Beyond Two Cans and a String

A practical guide to building a router

Ben Lewis

8 March 2020

whoami

- Software engineer by day
- I care too much about RFCs
- In my spare time, I work on my hackerspace's network.

I’m not my employer, and this talk is composed of my personal opinions and observations.

Consumer routers bore me

Random failures

Be it ARP table overflows, memory leaks, overheating, or poor antenna performance, I've just not had good luck with consumer routers.

Software stack

Weak configuration options-often limited to port forwarding or sticking a machine in a DMZ-and minimal firewall options beyond simply providing NAT.

A router's configuration screen, showing a list of "famous games" including Bittorrent. From @tjhorner on Twitter.

Security vulnerabilities

- Insecure default passwords and configurations

Security vulnerabilities

- Insecure default passwords and configurations
- Out-of-date protocol stacks

Security vulnerabilities

- Insecure default passwords and configurations
- Out-of-date protocol stacks
  - WPAv3

Security vulnerabilities

- Insecure default passwords and configurations
- Out-of-date protocol stacks
  - WPAv3
  - TLS 1.3

Security vulnerabilities

- Insecure default passwords and configurations
- Out-of-date protocol stacks
  - WPAv3
  - TLS 1.3
  - SHA-1 deprecation

Security vulnerabilities

- Insecure default passwords and configurations
- Out-of-date protocol stacks
  - WPAv3
  - TLS 1.3
  - SHA-1 deprecation
- Vendor security vulnerabilities

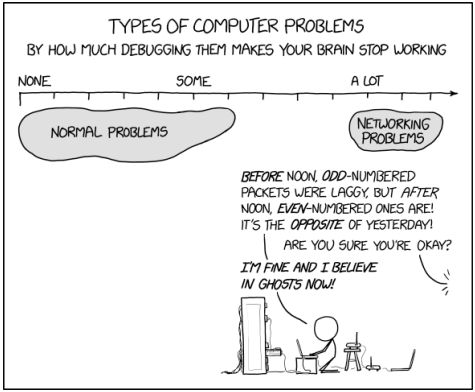
Security vulnerabilities

- Insecure default passwords and configurations
- Out-of-date protocol stacks
  - WPAv3
  - TLS 1.3
  - SHA-1 deprecation
- Vendor security vulnerabilities
  - Netgear routerlogin.net private cert

Security vulnerabilities

- Insecure default passwords and configurations
- Out-of-date protocol stacks
  - WPAv3
  - TLS 1.3
  - SHA-1 deprecation
- Vendor security vulnerabilities
  - Netgear routerlogin.net private cert
  - Cisco NX-OS CVE-2020-3119

Complex systems encounter complex failures



Take a bunch of complex systems and stick them together… it'll be fine, right?
(XKCD 2259, "Network Problems". CC-BY-NC 2.5)

Goals for our router

Be upgradeable

Be upgradeable

- Have expansion ports

Be upgradeable

- Have expansion ports
- Support future protocols and needs

Be upgradeable

- Have expansion ports
- Support future protocols and needs
- Avoid getting trapped with insufficient hardware

Be easy to manage

Be easy to manage

- Familiar network analysis tools

Be easy to manage

- Familiar network analysis tools
- Interfaces that we use routinely

Be easy to manage

- Familiar network analysis tools
- Interfaces that we use routinely
- Your choice of package management

Be a test bed for exploring network ideas

A full operating system on your router offers you lots of opportunities to try out different tools or technologies

Be a test bed for exploring network ideas

A full operating system on your router offers you lots of opportunities to try out different tools or technologies

- Docker containers on your router

Be a test bed for exploring network ideas

A full operating system on your router offers you lots of opportunities to try out different tools or technologies

- Docker containers on your router
- Dynamic DNS

Be a test bed for exploring network ideas

A full operating system on your router offers you lots of opportunities to try out different tools or technologies

- Docker containers on your router
- Dynamic DNS
- A little home website

Be a test bed for exploring network ideas

A full operating system on your router offers you lots of opportunities to try out different tools or technologies

- Docker containers on your router
- Dynamic DNS
- A little home website
- Logging or traffic analysis

Selecting hardware

Constraints & parameters

RAM

4GB of RAM is more than enough …

RAM

4GB of RAM is more than enough …

…but if you want to do exciting projects, you might want even 16 or 32.

CPU

You don’t need a lot of power here; an Intel Atom is plenty for basic routing needs.

CPU

You don’t need a lot of power here; an Intel Atom is plenty for basic routing needs.

I’d always recommend a multicore CPU, however.

Storage

Storage

- Minimal amount needed to start - a flash drive will do

Storage

- Minimal amount needed to start - a flash drive will do
- Speed of storage is not a priority

Storage

- Minimal amount needed to start - a flash drive will do
- Speed of storage is not a priority
- Extra capacity for bigger projects can be on a normal SATA HDD

NICs

NICs

- Onboard, on-motherboard NICs

NICs

- Onboard, on-motherboard NICs
- PCIe card NIC

NIC caveats

NIC caveats

- Drivers a consideration

NIC caveats

- Drivers a consideration
- Link Aggregation/Bonding and VLANs

NIC caveats

- Drivers a consideration
- Link Aggregation/Bonding and VLANs
- Initial capacity requirement

NIC caveats

- Drivers a consideration
- Link Aggregation/Bonding and VLANs
- Initial capacity requirement
- System I/O capacity for future upgrades

Approaches

Pre-built

Pre-built

- Netgate firewalls

Pre-built

- Netgate firewalls
- fitlet IoT Gateways

Pre-built

- Netgate firewalls
- fitlet IoT Gateways
- QNAP

Small Form Factor

Small Form Factor

- Standard desktop box

Small Form Factor

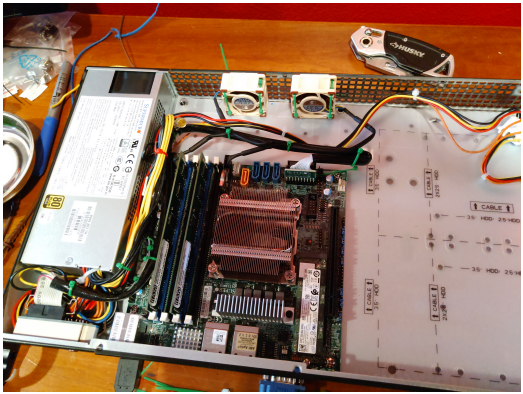
- Standard desktop box
- PCEngines APU units

Rackmount

Build your own

Rackmount

Build your own



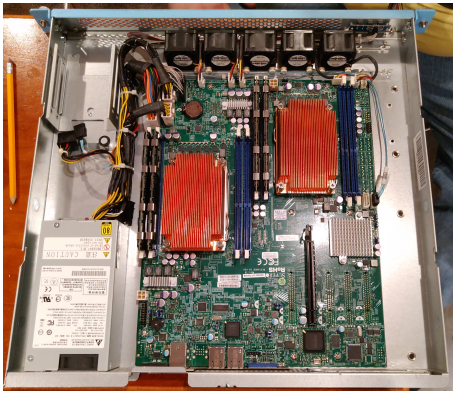
A custom Supermicro build in a 1U switch-depth chassis

Rackmount

…or buy used!

Rackmount

…or buy used!



A HYVE Zeus secondhand server

Considerations for expandability

Future network standards

Future network standards

- Gigabit now, what next?

Future network standards

- Gigabit now, what next?
- Wireless upgrades

Additional different hardware

Additional different hardware

- Tensorflow PCIe!

Additional different hardware

- Tensorflow PCIe!
- TPM for security stuff

Additional different hardware

- Tensorflow PCIe!
- TPM for security stuff
- Cellular connection for reliability

Additional different hardware

- Tensorflow PCIe!
- TPM for security stuff
- Cellular connection for reliability

Photo by Fons Heijnsbroek on Unsplash

Software stack

Axes of choice

Interface style

Interface style

- Graphical/Web
  - pfSense
  - OPNsense
  - clearOS
  - zeroshell

Interface style

- Graphical/Web
  - pfSense
  - OPNsense
  - clearOS
  - zeroshell
- Textual
  - firewall-cmd (on any distro that supports it)
  - shorewall
  - VyOS - both free and paid, this one's complicated. Derivative of Brocade Networks' OS.
  - Raw nftables on any recent Linux version

Preference in base OS

The mon0wall derivatives (pfSense, OPNsense) are all FreeBSD derivatives; in other cases, you may prefer running a Linux kernel-for familiarity's sake, or because of hardware support.

Support model

- Community-only
- Community version with paid extension
- Paid-only

Configuring a router

So, we have the parts, how do we set it up?

State should be self-maintaining

State should be self-maintaining

- System configuration should be automatic
- Reboot often to test your configuration

State should be self-maintaining

- System configuration should be automatic
- Reboot often to test your configuration
- High uptimes don't equate to reliability

State should be self-maintaining

- System configuration should be automatic
- Reboot often to test your configuration
- High uptimes don't equate to reliability
- Lots of tooling isn't necessary for a one-off build

Important software to install before starting your journey

Set up some basic tools to simplify your configuration process. I've chosen a few examples to use:

Important software to install before starting your journey

Set up some basic tools to simplify your configuration process. I've chosen a few examples to use:

- Text editor: `nano`

Important software to install before starting your journey

Set up some basic tools to simplify your configuration process. I've chosen a few examples to use:

- Text editor: `nano`
- Terminal multiplexer: `tmux`

Important software to install before starting your journey

Set up some basic tools to simplify your configuration process. I've chosen a few examples to use:

- Text editor: `nano`
- Terminal multiplexer: `tmux`
- System hardware monitoring: `ipmitool`

Choose your network configuration tool

Choose your network configuration tool

- systemd-networkd

Choose your network configuration tool

- systemd-networkd
- NetworkManager

Choose your network configuration tool

- systemd-networkd
- NetworkManager
- netplan

Interface-specific configuration

Different aspects of our network configuration will go through different areas:

Interface-specific configuration

Different aspects of our network configuration will go through different areas:

- `/etc/sysctl.d` files for interface-specific settings

Interface-specific configuration

Different aspects of our network configuration will go through different areas:

- `/etc/sysctl.d` files for interface-specific settings
- Network configuration depending on your management mechanism

Interface-specific configuration

Different aspects of our network configuration will go through different areas:

- `/etc/sysctl.d` files for interface-specific settings
- Network configuration depending on your management mechanism
  - `/etc/NetworkManager/NetworkManager.conf`

Interface-specific configuration

Different aspects of our network configuration will go through different areas:

- `/etc/sysctl.d` files for interface-specific settings
- Network configuration depending on your management mechanism
  - `/etc/NetworkManager/NetworkManager.conf`
  - `/etc/systemd/network` files

NetworkManager config

NetworkManager

Static or dynamic IP allocation

Static IP

Useful for your gateway address on internal networks, or if you have a static IP allocation from your ISP.

Dynamic IP

Handy when your upstream address is provided by DHCP; less handy if you're trying to have routes declared statically.

Multiple subnets and restricted routing

Multiple subnets and restricted routing

- Non-routing subnet (`net.ipv4.conf.$IFACE.forwarding = 0`)

Multiple subnets and restricted routing

- Non-routing subnet (`net.ipv4.conf.$IFACE.forwarding = 0`)
  - Keep IoT devices off the wider internet

Multiple subnets and restricted routing

- Non-routing subnet (`net.ipv4.conf.$IFACE.forwarding = 0`)
  - Keep IoT devices off the wider internet
  - Securely access IP cameras

Multiple subnets and restricted routing

- Non-routing subnet (`net.ipv4.conf.$IFACE.forwarding = 0`)
  - Keep IoT devices off the wider internet
  - Securely access IP cameras
  - Examine untrusted devices?

Multiple subnets and restricted routing

- Non-routing subnet (`net.ipv4.conf.$IFACE.forwarding = 0`)
  - Keep IoT devices off the wider internet
  - Securely access IP cameras
  - Examine untrusted devices?
- Egress-only subnet

Multiple subnets and restricted routing

- Non-routing subnet (`net.ipv4.conf.$IFACE.forwarding = 0`)
  - Keep IoT devices off the wider internet
  - Securely access IP cameras
  - Examine untrusted devices?
- Egress-only subnet
  - Can reach the broader internet

Multiple subnets and restricted routing

- Non-routing subnet (`net.ipv4.conf.$IFACE.forwarding = 0`)
  - Keep IoT devices off the wider internet
  - Securely access IP cameras
  - Examine untrusted devices?
- Egress-only subnet
  - Can reach the broader internet
  - Can't see other local subnets

Multiple subnets and restricted routing

- Non-routing subnet (`net.ipv4.conf.$IFACE.forwarding = 0`)
  - Keep IoT devices off the wider internet
  - Securely access IP cameras
  - Examine untrusted devices?
- Egress-only subnet
  - Can reach the broader internet
  - Can't see other local subnets
  - Great for devices that need to phone home, but that you don't trust

nftables versus frontends

nftables versus frontends

- Not really a “versus” situation

nftables versus frontends

- Not really a “versus” situation
- Pick your frontend, or just write scripts

nftables versus frontends

- Not really a “versus” situation
- Pick your frontend, or just write scripts
- For the demo, we’ll use `firewalld`

Don’t block ICMPv6!

(Short digression into IPv6 here.)
ICMPv6 does a lot of heavy lifting in the IPv6 world! Here’s a few things it does:

Don’t block ICMPv6!

(Short digression into IPv6 here.)
ICMPv6 does a lot of heavy lifting in the IPv6 world! Here’s a few things it does:

- Router solicitation and advertisement (replaces DHCP)

Don’t block ICMPv6!

(Short digression into IPv6 here.)
ICMPv6 does a lot of heavy lifting in the IPv6 world! Here’s a few things it does:

- Router solicitation and advertisement (replaces DHCP)
- MTU discovery

Don’t block ICMPv6!

(Short digression into IPv6 here.)
ICMPv6 does a lot of heavy lifting in the IPv6 world! Here’s a few things it does:

- Router solicitation and advertisement (replaces DHCP)
- MTU discovery
- Neighbor solicitation and advertisement

Forwarding and NAT

Running extra services

You have a server at your disposal, make use of it!
You can make your router accessible and functional when you're at home, and away.

SSH

Being able to remotely connect to your firewall and check on the state of your local network is useful–especially if you're away from home, and want to make sure that a service is working. However, you'll want to take some precautions.

SSH

Being able to remotely connect to your firewall and check on the state of your local network is useful–especially if you're away from home, and want to make sure that a service is working. However, you'll want to take some precautions.

- Before enabling access on your WAN interface

SSH

Being able to remotely connect to your firewall and check on the state of your local network is useful–especially if you're away from home, and want to make sure that a service is working. However, you'll want to take some precautions.

- Before enabling access on your WAN interface
  - Add a key to `authorized_keys`

SSH

Being able to remotely connect to your firewall and check on the state of your local network is useful–especially if you're away from home, and want to make sure that a service is working. However, you'll want to take some precautions.

- Before enabling access on your WAN interface
  - Add a key to `authorized_keys`
  - Disable password auth

SSH

Being able to remotely connect to your firewall and check on the state of your local network is useful–especially if you're away from home, and want to make sure that a service is working. However, you'll want to take some precautions.

- Before enabling access on your WAN interface
  - Add a key to `authorized_keys`
  - Disable password auth
  - Disable root login

SSH

Being able to remotely connect to your firewall and check on the state of your local network is useful–especially if you're away from home, and want to make sure that a service is working. However, you'll want to take some precautions.

- Before enabling access on your WAN interface
  - Add a key to `authorized_keys`
  - Disable password auth
  - Disable root login
- When enabling on your WAN interface

SSH

Being able to remotely connect to your firewall and check on the state of your local network is useful–especially if you're away from home, and want to make sure that a service is working. However, you'll want to take some precautions.

- Before enabling access on your WAN interface
  - Add a key to `authorized_keys`
  - Disable password auth
  - Disable root login
- When enabling on your WAN interface
  - Optionally, use a nonstandard port

SSH

Being able to remotely connect to your firewall and check on the state of your local network is useful–especially if you're away from home, and want to make sure that a service is working. However, you'll want to take some precautions.

- Before enabling access on your WAN interface
  - Add a key to `authorized_keys`
  - Disable password auth
  - Disable root login
- When enabling on your WAN interface
  - Optionally, use a nonstandard port
  - Consider fail2ban, logging

Nice-to-have services

Nice-to-have services

VPN

Nice-to-have services

VPN

OpenVPN Well-known, broadly supported with clients for most platforms.

Nice-to-have services

VPN

OpenVPN	Well-known, broadly supported with clients for most platforms.
Wireguard	Recently integrated into the next kernel release, a faster but less broadly supported option.

Nice-to-have services

VPN

OpenVPN	Well-known, broadly supported with clients for most platforms.
Wireguard	Recently integrated into the next kernel release, a faster but less broadly supported option.

Dynamic DNS

Nice-to-have services

VPN

OpenVPN	Well-known, broadly supported with clients for most platforms.
Wireguard	Recently integrated into the next kernel release, a faster but less broadly supported option.

Dynamic DNS

- no-ip or similar for quick access

Nice-to-have services

VPN

OpenVPN	Well-known, broadly supported with clients for most platforms.
Wireguard	Recently integrated into the next kernel release, a faster but less broadly supported option.

Dynamic DNS

- no-ip or similar for quick access
- Get a domain name and use a nameserver with a DDNS API, and use a script to update

Nice-to-have services

VPN

OpenVPN	Well-known, broadly supported with clients for most platforms.
Wireguard	Recently integrated into the next kernel release, a faster but less broadly supported option.

Dynamic DNS

- no-ip or similar for quick access
- Get a domain name and use a nameserver with a DDNS API, and use a script to update

Personal webpage

If you set up a domain name, consider hosting a webpage with a simple server

Something went wrong!
What to do?

Tools

`ping` and `traceroute` or `dig` are invaluable, as is understanding the output of the `ip` family of commands; those are the same as a desktop, however.
Some tools outside of the normal desktop network toolkit:

tcpdump	Collect logs directly from the firewall, on a given interface
Wireshark	Collect and examine *pcaps*, packet captures. Lets you investigate failures at your leisure, and sift through captures

Classes of failure

Classes of failure

- Physical

Classes of failure

- Physical
- Logical

Classes of failure

- Physical
- Logical
  - Configuration error

Classes of failure

- Physical
- Logical
  - Configuration error
  - Assumption fault

Recovery and fault-tolerance

Failures happen, unfortunately. Power outages strike, hard drives crash, stray voltage kills your SSD's controller…
… and then you need to get into your router.

Backups!

These can come in various flavors:

Backups!

These can come in various flavors:

- Full-system disk images

Backups!

These can come in various flavors:

- Full-system disk images
- Configurations in a private (or public if that's your thing) repo

Backups!

These can come in various flavors:

- Full-system disk images
- Configurations in a private (or public if that's your thing) repo
- Build log with detailed notes

Break glass

Maybe your only laptop with an SSH key for your router on it had a hard drive failure, or you had a device stolen. In this case, if you can't get into your router because you've properly locked it down, you're in a pickle.

Break glass

Maybe your only laptop with an SSH key for your router on it had a hard drive failure, or you had a device stolen. In this case, if you can't get into your router because you've properly locked it down, you're in a pickle.

- IPMI or Serial Console access

Break glass

Maybe your only laptop with an SSH key for your router on it had a hard drive failure, or you had a device stolen. In this case, if you can't get into your router because you've properly locked it down, you're in a pickle.

- IPMI or Serial Console access
- Spare private key

Other ideas

So, what else can we do?

Other ideas

Virtualize it!

Why virtualize?

Why virtualize?

- Reducing risk from compromise

Why virtualize?

- Reducing risk from compromise
- Virtual firewall for virtual machines

Why virtualize?

- Reducing risk from compromise
- Virtual firewall for virtual machines
- Quick update/deployment

How to virtualize?

How to virtualize?

- Pick your OS

How to virtualize?

- Pick your OS
- Connect your VM to the network

How to virtualize?

- Pick your OS
- Connect your VM to the network
  - Linux Bridge interface

How to virtualize?

- Pick your OS
- Connect your VM to the network
  - Linux Bridge interface
  - Promiscuous Mode NIC

How to virtualize?

- Pick your OS
- Connect your VM to the network
  - Linux Bridge interface
  - Promiscuous Mode NIC
  - PCIe Passthrough NIC

Issues you might run into

Issues you might run into

- Drivers and PCIe Passthrough

Issues you might run into

- Drivers and PCIe Passthrough
- Virtual bridged network

Other ideas

Have a firewall behind a firewall

Multiple firewalls

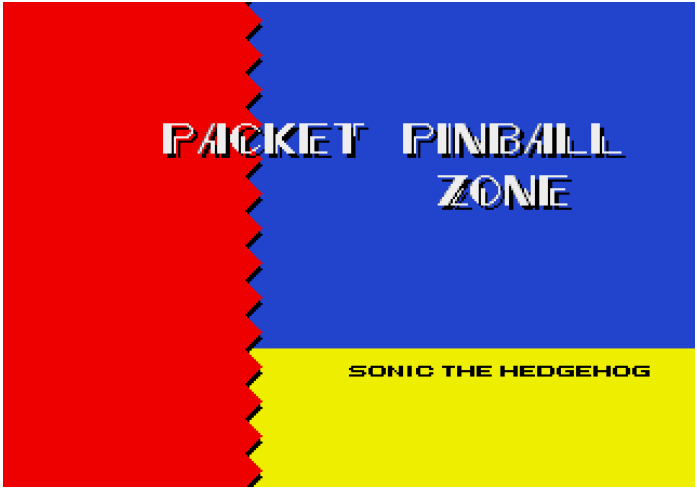
Multiple firewalls

- Keep internet-accessible devices separated from “soft hosts”

Multiple firewalls

- Keep internet-accessible devices separated from “soft hosts”
- DMZ is now outside of a firewall

Tuning for speed



Death Generator by @foone

Jumbo Frames

When looking particularly at speeds above gigabit, you might notice that you're not getting as close to the ripping-fast speeds you expected.

Your throughput might look *okay*, but not stellar.

Jumbo Frames

Ethernet has some fixed overhead per-packet:

- Preamble: 7 octets
- Delimiter: 1 octet
- Destination MAC address: 6 octets
- Source MAC address: 6 octets
- 802.1Q header: 2 octets (Optional!)
- Length: 2 octets
- Frame check sequence (CRC32): 4 octets
- Inter-packet gap: 12 octets

That's 38-40 bytes per frame of overhead, just in layer 2.

Jumbo Frames

But Layer 3 gives us some overhead too.

Jumbo Frames

But Layer 3 gives us some overhead too.

- IPv4 headers: $\geq$20 octets

Jumbo Frames

But Layer 3 gives us some overhead too.

- IPv4 headers: $\geq$20 octets
- IPv6 headers: $\geq$40 octets

Jumbo Frames

- 1500 octet MTU overheads:
  - IPv4: 1480 octet payload, 3.77% overhead
  - IPv6: 1460 octet payload, 5.07% overhead
- 9000 octet MTU overheads:
  - IPv4: 8980 octet payload, 0.64% overhead
  - IPv6: 8960 octet payload, 0.86% overhead

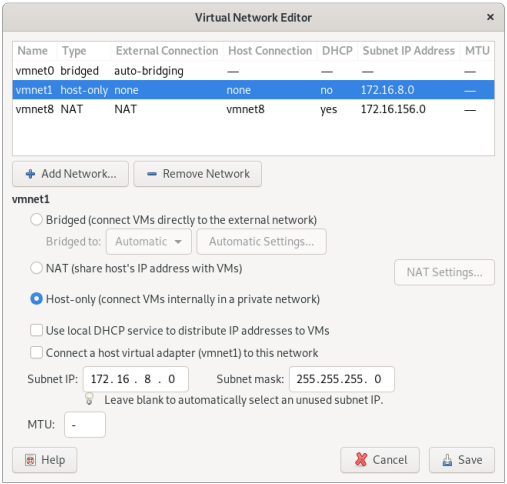
TCP offload

An interesting technology but not widely supported; the primary vendor who's pushing for this tech is Chelsio; they've attempted in the past to get offload support built into the Linux kernel, but were rebuked on the grounds that this moves kernel decisions into a black box; we may yet see some changes in this attitude, but it is generally outside the scope of this talk.

Demo: Build a virtual router

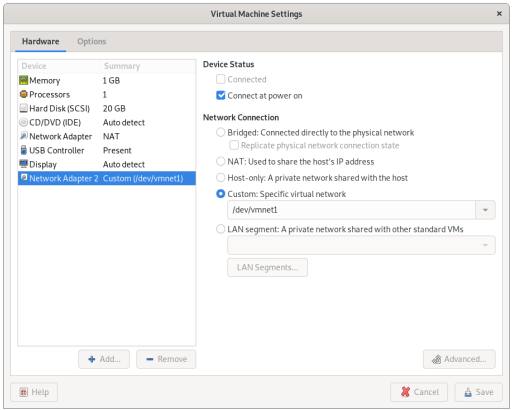
Let’s build a router real quick!
Set up a virtual firewall for other virtual hosts (mayyybe?)
For the purposes of this section, we’ll focus on a minimal firewall setup that covers the basic needs of a firewall, with clear extension points.

Setting up our virtual LAN



In VMware Workstation, you'll want to make sure that you have a host-only network where you've disabled DHCP and removed the host connection; we'll make this the private network managed by our firewall.

Setting up our virtual LAN



We'll also want to make sure to add this new network to our firewall VM. Any VMs created to use this network need to be configured to not have any other network connection.

Setting static IP

Create a static IP connection definition: `nmcli con add ifname ens36 type ethernet ip4 192.168.10.1/24 ipv4.dns 192.168.10.1`

Setting static IP

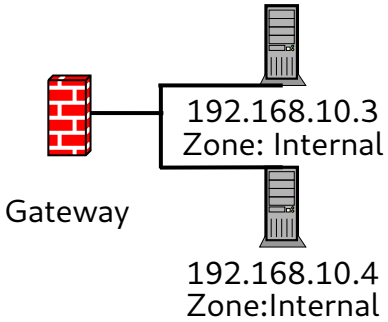
Create a static IP connection definition: `nmcli con add ifname ens36 type ethernet ip4 192.168.10.1/24 ipv4.dns 192.168.10.1`

Then bring up the interface:
`nmcli con up ethernet-ens36`

Firewall rules

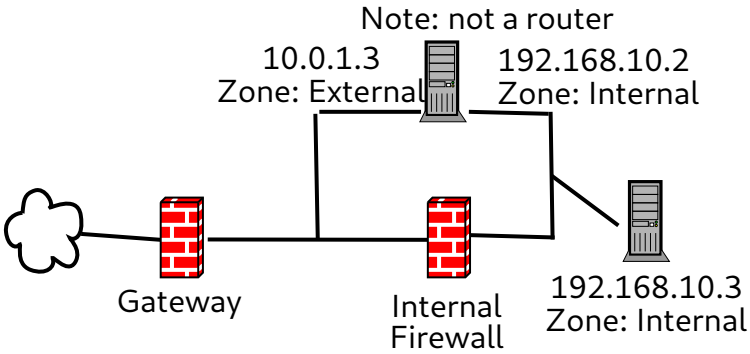
Our firewall configuration should allow SSH in from the internet, and should forward web traffic to a web server at `192.168.10.3`.
For this demo, we'll use `systemd-firewalld`.

firewalld zones



Simple network with only Internal zones.

firewalld zones



Multi-firewall, multi-zone network

`firewalld` zones

The router we're setting up looks a lot more like that second diagram.

- `external`
  - WAN-facing, locked down interface.
  - Forwards traffic with NAT
- `internal`
  - LAN-facing, has a few more ports open.
  - Trusted net.

Open ports for services

```
# Also run these in permanent mode.
firewall-cmd --zone=internal \
  --add-service=dns
firewall-cmd --zone=internal \
  --add-service=dhcp
firewall-cmd --zone=external \
  --add-service=http
firewall-cmd --zone=external \
  --add-service=https
```

Port forwarding for services

```
# This also needs to be run in  permanent mode, but
# the short demo works.
firewall-cmd --zone=external \
  --add-forward-port=port=80: \
      proto=tcp:toport=80: \
      toaddr=192.168.10.2
firewall-cmd --zone=external \
   --add-forward-port=port=443: \
      proto=tcp:toport=443: \
      toaddr=192.168.10.2
```

Now our internal web server can reach the internet!

/proc/sys/net & /etc/sysctl.d

System configuration is exposed through files in `/proc/sys/net/...`, such as:

```
$ cat /proc/sys/net/ipv4/conf/all/log_martians
0
# Enable martian logging
% echo 1 > \
   /proc/sys/net/ipv4/conf/all/log_martians
```

/proc/sys/net & /etc/sysctl.d

System configuration is exposed through files in `/proc/sys/net/...`, such as:

```
$ cat /proc/sys/net/ipv4/conf/all/log_martians
0
# Enable martian logging
% echo 1 > \
   /proc/sys/net/ipv4/conf/all/log_martians
```

These configurations can be configured via `.conf` files in `/etc/sysctl.d`:

/etc/sysctl.d/90-martians.conf

```
net.ipv4.conf.all.log_martians = 1
```

Forwarding traffic

For forwarding, we want `/proc/sys/net/ipv4/conf/all/forwarding` since we're not building a more esoteric network.

Forwarding traffic

For forwarding, we want `/proc/sys/net/ipv4/conf/all/forwarding` since we're not building a more esoteric network.

/etc/sysctl.d/20-ip-config.conf

```
net.ipv4.conf.all.forwarding = 1
```

DHCP and DNS

We're going to use dnsmasq for both our DHCP and our DNS needs!

/etc/dnsmasq.d/10-services.conf

```
# Enable responding on an interface
interface=ens36
domain=test.lan
dhcp-range=192.168.10.100,192.168.10.200,12h

# DNS settings (I like OpenNIC)
server=157.245.161.252
server=206.186.168.3
address=/router.test.lan/192.168.10.1
dhcp-host=anaheim,192.168.10.2
```

Making dnsmasq into a service

Adapted from the *libvirt Networking Handbook*:

/etc/systemd/system/dnsmasq.service

```
[Unit]
Description=DHCP and DNS caching server.
After=network.target
[Service]
ExecStart=/usr/sbin/dnsmasq -k
ExecReload=/bin/kill -HUP $MAINPID
Restart=on-failure
RestartSec=5
[Install]
WantedBy=multi-user.target
```

References

For this talk, and in my personal router-building, I've relied on lots of resources. Here's a few major ones:

- dnsmasq man page, online at `http://www.thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html`
- Kozierok, Charles M. *The TCP/IP Guide.* `https://nostarch.com/tcpip.htm`
- `/proc/sys/net/ipv4` documentation, online at `https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt`

Thanks!

- @benjf5
- @ben_zen@mastodon.social
- ben-zen

I'll open up the repo for this talk at
`https://github.com/ben-zen/scale-18x-router-talk`
The most up-to-date version there will be the LaTeX.