

# Today

Network perimeters

Network threats

VPNs

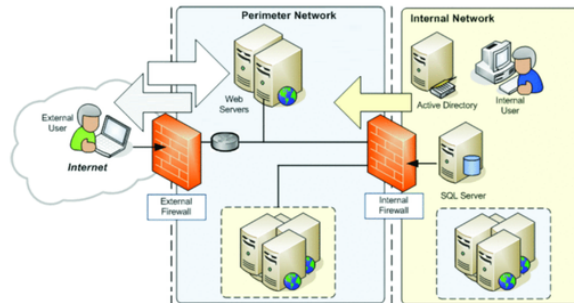
DIY

# Network security (traditional)

What's wrong with this picture?

Perimeter defence:

- "Good" people inside
- "Bad" people outside



Source: "Dgondim" via Wikimedia, CC BY-SA 4.0

3 / 18

People don't like to think about insider threats. We like to think of the people we work with as trustworthy in at least two ways:

1. they **have our interests at heart** and
2. they **are competent to protect our interests**.

Both are, undoubtably, true for some. However, for organizations that involve thousands of people, there will almost certainly be people for whom *neither* is true. Even worse, for most people in any organization, *at most one* of these two aspirations will be accurate, and falling down on either of them is sufficient for an attacker to get an "in".

## Users' view

What should you worry about on:

- public Wi-Fi (airport, coffee shop, etc.)?
- institutional Wi-Fi (corporate, University, etc.)?
- home network (wired or wireless)?

How much should you trust *any* network?

# Network threats

## Confidentiality

- Content data (DPI keywords, cookies, advertisements...)
- Metadata (hostnames, IPs, URLs...)

## Integrity

- Injecting advertisements
- Injecting malware

What's the right solution  
to these problems?

6 / 18

The best solution to these problems isn't to use a "more secure" network, it's to use

\_\_\_\_\_.

# Using untrusted networks

## The best answer: "so what?"

- Advice at a "cybersecurity awareness" seminar: "I wouldn't do online banking at a coffee shop" — with TLS and CT, **why not??**
- The *real* risks come from unprotected, "less important" stuff

7 / 18

Advice like this ("don't do important stuff at a coffee shop") is both misleading and

- It's factually inaccurate: good luck fooling a **browser that supports certificate transparency!**
- If computer security is presented as basically impossible in real-world settings, people won't both trying to keep up.
- If security is presented as something that's only for "important stuff", people will leave themselves vulnerable to the *real* risks.

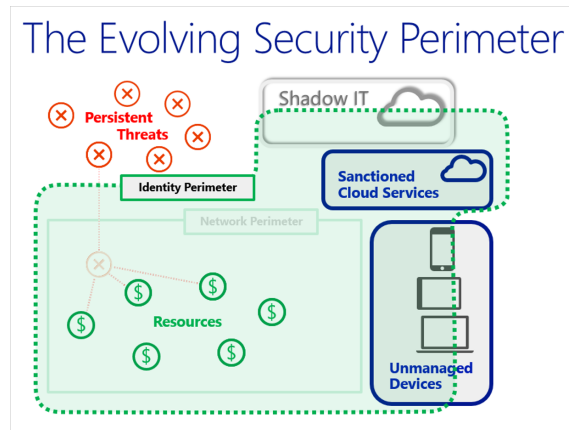
What you *really* need to worry about isn't the sites that use TLS, it's the old systems that use insecure protocols like HTTP, FTP (yes, this is still used, especially in the "creative" industry) and plain-text email. These protocols can allow an attacker to eavesdrop on credentials or other data and modify data in transit to, e.g., add a malware download to a web page.

# Administrator's view

Oooh, the scary cloud?

A more nuanced view:

- *which* perimeter?
- perimeters can increase attacker's work
- assume adversaries are **inside** the perimeter(s)



Source: *Microsoft*

9 / 18

There are \_\_\_\_\_ that we should think about. Let's not think of attackers as simply being "in" or "out"!

Software can be improved by providing least-privileged abstractions, such that malicious (or compromised) code is limited in the damage it can do. Static analysis, fuzzing, type- and memory-safe languages, sandboxing and OS-level access control are all incomplete but helpful layers of \_\_\_\_\_. This is a security strategy analogous to the \_\_\_\_\_ of risk management we've all learned about in the course of the COVID-19 pandemic.

Similarly, networks can be designed to be more sophisticated than a single flat namespace of IPs behind a firewall, where one administrator account can unlock any service on any computer. Every additional step that an attacker has to take to get to their objective increases the work they need to do, the vulnerabilities they need to find, they time they (likely) need to take and \_\_\_\_\_.

As the Dolev-Yao model reminds us, however, increasing the attacker's work factor doesn't mean we can assume that they aren't already inside any given part of a network. Even though we will make the attackers work hard to get through our layers of defence, we should still assume the attacker can strike anywhere.

# Passing through perimeters

Firewall rules

Proxies

Jump hosts

VPNs

DIY VPN

# Firewalls

## Purpose

### Air gap

- no network connectivity at all
- common in critical systems
- other (limited) forms of communication still possible: acoustic, optical, RF, temperature...

11 / 18

We've already seen that firewalls (whether physical or logical) have ports to let some things through. In a computing sense, that's kind of the point: if we didn't want to allow *any* traffic through, we could replace a firewall with a pair of scissors! This is commonly called an "air gap".

Air gaps are commonly employed in critical systems for two reasons:

1. the consequences of an attacker gaining access are severe, partially because
2. the lowest levels of the system are designed with **no security**.

It's worth noting that even air-gapped systems can communicate in very limited and creative ways, including acoustic communication (see [Fansmitter](#)), RF communication from a video display ([AirHopper](#)) or the \_\_\_\_\_ ([AIR-FI](#)), as well as temperature ([Hot or Not \(2006\)](#), [BitWhisper \(2015\)](#)). Sometimes it's a little less creative though no less shocking, with \_\_\_\_\_ in allegedly-air-gapped \_\_\_\_\_ [exploited by state-backed actors](#).



# Proxies

## HTTP

- e.g., <https://qe2a-proxy.mun.ca/Login?url=https://dl.acm.org>
- transparent proxying of HTTP content (including URL rewriting)

## SOCKS

- "please open a connection to 123.456.789.012:12345"
- often run on port 1080

12 / 18

e.g., on campus, I can run: `curl -v --proxy http://tony.engr.mun.ca:8888 http://myshopify.com`

and I'll see that the result has been transparently rewritten, with a new HTTP header added. I could use this functionality to work around network limitations. However, this is also an example of the kind of transparent proxying that someone doing a DNS attack could force on me if I'm using HTTP (not HTTPS)! Adding a new header isn't such a big deal, but I could've added anything, including false information or a drive-by download.

The **SOCKS protocol** can be used to open arbitrary TCP connections and (as of v5) can also forward UDP datagram packets. This can also be very helpful for accessing resources through a firewall, as we'll see next...

# Selective SOCKS proxying

Automatic proxy configuration URL

```
function FindProxyForURL(url, host)
{
  if (dnsDomainIs(host, ".aits.mun.ca"))
  {
    return "SOCKS localhost:1080";
  }
  /* ... */ else
  {
    return "DIRECT";
  }
}
```

13 / 18

Browsers will also let you configure SOCKS proxies to use selectively, so that you don't have to use this performance-affecting mechanism all the time, just for the places where it's important to you.

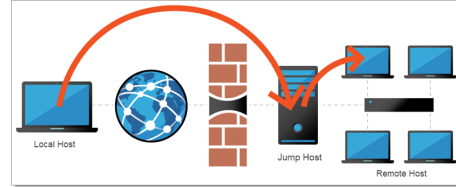
This example (taken from <https://github.com/trombonehero/config/blob/main/.config/mun-proxy.pac>) shows how I connect to a local SOCKS proxy for some Memorial services that aren't available outside of Memorial's network. This is helpful for doing some of my work remotely, and it's one example of a situation in which I don't actually need a VPN.

# Jump hosts

“The firewall won't let me SSH directly into my computer!”

## Establish *jump host* gateway:

- Connect to jump host (e.g., via SSH)
- Connect from jump host to target
- Can allow access logging
- Can also work around non-sensical security policies 😊



Source: *Remote Desktop Manager*

14 / 18

Using an SSH jump host used to be a manual thing: you would SSH into one host and then SSH into another one, possibly forwarding individual TCP ports if you wanted to access a service on a remote host. These days, however, you can automate the process with a **ProxyJump** configuration directive or the `-J` command-line argument:

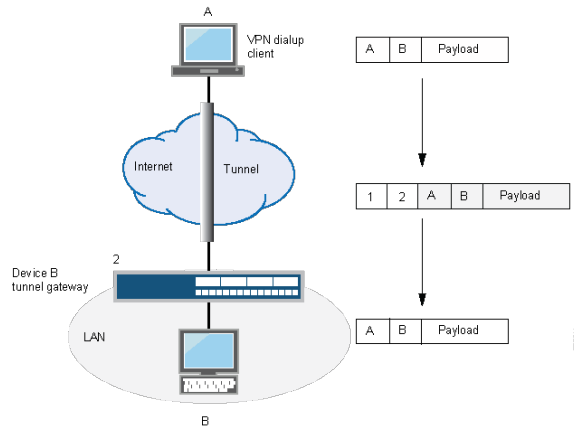
```
SSH(1) BSD General Commands Manual SSH(1)
NAME
  ssh - OpenSSH SSH client (remote login program)
[...]
-J destination
  Connect to the target host by first making a ssh connection to
  the jump host described by destination and then establishing a
  TCP forwarding to the ultimate destination from there. Multiple
  jump hops may be specified separated by comma characters. This
  is a shortcut to specify a ProxyJump configuration directive.
  Note that configuration directives supplied on the command-line
  generally apply to the destination host and not any specified
  jump hosts. Use ~/.ssh/config to specify configuration for jump
  hosts.
```

[...]

# Virtual private network

## How:

- Route all traffic over IPSec *tunnel* to... somewhere
- Packets "pop out" within firewall
- ... including DNS!



Source: *Juniper Networks*

15 / 18

A *virtual private network* is essentially a proxy for \_\_\_\_\_. First you connect to a *VPN concentrator* that's part of your home network (likely in the DMZ) and authenticate to it. Then, you establish a *tunnel* via IPSec, a UDP-based protocol for wrapping up IP packets and passing them to another host for retransmission. Then, every IP packet that you send from your computer isn't sent directly to the local network, but encapsulated and sent to the VPN concentrator, which actually sends the packet. So, from the perspective of firewalls and other services, your packets look like they're coming from the VPN concentrator rather than your computer.

Having packets pop out "within" your home or corporate network allows them to bypass perimeter firewalls and, e.g., access internal services that aren't presented to the outside world. This is helpful for \_\_\_\_\_. However, it is an \_\_\_\_\_. This has the potential downside of allowing (or at least not challenging) the designers and administrators of network services to \_\_\_\_\_, treating a perimeter as \_\_\_\_\_ rather than \_\_\_\_\_.

Many services (and users) would be better served by thinking holistically about remote access requirements and providing least-privileged access as needed instead of exposing "all the things" to remote devices. If you require a VPN to access your corporate email (and only email!), \_\_\_\_\_.

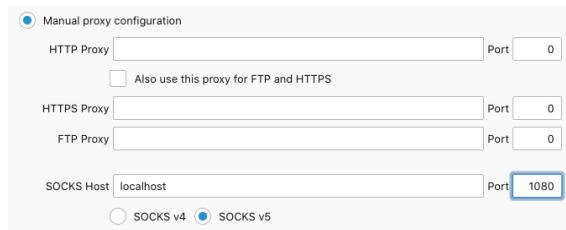
One helpful feature provided by a VPN is making even your DNS queries be served within

# Do you need a VPN?

Establish secure tunnel to jump host w/port forwarding, SOCKS:

```
$ ssh -L 24800:tony.engr.mun.ca:24800 -D 1080 garfield.cs.mun.ca
```

Proxy (some) traffic over it:



The screenshot shows a 'Manual proxy configuration' window. It has several input fields: 'HTTP Proxy' with a 'Port' of 0, 'HTTPS Proxy' with a 'Port' of 0, and 'FTP Proxy' with a 'Port' of 0. There is a checkbox for 'Also use this proxy for FTP and HTTPS'. Below these is a 'SOCKS Host' field set to 'localhost' and a 'Port' field set to '1080'. At the bottom, there are radio buttons for 'SOCKS v4' and 'SOCKS v5', with 'SOCKS v5' selected.

But who will fund  
the podcasters?

16 / 18

Lots of simple use cases don't actually require full VPNs. The secure shell client `ssh` provides for a lot of VPN-like functionality without having to move your entire network interface into a "remote" configuration.

Here, I've shown an example of:

- connecting to a server that Memorial's firewall will let me talk to from anywhere in the world (`garfield.cs.mun.ca`),
- forwarding a single TCP port from a Memorial-internal machine (port 24800 on `tony.engr.mun.ca`, the port for [Synergy](#) keyboard and mouse sharing) and
- running a SOCKS proxy on `localhost:1080` that I can use to connect to other Memorial-internal services as if I were connecting from `garfield.cs.mun.ca`

That's not to say that VPNs are all bad or that you should never use one. I don't use one very often, but I do have [ProtonVPN](#) installed because it provides a free tier for limited usage and the larger company is pretty privacy-focused. Most of the time, however, TLS, SOCKS and SSH more than meet my needs.

# Summary

Network perimeters

Network threats

VPNs

DIY