Sshuttle, the unilateral VPN

Avery Pennarun

2011 04 30 (rev.b)

What's a VPN?

- Connect to computers on the other side of a firewall as if you were there

IPsec Sucks

- The most nonstandard standard ever passed by the IETF
- Virtually all implementations are incompatible
- Not designed for NAT
  - every NAT router had to add *special support* for it
- Requires kernel-level driver
- Bloats packets
- Requires a genius to configure

OpenVPN Sucks Less

- There's only one, so it interoperates
- Free
- Not written by morons
- Still requires kernel-level support
- Still hard to configure
- Still requires support from your admin
  - ...who probably prefers IPsec...
- No blessing from the IETF

ssh

- Exists everywhere there's Unix
- Easy to set up: often installed by default
- Works with simple password authentication
- But allows fancy public key crypto
- Sucks at port forwarding

Sshuttle

- A VPN on top of ssh
- Works with any ssh server
- Exactly as easy as ssh
- Leaves all the crypto to ssh
- Gets through NAT as easily as ssh
- Needs no admin access on the server

Digression: TCP-over-TCP

- The obvious way to route packets over ssh is a disaster
  - I should know, I did it
- TCP depends on packet loss to control its speed
- But when carrying TCP over TCP, the inner TCP never experiences packet loss because the outer TCP fixes it

Double Digression: Rate Limiting 1a

- If both ends of a connection are on 100 MBit ethernet...
- How do they know the link between is only 1 MBit?

Double Digression: Rate Limiting 1b

- They don't!
- The slow link drops packets if you send too fast
- And TCP notices this and slows down

Double Digression: Rate Limiting 2a

- What if two people on your fast network are transmitting at once?
- How do they know to transmit at only 500 kbits each?

Double Digression: Rate Limiting 2b

- They don't!
- Statistically, the slow link drops more packets from the fastest sender
  - (5% of a larger number...)
- And so the fastest one slows down more!

Double Digression: Rate Limiting 3a

- What if there are 25 links in the chain and millions of computers? How do they know how fast to send?

Double Digression: Rate Limiting 3b

- You guessed it...

Triple Digression!!!1! Bufferbloat (Google it)

- Bufferbloat is caused by people who hate packet loss
- People see routers dropping packets and try to “fix” it by adding big buffers
- Your cable modem or DSL router does this
- The Linux kernel does this
- And it destroys TCP performance
- That's the *only reason* why large uploads (Bittorrent) make your Internet suck

Bufferbloat: The Delay*Bandwidth Product

- If it takes 0.1 secs to get from A to B...
- ...and the uplink is 500 kbits (50 kbytes/sec)
- Then you have about 0.1*50 = 5 kbytes in flight at any moment
- a 500 kbyte buffer means *10 seconds of latency*
- The “right” buffer is roughly bandwidth*delay
  = 5 to 10 kbytes
- “As big as possible” is **always wrong!**

Um, where were we?

    - Bufferbloat sucks
  - TCP depends on packet loss
- TCP-over-TCP is like infinite bufferbloat
  -> infinite suck
- So if you take TCP packets and route them over ssh (ie. TCP)...

...then you'd better be smart

- Sshuttle doesn't do TCP-over-TCP
- Doesn't use a “tap” interface and capture packets
- Uses “transparent proxying” instead
- Kind of like the old SLiRP tool
- Unwinds the inner TCP and sends the payload

Benefits

- No TCP-over-TCP crap
- Unfails TCP Slow Start algorithm by sharing it between connections
- **Reduces** TCP/IP overhead instead of adding it (especially by merging small packets)
- Improves crypto:
  - Removes obvious packet boundaries and timings (unlike IPsec/OpenVPN)
  - Can use simple, well-understood streaming encryption (ssh is more trustworthy than IPsec)

Bonus: Bufferbloat Defeater

- Like everyone else nowadays, ssh has its own extra layer of bufferbloat
- sshuttle uses a cheap hack to keep latency low even in heavy-traffic situations
- So if you upload and ssh at the same time, use sshuttle... it'll suck **less**

But that's not all...

Self-assembly

- You don't have to install sshuttle on the server
- It uploads and runs itself!
  - Internet worm style
- There's never a server version mismatch
- The server always has the latest features
- We don't have to standardize a protocol
  - ssh already did

Self-assembly: Phase 1

```
ssh myserver “
   python -c '
        import sys;
        skip_imports=1; verbosity=0;
        exec compile(sys.stdin.read(764),
          \"assembler.py\", \"exec\")
    '
”
```

Self-assembly: Phase 2

```
import sys, zlib
z = zlib.decompressobj()
mainmod = sys.modules[__name__]
name = 1
while name:
    name = sys.stdin.readline().strip()
    if name:
        nbytes = int(sys.stdin.readline())
        content = z.decompress(sys.stdin.read(nbytes))
        exec compile(content, name, "exec")
        mainmod.__dict__[name[:-3]] = mainmod
main()
```

DNS Remangling (--dns)

- Sshuttle mostly doesn't support UDP (yet)
  - UDP-over-TCP also sucks :(
- But we make an exception for DNS
- Packets destined for your local nameserver get bounced to the remote one instead
- And answers pretend to come from your local one

Magic DNS Discovery (--auto-hosts)

- But sometimes you don't want *all* your DNS going to the remote server
- Maybe you only want the remote computer names in your local DNS
- sshuttle polls the remote network to find all the nearby computers
  - nmblookup, smbclient, domain controller
  - reverse DNS, netstat -a
  - and so on

Magic Route Discovery (--auto-nets)

- Setting up a VPN is hard because you have to know which subnets to route
- --auto-nets just asks the server which routes are local
- And sets up your local routing to send those over the VPN

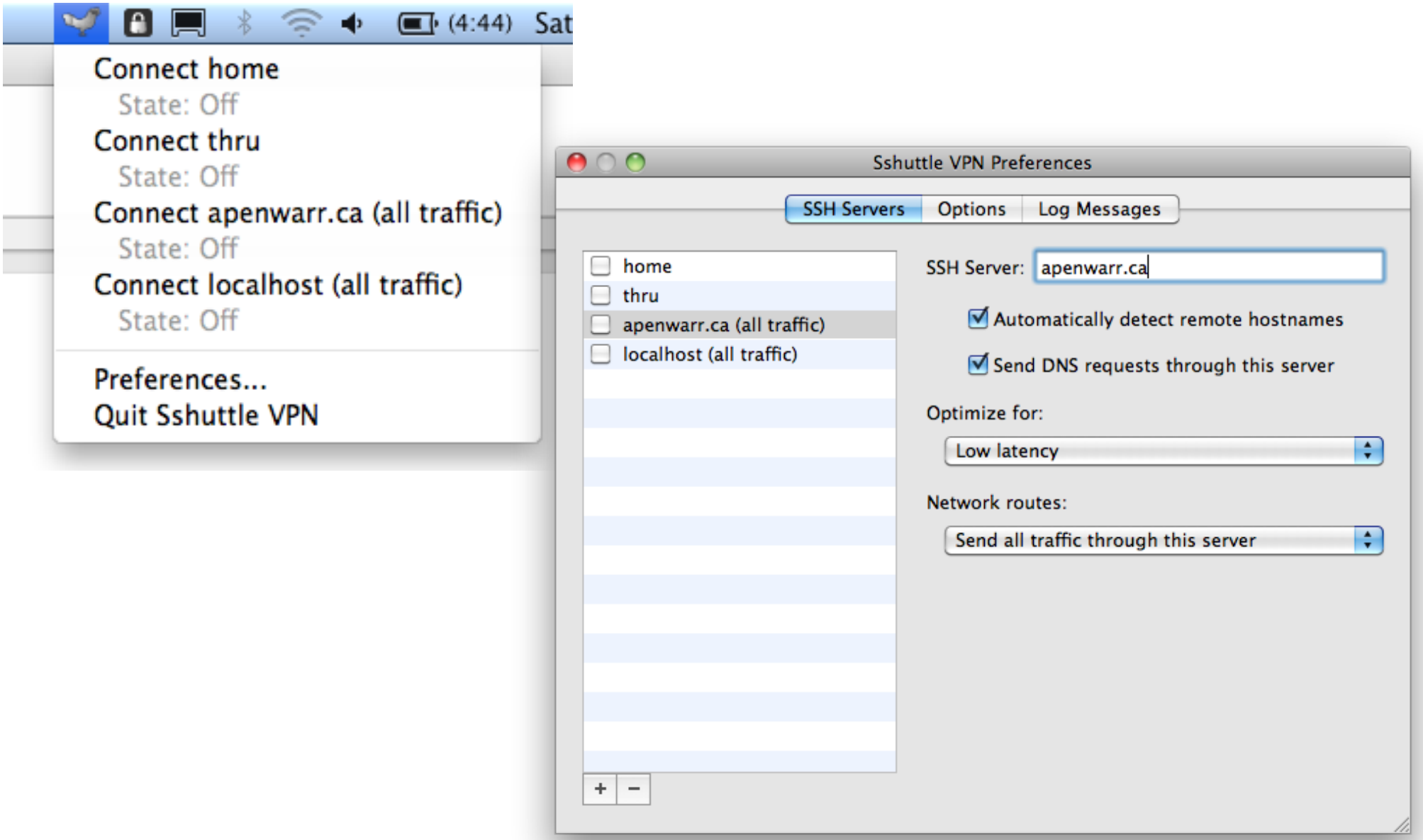
A little note about Firesheep

- Use sshuttle, and you're immune.

Net Result

- Everything you need to connect to your office network:
  - ./sshuttle -NHr myserver
- Everything you need to bypass Firesheep:
  - ./sshuttle --dns -r myserver 0/0

Also, there is a fancy MacOS GUI



Demo

Questions?