10. Secure Network Architectures

Computer Security Courses @ POLIMI

Prof. Carminati & Prof. Zanero

Firewalls

Firewall: network access control system that verifies all the packets flowing through it.

Its main **functions** are usually:

- IP packet filtering
- Network address translation (NAT)

Must be the single enforcement point between a screened network and outside networks.

A Side Note on Grammar

Bear in mind:

- "firewall" - correct
- “fire wall” - incorrect
- “un firewall” (IT) - correct
- “una firewall” (IT) - incorrect

A firewall (constructIon) is a *wall* designed to partition a building and stop fire spreading,

“Wall of fire” is a 4th level spell.

Firewalls are not Omnipotent

A firewall checks **all** the traffic flowing through it, and **only** the traffic flowing through it.

Powerless against **insider attacks** (unless the network is partitioned somehow).

In general, **powerless against unchecked paths**

- E.g. a modem or 4G connection of machine connected also to a LAN

Firewalls are Computers

The firewall itself is a computer: it could have vulnerabilities and be violated.

However, usually offers few or no services, so it has a small attack surface.

Most of the times it's an embedded appliance with just a firmware.

Security Policies = Firewall Rules

A firewall is a stupid “bouncer at the door”

- Just applies rules
- Bad rules = no protection

Firewall **rules** are essentially the implementation of **higher-level security policies**.

- E.g. “I want no clients to be able to download email from external email servers!”

Policy must be built on a **default deny base**:

- I.e., "deny all, except ..."

Firewall Taxonomy

We divide them depending on their **packet inspection capability** (from low to high layers).

Network layer firewalls

- Packet filters firewall (network)
- Stateful packet filters firewall (network-transport)

Application layer firewalls

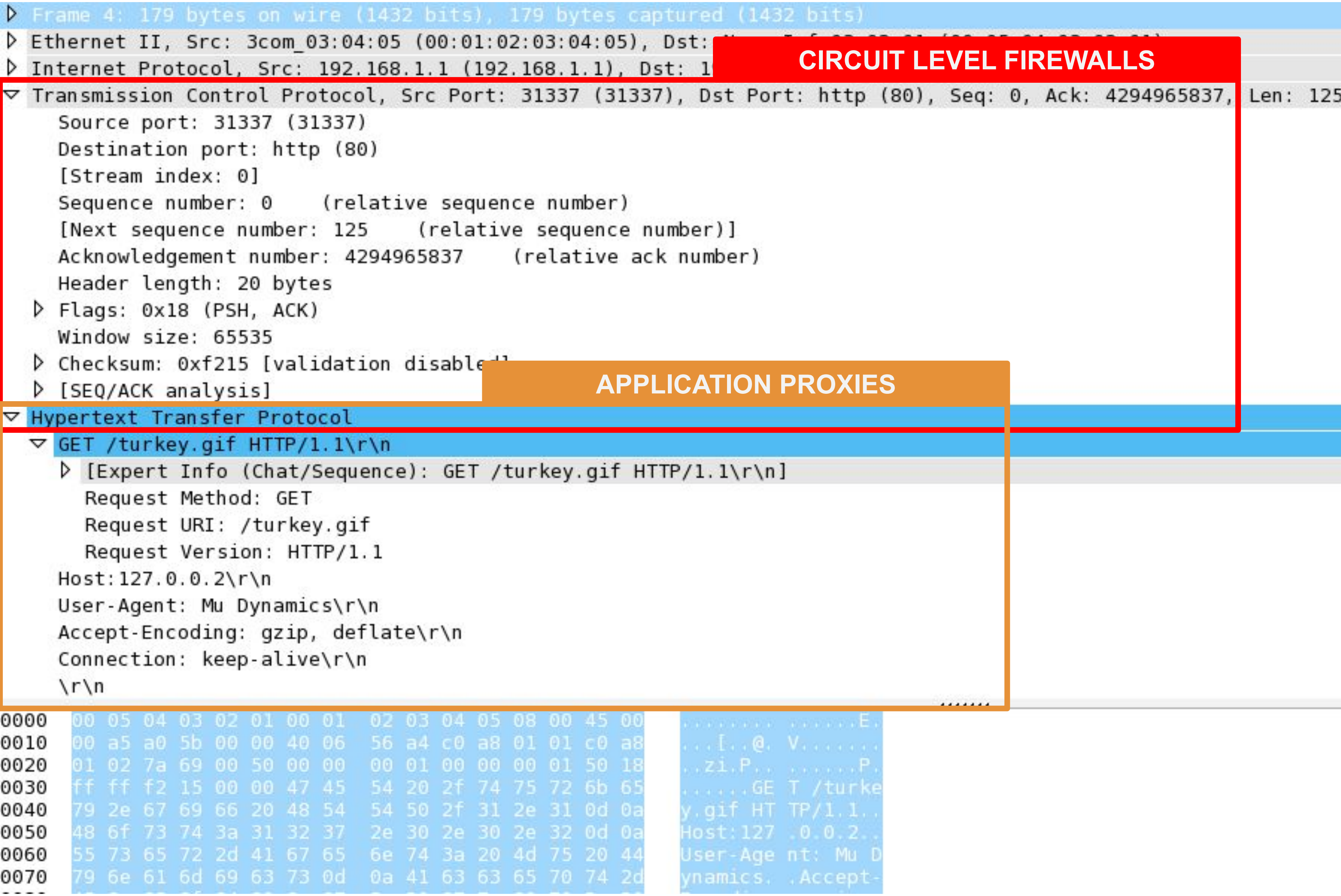
- Circuit level firewalls (transport-application)
- Application proxies (application)

NETWORK PACKET FILTERS

STATEFUL PACKET FILTERS

```
▷ Frame 1: 54 bytes on wire (432 bits), 54 bytes captured
▷ Ethernet II, Src: 3com_03:04:05 (00:01:02:03:04:05), Dst: NarayInf_03:02:01 (00:05:04:03:02:01)
▽ Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 40
    Identification: 0x3d2b (15659)
  ▷ Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
  ▷ Header checksum: 0xba51 [correct]
    Source: 192.168.1.1 (192.168.1.1)
    Destination: 192.168.1.2 (192.168.1.2)
▽ Transmission Control Protocol, Src Port: 31337 (31337), Dst Port: http (80), Seq: 4294967295, Len: 0
    Source port: 31337 (31337)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 4294967295    (relative sequence number)
    Header length: 20 bytes
  ▷ Flags: 0x02 (SYN)
    Window size: 65535
  ▷ Checksum: 0xb1d5 [validation disabled]
  ▷ [SEQ/ACK analysis]

0000  00 05 04 03 02 01 00 01  02 03 04 05 08 00 45 00   ........ ......E.
0010  00 28 3d 2b 00 00 40 06  ba 51 c0 a8 01 01 c0 a8   .(=+..@. .Q......
0020  01 02 7a 69 00 50 00 00  00 00 00 00 00 00 50 02   ..zi.P.. ......P.
0030  ff ff b1 d5 00 00                                  ......
```


CIRCUIT LEVEL FIREWALLS
APPLICATION PROXIES
Frame 4: 179 bytes on wire (1432 bits), 179 bytes captured (1432 bits)
Ethernet II, Src: 3com_03:04:05 (00:01:02:03:04:05), Dst:
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 1
Transmission Control Protocol, Src Port: 31337 (31337), Dst Port: http (80), Seq: 0, Ack: 4294965837, Len: 125
Source port: 31337 (31337)
Destination port: http (80)
[Stream index: 0]
Sequence number: 0 (relative sequence number)
[Next sequence number: 125 (relative sequence number)]
Acknowledgement number: 4294965837 (relative ack number)
Header length: 20 bytes
Flags: 0x18 (PSH, ACK)
Window size: 65535
Checksum: 0xf215 [validation disable
[SEQ/ACK analysis]
Hypertext Transfer Protocol
GET /turkey.gif HTTP/1.1\r\n
[Expert Info (Chat/Sequence): GET /turkey.gif HTTP/1.1\r\n]
Request Method: GET
Request URI: /turkey.gif
Request Version: HTTP/1.1
Host:127.0.0.2\r\n
User-Agent: Mu Dynamics\r\n
Accept-Encoding: gzip, deflate\r\n
Connection: keep-alive\r\n
\r\n
0000 00 05 04 03 02 01 00 01 02 03 04 05 08 00 45 00 ....E.
0010 00 a5 a0 5b 00 00 40 06 56 a4 c0 a8 01 01 c0 a8 ...[..@. V.......
0020 01 02 7a 69 00 50 00 00 00 01 00 00 00 01 50 18 ..zi.P.. ......P.
0030 ff ff f2 15 00 00 47 45 54 20 2f 74 75 72 6b 65 ......GE T /turke
0040 79 2e 67 69 66 20 48 54 54 50 2f 31 2e 31 0d 0a y.gif HT TP/1.1..
0050 48 6f 73 74 3a 31 32 37 2e 30 2e 30 2e 32 0d 0a Host:127 .0.0.2..
0060 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 75 20 44 User-Age nt: Mu D
0070 79 6e 61 6d 69 63 73 0d 0a 41 63 63 65 70 74 2d ynamics. .Accept-

Packet Filters

<u>Packet by packet processing.</u>

Decodes the IP (and part of the TCP) header:

- SRC and DST IP
- SRC and DST port
- Protocol type
- IP options

Stateless

- cannot track TCP connections.
- No payload inspection.

Mostly found on routers (“ACLs” on Cisco)

Expressiveness

Predicate only on what we can decode. E.g.:

- Block any incoming packet ("default deny")
  - **`iptables -P INPUT DROP`** `#P = policy`
- Allow incoming packet if going to 10.0.0.1
  - **`iptables -A INPUT -d 10.0.0.1 -j ALLOW`**
- Block anything out except SMTP (port 25)
  - **`iptables -P OUTPUT DROP`**
  - **`iptables -A OUTPUT --dport 25 -j ALLOW`**

What happens with packets with spoofed IP?

Rule ~> Reaction

Regardless of the specific syntax, every network packet filter allows to express the following concept:

- if (packet matches certain condition)
  - do this, e.g.:
    - block
    - allow
    - log
    - (other actions)

Stateful (or Dynamic) Packet Filters

Include network packet filters, plus:

- keep track of the **TCP state machine**
    - after SYN, SYN-ACK must follow
- we can **track connections** without adding a response rule.

Performance bounded on a **per-connection basis**, not only on a **per-packet basis**.

- The **number of simultaneous connections** are just as important as packets per second.

Better Expressiveness and Pluses

Tracking (Logging and accounting on) connections.

Deeper content inspection:

- Reconstruct application-layer protocols
- Application-layer filtering, e.g. ActiveX objects in HTTP responses disallowed.

Network Address Translation (**NAT**).

Packet defragmenting and reassembly (helps avoiding pathological fragmented packets, e.g., teardrop).

Session Handling

A **session** is an atomic, transport-layer exchange of application data between 2 hosts.

Main transport protocols:

- TCP (Transmission Control Protocol)
  - session =~ TCP connection
- UDP (User Datagram Protocol)
  - session = this concept does not exist

Session-handling is fundamental for NAT.

NAT Session Initialization (TCP)

SRC IP	192.168.2.1	131.175.124.139
DST IP	198.133.219.25	198.133.219.25
SRC PORT	1026	1026
DST PORT	80	80
ISN	49091	49769
ACK	. . .	. . .
FLAG	SYN	SYN

- NAT slot exists?
- If not, check rules
- If allowed, create new NAT slot.

198.133.219.25	198.133.219.25
192.168.2.1	131.175.124.139
80	80
1026	1026
92513	92513
49092	49770
SYN-ACK	SYN-ACK

- NAT slot exists? YES, If answer is not a SYN-ACK or not correct ISN, discard packet.

Stateful Firewall

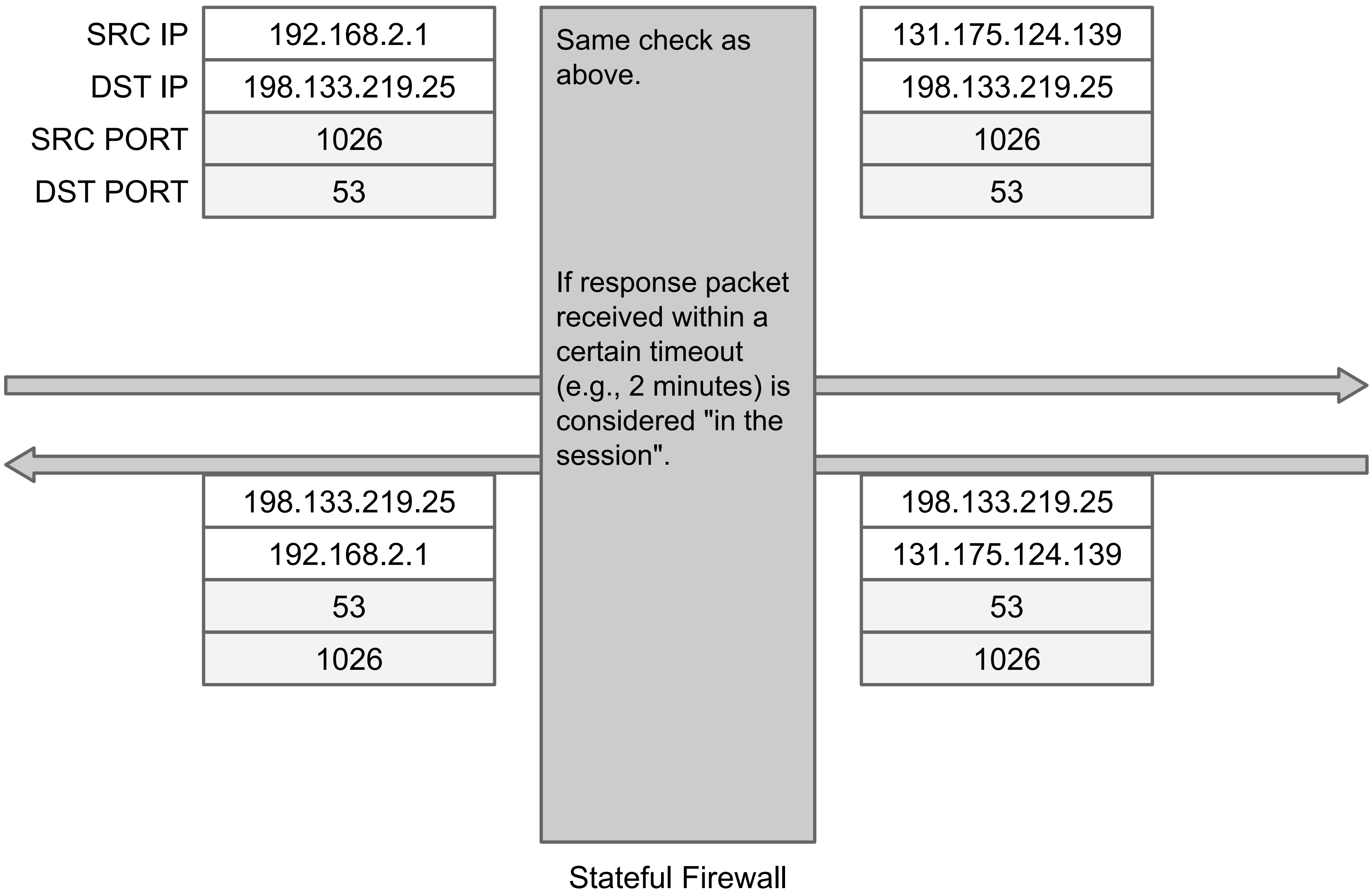
What about UDP

Widely used, so we can't dismiss it (e.g. DNS, VoIP H.323, video streaming).

Difficult to secure and handle, because there are no connections.

Connectionless, but a “session” can be inferred for NAT and controls.

UDP and NAT


SRC IP
DST IP
SRC PORT
DST PORT
192.168.2.1
198.133.219.25
1026
53
Same check as above.
If response packet received within a certain timeout (e.g., 2 minutes) is considered "in the session".
131.175.124.139
198.133.219.25
1026
53
198.133.219.25
192.168.2.1
53
1026
198.133.219.25
131.175.124.139
53
1026
Stateful Firewall

Application-layer Inspection for NAT

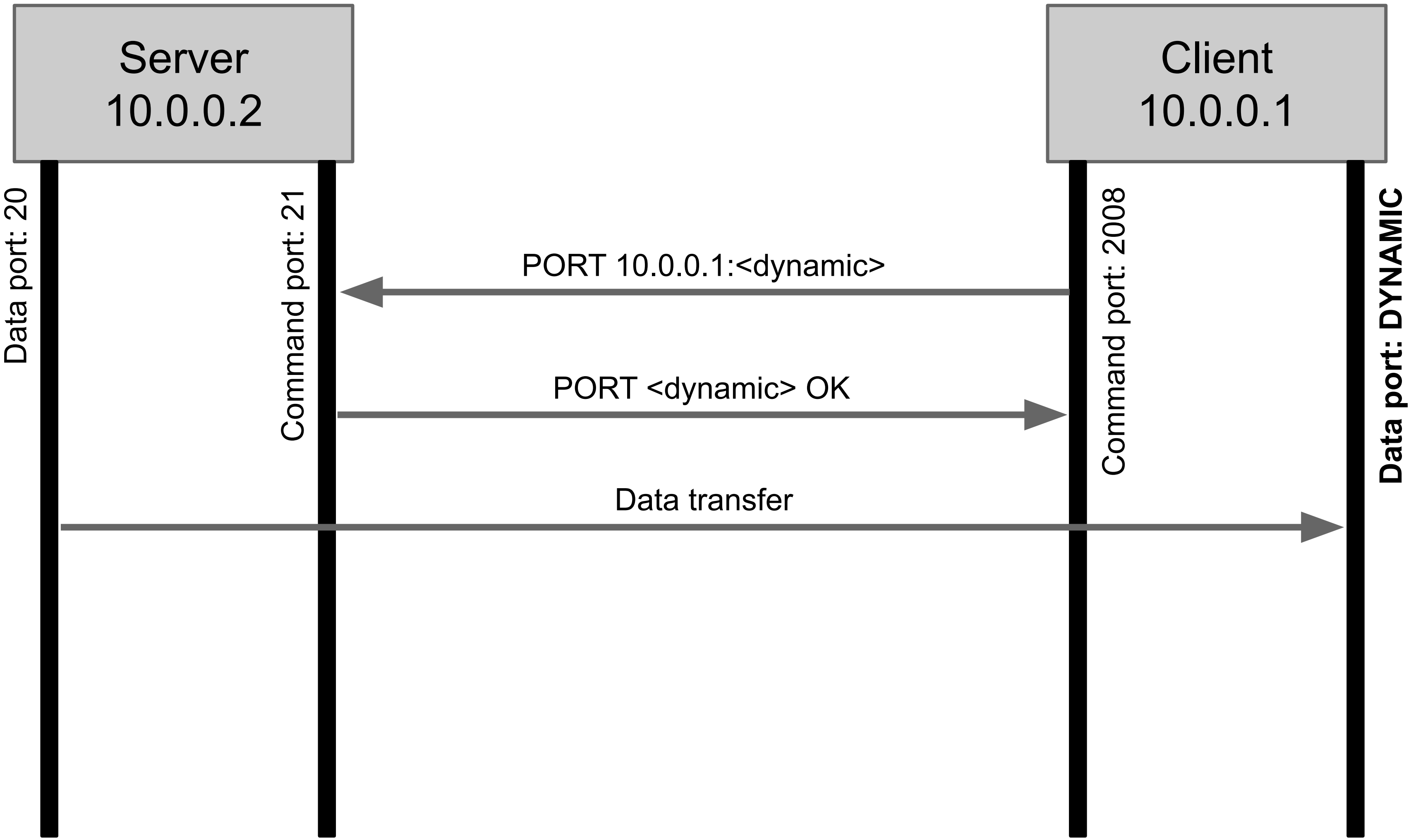
Some protocols (e.g., DCC, RDT, instant messengers, file transfer) transmit network information data (e.g., port) at application layer.

For instance, FTP uses dynamic connections:

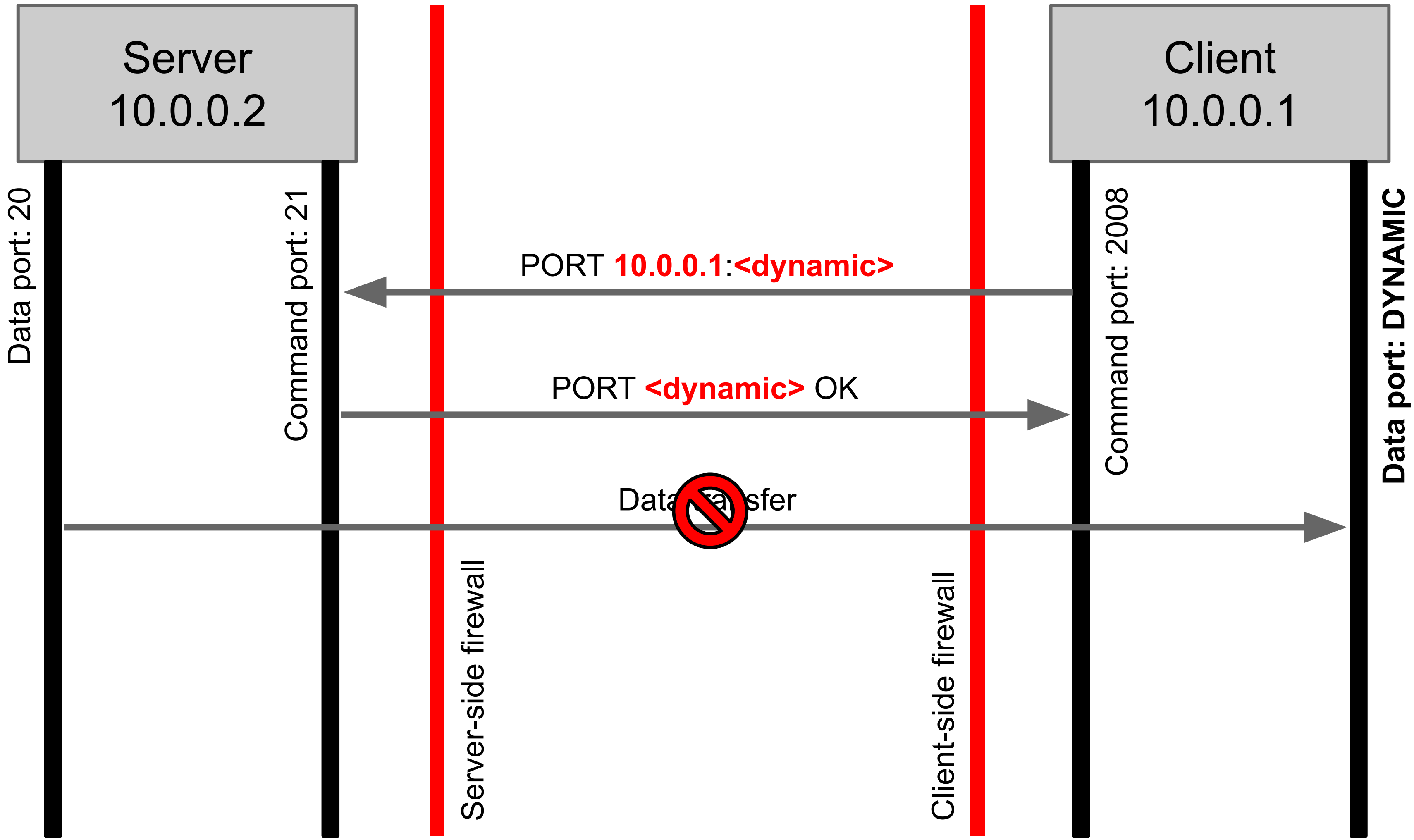
- Allocated for file uploads, downloads, output of commands
- "PORT" application command

<u>Stateful firewalls must take care of these.</u>

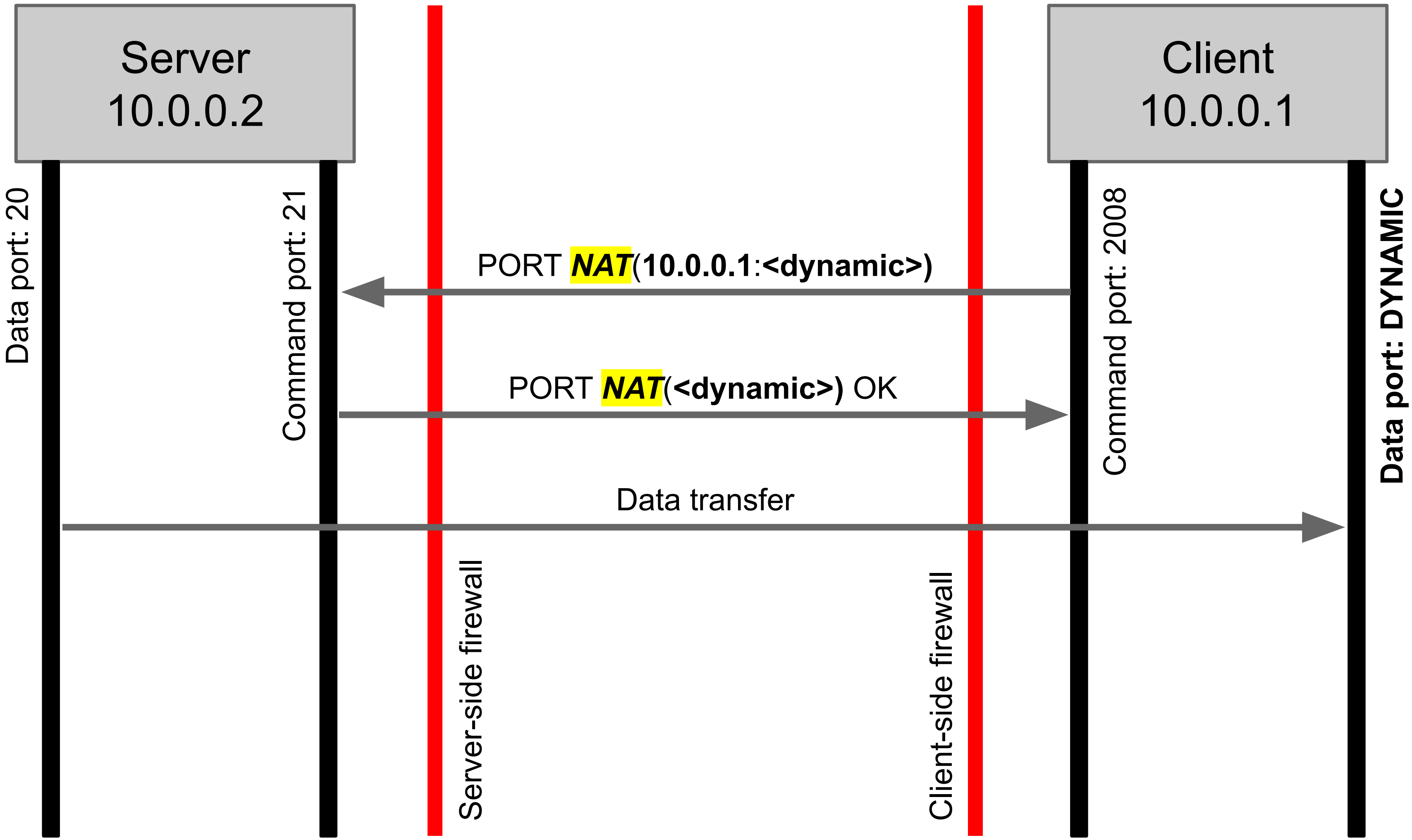
Example: FTP Standard Mode (NO FW)



Example: FTP Standard Mode (1)



Example: FTP Standard Mode (1)



```
▷ Frame 14: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
▷ Ethernet II, Src: Woonsang_04:05:06 (01:02:03:04:05:06), Dst: 06:05:04:03:02:01 (06:05:04:03:02:01)
▷ Internet Protocol, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▽ Transmission Control Protocol, Src Port: avt-profile-1 (5004), Dst Port: ftp (21), Seq: 41, Ack: 48, Len: 22
    Source port: avt-profile-1 (5004)
    Destination port: ftp (21)
    [Stream index: 0]
    Sequence number: 41    (relative sequence number)
    [Next sequence number: 63    (relative sequence number)]
    Acknowledgement number: 48    (relative ack number)
    Header length: 20 bytes
  ▷ Flags: 0x18 (PSH, ACK)
    Window size: 32768
  ▷ Checksum: 0xfc7b [validation disabl
  ▷ [SEQ/ACK analysis]
```

APPLICATION LAYER

```
▽ File Transfer Protocol (FTP)
  ▽ PORT 127,0,0,1,37,75\r\n
      Request command: PORT
      Request arg: 127,0,0,1,37,75
      Active IP address: 127.0.0.1 (127.0.0.1)
      Active port: 9547
```

```
0000  06 05 04 03 02 01 01 02  03 04 05 06 08 00 45 00   ........ ......E.
0010  00 3e 08 00 00 00 40 06  74 b8 7f 00 00 01 7f 00   .>....@. t.......
0020  00 01 13 8c 00 15 00 00  00 2a 00 00 00 31 50 18   ........ .*...1P.
0030  80 00 fc 7b 00 00 50 4f  52 54 20 31 32 37 2c 30   ...{..PO RT 127,0
0040  2c 30 2c 31 2c 33 37 2c  37 35 0d 0a               ,0,1,37, 75..
```

Example: FTP Standard Mode (2)

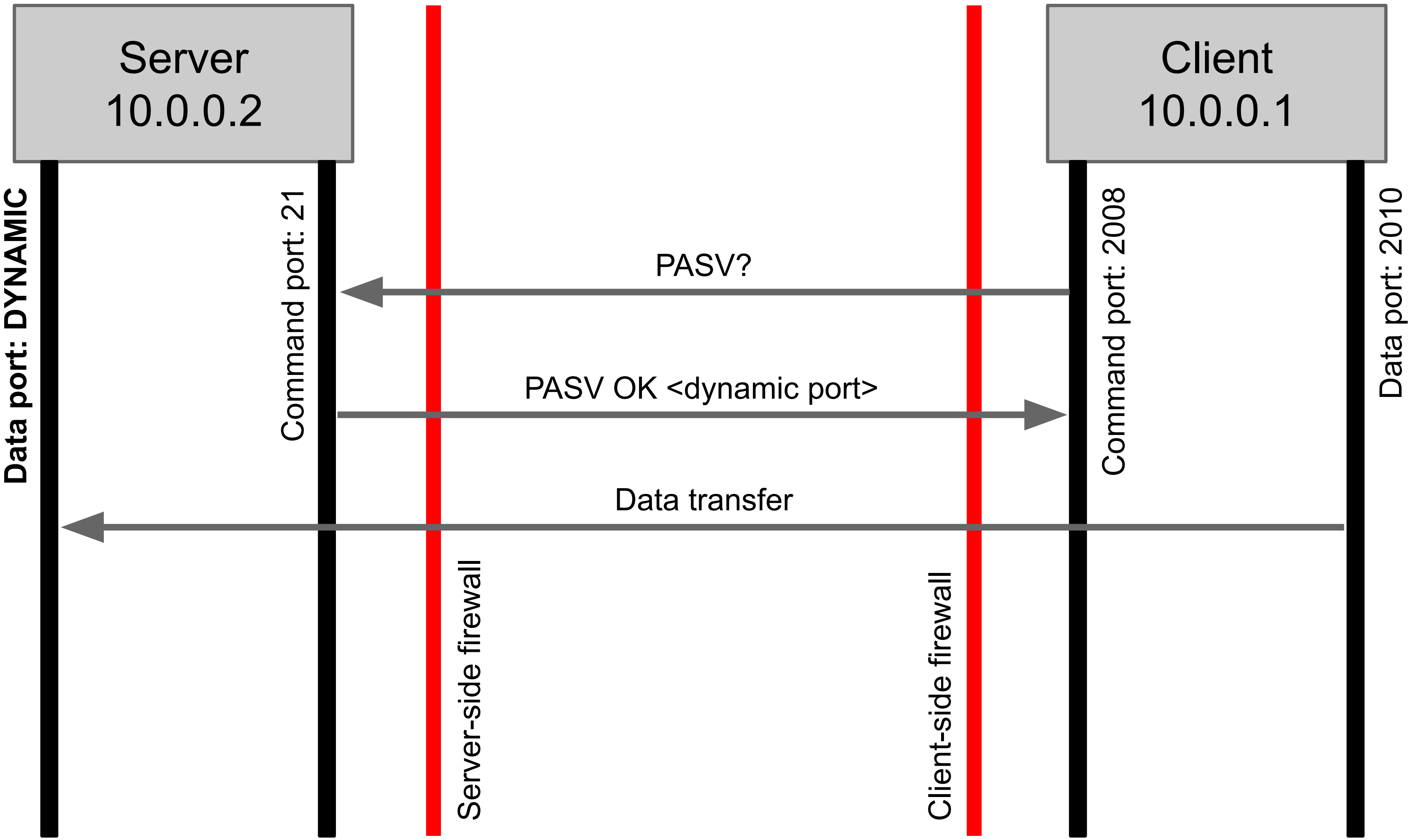
Client-side firewall:

- must open port 21 outbound
- must dynamically open/close the (inbound) ports that the client specifies in the command.

Server-side firewall:

- must open port 21 inbound
- must dynamically open/close the (outbound) ports that the client specifies in the command.

Example: FTP Passive Mode (1)



```
▷ Frame 50: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
▷ Ethernet II, Src: HewlettP_af:b0:ba (1c:c1:de:af:b0:ba), Dst: 54:04:a6:3c:ed:2b (54:04:a6:3c:ed:2b)
▷ Internet Protocol, Src: 192.168.1.182 (192.168.1.182), Dst: 192.168.1.101 (192.168.1.101)
▽ Transmission Control Protocol, Src Port: 48956 (48956), Dst Port: ftp (21), Seq: 100, Ack: 385, Len: 6
    Source port: 48956 (48956)
    Destination port: ftp (21)
    [Stream index: 0]
    Sequence number: 100    (relative sequence number)
    [Next sequence number: 106    (relative sequence number)]
    Acknowledgement number: 385    (relative ack number)
    Header length: 32 bytes
  ▷ Flags: 0x18 (PSH, ACK)
    Window size: 14720 (scaled)
  ▷ Checksum: 0x3574 [validation disabled]
  ▷ Options: (12 bytes)
  ▷ [SEQ/ACK analysis]
```

APPLICATION LAYER

```
▽ File Transfer Protocol (FTP)
  ▽ PASV\r\n
      Request command: PASV
```

```
0000  54 04 a6 3c ed 2b 1c c1  de af b0 ba 08 00 45 00   T..<.+.. ......E.
0010  00 3a 7b 02 40 00 40 06  3b 50 c0 a8 01 b6 c0 a8   .:{.@.@. ;P......
0020  01 65 bf 3c 00 15 2a 8c  25 b2 da d0 4f df 80 18   .e.<..*. %...O...
0030  00 73 35 74 00 00 01 01  08 0a 00 46 79 25 00 8a   .s5t.... ...Fy%..
0040  57 85 50 41 53 56 0d 0a                            W.PASV..
```

Example: FTP Passive Mode (2)

Both channels are client-initiated.

Client-side firewall:

- must open port 21 outbound
- must dynamically open/close the (outbound) ports that the client specifies in the command.

Server-side firewall:

- must open port 21 inbound
- must dynamically open/close the (inbound) ports that the client specifies in the command.

Deep Inspection & Intrusions

Modern firewalls can analyze packets and sessions even more in depth.

- E.g. recognize MIME multipart attachments in SMTP flows and send data to antiviruses.
- Recognize a set of “known attack packets” patterns to be blocked (IPS, Intrusion Prevention Systems)
- Update problems
- Zero-days

Circuit Firewalls (Legacy)

Relay TCP connections.

Client connects to a specific TCP port on the firewall, which then connects to the address and port of the desired server (**not transparent!**).

Essentially, a TCP-level proxy.

Only historical example: SOCKS

Application Proxies

Same as circuit firewalls, but at application layer.

Almost never transparent to clients:

- May require modifications
- Each protocol needs its own proxy server

Inspect, validate, manipulate protocol application data (e.g., rewrite HTTP frames)

Functionalities & Additional Benefits

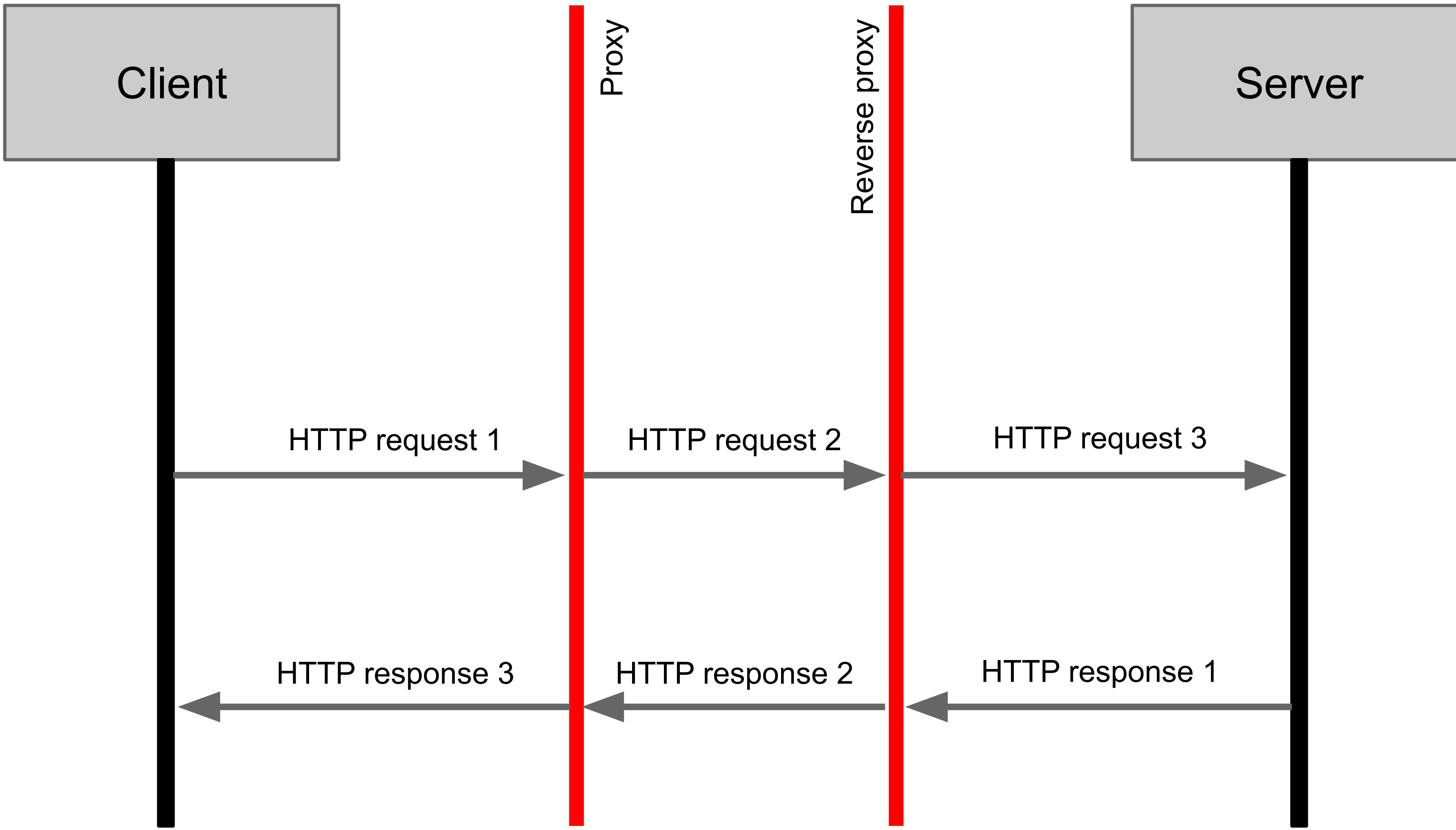
Can authenticate users, apply specific filtering policies, perform advanced logging, content filtering or scanning (e.g., anti-virus/spam).

Can be used to expose a subset of the protocol:

- to defend clients,
- to defend servers (“reverse proxy”).

Usually implemented on COTS OSs.

Example: HTTP Proxy (1)



Example: HTTP Proxy (2) - client

Example: HTTP Proxy (3)

Example request that the client sent to the proxy.

The proxy could modify it, if needed.

```
GET / HTTP/1.1
Host: maggi.cc
Proxy-Connection: keep-alive
Cache-Control: no-cache
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Pragma: no-cache
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/35.0.1916.99 Safari/537.36
DNT: 1
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,it;q=0.6
Cookie: __utma=15436074.439502269.1326703042.1399588315.1400190112.331;
__utmb=15436074.5.10.1400190112; __utmc=15436074;
__utmz=15436074.1376676768.202.4.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmctr=(n
ot%20provided)
```

Example: HTTP Proxy (4)

Example response that the proxy receives from the server.

The proxy could modify it, if needed.

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 15 May 2014 21:48:09 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Vary: Accept-Encoding
Content-Length: 19263

</doctype html lang="en">

</--
paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/
-->
```

Architectures for secure networks

Dual- or Multi-zone Architectures (1)

In most cases, the perimeter defense works on the assumption that what is “good” is inside, and what's outside should be kept outside if possible.

There are two counterexamples:

- Access to resources from remote (i.e., to a web server, to FTP, mail transfer).
- Access from remote users to the corporate network.

Dual- or Multi-zone Architectures (2)

Problem: if we mix externally accessible servers with internal clients, we lower the security of the internal network.

Solution: we allow external access to the accessible servers, but not to the internal network.

General idea: split the network by privileges levels. Firewalls to regulate access.

Dual- or Multi-zone Architectures (3)

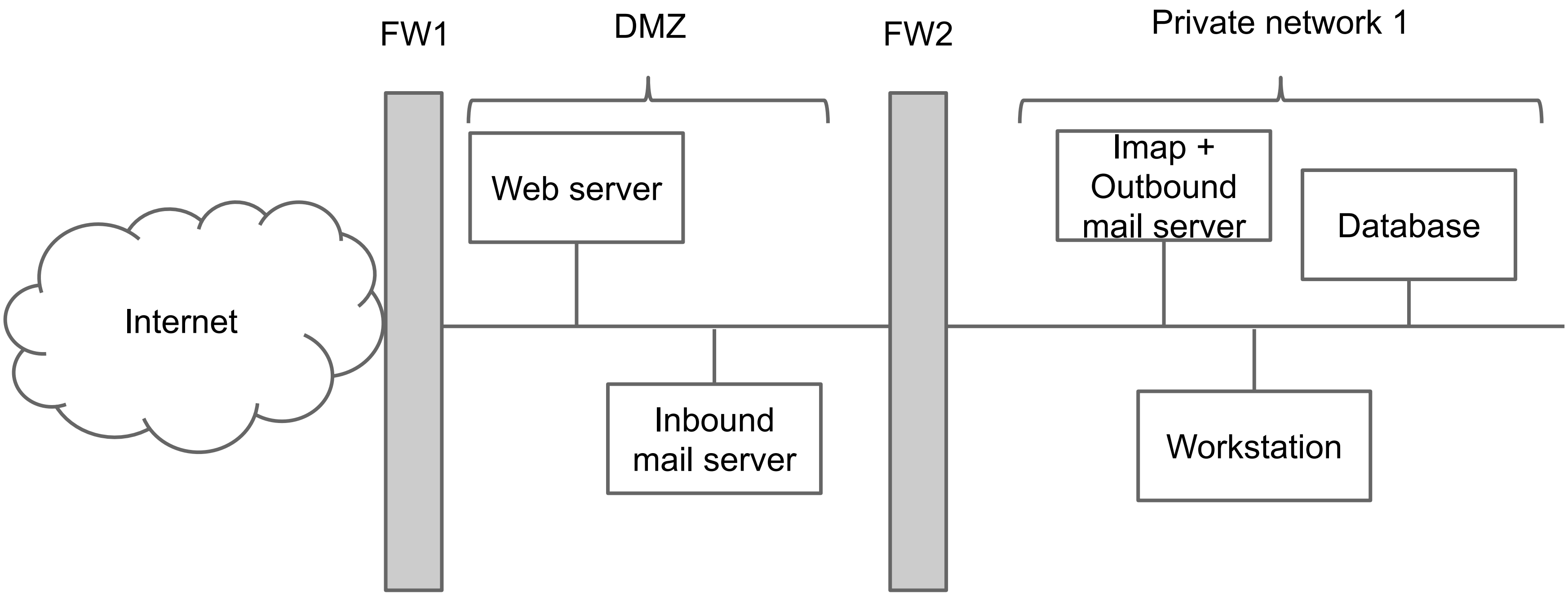
In practice, we create a semi-public zone called DMZ (demilitarized zone).

The DMZ will host public servers (web, FTP, public DNS server, intake SMTP).

On the DMZ no critical or irreplaceable data.

The DMZ is almost as risky as the Internet.

Example DMZ + Private Zone



Non critical machines.

No sensitive data.

Publicly-reachable.

Critical machines.

Sensitive data.

Not exposed to the public.

Exercise

Write the rules for the previous network layout.

Implement it with a single, multi-homed firewall (i.e., a firewall with more than one port).

Virtual Private Networks (VPNs)

Requirements:

- Remote employees need to work “as if” they were in the office, accessing resources on the private zone.
- Connecting remote sites without using dedicated lines.

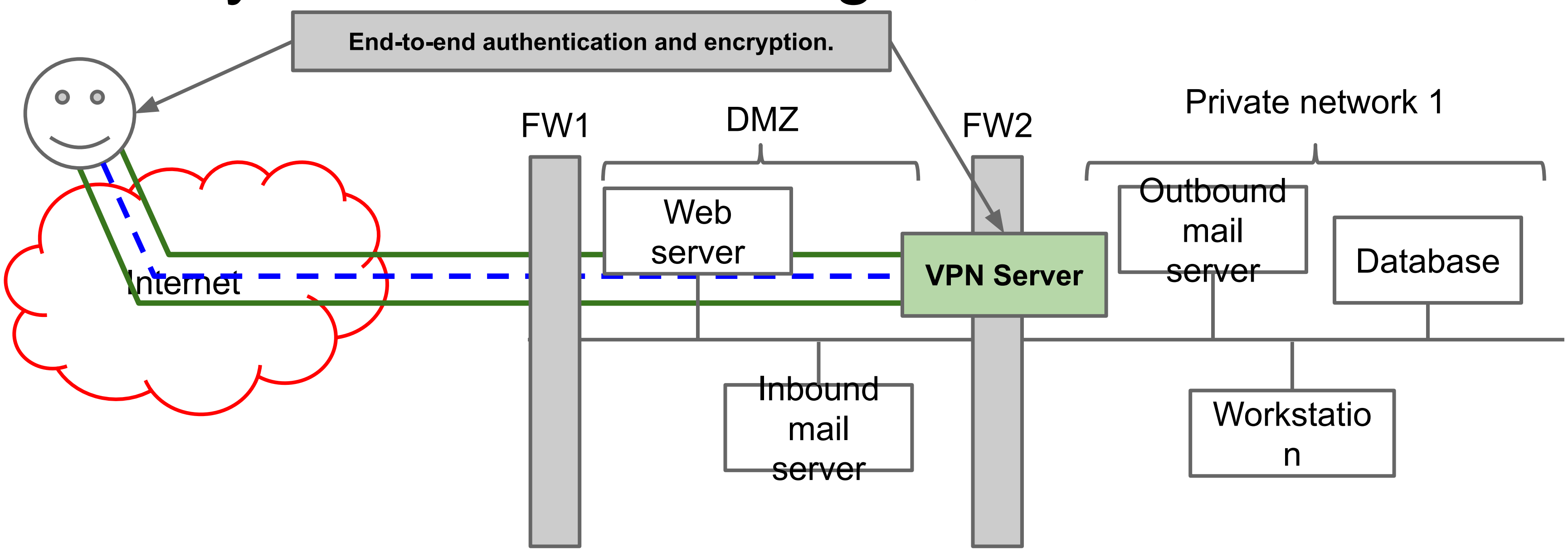
Which means:

Ensure CIA to data transmitted over a public network (i.e., the Internet).

VPNs: Basic Concept

Solution: use a VPN, an encrypted overlay connection over a (public) network.

Many different technologies, same basic idea.



Two VPN Modes: Full vs. Split

Full tunnelling

- Every packet goes through the tunnel.
- Traffic multiplication, could be inefficient.
- Single point of control and application of all security policies as if the client were in the corporate network.

Split tunnelling

- Traffic to the corporate network: in VPN; traffic to the Internet: directly to ISP.
- More efficient, less control.
- Just similar to the case of the PC connected via 4G modem to the Internet.

VPN Technologies

PPTP (Point-to-point Tunnelling Protocol): proprietary Microsoft protocol, variant of PPP with authentication and cryptography.

VPN over **SSL**, **SSH tunnel** (we will see the SSL protocol in detail), or **OpenVPN.net** (open source).

IPSEC

- Security extensions of IPv6, backported to IPv4
- Authentication and cryptography at IP layer

Conclusions

Firewalls can enforce policies only on the traffic that they can inspect, and up to the layer that they can decode.

Firewalls can be used to implement multi-zone architectures.

VPNs solve the problem of creating a trusted network transport over an untrusted channel.