Cyber 4-5-6

Semplici comandi docker

- Docker gestisce: network, images, container, volumes
  - Lista le immagini presenti nel tuo repository docker
    - docker images
    - docker image rm <id immagine>
    - eventualmente aggiungere -f nel caso in cui docker non volesse cancellare
    - docker image rm –f <id immagine>
  - Lista i container
    - docker container ls
    - docker container ls -a
    - docker container rm <id/nome>
    - eventualmente aggiungere -f nel caso in cui docker non volesse cancellare
    - docker container rm –f <id/nome>

Semplici comandi docker

- Lista tutti i container in esecuzione
  - docker ps
- Stop un container
  - docker stop <id del container>
  - docker kill <id del container>
  - Lo stop ha un timeout come parametro
- Riavviare un container
  - docker restart <id/name>
- Per i log
  - docker logs -f <id/name>

Parametri della riga di comando

- docker run
  - -i interattivo
  - -d in background
  - -t terminale con supporto escape sequences
  - -p <ip:porta host>:<porta guest>
  - -v <file/folder path host>:<file/folder path guest>
- Al termine della riga va messo il nome dell’immagine che si intende eseguire
- Dopo il nome dell’immagine potete inserire in nome di un comando da eseguire non appena l’immagine/container è partito

Parametri della riga di comando

- Elenco degli ip della mia macchina
  - ip a | grep inet
- Apre un terminale in un container in esecuzione
  - docker exec -it <id/nome> /bin/bash -i

Semplici comandi docker (kali linux)

- docker run -it --net host --name my-kali kalilinux/kali-last-release /bin/bash -i
  - Scarica e esegue kali linux
  - Al termine per uscire
    - exit
    - Ctrl-d
- docker start -ia my-kali

Due siti web

- In una cartella <esercizi> abbiamo due sottocartelle
  - immagini
    - Con alcuni file di immagine
  - suoni
    - Con alcuni file mp3 o altri formati
- Apriamo due terminali nella cartella <esercizi> ed eseguiamo, rispettivamente per ogni terminale
  - `docker run --rm -it -p 9000:80 -v ./suoni/:/usr/local/apache2/htdocs  httpd:latest`
  - `docker run --rm -it -p 9001:80 -v ./immagini/:/usr/local/apache2/htdocs  httpd:latest`
  - `docker run --rm -it -v ./:/copiare httpd:latest /bin/bash -i`
  - `cp -a conf /copiare/ #in questo modo copiamo la cartella conf nella nostra cartella corrente di lavoro`
  - `Uscire da docker e eseguire: sudo chown -R its:its conf`

- Discover

- Ho una macchina esposta su internet all’indirizzo: 35.157.221.134
- Ho anche accesso diretto a tale macchina in ssh
- Question?
  - Quali porte di questa macchina sono accessibili da internet?
  - Le porte internet su quali porte interne si mappano? (c’è qualche natting?)
  - NB: ho anche accesso tramite AWS alla macchina
- Prova n. 1: nmap 35.157.221.134
- Ma il mio openvpn utilizza la porta 4443
- Prova n. 1: nmap 35.157.221.134 -p 4443

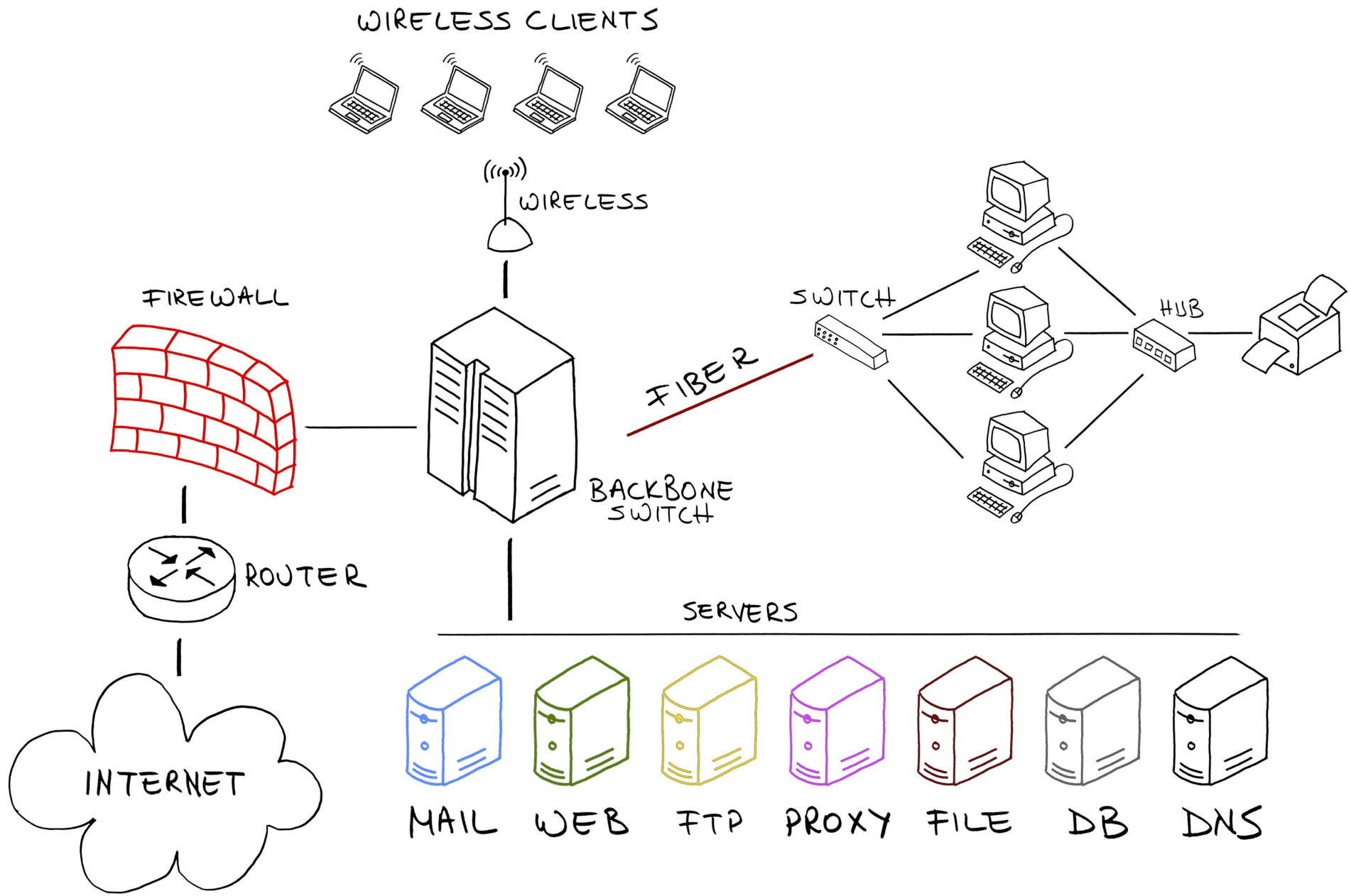
- Risultato di nmap

- PORT STATE SERVICE
- 53/tcp open domain
- 80/tcp open http
- 443/tcp open https
    - Come posso verificare che tali porte siano mappate sulle stesse porte interne alla macchina
- Ricordate che una macchina esposta su internet (in cloud) ha davanti un firewall a altri apparati di rete che riportano le connessioni internet all'interno della macchina stessa ma sulle porte che sono configurate nei firewall a nei Nat.
- Per elencare le comunicazioni attive
    - sudo netstat -anp –tcp | grep LISTEN

- # Lanciare un Web Server
- Per scoprire gli ip attivi sulla nostra macchina
  - Ip a | grep inet
- Per verificare installazione
  - Docker ps
- Per lanciare un web server
  - docker run --rm -it --net host httpd:latest
  - docker run --rm -it -p 8888:80 httpd:latest
- Per sapere i processi httpd attivi
  - ps -ef | grep httpd
- Per eseguire all’interno di docker
  - docker exec -it 4f20a590fba9 /bin/bash -i

Lanciare due Web Server e un Reverse Proxy

- Considerato che dovete mettere in esecuzione tre server TCP, dovete trovare tre porte TCP aperte.
  - sudo netstat -anp --tcp | grep LISTEN


WIRELESS CLIENTS
WIRELESS
FIREWALL
SWITCH
HUB
FIBER
BACKBONE
SWITCH
ROUTER
SERVERS
INTERNET
MAIL
WEB
FTP
PROXY
FILE
DB
DNS

Configurazione di Rete (IP Virtuali e DNS)

- Dobbiamo simulare tre macchine diverse sulla stessa macchina fisica (o VM).
  - Useremo l'IP Aliasing.
- Supponiamo che l’interfaccia di rete principale sia eth0.
  - Aggiungere gli IP secondari (temporanei)
  - sudo ip addr add 192.168.2.101/24 dev eth0 # WEB 0
  - sudo ip addr add 192.168.2.102/24 dev eth0 # WEB 1
  - sudo ip addr add 192.168.2.103/24 dev eth0 # Reverse proxy
- Configurare il DNS locale (/etc/hosts)
  - Modifica /etc/hosts per associare nomi a questi IP.
  - 192.168.2.101 proxy
  - 192.168.2.102 site1.backend
  - 192.168.2.103 site2.backend
- Creare tre cartelle
  - mkdir proxy backend1 backend2
  - In backend1 copiare la cartella immagini
  - In backend2 copiare la cartella suoni

Le pagine <dinamiche> del server

- Index.php

```php
<?php
$start = microtime(true);
// Simuliamo un carico CPU: calcolo hash ripetuto

$hash = "start";
//sleep(1);
for($i = 0; $i < 100000; $i++) {
   $hash = hash('sha256', $hash . $i);
}

$end = microtime(true);
$time = $end - $start;

echo "<h1>Benvenuto su Backend</h1>";
echo "<p>Pagina generata in " . round($time, 4) . " secondi.</p>";
echo "<p>IP Server: " . $_SERVER['SERVER_ADDR'] . "</p>";
?>
```

Attivare il server 1

- Copiare index.php in backend1
- In backend1 attivare
  - docker run --rm -it -p 192.168.2.102:80:80 -v ./:/var/www/html php:apache

Benchmark di un sito (che è anche D.O.S.)

- Una pagina statica
  - ab -n 1 -c 1 http://site1.backend/immagini/cool-4k-ultra-hd-laj5ot1mxghh0m4w.webp
- Il listato di una folder
  - ab -n 1 -c 1 http://site1.backend/immagini
- Una pagina dinamica
  - ab -n 1 -c 1 http://site1.backend/index.php
- Ricorda, per vedere pacchetti e tempi
  - sudo tcpdump -i any -n host 192.168.2.102

Come migliorare il servizio?

- Molte soluzioni ma tutte essenzialmente basate sul concetto di “load balancer”, sia fatto con apache-httpd, con docker (swarm), con Kubernetes (lo standard de facto)
- Noi lo faremo con apache-httpd

Attivare il server 2

- Copiare index.php in backend2
- In backend2 attivare
  - docker run --rm -it -p 192.168.2.103:80:80 -v ./:/var/www/html php:apache

Attivare il reverse proxy/load balancer

- Copiare la configurazione http nella cartella del proxy
  - Entrare nella cartella proxy(cd proxy) ed eseguire:
    - docker run --rm -v ./:/vapps httpd:latest cp -a conf /vapps/
    - Cambiare proprietario della cartella conf
      - sudo chown -R $USER:$USER conf
  - Modificare come segue

La configurazione del proxy

- Nel file httpd.conf, decommentare (togliere #) le righe LoadModule per:
  - mod_proxy.so
  - mod_proxy_http.so
  - mod_proxy_balancer.so
  - mod_lbmethod_byrequests.so
  - mod_slotmem_shm.so
- Decommentare la riga che include extra/httpd-vhosts.conf

La configurazione del proxy

Nel file conf/extra/httpd-vhosts.conf, inserire:

```
<VirtualHost *:80>
    ServerName proxy
    ProxyPreserveHost On
    ProxyRequests Off
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>
    <Proxy balancer://mycluster>
        # Nota: Qui usiamo gli IP VIRTUALI dell'host, perché i container
        # sono raggiungibili tramite quegli IP.
        BalancerMember http://192.168.2.102:80
        BalancerMember http://192.168.2.103:80
    </Proxy>
    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/
</VirtualHost>
```

Mettere in esecuzione in proxy

- Nella cartella proxy, eseguire:
  - docker run -it -p 192.168.2.101:80:80 -v ./conf:/usr/local/apache2/conf httpd:latest

Analisi dei risultati

- Tramite ab, verificare se il carico cambia con o senza proxy
- Fornire spiegazioni in merito
- Verificare cosa accade quando si mettono in parallelo richieste di pagine statiche e di pagine dinamiche
  - Come sarà possibile misurare i carichi di lavoro e il funzionamento del sito?

• Un codice python per collegarsi a un sito web

- import requests

Define the URL

url = "http://127.0.0.1/api"

Define the JSON payload

data = {

"key1": "value1",

"key2": "value2"

}

- # Define headers, if needed

headers = {

"Content-Type": "application/json",

"Authorization": "Bearer your_token_here" # Optional, if authentication is required

}

- # Send the POST request
- response = requests.post(url, json=data, headers=headers)
- # Check the response
- if response.status_code == 200:
- print("Success:", response.json())
- else:
- print("Failed:", response.status_code, response.text)

- Configurazione del server (come esempio docker)
  - Per eseguire un web server apache, senza specifiche configurazioni, utilizzare
  - docker run --rm -it -p 8888:80 httpd:latest
  - Si noti il natting tra porta host 8888 e porta guest 80
  - Alcune considerazioni
    - potete aggiungere tanti –p quanti ne servono (per fare natting di altre porte)
    - Con il flag –v potete condividere una parte del file system host con il file system guest
      - In tal modo potete, ad esempio, avereuna configurazione disponibile direttamente sulla macchina host e farla utilizzare al server web
  - Nel nostro caso ci converrà creare una cartella <html> e in questa cartella eseguire docker, in tal modo i file html presenti nella cartella saranno accessibili via browser
  - Per entrare in un container e attivare la console bash
    - Docker ps (per individuare il container id)
    - Docker exec –it <container id> /bin/bash –i
  - E quindi per condividere htdocs
    - docker run --rm -it -p 8888:80 -v ./:/usr/local/apache2/htdocs httpd:latest

Progetto del 11 dicembre 2025

1. Posizionarsi in una cartella nella quale sarà eseguito il codice (es: esercizio11122025)
2. Copiare i due file, passati su discord e sul materiale didattico, generatore.sh e client.sh
3. Creare una sottocartella <html> (mkdir html)
4. Mandare in esecuzione
    1. bash generatore.sh
5. Creare un indirizzo IP virtuale (sostituire il nome della scheda di rete)
    1. sudo ip addr add 192.168.2.101/24 dev wlan0
6. Mettere in esecuzione apache-php
    1. docker run --rm -it -p 192.168.122.1:80:80 -v ./html/:/var/www/html php:apache

- Esercizio attacco HTTP a un Web Server

1. Individuare (comando NMAP) tutti gli IP collegati alla sottorete VPN
2. Scegliere 4 IP da attaccare (senza dirlo a nessuno)
3. Attaccare con ab o con un programma scritto in Python i servizi web dei 4 IP (http://10.8.0.NN/p1,html o p2.html o pagine non esistenti)
4. Limitare l'attacco a 1000000 di richieste al più
5. Se utilizzate ab ricordate che il flag –c consente di indicare la concorrenza. NB;: la concorrenza deve essere inferiore al numero di richieste. Ab –n 1000 –c 64 http://....
6. Ricordate che potete anche attaccare con una POST per aumentare il traffico
    1. Per un file di testo: ab –n xx- c yy –p <file di testo> http://.....

- Esercizio attacco HTTP a un Web Server

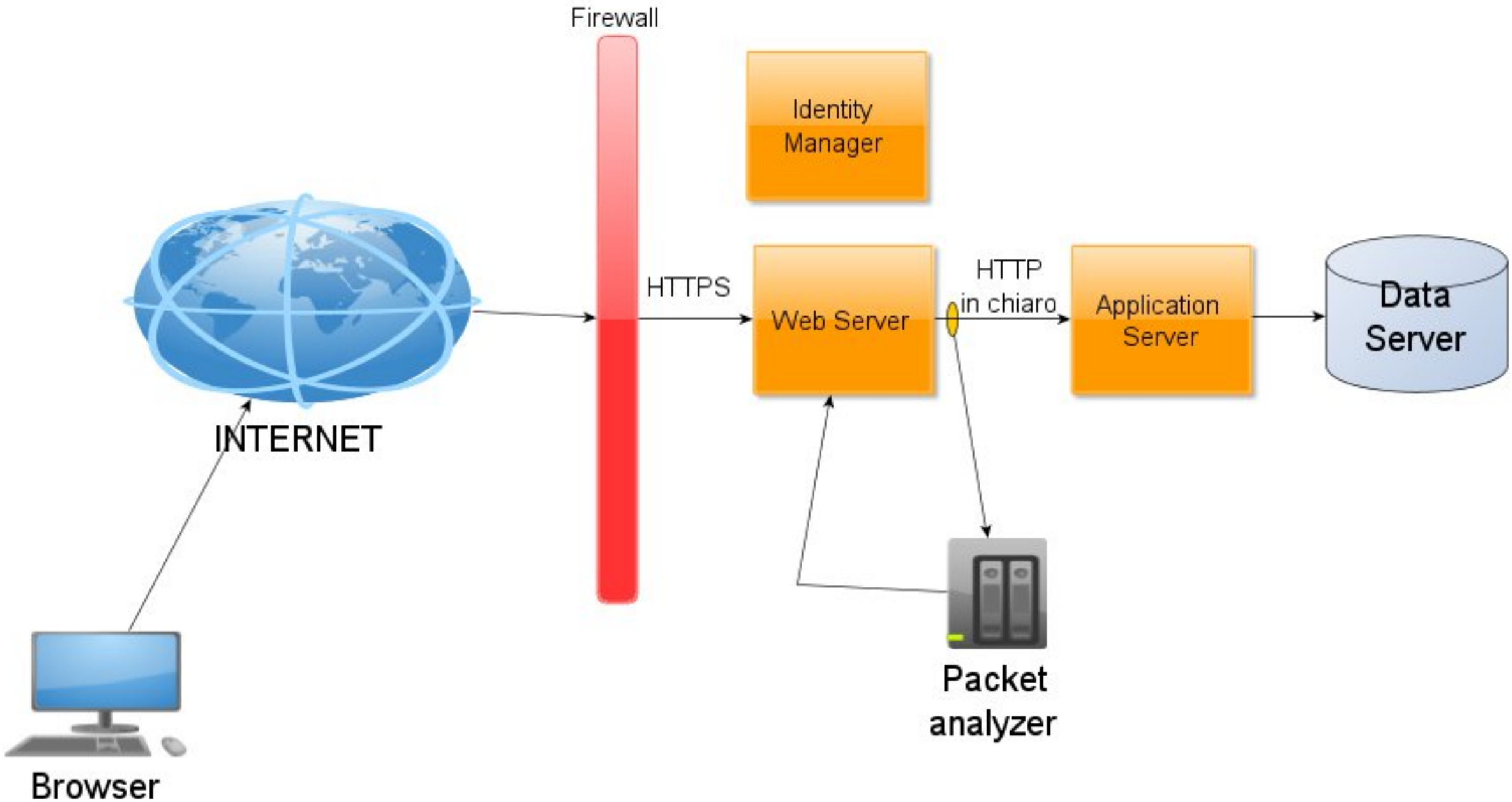
1. Individuare (comando NMAP) tutti gli IP collegati alla sottorete VPN
2. Scegliere 4 IP da attaccare (senza dirlo a nessuno)
3. Attaccare con ab o con un programma scritto in Python i servizi web dei 4 IP (http://10.8.0.NN/p1,html o p2.html o pagine non esistenti)
4. Limitare l'attacco a 1000000 di richieste al più
5. Se utilizzate ab ricordate che il flag –c consente di indicare la concorrenza. NB;: la concorrenza deve essere inferiore al numero di richieste. Ab –n 1000 –c 64 http://....
6. Ricordate che potete anche attaccare con una POST per aumentare il traffico
    1. Per un file di testo: ab –n xx- c yy –p <file di testo> http://.....

- Esercizio attacco HTTP a un Web Server

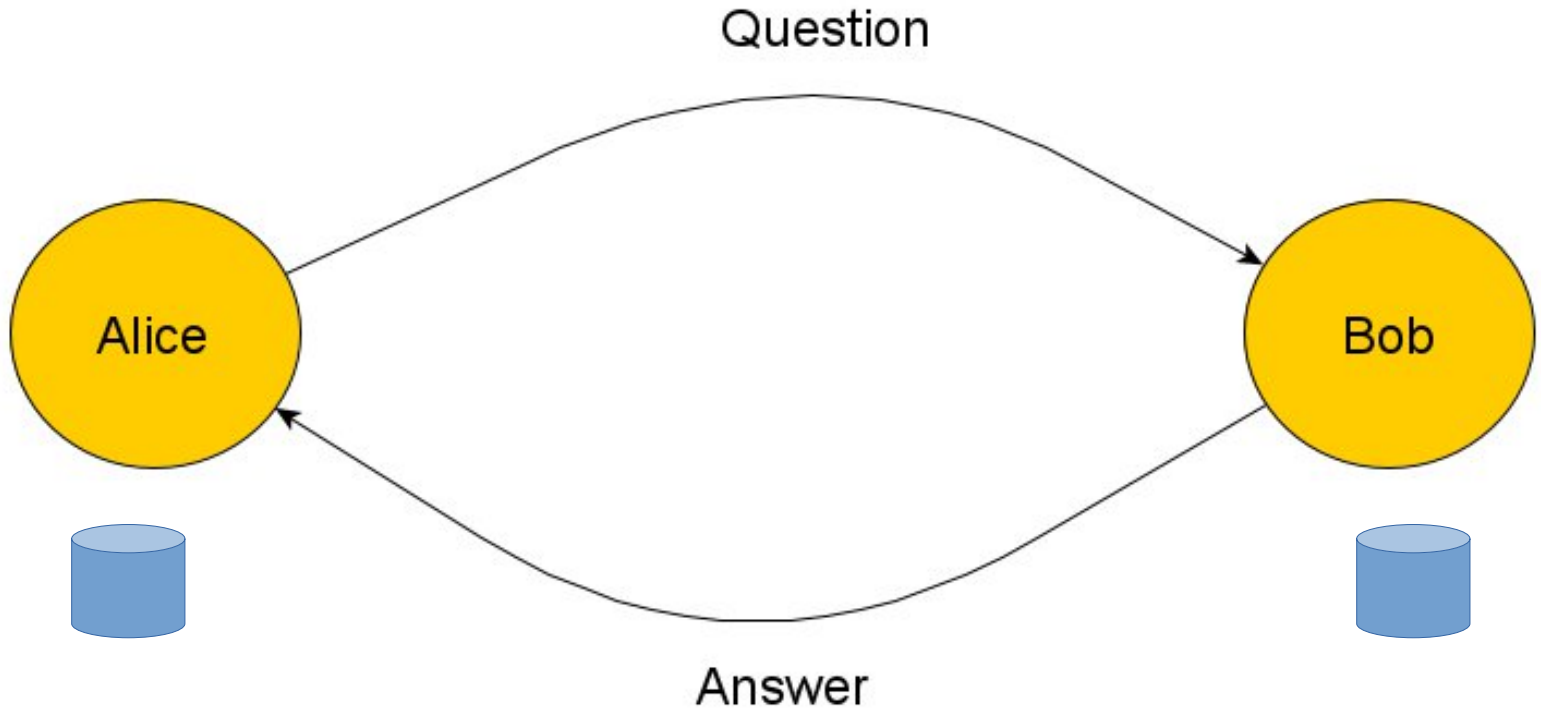
1. Mentre voi attaccate, sarete attaccati e quindi dovete
    1. Acquisire i dati di attacco
    2. Generare un report di attacco
2. Per acquisire i dati di attacco
    1. Sudo tcpdump –i tun0 –n –c <numero di pacchetti da acquisire> port 8888 >filedump
3. Poi, con python o altri strumenti/linguaggi analizzate il filedump e generate un report che contiene
    1. Ip sorgente – ip destinazione(il vostro ip) numero di byte inviati, numero di byte ricevuti
    2. Qualunque altro report vi venisse in mente (aggiuntivo a questo) è il benvenuto

- Esempio di codice bash per analisi del traffico

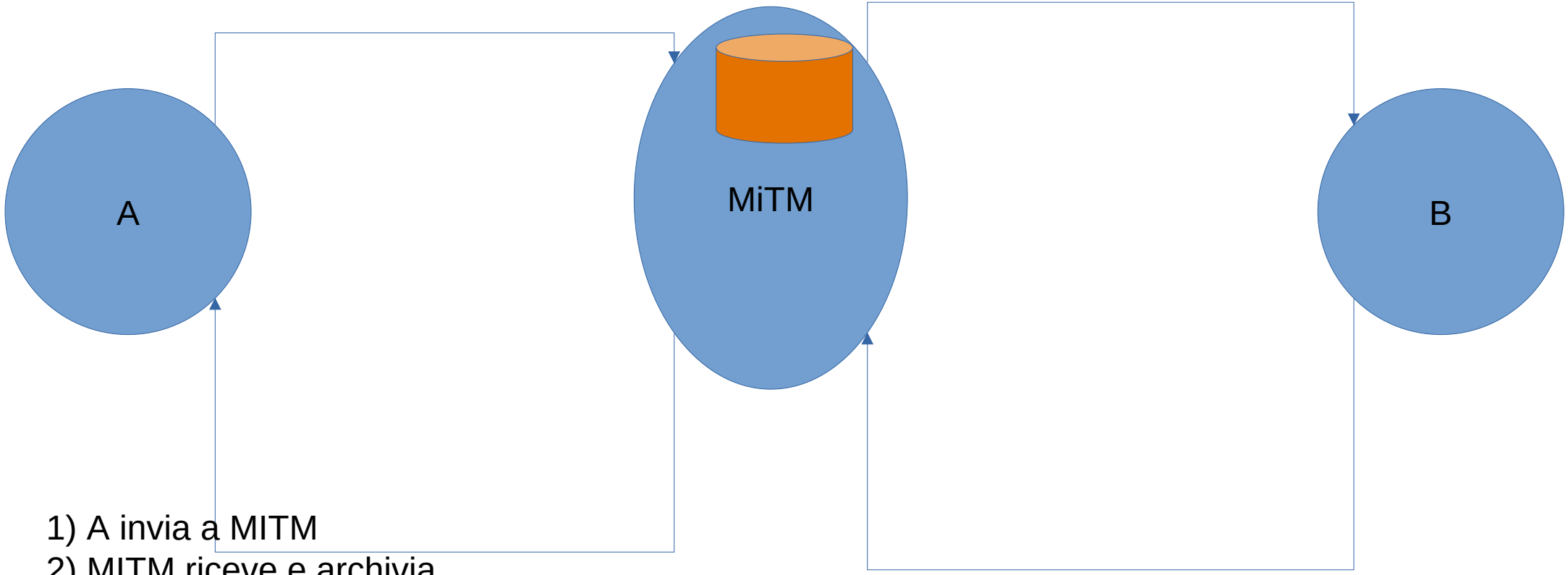
- cat attacco*.dmp | egrep -o "[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+.*$"|sed -E 's/: Flags .*length//'|grep -v " 0"|sed -E 's/([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)\.[0-9]+ > ([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)\.[0-9]+ ([0-9]+)/\1 \2 \3/'|sort|awk '{if ($1 < $2) {key=$1 " " $2;} else {key=$2 " " $1;} sum[key] += $3} END {for (c in sum) print c, "total length", sum[c]}'|sort -k5,5 -n -r


Firewall
Identity Manager
INTERNET
HTTPS
Web Server
HTTP in chiaro
Application Server
Data Server
Packet analyzer
Browser

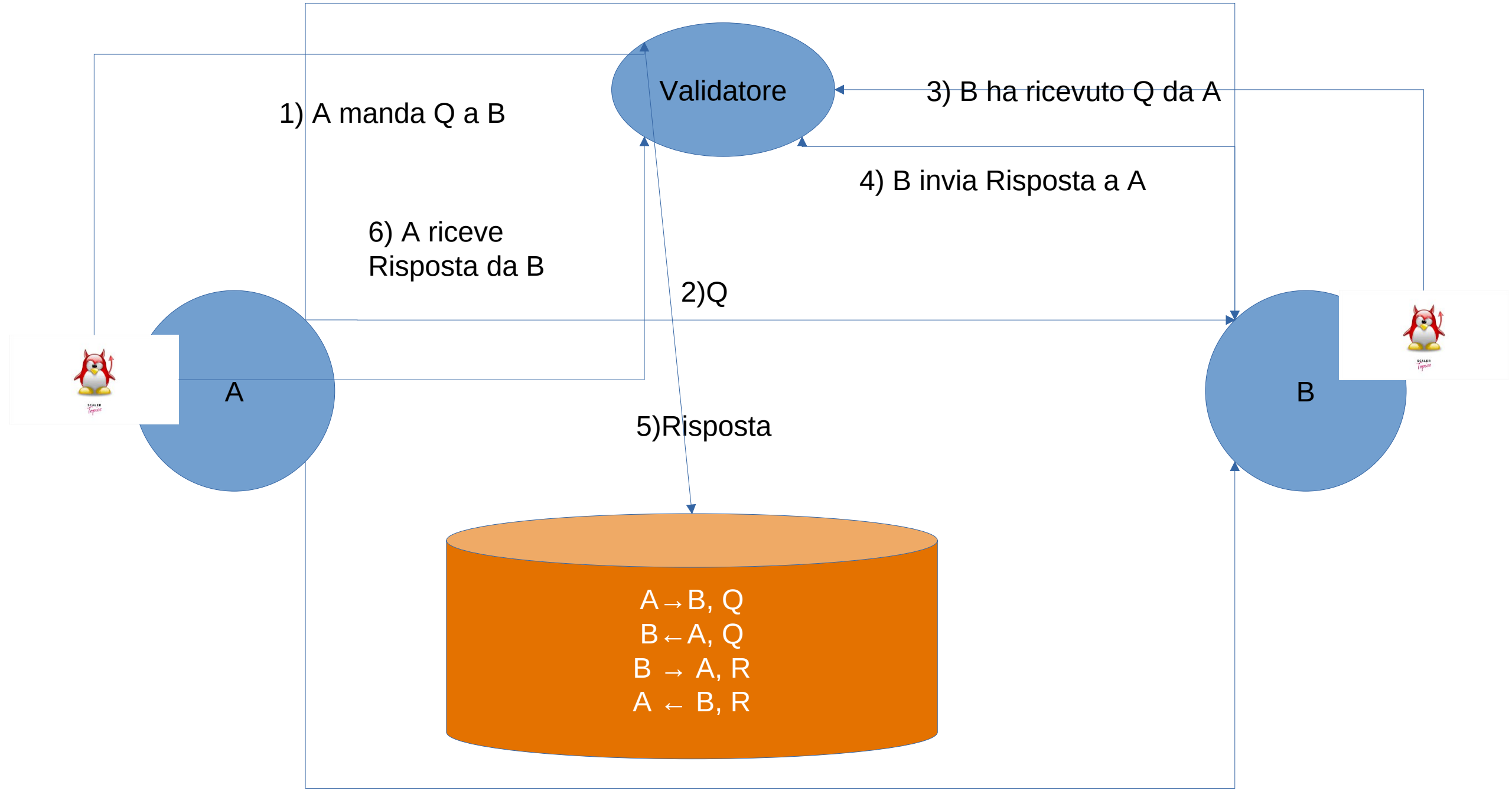
- Il modello di sicurezza della transazione elementare

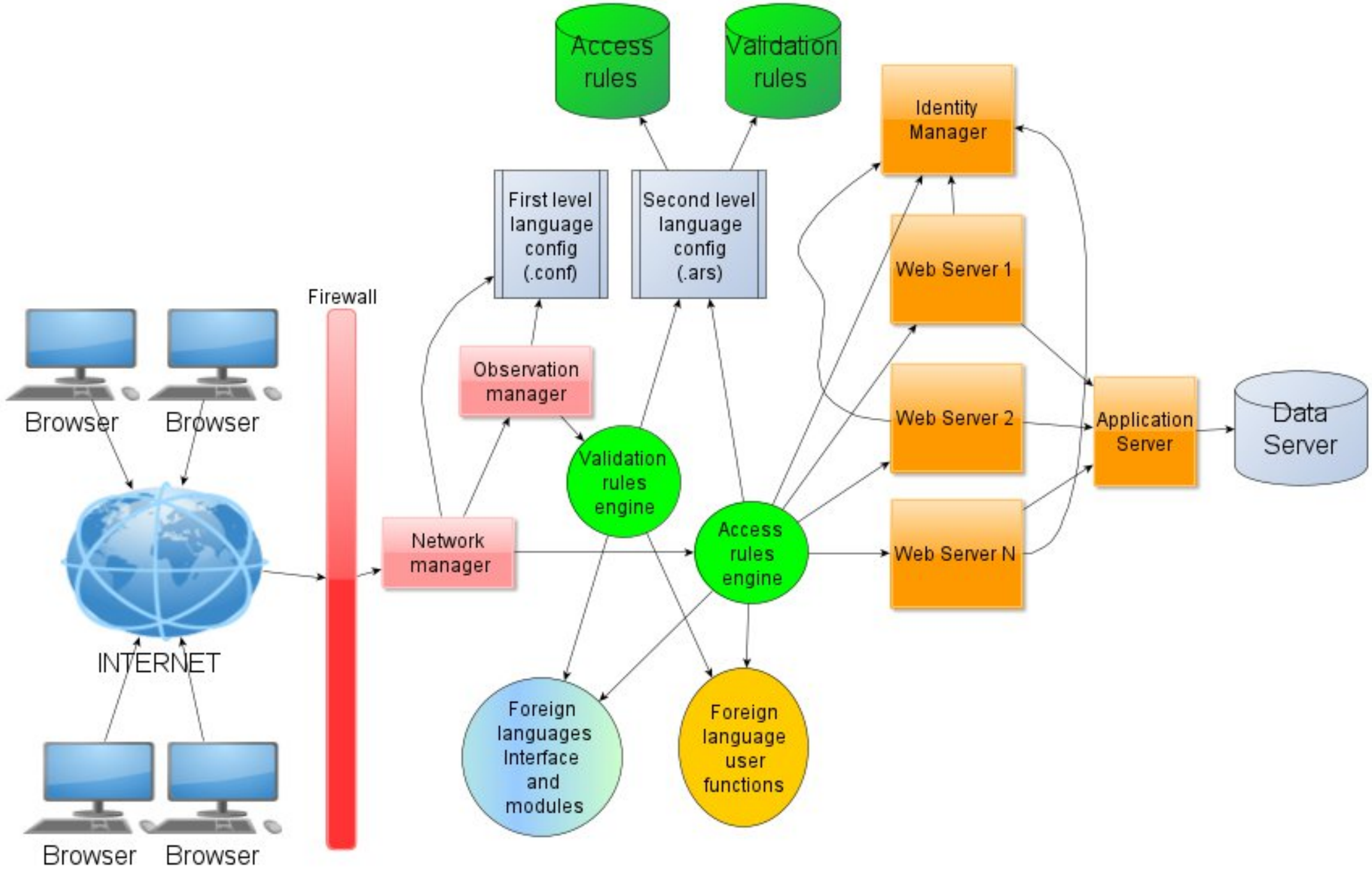


A chiede una cosa a B
A non riceve risposta da B
Cosa può fare A?



1) A invia a MITM
2) MITM riceve e archivia
3) MITM invia a B
4) MITM archivia che ha inviato a B
5) B risponde a MITM
6) MITM riceve e archivia
7) MITM invia a A
8) MITM archivia che ha inviato a A


Validatore
1) A manda Q a B
3) B ha ricevuto Q da A
4) B invia Risposta a A
6) A riceve
Risposta da B
2)Q
A
B
5)Risposta
A → B, Q
B ← A, Q
B → A, R
A ← B, R


Access rules
Validation rules
Identity Manager
First level language config (.conf)
Second level language config (.ars)
Web Server 1
Firewall
Observation manager
Web Server 2
Application Server
Data Server
Validation rules engine
Browser
Browser
Network manager
Access rules engine
Web Server N
INTERNET
Foreign languages Interface and modules
Foreign language user functions
Browser
Browser

Un primo esempio di utilizzo

- Sia dato un sistema di servizi WEB.
- Costruire un insieme di regole che:
  - per tutti gli indirizzi IP del server e per la porta 8086 consentano l'accesso alle pagine del sottoalbero /sito01/ del server mioserver:8088
  - per tutti gli indirizzi IP del server e per la porta 8087 consentano l'accesso alle pagine del sottoalbero /sito02 del server mioserver:8089
- In via temporanea, in attesa del completamento del sito per tutte le richieste su pagine differenti rispondere con il messaggo "sito in allestimento"

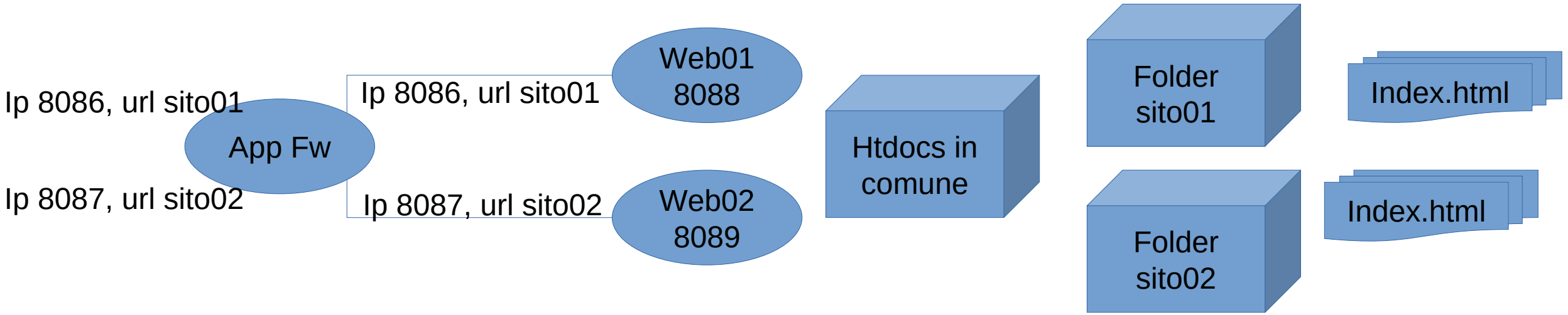
Per prima cosa realizziamo i server

- Creiamo due cartelle:
  - sito01
  - sito02
- Nelle due cartelle mettiamo almeno un index.html
  - Che mostra sito01 o sito02 (una scritta)
- E poi eseguiamo in due terminali differenti, nella cartella madre delle cartelle sito01 e sito02
  - docker run --rm -it -p 8088:80 -v "$PWD":/usr/local/apache2/htdocs httpd:latest
  - docker run --rm -it -p 8089:80 -v "$PWD":/usr/local/apache2/htdocs

Problemi di sicurezza di questa doppia configurazione

- Se non si scrive sito01 o sito02 nella url, il web server restituisce la lista dei file
- Se si va su
  - http://192.168.122.1:8088/sito01
- oppure su
  - http://192.168.122.1:8089/sito01
- I server web restituiscono la pagina corretta. Non sono quindi in grado di distinguere tra porta e sito web

Le porte che l’utente dovrà utilizzare sono la 8086 e 8087!!!!



NB: quando lanciate un’applicazione docker con il flag -p (le porte) allora se l’applicazione dentro docker cerca di andare verso la 127.0.0.1 resta dentro docker poiché le due macchine host (la vostra) e guest (il doker) non condividono la 127.0.0.1
L’unico modo per condividere la 127.0.0.1 è lanciare docker con il flag –network=host ma in questo caso tutte le porte utilizzate all’interno dell’applicazione docker sono condivise con l’host e questo non è possibile se lancio due server uguali, come in questo caso.
Inoltre in Windows e credo anche in Mac il flag –network non è implementato

Il file ARS

```
DEFINE ipset caso01_ip = {*.*.*.*:8086};
DEFINE ipset caso02_ip = {*.*.*.*:8087};
DEFINE ipset tuttigli_ip = {*.*.*.*:*};

DEFINE urlset caso01_url = { /sito01, /sito01*, /sito01/*/ }; // */
DEFINE urlset caso02_url = { /sito02, /sito02*, /sito02/*/ }; // */

DEFINE AR "Caso01"
    CONDITION
      http.url is in caso01_url
      net.ipdst is in caso01_ip
    ACTION
       TCP.REDIRECT "192.168.122.1:8088"
    ;

DEFINE AR "Caso02"
    CONDITION
      http.url is in caso02_url
      net.ipdst is in caso02_ip
    ACTION
       TCP.REDIRECT "192.168.122.1:8089"
    ;

DEFINE AR "Il Resto"
    CONDITION
      net.ipdst is in tuttigli_ip
    ACTION
       ANSWER "<h1>Sito in allestimento</h1>"
    ;
```

Come eseguire l’App firewall

- docker run -ti -p 8086:30002 -p 8087:30002 --volume "$PWD":/vapps dozenapps/frextva:latest bash -c "cd /vapps; /va cyber-01.conf"

Cosa accade dentro docker?

- Docker run -it –rm …. -p 8888:80 app:latest
- Supponiamo che ip dell’host sia 192.168.1.10
- Dentro docker esiste una classe di indirizzi che è del container e nulla ha a che vedere con i nostri indirizzi dell’host

Fare sicurezza senza modificare i sistemi preesistenti

- Nell’esercizio svolto abbiamo appreso come sia possibile rendere un sistema di servizi più sicuro senza dover modificare i software applicativi.
- La sicurezza è uno “strato/layer” ulteriore che si aggiunge agli strati/layer del sistema operativo
  - Nel nostro caso abbiamo aggiunto un Application Firewall che ci ha consentito di modificare il comportamento dei siti sito01 e sito02 senza modificare i due sistemi preesistenti

LE validation rule

- Ora aggiungiamo una regola (di tipo Validazione) che ci fa fare il log di tutto quello che invieremo ai siti WEB

```
DEFINE VR "log"
  CONDITION
     obs.event is net.send
  ACTION
                              REPORT pippo {
     CAT {
       "IP source: ", net.ipsrc, "\n",
       "IP dst: ", net.ipdst, "\n",
       "Host: ", http.host, ", ", http.method, ", ", http.uri, ", ",
OBS.TIME,
          "\n"
          }}
  ;
DEFINE VR "logback"
  CONDITION
     obs.event is net.recv
  ACTION
    REPORT pluto { CAT {"0: ", http.answer.code, " ",
    http.answer.data} }
;
```

Un semplice servizio WEB/API

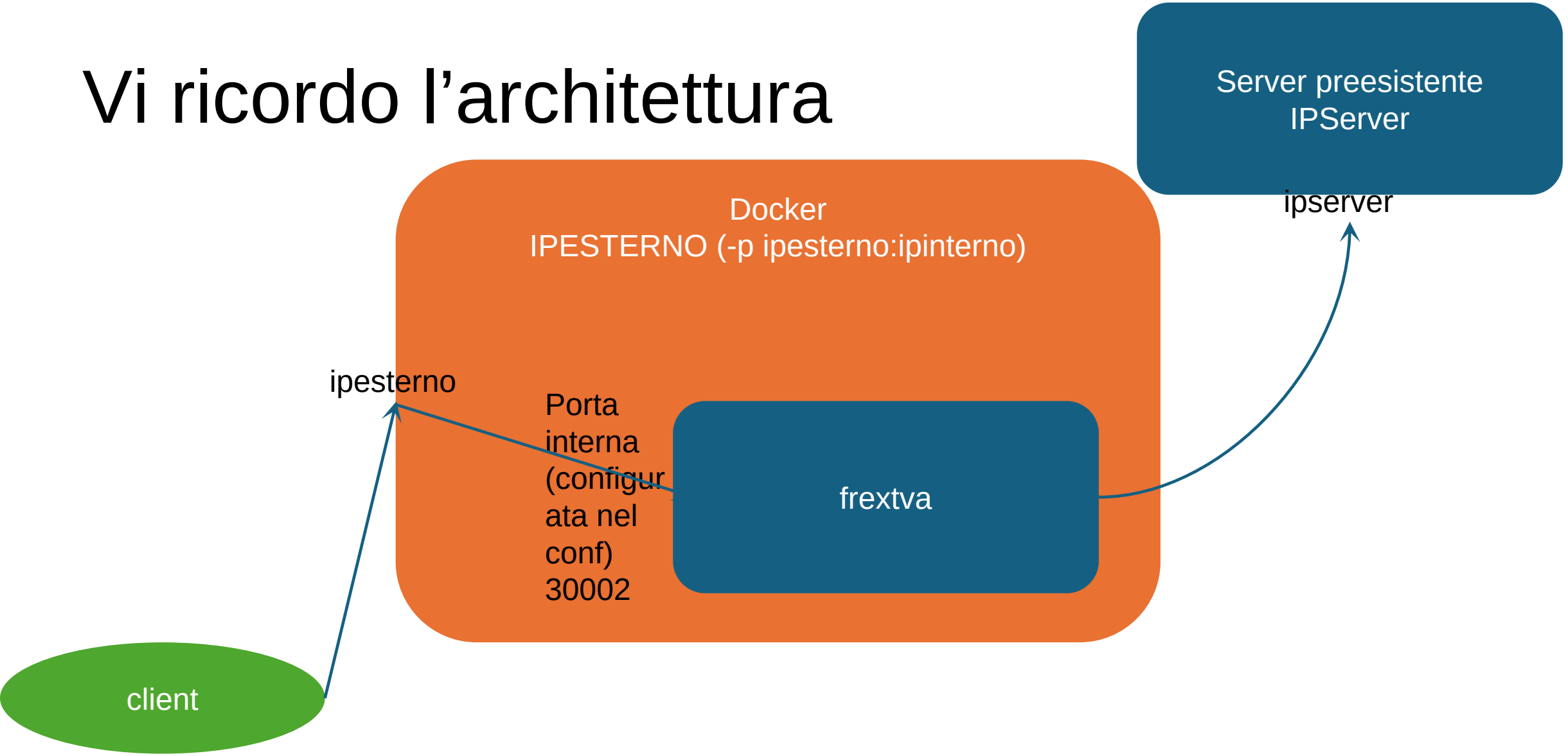
Utilizziamo express di node

```
/*
npm i express
npm i eventsource
*/

const http = require("http");
const fs = require("fs");
const express = require("express");
const app = express();

// Serve static files
app.use(express.static("./"));

// GET /api?add1=10&add2=20&username=user01
let add1 = req.query.add1;
let add2 = req.query.add2;
let user = req.query.username;
if (user === "user01") {
  res.json({ risposta: add1 + add2 });
} else {
  res.json({ risposta: add2 + add1 });
 }
});

let options = {};
```

Aggiunta sicurezza

1. Scrivere una regola di accesso che riceve sulla porta <yourIP>:8888 e inoltra la richiesta alla <yourIP>:3000 senza modificare i dati
2. Scrivere una regola di validazione che fa il log delle risposte
3. Scrivere una regola di validazione che fa il log delle richieste
4. Bloccate l’accesso all’utente arcieri, gli rispondete con http 401 Unauthorized

Vi ricordo l’architettura



Costruire una sequenza

Per come abbiamo realizzato le due regole di validazione per fare il logo delle API, queste generano due log differenti che, a posteriori, potremmo legare tramite il net.sesid che è identico tra i due log.

Supponiamo però di avere un modello nel quale dobbiamo verificare se una certa azione è stata compiuta successivamente a un’altra azione

- Prima di andare sulla pagina /privata.html, l’utente ha visitato la pagina /login.html
- Prima di andare alla pagina di pagamento, l’utente è andato sulla pagina di selezione dei prodotti
- Oppure prima di andare sulla pagina di accettazione di un progetto edilizio, il geometra comunale è andato a visitare la pagina del progetto edilizio
- …

Cioè la sicurezza di un sistema dipende anche dal fatto che gli utenti rispettino i workflow previsti dall’applicazione web

Facciamo un esempio

Utilizzando l’operatore next colleghiamo in un unico le richieste e le risposte alle nostre API

La next attiva la regola successiva solo se la regola precedente è stata attivata

MA

Le due regole potrebbero non condividere nulla

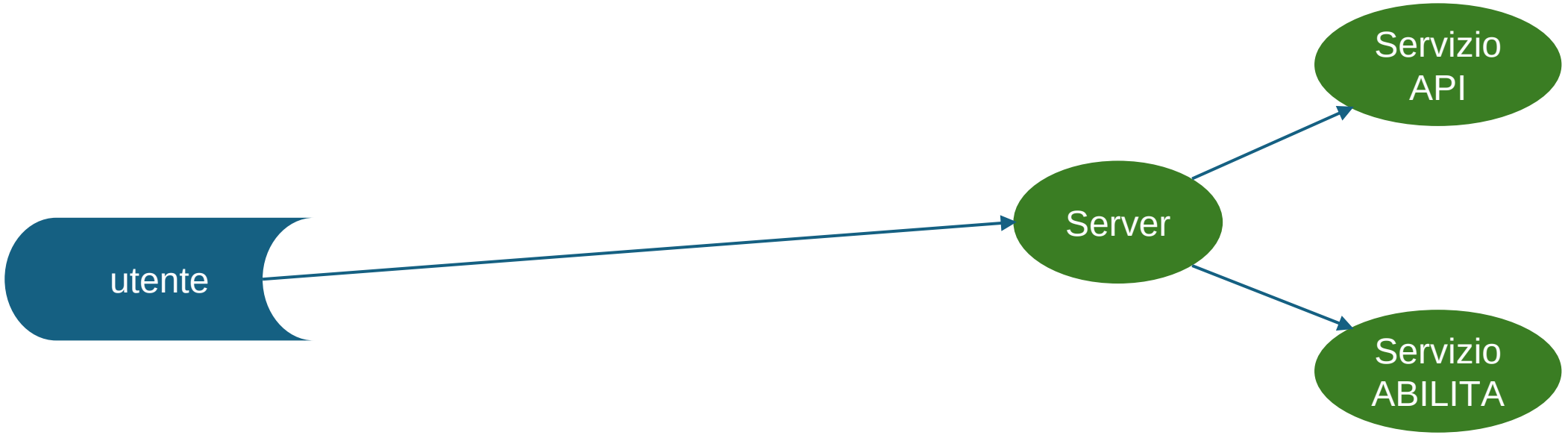
Se invece noi volessimo gestire una situazione in cui facciamo il log di una transazione completa, dobbiamo utilizzare anche l’operatore

VAR

Quindi

- Per poter tracciare un workflow possiamo utilizzare gli operatori
  - Next e var del linguaggio
- Un test
  - Supponiamo di voler consentire all’utente arcieri di accedere a un servizio (nel nostro caso ad una API)
    - Se e Solo Se
    - L’utente dimitri ha già effettuato una particolare operazione
    - Ad esempio: ha acceduto al servizio /abilita passandogli l’id arcieri

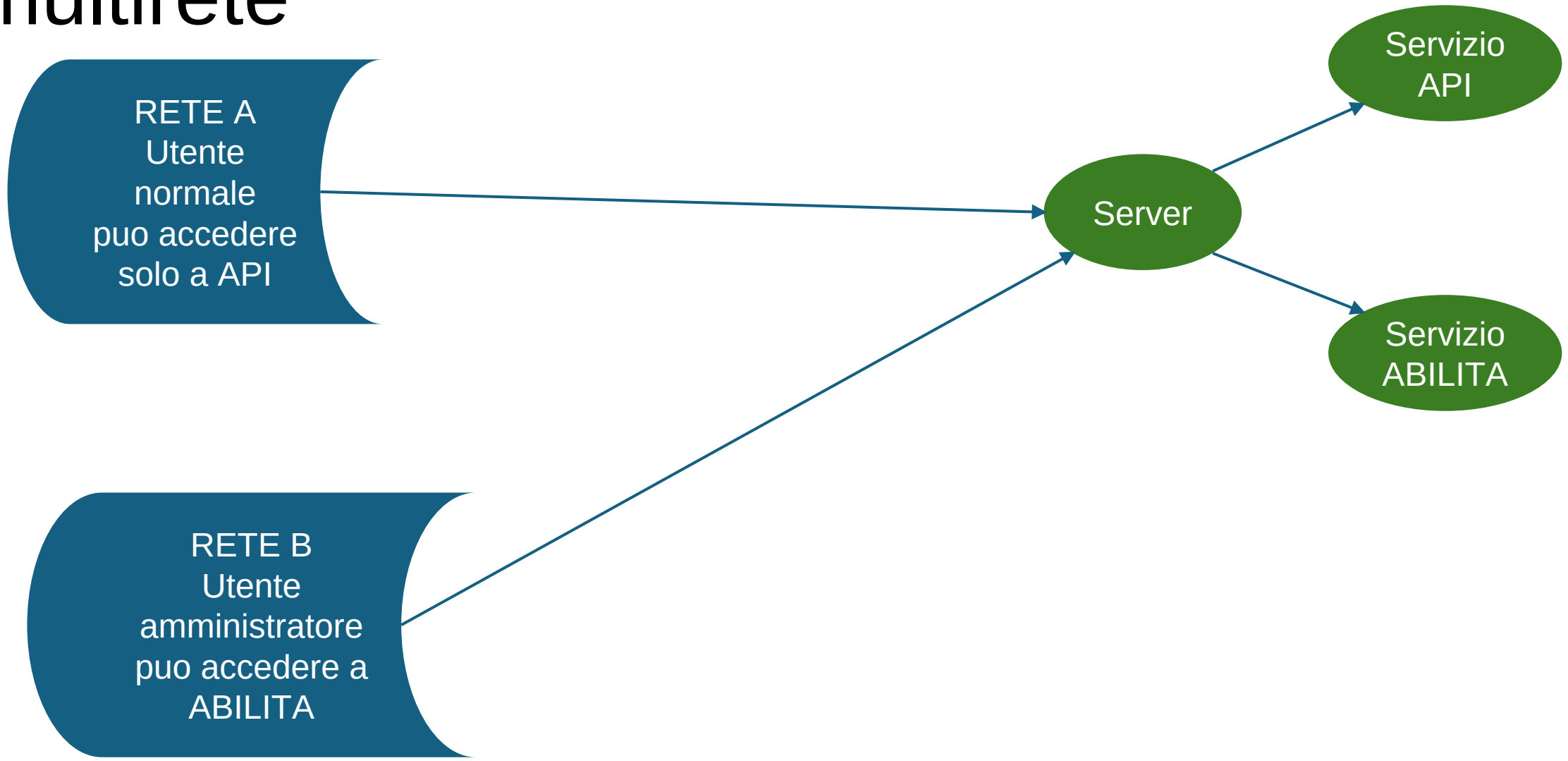
La nuova architettura di test



Qualsiasi utente può accedere a qualsiasi servizio, sia API ,sia ABILITA
Noi vogliamo aggiungere un livello di sicurezza che assicuri

- l'utente arcieri non può accedere al servizio API e al servizio ABILITA
- Chiunque, tranne arcieri, può accedere al servizio API
- Solo l'utente dimitri può accedere al servizio abilita
- Se l'utente dimitri accede al servizio abilita passandogli l'identificativo arcieri, allora arcieri potrà accedere al servizio API

La nuova architettura di test, un modello multirete



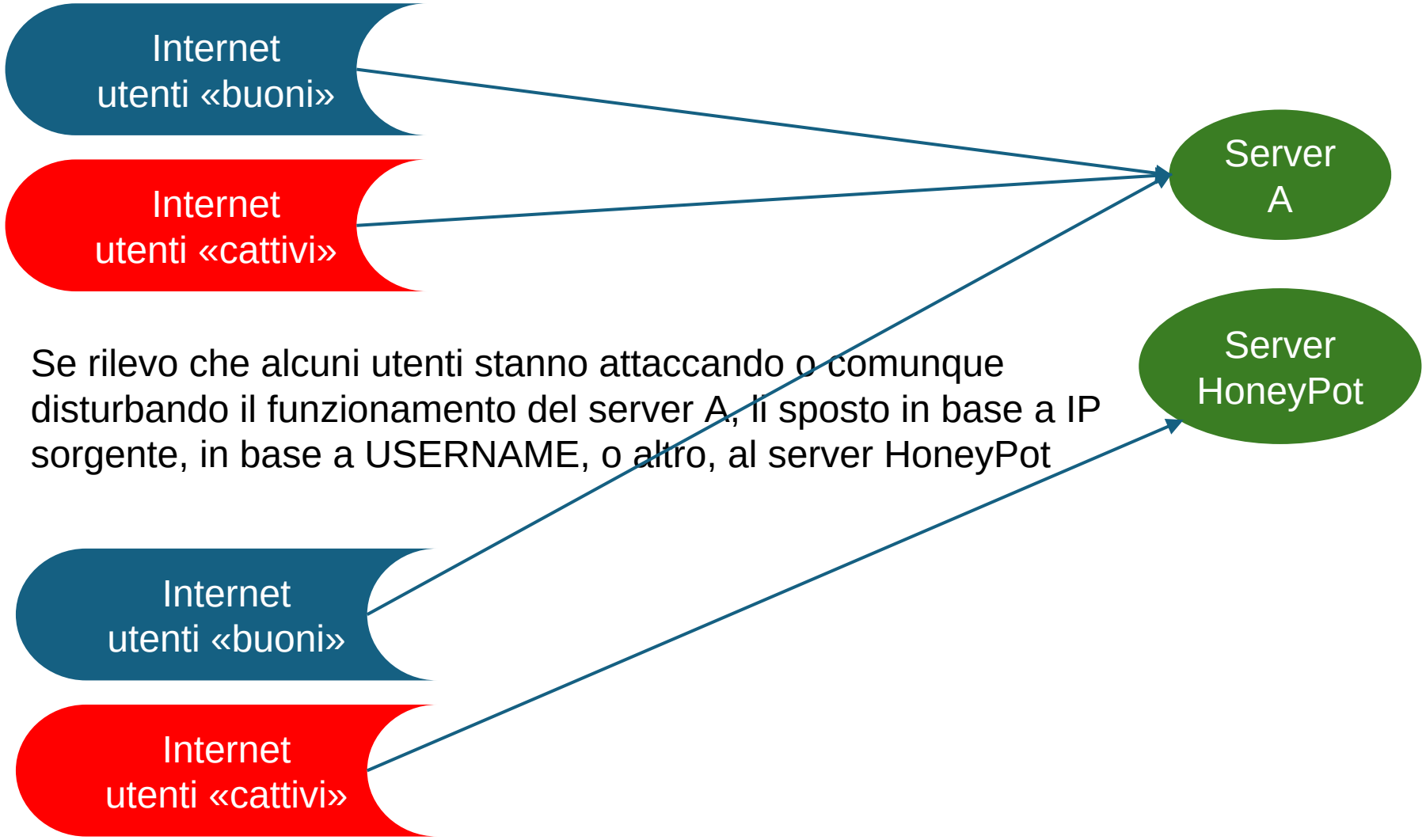
Inserimento e eliminazione di elementi in un set

- Tramite add e del potete costruire regole che inseriscono o eliminano elementi da un set
- Nota che in fondo alla istruzione potete aggiungere un timeout in secondi. Al termine del timeout l’operazione viene annullata

Problema

- Sia data un’architettura con due server A e B
- Tutti gli utenti normalmente vanno al server A (di produzione)
- Tramite analisi dei carichi di rete vi accorgete che un gruppo di utenti sta cercando di attaccare il server A
  - Tramite frextva dovete deviare questi utenti verso il server B che è la nostra HoneyPot

Lo switch su un honeypot



10 febbraio 2025
realizzazione di un sito web, attacchi sql injection, difesa

Realizziamo un sito web con autenticazione

- Il sito (nella cartella P20250210_01_WebSite) ha
  - una parte statica aperta
  - Una parte statica protetta da username e password
  - Una parte dinamica tramite cui eroga servizi
  - Per gestire il riconoscimento delle credenziali il sito utilizza un rdbms (sqlite3)
- Per semplicità sviluppiamo il sito utilizzando

Gli step

- sudo apt install python3-flask sqlite3 sqlite3-tools
- Creare file mysite.py
- Inserire righe
  - import sqlite3
  - import os

  - from flask import Flask, request, g, redirect, url_for, render_template, session, flash

Una breve digressione, la commit degli RDBMS

- Supponiamo che abbiate progettato un DataBase e avete fatto la scelta di gestire direttamente la creazione di un ID unico.
- Quando inserite una nuova informazione nel DB, supponiamo un utente, per prima cosa create un id unico e poi lo associate al record de ìl nuovo utente
- Quindi abbiamo le seguenti operazioni da gestire
- InsertNewUser(username, password) {
  - //Genero un nuovo id unico
  - Id = «select id from id_unico»;
  - «update id_unico set id = id + 1;»
  - //Inserisco il nuovo utente con id unico
  - «Insert into Utenti (id, name, pwd) values («id», «username», «password»);»
- }
- Il problema di carattere generale è che spesso vi trovate nella situazione di

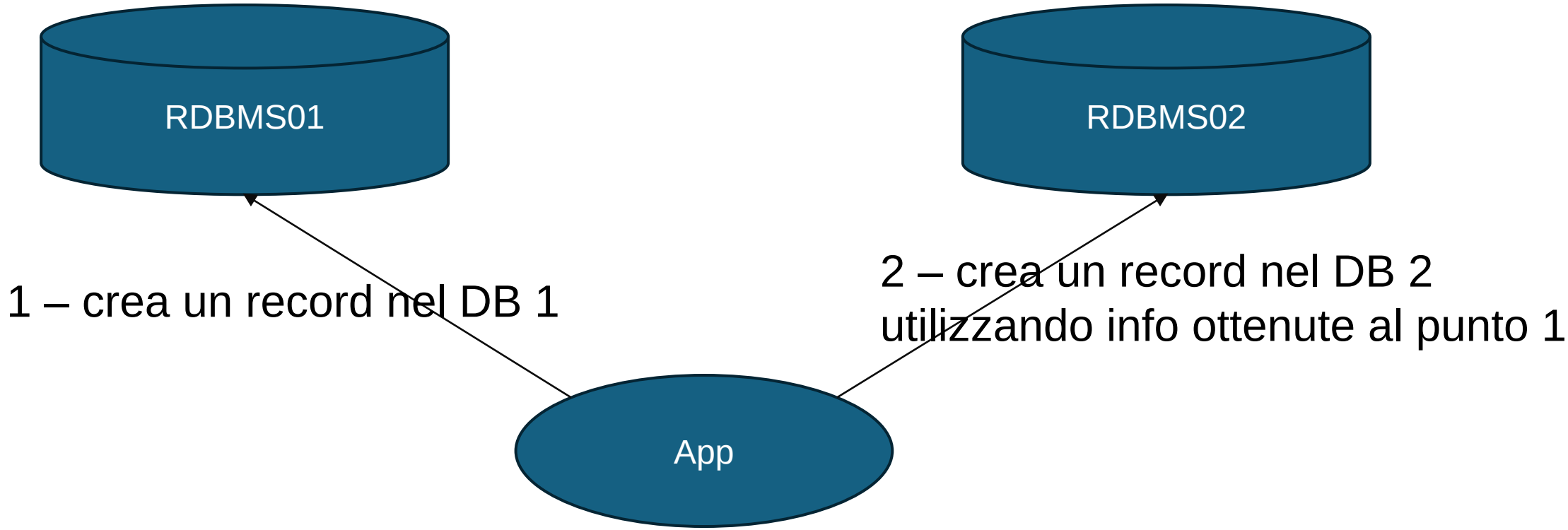
Un esempio più ficcante

- Nel DB ho due tabelle
  - Codice fiscale (con CF, cognome, nome, data di nascita, comune di nascita….
  - Dati fiscali (con CF, cognome, …)
- Quando creo un nuovo individuo
  - Prima creo il CF (nella tabella codiceFiscale)
  - Poi creo un record nella tabella DatiFiscali (con anche l’indirizzo di residenza)…
- Quando elimino un CF
  - Devo eliminarlo dalla tabella Codice fiscale
  - Devo eliminarlo dalla tabella Dati Fiscali

La transazione e la commit

- Tutte le volte in cui ho un insieme di operazioni sul DB che devono essere eseguite come un’unica operazione (operazione atomica) non interrompibile, allora posso utilizzare
- Begin transaction
  - Elenco di operazioni di modifica del DB
  - ...
  - ...
- Se tutto va bene
  - Commit
- Altrimenti
  - Rollback

Two phase commit (due RDBMS in rete)



Se 1 va in errore? Nulla poiché i due DB sono ancora coerenti
Se 2 va in errore? Come fate a ripristinare la coerenza?
Usate UNDO? Ma non potete poiché non siete i soli utenti del RDBMS1 e quindi è possibile che il RDBMS modificato dall'operazione 1 sia già stato utilizzato, aggiornato e modificato considerando proprio il risultato dell'operazione 1

Sql injection

- Sito web che richiede username e password
- L’application server deve formulare una query verso il data server (che ha una tabella con ID, USERNAME e PASSWORD) del tipo

SELECT id FROM utenti WHERE

username = “<lo username che viene dal browser>” AND

password= “<la password che viene dal browser>”;

- Supponiamo di utilizzare PYTHON e che le due variabili valorizzate dal browser sono: user and pwd
- Query = f’SELECT id FROM utenti WHERE username=”{user}” and password=”{pwd}”;’

Sql injection

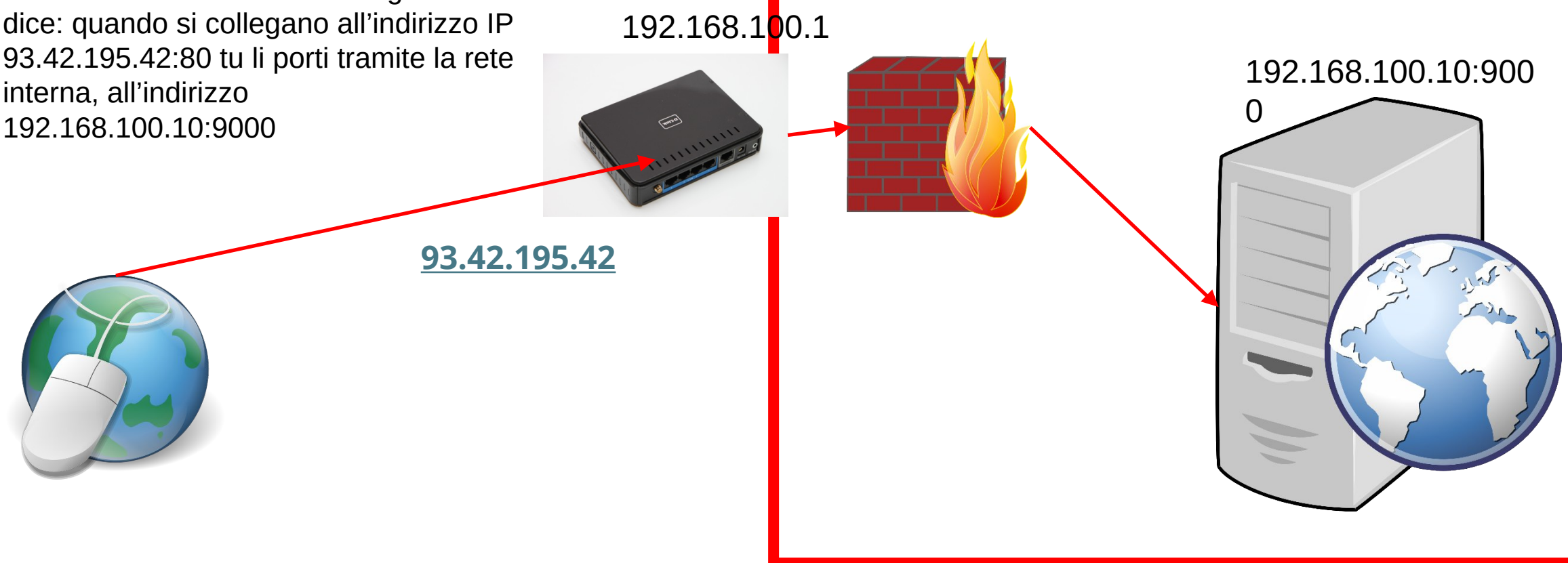
- La stringa di query utilizzata dal progettista
- 'SELECT * FROM users WHERE username =<un> and password = <pwd>

- ATTACCO
  - Nella Maschera di login, al posto dello username metto:
  - a'' or 1=1 --

Una semplice architettura di un server WEB

Come fa il browser a collegarsi al server WEB?

1) DEVE andare all'indirizzo IP pubblico che avete avete acquistato per il server WEB e che avete associato alla sche da di rete verso internet del router
2) nel router avete messo una regola che dice: quando si collegano all'indirizzo IP 93.42.195.42:80 tu li porti tramite la rete interna, all'indirizzo 192.168.100.10:9000



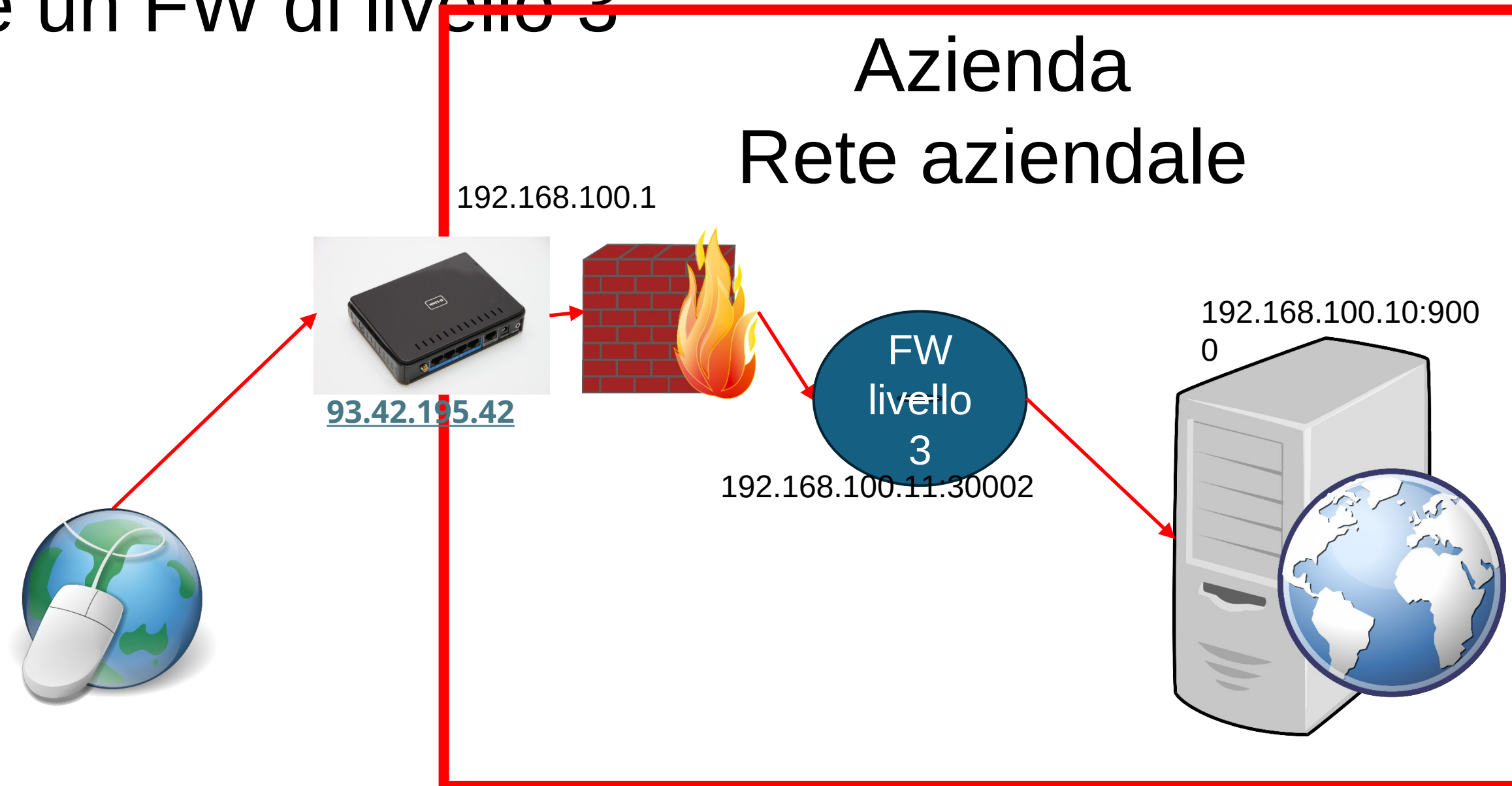
Ruolo del firewall

- Da internet al dispositivo di rete più esterno dell’azienda, sia esso un router o un firewall, arrivano richieste destinate agli IP pubblici associati al dispositivo e su tutte le porte possibili
- Al nostro router di casa (che ha un solo IP) da internet arrivano richieste su una molteplicità di porte TCP
  - Ssh http https Pop3 Imap Smtp ftp telnet
- A livello Aziendale, quindi, alla macchina più esterna arrivano sui suoi IP pubblici, richieste per tutte le possibili porte (0-65535)
- Uno dei ruoli dei Firewall è quello di impedire che richieste su porte «non previste dal progetto della rete» arrivino ai sistemi interni dell’azienda

Esempio «semplici» di regole di firewall

- SRC=10.9.4.1,DST==93.42.195.42:80, Sposta su 192.168.160.10:8990
- SRC=*.*.*.*:*, DST=93.42.195.42:80, Sposta su 192.168.160.10:9000
- SRC=*.*.*.*:*, DST=93.42.195.42:8080, Sposta su 192.168.160.12:89
- …

Una semplice architettura di un server WEB e un FW di livello 3



Architettura semplificata di un sistema di servizi

- Un’azienda che non ha mai utilizzato tecnologie informatiche, decide di mettere in campo un sistema di servizi e di gestirlo direttamente in casa. I servizi che ritiene di erogare sono:
  - Servizi per i dipendenti (accessibili tanto da Internet, tanto da rete interna. Aggiornamenti sono da rete interna)
    - Email aziendale
    - Web aziendale
    - Ftp aziendale
  - Servizi al pubblico (accessibili da Internet in consultazione/uso, accessibili da rete interna per aggiornamento)
    - Web dei servizi al pubblico
      - A pagamento
      - Gratuiti

Nomi di dominio (register DNS)
wp.azienda.cyber.arm – 93.42.195.42
azienda.cyber.arm - 93.42.195.43

Ipdst –	porta dst –	ip inoltro –	porta inoltro
93.42.195.42	80	192.168.101.10	8087
93.42.195.43	80	192.168.100.10	9876
93.42.195.43	25	192.168.100.11	1234
93.42.195.43	110	192.168.100.11	2345
93.42.195.43	993	192.168.100.11	3456
93.42.195.43	21	192.168.100.12	666

Router

93.42.195.4[23]
web servers:80
Smtp: 25 (della 43)
pop3: 110 (della 43)
IMAP: 993 (della 43)
FTP: 21 (della 43)

172.10.10.105

192.168.102.1
172.10.10.104
Firewall
192.168.100.1
192.168.101.1

192.168.100.*

WEB aziendale
192.168.100.10:9876

Email aziendale
192.168.100.11
1234 SMTP,
2345:pop3, 3456:IMAP

FTP aziendale
192.168.100.12:666

192.168.101.*

Web Pubblico
192.168.101.10:8087

192.168.102.*

WebMaster

Admin

PC

PC

PC

Evoluzione
Nomi di dominio (register DNS)
wp.azienda.cyber.arm – 93.42.195.42
azienda.cyber.arm - 93.42.195.42

Ipdst –	porta dst –	ip inoltro –	porta inoltro
93.42.195.42	80	192.168.101.10	8087
93.42.195.42	80	192.168.100.10	9876
93.42.195.42	25	192.168.100.11	1234
93.42.195.42	110	192.168.100.11	2345
93.42.195.42	993	192.168.100.11	3456
93.42.195.42	21	192.168.100.12	666

Router

93.42.195.4[23]
web servers:80
Smtp: 25 (della 43)
pop3: 110 (della 43)
IMAP: 993 (della 43)
FTP: 21 (della 43)

172.10.10.105

192.168.100.1

Firewall

172.10.10.104

192.168.150.1

192.168.100.2

HTTP Reverse proxy/dispatcher/app firewall

192.168.150.2

192.168.101.2

192.168.100.*

WEB aziendale
192.168.100.10:9876

Email aziendale
192.168.100.11
1234 SMTP,
2345:pop3, 3456:IMAP

FTP aziendale
192.168.100.12:666

192.168.101.*

Web Pubblico
192.168.101.10:8087

192.168.102.*

WebMaster

Admin

PC

PC

PC

Architettura semplificata di un sistema di servizi

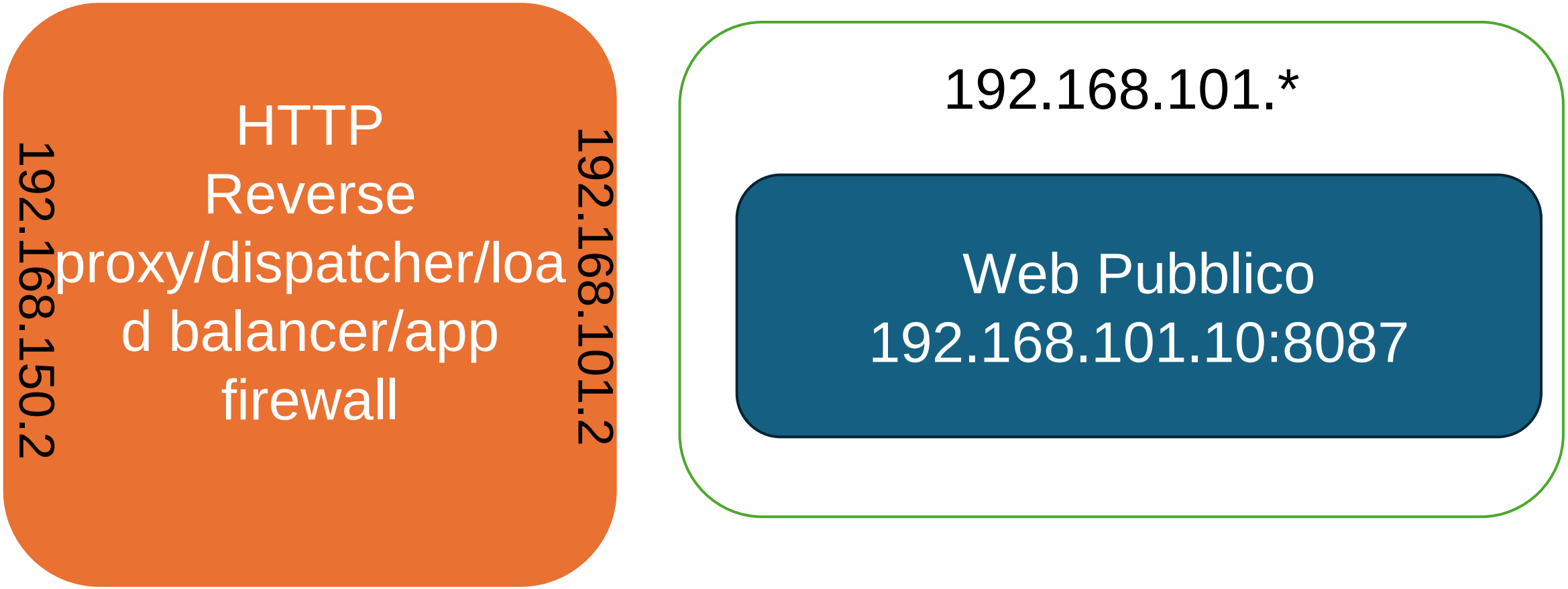
- I dipendenti accedono a
  - Web aziendale
  - Email aziendale
  - ftp aziendale
- Gli sviluppatori accedono a
  - Web aziendale
  - Web pubblico
- L’amministratore di sistema accede a
  - Tutti i sistemi
- Il web master accede a
  - I siti web (aziendale e pubblico)

NB

- Le uniche regole che possono essere impostate su un router (e per la maggior parte anche su un firewall) sono del tipo
- Ip azienda: 213.89.48.137
- Nome di dominio; www.azienda.it
    - Se arrivi sulla porta 80, ti trasferisco all'indirizzo IP 192.168.100.20, porta 57324 (web server dipendenti)
    - Se arrivi sulla porta 81, ti trasferisco all'indirizzo IP 192.168.101.20, porta 65535 (web server pubblico)
    - Ecc
    - Ecc
- Ma nelle comunicazioni WEB conta anche il nome di dominio !!!che deve essere letto e in base a questo le comunicazioni possono essere dirottate»
    - MA I ROUTER NON LEGGONO IL PROTOCOLLO HTTP, lo fanno passare e basta

Abbiamo la ns applicazione in python

- Su quale indirizzi abbiamo detto che la vogliamo?
  - 192.168.101.10:8087
- Domanda
  - Ma sulla vs macchina avete questo indirizzo?
    - In genere no
- Creiamo un indirizzo ip temporaneo
  - sudo ip addr add <ip_address>/<subnet_mask> dev <interface_name>
- Rivediamo in breve la nostra architettura risultante dalla semplificazione e attualizzazione del Progetto sopra esposto


HTTP
Reverse
proxy/dispatcher/loa
d balancer/app
firewall
192.168.150.2
192.168.101.2
192.168.101.*
Web Pubblico
192.168.101.10:8087

Quindi

- L’applicazione è in ascolto sull’indirizzo IP
  - 192.168.101.10 porta 8087
- L’application firewall si attesta (fa la bind) in ascolto sull’indirizzo ip
  - 192.168.150.2, porta 80
- L’application firewall si attesta in trasmissione verso l’applicazione sull’indirizzo ip
  - 192.168.101.2
- Quindi sulla nostra macchina abbiamo bisogno di tre indirizzi IP
  - 192.168.101.10
  - 192.168.101.2
  - 192.168.150.2

Soluzione pubblicata su moodle

Cosa manca per completare il progetto di difesa contro la sql injection?

- Dobbiamo gestire non solo il login ma gestire la registrazione durante la quale un utente fornisce le proprie credenziali
- In tal modo il layer di sicurezza di predispone a riconoscere anche gli utenti che si dovessero registrare durante una sessione di lavoro
- Quindi, tramite una VR, sarà necessario riconoscere la richiesta di registrazione, estrarre username e password, riconoscere la conseguente risposta positiva del sistema di servizi e solo allora aggiornare le strutture (set degli utenti) interne del layer di sicurezza per consentirgli di riconoscere immediatamente il nuovo utente che si fosse appena registrato
- La soluzione su moodle