

SSL VPN und die Wahl einer geeigneten VPN-Lösung

7. Juni 2004

Rechtlicher Hinweis

Dieser Beitrag ist lizenziert unter der UVM Lizenz für die freie Nutzung unveränderter Inhalte.

Zusammenfassung

In dem Vortrag/Paper soll es um die Thematik der SSL VPNs gehen, das heißt VPN basierend auf dem SSL/TLS-Protokoll. Eine Reihe von Herstellern bieten VPN-Appliance-Lösungen an, bei denen kein expliziter VPN-Client installiert wird, sondern ein herkömmlicher Browser als Client genutzt wird. Zur Absicherung wird der integrierte SSL/TLS-Stack verwendet. Man verspricht sich von dieser Lösung ein "Clientless VPN", das es für den Benutzer deutlich einfacher macht, den Zugang zum privaten Netzwerk zu erhalten. Darüberhinaus soll jeder beliebige Internet-fähige Rechner als VPN-Client genutzt werden können. Der Vortrag erklärt die Idee und Technologie, die zur Realisierung eines solchen "Clientless VPN" verwendet wird. Es wird eine Einordnung in die konventionellen VPN-Technologien (IPSec/L2TP, PPTP, SSH) gegeben und aufgezeigt, welche Variante für welche Umgebungen am besten geeignet ist. Mit OpenVPN soll der aktuelle Stand von SSL/TLS-basierten VPN-Lösungen in der Open Source-Welt demonstriert werden.

SSL VPN und die Wahl einer geeigneten VPN-Lösung

Lizenz für die freie Nutzung unveränderter Inhalte (UVM), Version 1.0

In diesem Dokument geht es um die Thematik der SSL VPNs, das heißt VPN basierend auf dem SSL/TLS-Protokoll. Eine Reihe von Herstellern bieten VPN-Appliance-Lösungen an, bei denen kein expliziter VPN-Client installiert wird, sondern ein herkömmlicher Browser als Zugangssoftware genutzt wird. Zur Absicherung wird der integrierte SSL/TLS-Stack verwendet. Man verspricht sich von dieser Lösung ein „Clientless VPN“, bei dem jeder beliebige Internet-fähige Rechner als VPN-Terminal verwendet werden kann sowie der Konfigurationsaufwand einer Client-Software entfällt. In diesem Dokument werden die Grundideen von SSL VPN beschrieben sowie ein Vergleich mit konventionellen VPN-Technologien wie IPSec und PPTP gegeben. Anhand der Software OpenVPN wird der aktuelle Stand von SSL/TLS-basierten VPN-Lösungen in der Open Source-Welt demonstriert.

1 Einleitung

Als Virtual Private Network (VPN) bezeichnet man die sichere Verbindung von Rechnern und Netzwerken über unsichere, öffentliche Leitungen. Die Abhörsicherheit und Integrität der Daten wird hierbei mit Hilfe von kryptographischen Verfahren realisiert. Grundsätzlich unterscheidet man zwei verschiedene Topologien: beim Site-to-Site-VPN werden zwei Netzwerke transparent miteinander verbunden. Das „Tunneln“ der Daten übernehmen jeweils zwei VPN-Gateways, die sämtlichen Datenverkehr zwischen den beiden Lokationen verschlüsseln. Bei Remote-Access-VPN geht es um den Zugriff auf ein internes Netz durch mobile Benutzer, auch Roadwarrior genannt.

Bisher kommen als VPN-Technologien vor allem IPSec und PPTP (für Remote Access) zum Einsatz. SSL (Secure Sockets Layer) und dessen Nachfolger TLS (Transport Layer Security) wurden als Protokolle zur sicheren Kommunikation zwischen Anwendungen entworfen. In der Form von SSL-VPN-Gateways wird dieses Protokoll neuerdings auch als VPN-Lösung eingesetzt.

2 Was ist ein SSL VPN?

Der herkömmliche Ansatz zur Realisierung eines Remote-Access-VPNs ist die Verwendung einer speziellen Client-Software, mit deren Hilfe die VPN-Verbindung geöffnet wird. Für IPSec-basierte Systeme ist dies z.B.

SafeNet SoftRemote oder Sentinel. Grundidee eines SSL VPN ist die Verwendung eines herkömmlichen Browsers als VPN-Client. Da heute jeder gängige Browser eine vollständige Verschlüsselungskomponente in Form von SSL/TLS integriert hat, ist der Gedanke bestechend, dies für einen VPN-Zugang auszunutzen. SSL bietet sicherheitstechnisch alle Eigenschaften, die man für ein VPN-Protokoll benötigt und enthält neben allen gängigen Verschlüsselungsverfahren eine vollständige Zertifikats-Unterstützung durch X.509v3. SSL ist heute der De-Facto-Standard für Online-Shops und E-Commerce.

Wie kann ein SSL VPN nun konkret aussehen? Der Benutzer ruft im Browser einen speziellen URL auf, z.B. <https://vpn.example.org> und gibt auf der vom SSL-VPN-Gateway generierten Anmeldeseite seinen Benutzernamen und sein Kennwort ein. Idealerweise verfügt das VPN-Gateway über ein offiziell signiertes Zertifikat, damit der Client sicherstellen kann, mit dem richtigen Server verbunden zu sein und nicht Opfer einer Man-in-the-Middle-Attacke zu werden. Nach erfolgreicher Anmeldung bekommt der Benutzer ein Web-Portal als Einstiegsseite präsentiert. Von dort kann er über vordefinierte Hyperlinks auf verschiedene Dienste und Server zugreifen, wie zum Beispiel Dateiserver, E-Mail und Intranet-Dienste. Jeder Benutzer bekommt entsprechend seiner Zugriffsrechte ein persönlich angepasstes Portal präsentiert. Die Kommunikation mit dem VPN-Gateway und den dahinter liegenden Servern ist über die gleiche SSL-Verbindung abgesichert. Ermöglicht wird dies durch ein Proxy-Verhalten des VPN-Gateways, das die Anfragen des Clients an die entsprechenden Server weiterleitet.

Nun gibt es je nach Leistungsumfang des VPN-Gateways verschiedene Komfortstufen für einen solchen VPN-Zugang. In seiner einfachsten Form werden dem Benutzer sämtliche Dienste als Web-Interface dargestellt. Allgemein vertraut ist dies beim Zugriff auf einem WebMail-Dienst. Doch auch andere Anwendungen wie Dateidienste oder Datenbankzugriffe lassen sich als Web-Oberfläche darstellen. An dieser Stelle tritt die eigentliche Leistung des SSL-VPN-Gateways in Erscheinung. Für solche Dienste, bei denen ein Server selbst kein Web-Interface bietet, generiert das Gateway ein solches Interface für den Benutzer. Es kommuniziert mit den Servern daher über die „normalen“ Protokolle wie FTP, NFS, SMTP, etc. Der Client sieht dagegen die Dienste als Web-Sicht. Das VPN-Gateway ist somit Vermittler und Übersetzer zwischen Client und Server (Abb. 1).

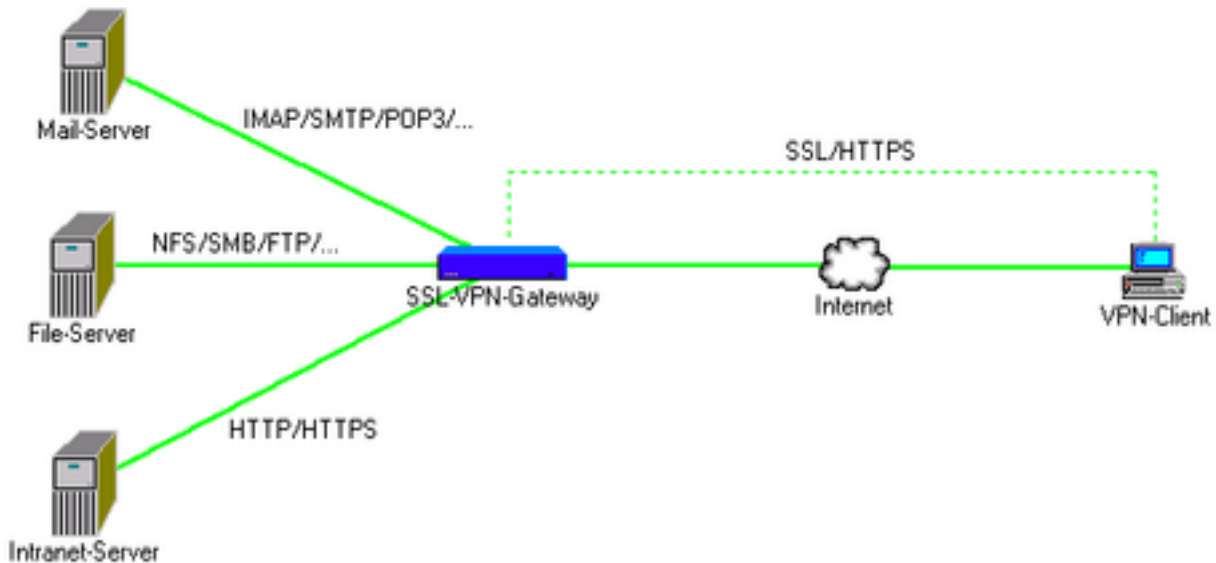


Abb. 1: Ein SSL-VPN-Gateway übersetzt Protokolle auf Web-Ebene. Es ist offensichtlich, daß je nach Protokoll und Dienst hierfür ein großer Aufwand auf dem Gateway nötig ist. Es muss die Web-Oberfläche erstellen und gleichzeitig den Zustand der Anwendungen intern verwalten. Der Leistungsumfang der Hersteller unterscheidet sich daher auch beträchtlich in der Anzahl und der Art der unterstützten Anwendungen. Ob dem Benutzer diese Art der Darstellung von Diensten gefällt, ist eine andere Frage. Den Komfort einer „richtigen“ graphischen Anwendung wird man über ein Web-Interface nicht erreichen können, trotz aller Bemühungen der Hersteller, dies so komfortabel wie möglich zu realisieren. Die Performance leidet unter der Browser-Integration und das parallele Arbeiten mit mehreren Anwendungen wird erschwert. Das Beispiel Dateiservice zeigt deutlich, dass man den Komfort von lokal installierten Anwendungen mit Hilfe einer Web-Oberfläche nicht erreicht. Ein Dateimanager bietet hier deutlich komfortableres und zügigeres Arbeiten als die Verwendung von Hyperlinks und Upload-Buttons zum Austausch von Dateien.

3 Port Forwarding und Tunneling

Aus diesem Grund bieten SSL-VPN-Gateways die Möglichkeit, „echte“ VPN-Verbindungen zu realisieren, um lokal installierte Anwendungen zu nutzen. Ermöglicht wird dies mit Hilfe von Browser Applets und Port Forwarding. Das Applet installiert sich bei Bedarf automatisch auf dem Client-Rechner und bindet sich an einen oder mehrere lokale Ports. Wird eine Verbindung zu einem solchen lokalen Port aufgebaut, so leitet das Applet die Daten über die SSL-Verbindung an das Gateway weiter, das die Pakete wiederum an einen bestimmten Server weiterreicht. Gängig ist diese Art des Tunnelns einer TCP-Verbindung beim Secure-Shell-Protokoll. Die Datenpakete nehmen also quasi eine Umleitung über den verschlüsselten Tunnel. Woher weiß die lokale Anwendung aber nun, mit welchem lokalen Port sie sich verbinden soll, unter dem ein bestimmter Server zu erreichen ist? SSL-VPN-Gateways lösen dieses Problem mit Hilfe des Name Services, indem sie die lokale Hosts-Datei umschreiben. Für jeden entfernten Server (entsprechend dem Benutzerprofil des VPN-Teilnehmers) wird in der Hosts-Datei ein Eintrag für eine Loopback-Adresse erzeugt. Das Applet bindet sich an die gleiche Portnummer wie der gewünschte Dienst auf dem Server sowie an die jeweilige Loopback-Adresse aus dem Bereich 127.0.0.0/8, die den Server repräsentiert. Zum Beispiel könnte ein Mailserver mail.example.org der Adresse 127.1.2.3 zugeordnet werden. Das Browser-Applet bindet sich an die Adresse 127.1.2.3 und die Ports von SMTP und POP3. Das lokale Mailprogramm auf dem Client verbindet sich dann auf das Applet, da es per Namensauflösung die Adresse 127.1.2.3 für den Host mail.example.org erhält. Nach Beenden der VPN-Verbindung wird die Hosts-Datei wieder in ihren ursprünglichen Zustand versetzt. Aus Anwendungssicht geschieht das Tunneln des Protokolls über SSL daher transparent.

Problematisch bei Port Forwarding sind Dienste, die mehrere Ports benutzen und Rückverbindungen öffnen, wie z.B. FTP. Um auch ein solches Protokoll mit lokalen Anwendungen nutzen zu können, ist ein vollwertiger Netzwerk-Zugang in der Art von IPsec oder PPTP notwendig. Einige Hersteller von SSL-VPN-Appliances realisieren daher einen vollständigen Netzwerk-Zugriff mit Hilfe einer virtuellen Netzwerkkarte. Hierbei wird dem Netzwerkadapter bei Einwahl eine spezielle, virtuelle IP-Adresse zugewiesen. Sämtlicher Netzwerkverkehr über diesen Adapter wird dann automatisch über die SSL-Verbindung getunnelt. Man sieht gleich, dass eine solche Lösung sehr tief in das System eingreifen muß, was mit normalen Java-Applets nicht mehr zu realisieren ist. Daher kommen an dieser Stelle ActiveX-Controls zum Einsatz. Leider verliert man hierbei die Plattform-Unabhängigkeit und Flexibilität, denn damit ist man auf einen Windows-Client-Rechner und auf den Internet Explorer als Browser angewiesen. Andere Betriebssysteme und Browser können nicht genutzt werden.

Streng genommen kann man bei diesem Szenario nicht mehr von „Clientless“ sprechen. Denn ein solches ActiveX-Control ist ein eigenständiges Programm und benötigt weit reichende Befugnisse. Es hinterlässt deutliche Spuren im System. „Automatische Installation“ wäre daher eine passendere Umschreibung. Zusätzlich bringt die Verwendung von ActiveX neue Sicherheitsprobleme.

4 Risiken und Nebenwirkungen

SSL VPN verspricht viele der Probleme von herkömmlichen VPN-Technologien zu lösen. Sie bringt jedoch neue Gefahren mit sich, die die anderen Varianten nicht kennen. Hauptproblem hierbei ist die Philosophie, jeden beliebigen Internet-fähigen Rechner als VPN-Client zu benutzen. Da über ein VPN sensible Daten ausgetauscht werden, stellt es grundsätzlich eine Gefahr dar, diese Informationen mit einem fremden Rechner zu bearbeiten. Bei einem Web-basierten SSL VPN bergen bereits gängige Browser-Features die Gefahr des nachträglichen Ausspähs. AutoComplete und Caching sorgen dafür, dass Informationen, Zugangsdaten und Passwörter auch nach Abmelden vom System dauerhaft auf dem Rechner gespeichert werden. Geöffnete Dokumente wie Office-Dateien bleiben als temporäre Dateien auf der Festplatte versteckt erhalten und können nachträglich wiederhergestellt werden. Die History-Funktion des Browsers speichert abgerufene URLs, die Hinweise auf die interne Struktur des zu schützenden Netzwerks und Kundenkontakte liefern können. Bei der Nutzung lokaler Anwendungen durch Port Forwarding oder transparentem Netzwerkzugang erstrecken sich die sensiblen Informationen über den Browser hinaus auch auf die verwendeten Anwendungen.

Doch ein unbekannter Rechner birgt unter Umständen weitere Gefahren. Durch Spionageprogramme und Trojaner können Zugangsdaten und Informationen unbemerkt ausgespäht werden. Der Besitzer des Rechners muss sich noch nicht einmal über derartige Hintertüren bewusst sein. Durch eine Sicherheitslücke im Betriebssystem oder einer Anwendung kann schädlicher Code unbemerkt auf dessen Rechner eingerichtet werden

sein. Grundsätzlich sollte also bei der Auswahl eines Client-Rechners Sorgfalt geübt werden. Ein Rechner, den man nicht bedenkenlos auch für Online Banking verwenden würde, ist als VPN-Client daher ungeeignet. Internet-Cafes fallen damit als Zugangsstationen bereits vollständig aus. Allenfalls Rechner in befreundeten Umgebungen wie z.B. von Partnerfirmen sind als Clients geeignet, wobei man auch in diesem Fall Vorsicht walten lassen sollte.

Was die Verfügbarkeit betrifft, so hat man bei „vollwertigem“ VPN neben der Einschränkung auf die Windows-Plattform zusätzlich das Problem, dass ActiveX benötigt wird. Dies ist jedoch in vielen Umgebungen nicht vorhanden, da ActiveX-Controls aus Sicherheitsgründen blockiert werden. Das Versprechen, ein „Anywhere“-VPN mit SSL realisieren zu können, kann derzeit in der Realität also nicht wirklich erreicht werden. Zusammenfassend kann man sagen, dass der Anspruch eines „Anywhere, Secure, Clientless“-VPN für Remote Access auch mit SSL VPN nicht erreicht wird. „Clientless“ ist eher als „Automatische Installation“ zu bezeichnen, „Anywhere“ wird in der Praxis so nicht erreicht und „Secure“ hängt immer von dem jeweiligen Rechner ab, den man verwendet. Die Einrichtung auf Client-Seite und damit die Benutzbarkeit kann allerdings deutlich vereinfacht werden.

5 VPN-Alternativen - IPSec, PPTP und SSH

Wir wollen nun SSL mit den anderen etablierten VPN-Technologien vergleichen, um entscheiden zu können, welche Variante für welchen Einsatzzweck geeignet ist.

Derzeitiger De-Facto-Standard für VPN ist IPSec, das als Kurzform für IP Security steht. Es ist die allge-meinste und flexibelste VPN-Technologie, die sowohl für Site-to-Site als auch für Remote Access gleichermaßen geeignet ist. Großer Vorteil ist seine Unabhängigkeit vom jeweiligen Hersteller. Es lassen sich VPN-Tunnel zwischen unterschiedlichsten Gateways aufbauen und auch in der Wahl des VPN-Clients ist man flexibel. Unter Linux werden bisher auf FreeS/WAN basierende Implementierungen eingesetzt: Openswan (das ehemalige Super FreeS/WAN), Strongswan oder reines FreeS/WAN mit ausgewählten Patches. In den Kernel 2.6 wurde IPSec direkt integriert, so dass das bisherige Kernelmodul KLIPS überflüssig geworden ist. Was den Schlüsselaustauschdienst IKE betrifft, so kann der Administrator inzwischen aus drei verschiedenen Implementierungen wählen: Pluto (aus dem FreeS/WAN-Projekt), Racoon (aus dem KAME-Projekt, das z.B. in NetBSD eingesetzt wird) und Isakmpd (aus dem OpenBSD-Projekt). Alle drei Varianten arbeiten mit dem Kernel 2.6 zusammen und haben ihre individuellen Stärken und Schwächen.

IPSec steht in dem Ruf, zwar das sicherste VPN-Protokoll, aber auch das komplizierteste zu sein. Gängige IPSec-Clients bieten eine Vielzahl von Optionen (virtuelle IP-Adressen, NAT-Traversal, Lifetimes, etc.), die einen durchschnittlichen Anwender leicht überfordern. Daher wird ein solcher Client üblicherweise von der IT-Serviceabteilung eingerichtet. Mit Hilfe von Konfigurationsdateien (Policy Files) läßt sich die Einrichtung deutlich vereinfachen und zentral verwalten. Was die Verfügbarkeit betrifft, so hat man im Gegensatz zu SSL/HTTPS leider nicht aus jedem Netzwerk heraus die Möglichkeit, IPSec zu verwenden, da Firewalls den Traffic eventuell blockieren.

Das Point-to-Point Tunneling Protocol (PPTP) wird üblicherweise als reine Remote-Access-Lösung verwendet. Es stellt eine Erweiterung des PPP (Point-to-Point Protocol) um Verschlüsselung und Authentifizierung dar und verwendet den Microsoft-Verschlüsselungsmechanismus MPPE. Durch die Verwendung von PPP können dessen Features voll genutzt werden, z.B. die automatische Zuweisung von virtuellen IP-Adressen und Name-Server-Adressen. Ein PPTP-Client ist in Windows-Systemen direkt integriert und sehr leicht einzurichten. Für Linux existiert serverseitig das Poptop-Projekt (PPTPD, <http://www.poptop.org>) und auf der anderen Seite ein PPTP-Client (<http://pptpclient.sourceforge.net>). Für PPTP spricht die einfache Konfiguration mit sehr wenigen Optionen. Dagegen spricht sein eher schlechter Ruf, was die Sicherheit des Protokolls betrifft. Hauptproblem hierbei ist, daß sich der Sitzungsschlüssel des Tunnels direkt aus dem MD4-Hashwert des Passworts ableitet. Wird dieses Passwort geknackt, so können alle bisherigen Verbindungen nachträglich entschlüsselt werden. Die Sicherheit des VPN ist also unmittelbar von der Qualität der Passwörter abhängig. Als Alternative zu PPTP mit ähnlichen Möglichkeiten und Komfort bietet sich L2TP over IPSec an, das ebenfalls PPP verwendet, jedoch auf der Sicherheit von IPSec beruht.

Schließlich läßt sich auch das Secure-Shell-Protokoll als VPN verwenden. Durch das integrierte Port Forwarding können lokale Anwendungen durch die verschlüsselte Verbindung auf entfernte Dienste zugreifen. Umgekehrt können auch in umgekehrter Richtung solche Anwendungstunnel aufgebaut werden. Ausgenutzt wird dies üblicherweise für das Tunneln von X11-Traffic zur Remote-Ausführung von graphischen

Anwendungen. Mit Hilfe von ein paar zusätzlichen Tricks ist es möglich, Secure Shell zur Realisierung eines vollständigen VPN zu verwenden, jedoch wird dies in der Praxis nur sehr selten durchgeführt. Secure Shell als VPN ist also eher ein exotisches Verfahren. Die Stärken liegen primär im sicheren Terminal-Zugriff auf entfernte Rechner.

In der folgenden Tabelle sind die individuellen Eigenschaften der verschiedenen Verfahren als VPN-Protokoll noch einmal aufgeführt. Auch wenn man zumindest theoretisch jedes Verfahren sowohl für Site-to-Site als für Remote Access verwenden kann, ist die Auswahl einer geeigneten Lösung sehr stark von den jeweiligen Anforderungen abhängig.

Tabelle 1:

	SSL
Vorteile	Komfortabel für Benutzer Automatische Installation und Konfiguration Einfache Firewall-Konfiguration
Nachteile	Neue Sicherheitsprobleme durch „Anywhere“-Konzept Meist auf Windows-Clients beschränkt
Empfohlenes Szenario	Remote Access in Windows-Umgebungen

6 SSL VPN mit Open Source: OpenVPN

Im Open-Source-Bereich gibt es derzeit keine Software, die ein SSL VPN zur Verfügung stellt, wie man es von kommerziellen Lösungen gewohnt ist. Doch es gibt einige Projekte, die SSL/TLS als Basisprotokoll für VPN verwenden. Dies alleine kann unter Umständen schon eine einfachere Einrichtung des Systems darstellen und eine höhere Verfügbarkeit garantieren. Der HTTPS-Port 443 wird normalerweise von jeder Firewall durchgelassen. Zusätzlich bereitet Network Address Translation (NAT) dem SSL-Protokoll keine Probleme. OpenVPN (<http://openvpn.sourceforge.net>) ist eine SSL/TLS-basierte VPN-Software, die sich sowohl für Site-to-Site- als auch für Remote-Access-Umgebungen eignet. Das Projekt wird sehr gut gepflegt und hat sich zu einer leistungsfähigen und beliebten VPN-Lösung entwickelt. OpenVPN unterstützt eine breite Palette von Betriebssystemen (Linux, Windows 2000/XP, Mac OS X, Solaris, OpenBSD, FreeBSD, NetBSD) und ist darüber hinaus sehr einfach zu konfigurieren.

Technisch realisiert wird OpenVPN mit Hilfe von virtuellen TAP/TUN-Interfaces. TAP-Interfaces erlauben hierbei ein Tunneln von Layer-2-Paketen (Bridging), was die Übertragung von Nicht-IP-Protokollen wie NetBEUI, IPX oder AppleTalk ermöglicht. Bei der Verwendung von TUN-Interfaces werden Layer-3-Pakete übertragen; es wird also normales Routing durchgeführt. Eine TUN-basierte Lösung ist deutlich einfacher einzurichten als die TAP-Variante.

Obwohl OpenVPN auch mit statischen Schlüsseln umgehen kann, empfiehlt sich der Einsatz von X.509-Zertifikaten und RSA-Signaturen für den Schlüsselaustausch. Die hierzu benötigten RSA-Schlüssel und Zertifikate können mit Hilfe der OpenSSL-Bibliothek erzeugt werden. Eine Anleitung für FreeS/WAN findet sich in (2), die Zertifikate können in der gleichen Art für OpenVPN verwendet werden.

6.1 Site-to-Site

Um zwei Netzwerke miteinander per OpenVPN zu koppeln, ist jeweils auf beiden Gateways ein dedizierter Port notwendig. Per Default ist dies die Portnummer 5000. Empfohlen wird die Verwendung von UDP als Basisprotokoll, TCP ist jedoch ebenfalls möglich. Die beiden Gateways verwenden statische, virtuelle IP-Adressen, die lediglich für die VPN-Verbindung verwendet werden. Um Subnetze per VPN miteinander zu koppeln, genügt die Definition von entsprechenden Routing-Einträgen, die automatisch von OpenVPN erzeugt werden können. Eine Anleitung zur Einrichtung eines Site-to-Site-VPNs ist auf der OpenVPN-Homepage zu finden.

6.2 Remote Access

Mit der im Beta-Stadium befindlichen Version 2.0 sind die Remote-Access-Möglichkeiten deutlich verbessert worden. In früheren Versionen war es notwendig, jedem Client einen dedizierten Port zuzuweisen. Der Cli-

ent musste diesen Port kennen und sich darauf verbinden. In Umgebungen mit mehreren Clients ist dies so nicht mehr handhabbar. Mit OpenVPN 2.0 gibt es nun zwei verschiedene Möglichkeiten, komfortables Remote Access zu realisieren. Für UDP können sich Clients auf den gleichen UDP-Port simultan verbinden. Der Daemon-Prozess verwaltet hierbei eine beliebige Anzahl von Clients. Als zweite Option kann TCP mit Hilfe des inetd verwendet werden. In diesem Fall wird für jeden Client ein separater Prozess erzeugt.

Wir wollen nun als Beispiel eine UDP-basierte Remote-Access-Lösung einrichten. Die Topologie ist in Abb. 2 dargestellt. Die Clients möchten das interne Netz 192.168.1.0/24 erreichen. Als virtuelle IP-Adressen kommt der Adressbereich 192.168.0.0/24 zum Einsatz, der es bei OpenVPN erlaubt, bis zu 63 Clients gleichzeitig zu bedienen (OpenVPN verbraucht für jeden Client 4 IP-Adressen, und benötigt selber eine IP-Adresse aus dem gleichen Subnetz).

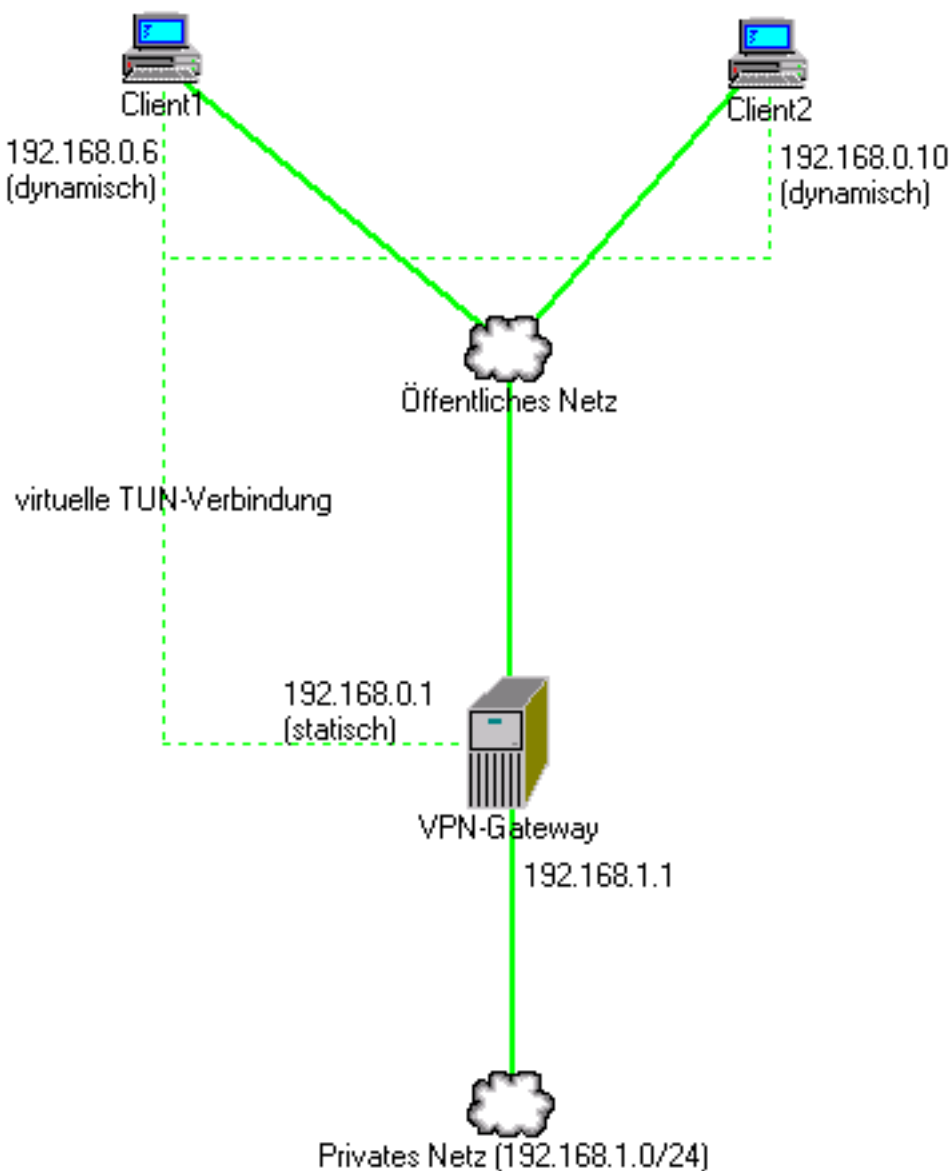


Abb. 2: Ein Remote-Access-Szenario mit OpenVPN

Nachdem die Schlüssel und Zertifikate als my-ca.pem (CA-Zertifikat), officekey.pem, officecert.pem (Schlüssel und Zertifikat des Gateways) sowie userXcert.pem und userXkey.pem (für den User X) generiert wurden, benötigen wir können noch die Diffie-Hellman-Parameter. Die Generierung wird ebenfalls in dem OpenVPN-

Howto beschrieben. Die Konfigurationsdateien für Server und Clients sehen dann wie folgt aus.

6.3 VPN-Gateway („Office“), `tls-office.conf`

```
# Use TUN device as virtual interface
dev tun
# Operate as VPN server
mode server
# Authentication algorithm (see all with openvpn --show-digests)
auth SHA1
# Encryption algorithm (see all with openvpn --show-ciphers)
cipher AES-256-CBC
# Virtual IP addresses. The first address is the address of
# the tun interface. The second address is a dummy and
# actually not used.
ifconfig 192.168.0.1 192.168.0.2
# Client address pool. Each client needs a subnet of
# 30 Bit netmask, i.e. 4 IP addresses
ifconfig-pool 192.168.0.4 192.168.0.255
# Push route to client so that it can reach the server's
# virtual IP address. The client uses its tun device as
# routing interface.
push "route 192.168.0.1 255.255.255.255"
# Set route to local subnet (VPN subnet)
push "route 192.168.1.0 255.255.255.0"
# Delete client instances after some period of inactivity
# (10 minutes)
inactive 600
# Set a local route to the virtual IP address net
# with the local tun device as routing interface
route 192.168.0.0 255.255.255.0
# Act as TLS server
tls-server
# Diffie-Hellman Parameters (tls-server only). Replace by your
# parameter file
dh /etc/openvpn/dh1024.pem
# Certificate Authority file
ca /etc/openvpn/my-ca.pem
# Our certificate/public key
cert /etc/openvpn/officecert.pem
# Our private key
key /etc/openvpn/officekey.pem
# Listening port (default is 5000)
# port 5000
# Verbosity level.
# 3 -- medium output, good for normal operation.
verb 3
```

6.4 VPN-Client („UserX“), `tls-userX.conf`

```
# User TUN device as virtual interface
```

```

dev tun
# Operate as VPN client.
# The pull parameter is required for connecting to a
# multi-client server. It tells the client the client to accept
# options which the server pushes to the client.
pull
# Authentication algorithm (see all with openvpn --show-digests)
auth SHA1
# Encryption algorithm (see all with openvpn --show-ciphers)
cipher AES-256-CBC
# Specify the remote VPN gateway (official address)
remote 192.168.2.1 # Replace by real IP address
# Act as TLS client
tls-client
# Certificate Authority file
ca my-ca.pem
# Our certificate/public key
cert userXcert.pem
# Our private key
key userXkey.pem
# Server port (default is 5000)
# port 5000
# Verbosity level.
# 3 -- medium output, good for normal operation.
verb 3

```

6.5 Starten der Verbindung

Bevor die VPN-Verbindung aufgebaut werden kann, sind noch einige Voraussetzungen zu kontrollieren:

- TUN-Device /dev/net/tun vorhanden?
 - Falls nicht: `mkdir /dev/net mknod /dev/net/tun c 10 200`
- Kernel-Modul geladen?
 - Falls nicht: `modprobe tun`
- IP-Forwarding auf Gateway eingerichtet?
 - Falls nicht: `echo 1 > /proc/sys/net/ipv4/ip_forward`

Der Server-Prozess wird nun mit dem folgenden Kommando gestartet:

```
openvpn --config /etc/openvpn/tls-office.conf
```

Ein Client wird mit dem folgenden Kommando zum Aufbau der Verbindung angestoßen:

```
openvpn --config tls-userX.conf
```

Unter Windows sollte die Date `tls-userX.conf` am besten in `tls-userX.ovpn` umbenannt werden, damit der OpenVPN-Client über die rechte Maustaste gestartet werden kann („Start OpenVPN on this config file“).

7 Fazit und Ausblick

Zwar lässt sich derzeit mit Open-Source-Software noch kein vollständiges SSL-VPN-Gateway bauen. Einige der Vorteile von SSL lassen sich jedoch mit Hilfe von OpenVPN auch mit freier Software nutzen. Hierzu zählt die Verfügbarkeit, wenn man den OpenVPN-Serverport auf HTTPS (443) legt und somit in der Praxis von überall aus Zugriff auf das VPN hat. OpenVPN ist einfach zu konfigurieren und sowohl für Site-to-Site als

auch für Remote Access einsetzbar. Denkbar wäre zukünftig die Entwicklung eines Browser-Applets als freie Software, die mit Apache funktioniert und ein SSL VPN nach kommerziellem Vorbild realisiert.

SSL wird IPSec und andere VPN-Protokolle nicht ersetzen. Jede Variante hat ihre spezifischen Stärken, Schwächen und ihr jeweiliges Einsatzgebiet. Die Wahl einer VPN-Lösung ist jeweils aus den Gegebenheiten und Anforderungen abzuleiten.

8 Literaturangaben und Links

1. OpenVPN-Homepage: <http://openvpn.sourceforge.net>
2. Dr. Andreas Steffen: Eigener Schlüsseldienst. VPNs und Zertifikate im Eigenbau. c't 5/2002, S. 220
3. Jörg Fritsch, Jürgen Luksch: Schutzportale - SSL-Gateways als „Volks-VPNs“. iX 1/2004, S. 110
4. Achim Leitner: Virtuelle private Netze ohne Kernel-Modifikation: OpenVPN. Linux-Magazin 10/2003, S. 29
5. RFC 2246: The TLS Protocol Version 1.0. <http://www.faqs.org/rfcs/rfc2246.html>
6. Bruce Schneier, David Wagner: Analysis of the SSL 3.0 Protocol. <http://www.schneier.com/paper-ssl.html>
7. Bruce Schneier, Niels Ferguson: A Cryptographic Evaluation of IPsec. <http://www.schneier.com/paper-ipsec.html>
8. Bruce Schneier, Mudge, David Wagner: Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2). <http://www.schneier.com/paper-pptpv2.html>