ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea in Ingegneria e Scienze Informatiche

TELEPORT, UN SERVIZIO PER L'ACCESSO AI PC DEL LABORATORIO E AGLI APPLICATIVI DEL CAMPUS

Tesi di laurea in:
Virtualizzazione e Integrazione di Sistemi

Relatore
Prof. Vittorio Ghini

Candidato
Michele Montesi

Correlatore
Dott. Ciro Barbone

I Sessione di Laurea
Anno Accademico 2024-2025

Sommario

L'accesso remoto alle risorse universitarie ha giocato un ruolo cruciale nella trasformazione del panorama accademico. Questa ricerca si concentra sulla fase di integrazione del servizio Teleport nel campus universitario, rappresentando un ulteriore passo in avanti nell'evoluzione delle risorse e dell'accessibilità. L'obiettivo principale di questo progetto è di agevolare gli studenti, i docenti e il personale universitario nell'accesso a diverse risorse interne, ampliando le opportunità di apprendimento, insegnamento e gestione amministrativa.

Teleport non rappresenta semplicemente una soluzione tecnologica, ma un cambiamento significativo nel modo in cui la comunità accademica può sfruttare il potenziale delle risorse universitarie. Gli studenti avranno la possibilità di accedere a strumenti, risorse e macchine da qualsiasi luogo, potenziando l'apprendimento a distanza e la flessibilità del programma di studio. Per i docenti, questo servizio offre un'efficace piattaforma per condividere materiali didattici, promuovendo una didattica più dinamica e interattiva. Inoltre, il personale universitario trarrà vantaggio dall'efficienza operativa migliorata, semplificando processi amministrativi e garantendo una manutenzione agevole delle risorse tecniche.

Questo servizio permette di accedere a risorse interne al campus quali desktop remoti, ssh, kubernetes, database e altro ancora, utilizzando le credenziali istituzionali dell'Università di Bologna.

Il documento si sviluppa in 4 capitoli principali. Nel primo viene affrontato dettagliatamente il contesto di sviluppo del sistema, esponendo le motivazioni alla base del progetto. Nel secondo capitolo vengono delineati i requisiti di progettazione del sistema e le tecnologie utilizzate. Nel terzo capitolo si analizzano le fasi e i dettagli dell'implementazione del servizio accentuando le modifiche effettuate al prodotto e le motivazioni che le hanno rese necessarie. Infine, nel quarto capitolo, si discutono le modalità con cui questo è stato testato e si analizzano i risultati ottenuti.

"Sembrerebbe che abbiamo raggiunto i limiti di ciò che è possibile ottenere con la tecnologia informatica, anche se bisogna stare attenti con tali affermazioni, poiché tendono a sembrare piuttosto sciocche in 5 anni."

Jhon von Neumann

Indice

Sommario **iii**

1 Contesto **1**
1.1 Teleport 1
1.2 Accesso remoto sicuro 2
1.2.1 Secure Shell 3
1.2.2 Transport Layer Security 3
1.2.3 Virtual Private Network 4
1.2.4 Remote Desktop Protocol 5
1.3 Domini Active Directory 5
1.3.1 Remote Authentication Dial-In User Service 6
1.3.2 Lightweight Directory Access Protocol 7
1.4 Virtualizzazione 7
1.4.1 Docker 8
1.5 Contesto applicativo e confronto con la soluzione preesistente 8

2 Analisi e Progettazione **11**
2.1 Analisi dell'architettura del campus 11
2.2 Analisi dei requisiti 12
2.3 Analisi delle tecnologie utilizzate 12
2.3.1 Teleport 12
2.3.2 Active Directory 15
2.3.3 RADIUS 17
2.3.4 Nginx 18
2.3.5 PostgreSQL 19
2.4 Strumenti relativi a Docker 19
2.4.1 Docker Compose 19
2.4.2 Docker Swarm 19
2.5 Progetto Finale 19

3 Realizzazione **21**
3.1 Installazione di Teleport su macchina fisica 21
3.1.1 Installazione di un server Teleport 22
3.1.2 Implementazione dell'autenticazione RADIUS 22
3.1.3 Installazione dei client 25
3.2 Creazione delle immagini 27
3.2.1 Immagine di Teleport 27
3.2.2 Immagine di PostrgreSQL 29
3.2.3 Immagine di Nginx 31
3.3 Integrazione tramite Docker Compose 31
3.3.1 File di dispiegamento 32

4 Testing **35**
4.1 Fasi di creazione dell'architettura di test 35
4.1.1 Singola istanza di test 35
4.1.2 Parallelizzazione del test 38
4.1.3 Test dockerizzato, parallelo e remoto 41
4.2 Esecuzione del test 44
4.2.1 Scenario di test 44
4.2.2 Esecuzione e monitoraggio del test 44
4.3 Analisi dei risultati 45
4.3.1 Utilizzo della CPU 46
4.3.2 Utilizzo della memoria 47
4.3.3 Utilizzo della rete 50
4.3.4 Processi attivi 52
4.4 Conclusioni del test 53

5 Scenari d'uso **55**
5.1 Gestione dell'accesso remoto a risorse accademiche 55
5.1.1 Laboratori virtuali e cluster di calcolo 55
5.1.2 Progetti di ricerca e accesso a dati sensibili 56
5.2 Didattica e supporto tecnico a distanza 56
5.2.1 Accesso remoto a desktop e applicazioni 56
5.2.2 Assistenza tecnica e gestione remota 56
5.3 Gestione centralizzata delle risorse IT 57
5.3.1 Controllo degli accessi a server e database 57
5.3.2 Monitoraggio e auditing delle sessioni 57

6 Conclusioni **59**
6.1 Lavori futuri 60

7 Ringraziamenti **61**

63

Bibliografia **63**

List of Listings

3.1 Funzione di autenticazione tramite RADIUS 23
3.2 Funzione di verifica della password 24
3.3 File di configurazione di Teleport 28
3.4 Script di inizializzazione di PostgreSQL 30
3.5 Dockerfile di Teleport 32
3.6 Dockerfile di PostgreSQL 32
3.7 docker-compose.yml 32
4.1 LocalLoginTest.java 36
4.2 BaseLoginTest.java 36
4.3 create80Users.ps1 38
4.4 ParallelLocalLoginTest.java 39
4.5 ParallelBaseLoginTest.java 39
4.6 runSelenium.sh 42
4.7 stopSelenium.sh 42
4.8 ParallelRemoteLoginTest.java 43

Elenco degli Acronimi

AAA Authentication, Authorization and Accounting. 11, 17

AD Active Directory. 1, 5, 6, 15, 17, 22, 23, 25, 27, 44

API Application Programming Interface. 2, 13, 15

AWS Amazon Web Services. 15

CA Certificate Authority. 12

CLP Connection Layer Protocol. 3

DC Domain Controller. 38

DevOps Development and Operations. 13

DIT Directory Information Tree. 7

DNS Domain Name System. 5, 16

HTTP Hypertext Transfer Protocol. 15

HTTPS Hypertext Transfer Protocol Secure. 2, 13

IETF Internet Engineering Task Force. 6, 17

ISP Internet Service Provider. 11

LDAP Lightweight Directory Access Protocol. 5, 7, 16, 17

MFA Multi-Factor Authentication. 14

NAS Network Access Server. 6

NoSQL Not Only SQL. 13

PAP Password Authentication Protocol. 22

RADIUS Remote Authentication Dial-In User Service. 6, 11, 17, 18, 20, 53

RDP Remote Desktop Protocol. 5, 13, 15, 26, 27, 44, 46–48, 50, 52, 56

SFTP Secure File Transfer Protocol. 26

SQL Structured Query Language. 13

SSH Secure Shell. 2, 3, 13–15, 25, 26, 28, 56

SSL Secure Socket Layer. 3

SSL/TLS Secure Socket Layer/Transport Layer Security. 18

SSO Single Sign-On. 1, 6, 16

TCP Transmission Control Protocol. 7, 11, 15

TCP/IP Transmission Control Protocol/Internet Protocol. 7, 11

TLP Transport Layer Protocol. 3

TLS Transport Layer Security. 2–4

UAP User Authentication Protocol. 3

UDP User Datagram Protocol. 11, 17

VM Virtual Machine. 7, 42, 44

VPN Virtual Private Network. 4, 6, 15, 17

YAML YAML Ain't Markup Language. 19

Capitolo 1

Contesto

Questo capitolo delinea i pilastri concettuali fondamentali per comprendere il lavoro svolto. Al fine di consentire all'infrastruttura di Unibo l'accesso in modalità *Single Sign-On (SSO)*, è stato implementato un sistema di autenticazione e autorizzazione all'interno del servizio open-source **Teleport**. Questo sistema si basa su **RADIUS** e **Active Directory (AD)**, un servizio centrale per la gestione delle identità e degli accessi. In questo contesto, saranno presentati i concetti basilari relativi ad Active Directory, le sue funzionalità e le caratteristiche chiave che hanno reso possibile l'integrazione con Teleport utilizzando le credenziali universitarie.

Parallelamente, verranno esplorati i concetti fondamentali dell'accesso remoto sicuro, poiché l'interconnessione digitale odierna richiede un'accessibilità sicura e affidabile. In questo progetto, sono stati esaminati in dettaglio i principali concetti e strumenti relativi all'accesso remoto sicuro, ai protocolli di desktop remoto e alla virtualizzazione.

1.1 Teleport

Teleport è un servizio open-source per l'accesso remoto sicuro e la gestione degli accessi. È sviluppato da *Gravitational, Inc.* e rilasciato sotto licenza *Apache 2.0*. È stato progettato per consentire agli utenti di accedere in modo sicuro a server, applicazioni e altri servizi di rete utilizzando un'unica identità aziendale. [Tel23a]

Una delle sue caratteristiche principali è la registrazione di tutte le attività, che fornisce un tracciamento completo per scopi di audit. Inoltre, si integra con i provider d'identità esistenti e supporta l'autenticazione a due fattori, rendendo l'accesso sia sicuro che conveniente. Teleport fornisce anche un controllo degli accessi basato sui ruoli, permettendo agli amministratori di definire precisamente quali utenti possono accedere a quali risorse. Queste funzionalità rendono Teleport uno strumento potente per la gestione dell'accesso remoto sicuro.

Teleport consente la condivisione delle sessioni di terminale, rendendo più semplice la collaborazione e la risoluzione dei problemi. Supporta certificati a breve durata, che riducono il rischio di attacchi basati su credenziali rubate. Fornisce un piano di accesso unificato, che semplifica la gestione dell'accesso a una vasta gamma di risorse. Infine, supporta le richieste di accesso e le revisioni, permettendo un controllo più stretto sull'accesso alle risorse critiche. Queste caratteristiche aggiuntive rendono Teleport uno strumento ancora più potente per la gestione dell'accesso remoto sicuro.

Utilizza diversi protocolli standard per fornire accesso alle varie risorse. Di seguito alcuni esempi: per l'accesso ai server, utilizza Secure Shell (SSH), un protocollo ben consolidato per l'accesso remoto sicuro. Per l'accesso ai cluster *Kubernetes*, utilizza l'Application Programming Interface (API) di Kubernetes, che fornisce un'interfaccia potente e flessibile per la gestione dei cluster. Per l'accesso alle applicazioni web, utilizza Hypertext Transfer Protocol Secure (HTTPS), il protocollo standard per la comunicazione sicura sul web. In tutti questi casi, Teleport utilizza il Transport Layer Security (TLS) reciproco per l'autenticazione e la crittografia, garantendo che solo le parti autorizzate possano accedere alle risorse e che tutte le comunicazioni siano protette da intercettazioni e manomissioni.

1.2 Accesso remoto sicuro

Nel contesto sempre più interconnesso della moderna era digitale, l'accesso remoto a risorse e servizi è diventato un pilastro fondamentale. Questa pratica, resa possibile da una serie di protocolli e tecnologie, ha permesso l'accessibilità a distanza garantendo contemporaneamente sicurezza e affidabilità.

1.2.1 Secure Shell

Lo **SSH** è un protocollo crittografico progettato per fornire comunicazioni sicure su una rete non sicura. Fu sviluppato da *Tatu Ylönen*, un ricercatore finlandese, nel 1995. Ylönen creò SSH come risposta alla vulnerabilità dei protocolli di accesso remoto precedenti, come *Telnet* e *rlogin*, che trasmettevano le informazioni, inclusa la password, in chiaro, rendendo le comunicazioni suscettibili all'intercettazione.

SSH utilizza una coppia di chiavi crittografiche, una pubblica e una privata, per autenticare il client e il server durante il processo di connessione. Quando un utente si connette a un server remoto, il server invia la sua chiave pubblica, e se il client la riconosce e accetta, viene generata una sessione crittografata tra i due dispositivi.

Il protocollo offre una serie di funzionalità, tra cui l'accesso remoto a un computer, il trasferimento di file in modo sicuro e la possibilità di eseguire comandi remoti in un ambiente crittografato. La sua versatilità e la sua capacità di garantire connessioni sicure ne fanno uno degli strumenti principali nell'ambito dell'amministrazione di sistemi e della sicurezza informatica.

All'interno dell'ecosistema di SSH, troviamo protocolli complementari come il **Transport Layer Protocol (TLP)**, **User Authentication Protocol (UAP)** e il **Connection Layer Protocol (CLP)**. Il primo è un protocollo di livello di trasporto che fornisce un canale sicuro per le comunicazioni tra client e server. Il secondo è un protocollo di autenticazione che consente di autenticare l'utente e il server durante il processo di connessione. Infine, il terzo è un protocollo di livello di connessione che consente di eseguire comandi remoti in un ambiente crittografato. L'implementazione di questi protocolli aggiuntivi amplia le funzionalità di sicurezza e controllo di SSH durante le comunicazioni e gli accessi remoti, fornendo un livello aggiuntivo di protezione e controllo. [Ale23]

1.2.2 Transport Layer Security

Il **TLS**, evoluzione del precedente **Secure Socket Layer (SSL)**, è stato sviluppato per garantire la privacy e la sicurezza delle comunicazioni su Internet. La sua genesi risale agli anni '90 come risposta alla necessità di proteggere le informazioni scambiate su reti non sicure.

Il TLS impiega una combinazione di tecniche crittografiche, tra cui algoritmi di *crittografia asimmetrica* e *simmetrica*, per garantire la confidenzialità, l'integrità e l'autenticazione dei dati scambiati tra i dispositivi connessi. Durante il processo di negoziazione della connessione, TLS stabilisce un canale sicuro tra il client e il server, verificando l'autenticità delle identità e crittografando i dati scambiati. TLS include una serie di protocolli di sicurezza aggiuntivi come **Handshake Protocol**, che gestisce la negoziazione della connessione sicura, e **Record Protocol**, responsabile della crittografia e del trasferimento sicuro dei dati tra client e server.

Questo protocollo offre un livello di sicurezza che consente di proteggere vari tipi di comunicazioni su Internet, come ad esempio l'accesso a siti web (**HTTPS**), l'invio di e-mail, il trasferimento di file e altre applicazioni che richiedono una connessione sicura. [Moz23] [Clo23]

1.2.3 Virtual Private Network

Le **Virtual Private Network (VPN)** sono nate dall'esigenza di creare reti private virtuali attraverso reti pubbliche come Internet. Il concetto di VPN ha origini negli anni '90, quando le aziende cercavano soluzioni sicure per connettere le loro sedi remote a una rete aziendale centrale. La necessità di trasferire dati in modo sicuro su reti non affidabili ha portato allo sviluppo di protocolli e tecnologie per garantire la privacy e la sicurezza delle comunicazioni. [Moc21]

Il suo funzionamento prevede la creazione di un tunnel crittografato tra il dispositivo del cliente e il server VPN. Durante questo processo, i dati vengono crittografati all'interno del tunnel, impedendo a terze parti d'intercettare o comprendere il contenuto delle comunicazioni. Le VPN utilizzano protocolli crittografici, come *IPsec* o *OpenVPN*, per garantire la sicurezza e l'integrità dei dati trasmessi.

Ne esistono diverse tipologie, tra cui:

- *hub-and-spokes VPN*: permette agli utenti di connettersi in modo sicuro alla rete aziendale, o a server specializzati, da remoto.

- *site-to-site VPN*: consente la connessione sicura tra sedi aziendali distanti.

[Ara]

1.2.4 Remote Desktop Protocol

Il **Remote Desktop Protocol (RDP)** è stato sviluppato da *Microsoft* negli anni '90 con l'obiettivo di consentire agli utenti di accedere e controllare un computer remoto tramite una connessione di rete. Questo protocollo è stato ideato per facilitare l'accesso remoto a un sistema e consentire l'esecuzione di applicazioni, l'accesso ai file e il controllo completo del desktop da un dispositivo remoto.

Il suo funzionamento coinvolge la comunicazione tra un computer client e un computer host tramite una connessione di rete. Il computer host deve eseguire un software server RDP, mentre il computer client utilizza un'applicazione client per connettersi al server.

RDP fornisce un'interfaccia grafica remota che consente all'utente di interagire con il desktop del computer host come se fosse fisicamente presente davanti ad esso. Questo protocollo consente il trasferimento di dati di input come movimenti del mouse e tastiera dall'utente al computer host, nonché il trasferimento di dati di output come il display grafico dal computer host al computer client. [Zer20]

1.3 Domini Active Directory

Active Directory è un servizio di directory sviluppato da *Microsoft* per la gestione delle identità e degli accessi. Questo servizio è stato introdotto per la prima volta con Windows 2000 Server e da allora è stato aggiornato e migliorato con ogni nuova versione di Windows Server. [Seq17]

Il concetto di AD nacque dalla necessità di semplificare e rendere più efficiente la gestione delle risorse di rete in ambienti aziendali complessi. Offre un modello gerarchico e strutturato per organizzare e gestire gli oggetti di rete, consentendo una gestione centralizzata degli utenti, dei gruppi, delle risorse di rete e delle politiche di sicurezza.

Si basa su un modello a directory che consente di organizzare gli oggetti di rete in una struttura gerarchica basata su domini, alberi e foreste. Utilizza protocolli standard come Lightweight Directory Access Protocol (LDAP), per accedere e gestire le informazioni contenute nella directory, e Domain Name System (DNS) per la gestione delle informazioni e la distribuzione dei servizi di rete.

Tra le sue funzionalità principali ci sono l'autenticazione degli utenti, la gestione dei privilegi di accesso, la distribuzione delle politiche di gruppo, la gestione centralizzata delle risorse di rete come stampanti, file e cartelle condivise, e la facilitazione della ricerca e dell'accesso alle risorse all'interno della rete.

AD è ampiamente utilizzato nell'ambito aziendale per semplificare la gestione degli account utente, delle risorse di rete e delle politiche di sicurezza. Consente ai dipendenti di accedere a risorse condivise, stampanti di rete e altri servizi utilizzando un'unica identità aziendale.

Viene, inoltre, impiegato per implementare servizi di SSO, consentendo agli utenti di accedere a diversi servizi e applicazioni con un'unica autenticazione. Questo facilita la gestione delle credenziali e migliora l'esperienza dell'utente. [Mic22]

1.3.1 Remote Authentication Dial-In User Service

Il **Remote Authentication Dial-In User Service (RADIUS)** è un protocollo di autenticazione, autorizzazione e accounting sviluppato originariamente dalla *Internet Engineering Task Force (IETF)*. È stato introdotto per la prima volta negli anni '90 come soluzione per gestire l'autenticazione degli utenti che si connettono a una rete tramite accesso remoto, come ad esempio le connessioni dial-up e le reti VPN.

L'obiettivo principale è stato quello di fornire un server centralizzato per l'autenticazione degli utenti, semplificando la gestione degli accessi a reti remote e migliorando la sicurezza delle connessioni.

RADIUS è un protocollo client/server che si basa sul concetto di autenticazione centralizzata. Il server gestisce le richieste di autenticazione provenienti dai client, come ad esempio un *Network Access Server (NAS)* o un access point wireless. Quando un utente tenta di accedere alla rete, il NAS inoltra la richiesta di autenticazione al server. Questo verifica le credenziali dell'utente, autenticandolo attraverso le informazioni contenute nel suo database. Una volta autenticato con successo, il server RADIUS autorizza l'accesso alla rete e tiene traccia delle attività dell'utente per scopi di accounting. [Elc23]

1.3.2 Lightweight Directory Access Protocol

Il **LDAP** è un protocollo di accesso ai servizi di directory nato negli anni '90 come soluzione leggera ed efficiente per accedere e gestire informazioni all'interno di un servizio di directory. LDAP è stato progettato per offrire un modo standardizzato per accedere a directory distribuite e memorizzare informazioni relative agli oggetti di rete.

L'obiettivo principale è semplificare l'accesso alle informazioni contenute nelle directory, consentendo agli utenti di recuperare dati come informazioni sull'utente, gruppi, risorse di rete e altre informazioni di configurazione in modo efficiente e scalabile.

LDAP è basato su un modello di dati ad albero gerarchico in cui gli oggetti sono organizzati in una struttura gerarchica nota come **Directory Information Tree (DIT)**. Utilizza il protocollo Transmission Control Protocol/Internet Protocol (TCP/IP) per la comunicazione ed è basato su un modello client/server.

I client inviano richieste al server per accedere e modificare le informazioni nella directory. Il server elabora le richieste dei client, autentica gli utenti, ricerca e fornisce le informazioni richieste. Utilizza il protocollo di trasporto Transmission Control Protocol (TCP) per le comunicazioni, solitamente sulla porta 389. [Hat22]

1.4 Virtualizzazione

La virtualizzazione è una tecnologia nata negli anni '60 con l'introduzione delle prime Virtual Machine (VM). Originariamente sviluppata per sfruttare al massimo le risorse hardware limitate, la virtualizzazione è cresciuta nel tempo diventando una tecnologia fondamentale per ottimizzare l'utilizzo delle risorse hardware e semplificare la gestione delle infrastrutture IT.

Negli anni '90, la virtualizzazione è stata ulteriormente sviluppata con l'introduzione di soluzioni più avanzate che consentivano di eseguire diverse macchine virtuali su un singolo host fisico.

1.4.1 Docker

Docker è stato introdotto per la prima volta nel 2013 da *Docker, Inc.* come un'innovativa piattaforma open-source per la creazione, la distribuzione e l'esecuzione di applicazioni in contenitori. Si basa sul concetto di containerizzazione, che è stato semplificato e reso accessibile a un vasto pubblico di sviluppatori.

La tecnologia dei container esisteva già da tempo, ma Docker ha reso la sua adozione più ampia semplificandone l'uso attraverso strumenti user-friendly e offrendo un ecosistema integrato per lo sviluppo e il deployment delle applicazioni.

Utilizza una tecnologia di containerizzazione per incapsulare le applicazioni insieme alle loro dipendenze e risorse in contenitori leggeri e portabili. Ogni container offre un ambiente isolato e autonomo in cui un'applicazione può essere eseguita indipendentemente dall'ambiente in cui è ospitata.

La sua architettura modulare comprende Docker Engine, che fornisce la piattaforma per la gestione dei container, Docker Hub, un registro per condividere e distribuire immagini di container, e Docker Compose, uno strumento per la definizione e la gestione di applicazioni multi-container.

La sua portabilità e scalabilità lo rendono ideale per l'orchestrazione di applicazioni, la gestione dei microservizi e l'implementazione di architetture basate su container. È utilizzato anche nei contesti DevOps per automatizzare il deployment delle applicazioni e migliorare l'efficienza delle operazioni IT. [Wik23]

1.5 Contesto applicativo e confronto con la soluzione preesistente

Attualmente, l'accesso remoto alle risorse universitarie viene gestito principalmente tramite il servizio **Apache Guacamole**, una piattaforma clientless che consente l'accesso a desktop remoti e terminali via browser. Guacamole supporta i protocolli **RDP**, **SSH** e **VNC** ed è stato a lungo utilizzato per permettere a studenti e docenti di connettersi ai computer dei laboratori o ad alcuni server interni del campus.

Tuttavia, le esigenze dell'attività didattica e scientifica hanno evidenziato alcuni limiti strutturali di questa soluzione. In particolare:

- Guacamole consente l'accesso solo a desktop e terminali remoti, ma **non permette di esporre direttamente servizi applicativi**, come ad esempio **MySQL**, **PostgreSQL**, o interfacce web di applicazioni interne al campus;

Per rispondere a queste nuove necessità, è stata avviata la sperimentazione della piattaforma **Teleport**, uno strumento open-source che abilita un accesso remoto sicuro, tracciabile e versatile a molteplici tipologie di risorse, andando oltre il semplice desktop remoto.

Con Teleport, l'utente:

- **Accede tramite un browser** all'interfaccia della piattaforma;
- Si autentica usando le **credenziali universitarie** personali (grazie all'integrazione con **Active Directory** via **RADIUS**);
- Una volta autenticato, viene presentato con una dashboard che mostra in modo chiaro tutte le **risorse a cui ha diritto di accedere**, in base al proprio ruolo e ai permessi assegnati;
- Può quindi accedere con un solo click, direttamente dal browser, a servizi remoti via SSH, desktop Windows (RDP), database, applicazioni web e altri endpoint, **senza la necessità di usare una VPN**.

Uno degli aspetti più rilevanti riguarda proprio l'eliminazione della VPN, che in precedenza era spesso necessaria per accedere a risorse interne. Grazie a Teleport, le risorse sono **esposte in modo sicuro tramite reverse proxy**, il che offre diversi vantaggi:

- Aumenta la **comodità d'uso**: l'utente non deve più installare software aggiuntivo né avviare tunnel VPN manualmente;
- Aumenta la **sicurezza dell'infrastruttura universitaria**: l'università non deve più accettare connessioni da PC potenzialmente vulnerabili o compromessi che, una volta in VPN, potrebbero rappresentare un rischio per la rete interna.

Capitolo 2

Analisi e Progettazione

2.1 Analisi dell'architettura del campus

L'Università di Bologna utilizza *Active Directory* come sistema di gestione delle risorse all'interno dell'organizzazione. Questo è il servizio di Directory proprietario di Microsoft, che consente di gestire in modo centralizzato le risorse di rete, come ad esempio i computer, i gruppi di lavoro, gli utenti e le risorse di stampa.

RADIUS [Elc23] è il protocollo Authentication, Authorization and Accounting (AAA) utilizzato in applicazioni di accesso ad altre reti o di mobilità IP. Il protocollo è stato progettato per funzionare con diversi protocolli di rete e supportare diversi metodi di autenticazione. È spesso utilizzato dagli Internet Service Provider (ISP) e dalle aziende per autenticare gli utenti prima di consentire loro l'accesso alla rete.

Viene eseguito nel livello di applicazione del TCP/IP e può utilizzare il protocollo TCP o User Datagram Protocol (UDP) per il trasporto. I server di accesso alla rete e i *gateway* che controllano l'accesso a una rete di solito contengono un componente *client RADIUS* che comunica con il *server RADIUS*, solitamente un processo in background in esecuzione su un server UNIX o Windows. Attualmente rappresenta lo standard de facto per l'autenticazione remota degli utenti.

Nel caso specifico di **Teleport**, questo non effettuerà direttamente l'autenticazione degli utenti, ma si limiterà a comunicare con il server RADIUS, che a sua volta si occuperà dell'autenticazione.

2.2 Analisi dei requisiti

Prima di procedere con la progettazione del sistema, è necessario effettuare una chiara analisi dei requisiti dello stesso.

Dal punto di vista dei **requisiti funzionali**, il sistema deve essere in grado di:

1. Integrare l'autenticazione tramite server RADIUS;
2. Garantire il funzionamento con i sistemi che si vogliono integrare;
3. Assicurare la sicurezza dei dati degli utenti sfruttando il protocollo HTTPS.

Per quanto riguarda i **requisiti non funzionali**, il sistema deve essere:

1. Composto esclusivamente da tecnologie open source;
2. Trasparente per gli amministratori di sistema e facilmente manutenibile;
3. Di immediato utilizzo per gli studenti;
4. Reattivo e accessibile da qualunque dispositivo.

Per quanto riguarda il primo requisito funzionale, ovvero l'integrazione con il server RADIUS, è necessario effettuare un'analisi approfondita dell'architettura del campus, in modo da poter comprendere come è strutturato il sistema di autenticazione e autorizzazione degli utenti.

2.3 Analisi delle tecnologie utilizzate

2.3.1 Teleport

Teleport [Tel23b] è una soluzione open source che fornisce un accesso sicuro alle risorse, alle applicazioni e ai sistemi. È ampiamente utilizzato in ambienti aziendali, universitari e governativi per garantire un accesso remoto sicuro ai server e alle applicazioni sensibili. Teleport offre funzionalità avanzate di autenticazione, autorizzazione e registrazione delle sessioni per proteggere l'accesso ai sistemi critici. È una Certificate Authority (CA) e un proxy di accesso multi-protocollo

consapevole dell'identità. Implementa protocolli come SSH, RDP, HTTPS, Kubernetes API e una varietà di database Structured Query Language (SQL) e Not Only SQL (NoSQL). È progettato per essere completamente trasparente per gli strumenti lato client e funziona con tutti gli elementi dell'ecosistema Development and Operations (DevOps) odierno.

Architettura

L'architettura chiave di Teleport si basa sul concetto di "cluster". Un cluster Teleport è composto dal **Teleport Auth Service**, dal **Teleport Proxy Service**, dagli **agenti Teleport** e dalle risorse alle quali si desidera connettersi. Queste risorse possono includere server Linux o Windows, database, cluster Kubernetes, desktop Windows e applicazioni web interne.

Per creare un cluster Teleport di base, è necessario avviare tre servizi chiave:

- **Teleport Auth Service**: L'autorità di certificazione del cluster, emette certificati per i client e mantiene il registro degli audit.
- **Teleport Proxy Service**: Il proxy che consente l'accesso alle risorse del cluster dall'esterno. Solitamente è l'unico servizio disponibile dalla rete pubblica.
- **Teleport Agent Service**: Gli agenti Teleport funzionano nella stessa rete delle risorse di destinazione e parlano il protocollo nativo di queste risorse, come SSH, Kubernetes API, HTTPS, PostgreSQL e MySQL. Si possono considerare come dei proxy che instradano le richieste degli utenti verso le risorse di destinazione.

I client devono autenticarsi con Teleport e ricevere un certificato client che funziona automaticamente per tutte le risorse all'interno di un cluster. Dopo l'autenticazione, i comandi di accesso remoto come `ssh`, `kubectl`, `psql`, `mysql` e altri saranno configurati con l'identità dell'utente. Teleport offre un database integrato per la gestione degli utenti. Nel caso specifico di questo progetto, si utilizzerà un database PostgreSQL esterno per la gestione degli utenti.

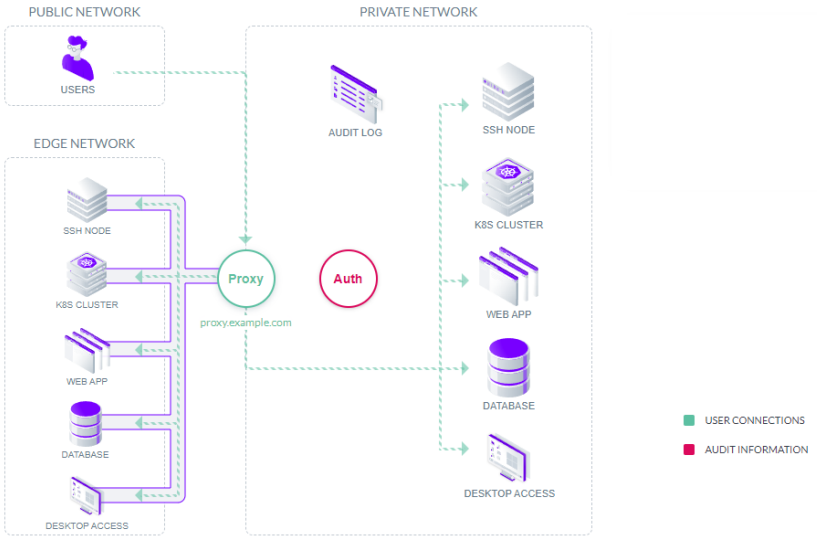

Figura 2.1: Architettura di Teleport

Core Concepts

Cluster Il concetto di cluster è fondamentale per il modello di sicurezza di Teleport. Utenti e server devono unirsi allo stesso cluster prima che l'accesso possa essere concesso. Entrambi devono autenticarsi e ricevere certificati per farlo. Il *Teleport Auth Service* è l'autorità di certificazione del cluster, emettendo certificati per utenti e server con tutti i protocolli supportati.

Autenticazione a due fattori Teleport supporta il Multi-Factor Authentication (MFA) per garantire un accesso altamente sicuro. Gli utenti devono fornire prove multiple della loro identità, come password e token generati da app di autenticazione, per accedere al sistema.

SSH Bastion Teleport è un server SSH che funge da *bastion host* per l'accesso remoto. I bastion host sono server che si trovano in una rete protetta e consentono l'accesso a una rete privata da una rete pubblica. Questo elimina la necessità di esporre direttamente i server al traffico SSH non protetto.

Servizi

Application Service Il Teleport Application Service si occupa di instradare il traffico Hypertext Transfer Protocol (HTTP) e TCP verso gli endpoint configurati dall'utente, ad esempio applicazioni web interne o la console Amazon Web Services (AWS). Questo servizio consente agli utenti di accedere in modo sicuro a risorse web interne senza dover utilizzare una VPN.

Database Service Il Teleport Database Service funge da proxy per il traffico TCP nei protocolli nativi dei database più diffusi, inclusi PostgreSQL e MySQL. Fornisce un accesso sicuro e controllato alle risorse del database, garantendo che solo gli utenti autorizzati possano interagire con i dati.

Desktop Service Il Teleport Desktop Service funge da proxy per il traffico del protocollo RDP verso desktop Windows. Questo servizio consente agli utenti di connettersi in modo sicuro a desktop Windows remoti e di eseguire sessioni RDP sicure.

Kubernetes Service Il Teleport Kubernetes Service instrada il traffico HTTP verso il server API di Kubernetes. Questo servizio semplifica l'accesso alle risorse e alle applicazioni Kubernetes, garantendo un controllo completo e un audit delle attività degli utenti.

SSH Service Il Teleport SSH Service è un'implementazione di un server SSH che consente agli utenti di eseguire comandi su macchine remote. Offre funzionalità avanzate di controllo degli accessi, registrazione delle sessioni e auditing, garantendo un accesso SSH sicuro e controllato.

2.3.2 Active Directory

AD è un servizio di directory sviluppato da Microsoft per reti Windows Domain. Originariamente lanciato con Windows 2000 Server, è diventato uno standard per la gestione delle identità e l'accesso in ambienti aziendali.

Caratteristiche principali di Active Directory:

- **Gestione delle Identità e Accesso**:
  - *Account Utente*: Gestione delle credenziali e proprietà degli utenti.
  - *Gruppi*: Organizzazione degli utenti in gruppi per una facile assegnazione di permessi.
- **Integrazione con Windows Environment**:
  - Integrazione con altri servizi Windows come Microsoft Exchange Server e SharePoint.
- **Servizi di Directory:**
  - *LDAP*: Supporto per LDAP, un protocollo standard per accedere alle informazioni di directory.
  - *DNS*: Utilizzo del DNS per la risoluzione dei nomi di dominio.
- **Politiche di Gruppo:**
  - Utilizzo delle Politiche di Gruppo per controllare le impostazioni di configurazione.
- **Autenticazione e Autorizzazione**:
  - Meccanismo centralizzato per l'autenticazione e l'autorizzazione degli utenti.
- **Servizi di Federazione**:
  - *ADFS*: Active Directory Federation Services per SSO.
- **Sicurezza**:
  - Implementazione di controlli di sicurezza come la crittografia Kerberos.
- **Scalabilità e Affidabilità**:
  - Progettato per supportare grandi reti aziendali con funzionalità di replica.

- **Interoperabilità**:
  - Supporto per l'interoperabilità con sistemi non-Microsoft.

In contesti aziendali, AD è spesso utilizzato per gestire l'accesso a risorse di rete, implementare politiche di sicurezza e centralizzare l'amministrazione IT.

2.3.3 RADIUS

RADIUS è un protocollo di rete AAA ampiamente utilizzato per fornire accesso centralizzato a reti e servizi. Originariamente sviluppato da Livingston Enterprises nel 1991, è stato successivamente standardizzato dall'IETF.

Caratteristiche principali di RADIUS:

- **Autenticazione e Autorizzazione**:
  - Fornisce un metodo centralizzato per autenticare gli utenti e autorizzare l'accesso a servizi e risorse di rete.
  - Comunemente usato per il controllo degli accessi in reti VPN, wireless e dial-up.
- **Protocollo**:
  - Utilizza il protocollo UDP per la comunicazione tra il client RADIUS (tipicamente un server di accesso alla rete) e il server RADIUS.
  - I messaggi tra client e server sono autenticati con un segreto condiviso.
- **Estensibilità**:
  - Supporta un'ampia gamma di metodi di autenticazione, come password, token hardware, certificati digitali e altri.
  - Può essere integrato con altri sistemi di autenticazione, come Kerberos o LDAP.
- **Sicurezza**:

  - Offre meccanismi per proteggere la confidenzialità e l'integrità dei dati durante la trasmissione.
  - Supporta la crittografia per aumentare la sicurezza delle comunicazioni.

- **Scalabilità**:
  - Può gestire un grande numero di richieste di autenticazione e autorizzazione.
  - Scalabile per soddisfare le esigenze di grandi organizzazioni e fornitori di servizi.

- **Interoperabilità**:
  - Compatibile con un'ampia varietà di dispositivi e sistemi di rete.
  - Supporta l'interoperabilità tra diversi fornitori e piattaforme.

RADIUS è una componente chiave nelle moderne infrastrutture di rete, fornendo un framework affidabile e sicuro per la gestione dell'accesso e delle credenziali degli utenti.

2.3.4 Nginx

NGINX è un server web multipiattaforma utilizzato nel progetto per implementare il reverse proxy e quindi controllare l'accesso al sistema. Creato da Igor Sysoev per uno dei più grandi portali web russi, supporta tutte le classiche funzionalità di un web server, incluso il supporto alla tecnologia Secure Socket Layer/Transport Layer Security (SSL/TLS), la funzionalità di reverse proxying, bilanciamento di carico e controllo degli accessi. È stato scelto in quanto semplice da configurare e performante nella gestione di grandi carichi di lavoro. Infatti l'architettura di NGINX permette di gestire molte richieste con un basso overhead di risorse, attraverso il seguente meccanismo di funzionamento: un processo master delega la gestione delle richieste effettuate al server a dei processi worker, i quali riescono ad elaborare richieste multiple attraverso un meccanismo di funzionamento asincrono implementando una modalità di lavoro concorrente. La conseguenza a livello funzionale più importante è la possibilità di sfruttare nella maniera più efficiente possibile le risorse hardware.

2.3.5 PostgreSQL

PostgreSQL è un DBMS ad oggetti open source sviluppato all'Università di Berkley in California. Largamente utilizzato per la sua flessibilità, viene utilizzato all'interno del progetto per la creazione di un database nel quale salvare utenti, sessioni e dati di configurazione.

2.4 Strumenti relativi a Docker

Segue una breve descrizione degli strumenti utilizzati per la gestione dei container utili al dispiegamento dell'applicazione dapprima su singolo host e successivamente in modalità *swarm* su più host.

2.4.1 Docker Compose

Docker Compose facilita la definizione e l'esecuzione di applicazioni multi-container attraverso file YAML Ain't Markup Language (YAML), detti compose file, che specificano i servizi, le reti e i volumi. Questo strumento consente una gestione efficiente dell'intero ciclo di vita dell'applicazione, semplificando le operazioni sui container e offrendo una visione complessiva del sistema.

2.4.2 Docker Swarm

Docker Swarm è un sistema di orchestrazione per gestire container su un cluster di Docker Engine, consentendo la definizione dello stato ottimale di ogni servizio e la loro scalabilità su più host. Incorpora concetti come swarm, nodi (manager e worker), services, stack e task, e offre funzionalità come overlay network, bilanciamento del carico e un servizio DNS interno. Permette aggiornamenti dinamici e gestione efficiente dei container nel cluster.

2.5 Progetto Finale

In seguito all'analisi dei requisiti e delle tecnologie utilizzate, è possibile procedere con la progettazione software del sistema. Il progetto finale è quindi composto da:

- Un **server Teleport** che include `teleport`, `tsh` e `tctl`;
- Un **server RADIUS** che si occupa di gestire l'autenticazione degli utenti;
- Un **server Nginx** che si occupa di gestire il reverse proxy e quindi l'accesso al sistema;
- Un **database PostgreSQL** che si occupa di gestire gli utenti e le sessioni;
- Più **client Teleport** i quali permettono agli utenti di connettersi al sistema.

Saranno inoltre presenti un **server Active Directory** e un **client RADIUS** che si occuperanno di gestire l'autenticazione degli utenti. L'intera architettura è rappresentata in figura 2.2.

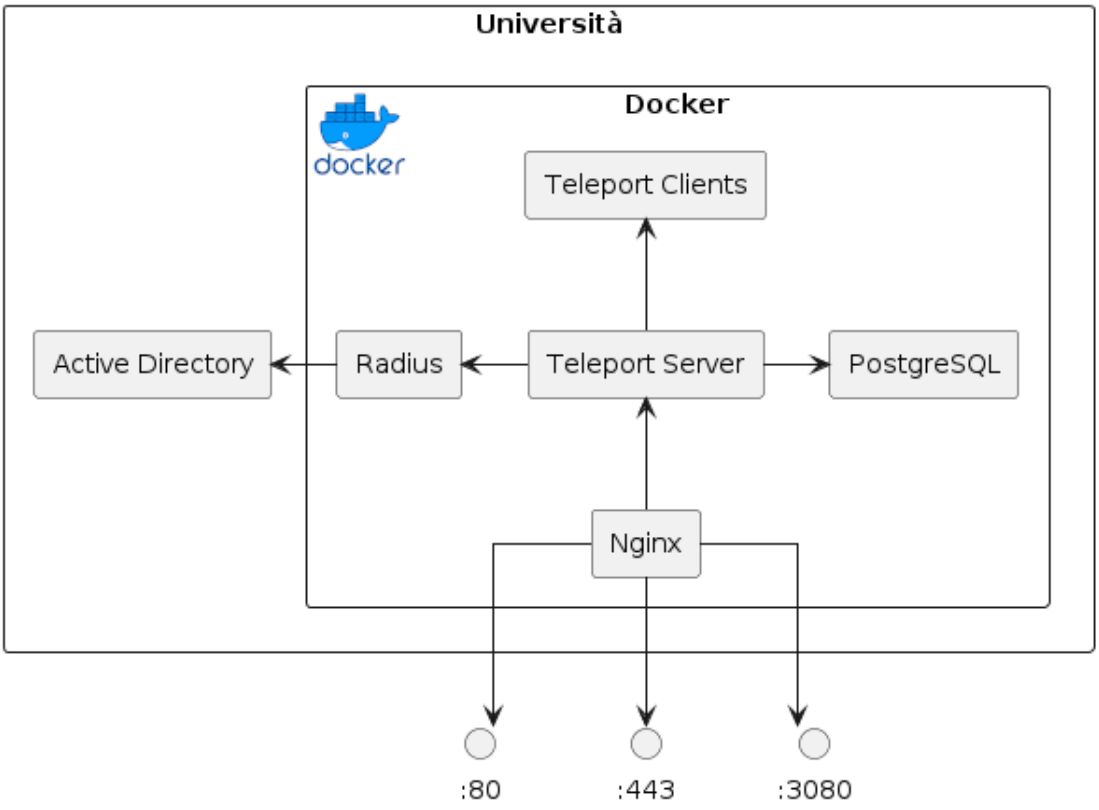

Figura 2.2: Architettura progetto finale

Capitolo 3

Realizzazione

La realizzazione del progetto ha seguito un percorso incrementale che è possibile suddividere in tre fasi principali:

1. Installazione di Teleport su una macchina fisica;
2. Creazione delle immagini dei componenti descritti in fase progettuale;
3. Integrazione dei container tramite Docker Compose;

Oltre a queste tre fasi principali, sarebbe possibile eseguire il dispiegamento dell'applicazione su più macchine separate tramite Docker Swarm, con l'obiettivo di simulare un ambiente di produzione reale. In questo progetto non è stato possibile completare questa fase per motivi di tempo, ma potrebbe essere implementata in futuro.

3.1 Installazione di Teleport su macchina fisica

Come prima fase del progetto, è stato necessario installare Teleport su una macchina fisica per poter comprendere meglio il funzionamento del software e per poterlo testare in un ambiente reale. L'installazione e lo studio sono stati divisi in due fasi principali:

1. Installazione di un server Teleport;
2. Implementazione dell'autenticazione RADIUS;

3. Installazione dei client.

Questi sono stati suddivisi in ulteriori sottofasi, che verranno descritte nel dettaglio nei paragrafi successivi.

3.1.1 Installazione di un server Teleport

Il nodo host è stato configurato con un sistema operativo Ubuntu 22.04 LTS, su cui è stato installato Teleport. L'installazione è stata effettuata seguendo la guida ufficiale [Tel23a]. Per poter installare il servizio è stato necessario configurare un file di configurazione `teleport.yaml` in cui sono stati specificati i parametri di configurazione del servizio. In particolare, è stato necessario configurare il servizio per poter avviare Teleport all'avvio del sistema, e per poter abilitare l'accesso HTTPS al servizio, è stato necessario creare un certificato SSL tramite `Let's Encrypt`. Inoltre, è stato necessario configurare il firewall per permettere l'accesso al servizio. Infine, è stato necessario installare anche il servizio di autenticazione e di proxy per poter permettere l'accesso da remoto.

Una volta completata con successo l'installazione del nodo host, è stato possibile approfondire la comprensione del funzionamento del servizio stesso. Questo ha permesso di personalizzare ulteriormente l'ambiente per rispondere a specifiche esigenze, come l'integrazione dell'autenticazione tramite RADIUS con un server Active Directory.

Per implementare l'autenticazione RADIUS con un server AD, è stato necessario esaminare il codice sorgente di Teleport. Modificando il codice, è stato possibile aggiungere il supporto per il protocollo RADIUS e configurare l'integrazione con il server AD per la gestione degli utenti.

Questa personalizzazione ha consentito di estendere le capacità di Teleport, rendendolo compatibile con l'infrastruttura esistente basata su AD.

3.1.2 Implementazione dell'autenticazione RADIUS

È stato necessario configurare AD per poter fungere da server RADIUS. Per motivi di semplicità, è stato deciso di utilizzare il metodo di autenticazione Password Authentication Protocol (PAP) per la comunicazione tra Teleport e AD. PAP è

un protocollo di autenticazione che invia le credenziali in chiaro, senza alcuna forma di crittografia. Non è considerato sicuro, ma è stato sufficiente per dimostrare la fattibilità dell'integrazione dei due servizi.

Per permettere l'autenticazione tramite RADIUS, è stato intercettato il flusso di autenticazione e modificato per poter inviare le credenziali al server AD tramite il suddetto protocollo. Tuttavia, ciò rende il servizio autenticabile solo tramite questo metodo, escludendo altre modalità di autenticazione. Si è deciso di percorrere questa strada per verificare la fattibilità dell'integrazione, ma in un contesto reale sarebbe necessario implementare un sistema di scelta del metodo di autenticazione.

Per abilitare l'autenticazione RADIUS in Teleport, è stata introdotta una nuova funzione denominata `authenticateWithRadius`. Questa funzione gestisce il processo di autenticazione tramite il protocollo RADIUS e viene successivamente utilizzata all'interno della funzione `checkPasswordWOToken`, che verifica la password dell'utente.

Listing 3.1: Funzione di autenticazione tramite RADIUS

```
import (
  "layeh.com/radius"
  R "layeh.com/radius/rfc2865"
)

func authenticateWithRadius(user string, password []byte) error {
  passwordString := string(password[:])
  packet := radius.New(radius.CodeAccessRequest, []byte(`1234567890`))
  R.UserName_SetString(packet, user)
  R.UserPassword_SetString(packet, passwordString)

  response, err := radius.Exchange(context.Background(), packet, "{radius_address
      }:{port}")
  if err != nil {
    return err
  }
  if response.Code == radius.CodeAccessAccept {
    return nil
  } else {
    return fmt.Errorf("radius authentication failed")
  }
}
```

La funzione `authenticateWithRadius` riceve il nome utente e la password in formato di array di byte, converte la password in una stringa e crea un pacchetto RADIUS di tipo AccessRequest. Successivamente, imposta il nome utente e la password nel pacchetto RADIUS e lo scambia con il server RADIUS specificato. Se il server risponde con un codice di accettazione, l'autenticazione tramite RADIUS è considerata riuscita.

Listing 3.2: Funzione di verifica della password

```
func (a *Server) checkPasswordWOToken(user string, password []byte) error {
  const errMsg = "invalid username or password"
  const radiusErrMsg = "radius authentication failed"

  radiusErr := authenticateWithRadius(user, password)
  if radiusErr != nil {
    return trace.BadParameter(radiusErrMsg)
  }

  err := services.VerifyPassword(password)
  if err != nil {
    return trace.BadParameter(errMsg)
  }

  hash, err := a.GetPasswordHash(user)
  if err != nil && !trace.IsNotFound(err) {
    return trace.Wrap(err)
  }
  userFound := true
  if trace.IsNotFound(err) {
    userFound = false
    log.Debugf("Password for username %q not found, using fake hash to mitigate
        timing attacks.", user)
    hash = fakePasswordHash
  }

  if !userFound {
    // create user if it does not exist
    user, err := types.NewUser(user)
    if err != nil {
      return trace.Wrap(err)
    }

    user.SetRoles([]string{"access"})

    // create a logins variable to store the user's logins possible names, and
        insert its own name
    logins := []string{user.GetName()}
    user.SetLogins(logins)
```

```
38      user.SetWindowsLogins(logins)
39
40      user.SetCreatedBy(types.CreatedBy{
41        User: types.UserRef{Name: "teleport"},
42        Time: time.Now().UTC(),
43      })
44
45      ctx := context.Background()
46      if _, err := a.CreateUser(ctx, user); err != nil {
47        return trace.Wrap(err)
48      }
49      if err := a.UpsertPassword(user.GetName(), password); err != nil {
50        return trace.Wrap(err)
51      }
52      return nil
53    }
54
55    if err = bcrypt.CompareHashAndPassword(hash, password); err != nil {
56      log.Debugf("Password for %q does not match", user)
57      return trace.BadParameter(errMsg)
58    }
59
60    if !userFound {
61      return trace.BadParameter(errMsg)
62    }
63
64    return nil
65  }
```

La funzione `checkPasswordWOToken` utilizza la funzione di autenticazione tramite RADIUS prima di procedere con la verifica della password. Se l'autenticazione tramite RADIUS ha successo, la funzione procede a verificare ulteriormente la password dell'utente. In caso contrario, viene restituito un errore specifico. Nel caso l'utente sia correttamente autenticato da AD ma non esiste all'interno del database di Teleport, la funzione procede alla creazione di un nuovo utente.

3.1.3 Installazione dei client

Una volta completata l'installazione del server, è stato necessario installare i client su alcune macchine virtuali per poter testare il funzionamento del servizio. Su ciascuno di essi è stato installato un servizio diverso:

- Un servizio di SSH, per poter testare l'accesso alla console di un server.

- Un servizio web, per poter testare l'accesso a un'applicazione web;
- Un servizio di database, per poter testare l'accesso a un database;
- Un servizio di RDP per poter testare l'accesso a un server Windows.

Servizio di SSH

Il servizio di esposizione di SSH consente di accedere in modo sicuro alle macchine all'interno di una rete locale, consentendo un accesso controllato da parte di utenti autorizzati. Attraverso questo servizio è possibile accedere a una console e all'albero di file di una macchina utilizzando anche il protocollo Secure File Transfer Protocol (SFTP).

Servizio web

Per testare l'accesso a un'applicazione web, è stato necessario installare un servizio web su una macchina virtuale. Inserendo, nel file di configurazione del client Teleport, il nome e l'URL del servizio, è stato possibile testare l'accesso al servizio tramite Teleport. Tramite la Web UI di Teleport, è stato possibile collegarsi al servizio web e navigare nell'applicazione.

Servizio di database

Il servizio di esposizione di database consente di esporre in modo sicuro un database all'esterno della rete locale, consentendo un accesso controllato da parte di utenti autorizzati. Attraverso la configurazione del client Teleport, è stato possibile esporre un database MySQL al quale è possibile accedere sia tramite console che tramite un'applicazione di amministrazione di database come *DBeaver*.

Servizio di RDP

Per testare l'accesso a un server Windows tramite il protocollo RDP, è stato configurato il servizio di RDP su una macchina virtuale. Utilizzando il client Teleport, è possibile creare un client RDP che espone o singoli desktop Windows o un intero sistema Active Directory.

Nel caso si scelga di condividere un singolo desktop, il single sign-on non sarà disponibile e sarà necessario inserire la password per l'account desiderato. Tuttavia, se si opta per l'esposizione di un sistema AD, si avrà la possibilità di accedere in single sign-on a tutti i nodi collegati al dominio AD esposto.

Data l'importanza di questo servizio nella presente tesi, è stato particolarmente rilevante focalizzarsi sull'aspetto del single sign-on. Ciò consente agli utenti di accedere sia a Teleport che alle risorse RDP con il proprio profilo universitario senza la necessità di inserire la password più volte.

3.2 Creazione delle immagini

Come annunciato in precedenza, dopo l'installazione di Teleport su una macchina fisica, è stato eseguito il processo di creazione delle immagini relative ai componenti descritti in fase progettuale.

Questo processo è avvenuto per il nodo principale di Teleport, il quale gestisce l'autenticazione, l'accesso ai servizi, la gestione degli utenti e dell'intero *cluster*. Un cluster è un insieme di nodi che condividono le stesse informazioni di configurazione e che sono in grado di comunicare tra loro. Teleport permette di creare un cluster di nodi per poter gestire l'accesso a più servizi e risorse all'interno di un'organizzazione.

Per i nodi client, non è stato necessario creare immagini personalizzate, poiché questi, attraverso il loro file di configurazione, possono comunicare con il nodo principale ed esporre i servizi desiderati tramite esso.

Vengono di seguito descritte le immagini create per ciascun componente e microservizio.

3.2.1 Immagine di Teleport

Per la generazione dell'immagine di Teleport, non è stato possibile utilizzare l'immagine ufficiale in quanto non supporta l'autenticazione tramite RADIUS. È stato quindi necessario creare una nuova immagine personalizzata che includa il codice sorgente modificato per supportare l'autenticazione RADIUS. Per fare ciò è stato utilizzato un file `Dockerfile`, che contiene le istruzioni per la creazione dell'im-

magine. Il Dockerfile è stato configurato per eseguire un controllo iniziale per verificare se l'utente amministratore esiste all'interno del database di Teleport. Se l'utente non esiste, viene creato automaticamente un nuovo utente amministratore. Per impostare le credenziali si dovrà accedere a un link generato automaticamente all'avvio del container.

È stato creato un file `docker-compose.yml` per definire la composizione dell'applicazione e per avviare il servizio. All'interno del servizio sono stati definiti diversi volumi, ovvero directory all'interno del container che consentono la persistenza dei dati, in quanto separate dal ciclo di vita del container stesso. Questi volumi contengono i dati di configurazione, i certificati SSL, gli script di inizializzazione e i dati del software. In questo modo, i dati sono conservati anche in caso di rimozione o riavvio del container. Il servizio è stato reso dipendente da PostgreSQL, in modo che possa comunicare con il database.

Il file di configurazione di Teleport permette di definire le impostazioni del servizio. Nel nostro caso, si è deciso di definire:

- lo storage di backend come PostgreSQL, per permettere la memorizzazione dei dati di autenticazione e configurazione;
- il servizio di autenticazione, che permette l'autenticazione di utenti e nodi Teleport;
- il servizio di SSH, che permette l'accesso alla console del server;
- il servizio di proxy, che permette l'accesso al servizio attraverso un dominio personalizzato e attraverso una connessione sicura.

Viene riportato di seguito il contenuto del file di configurazione di Teleport:

Listing 3.3: File di configurazione di Teleport

```
version: v3
teleport:
  nodename: teleport-build
  data_dir: /var/lib/teleport
  storage:
    type: postgresql
```

```
7       conn_string: "postgresql://teleport@postgres:5432/
            teleport_backend?sslcert=/certs/post_certs/client.crt&sslkey=/
            certs/post_certs/client.key&sslrootcert=/certs/post_certs/ca.
            crt&sslmode=verify-full&pool_max_conns=20"
8     log:
9       output: stderr
10      severity: INFO
11      format:
12        output: text
13    ca_pin: ""
14    diag_addr: ""
15  auth_service:
16    enabled: "yes"
17    listen_addr: 0.0.0.0:3025
18    tokens:
19    - "node,App,Db,WindowsDesktop:michi-teleport"
20    proxy_listener_mode: multiplex
21    authentication:
22      second_factor: off
23  ssh_service:
24    enabled: "yes"
25    commands:
26    - name: hostname
27      command: [hostname]
28      period: 1m0s
29  proxy_service:
30    enabled: "yes"
31    public_addr: teleport.campusfc.unibo.it:3080
32    https_cert_file: /certs/tcerts/fullchain.pem
33    https_key_file: /certs/tcerts/privkey.pem
34    https_keypairs: []
35    https_keypairs_reload_interval: 0s
36    acme: {}
```

3.2.2 Immagine di PostrgreSQL

Per la generazione dell'immagine di PostgreSQL, è stato creato un `Dockerfile` che contiene le istruzioni per la creazione dell'immagine partendo dall'immagine uffi-

ciale di PostgreSQL. Per permettere la comunicazione tra Teleport e PostgreSQL, è stato necessario installare un plugin richiesto della guida ufficiale di Teleport [Tel23c]. Tramite uno script si sono dovuti creare dei certificati SSL per permettere la comunicazione sicura tra i due servizi. Questi certificati sono stati inseriti all'interno dell'immagine di PostgreSQL e di Teleport.

All'interno del file `docker-compose.yml` è stato definito anche il servizio di PostgreSQL, specificando i volumi necessari per la persistenza dei dati, per gli script di inizializzazione e le variabili d'ambiente per la configurazione.

Lo script di inizializzazione di PostgreSQL è stato creato per modificare la configurazione del database attraverso due file di configurazione: `pg_hba.conf` e `postgresql.conf`. In questi si definisce la configurazione del database, le regole di accesso e le impostazioni di sicurezza. Riporto di seguito il contenuto del file di configurazione di PostgreSQL.

Listing 3.4: Script di inizializzazione di PostgreSQL

```
#!/bin/bash
set -e

# Sovrascrive il file pg_hba.conf
cat > /var/lib/postgresql/data/pg_hba.conf <<EOF
# TYPE  DATABASE        USER            CIDR-ADDRESS            METHOD
local   all             all                                     trust
hostssl all             all             ::/0                    cert
hostssl all             all             0.0.0.0/0               cert
EOF

# Sovrascrive il file postgresql.conf
cat > /var/lib/postgresql/data/postgresql.conf <<EOF
listen_addresses = '*'
port = 5432
max_connections = 20
shared_buffers = 128MB
temp_buffers = 8MB
work_mem = 4MB

wal_level=logical
max_replication_slots=10

ssl=on
ssl_ca_file='/certs/ca.crt'
ssl_cert_file='/certs/server.crt'
ssl_key_file='/certs/server.key'
EOF
```

3.2.3 Immagine di Nginx

Come servizio di Nginx è stata utilizzata l'immagine ufficiale disponibile su Docker Hub. Nginx è stato configurato come proxy inverso per permettere l'accesso al servizio Teleport tramite un dominio personalizzato e per gestire il traffico web.

Per creare il file di configurazione del reverse proxy, è stato necessario creare un file `teleport.campusfc.unibo.it.conf` all'interno della directory della macchina host `/etc/nginx/conf.d`. Questo file contiene le istruzioni per il proxy inverso, che permettono di reindirizzare il traffico HTTP e HTTPS al servizio Teleport.

È stato poi necessario creare i certificati SSL per permettere la comunicazione sicura tramite HTTPS [oF21]. Questo è stato fatto tramite *Let's Encrypt*, un servizio che permette di creare certificati SSL gratuiti e di rinnovarli automaticamente. Questi certificati sono stati inseriti all'interno dell'immagine di Nginx e di Teleport tramite la mappatura dei volumi.

3.3 Integrazione tramite Docker Compose

I container creati in precedenza sono stati integrati mediante Docker Compose, un potente strumento per la definizione e l'orchestrazione di applicazioni multi-container. È stato creato un file YAML denominato `docker-compose.yml` per definire la composizione dell'applicazione. Tale file specifica:

- I servizi da avviare, ossia i container necessari per l'esecuzione dell'applicazione.
- Le immagini Docker da utilizzare per ciascun servizio.
- Le dipendenze tra i servizi.
- Le configurazioni di rete e dei volumi.

Per la persistenza dei dati, sono stati definiti volumi per ciascun servizio che necessita di archiviare informazioni. I volumi sono stati creati come volumi anonimi, non associati ad alcun ciclo di vita del container, garantendo la persistenza dei dati anche in caso di rimozione o riavvio del container.

I container sono stati collegati a una rete condivisa per permettere la comunicazione inter-container. La rete è stata configurata con un nome specifico per identificarla all'interno dell'ambiente Docker.

Oltre a definire i servizi e le loro configurazioni, sono state specificate le dipendenze tra i container. Nel caso specifico, il container di Teleport è dipendente da:

- PostgreSQL: Per la memorizzazione dei dati di autenticazione e configurazione.
- Nginx: Per il proxy inverso e la gestione del traffico web.

3.3.1 File di dispiegamento

Sono riportati di seguito i file di configurazione per Teleport e PostgreSQL:

Listing 3.5: Dockerfile di Teleport

```
FROM ubuntu:22.04

RUN apt-get update
RUN apt-get install -y netcat

WORKDIR /app

CMD ["/bin/bash", "-c", "/docker-entrypoint-init.d/create_admin.sh & ./teleport
    start -c ./teleport.yaml && fg"]
```

Listing 3.6: Dockerfile di PostgreSQL

```
FROM postgres:latest
RUN apt-get update
RUN apt-get install -y postgresql-16-wal2json
COPY --chown=999:999 ./certs/post_certs /certs
```

Listing 3.7: docker-compose.yml

```
version: '3'

services:
  teleport:
    build:
      context: .
      dockerfile: Dockerfile.teleport
```

```
    volumes:
      - ./teleport/bin:/app
      - ./teleport/data:/var/lib/teleport
      - ./certs:/certs
      - ./teleport/init:/docker-entrypoint-init.d
    ports:
      - "3080:3080"
      - "3022:3022"
      - "3025:3025"
    depends_on:
      - postgres
    restart: always

  postgres:
    build:
      context: .
      dockerfile: Dockerfile.postgresql
    volumes:
      - ./postgresql/data:/var/lib/postgresql/data
      - ./postgresql/init:/docker-entrypoint-initdb.d
    environment:
      POSTGRES_DB: teleport-db
      POSTGRES_USER: michi
      POSTGRES_PASSWORD: 3600MMcm
    ports:
      - "5432:5432"
    restart: always

  nginx:
    image: nginx:latest
    volumes:
      - /etc/nginx/conf.d:/etc/nginx/conf.d
      - /etc/letsencrypt:/etc/letsencrypt
    ports:
      - "80:80"
      - "443:443"
    restart: always
```

Capitolo 4

Testing

La fase di testing di Teleport si pone due obiettivi principali: verificare il corretto funzionamento del sistema e analizzarne le prestazioni. Per raggiungere questi obiettivi, è necessario definire un'architettura di test che permetta di eseguire test ripetibili e confrontabili. In questo capitolo, verrà descritta l'architettura e verranno descritti i test eseguiti e i risultati ottenuti.

Il test è stato realizzato seguendo i seguenti passaggi:

1. Implementazione di una singola istanza di test tramite Selenium;
2. Parallelizzazione del test tramite thread e TestNG;
3. Esternalizzazione del test tramite RemoteWebDriver.

4.1 Fasi di creazione dell'architettura di test

4.1.1 Singola istanza di test

Per testare Teleport, è stato necessario scegliere un ambiente di test che permettesse di simulare l'interazione di un utente con l'interfaccia web. La scelta è ricaduta su **Selenium**, un insieme di librerie sviluppato inizialmente da Jason Huggins nel 2004 per ThoughtWorks e successivamente rilasciato come software libero. Selenium permette l'interazione con le pagine web attraverso un'API che manipola il DOM. Questo, infatti, è in grado di simulare azioni come il click di un pulsante,

l'inserimento di testo in un campo di input, la selezione di un elemento da una lista, ecc. Questi elementi sono selezionabili attraverso un selettore CSS o XPath.

Il primo passo è stato quello di creare un test che simulasse l'accesso alla pagina di login di Teleport, l'inserimento delle credenziali e il click sul bottone di login:

1. Collegamento al server di test all'indirizzo https://teleport.campusfc.unibo.it sulla porta 3080;
2. Inserimento delle credenziali di test;
3. Click sul bottone di login.

Listing 4.1: LocalLoginTest.java

```
1  package org.example;
2
3  import org.openqa.selenium.WebDriver;
4  import org.openqa.selenium.chrome.ChromeDriver;
5  import org.openqa.selenium.chrome.ChromeOptions;
6
7  import java.util.concurrent.atomic.AtomicInteger;
8
9  public class LocalLoginTest extends BaseLoginTest {
10
11     public static void main(String[] args) {
12         String username = "test1";
13         String password = "TesT09785!";
14
15         System.setProperty("webdriver.chrome.driver", "/opt/homebrew/bin/
               chromedriver");
16
17         ChromeOptions options = new ChromeOptions();
18         WebDriver wd = new ChromeDriver(options);
19
20         runTest(wd, username, password);
21     }
22 }
```

Listing 4.2: BaseLoginTest.java

```
1  package org.example;
2
3  import org.openqa.selenium.By;
4  import org.openqa.selenium.WebDriver;
5
6  import java.util.Random;
```

```
7
8  public abstract class BaseLoginTest {
9
10     protected void runTest(WebDriver wd, String username, String password, int
           userNum) {
11         String loginXpath = "//*[@id=\"app\"]/div/div/div[2]/div/div/div/form/
               button";
12         String usernameXpath = "//*[@id=\"app\"]/div/div/div[2]/div/div/div/form/
               div[1]/label/input";
13         String passwordXpath = "//*[@id=\"app\"]/div/div/div[2]/div/div/div/form/
               div[2]/div/label/input";
14
15         wd.get("https://teleport.campusfc.unibo.it:3080");
16
17         try {
18             Thread.sleep(5000);
19         } catch (InterruptedException e) {
20             throw new RuntimeException(e);
21         }
22
23         wd.findElement(By.xpath(usernameXpath)).sendKeys(username);
24         wd.findElement(By.xpath(passwordXpath)).sendKeys(password);
25
26         try {
27             Thread.sleep(1000 + (new Random().nextInt(100, 3000)));
28         } catch (InterruptedException e) {
29             throw new RuntimeException(e);
30         }
31
32         wd.findElement(By.xpath(loginXpath)).click();
33
34         try {
35             Thread.sleep(30000);
36         } catch (InterruptedException e) {
37             throw new RuntimeException(e);
38         }
39
40         wd.quit();
41     }
42 }
```

Il codice riportato in 4.1 mostra il test di login locale, mentre quello in 4.2 mostra la classe base che contiene i metodi comuni a tutti i test. Il test è stato eseguito con successo e ha permesso di verificare il corretto funzionamento del sistema. Una sola istanza di test, però, non è sufficiente per verificare le prestazioni del sistema, tanto meno senza ulteriori interazioni, come il collegamento in RDP ad una macchina remota presente nel cluster. Si è quindi deciso di parallelizzare

il test e di eseguirlo utilizzando ulteriori utenti.

4.1.2 Parallelizzazione del test

Inizialmente, non sapendo quanti utenti fossero necessari per testare Teleport, si è deciso di creare uno script in PowerShell, sul Domain Controller (DC), che creasse un numero variabile di utenti. Allego lo script in 4.3.

Listing 4.3: create80Users.ps1

```
Import-Module ActiveDirectory

# Imposta il dominio
$domain = "teleport.local"
$password = "TesT09785!"

for ($i = 1; $i -le 80; $i++) {
    $username = "test$i"

    Write-Host $username
    Write-Host $password

    # Crea l'utente in Active Directory
    New-ADUser -Name $username -SamAccountName $username -
        UserPrincipalName "$username@$domain" -AccountPassword (
        ConvertTo-SecureString $password -AsPlainText -Force) -Enabled
         $true -PasswordNeverExpires $true -ChangePasswordAtLogon
        $false -PassThru

    Write-Host "Utente $username creato con successo."
}
```

Il test è stato parallelizzato utilizzando **TestNG** [Tes23], un framework di testing per Java. TestNG è pensato per semplificare vari aspetti riguardanti il testing, come lo *unit test*, il test funzionale, il test di integrazione, ecc...

Codice di Implementazione

Il test parallelo è stato configurato per creare due istanze di WebDriver, ciascuna eseguita in un thread separato. Questo simula due utenti che accedono contem-

poraneamente al sistema. Ogni sessione di WebDriver esegue una serie di azioni predefinite che includono il login e la connessione remota attraverso il sistema Teleport. Il codice sotto mostra il test è stato strutturato:

Listing 4.4: ParallelLocalLoginTest.java

```java
package org.example;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.testng.annotations.Test;

import java.util.concurrent.atomic.AtomicInteger;

public class LocalLoginTest extends BaseLoginTest {
    private final AtomicInteger sequence = new AtomicInteger(0);
    private final int containerNum = 2;

    @Test(threadPoolSize = containerNum, invocationCount = containerNum)
    public void testLogin() {
        int count = sequence.incrementAndGet();
        String username = "test" + count;
        String password = "TesT09785!";

        System.out.println("Running locally on macOS...");

        System.setProperty("webdriver.chrome.driver", "/opt/homebrew/bin/
            chromedriver");

        ChromeOptions options = new ChromeOptions();
        WebDriver wd = new ChromeDriver(options);

        runTest(wd, username, password, count);
    }
}
```

Listing 4.5: ParallelBaseLoginTest.java

```java
package org.example;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;

import java.util.Random;
import java.util.Set;

public abstract class BaseLoginTest {

```

```
11      protected void runTest(WebDriver wd, String username, String password, int
            userNum) {
12          String loginXpath = "//*[@id=\"app\"]/div/div/div[2]/div/div/div/form/
                button";
13          String usernameXpath = "//*[@id=\"app\"]/div/div/div[2]/div/div/div/form/
                div[1]/label/input";
14          String passwordXpath = "//*[@id=\"app\"]/div/div/div[2]/div/div/div/form/
                div[2]/div/label/input";
15          String usernameButtonXpath = "//*[@id=\"app\"]/div/nav/div[3]/div[2]/div
                [1]";
16          String rdpConnectButtonXPath = "//*[@id=\"app\"]/div/div/div/div/div/div
                [2]/div/div[5]/div[1]/div/div/div[3]/div[1]/button";
17          String aHrefUserXPath = "/html/body/div[2]/div[2]/div/div/a[2]";
18
19          wd.get("https://teleport.campusfc.unibo.it:3080");
20          String originalWindowHandle = wd.getWindowHandle();
21
22          try {
23              Thread.sleep(5000);
24          } catch (InterruptedException e) {
25              throw new RuntimeException(e);
26          }
27
28          // Enter username and password
29          wd.findElement(By.xpath(usernameXpath)).sendKeys(username);
30          wd.findElement(By.xpath(passwordXpath)).sendKeys(password);
31
32          try {
33              Thread.sleep(1000 + (new Random().nextInt(100, 3000)));
34          } catch (InterruptedException e) {
35              throw new RuntimeException(e);
36          }
37
38          wd.findElement(By.xpath(loginXpath)).click();
39
40          boolean elementFound = false;
41          for (int i = 0; i < 4; i++) {
42              while (!elementFound) {
43                  try {
44                      wd.findElement(By.xpath(rdpConnectButtonXPath));
45                      elementFound = true;
46                  } catch (Exception e) {
47                      try {
48                          Thread.sleep(10000);
49                      } catch (InterruptedException ex) {
50                          throw new RuntimeException(ex);
51                      }
52                  }
53              }
54
```

```
55              wd.findElement(By.xpath(rdpConnectButtonXPath)).click();
56              wd.findElement(By.xpath(aHrefUserXPath)).click();
57
58              try {
59                  Thread.sleep(300000);
60              } catch (InterruptedException e) {
61                  throw new RuntimeException(e);
62              }
63
64              Set<String> windowHandles = wd.getWindowHandles();
65              String newTabHandle = windowHandles.toArray()[windowHandles.size() -
                    1].toString();
66              wd.switchTo().window(newTabHandle);
67              wd.close();
68
69              wd.switchTo().window(originalWindowHandle);
70          }
71
72          wd.findElement(By.xpath(usernameButtonXpath)).click();
73          wd.quit();
74      }
75  }
```

Nel codice sopra, `@Test(threadPoolSize = 2, invocationCount = 2)`, annotazione di TestNG, specifica che il test dovrebbe essere eseguito con due thread, ognuno dei quali esegue una volta il metodo `testLogin()`. Questo assicura che i test vengano eseguiti in parallelo su due thread separati, simulando due utenti che eseguono il test simultaneamente.

La soluzione proposta è scalabile, flessibile e riutilizzabile con un numero variabile di utenti. Tuttavia, il test è ancora eseguito localmente, il che può portare a risultati non affidabili a causa di variabili come la potenza di calcolo della macchina, la connessione internet, e altri fattori. Per risolvere questo problema, è necessario eseguire il test in un ambiente remoto, come illustrato nella sezione successiva.

4.1.3 Test dockerizzato, parallelo e remoto

L'ultima fase di implementazione del test è stata ultimata grazie all'adozione di *Selenium Remote WebDriver* [Sel24], un driver che permette di eseguire test su un server remoto. Abbiamo quindi la possibilità di eseguire test in un ambiente controllato, senza influenze esterne. Il server dedicato è stato configurato su un

VM con sistema operativo Ubuntu 22.04 e risorse hardware sufficienti per eseguire i test, anche con un numero elevato di utenti (64 GB di RAM).

Per creare un ambiente controllato, sono stati utilizzati i *Container Docker*. La creazione di questi container è stata automatizzata attraverso uno script bash, il quale crea container a partire da `selenium/standalone-chrome-debug`, l'immagine ufficiale di Selenium per Chrome. Questa immagine contiene tutto il necessario per eseguire test con Selenium, incluso un ambiente grafico e un server VNC per visualizzare l'interfaccia grafica del browser. Configurando quindi il *mapping* delle porte è possibile accedere al server VNC e controllare il corretto funzionamento del test.

Gli script per la creazione e la distruzione dei container sono riportati di seguito:

Listing 4.6: runSelenium.sh

```
#!/bin/bash
source selenium.env

# Genera una lista di numeri da 1 a CONTAINER_NUM
seq_list=$(seq 1 $CONTAINER_NUM)

# Usa xargs per eseguire il comando docker run in parallelo
echo "$seq_list" | xargs -n 1 -P 0 -I {} bash -c '
    X={}
    echo "Starting container teleportTest$X"
    docker run -d -p$(( 50000+$X)):4444 -p $((50200+$X)):5900 -e VNC_NO_PASSWORD=1
        --shm-size=2g --name=teleportTest$X --rm selenium/standalone-chrome-debug
'
```

Listing 4.7: stopSelenium.sh

```
#!/bin/bash
source selenium.env

seq_list=$(seq 1 $CONTAINER_NUM)
echo "$seq_list" | xargs -n 1 -P 0 -I {} bash -c '
    X={}
    echo "Ending container teleportTest$X"
    docker stop teleportTest$X
'
```

Per rendere più efficiente il dispiegamento dei container, si è utilizzato `xargs` per eseguire comandi in parallelo. In questo modo, per qualsiasi numero di container, il tempo di esecuzione rimane simile.

Codice di Implementazione

Il codice di test è stato modificato per utilizzare il WebDriver remoto, come mostrato di seguito:

Listing 4.8: ParallelRemoteLoginTest.java

```
package org.example;

import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.CapabilityType;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.testng.annotations.Test;

import java.net.MalformedURLException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.util.concurrent.atomic.AtomicInteger;

public class RemoteLoginTest extends BaseLoginTest {
    private final AtomicInteger sequence = new AtomicInteger(0);
    private final int containerNum = 80;
    private static final String IP = "192.168.0.156";

    @Test(threadPoolSize = containerNum, invocationCount = containerNum)
    public void testLogin() {
        URL remoteUrl;
        int count = sequence.incrementAndGet();
        String username = "test" + count;
        String password = "TesT09785!";
        String port = String.valueOf(50000 + count);

        ChromeOptions options = new ChromeOptions();
        options.setCapability(CapabilityType.PLATFORM_NAME, "LINUX");
        options.setCapability(CapabilityType.BROWSER_NAME, "chrome");

        System.out.println("Running on remote machine: " + IP + " port: " + port);

        try {
            remoteUrl = new URI("http://" + IP + ":" + port + "/wd/hub").toURL();
        } catch (URISyntaxException | MalformedURLException e) {
            throw new RuntimeException("Invalid remote WebDriver URL", e);
        }

        RemoteWebDriver wd = new RemoteWebDriver(remoteUrl, options);
        runTest(wd, username, password, count);
    }
}
```

Avendo modularizzato il codice, `ParallelBaseLoginTest` 4.5 non ha subito modifiche.

4.2 Esecuzione del test

Elencati i passi per la creazione dell'architettura di test, è necessario eseguire il test per verificare il corretto funzionamento del sistema.

4.2.1 Scenario di test

Il test è stato eseguito con 2 utenti che accedono al sistema in parallelo. Questo, pur non essendo un test esaustivo, permette di verificare il corretto funzionamento del sistema e di analizzarne le prestazioni. È stato scelto di eseguire il test con solo 2 utenti in quanto per ogni utente, è necessario creare un nodo AD all'interno del reame. La necessità scaturisce dal fatto che ogni istanza di test mira ad aprire una connessione RDP con una macchina remota, la quale deve essere autenticata tramite AD. Questo, oltre a richiedere molto tempo, chiederebbe la creazione di un numero elevato di VM all'interno del server del campus di Cesena. Una situazione verosimile richiederebbe circa la creazione di 80 nodi AD.

4.2.2 Esecuzione e monitoraggio del test

Il test viene lanciato attraverso l'IDE IntelliJ IDEA sulla mia macchina locale che, utilizzando il WebDriver remoto, esegue il test sul server remoto, contattando i 2 container Docker creati. Questi, a loro volta, richiedono la pagina di login di Teleport all'indirizzo `https://teleport.campusfc.unibo.it:3080` e inseriscono le credenziali di test per ogni utente. A questo punto, ogni utente apre una connessione RDP con una macchina remota presente nel cluster e mantiene la connessione attiva per 5 minuti. Al termine del tempo, esegue il logout chiudento la sessione. Questo processo viene ripetuto quattro volte per ogni utente, in modo da monitorare le prestazioni del sistema nel tempo.

Il test, visibile in fig. 4.1, è stato eseguito con successo e i risultati sono stati raccolti e analizzati.

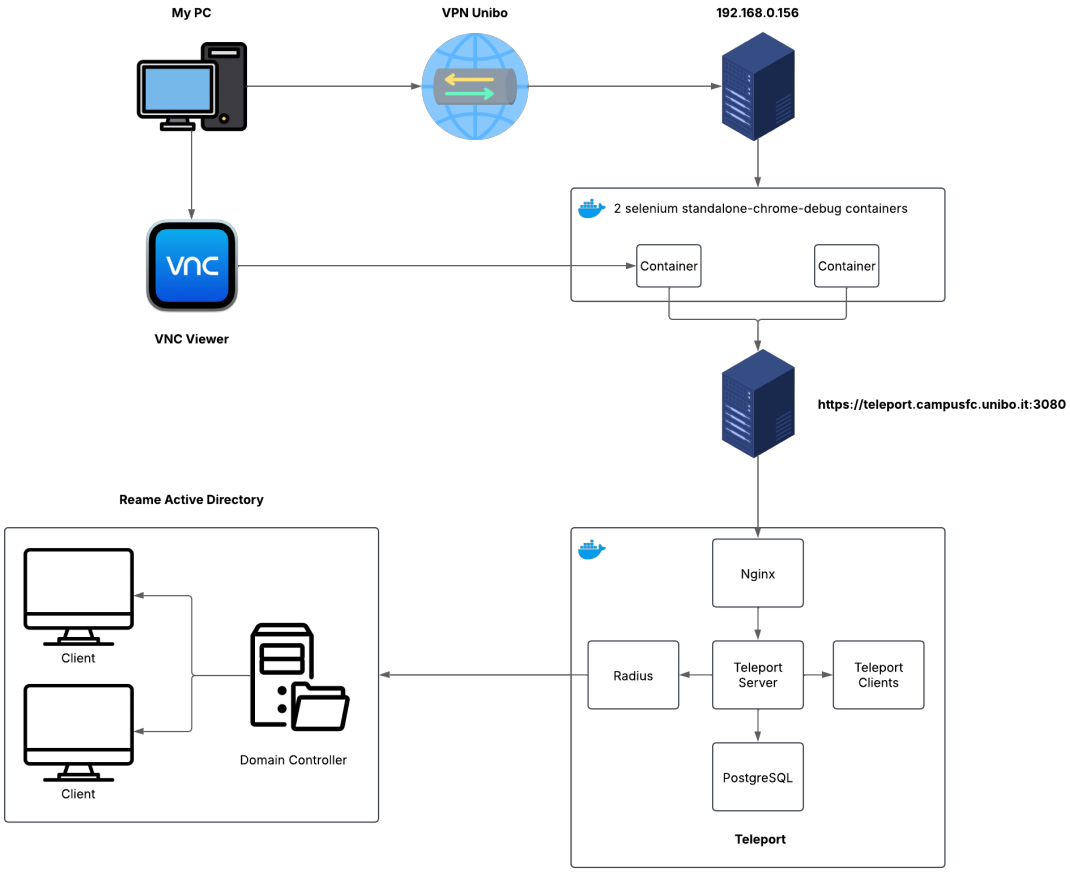

Figura 4.1: Architettura di test di Teleport

4.3 Analisi dei risultati

Per monitorare l'utilizzo delle risorse del server, è stato utilizzato **Grafana** [Gra24], un software open-source per la visualizzazione e il monitoraggio dei dati. Grafana permette di visualizzare i dati in tempo reale e di creare dashboard personalizzate collegnadosi a diversi software di monitoraggio, come **Prometheus** [Pro24].

Sono stati dispiegati, tramite l'utilizzo di *Docker Compose*, due container con le immagini ufficiali di Grafana e Prometheus. I dati sono stati analizzati su due macchine differenti:

- cluster principale di Teleport;
- nodo che si occupa del traffico RDP.

I parametri che sono stati ritenuti significativi per l'analisi sono:

- utilizzo della CPU;
- utilizzo della memoria;
- utilizzo della rete.

Si procede ora con l'analisi dei risultati ottenuti, corredati da grafici e spiegazioni.

4.3.1 Utilizzo della CPU

Sia il cluster principale di Teleport che il nodo che si occupa del traffico RDP hanno mostrato picchi di utilizzo della CPU ogni volta che gli utenti aprono una connessione RDP. Questo è dovuto al fatto che ogni connessione RDP richiede l'apertura di un nuovo processo che gestisce la connessione. L'aumento dell'utilizzo della CPU è quindi da attribuire a due fattori principali:

- l'allocazione di risorse per ogni cuova connessione;
- la gestione del traffico dati da parte del nodo che si occupa del traffico RDP.

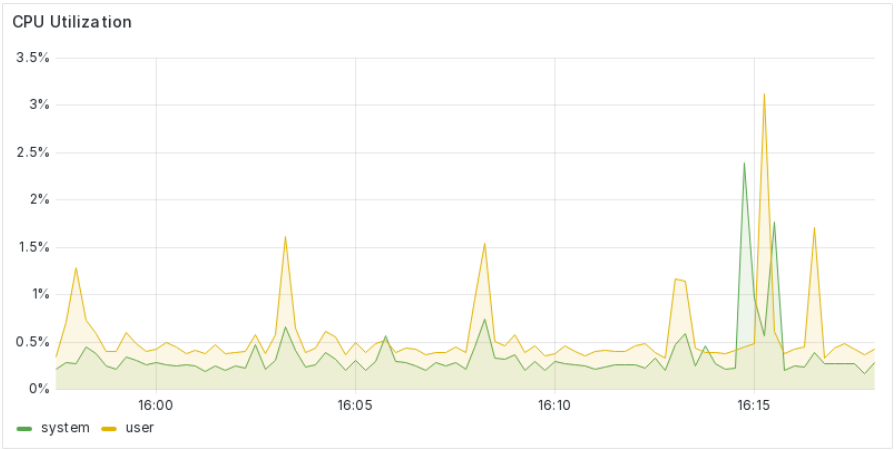

Figura 4.2: Utilizzo della CPU sul cluster principale di Teleport

Come si può notare dai grafici in fig. 4.2 e fig. 4.3, il cluster principale utilizza meno risorse rispetto al nodo che si occupa del traffico RDP. Questo è normale, in quanto il cluster principale si occupa principalmente della gestione centralizzata degli accessi e della comunicazione tra i nodi, mentre il nodo che si occupa del traffico RDP è responsabile della gestione diretta delle sessioni utente e del traffico dati.

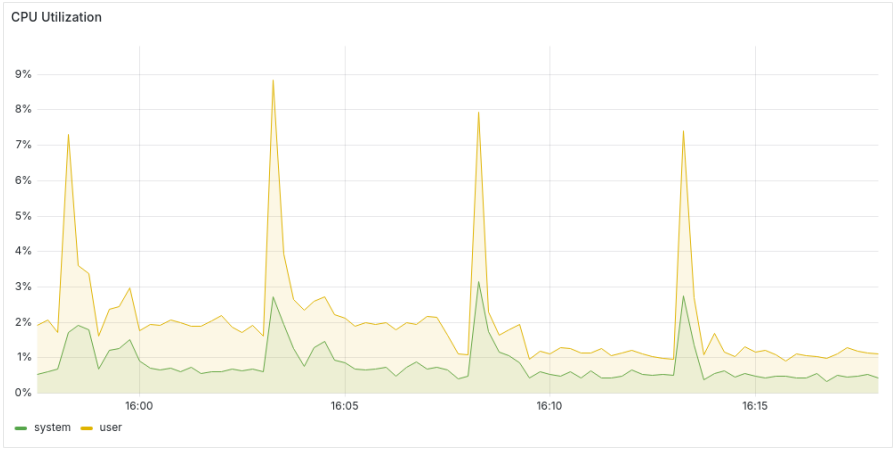

Figura 4.3: Utilizzo della CPU sul nodo che si occupa del traffico RDP

4.3.2 Utilizzo della memoria

L'analisi dell'utilizzo della memoria evidenzia un comportamento molto simile a quello della CPU. Sia il cluster principale di Teleport che il nodo che dedicato al traffico RDP mostrano variazioni significative riguardanti la quantità di memoria libera disponibile durante il test. Questo è dovuto al fatto che ogni connessione RDP richiede l'allocazione dinamica di risorse.

Figura 4.4: Utilizzo della memoria sul cluster principale di Teleport

Dal grafico in fig. 4.4, relativo all'utilizzo della memoria sul cluster principale di Teleport, si notano fluttuazioni della memoria libera disponibile meno marcate rispetto al nodo dedicato al traffico RDP, come mostrato in fig. 4.5. Questo è coerente con i ruoli dei due nodi, come spiegato nella sezione precedente 4.3.1.

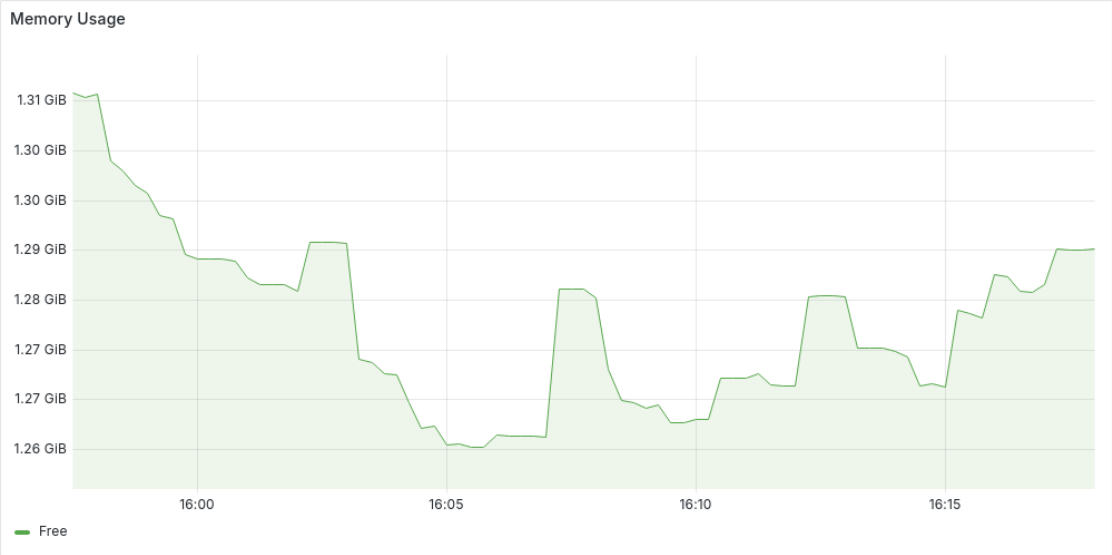

Figura 4.5: Utilizzo della memoria sul nodo che si occupa del traffico RDP

4.3.3 Utilizzo della rete

L'analisi dell'utilizzo della rete, per quanto riguarda il cluster principale e il nodo dedicato al traffico RDP, mostra un andamento diverso rispetto a CPU e memoria. L'utilizzo della rete è costante e non subisce variazioni significative durante il test. Nel primo grafico, fig. 4.6, si osserva un traffico di rete stabile e costante, con una velocità di trasmissione di circa **800-900 KB/s** (`eth0 transmit`) e una velocità di ricezione sotto ai **100 KB/s** (`eth0 receive`), significativamente più bassa.

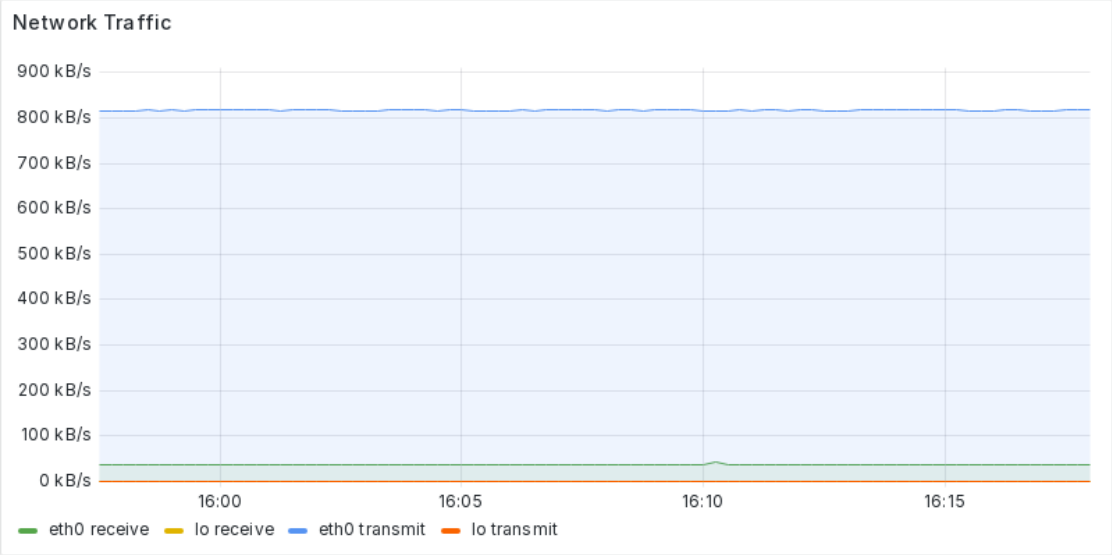

Figura 4.6: Utilizzo della rete sul cluster principale di Teleport

Nel secondo grafico, fig. 4.7, il comportamento è simile. Anche qui il traffico è costante, con valori di trasmissione leggermente più bassi rispetto al cluster principale, ma senza picchi che ci si potrebbe aspettare durante l'apertura di connessioni RDP.

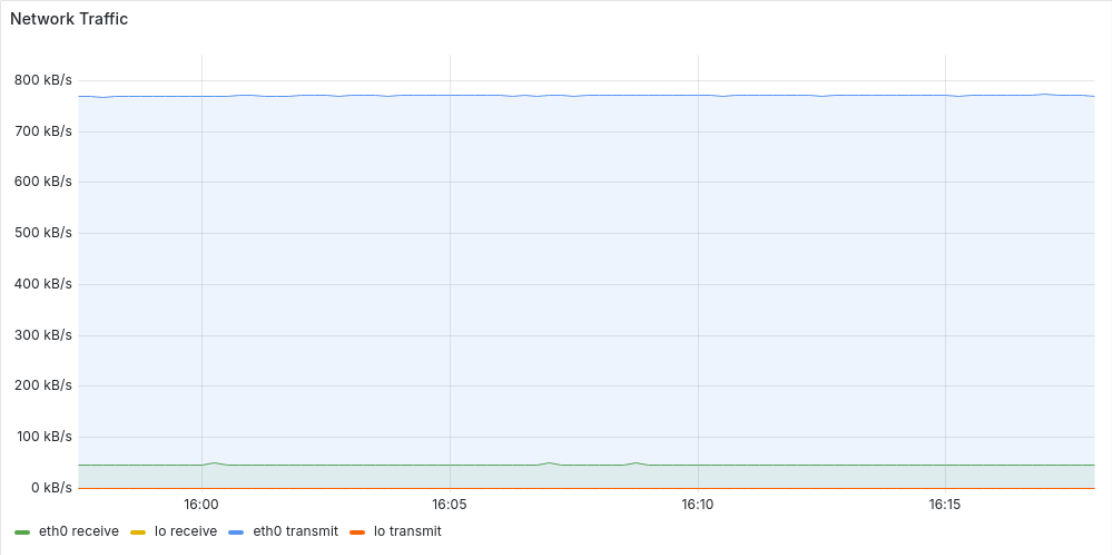

Figura 4.7: Utilizzo della rete sul nodo che si occupa del traffico RDP

4.3.4 Processi attivi

Quest'ultima analisi, relativa al numero di processi attivi sui due nodi, evidenzia comportamenti differenti tra il cluster principale e il nodo dedicato al traffico RDP. Il cluster principale, in fig. 4.8, mostra picchi di processi attivi generalmente intorno ai **1-3 processi**. Questo è coerente con il ruolo del cluster principale, che si occupa della gestione centralizzata degli accessi e della comunicazione tra i nodi.

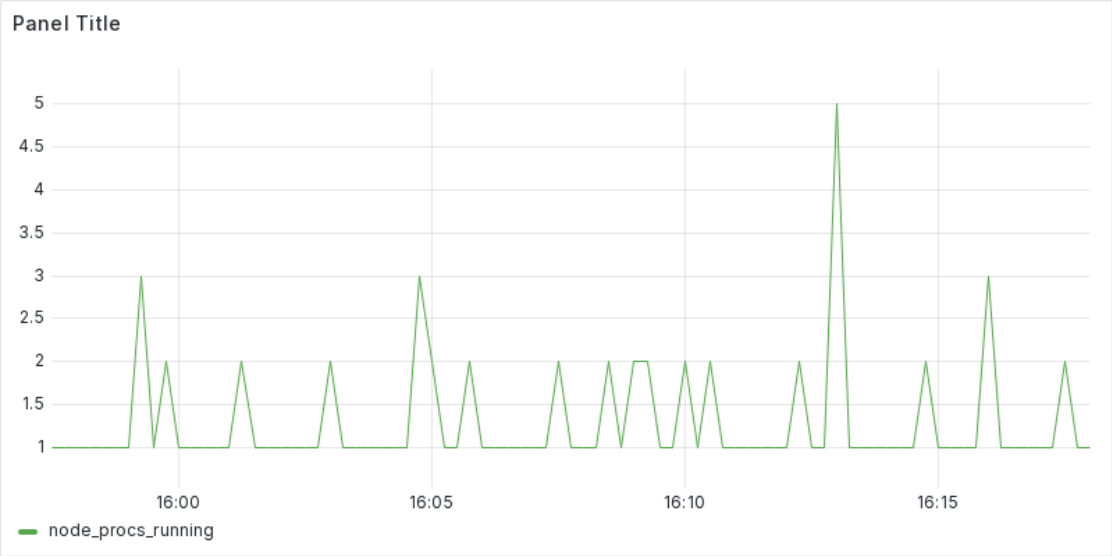

Figura 4.8: Processi attivi sul cluster principale di Teleport

Il nodo dedicato al traffico RDP, in fig. 4.9, mostra un comportamento diverso. Inizialmente, all'apertura delle prime 2 connessioni, si nota un picco di 4 processi attivi. Successivamente, sia durante l'apertura delle connessioni successive che non, si verificano picchi di **2-3 processi attivi**. Questo è coerente con il ruolo del nodo, che si occupa della gestione diretta delle sessioni utente e del traffico dati.

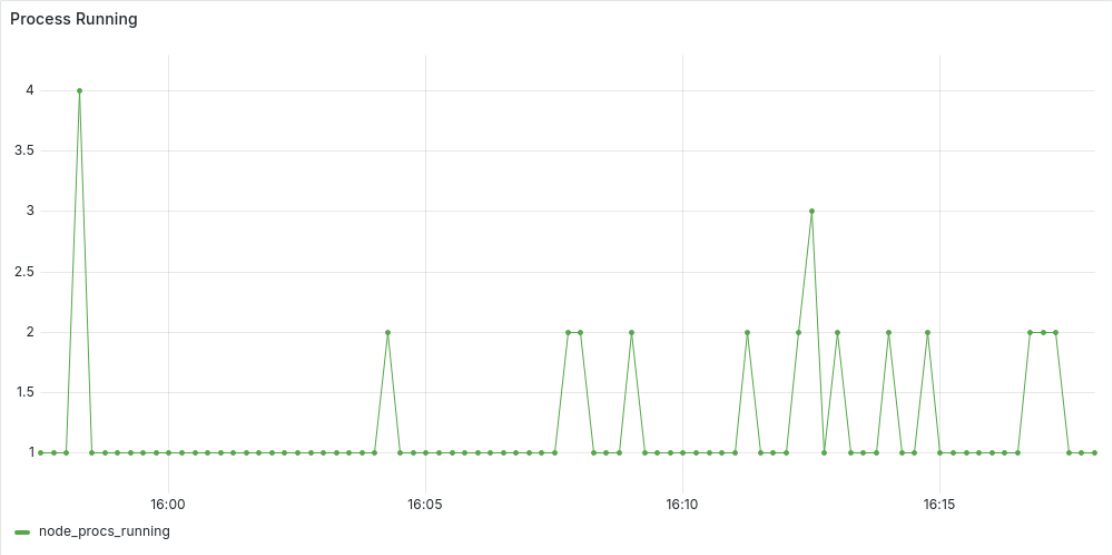

Figura 4.9: Processi attivi sul nodo che si occupa del traffico RDP

4.4 Conclusioni del test

Dai test eseguiti, si può concludere che Teleport è un sistema robusto e scalabile. I risultati ottenuti mostrano che il sistema è in grado di gestire più utenti contemporaneamente, senza subire rallentamenti significativi. L'utilizzo delle risorse, seppur testato con solo 2 utenti, è relativamente basso e sostenibile, anche con un numero maggiore di utenti, dai sistemi attualmente presenti all'interno del campus di Cesena.

Il test ha permesso inoltre di verificare il corretto funzionamento del metodo di autenticazione tramite RADIUS, introdotto nel capitolo 3 alla sezione 3.1.2. Questo permette quindi l'introduzione di **Teleport** all'interno del campus di Cesena, senza la necessità di modificare l'infrastruttura attuale.

Capitolo 5

Scenari d'uso

Teleport rappresenta una soluzione avanzata per la gestione degli accessi remoti e per la sicurezza informatica nel contesto accademico dell'Università di Bologna. La sua implementazione nel campus consente un accesso sicuro e controllato alle risorse IT, migliorando l'efficienza operativa di studenti, docenti e amministratori di sistema. In questo capitolo vengono analizzati diversi scenari d'uso per evidenziare le potenzialità di Teleport nella didattica, nella ricerca e nella gestione IT.

5.1 Gestione dell'accesso remoto a risorse accademiche

L'accesso a macchine fisiche e virtuali è una necessità fondamentale per laboratori didattici, progetti di ricerca e attività amministrative. Teleport semplifica questo processo eliminando la necessità di VPN complesse e garantendo una connessione sicura tramite autenticazione centralizzata.

5.1.1 Laboratori virtuali e cluster di calcolo

Molti corsi universitari richiedono l'uso di software specifici installati su server remoti o cluster di calcolo distribuito. Con Teleport, gli studenti possono accedere in sicurezza alle risorse necessarie per lo svolgimento di simulazioni ed esperimen-

ti, beneficiando di un sistema di autenticazione integrato con Active Directory e RADIUS. Questo permette di semplificare la gestione degli account e garantire un accesso controllato alle risorse.

5.1.2 Progetti di ricerca e accesso a dati sensibili

I ricercatori spesso lavorano con dati sensibili che richiedono un alto livello di sicurezza. Teleport consente di definire criteri di accesso granulari in base ai ruoli, garantendo che solo utenti autorizzati possano interagire con determinati server o database. Inoltre, la funzionalità di *audit logging* permette di monitorare le sessioni di accesso, aumentando la trasparenza e la sicurezza.

5.2 Didattica e supporto tecnico a distanza

L'adozione di Teleport migliora l'efficienza della didattica a distanza e del supporto tecnico, fornito dagli amministratori di sistema e dai docenti.

5.2.1 Accesso remoto a desktop e applicazioni

Grazie all'integrazione con il protocollo RDP e SSH, docenti e studenti possono connettersi alle proprie macchine o a risorse condivise del campus senza configurazioni complicate. Questo facilita la continuità didattica e consente agli studenti di eseguire esercitazioni pratiche anche al di fuori dell'università.

5.2.2 Assistenza tecnica e gestione remota

I tecnici IT possono fornire supporto agli utenti accedendo ai loro sistemi in modo sicuro tramite Teleport, senza la necessità di installare software aggiuntivi. Questa funzionalità è particolarmente utile in scenari di emergenza, come la risoluzione di problemi durante un esame o una lezione online.

5.3 Gestione centralizzata delle risorse IT

Teleport consente di amministrare in modo unificato l'accesso a un'ampia gamma di servizi IT all'interno del campus, migliorando la sicurezza e la gestione operativa.

5.3.1 Controllo degli accessi a server e database

Gli amministratori di sistema possono configurare facilmente criteri di accesso in base ai ruoli degli utenti, garantendo che solo personale autorizzato possa interagire con server, database e applicazioni web.

5.3.2 Monitoraggio e auditing delle sessioni

Un aspetto chiave della gestione IT è il monitoraggio delle attività degli utenti. Teleport fornisce un sistema avanzato di logging per tracciare gli accessi, registrare sessioni e rilevare eventuali attività sospette. Questo non solo migliora la sicurezza, ma aiuta anche nella conformità con le normative di protezione dei dati.

Capitolo 6

Conclusioni

Il progetto ha raggiunto gli obiettivi prefissati, rispettando tutti i requisiti stabiliti in fase di progettazione e implementazione. Dopo aver testato Teleport in un ambiente di sviluppo, si può concludere che la soluzione è pronta per essere integrata all'interno del campus dell'Università di Bologna.

Teleport rappresenta un'alternativa moderna, sicura e scalabile per la gestione degli accessi remoti e per il rafforzamento della sicurezza informatica, migliorando al contempo l'efficienza operativa di studenti, docenti e amministratori di sistema.

La natura open-source di Teleport e la sua architettura modulare rendono il sistema altamente flessibile, adattabile a diversi scenari e facilmente estensibile nel tempo. In particolare, l'integrazione con Active Directory e RADIUS consente di sfruttare le credenziali istituzionali esistenti, evitando la necessità di gestire ulteriori account o credenziali esterne.

È importante chiarire che la disabilitazione degli altri metodi di autenticazione nativamente supportati da Teleport (come GitHub, ecc.) è stata una scelta progettuale, finalizzata a garantire che l'accesso avvenga esclusivamente tramite le credenziali ufficiali dell'ateneo. Questo non ha alcun impatto sui sistemi già esistenti all'interno dell'infrastruttura universitaria: servizi come Guacamole o altri strumenti continuano a funzionare come prima, secondo le proprie modalità di autenticazione.

L'adozione di Teleport aggiunge quindi un'opzione in più, più potente e sicura, per accedere alle risorse, senza compromettere la compatibilità con i sistemi

precedenti.

L'implementazione di questa soluzione renderà l'accesso alle risorse IT più comodo, tracciabile e sicuro per tutti gli utenti, migliorando sensibilmente l'esperienza complessiva di accesso remoto.

6.1 Lavori futuri

Nonostante il progetto sia stato completato con successo, ci sono alcuni aspetti che potrebbero essere migliorati o estesi in futuro.

Attualmente, con le modifiche apportate al codice, Teleport supporta l'autenticazione tramite Active Directory e RADIUS, ma non consente più l'uso degli altri metodi di autenticazione nativamente previsti dal progetto originale. Questa limitazione non è tecnica, bensì progettuale: l'autenticazione alternativa è stata volutamente disabilitata per garantire che l'accesso avvenga esclusivamente tramite le credenziali Unibo.

Un possibile sviluppo futuro potrebbe consistere nell'astrarre il sistema di autenticazione rendendolo più modulare, ad esempio separando la logica di autenticazione AD/RADIUS in un modulo dedicato. Ciò permetterebbe maggiore flessibilità e manutenibilità del codice, pur mantenendo le politiche di sicurezza desiderate.

Capitolo 7

Ringraziamenti

I miei più sinceri ringraziamenti vanno al Dott. Ciro Barbone, correlatore di questo progetto, che mi ha supportato, seguito e guidato durante tutto il percorso di tesi, dimostrando sempre grande disponibilità e competenza.

Vorrei ringraziare, inoltre, il Prof. Vittorio Ghini, relatore della tesi, e modello di competenza, simpatia e professionalità. Grazie alle sue doti da insegnante è riuscito a trasmettermi la sua passione, a farmi apprezzare i suoi corsi e a farmi scegliere *System Integration* come percorso di tesi.

Un ringraziamento speciale va alla mia famiglia, che mi ha sempre sostenuto e incoraggiato in ogni mia scelta. Grazie ai miei genitori, che mi hanno sempre spronato a dare il meglio di me stesso, e a mia sorella, che mi ha sempre supportato e aiutato nei momenti di difficoltà. Importantissimi sono stati i miei amici e compagni di corso, che mi hanno aiutato e sostenuto in ogni momento, rendendo il mio percorso di studi più piacevole e divertente, e facendomi sentire parte di una grande famiglia.

Come ultimi, ma non per importanza, ci tengo a ringraziare Carla e Stefano, esempi per me e fonte di ispirazione, che mi hanno sempre spronato a dare il meglio di me e a non mollare mai. Grazie per il vostro costante supporto e per avermi sempre dato la forza di andare avanti.

Bibliografia

[Ale23] Marko Aleksic. What is ssh (secure shell) and how does it work? Link alla risorsa, Nov 2023.

[Ara] Salvatore Aranzulla. Vpn: Come funziona. Link alla risorsa.

[Clo23] Cloudflare. What is tls (transport layer security)? - cloudflare. Link alla risorsa, 2023.

[Elc23] William Elcock. What is radius (remote authentication dial-in user service)? Link alla risorsa, Sep 2023.

[Gra24] Grafana. Grafana. Link alla risorsa, 2024.

[Hat22] Red Hat. What is lightweight directory access protocol (ldap) authentication? Link alla risorsa, Jun 2022.

[Mic22] Microsoft. Active directory domain services overview. Link alla risorsa, Aug 2022.

[Moc21] Septimiu-Vlad Mocan. Vpn history - everything you need to know about vpn development over the past 25 years (and a quick glimpse of the future). Link alla risorsa, Jul 2021.

[Moz23] MozDevNet. Transport layer security - security on the web: Mdn. Link alla risorsa, Jul 2023.

[oF21] Amir Rawdat of F5. Update: Using free let's encrypt ssl/tls certificates with nginx. Link alla risorsa, Jan 2021.

[Pro24] Prometheus. Prometheus. Link alla risorsa, 2024.

[Sel24] Selenium.dev. Selenium remote web driver. Link alla risorsa, 2024.

[Seq17] Anthony Sequeira. A brief history of microsoft's active directory. Link alla risorsa, Sep 2017.

[Tel23a] Teleport. Get started with teleport. Link alla risorsa, 2023.

[Tel23b] Teleport. Teleport architecture. Link alla risorsa, 2023.

[Tel23c] Teleport. Teleport with postgresql. Link alla risorsa, 2023.

[Tes23] TestNG. Testng documentation. Link alla risorsa, 2023.

[Wik23] Wikipedia. Docker. Link alla risorsa, Nov 2023.

[Zer20] ZeroUno. Remote desktop protocol (rdp), ecco cos'è e come funziona. Link alla risorsa, Oct 2020.