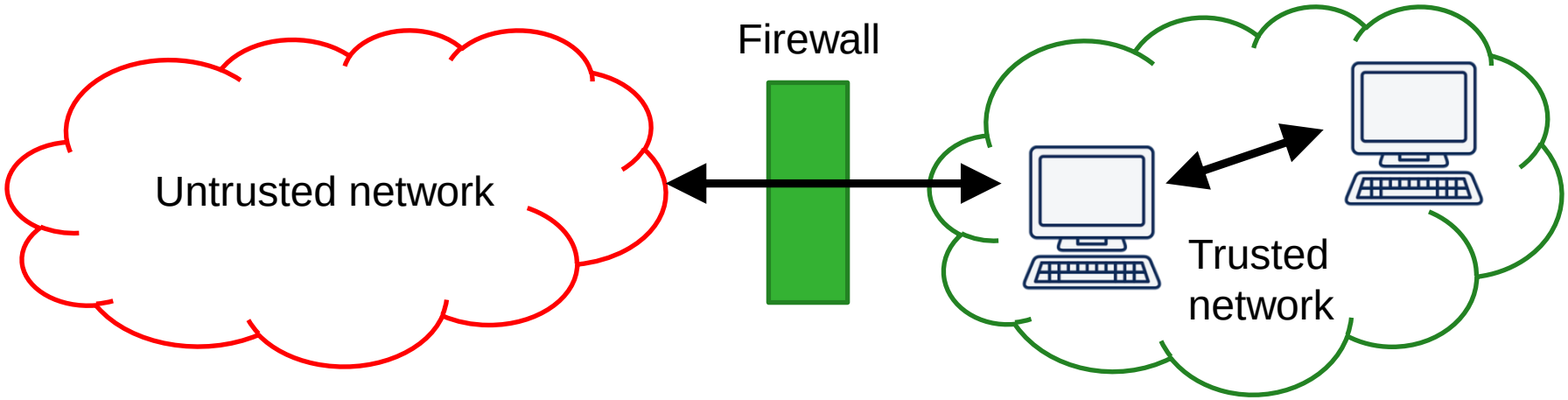
Firewalls

Ramin Sadre

Firewalls



- A firewall tries to improve security by isolating a *trusted* network from an *untrusted* network
  - Example: The internal network of a company should be isolated from the Internet
- The firewall is a check point where all traffic between the two networks has to pass through
  - Basic assumption: Attacks are coming from the untrusted network

Types of firewalls

- Firewalls can be classified by their
  1. Layer of operation:
     - Network-layer firewall (IPv4, IPv6,…)
     - Transport-layer firewall (UDP, TCP,…)
     - Application-layer firewall (HTTP,…)
  2. Internal state
     - Stateless firewall
     - Statefull firewall
  3. Location:
     - Network firewall
     - Personal (host) firewall

Layer of operation

- Firewalls can inspect the network traffic on different protocol level
- Examples:
  - Network level: “Do not allow traffic from 1.2.3.4“
  - Transport level: “Only allow traffic to port 80“
  - Application level: “Do not allow HTTP POST requests to our web server 2.3.4.5“

Stateless firewalls

- Let's start with the simplest case: A stateless firewall
- Such firewalls operate as rule-based ***packet filters***
  - They look at every packet
  - Rules decide whether packet should be dropped or forwarded
- Examples for rules:
  - Permit all TCP packets from a certain host and port to a certain host and port

```
allow tcp 4.5.5.4:1025 -> 3.1.1.2:80
```

  - Drop all TCP packets from a certain host to a certain host and port

```
deny tcp 4.5.5.4:* -> 3.1.1.2:80
```

Rule lists

- The firewall will go through all rules in their rule list until one rule matches and execute ist operation (allow or deny)

- What does this rule list do?

```
deny tcp 4.5.5.4:* -> 3.1.1.2:80
allow tcp 4.5.5.4:1025 -> 3.1.1.2:80
```

  First rule will drop all TCP packets from 4.5.5.4 to 3.1.1.2:80.

  Second rule is useless.

- You probably wanted this:

```
allow tcp 4.5.5.4:1025 -> 3.1.1.2:80
deny tcp 4.5.5.4:* -> 3.1.1.2:80
```

Direction

- Note that a packet filter does not care about the direction of the traffic
- This rule

```
deny tcp *:* -> *:22
```

will drop all TCP traffic to port 22 (ssh), including outbound connections!

Default policy

- Typically, your rule list will end with…
  - Default deny

```
allow tcp 4.5.5.4:1025 -> 3.1.1.2:80
deny tcp 4.5.5.4:* -> 3.1.1.2:80
...
drop * *:* -> *:*
```

  - Default allow

```
allow tcp 4.5.5.4:1025 -> 3.1.1.2:80
deny tcp 4.5.5.4:* -> 3.1.1.2:80
...
allow * *:* -> *:*
```

- In general, a default-deny policy is recommended
  - More conservative (“Block what I don’t know”)
  - If you make a mistake you will notice it quickly

Limitation of stateless firewalls

- Example:
  - Company network: 1.2.3.0/24
  - Our goal: We want maximum security in our company network against attacks from outside. Only allow outbound TCP connections .
- Rules:

```
allow tcp 1.2.3.0/24:* -> *:*
drop * *:* -> *:*
```

- Will this work?
- No! Our stateless firewalls does not know what a TCP connection is. It will drop incoming packets of the outbound TCP connection.

Stateful firewalls

- A **statefull** firewall has an internal list („state“) of the existing connections it has seen previously
  - If a packet comes in, the firewall checks the list whether the packet belongs to one of the existing connections
- In a stateful firewall, this rules will work as expected:

```
allow tcp 1.2.3.0/24:* -> *:*
drop * *:* -> *:*
```

- A statefull firewall understands the protocol (TCP in our example)
  - It knows that a SYN packet send from A to B has to be followed by an SYN-ACK packet from B to A
  - …

Cost of stateful firewalls

- Stateful firewalls need more resources than stateless firewalls because they have to keep track of the state of connections
  - Thousands of open connections even in a small network!
- State information is removed based on
  - observed packets, for example TCP FIN packets
  - timeouts (removes unused connections)

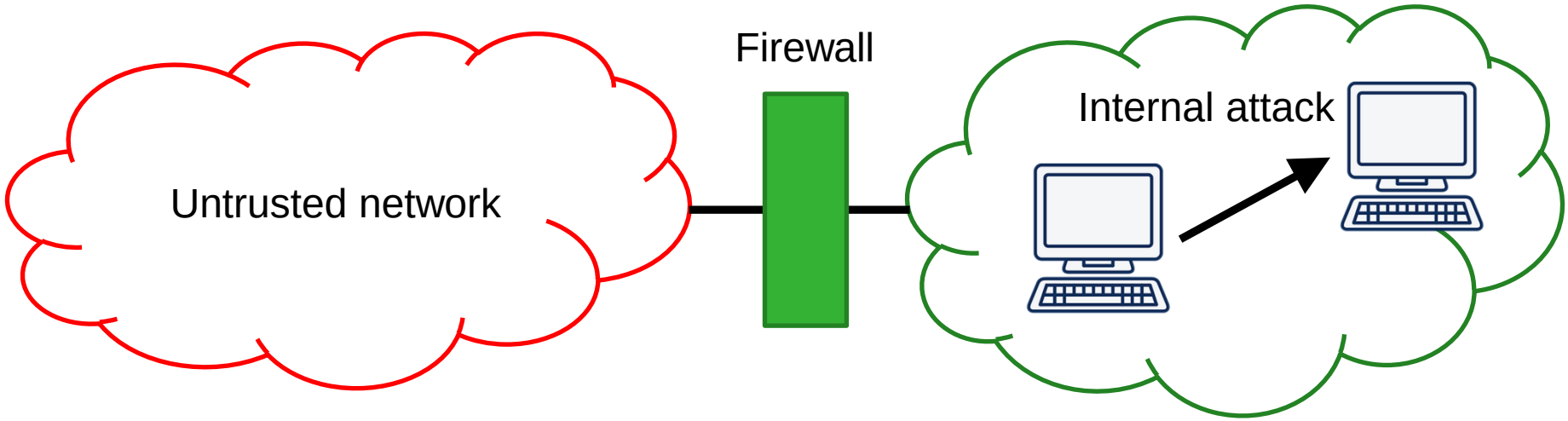
- A timeout will also remove connections that have been inactive for a long time, for example an SSH session
  - To avoid this, protocols like SSH regularly send *Keep-Alive* packets

Stateful firewalls and complex protocols

- Not always trivial to implement a stateful firewall for more complex protocols
- Example: FTP
- Active (Non-passive) FTP connection:
  1. Client opens a control connection to port 21 of server
  2. The server opens a data connection from port 20 to the client
- A stateful firewall that does not know application protocols like FTP would reject the data connection from the server!
- Not surprisingly, application-layer firewalls need more resources than lower-layer firewalls
  - Of course, the more complex a firewall, the higher the risk that it has weaknesses or bugs. Attacker might attack the firewall

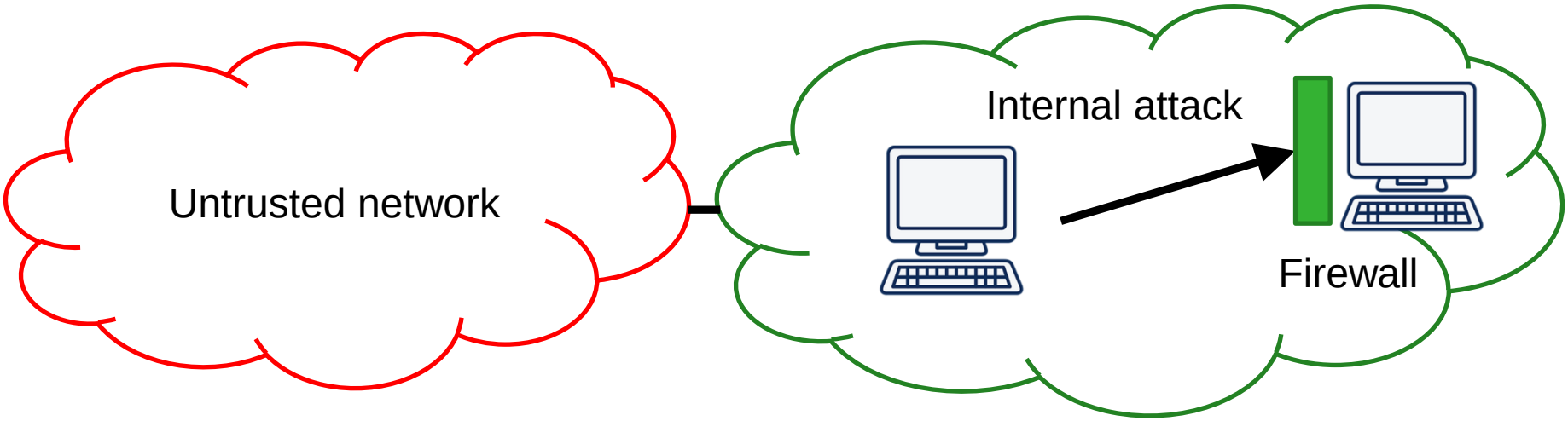
Location: Network firewall

- Easy to administrate
- Protects the entire network
- Does not protect hosts against internal attacks coming from inside the network



Location: Personal firewall

- Runs on a host
- Large administration overhead if network contains hundreds of hosts
- Protects also against internal attacks

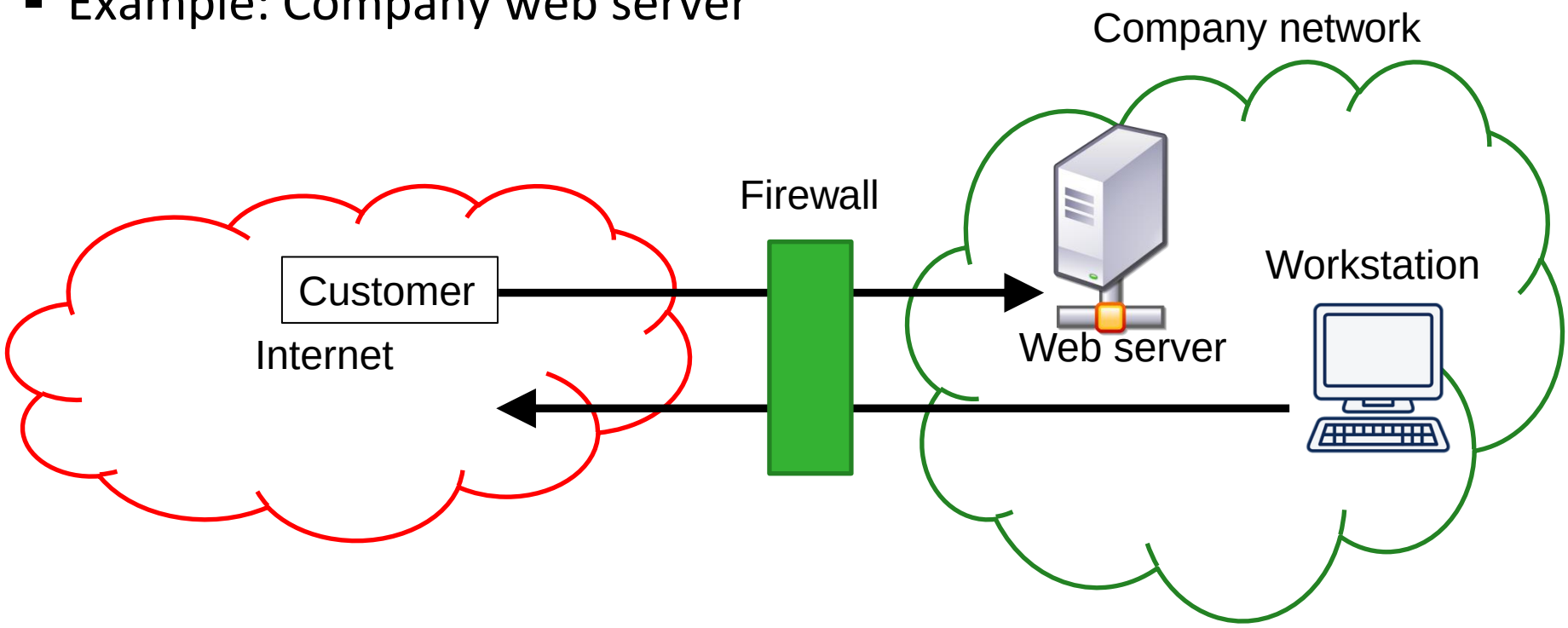


More complex firewall deployments

- Combinations of firewalls are possible
- Example in a company:
  - An expensive (resource-consuming) application-layer firewall for the internal web application server
  - A fast and light-weight transport-layer firewall for the workstations

More complex firewall deployments (2)

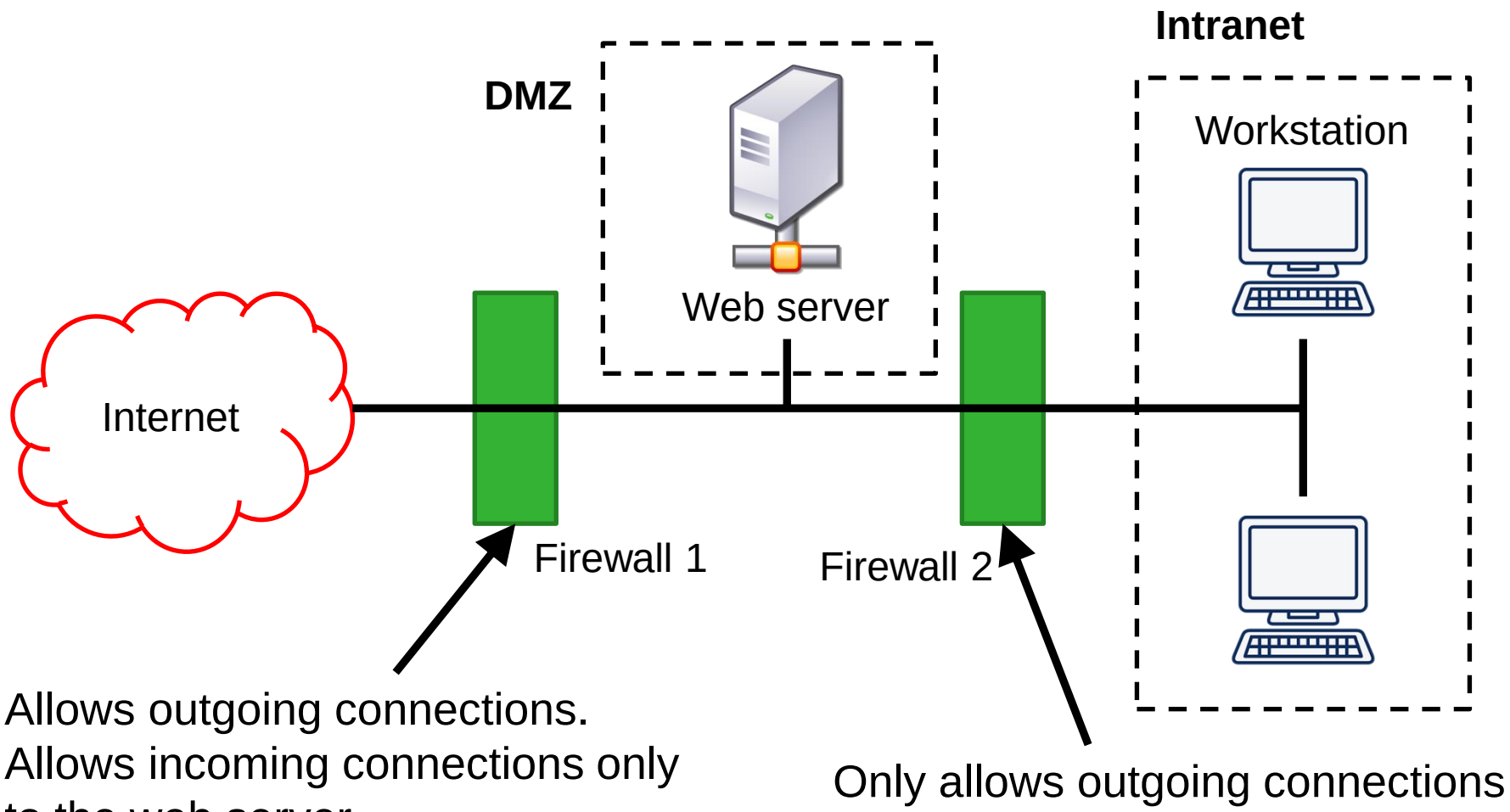
- Often, a company also needs to be reachable from outside
- Example: Company web server



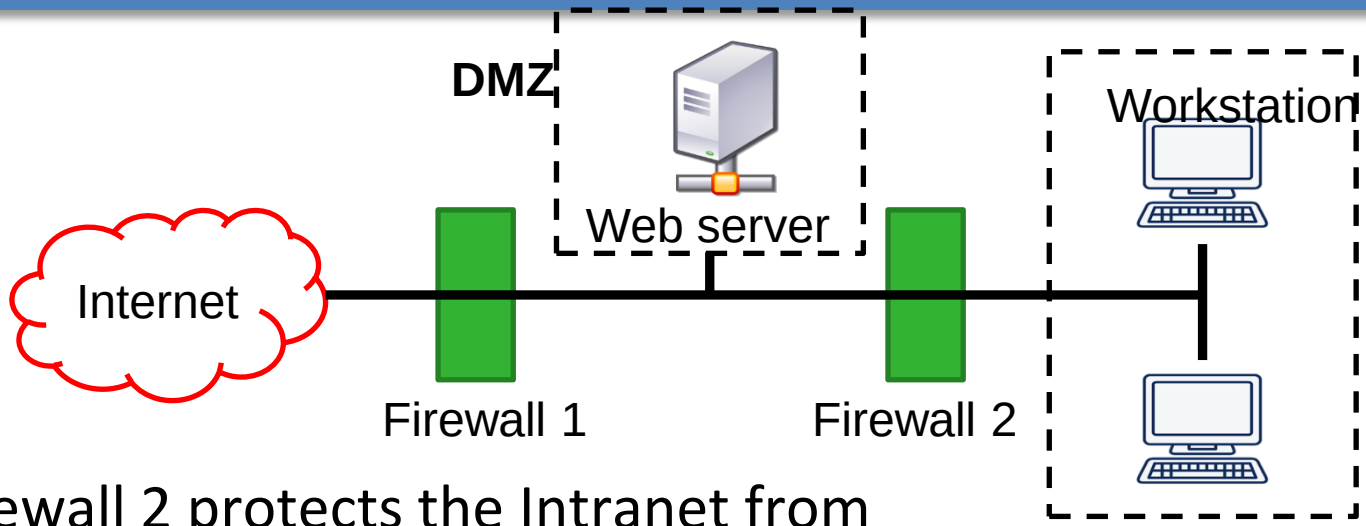
- What happens if an attacker manages to infect the web server?

Demilitarized Zone (DMZ)

- With a DMZ, we accept the fact that hosts exposed to the Internet are less trustable



DMZ Attacks

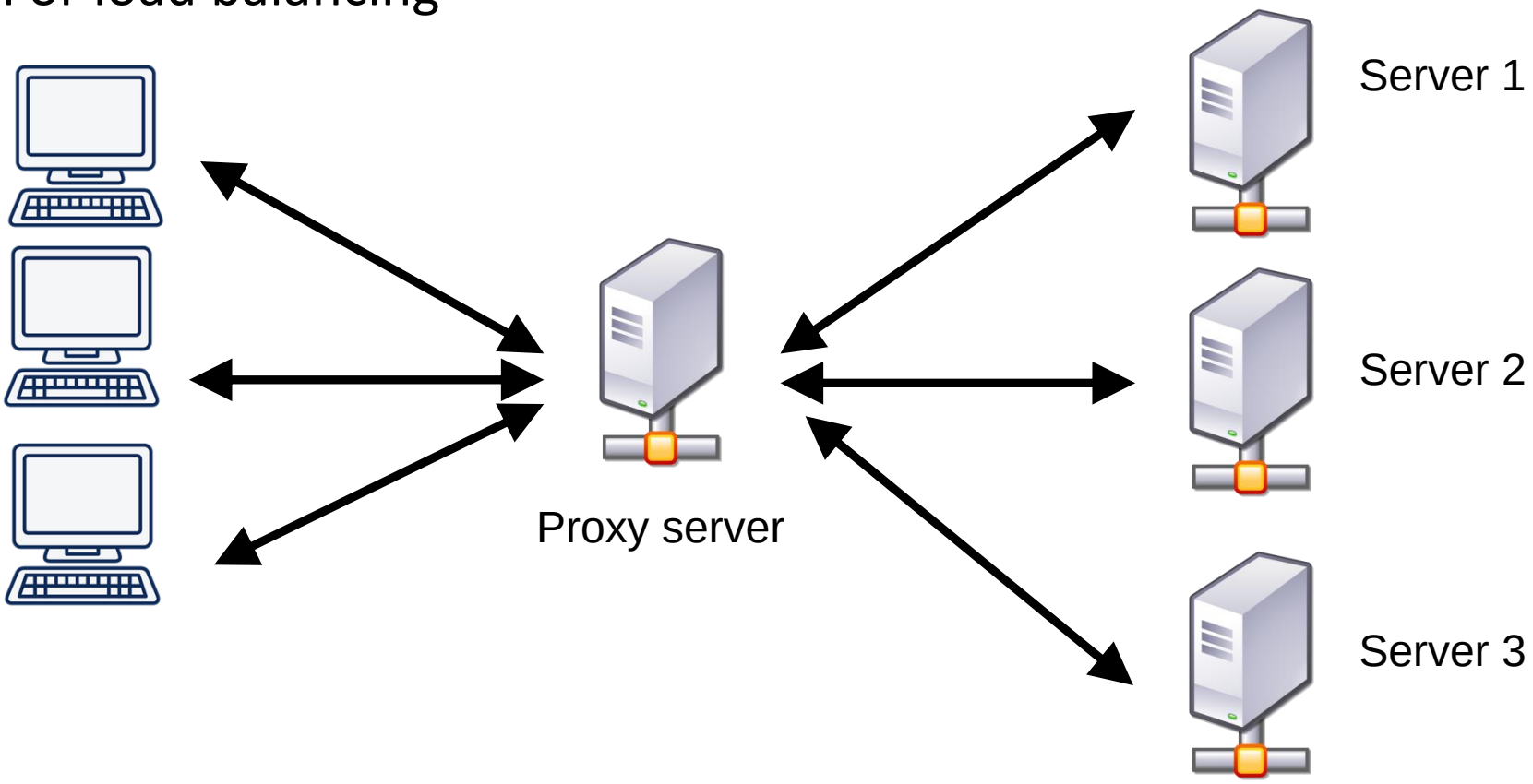


- Firewall 2 protects the Intranet from
  - attacks from the Internet
  - attacks from compromised servers in the DMZ
- Still some attack possibilities:
  - Attacker compromises the firewall 2
  - Through connections from the Intranet to the servers (admin tools etc.)
  - Sometimes, servers in the DMZ have connections to the Intranet, for example to a database

Proxy Firewalls

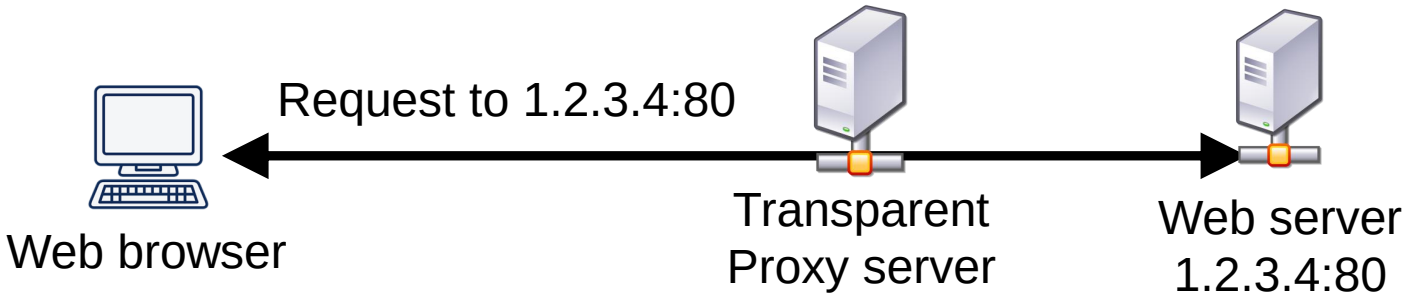
Proxies

- Proxies are useful in many situations
  - As caches (reduce response time, reduce network traffic to origin server,...)
  - For load balancing

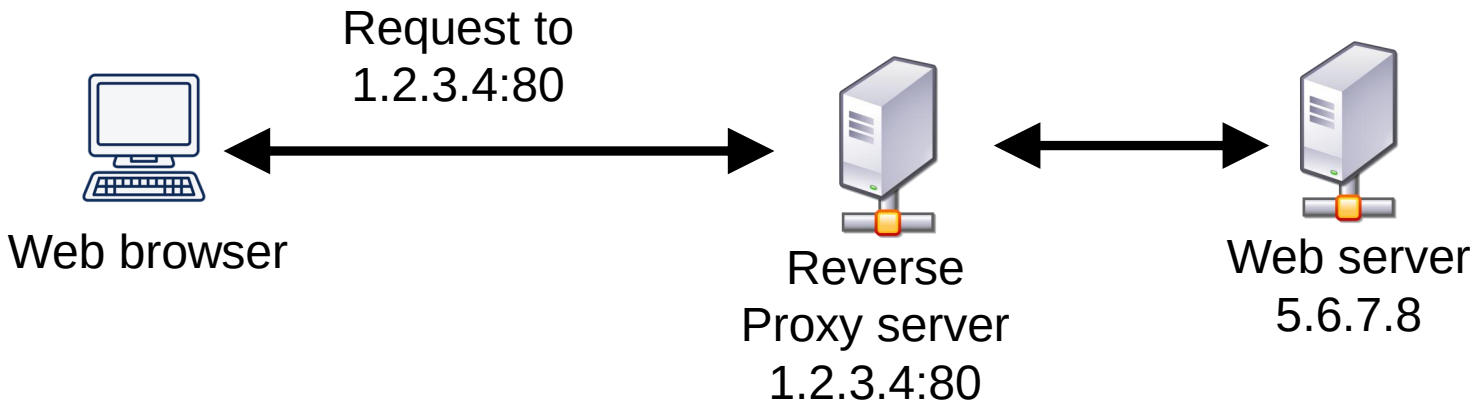


Proxy types

- *Transparent proxies* intercept queries to the original web server
  - Client is not aware that it is not talking to the original server

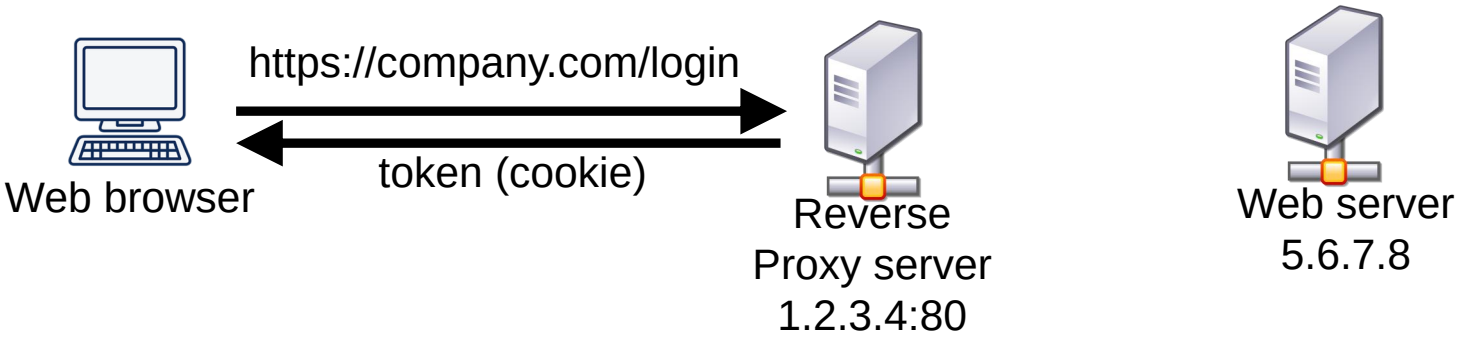


- *Reverse proxies* are servers that hide the origin servers from the client
  - Client has no knowledge of the origin servers exist



Proxy firewalls

- Proxies can operate as application firewalls
  - Analyze and filter (drop) malicious requests
  - For example, looking for SQL injection attempts
- Reverse proxies can handle authentication of clients
  - Cannot attack the origin servers unless authenticated



- Reverse proxies are not necessarily operated by the same company as the origin servers
  - Example: DDoS protection services by Cloudflare, etc.

Lessons learned (Not only for firewalls!)

- **Deny by default**: Deny what you have not explicitly allowed
- **Principle of least privileges:** Only allow what is absolutely needed to perform a job (and not more)

  Examples:
  - Allow only the traffic that is needed
  - Normal users do not need admin rights
  - …
- **Choke point:** It is easier to implement security if all data has to go through one point
- **Defense in depth:** Deploy multiple defense systems
  - Requires more attack skills
  - A single vulnerability in one system does not endangers the entire system
- **Zoning:** Establish trust zones (but dangerous if attack comes from inside)