OpenVPNによる安全なリモートアクセス環境の構築

OpenVPN による安全なリモートアクセス環境の構築

宮崎大学　工学部教育研究支援技術センター
○森　圭史朗

1. はじめに

近年、不正アクセスによるアカウント ID や個人情報の流出など社会的信用を失うようなセキュリティインシデントが増加傾向にある。このようなセキュリティインシデントを未然に防止するため、昨年から 5 年計画で学内ネットワークに接続する全ての機器を監査するセキュリティ監査が行われることとなった。このセキュリティ監査に伴い、支援先にあるサーバやワークステーションのセキュリティ設定見直しを順次行っている。支援先のサーバやワークステーションには、リモートアクセスを可能とするために OpenSSH が導入されているが、OpenSSH は、不正アクセスを受け易く、設定ミスや脆弱性が原因で外部から不正にサーバを利用されてしまうことが多い。そこで、外部ネットワークからのリモートアクセスをより安全に使用するため、OpenVPN を利用した VPN ネットワークを導入することにした。

本報では、支援先にこれまでの利便性を保ちつつ、より安全なリモートアクセス環境を提供するために導入した OpenVPN による VPN ネットワーク環境を構築したことについて報告する。

キーワード：FreeBSD, OpenVPN, BRIDGE, TAP, IPNAT, IPFW

2. VPN の仕組み

2.1 VPN について

VPN（Virtual Private Network）とは、外部から閉ざされた仮想的なネットワーク回線のことである。VPN は、サーバとクライアント（WindowsOS 等）間に OpenSSL の暗号化アルゴリズムやハッシュアルゴリズムを用いて暗号化された回線により接続し、第 3 者に通信内容を盗まれることのないようにする仕組みである。

VPN には、ルーティング機能を持ったネットワークセグメント間を接続する拠点間接続（図 1）と複数のクライアント端末から VPN サーバに接続するリモートアクセス（図 2）を行う方法がある。

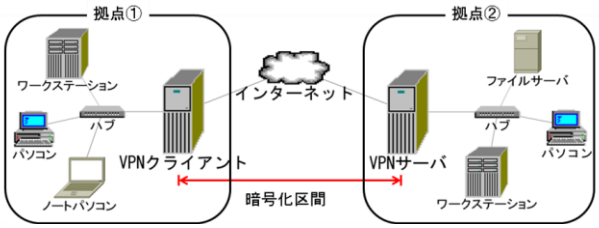

図 1　拠点間接続

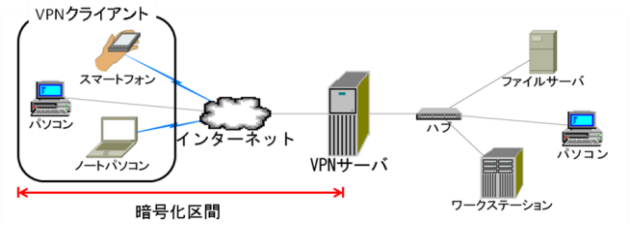

図 2　リモートアクセス

2.2 OpenSSL について

OpenSSL とは、様々な暗号化アルゴリズム（3DES、AES 等）やハッシュアルゴリズム（MD5、SHA-1 等）を提供するオープンソースソフトウェアである。リモートアクセスを行う際に使用する OpenSSH や、個人情報入力やパスワード認証を必要とするサイトで利用される HTTPS（Hypertext Transfer Protocol Secure）等の多くは、この OpenSSL ライブラリを用いて通信内容が暗号化されている。

3. 構築した VPN ネットワーク環境

構築した VPN ネットワークの概要を図 3 に示す。

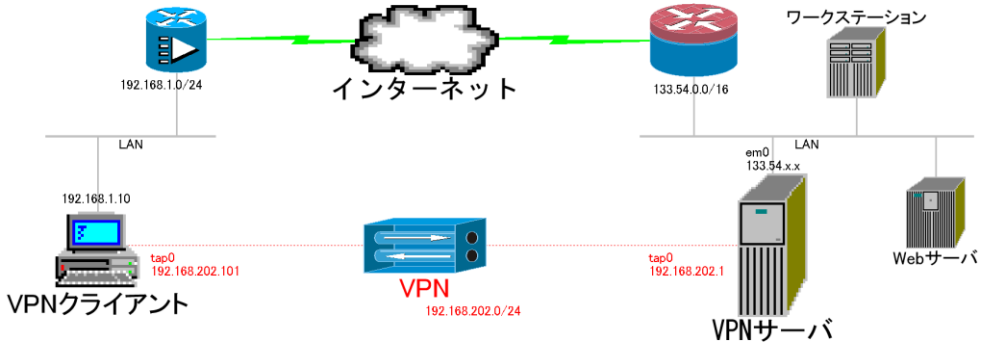

図 3　構築した VPN ネットワーク

VPN サーバは学内ネットワークに設置し、VPN クライアントは外部にある任意のネットワーク上にある。まず、VPN クライアントは、インターネット上を通じて学内にある VPN サーバの 1194udp ポートへアクセスを行う。VPN サーバとクライアント間では、暗号化された OpenSSL の RSA 証明書を通じて互いの 1194udp ポートを使用した認証が開始される。RSA 証明書による認証において通信の許可が得られると VPN サーバはクライアントに対し、仮想ネットワークデバイス TAP の IP アドレスや DNS サーバ等のネットワーク接続情報を送信する。VPN クライアントがこの接続情報を受信すると、VPN クライアントの TAP デバイスネットワークは有効となり、VPN サーバ内のプライベートネットワーク上に存在できるようになる。TAP デバイスが有効となった VPN クライアントは、VPN サーバからアクセスしているような状態となり、サーバの BRIDGE+NAPT を介して学内制限のかかったサーバやワークステーションに対してアクセスが可能となる。

4. VPN ソフトウェアの選定

VPN ソフトウェアは、図 1 の LAN 間接続タイプと図 2 のリモートアクセスタイプの 2 つに分けられる。この 2 つのタイプに対応する VPN ソフトウェアは数多くが存在するが、その中でもクライアント側への導入が容易であり、導入コストのかからないソフトウェアを 3 つ選定し表 1 に比較検討した。この 3 つのソフトウェアについて比較検討を行った結果、通信内容の暗号化による安全性とソフトウェア導入が容易である点から、OpenVPN を導入することにした。

表 1　VPN ソフトウェアの比較

	OpenVPN	PPTP	L2TP（L2TP/IPSec）
導入の際に必要なソフトウェア	・OpenVPN ・OpenSSL ・LZO	・Poptop ・PPP（標準実装）	・IPsec-Tools ・PPP（標準実装） ・OpenSSL
対応 OS （サーバ側）	・Linux　・FreeBSD ・MacOS　・Windows	・Linux　・FreeBSD ・MacOS　・Windows	・Linux　・FreeBSD ・MacOS　・Windows
対応 OS （クライアント側）	・ソフトウェアのインストールが別途必要	・標準実装及びオプションパッケージ	・標準実装及びオプションパッケージ
使用する IP プロトコル及びポート番号	1194/udp （ポート番号の変更可）	GRE 1723/tcp	ESP　500/udp 1701/udp　4500/udp
暗号化アルゴリズムが AES に対応	○	× MPPE-128	△ （Windows7 の場合、3DES）
ハッシュアルゴリズムが SHA-1 に対応	○	－ （設定項目なし）	○

5. OpenVPN を用いた VPN 環境の構築

5.1 VPN サーバの構築

VPN サーバの OS には、FreeBSD-8.3 を用いた。FreeBSD-8.3 を導入後、カーネルの再構築を行う。カーネル再構築のための設定ファイルは、32bitOS と 64bitOS では、異なる場所にある。32bitOS の場合は、/usr/src/sys/i386/conf、64bitOS の場合は、/usr/src/sys/amd64/conf ディレクトリ（フォルダ）内にある。BRIDGE+TAP+IPNAT+IPFW を構築する際は、該当する設定ファイルに以下を追加し、コンパイルとインストールを行う。

```
options    IPFILTER                ←  IPNAT 有効
options    IPFIREWALL              ←  IPFW 有効
device     tap                     ←  仮想ネットワークインターフェイス TAP デバイス有効
device     if_bridge               ←  ブリッジ有効
```

カーネルの再構築後、暗号化アルゴリズム OpenSSL、通信データの転送効率を高めるための圧縮アルゴリズム LZO、VPN ソフトウェアである OpenVPN の順にそれぞれ 配布元サイトからソースコードをダウンロードし、コンパイルとインストールを行う。OpenVPN の場合は、先にインストールした LZO と OpenSSL を利用するため、コンパイル時には LDFLAGS と CPPFLAGS を追加する。

```
#  tar  xvf  openvpn-2.3.4.tar.gz
#  cd  openvpn-2.3.4/
#  ./configure  LDFLAGS="-L/usr/local/ssl/lib  -L/usr/local/lib"
                          CPPFLAGS="-I/usr/local/ssl/include  -I/usr/local/include"
#  make  install
```

※LZO は/usr/local、OpenSSL は/usr/local/ssl 以下にインストールされている

サーバ及びクライアント用設定ファイルのサンプルは、openvpn-2.3.4/sample ディレクトリ内にある。/usr/local/etc 内に openvpn ディレクトリを作成し、その中に server.conf ファイルを作成する。但し、iOS の OpenVPN ソフトウェアでは、TAP デバイスが利用できない。従って、iOS の場合は、サーバ側設定ファイル server.conf を TUN デバイスで使用するように変更する必要がある。以下に作成した server.conf ファイルの内容を示す。

```
port          1194                    ←  サーバ側ポート番号に 1194 を使用
proto         udp                     ←  UDP プロトコル使用
dev           tap0          ←  VPN 用仮想ネットワークデバイスに tap0 を指定
ifconfig      192.168.202.1  255.255.255.0  ←  tap0 デバイスに IP 付与          TAP 使用時
server-bridge 192.168.202.1  255.255.255.0  192.168.202.101  192.168.202.130
        ←  クライアントの IP を 192.168.202.101～192.168.202.130 の範囲で割当て
dev           tun0            ←  VPN 用仮想ネットワークデバイスを tun0 を指定
server        192.168.202.144  255.255.255.240      ←  クライアントに IP 割り当て   TUN 使用時
        (割り当てられる IP は、192.168.202.150、192.168.202.154、192.168.202.158)
inactive      3600                    ←  1 時間ごとに再認証
push          "inactive  3600"        ←  クライアントにも設定
ca            /usr/local/etc/openvpn/easy-rsa/2.0/keys/ca.crt
cert          /usr/local/etc/openvpn/ easy-rsa/2.0/keys/server.crt      ←  RSA 証明書
key           /usr/local/etc/openvpn/ easy-rsa/2.0/keys/server.key
dh            /usr/local/etc/openvpn/ easy-rsa/2.0/keys/dh2048.pem
push          "dhcp-option  133.54.x.x"←  クライアントの DNS サーバ設定
push          "redirect-gateway  def1"  ←  クライアントの通信は、全てサーバ経由
cipher        AES-256-CBC             ←  暗号化アルゴリズム指定
```

```
tls-auth        /usr/local/etc/openvpn/easy-rsa/2.0/keys/ta.key  0
                                        ←  TLS 有効（サーバは"0"、クライアントは"1"）
comp-lzo                                ←  圧縮アルゴリズム使用
keepalive       10  120  ←  「ping 10  ping-restart 120  push "ping 10"  push "ping-restart 120"」と同様
chroot          "/usr/local/etc/openvpn"  ←  プロセスのディレクトリアクセス制限
user            nobody
group           nobody                  ←  管理者権限以外を使用してサーバ OS の安全性強化
persist-key
persist-tun
status          /var/log/openvpn-status.log
log-append      /var/log/openvpn.log    ←  ログ出力関連の設定
verb            3
mute            20
```

通信の暗号化及び認証のための RSA 証明書を作成する。通常は、OpenSSL のコマンドによって作成するが、旧バージョンの openvpn-2.2.2/easy-rsa ディレクトリ内にある証明書作成スクリプトを利用すると簡単に作成できる。easy-rsa には、Windows 用と Unix 用の作成スクリプトがあるが、ここでは、Unix 用の 2.0 を利用した場合を示す。この RSA 証明書作成スクリプトを実行する前に設定ファイル vars の修正を行う必要がある。

```
export OPENSSL="openssl"                →  export OPENSSL="/usr/local/ssl/bin/openssl"
export KEY_SIZE=1024                    →  export KEY_SIZE=2048
export KEY_COUNTRY="US"                 →  export KEY_COUNTRY="JP"
export KEY_PROVINCE="CA"                →  export KEY_PROVINCE="MIYAZAKI"
export KEY_CITY="SanFrancisco"          →  export KEY_CITY="miyazaki"
export KEY_ORG="Fort-Funston"           →  export KEY_ORG="University of Miyazaki"
export KEY_EMAIL=me@myhost.mydomain     →  export KEY_EMAIL=""
```

RSA 証明書作成スクリプトは、bash シェルで作成されているため、先に bash のインストールする。bash シェルをインストールした場合は、各証明書作成スクリプトにある実行シェルのパスを以下のように変更する。

```
#!/bin/bash         →  #!/usr/local/bin/bash
```

以下のコマンドを順番に実行し、OpenVPN 用の RSA 証明書作成を行う。(RSA 証明書作成時の詳細は省略)

```
#  /usr/local/bin/bash                          ←  bash シェルを使用
#  .  ./vars                                    ←  RSA 証明書作成用シェル環境の読み込み
#  ./clean-all                                  ←  ./keys ディレクトリ以下のファイル削除（初期化）
#  ./build-ca                                   ←  SSL 認証局作成
#  ./build-key-server  server                   ←  認証局秘密鍵作成
#  ./build-dh                                   ←  暗号鍵作成
#  ./build-key-pass  client                     ←  パスワード認証付きクライアント証明書及び秘密鍵作成
#  /usr/local/sbin/openvpn  --genkey  --secret  ./keys/ta.key      ←  TLS 鍵作成
```

サーバ起動時に BRIDGE+TAP+IPNAT+IPFW を有効にするため、rc.conf、sysctl.conf、ipnat.rules、rc.firewall ファイルに以下を追加する。LAN のネットワークインターフェイスは、em0 を使用した例である。

/etc/rc.conf（ブリッジの作成）

```
cloned_interfaces="bridge0 tap0"
ifconfig_bridge0="up"
ifconfig_tap0="up"
autobridge_interfaces="bridge0"
autobridge_bridge0="em0 tap0"
```

ipnat_enable="YES"
ipnat_flags="-CF"

/etc/sysctl.conf（IPFW のブリッジ許可）

net.link.bridge.ipfw=1

/etc/ipnat.rules（TAP デバイスの IP アドレスを VPN サーバの IP アドレスへ変換）

map em0 192.168.202.0/24 -> 0/32 portmap tcp/udp auto mssclamp 1460
map em0 192.168.202.0/24 -> 0/32 mssclamp 1460

/etc/rc.firewall（OpenVPN と TAP デバイス間の通信許可）

ipfw add allow udp from any to me 1194 keep-state via em0
ipfw add allow udp from me 1194 to any out keep-state via em0
ipfw add allow log ip from 192.168.202.0/24 to any via tap0

最後に FreeBSD 用のサーバ起動スクリプトがないため、以下にあるサービスを起動させるスクリプトファイルを/usr/local/etc/rc.d/に作成する。

/usr/local/sbin/openvpn --writepid /var/run/openvpn.pid --config /usr/local/etc/openvpn/openvpn.conf

5.2 VPN クライアントの構築

サーバ側でクライアント用の RSA 証明書を作成する。複数のクライアント証明書を作成する場合は、先ほどの RSA 証明書作成スクリプト easy-rsa/2.0 において build-key-pass を実行し、クライアントごとに作成する。

```
#  /usr/local/bin/bash
#  .  ./vars
#  ./build-key-pass  client                    ※client の部分は、クライアントごとに異なる名称にする。
```

OpenVPN 用のクライアントソフトウェアは、配布元である http://swupdate.openvpn.org/community/releases よりクライアント OS と一致するものをダウンロードし、インストールする。インストール後、クライアント設定ファイルは、RSA 証明書（ca、cert、key、tls-auth）を含めたファイルを作成する。設定ファイルを作成後、インストール先の conf フォルダ内にコピーし、拡張子は「.ovpn」とする。iOS の場合は、メールに添付して送信してメールソフトから添付ファイルを開くか、iTunes の同期を利用して VPN クライアントソフトにインポートする。

```
client                          ←  クライアントとして起動
port      1194
proto     udp
dev       tap0（TUN の場合は、tun0）
remote    133.54.x.x   1194     ←  接続先 VPN サーバの IP アドレスとポート番号
tun-mtu   1392                  ←  iOS 使用時の MTU 設定
mssfix    1352                  ←  iOS 使用時の MSS 設定
cipher    AES-256-CBC
nobind                          ←  ローカルポート（udp 1194）を開かない。
comp-lzo
key-direction       1           ←  <ca>行以降にあるキーダイレクションを使用
<ca>
-----BEGIN CERTIFICATE-----
MIIEUTCCAzmgAwIBAgIJAM6TtzarGhkRMA0GCSqGSIb3DQEBBQUAMHg・・・・（以下省略）
-----END CERTIFICATE-----
</ca>
<cert>
```

-----BEGIN CERTIFICATE-----
MIIEuDCCA6CgAwIBAgIBBjANBgkqhkiG9w0BAQUFADB4MQswCQYDV・・・・・(以下省略)
-----END CERTIFICATE-----
</cert>
<key>
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFDjBABgkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIzRkfJ56RU・・・・・(以下省略)
-----END ENCRYPTED PRIVATE KEY-----
</key>
<tls-auth>
-----BEGIN OpenVPN Static key V1-----
e0639a43e0543e11e6304bbe78763323a855636140fc26ce96039daf4b11d59b8422・・・・・(以下省略)
-----END OpenVPN Static key V1-----
</tls-auth>

6. 動作確認及びスループット速度の計測

表 2 にある VPN サーバと VPN クライアントを用いて、動作確認とスループット速度の計測を行った。

表 2　動作確認及びスループット測定に用いたハードウェア

	VPN サーバ	VPN クライアント①	VPN クライアント②
ハードウェア構成	CPU Pentium4 3.06GHz	CPU Core-i5 3GHz	iPhone5
	Memory DDR-266 1GB	Memory DDR3-1333 4GB	
	HDD 7,200rpm UDMA100	HDD 7,200rpm SATA3.0	
	LAN Intel PRO/1000	LAN Realtek Giga	
OS	FreeBSD-8.3	Windows7	iOS7.0.6

6.1 動作確認

VPN クライアントソフトウェアである OpenVPN GUI を起動し、タスクバーに OpenVPN のアイコンを表示させる（図 4）。このタスクバーのアイコンを実行して RSA 証明書の認証に成功すると、タスクバーアイコンは VPN 接続済み（図 5）となり、サーバからクライアントへネットワーク接続情報が送信される（図 6）。ネットワーク接続情報を受信後、VPN 通信が開始され、VPN クライアントから学内ホストへアクセスする全ての通信は、VPN サーバからアクセスしているような状態となる（図 7）。図 8 は、iOS7 の OpenVPN クライアントソフトを用いて VPN 接続したものである。

図 4　VPN 未接続

図 5　VPN 接続済み

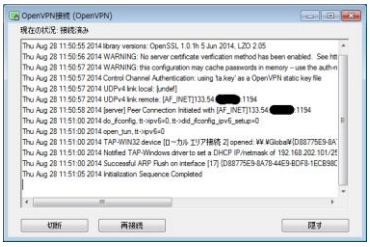

図 6　VPN 認証ログ

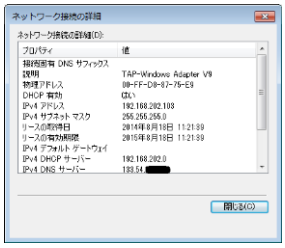

図 7　VPN 接続後の TAP の情報

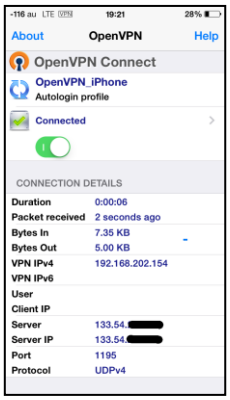

図 8　iOS7 による VPN 接続

図 9 と図 10 は、VPN クライアントが VPN サーバに接続後、学内ホストに対し telnet でリモートアクセスした際の通信内容を tcpdump によりテキスト化したものである。VPN サーバと VPN クライアントの間では、通信内容が暗号化されていることがわかる。

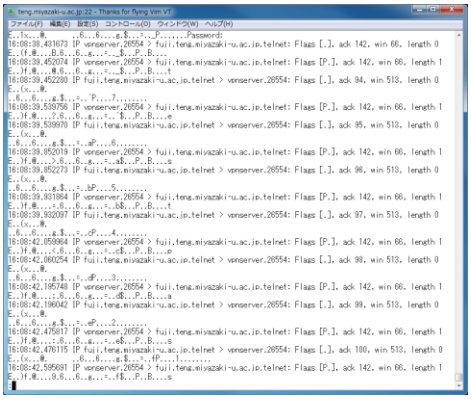

図 9　学内ホストと VPN サーバ間の通信内容

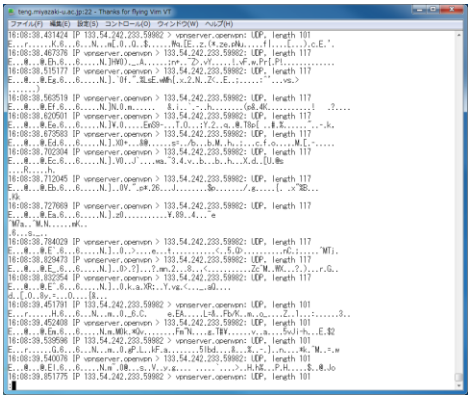

図 10　VPN サーバと VPN クライアント間の通信内容

6.2 スループット速度の計測

表 3 は、OpenVPN によるスループット速度の影響を調査した結果である。この結果は、表 2 のサーバ及びクライアントを用いて、学内ホストにある 200MB のファイルを各 5 回ずつ速度計測し平均したものである。

表 3　スループット速度の計測

	OpenVPN 無し		OpenVPN	
	FTP	SFTP	FTP	SFTP
Windows7	772.17Mbps	122.33Mbps	63.04Mbps	56.35Mbps
VPN サーバ	151.73Mbps	102.08Mbps	―	―
iOS7　LTE 回線	―	13.07Mbps	―	7.92Mbps

OpenVPN による通信は、OpenVPN 無しに比べてスループット速度が低下している。これは、VPN サーバのハードウェア性能不足と VPN サーバ及びクライアントにおける通信内容の暗号化及び復号化処理に負荷がかかってしまうことが原因であると考えられる。

7. まとめ

OpenVPN を導入したことにより、OpenSSH にはない TLS 鍵の利用が可能となる他、安全性の高い暗号化アルゴリズム及びハッシュアルゴリズムの選択にも幅が広がり、これまでよりも安全なリモートアクセス環境を支援先に提供することができた。また、サーバを管理する側にとっても、外部ネットワークからの ssh ポートに対するアクセスをファイアーウォールで閉じてしまうことが可能となるため、頻繁に行われる不正アクセスによる大量のサーバログがなくなり、ログ管理の負担も大幅に軽減された。

表 3 のスループット速度計測結果より、OpenVPN 接続時のスループット速度は、サーバのハードウェア性能と通信内容の暗号化処理によって低下しているが、1Mbps 以上のスループット速度を確保できているため、リモートアクセス環境において速度低下の影響はないものと考えられる。