Securing private networks

Tomáš Pevný

October 20, 2022

Plan

Networking 101

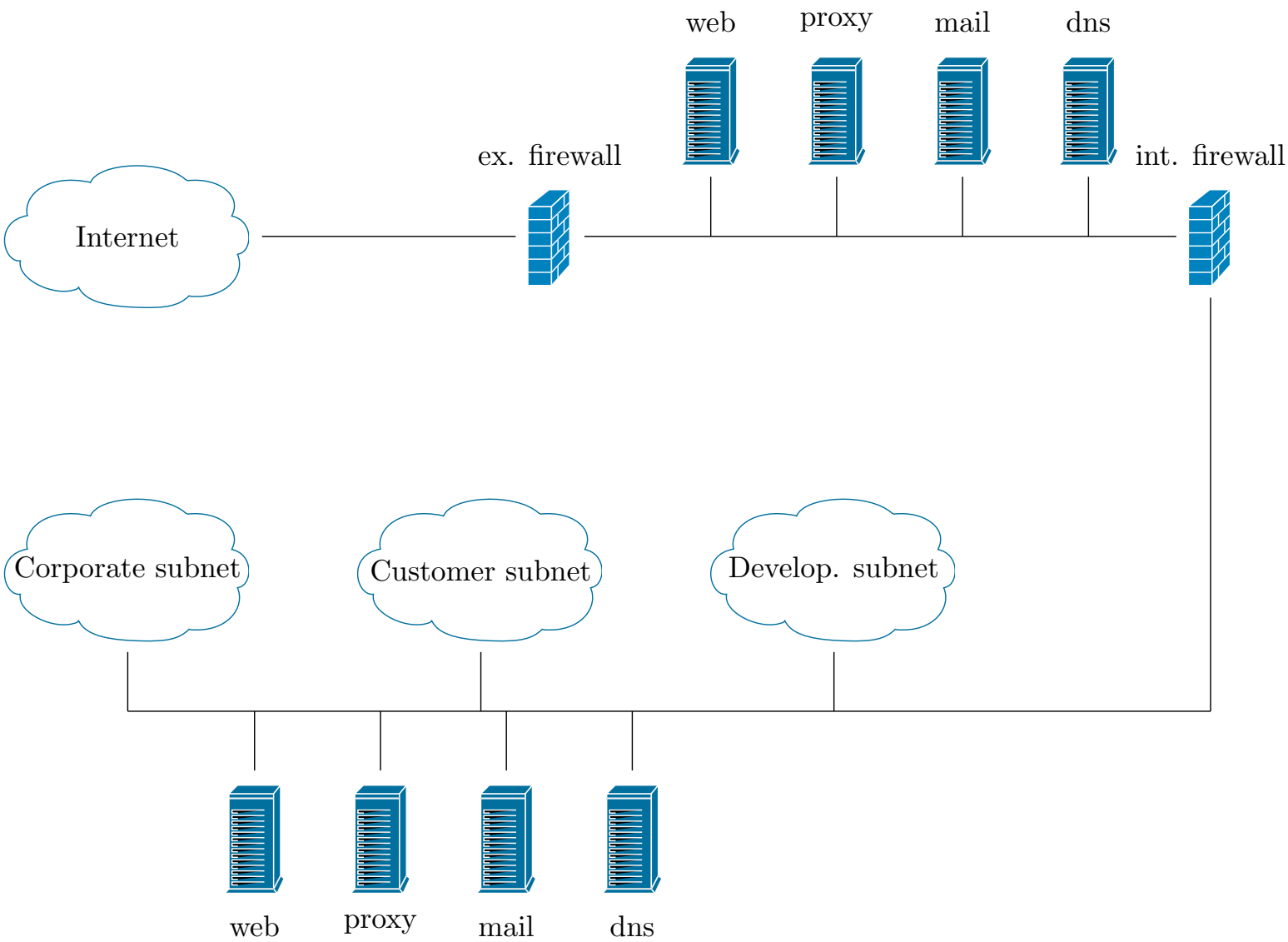
Firewalls

Intrusion detection

Motivation

visualization

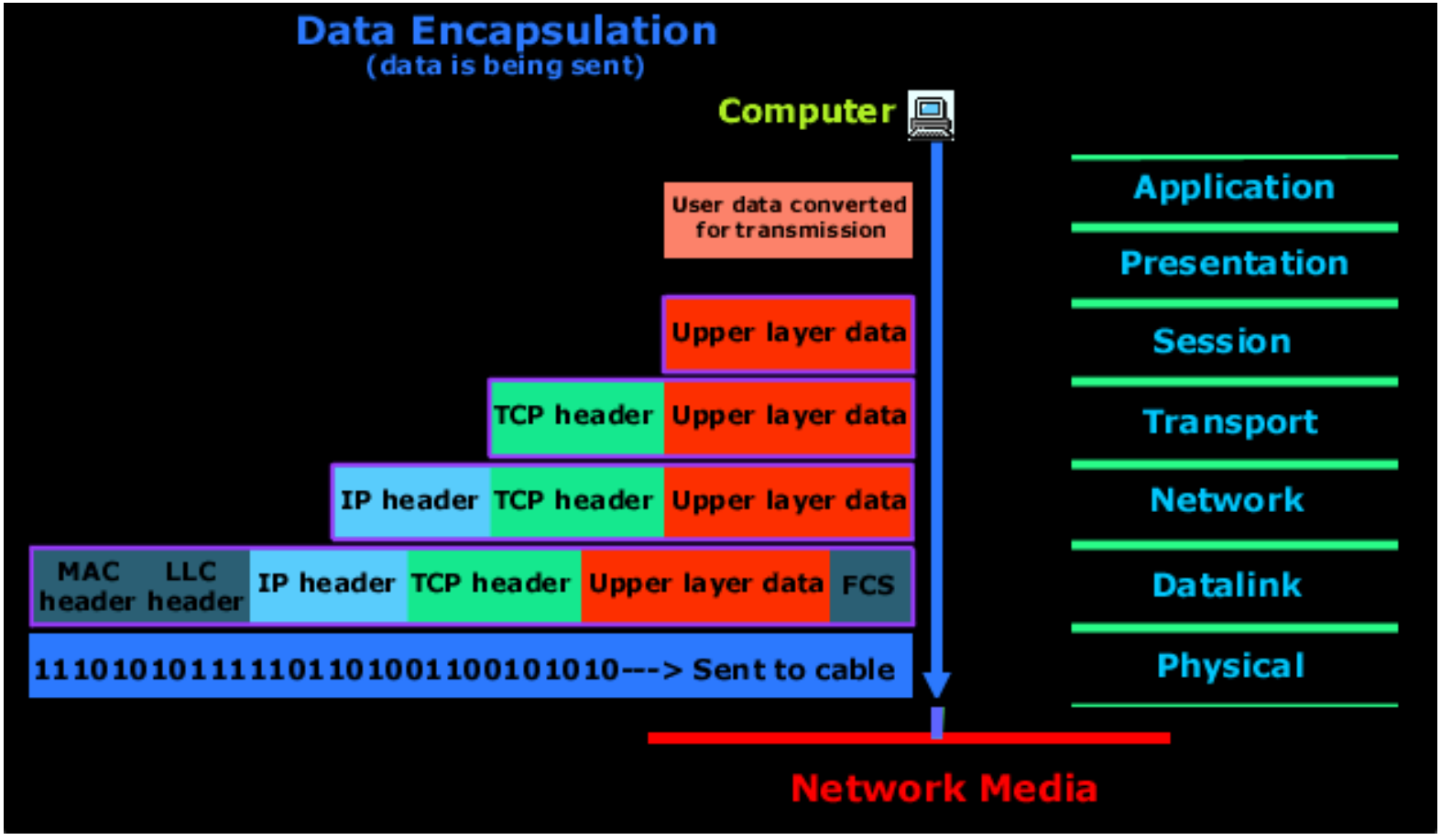
Design a policy



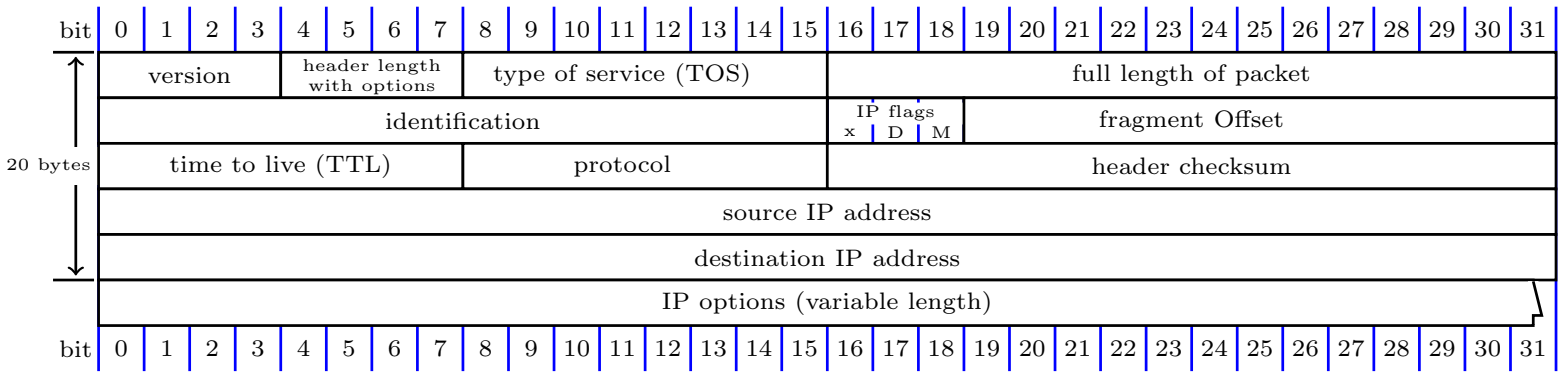
What are strategies to protect

- Make the attack impossible
- Log for later audit
- Detect
- Deter

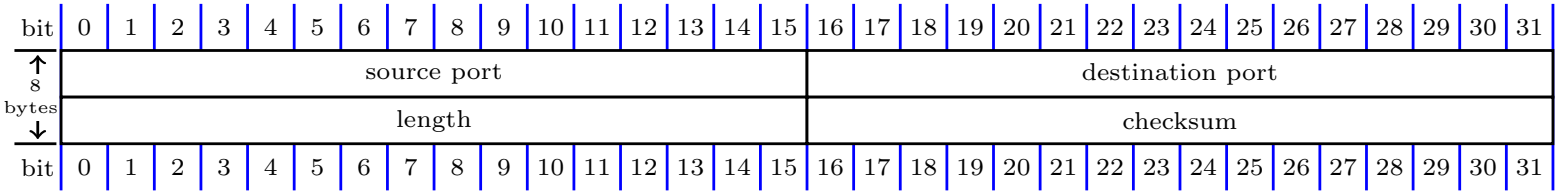
OSI model and packets structure



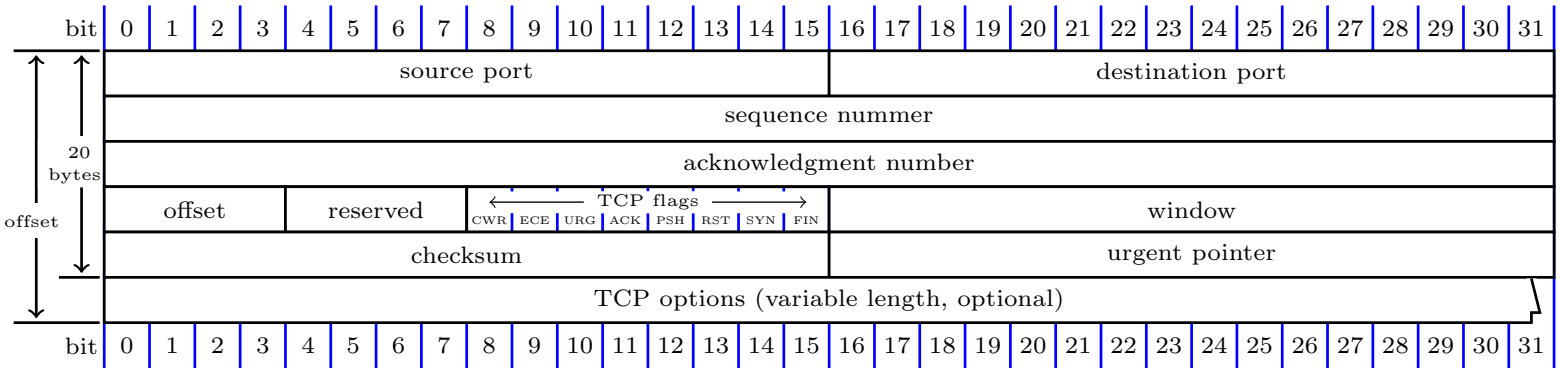
IPv4 packet header



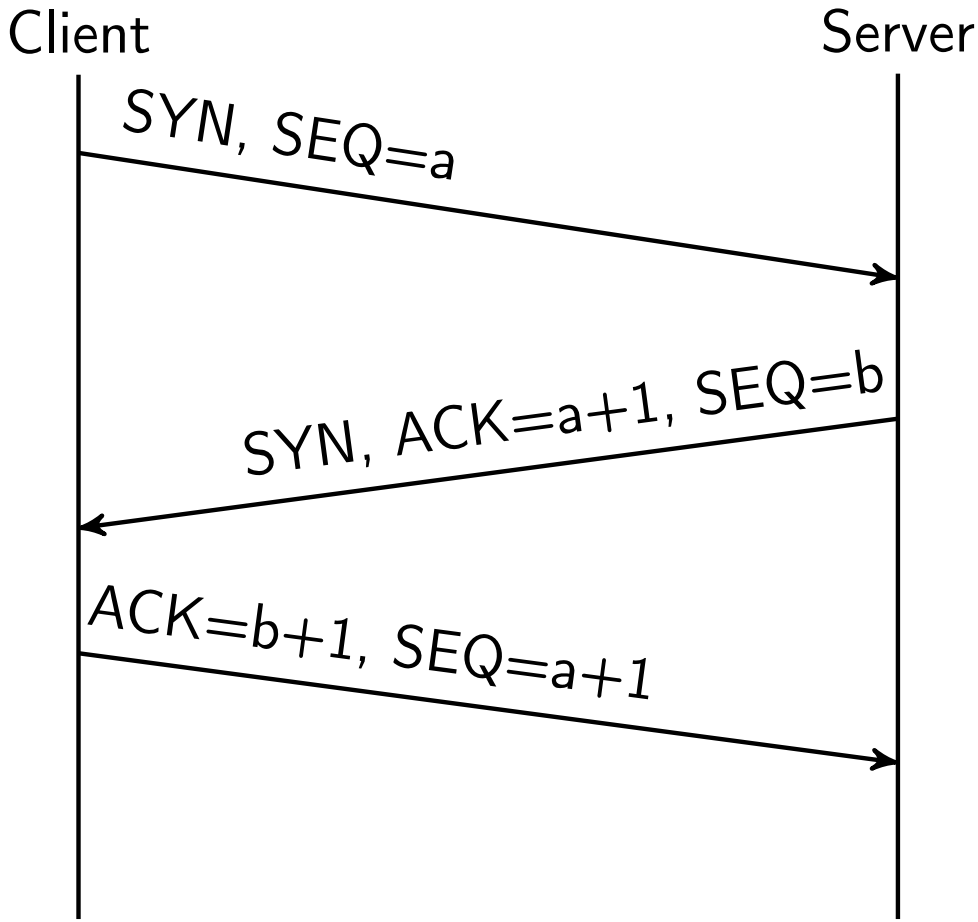
UDP packet header



TCP packet header



TCP handshake



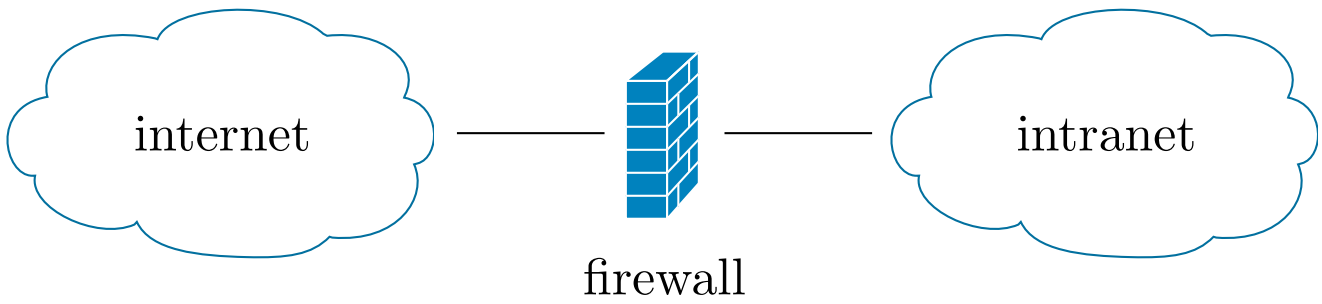
Plan

Networking 101

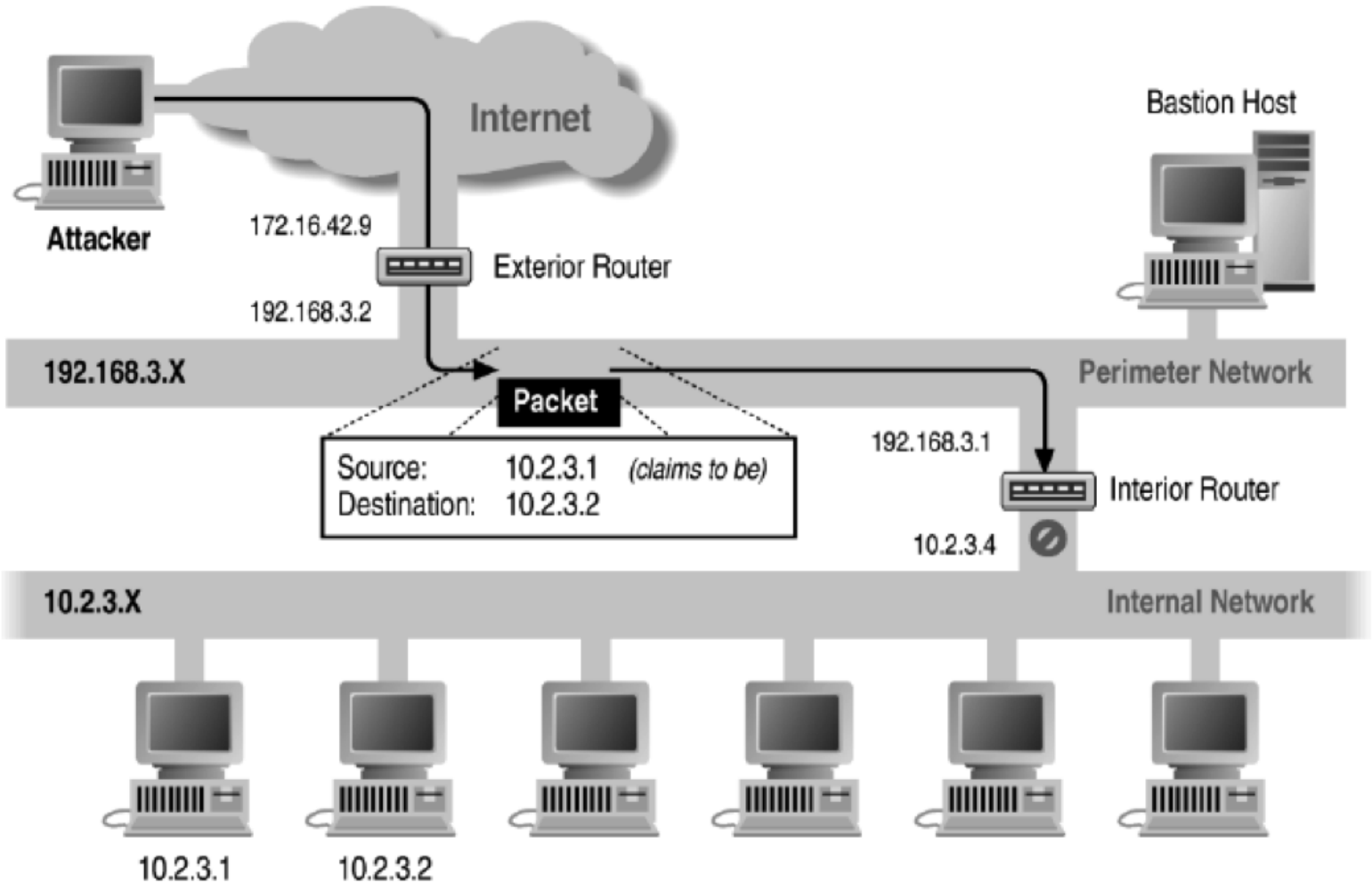
Firewalls

Intrusion detection

Which policies firewall can enforce?



Source address forgery



Example: prevent connection to int. hosts

Stateless firewall:

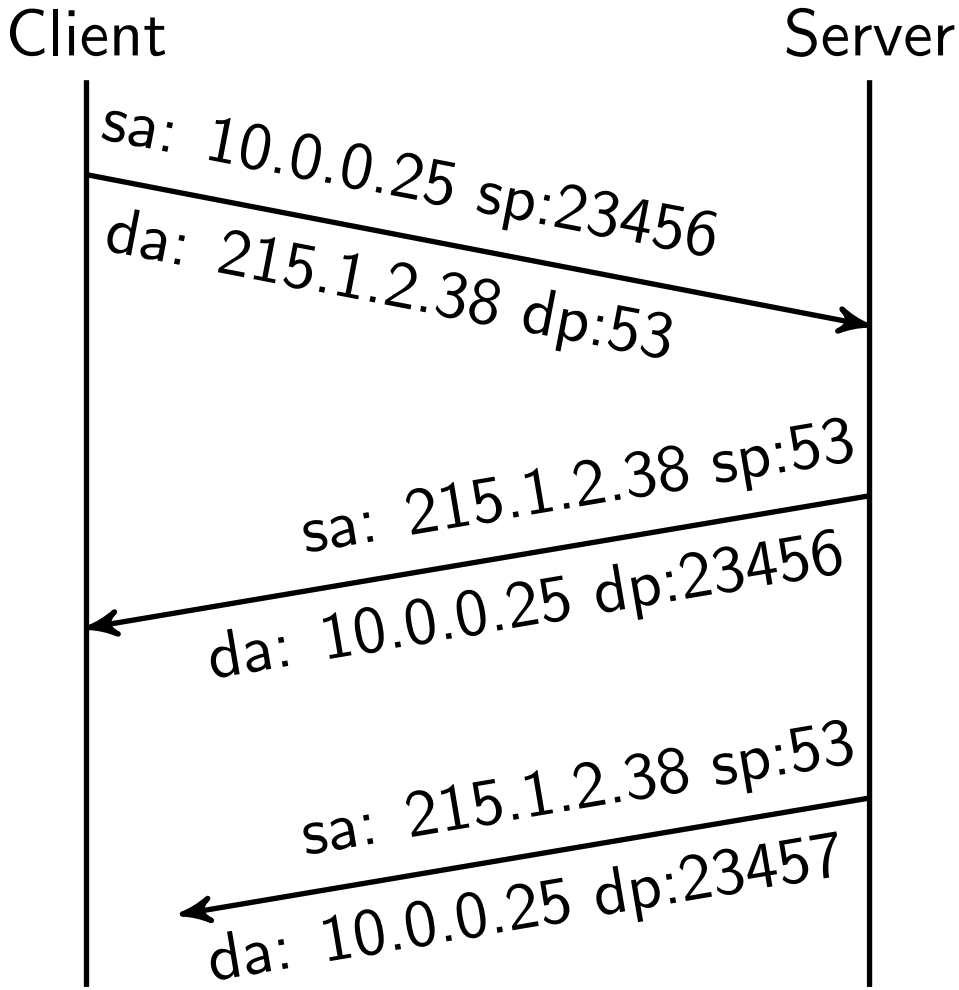
```
allow tcp *:*/out -> 1.2.3.4:25/in
allow tcp *:*/in  -> *:*/out
allow tcp *:*/out -> *:*/in (if ACK bit set)
drop   *  *:*     -> *:*
```

Example: prevent connection to int. hosts

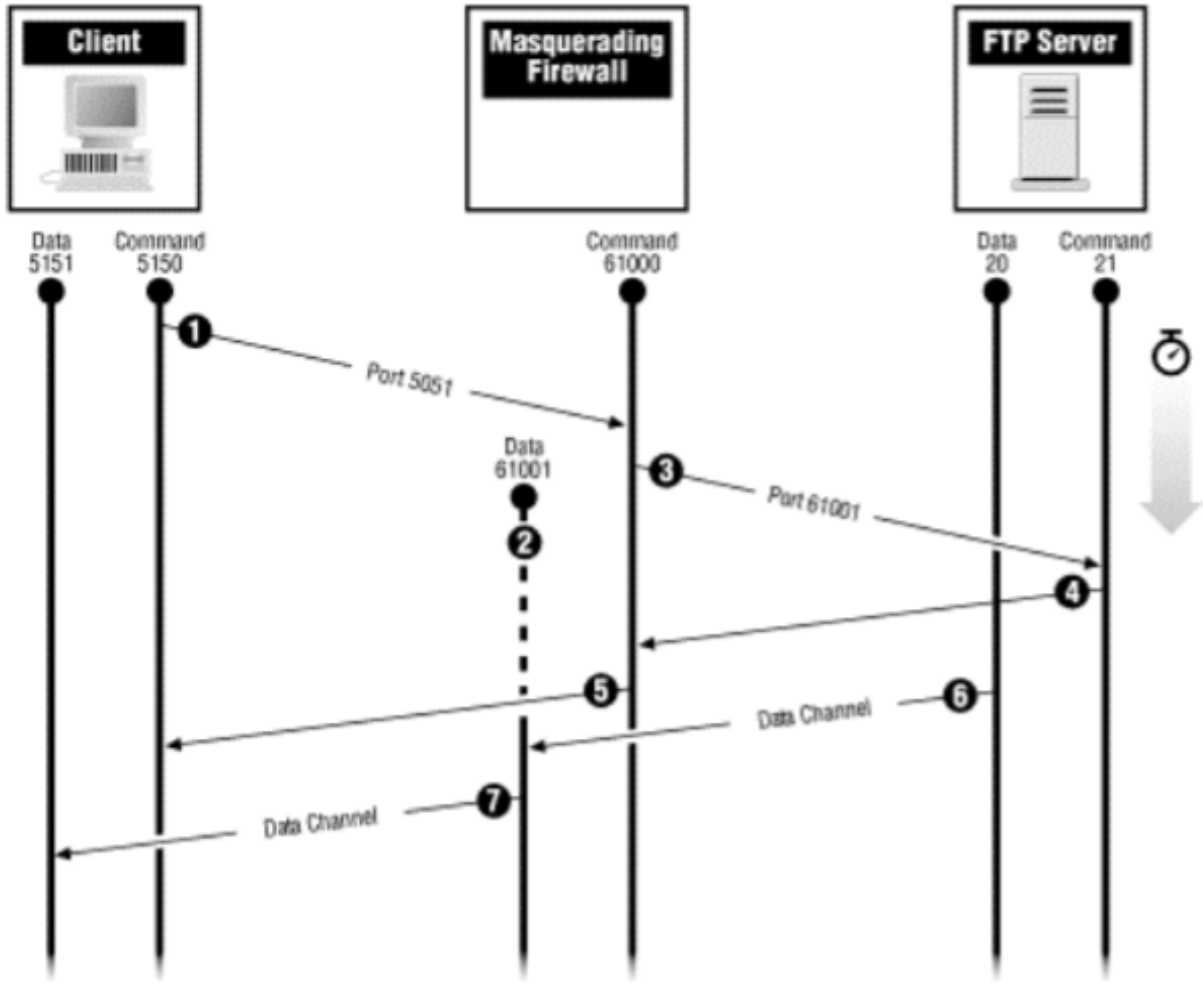
Statefull firewall:

```
allow tcp connection *:*/in  -> *:*/out
allow tcp connection *:*/out -> 1.2.3.4:25/in
drop   *  *:*     -> *:*
```

Dynamic UDP filtering



Firewalls can break the protocols

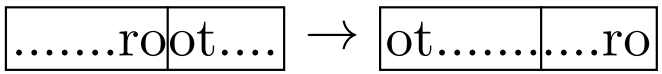


Packet fragmentation attack

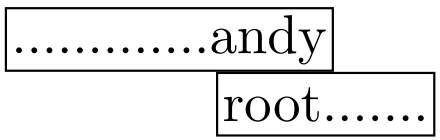
- Firewall configuration
  - TCP port 23 is blocked but SMTP port 25 is allowed
- First packet
  - Fragmentation Offset = 0.
  - DF bit = 0 : "May Fragment"
  - MF bit = 1 : "More Fragments"
  - Destination Port = 25. TCP port 25 is allowed, so firewall allows packet
- Second packet
  - Fragmentation Offset = 1: second packet overwrites all but first 8 bits of the first packet
  - DF bit = 0 : "May Fragment"
  - MF bit = 0 : "Last Fragment."
  - Destination Port = 23. Normally be blocked, but sneaks by!
- What happens
  - Firewall ignores second packet "TCP header" because it is fragment of first
  - At host, packet reassembled and received at port 23

Issues of application-level firewalls

- Packet reordering

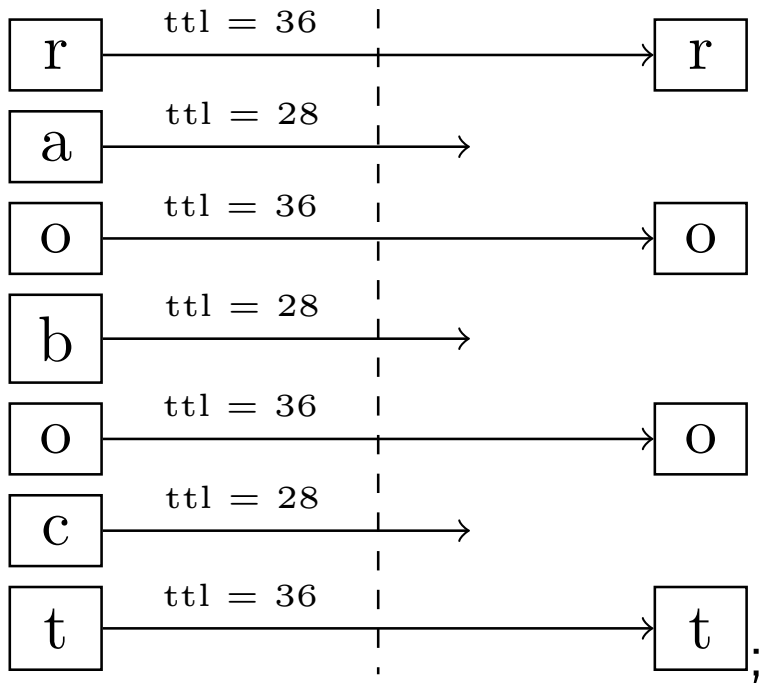


- Packet fragmentation



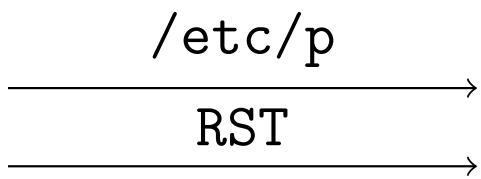
- TTL:
  send `andy` with `ttl` 26 and `root` with `ttl` 32.
- Discrepancy in protocol understanding.

Issue: TTL incosistency



Issue: Unable to decide

What if NIDS see?



Should it assume `RST` has arrived or not?

Issue: Incosistency in parsing / encrypted traffic

- What if /%65%74%63/%70%61%73%73%77%64?
- What if ../../../../../../../?
- What to do with HTTPs traffic?

Confusing specification of HTTP 1.1

```
GET http://www.fuzzybunnies.com/ HTTP/1.1
Host: www.bunnyoutlet.com
```

Confusing specification of HTTP 1.1

```
Content-Disposition:
    attachment; filename="evil_file.exe;.txt"
```

Firewalls are *reference monitors*

- unbypassable
- tamper resistent
- and verifiable

Plan

Motivation

Detect the attack as early as possible and cut off the source.

Variants of Intrusion detection

Deployment point:

- Host
- Server
- Network
- Distributed

Detection engine:

- Signature
- Vulnerability
- Behaviour
- Anomaly

Source of data:

- Live pass thorough
- Syscalls
- Logs

Network intrusion detection systems

Pros.

- Cheap to deploy and maintain.
- Easy to add to existing infrastructure.
- Cover all hosts inside the traffic.
- Does not consume production resources.
- MITM for HTTPs.
- Extensive logging.
- Autoupdate of (behavioural) signatures.

Signature based IDS

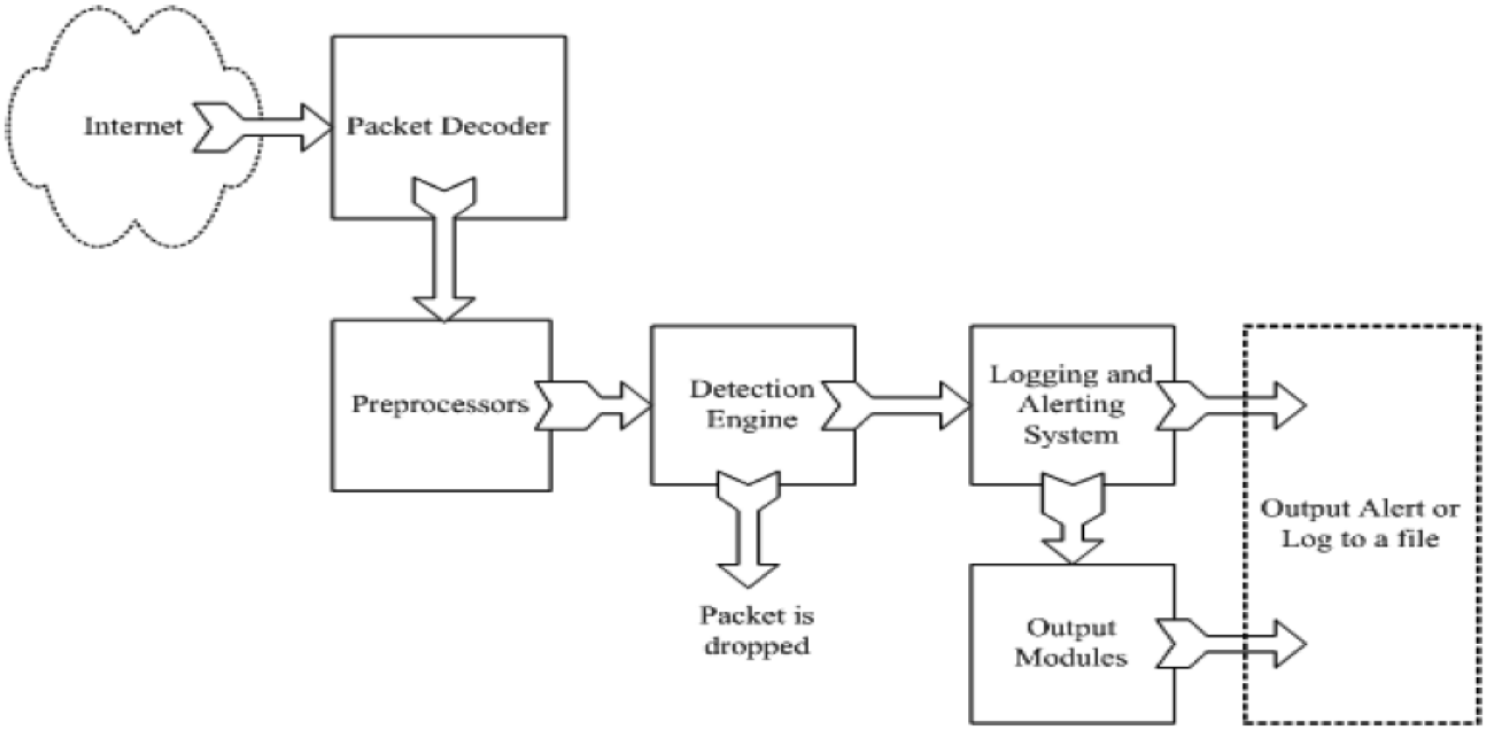
Pros.

- Conceptually fairly simple
- Takes care of known attacks
- Easy to share signatures, build up libraries
- Can detect variants of known attacks
- Much more concise than per-attack signatures

Cons.

- Size of the database (3500).
- Most time spent on signature matching.
- Cannot detect new threats or variants of existing treats.
- prone to problems with protocol understanding

Example: Signature-matching IDS — SNORT



Examples of SNORT rules

```
alert tcp any any -> any 139 \
      (content:"|5c 00|P|00|I|00|P|00|E|00 5c|";)

alert tcp any any -> any 80 (content:!"GET";)

alert tcp !$HOME_NET any -> $HOME_NET 21 \
      (msg:"cd incoming detected"; flow:from_client; \
       content:"CWD incoming"; nocase;)
```

Vulnerability signatures

Do not match signature but known vulnerability.

Example with Snort

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80
uricontent: ".ida?"; nocase; dsize: > 239; flags:A+
msg:"Web-IIS ISAPI .ida attempt"
reference:bugtraq,1816
reference:cve,CAN-2000-0071
classtype:attempted-admin
```

Host- / server- based IDS

Pros.

- Have full visibility into arguments and network traffic.
- Can instrument programs in sandbox.

Cons.

- Needs to be tailored to each app.
- Does not solve the problems with filename semantics `../../../../../.`

Behavior-based intrusion detection

Attack's follow patterns, detect them!

1. Reconnaissance
2. Initial exploit
3. Establishing presence
4. Installing tools
5. Lateral movement

Anomaly-based intrusion detection

Try to model typical users and detect deviations from their behaviour.

Log based IDS

Search *extensive* logs for the presence of attacks.

Log based IDS

Pros:

- Cheap, since important daemons have logging support.
- No problems with %-escapes, encrypted HTTPS

Cons:

- Filename tricks still posses a problem.
- Can't block attacks & prevent from happening.
- Detection delayed, so attack damage may compound.
- If machine is compromised, logs might be altered.

System Call Monitoring (HIDS)

Monitor system call activity of processes and look for manipulation with suspicious resources or suspicious sequences.

System Call Monitoring (HIDS)

Pros:

- No issues with any protocol complexities.
- May avoid issues with filename tricks.
- Cannot prevent the attack.

Cons:

- False positives.
- Cannot detect failed attempts.

Modern HIDS

- Can execute in sandbox.
- Analyse system calls, registry key, mutexes, files.
- Apply heuristics and signatures.
- Scans memory for malware that does not install on disk.