通るのである。

この keep state の性質について理解をすれば，traceroute -I はサーバからの応答があるのに対し，traceroute はサーバとの応答がないのも理解ができる。traceroute は UDP で Request をするが，サーバは UDP ではなく，ICMP で信号を返しているため，「かかわりのある一連の信号」とはみなされず，keep state の対象とならない。そのため“block on $ext_if inet proto icmp all”という記述に引っかかり，そこで信号がストップしたと考えられる。

その後いろいろ試行した結果，pass もしくは block に in もしくは out という方向を指定する助詞がないのは，どちら側に向かっていく通信を制限しているのか，分かりにくくなるとの反省と，log 出力をしたほうがよいだろうとの考えから，以下のような記述となった。

```
#macros
ext_if = "bfe0"
allow_nets = "{ 127.0.0.0/8}"
 test_nets = "{ 192.168.40.0/24, ¥
 192.168.50.1 }"

block in log on $ext_if inet proto ¥
 icmp all
block out log on $ext_if inet ¥
 proto icmp all
pass in log on $ext_if inet proto ¥
 icmp from $test_nets keep state
pass out log on $ext_if inet ¥
 proto icmp to $test_nets
```

block，もしくは pass の行で in と書かれている行はサーバに入ってくる（in）信号に対してのものであり，out はサーバから出ていく信号に対してのものである。

この設定で，以下のことを確認した。

1) $test_nets の 192.168.40.0/24 という IP アドレスをそのままにして，ubuntu の PC からの traceroute は通すが，筆者の PC からの tracert はとおさないことを確認。
2) $test_nets の 192.168.40.0/24 という IP アドレスを 192.168.100.0/24 にして，ubuntu からの traceroute コマンドはとおさないが，筆者の PC の tracert は通す。

2.3 pflog のログの見方

pf のログを見るには，まず，ルール上 /etc/pf.conf に"log"という記述がなければならない。

記述があれば，/etc/rc.conf に記載された /var/log/pflog にログが保存される。しかしログファイルの中身がテキストファイルではないので tcpdump コマンドを使って参照する。

```
# tcpdump -n -e -ttt -r /var/log/pflog
```

リアルタイムでログを確認することもできる。その場合は，以下のようにする。

```
# tcpdump -n -e -ttt -i pflog0
```

なお, オプション部分にある t の数を“-tttt”4 つにすると，時刻表記になって非常に見やすくなる。

3. Postfixのaliases設定

今度は，ウェブ・メールサーバの更新作業を行った。その際，メールサーバソフトウェアはPostfixを使うことにした。そのインストール・設定のときに, aliases.dbというPostfixのaliasデータベースの場所や, そのaliasデータベースを生成するnewaliasesコマンドが２つできるので，どちらが使われるのか，といったことがすぐにはわからなかった。

ここでは，その不明な点およびそれに対して，どのように設定したかを記述していく。

3.1 newaliasesコマンド

postfix の設定ファイルは main.cf である。そこに各種設定を記述していく。

main.cf の中で, aliases については以下のように書かれている。

```
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
```

また，/etc/ailases とリンクがはられた /etc/mail/aliases という同じファイルがある。

この aliases というファイルはメール転送の設定を記述するのに使用しているが, これは人間が使用しているだけであり, コンピュータは aliases.db というファイルを参照している。この aliases.db を aliases ファイルから作成するコマンドが newaliases である。

この newaliases の場所は以下のようになっている。

```
>which newaliases
/usr/bin/newaliases
```

これは FreeBSD に最初から存在する newaliases コマンドである。マニュアルによると，newaliases コマンドを実行すると，sendmail コマンドの"sendmail -bp"を起動する，とある。

そこで, ls コマンドで/usr/bin/newaliases コマンドを調べてみた。

その結果を以下に示す。

```
$ ls -l /usr/bin/newaliases
lrwxr-xr-x  1 root  wheel  21 Feb 18 ¥
 2011 /usr/bin/newaliases ->  ¥
 /usr/sbin/mailwrapper
```

上記のように，newaliases は mailwrapper という別のコマンドにリンクされていることがわかった。

この mailwrapper 実行時に,/etc/mail の下にある mailer.conf が参照される。

mailer.conf は以下のようになっていて

```
sendmail      /usr/libexec/sendmail/sendmail
send-mail     /usr/libexec/sendmail/sendmail
mailq        /usr/libexec/sendmail/sendmail
newaliases    /usr/libexec/sendmail/sendmail
hoststat     /usr/libexec/sendmail/sendmail
purgestat     /usr/libexec/sendmail/sendmail
```

実際に動く sendmail コマンドが指定されている。newaliases の場合，/usr/libexec/sendmail/sendmail が実行される。この sendmail コマンドは，FreeBSD に

デフォルトでインストールされているメールサーバソフトウェアSendmailのものである。

しかし，newaliasesコマンドはもう一つ存在する。Postfixをインストールしたときに，/usr/local/binの下にインストールされる，Postfixのnewaliasesコマンドである。今回のメールサーバ作成作業では,こちらのnewaliasesコマンドを使う。

3.2 aliases.dbが作成される箇所

Sendmailのnewaliasesコマンドである/usr/bin/newaliasesコマンドを使用した際,/etc/mailの下にaliases.dbを作成する（この場所はSendmailの設定ファイルであるsendmail.cfというファイルで決まる）。

しかし，Postfixのmain.cfでは，/etcの下にあるaliases.dbを参照する設定になっている。

```
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
```

前述したように，aliasesファイルはリンクが張られているがデータベース本体の/etc/mail/aliases.dbと/etc/aliases.dbはリンクがはられてないので,/etc/mail/aliases.dbが存在しても，postfixはそれを参照しない。

Postfixにそれを参照させたければ,/etc/mail/aliases.dbを使用するように,Postfixのmain.cfを変更しなければならない。以下の様に設定する。

```
alias_maps = hash:/etc/mail/aliases
alias_database ¥
 = hash:/etc/mail/aliases
```

しかし,今回はSendmailのものではなく,Postfixのnewaliasesコマンドを使いたい。

Postfixのnewaliasesコマンドを,lsコマンドで調べた結果を示す。

```
$ ls -l /usr/local/bin/newaliases
lrwxr-xr-x  1 root  wheel  32 Oct  6 ¥
 14:49 /usr/local/bin/newaliases ¥
 -> ../../../usr/local/sbin/sendmail
```

このnewaliasesは，Postfixのmain.cfを参照して実行される。

Postfixをインストールした直後のデフォルトのmain.cfにおいては，以下のように

```
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
```

/etc/alaisesを参照して，/etc/aliases.dbを作成する設定である。

しかし，“newaliases”とだけコマンドを打つと，Sendmailの/usr/bin/newaliasesのコマンドが使用されてしまう。Postfixのnewaliasesコマンドを使用するには,“/usr/local/bin/newaliases”と，絶対パスで使用しなければならない。それは非常に煩雑であり，また，絶対パスで記述しなければならないことを忘れることも考えられる。

その対策として，/etc/mail/mailer.confの記述を変更する。/usr/libexec/sendmail/sendmailをPostfixの/usr/local/sbin/sendmailにするのである。変更の際，newaliasesコマンドだけでなく,mailer.confに記述されている他のコマンドも，/usr/local/sbin/sendmailを使うようにする。

/etc/mail/mailer.confの設定は以下のようになる。

```
sendmail  /usr/local/sbin/sendmail
send-mail /usr/local/sbin/sendmail
mailq     /usr/local/sbin/sendmail
newaliases /usr/local/sbin/sendmail
hoststat  /usr/local/sbin/sendmail
purgestat /usr/local/sbin/sendmail
```

これで，newaliasesを実行した際，main.cfのデフォルトの設定どおり，/etcの下にaliases.dbが作成される。

4. opensslのコマンドの使い方･認証局作成からサーバの秘密鍵作成

メールサーバの設定において考慮しなけれ

ばならないことの一つが，クライアントのアクセスである。セキュリティを第一に考えるならば，職場の限られた場所からのみアクセスを許可すればよい。しかし，ユーザーは多くの場合，どこからでもアクセスできることを希望する。

今回のサーバ設定においても，ユーザが安全に，なるべくあらゆる場所からアクセスできるようにするため，POP over SSLの設定をした。その際，いくつかの不明な点が生じた。

ここでは，POP over SSLの設定時に気づいた，openssl コマンドの使い方と，SSLを使うときに必要となる認証局の作成等についてまとめることにする。

4.1　SSLの原理

POP3 over SSLとは，ユーザがメールサーバに届いたメールを受け取るPOP3というプロトコルをSSL（暗号化された経路）を用いて行うものである。そのため，ユーザとサーバの間の通信は安全になる。

SSL通信を実現するためには，サーバ管理者は認証局・証明書・秘密鍵・公開鍵等を作成しなければならない。これらを用いて，どのようにSSL通信が実現されるかを図2に示す。

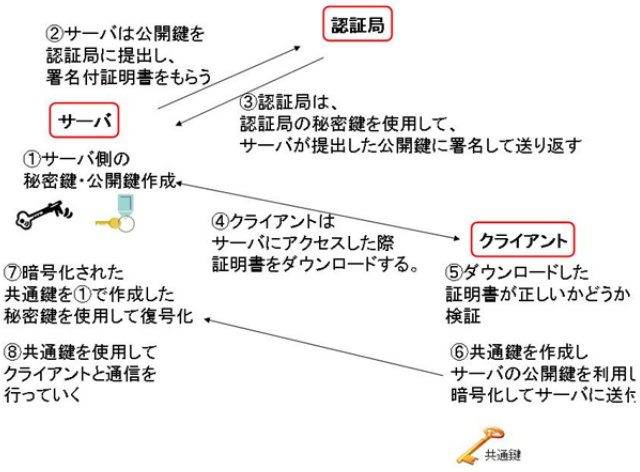

図2 サーバと認証局とクライアントの関係

SSL通信を行うためには，まず証明書を用意する必要がある。そこで，サーバ管理者は，作成した公開鍵を認証局に送付し，署名付の公開鍵（署名が付いた時点で証明書とも呼ばれる。以下，証明書。）を準備する。今回の作業においては，認証局もサーバ管理者が作成した。

サーバが準備した証明書を，クライアント側のブラウザやメーラが，そのソフトウェアに，最初から入っている認証局の公開鍵を利用して復号化する。復号化できれば，正式な認証局から署名してもらったサーバ公開鍵と判断される。今回の設定では，前述したとおり，認証局もサーバ管理者が作成したので，証明書を受け入れるかどうかを確認するウィンドウが，クライアント側のパソコンに現れる。

サーバからの証明書をクライアントが受け入れると，クライアントはまずそれ以降の通信に使う共通鍵を作る。その共通鍵をさきほど受け入れたサーバの証明書にふくまれるサーバの公開鍵を使って暗号化して，サーバに送り返す。サーバはその共通鍵を自身の秘密鍵を使って復号化し，それができれば，以後はその共通鍵を使用して，共通鍵暗号方式で，クライアントと通信を行う。

サーバが公開鍵を認証局に送付する際，留意することがある。その公開鍵が認証局への署名要求段階なのか，それとも認証局の署名が終わり証明書になった段階かを把握しておくことである。見分けるには，公開鍵の1行目を見る。

公開鍵の1行目が

-----BEGIN CERTIFICATE REQUEST-----

となっている場合，署名要求段階であり，

-----BEGIN CERTIFICATE-----

となっている場合は，署名が終わっている。

4.2　openssl コマンドの使い方と例

前述したように，サーバ管理者は認証局・秘密鍵・公開鍵などを作らなければならない。このために，openssl コマンドを使う。

openssl コマンドは，そのあとに続く文字列（＝2つめの引数）によって，行う作業が変わ

る。manコマンドで調べる場合，openssl の２つめの引数の文字を入力することで，どのようなコマンドであるかわかる。たとえば，openssl req .....というopensslのreqコマンドを調べたいときには，

```
$ man req
```

とする。以下，いくつかの例を示す

例1.
```
openssl req -new -x509 -nodes ¥
-days 1 –out myhost.pem ¥
–keyout myhost.key
```

openssl reqなので，証明書の署名要求を作るのだが，-x509とついているので，その証明書に署名までしてもらう。また，サーバの秘密鍵も作成する。

myhost.pem という名前の証明書を出力する。myhost.keyという名前の秘密鍵も出力する。

-nodes　は暗号化しないためのオプションである。そのため，秘密鍵にかけるパスフレーズは無い。また，パスフレーズ入力の要求もない。

例2.
```
openssl genrsa -out server.key 1024
```

openssl genrsaなので，rsa形式の秘密鍵であるserver.keyが作成される。公開鍵は作られない。

例3.
```
openssl req -new -days 1 ¥
 –key server.key -out server.csr
```

openssl reqなので，証明書要求を作る。使用するキーは，-keyの後にあるserver.keyである。なお，このserver.keyは秘密鍵である。この秘密鍵に対応する公開鍵から，証明書要求を作ることに注意。今回の例では，-x509が無いので，できた証明書は，
---BEGIN CERTIFICATE REQUEST-----
という一行が頭にある。これは，証明書署名要求の段階。まだ証明書になってない。その証明書署名要求段階のファイル名は，server.csrとして出力されている。

例4.
```
openssl x509 -in server.csr ¥
 -out server.crt -req ¥
 –signkey server.key -days 365
```

openssl x509 なので，署名要求にこたえた署名付の公開鍵（=証明書）が作成される。

使用する証明書署名要求のファイルは，-inの後ろにあるserver.csrである。署名後のファイルは，-outの後ろにあるserver.crtである。-reqオプションがあるので，-inの後のファイルを証明書署名要求であるとみなす。もし-reqオプションがなければ，このopenssl x509コマンドは-inの後は証明書であると判断することに注意。

4.3　認証局の作成

認証局の作成に際し，ネットで調べたホームページを参照しながら行った[1)]。

そのホームページによると，前準備として，まず/etc/ssl/openss.cnfのdirに書かれているディレクトリを確認し，その下に，certs，private，crl，newcertsという名前のディレクトリを作成する。確認したところ，demoCAという名前の下に，それらのディレクトリを作成すれば良いことが分かった。コマンドを以下に記述する。

```
# mkdir demoCA
# cd demoCA
# mkdir certs private crl newcerts
```

privateというディレクトリには，秘密鍵が保存されるので，権限を変更する。

```
# chmod 700 private
```

その後，シリアルを初期化する。

```
# echo "01" > serial
```

最後に,証明書データベースを初期化する。

```
# touch index.txt
```

serial，index.txtはdemoCA配下におく。

次に認証局の証明書，秘密鍵を作成する。認証局が最上位の認証局であるか，他の認証局の下位認証局になるかで操作が異なってくるが，最上位認証局になる場合，自己署名済みの証明書と秘密鍵を生成する。そのコマンドは以下の通りである。

```
# openssl req -new -x509 -newkey ¥
 rsa:2048 -out cacert.pem -keyout ¥
 private/cakey.pem
```

この結果, demoCAフォルダに, cacert.pemという認証局の自己署名済み証明書が作成され，demoCAの下にあるprivateにおいて，認証局の秘密鍵である，cakey.pemが作成される。

以上で認証局が作成された。

4.4　サーバの証明書・秘密鍵作成

demoCAと同じ場所にtestという名前のディレクトリを作成し，サーバがクライアントに渡す証明書およびサーバが所持しておく秘密鍵を作成する。

まずはサーバの秘密鍵，証明書署名要求（CSR）を作成する。

```
# openssl req -new -keyout key.pem ¥
  -out csr.pem
```

この後，クライアントからメールを受信するテストをしたが，失敗した。それはこのコマンド実行時，-nodesオプションをつけてなかったため，サーバの秘密鍵のパスフレーズを入力しなければならなかったからである。

そこで，以下のコマンドを使用して，パスフレーズ無しの秘密鍵にした。

```
# openssl rsa -in key.pem -out key.pem
```

このコマンドでは，出力の際に秘密鍵を暗号化をしないので，パスフレーズの入力をする必要がない秘密鍵となる。このように，パスフレーズありの秘密鍵を上書きする形で，パスフレーズ無しの秘密鍵を準備した。

次にCSRに署名して証明書を作成する。この処理は認証局の管理者の立場で実行する。認証局の管理者が認証局の秘密鍵にアクセス権限がある状態にしておく。そして, demoCAの一つ上のディレクトリで以下のコマンドを実行することで，CSRに署名する。

```
# openssl ca -out ./test/cert.pem ¥
 -infiles ./test/csr.pem
```

この際，途中で認証局の秘密鍵のパスフレーズを入力する必要がある。

これで，認証局の秘密鍵を使って署名した証明書を用意することができた。

最後に，これらのサーバ証明書およびサーバの秘密鍵をPOP3通信で使えるように，その証明書，秘密鍵が置かれているディレクトリを/etc/qpopper.confに記述すればよい。

5.　おわりに

サーバ作成作業において，途中，不明な点が多数あったが，納得できるまで議論しながら，難しいところを理解することができた。今後も積極的に知識を深めながら作業をしていきたい。

参考文献：

1）CA 構築のための OpenSSL の設定
http://moca.wide.ad.jp/notes/ca_doc/openssl.html