Network Security

Firewall of Udûn: You shall not pass

Gwendal Patat
Univ Rennes, CNRS, IRISA
2025/2026

Firewall Overview

Firewalls: Motivation

One of the major feature of Internet is to allow **end-to-end connections**.

- Once you are connected to the network, you can reach anyone, you can become a server, reachable by everyone, without asking anybody for permission.

But it is also means that everyone can connect everywhere:

- even on your own private services,
- even your desktop computers,
- or smart devices…

End-to-end connectivity does not means it is mandatory to accept everything from everybody. You may want to filter things.

Firewalls: Motivation

Firewalls are one of the basic building blocks of network security.

They are used to:

- Control which network traffic is allowed to enter or leave a network.
- Separate zones with different levels of trust (e.g., Internet, internal network).
- Enforce security policies by applying rules to network flows.
- Reduce the attack surface by limiting unnecessary or unwanted communication.
- Monitor and log traffic for auditing and incident analysis.

Firewall Rules

Firewall rules are expressed using conditions and apply policy

- If (condition) { Policy }
- Condition: does packet field X match or not
- Policy:
  - Allow.
  - Deny.
    - Silently: just drop the packet
    - Explicitly: issue ICMP packet
  - Other policies: log, slow down, etc.

Firewall Filtering

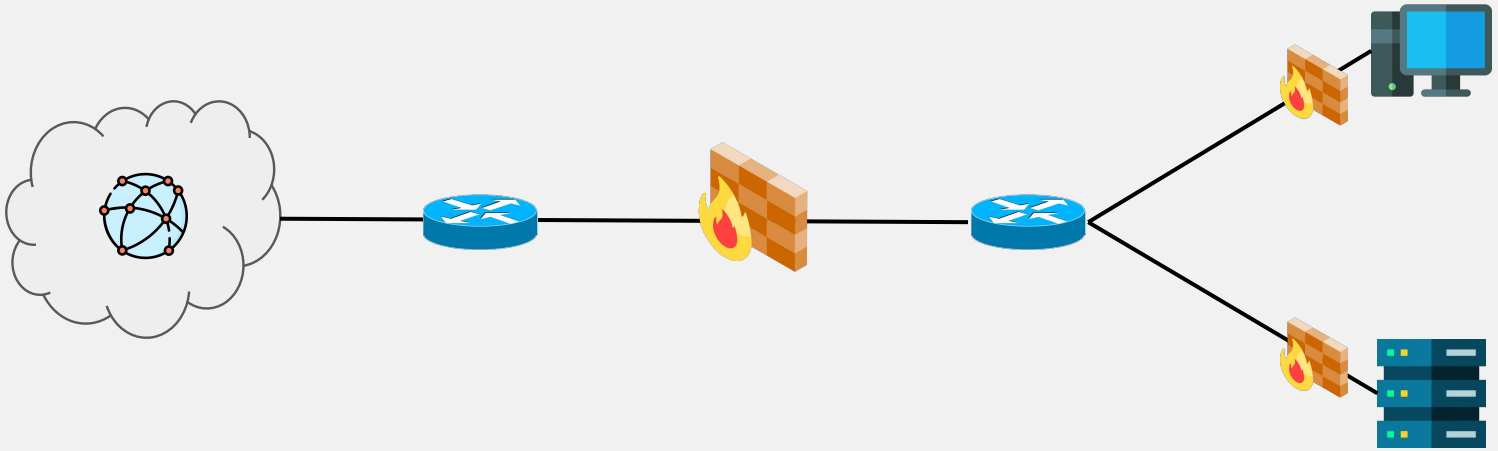
The filtering can be done on all layer of the TCP/IP model:

- Layer 1: Interfaces
  - If an interface should not be used: disallow it
- Layer 2: Ethernet field
- Layer 3: IP fields: Source IP, destination IP
- Layer 4: TCP/UDP fields: Source Port, destination port, TCP fields
- Layer 5: application information:
  - E.g., Blocking domains, URLs, etc.

Firewall Implementations

- **Network-based firewall:**
  - Implements Access Control Lists (ACLs)
  - Filters traffic at network layer (L3).
  - Used to control inter-network flows (e.g., LAN ↔ Internet).
  - *Dedicated hardware*:
    - Specialized device on the network whose sole purpose is security.
- **Host-based firewall:**
  - Runs on the machine itself.
  - Filters incoming and outgoing traffic per host.
  - Can be stateful.
  - Often complements router ACLs for defense in depth.

Host-based vs Network-based Firewalls

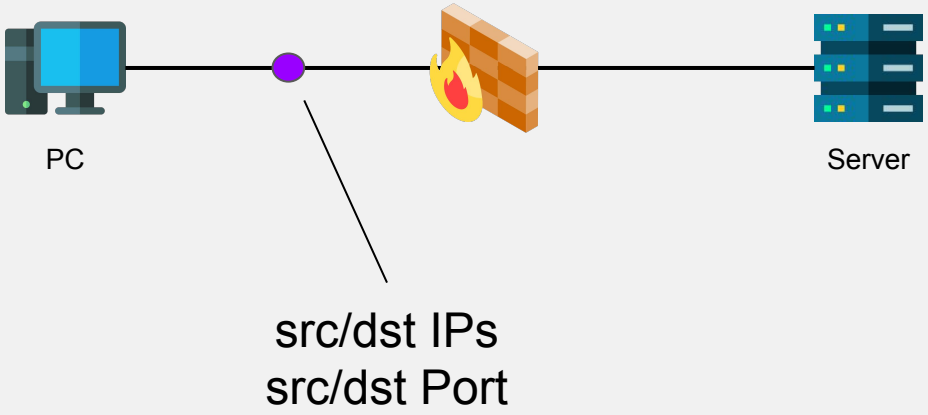


Filtering Techniques

- **Packet Filtering**
  - Filters traffic based on information in the packet headers, such as source/destination IP address, protocol, and port numbers.
  - Works at the network and transport layers (Layer 3/4).
  - Example: Cisco standard / extended ACLs used on routers to allow or deny packets based on IP and port.
- **Stateful Filtering**
  - Tracks the state of connections (e.g., a TCP handshake) to understand whether a packet belongs to an existing, valid flow.
  - Decisions consider the context of the communication, not only the packet itself.
- **Application Filtering**
  - Inspects and makes decisions based on application-level data, such as HTTP requests, domain names, or protocol commands.
  - Works beyond ports and IPs, focusing on how the application is actually being used.

Stateless vs Stateful

Stateless Firewall

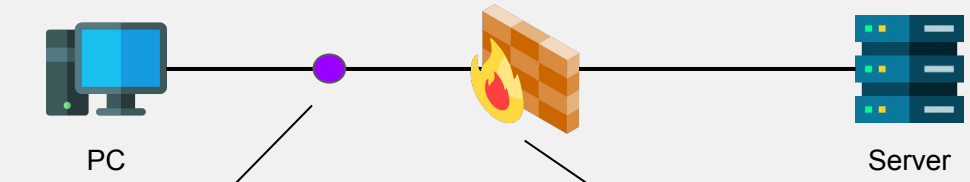


Stateless Firewall

Stateless Filtering:

- All packets are inspected by the firewall as an independent unit.
- The context with which rules will be applied is self contained in the packet with: IPs, protocol, port numbers, etc.
- **Advantages**:
  - Few resources needed.
- **Disadvantages**:
  - No awareness of previous communication.

Stateful Firewall



src/dst IPs
src/dst Port

Has knowledge of previously exchange packets

Stateful Firewall

Stateful Filtering: Packets are inspected as part of a communication, not as isolated units.

- The firewall keeps a state table that records active sessions (e.g., TCP handshake, flow identifiers).
- Decisions are made based on whether the packet belongs to a known and valid connection.
- **Advantages**:
  - Automatically allows return traffic for established connections.
  - Policies can be defined at the flow level rather than packet-by-packet.
- **Disadvantages**:
  - Requires more memory and processing to maintain state.
  - Can be targeted by state exhaustion attacks (e.g., many half-open connections).

Stateful Firewall and NAT

Stateful firewall has a control flow such as:

```
if reply_of_a_request:
        accept;
otherwise drop;
```

Kind of look like what a NAT would do with unsolicited traffic, and people often mistake NAT and stateful firewalling for this reason.

But remember:

- Filtering on NAT is **only a side effect**, the idea was to share IPs.
- With NAT with only one IP: only one service on a given port (i.e., 80/tcp)
- With stateful firewalling, we can allow every host to have their own HTTP service:
  - if reply_of_a_request, accept;
  - if to port 80, accept
  - otherwise drop
- You can’t express that with NAT

Firewalls in Practice

Firewalling in practice

Most operating systems, whether desktop, server, or router systems, include built-in firewall capabilities:

- **xBSD**: packet filter (pf), and sometimes other implementations like ipfw or ipfilter.
- **Linux**: previously ipchains, then netfilter/iptables, and now nftables (nft) intended as its replacement.
- **Windows and macOS**: native firewall solutions provided in the system.

These are relatively low-level. Higher-level tools exist:

- Command-line tools: ufw on Ubuntu, shorewall, ferm.
- GUI or web-based frontends: fwbuilder, firewalld, gufw, etc.

Commercial vendors also offer dedicated firewall hardware appliances, often running Linux or BSD internally.

In many setups, NAT configuration is handled through the firewall as well, since both functions are closely connected.

Firewalling in Linux

Linux’s firewall implementation, netfilter, exposed through iptables or nft, allows to have hooks in various steps of a packet flow, based on:

- **Tables**: identifying which kind of action we want to perform (e.g., filter, NAT)
- **Chains**: identifying at which step of the packet flow we are: Input packets, Output, Forward (if your host act as a router), user-defined ones, etc.

iptables Tables

- **Filter:** Packet filtering.
- **NAT:** NAT rules to apply.
- **Mangle:** IP header modification (e.g., TTL modification)
- **Raw:** Connection tracking, pkt sequence.
- **Security:** SElinux tag.

iptables Chains

- **Pre Routing**: Apply on any incoming packets
- **Input**: Apply when entering the system
- **Forward**: Apply when the packet is routed through the system
- **Output**: Apply when the packet is going out of the system.
- **Post Routing**: Apply after routing decision and sending on the wire

Chain order

Chain are evaluated in a transversal order based on the traffic source and destination, for instance:

- **Incoming packets for the local system**: PREROUTING -> INPUT
- **Incoming packets to be routed**: PREROUTING -> FORWARD -> POSTROUTING
- **Sent by the local system**: OUTPUT -> POSTROUTING

Rules

Rules with iptables are to be defined within a chain an assigned with a matching component and a target (action).

- **Matching Components**:
  - IPs
  - Protocols
  - Ports
  - etc…
- **Targets**:
  - Terminating: accept, drop, queue, etc.
  - Non-Terminating: continue.

iptables Syntax

List your rules:

```
iptables -L -v [-t
filter]
```

Appending a rule at the end of the chain:

```
iptables -A INPUT --src youtube.com -j DROP
iptables -A INPUT --src youtube.com -j REJECT // Q: What diff with drop?
iptables -A INPUT -p tcp -m tcp --sport 443 -j ACCEPT
```

Deleting a rule:

```
iptables -D INPUT --src youtube.com -j DROP
```

Using states:

```
iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -j DROP
// Q: What does these two lines do?
```

iptables vs nftables

Nftables:

- No predefined chains.
- A single rule can take multiple actions.
- Hooks are now the default chains.

nft Syntax

- Tables and chains are still here.
- Can be configured using:
  - the terminal.
  - .nft file to define tables, chains, and rules.

```
nft add table inet [table name]
nft add chain inet [table name] [chain name] {
   type filter hook input priority 0 ; policy accept ;
}
```

Common Policies

Most common policy:

- Allow everything output
- Deny everything input
- except responses to requests
- except allowed services (ie http on a web server)

You could also filter output, for example on servers:

- Allow only DNS 53/UDP and TCP.
- Allow connections in progress.
- Allow ICMP.

Application Firewall

Application Firewall

Most firewalls only act on lower level layers:

- Transport, Network, Data Link

Generally, application layer filtering is delegated to the application process itself.

- A Web server or proxy can filter which set of request it accept or forward
- A DNS server also
- Some traffic can be intercepted and redirected on the fly (HTTP, DNS)
- Some require to be explicitly redirected (HTTPS/TLS)

Commercial solutions may offer it, but it involves to parse application protocol (and re-order/reassemble packets).

Web Application Firewall

A **Web Application Firewall (WAF)** is an example of an Application firewall:

- It analyzes HTTP/HTTPS requests to detect suspicious or malicious patterns.
- Unlike traditional firewalls, it understands application semantics (URLs, headers, cookies, parameters).

Common goals:

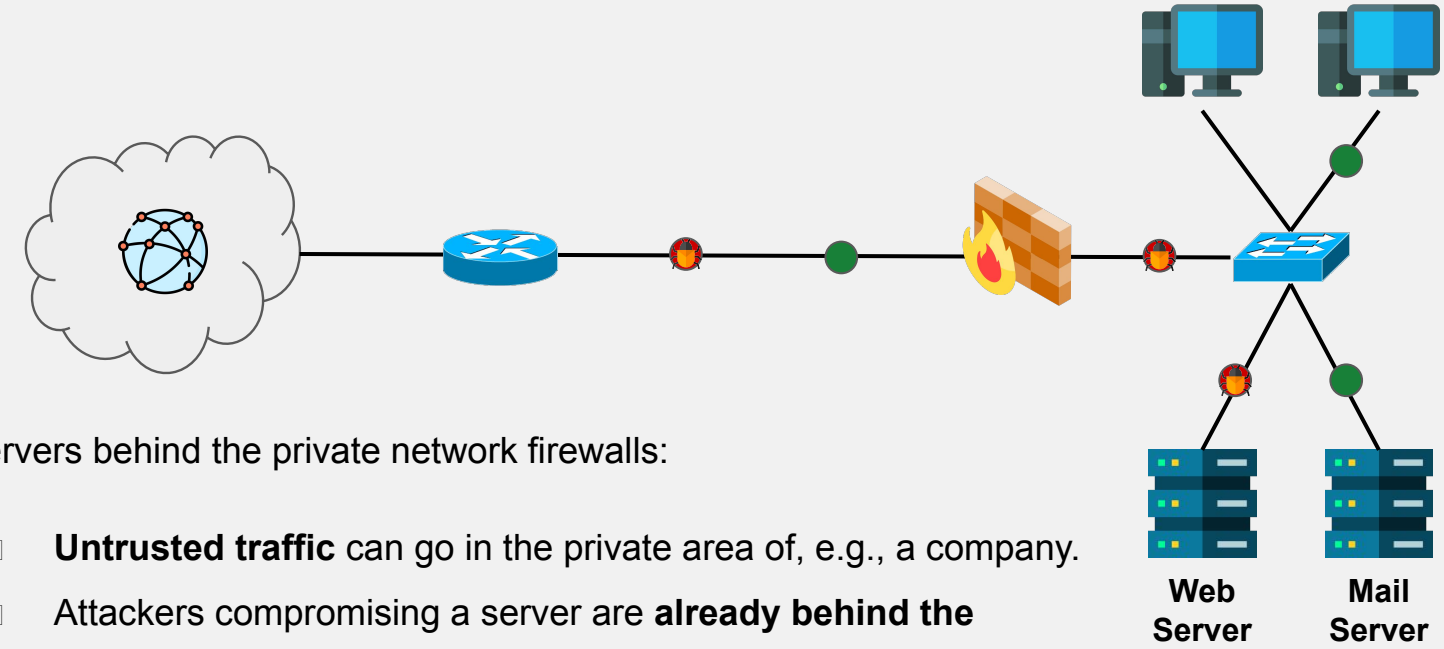
- Prevent SQL injection, XSS, command injection, and similar attacks.
- Enforce input validation rules.
- Apply rate limiting and bot detection.
- Cloud-based filtering services.
- etc.

DMZ

Demilitarized Zone

A **DMZ (Demilitarized Zone)** is a network configuration used to improve the security of a private network by segregating exposed devices from the private hosts.

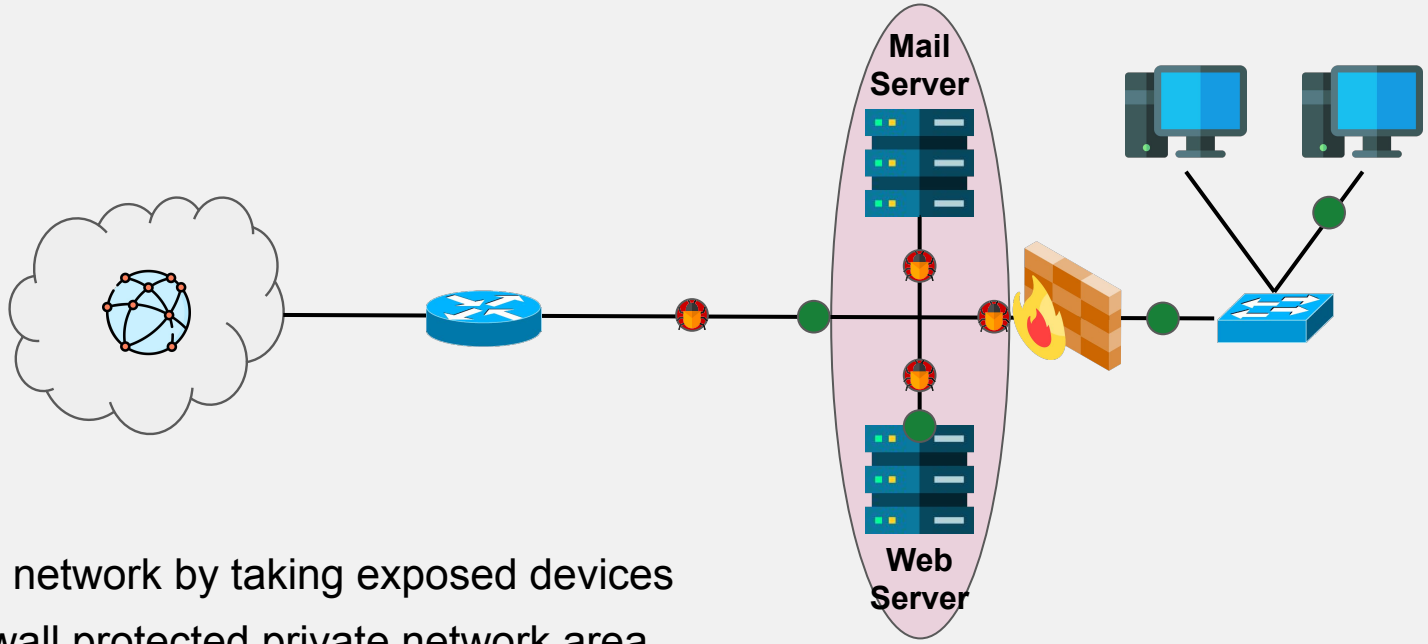
DMZ



Servers behind the private network firewalls:

- **Untrusted traffic** can go in the private area of, e.g., a company.
- Attackers compromising a server are **already behind the firewall**.

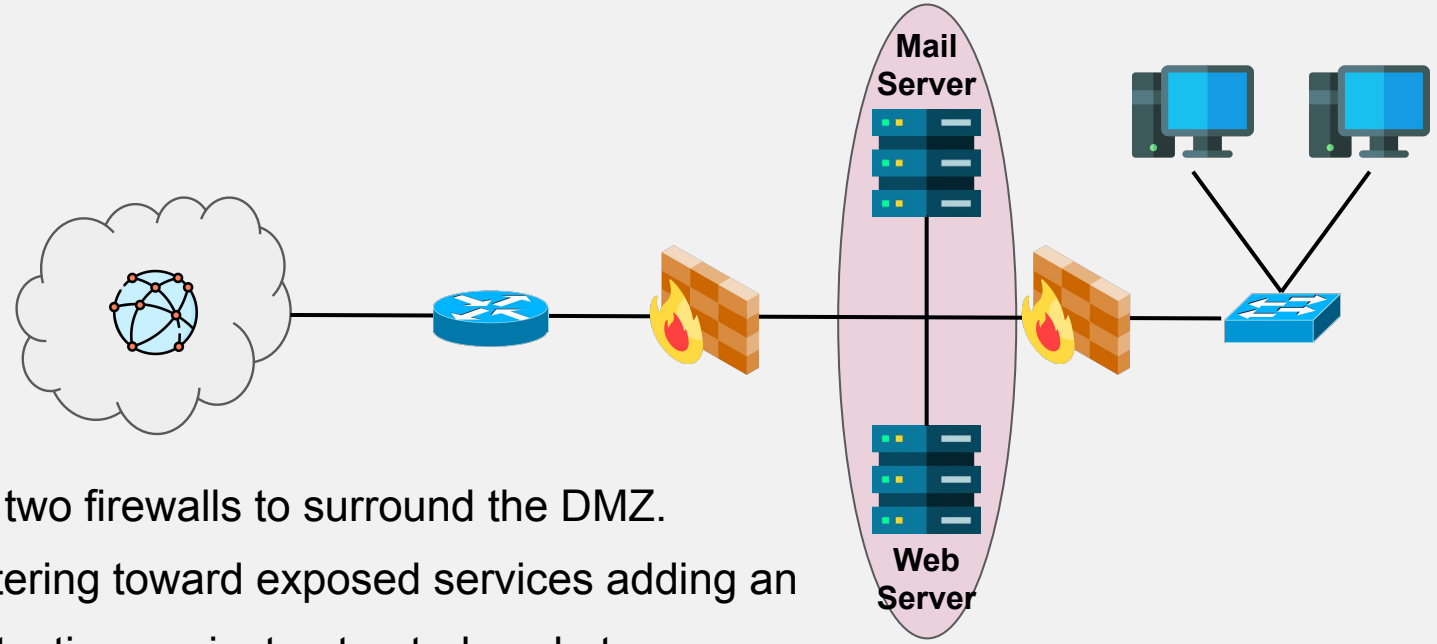
DMZ



DMZ: Dividing the network by taking exposed devices outside of the firewall protected private network area.

- Devices are now fully exposed to the untrusted traffic but separated from other resources.

DMZ



DMZ: Best using two firewalls to surround the DMZ. Allowing traffic filtering toward exposed services adding an extra layer of protection against untrusted packets.

- An attacker would now need to bypass two firewalls before reaching the private network.

Firewall Bypass

Bypass Techniques 1/2

Common techniques attackers use to circumvent firewall controls:

- **Tunnelling through allowed protocols**
  - Encapsulate forbidden traffic inside an allowed protocol (HTTP(S), DNS, ICMP). Example: DNS tunnelling encodes payloads in DNS queries/responses; HTTP(S) tunnels carry arbitrary data inside requests/POST bodies.
- **Using proxies and VPNs**
  - Connect to a remote proxy or VPN endpoint that the firewall permits (or that is reachable via an allowed outbound flow). All subsequent traffic is then relayed through that remote endpoint.
- **Application-layer evasion**
  - Use application semantics to appear legitimate (e.g., crafting HTTP requests that look like normal web traffic, using well-formed TLS handshakes). When the firewall only filters by ports or IPs, application-layer payloads pass unchecked.
- **Port-hopping and dynamic ports**
  - Move services to ports that are permitted by policy (e.g., run a reverse shell over TCP/443) or use randomized ports to avoid static rules.

Bypass Techniques 2/2

- **Protocol obfuscation / encapsulation**
  - Wrap traffic in another protocol (e.g., wrap a custom protocol inside WebSockets, HTTP/2, or QUIC) so traditional L3/L4 filters cannot detect it.
- **Abuse of allowed services**
  - Exploit services that are intentionally exposed (web servers, mail relays, cloud APIs) as relays to reach internal resources or exfiltrate data.
- **IP spoofing and source routing**
- **Fragmentation and packet crafting**
  - Split payloads across fragments or craft unusual packet sequences.
- **Compromised internal hosts / lateral movement**
  - Bypass perimeter controls entirely by compromising a host inside the protected network (insider or malware) and using its legitimate network access.

Why does it work? 1/2

- **Assumptions on protocol semantics**
  - Many firewalls make decisions based on ports, IPs, or simple header fields (e.g., HTTPS on non-standard ports).
- **Default outbound-allow policies**
  - Typical deployments allow most outbound connections and only restrict inbound flows. This permits an attacker inside the network to initiate tunnels or VPNs outward.
- **Insufficient application-layer inspection**
  - Deep parsing of application protocols is expensive and error-prone.
- **Encrypted traffic**
  - TLS encryption prevents inspection unless TLS interception is performed.

Why does it work? 2/2

- **Complexity and misconfiguration**
  - Large rule sets, exceptions, and legacy configurations create gaps: overlapping rules, or unintended allow entries.
- **Statefulness and resource limits**
  - Stateful firewalls can be evaded if stateful tracking is bypassed (e.g., asymmetric routing) or if state tables are exhausted.
- **Use of legitimate third-party services.**

Mitigations 1/2

- **Least-privilege, explicit allow-lists**
  - Prefer deny-by-default with narrow allow-lists for both inbound and outbound traffic. Specify allowed hosts, protocols and ports rather than broadly permitting ranges.
- **Egress filtering**
  - Apply strict outbound filtering: only allow necessary destinations and protocols.
- **Block or monitor suspicious channels**
  - Restrict direct outbound VPN and proxy protocols where possible.

Mitigations 2/2

- **Deploy application-aware controls**
  - Use proxies, reverse proxies, or WAFs for application filtering.
- **TLS inspection**
  - Explicit proxy or TLS termination.
    - Not very free speechy…
- **Network segmentation and DMZs**
  - Isolate exposed services in a DMZ.
- **Behavioral detection and IDS/IPS**
  - Complement filtering with anomaly detection, flow analysis, and signatures for known tunnelling techniques.
  - More on this in another lecture.

Firewall Summary

By default Internet end-to-end design could allow anyone to connect everywhere:

- But there are services we might want to keep private

Firewalls allow to filter traffic:

- On L1/L2 (Interface, Mac addr), L3 (IP addresses), L4 (Ports, TCP info), sometimes upper layer with applications (harder, more resource intensive and bug prone).
- When sending, receiving, forwarding.
- On Hosts and/or Routers.

Can be stateless or stateful:

- Stateless: fast, only matching on packets fields.
- Stateful: maintain states, more ressources intensive but allow rules such as accept responses to requests; else drop

Resources and Acknowledgements

- Computer Networking: A Top-down Approach by James F. Kurose, Keith W. Ross
- Internet Security: A Hands-on Approach, 3rd Edition, Du Wenliang
- External materials from Mathieu Goessens.