Creating VPN Overlay Networks

Securing *Your* Internet For Fun And Profit

Travis H.

AHA 29 Apr 2009

Problem Statement

- Have multiple machines in multiple locations
- Wish to secure as much traffic between them as possible
- For extra credit, assume heterogeneous OSes
- Solution should be as transparent as possible
- Solution should work with NAT
- For extra credit, support non-NAT-able protocols

Multiple Machines

- You're all hackers, so you probably already have multiple machines
- Redundancy (esp. DNS, SMTP)
- Segregate functions to minimize impact of a system compromise

Multiple Locations

- Systems at home on cheap providers (and thus, dynamic IPs)
- Hosted or co-located servers at remote data centers:
  - Good bandwidth
  - Better connectivity
  - Static IPs
- Laptop which could be used at any wifi hotspot

Heterogeneous OSes

- Flexibility is nice in general
- Not all software is available for (your favorite OS here)
- You may have limited choices of OSes that a hosting provider will install

Transparency

- Many solutions require users to change their behavior
- This is unreliable
- Security is used most often when it requires no extra effort
- Goal: To secure the traffic without requiring user to change how he works

Working With NAT

- Native IPv6 isn't available from most residential ISPs
- IPv6 tunnels add to latency, provide points for centralized monitoring
- Paying for multiple static IPv4s can be pricey
- NAT is a very cost-efficient kluge

Non-NAT-able Protocols

- Certain protocols embed IP addresses in the layer 7 data
- FTP, talk, IRC DCC, SIP, etc.
- It sure would be nice to be able to use them within the VPN's borders

Securing Traffic

- Many protocols (e.g. telnet) are impossible to encrypt at application level
- Some protocols (e.g. SMTP) are possible to encrypt but takes effort on a per-node basis
- By VPNing the machines together, we secure **all** traffic between our systems

Dynamic IP Resilience

- When you pick up a laptop and move to another hot spot, all your connections die
- This need not be necessary when using VPNs

Untrusted WLANs

- Wifi networks are notoriously easy to sniff or tamper with
- Allow for tunnelling **all** of laptop traffic to another host
- Now we don't care about the security of the WLAN

Browsing on Untrusted Networks

- If you rely on browser proxies to secure your traffic
- It may be a good idea to set your home page to about:blank or a file on the local file system
- This allows you to start up your browser safely so that you can configure the proxy
- It also prevents exposing non-secure cookies to the untrusted network

Untrusted ISPs

- ISPs are starting to deploy Deep Packet Inspection (DPI) devices
- DPI allows for monitoring and tampering with data stream
- Same threats as untrusted WLAN
- Allow for tunnelling **all** home network traffic to another host

Extending The VPN

- Securing traffic between our own systems is a good idea
- Why not extend that to secure traffic between ourselves and our friends?
- By offering some services such as video over SIP, we can have e.g. secure videoconferencing

Thwarting Eavesdropping

- When you're done it's easy to do almost *everything* over the VPN
- This means that eavesdroppers only see VPN traffic, don't know what it is

Implications of Having VPN

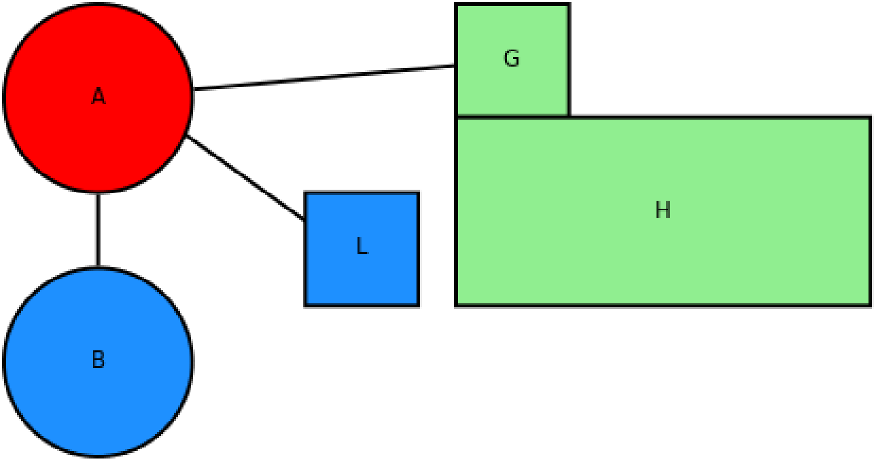
- If your VPN is reliable enough, you can now firewall off e.g. SSH
- Instead, you SSH over the VPN
- Neatly eliminates exposure to SSH flaws or password guessing

Concrete Scenario

For concreteness, let's assume the following scenario:

- Network composed of four machines (A,B,G,L) and one network (H)
- Two servers (A, B) at geographically dispersed locations
- One home network (H) with a gateway (G), one laptop (L)

Concrete Scenario



Why Two Servers?

- Nearly eliminates single point of failure (SPOF)
- Allows for geographically redundant services (DNS, SMTP)
- Don't have to trust other people with these critical services

Quote

We control the horizontal, and the vertical.
– *The Outer Limits*

VPN IP Addresses

- Every VPN solution I've used requires distinct IP addresses for the VPN
- Every host has its normal IP address and a VPN IP address
- But what addresses should we use?

RFC 1918

- class A: **10.0.0.0 - 10.255.255.255** (10/8)
- class B: **172.16.0.0 - 172.31.0.0** (172.16/12)
- class C: **192.168.0.0 - 192.168.255.255** (192.168/16)
- But which to use?

Unused IP Range

- If there is a conflict between VPN IPs and “real” (non-VPN) IPs, you will have problems
- Routing tables will have two routes with same network address
- VPN will not work if local LAN has same range
- Your systems won’t know which interface to route on

Use of RFC 1918

- **192.168.0.0/24** and **192.168.1.0/24** often used by wireless routers
- **10.0.0.0/24** and **10.1.1.0/24** used by companies and some ISPs
- NOTE: TW/RR is known to expose **10.0.0.0/24** to customers as default routes!

Picking IP Blocks

- It appears that the "class B" is used least frequently
- When you need a network, randomly select a /24 from RFC 1918 class B addresses

IPSEC With Static Keys

- Simplest configuration (for IPSec)
- Most IPSec VPNs are configured this way
- Similar to the way WLANs are configured
- Not very secure, we can do better

Static Key Tradeoffs

- Advantages:
  - Simple to configure (as IPSec goes)
  - Most compatible IPSec configuration
- Disadvantages:
  - Every node knows the shared key, so every node can impersonate every other node
  - Compromise of one node means having to re-key the entire network

ISAKMPD

- The hotness from OpenBSD
- Manages keys for IPSec
- Uses Keynote trust management system
- Available on Linux
- Can detect going through NAT and automagically enable IPSec-in-UDP encapsulation
- Provides "Perfect Forward Secrecy" (PFS)

ISAKMPD Disadvantages

- Only available on OpenBSD and Linux - no Windoze, Mac, Cisco
- Unreliable on Linux
- Very difficult to configure (esp. with x.509 certificates)
- Nearly impossible to troubleshoot (esp. on Linux)

IPSec Summary

- Complex[4]
- Difficult to configure
- Very difficult to troubleshoot
- Generally relies upon seperate IP protocols ESP and AH which may not pass through some network devices
- Generally does not play well with NAT

SSH Tunnelling
Generally Speaking

- SSH is a TCP-based protocol, connection-oriented
- Require a special procedure to establish a connection, similar to having to dial a modem
- Unstable, connections broken due to:
  - Network problems
  - Changing source IPs
- Can be made to reconnect automatically with autossh

TCP over TCP

This VPN Is Problematic

- Lose some MTU due to TCP protocol overhead
- Tunnelling TCP over TCP can lead to excessive retransmission after TCP timeouts
- This gradually creates more and more retransmissions, leading to loss of efficiency

SSH Port Forwarding

- Easy to configure; AllowTcpForwarding defaults to yes
- Users tend to get local and remote sides confused
- Non-transparent: Must remember to connect to a certain port on localhost instead of the intended destination
- Audit Unfriendly: When logging a connection, source IP address is wrong
- Point solution: must configure a tunnel for every destination host/port combination
- Supports forwarding TCP only! UDP goes out as normal.

Dynamic Port Forwarding With SSH

- Works as SOCKS v4/5 proxy
- Non-universal: Requires that client applications be SOCKSified
- Still only works with TCP
- Tip: Set network.proxy.socks_remote_dns = true in about:config to tunnel DNS through SOCKS

Layer 2/3 Forwarding With OpenSSH

- Dark Horse VPN candidate
- Allows you to route (layer 3) or bridge (layer 2) connections over OpenSSH
- Controlled by PermitTunnel sshd_config entry
- Uses TUN device
- Usability: requires manually configuring IPs and routes through tunnel

OpenVPN

- OpenVPN[1] is built around OpenSSL and the TUN/TAP drivers
- It uses UDP by default, but can use TCP
- Can work on any port to evade port-based filtering
- Very easy to configure
- Supports easily calling external programs on certain events, making customization trivial

OpenVPN Drawbacks

- Although it uses UDP, it relies upon client/server model
- Tunnels are point-to-point
- Not clear how to configure overlay without a central node (SPOF)

Terminology

hub The central VPN server for the network.

core network Those systems which talk directly to the hub.

extension network Those systems which talk to the hub only through a gateway which is on the core network.

The Core Network

1. Pick a random /24 for all nodes in the core. We use **172.20.219.0/24**.
2. Pick one of the servers to use as the hub. We use **A**. Assign it **172.20.219.1**.
3. Every system in the core needs to be assigned an IP address from this IP range. OpenVPN uses two IPs for every point-to-point connection, so only use odd addresses. E.G:
4. Node **B** might be **172.20.219.3**.
5. Node **G** might be **172.20.219.5**.

Certificate Creation

With Easy-RSA Scripts

- edit file "vars"
- ./build-dh
- ./build-ca
- ./build-key-server **A**
- ./build-key-server **B**

Certificate Creation

Other Options

- Graphical CA software:
  - tinyca2
  - xca

Hub Configuration

IP Forwarding

- BSD:
  - sysctl net.inet.ip.forwarding=1
  - enable same in /etc/sysctl.conf
- Linux:
  - echo 1 > /proc/sys/net/ipv4/ip_forward
  - enable same in /etc/rc.local

Hub Configuration

OpenVPN Basics

proto udp
dev tun0
ca ca.crt
cert *hostname*.crt
key *hostname*.key
dh dh2048.pem
server **172.20.219.1** 255.255.255.0
verb 3

Hub Configuration
OpenVPN Extras

Numerous useful options:

- **float** Allows remote peers to change IP addresses at will w/o restarting tunnel
- **ifconfig_pool_persist** Gives clients the same IP each time they come back
- **client-to-client** Allows core clients to communicate with each other through hub
- **comp-lzo** Enable adaptive compression on link
- **persist-tun** follow the DNS name of the *server* if it changes its IP address
- **keepalive 10 120** keep a connection through a NAT router/firewall alive

Extending The Network

Network Planning

- Each extended network needs unique VPN IP block
- Pick at random: **172.26.238.0/24**
- Must not conflict with real IPs or other VPN IPs
- This means that extension networks may need renumbering
- Easiest to do if you use DHCP and DNS internally

Extending The Network

Internal DHCP and DNS

- Assign static IPs in **172.26.238.0/24** (randomly chosen)
- If you have to renumber, it may be best to configure DNS and DHCP on **H** network
- We'll get back to this later

Extending The Network

Implementation

- Turn on IP forwarding in the gateway host (**G**)
- That's the only change on the client side!

Extension Networks

Hub Global Configuration

- push "route **172.26.238.0** 255.255.255.0"
- route **172.26.238.0** 255.255.255.0
- client-config-dir ccd

Extension Networks

Client Config Directory

- iroute **172.26.238.0** 255.255.255.0
- ifconfig-push **172.20.219.5 172.20.219.1**

Configuring DNS

- We've got an IP-layer network set up.
- Can use VPN based on IP addresses all we want. Could stop here!
- Next step for usability is to configure DNS for this network
- This involves creating bogus domains
- Let's call our bogus TLD **mine**
- We'll put all core VPN nodes in a SLD called **c.mine**

Configuring DNS

Hub Changes For Forward DNS

- Make **A** authoritative for **mine**, **c.mine**:
- zone "mine" { type master; file "mine"; };
- zone "c.mine" { type master; file "c.mine"; };
- **mine** will only delegate to SLDs
- In this case, the name server for **mine** and **c.mine** should be **A**
- **c.mine** will map hostnames to core VPN IPs

Configuring DNS
Hub Changes For Reverse DNS

- Reverse DNS isn't required but is nice for logging purposes
- Make **A** authoritative for reverse DNS zone
- zone "219.20.172.IN-ADDR.ARPA" { type master; file "172.20.219"; };
- This should map VPN IPs back into *hostname*.**c.mine**
- Make **A** the official nameserver for this zone
- Now we have DNS configured minimally!

Configuring DNS
Adding DNS Redundancy

- Now configure the other server **B** as your DNS backup
- zone "mine" { type slave; file "mine"; masters { 172.20.219.1; }; };
- zone "c.mine" { type slave; file "c.mine"; masters { 172.20.219.1; }; };
- zone "219.20.172.IN-ADDR.ARPA" { type slave; file "172.20.219"; masters { 172.20.219.1; }; };
- Make **B** another name server for all three zone files

Configuring DNS
Resolver Configuration

- Need to think about how DNS lookups are performed
- Every node on VPN needs to have its resolv.conf point to a DNS server that has been configured to know about these bogus domains

Configuring DNS
Simplest Solution

- Configure all resolv.conf files to point to **A** and **B**'s VPN IPs
- nameserver 172.20.219.1
- nameserver 172.20.219.3
- Disadvantages:
  - slow
  - nothing will work if VPN connection is down

Configuring DNS
Best Solution

- For every node in the core, configure its DNS server to slave that domain from **A**
- This is done identically to the changes we did on **B** earlier
- It is not necessary to slave **c.mine** because **mine** already delegates to **A** and **B** as **c.mine**'s name server
- However it could be useful for performance to "cache" the zone locally

Resolver Search Paths

- Now we can use fully-qualified (bogus) domain names for all purposes
- The next step is to make it *easier* to use bogus domain names than normal ones
- This is done with "search" command in /etc/resolv.conf
- search c.mine *yourdomain.tld*
- Now you can access any core VPN host using only its hostname!

DNS For Extension Networks

- Run two DNS servers internally (master **M**, slave **S**)
- Create a bogus SLD, like **h.mine** with **M** and **S** as name servers
- Delegate this domain to them in the **mine** TLD
- Also delegate reverse DNS zones
- By segregating zone files each extension network can be independently administered
- Add **h.mine** to search paths in /etc/resolv.conf

DHCP For Extension Networks

Optional But Makes Renumbering Easy

- Run two DHCP servers internally (primary, secondary)
- Assign DNS names based on MAC address
- DHCP server is smart enough to resolve them to IPs when assigning to clients
- Makes renumbering the network easy; just change DNS and reboot every system

DHCP For Extension Networks

Optional But Makes Resolver Configuration Easy

- Using DHCP, we can set the search path and name servers for all clients on the LAN
- Avoids manually configuring /etc/resolv.conf on every host on the extension network

NTP

- Most people never consider it
- Essential for security log correlation
- OpenVPN uses timestamps in the protocol to detect backtracking
- Set it up on **A** and **B**, peer with each other
- Also configure it to use public NTP servers
- Can be done over VPN IPs but will increase jitter and latency
- Set it up on every node in VPN, use core systems as servers

Syslog

- It can be handy to have every node syslog to a central logging host
- Remote logging can help forensics if a machine is compromised and logs wiped
- Definitely want to do this using VPN domain names or IPs
- Easy to do:
- *.* @loghost.c.mine

OpenVPN Hardening

Misc Commands

user _openvpn Run as pseudo-user _openvpn

group _openvpn Run in special group _openvpn

persist-key Keep the key file descriptor open across restarts

OpenVPN Hardening

MITM Protection

- Want to keep other nodes from impersonating the server
- In each client, configure the following:

remote-cert-tls server
: For recent versions of OpenVPN

ns-cert-type server
: For older versions of OpenVPN

OpenVPN Hardening

Crypto Enhancements - Ciphers

- openvpn –show-ciphers
- Pick the strongest one that all your nodes support
- I recommend: cipher AES-256-CBC

OpenVPN Hardening

Crypto Enhancements - Digests

- openvpn –show-digests
- Pick the strongest one that all your nodes support
- I recommend: auth SHA512

OpenVPN Hardening

Crypto Enhancements - TLS cipher

- openvpn –show-tls
- I recommend tls-cipher DHE-RSA-AES256-SHA

OpenVPN Hardening

Crypto Enhancements - TLS auth

- This is to require a symmetric key operation (HMAC) before doing expensive public-key
- Mitigates CPU-based DoS
- openvpn –genkey –secret tls-auth.txt
- Securely copy it to every node and add this to every config:
- tls-auth tls-auth.txt

OpenVPN Stealth

When You Want a Virtual *Private* Network

- Change the default port to HTTPS, route through squid, pretend to be Mozilla on W2k
- port 443
- proto tcp-client
- http-proxy squidhost 3128
- http-proxy-retry
- http-proxy-option AGENT Mozilla/4.0 (compatible; MSIE 4.01; Windows NT 5.0)

OpenVPN Customization

Places You Can Invoke Custom Scripts

learn-address	When the IP of a VPN partner changes
ipchange	When the IP of the server changes
client-connect	When a client connects
client-disconnect	When a client disconnects
up,down	After configuration of the TUN/TAP device
down-pre	Before shttting down the TUN/TAP device
up-restart	When tunnels are restarted up/down scripts are also run

VPN Perimeter

- Every node which *directly* participates in the network forms part of the perimeter.
- These nodes should have roughly consistent packet filters enabled
- This prevents easy access to other VPN nodes
- These rules should be set up on the external interfaces
- Should avoid accepting packets destined for VPN IP ranges

Hub Firewall

- The central node can perform access control between clients.
- This will define the *maximum* access one client has to another
- This prevents easy "node hopping" once the outer perimeter is breached
- These rules should be set up on the tun device

Gateway Firewalls

- Gateway hosts (e.g. on home LAN) can do more filtering on tun device
- This allows each extension network to define what may or may not be accessed from the VPN

Name Exposure

- These DNS zones can be available to everyone or only when on VPN
- If we don't expose any, then we must use IP addresses in OpenVPN config files
- Safer to make these zones only visible once on VPN
- But name lookups will fail if the VPN is down

DNS Lockdown
Generally Speaking

- Good reference on the net[6]
- Disable recursive queries
- Restrict zone transfers to legal slaves
- Disable queries from other machines

DNS Lockdown

Defining ACLs

- You can use ACLs to define netblocks
- acl clients { localhost; ::1; };
- acl corevpn { 172.20.219.0/24; }
- acl allvpn { 172.20.219.0/24; 172.26.238.0/24; };

DNS Lockdown

Disabling Recursive Queries

- options { allow-recursion { clients; }; };
- This also helps prevent cache poisoning and laundering communication through your DNS server

DNS Lockdown

Restricting Zone Transfers

- This defines who can download your entire zone file
- zone "**mine**" { allow-transfer { allvpn; clients; }; };

DNS Lockdown

Disable Queries From Other Machines

- This defines who can make a query against your server
- options { allow-query { localhost; }; };
- This states that VPN nodes and localhost can query the bogus TLD domain
- zone "**mine**" { allow-query { allvpn; clients; }; };

SSH Security

- Already using AES256-CBC for VPN
- Can use a faster cipher (blowfish) for intra-VPN connections
- This is superencryption (two different ciphers), much better than just one
- SSH config file:
- Host *.c.mine
- Cipher blowfish

Summary

- VPNs provide a number of advantages to the security-conscious user
- They aren't that hard to configure

For Further Reading I

- OpenVPN Homepage
  http://openvpn.net/
- M. Feilner.
  OpenVPN: Building and Integrating Virtual Private Networks
  Packt Publishing, 2006.
- RFC 1918
  http://www.faqs.org/rfcs/rfc1918.html
- N. Ferguson, B. Schneier
  A Cryptographic Evaluation of IPsec
  http://www.schneier.com/paper-ipsec.html

For Further Reading II

D. Mazzochio
Building VPNs on OpenBSD
http://www.kernel-panic.it/openbsd/vpn/index.html

J. Norish
DNS Basic Security Options
http://www.langfeldt.net/DNS-HOWTO/BIND-9/DNS-HOWTO-6.ht