

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



Integrazione di Single Sign-On in Unix
Pluggable Authentication Module (Unix
PAM)

Tesi di laurea

Relatore

Prof. Davide Bresolin

Laureando

Ivan Antonino Arena

ANNO ACCADEMICO 2022-2023

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

— Oscar Wilde

Dedicato a ...

Abstract

Lo scopo della tesi è illustrare il lavoro eseguito con Athesys, system integrator specializzato nello sviluppo di soluzioni di Identity and Access Management (IAM), in termini di innovazione e ricerca applicate all'ambito dell'integrazione di servizi web con sistemi unix-based. Nello specifico, si discute la collaborazione con il team infrastructure ed il team dedicato alla ricerca in materia di identità digitale e la conseguente realizzazione di un componente Single Sign-On compatibile con Unix Pluggable Authentication Module (Unix PAM).

“Life is really simple, but we insist on making it complicated”

— Confucius

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Davide Bresolin, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

In secondo luogo, vorrei ringraziare di cuore l'azienda ospitante, Athesys Srl, in particolare, il mio tutor esterno Mattia Zago, Roberto Griggio e Leonardo Speranzon, per la disponibilità e l'impegno con cui mi hanno affiancato durante il periodo di stage.

Desidero, inoltre, ringraziare con affetto i miei genitori per avermi appoggiato in ogni mia scelta durante il mio periodo universitario e per avermi fornito il supporto ed i mezzi per portarlo a termine con serenità.

Infine, ci terrei a ringraziare tutte le amicizie più significative che ho stretto a Padova, che mi hanno regalato gioie e sorrisi durante questi tre emozionanti anni.

Padova, Maggio 2023

Ivan Antonino Arena

Indice

1	Introduzione	1
1.1	L'azienda	1
1.2	Il progetto	2
2	Processi e metodologie	3
2.1	Organizzazione del lavoro	3
2.2	Tecnologie utilizzate	3
2.2.1	FreeIPA	3
2.2.2	LXC	4
2.2.3	SSH	4
3	Studio tecnologico	5
3.1	Single Sign-On	5
3.2	Self-Sovereign Identity	5
3.3	OAuth2	6
3.4	OpenID Connect	6
3.5	Linux PAM	7
3.6	IAM	7
3.7	IDaaS	7
4	Configurazione dei sistemi	9
4.1	Creazione ambienti virtuali	9
4.2	Configurazione FreeIPA	9
4.3	Configurazione Monokee	9
5	Ricerca e sperimentazione	11
5.1	Unix PAM	11
5.2	FreeIPA IdP	11
6	Implementazione e documentazione	13
6.1	Configurazione Monokee	13
6.2	Configurazione FreeIPA	13
6.3	Testing	13
6.4	Troubleshooting	14
6.5	Sviluppi futuri	14
6.6	Documentazione	14
7	Conclusioni	15
7.1	Consuntivo finale	15

7.2	Raggiungimento degli obiettivi	15
7.3	Conoscenze acquisite	15
7.4	Valutazione personale	15
A	Appendice A	17
A.1	Integrazione di Monokee Single Sign-On (OIDC) tramite FreeIPA . . .	17
A.1.1	Indice	17
A.1.2	Introduzione	17
A.1.3	Installazione di FreeIPA	17
A.1.4	Setup Monokee	19
A.1.5	Setup IdP FreeIPA	21
A.1.6	Problemi	24
B	Appendice B	25
B.1	Codice C dell'applicazione PAM-aware	25
B.2	Codice C del modulo PAM	26
B.3	Script bash per compilare ed eseguire l'applicazione PAM-aware	31
	Acronimi e abbreviazioni	33
	Bibliografia	35

Elenco delle figure

1.1	Logo di Athesys Srl	1
1.2	Logo di Monokee	1
4.1	Vista parziale del template download di LXC	10
A.1	Vista schermata di login FreeIPA	20
A.2	Vista nuova app OAuth2 da Monokee	20
A.3	Vista OAuth users	21
A.4	Vista OAuth client configuration da Monokee	21
A.5	Vista client scopes	21
A.6	Vista application assets	22
A.7	Vista nuovo OpenID provider da Monokee	22
A.8	Valori OpenID provider	23
A.9	Valori OpenID provider Advanceds	23
A.10	Valori Identity Provider Server da FreeIPA	24
A.11	Setup utente FreeIPA	24
A.12	Monokee SSO da CLI	24

Elenco delle tabelle

Capitolo 1

Introduzione

In questo capitolo vengono descritti l'azienda ospitante ed il progetto dell'attività di stage curriculare.

1.1 L'azienda

Athesys Srl è un'azienda di consulenza informatica nata a Padova nel 2010 "dalla sinergia di affermati professionisti del settore IT", specializzata in ambito System Integration, Database Management, Sicurezza applicativa, Governance Cloud Platform, Hyperconvergenza e Sviluppo Software in modalità Agile.

Athesys comprende la start-up Monokee, fondata nel 2017 come soluzione cloud-based per la gestione dell'identità e dell'accesso (IDaaS), la quale offre, come funzionalità principale, un sistema di Single Sign-On basato su diversi tipi di autenticazione, sia passwordless che tramite soluzioni di Self-Sovereign Identity (SSI).



Figura 1.1: Logo di Athesys Srl



Figura 1.2: Logo di Monokee

1.2 Il progetto

L'idea per l'attività di stage nasce proprio dall'esigenza dell'azienda di aumentare la portata di Monokee estendendo il relativo Single Sign-On anche a livello macchina, per poter, successivamente, configurare dei terminali che possano gestire accesso e sessioni degli utenti Monokee già da sistema.

L'obiettivo del progetto del tirocinio era, dunque, era la ricerca e l'eventuale sviluppo di una soluzione che consentisse di accedere a macchine UNIX Debian-like e RHEL-like tramite il proprio account Monokee in modo nativo, sfruttando l'infrastruttura di Single Sign-On fornita dalla start-up.

Il framework da utilizzare era FreeIPA, un gestore delle identità e degli accessi (Identity and Access Management, IAM) gratuito ed open-source che combina tecnologie quali Linux, LDAP, MIT Kerberos, NTP, DNS ed SSSD e consta di un'interfaccia web e di strumenti di amministrazione tramite command-line.

L'attività, dalla durata totale di circa trecento ore, si è sviluppata inizialmente in un fase di ricerca e sperimentazione con l'installazione del software FreeIPA su più macchine virtuali CentOS, RHEL e Ubuntu, messe a disposizione dall'azienda.

La seconda fase è stata dedicata alla ricerca di un metodo che consentisse di effettuare l'autenticazione con il proprio account Monokee su tali macchine; in tal senso, è stata approfondita la parte relativa a Unix PAM (Pluggable Authentication Modules) per studiare la possibilità dello sviluppo di un modulo aggiuntivo.

In seguito a tale ricerca, ho deciso di optare per il sistema di autenticazione tramite Identity Provider esterno messo a disposizione dall'applicativo di FreeIPA e di procedere, dunque, con la configurazione di un'applicazione Monokee OAuth2 e di un provider OpenID Connect (OIDC) che fornissero gli end-point e l'infrastruttura necessaria alla comunicazione con il server di FreeIPA e la successiva implementazione degli stessi su di esso.

Verificato il corretto funzionamento di questo sistema di autenticazione da [Command-Line Interface \(CLI\)](#), ho proseguito cercando di implementare questo sistema anche tramite SSH fino al raggiungimento delle ore previste.

Capitolo 2

Processi e metodologie

In questo capitolo vengono descritte le modalità con cui si è svolto lo stage e le tecnologie utilizzate.

2.1 Organizzazione del lavoro

Il lavoro è stato suddiviso in n periodi.

Nel primo ho fatto xyz

2.2 Tecnologie utilizzate

2.2.1 FreeIPA

FreeIPA è una soluzione open-source gratuita (GNU General Public License) di gestione dell'identità e dell'accesso per ambienti di rete basati su Linux/UNIX, originariamente sviluppato dalla comunità Fedora ed ora supportato da diverse organizzazioni, tra cui Red Hat e la FreeIPA Foundation. Consiste in un insieme di servizi integrati, che consentono di centralizzare l'autenticazione, l'autorizzazione e la gestione degli utenti e delle risorse in un'organizzazione.

FreeIPA è progettato per semplificare la gestione dell'identità e dell'accesso in ambienti di rete complessi, con molti utenti e computer. Consente agli amministratori di gestire facilmente l'accesso degli utenti a risorse e applicazioni, di delegare i privilegi di amministrazione e di definire ed applicare politiche di sicurezza coerenti in tutta la rete, come, ad esempio, limitare l'accesso alle risorse in base al ruolo dell'utente. Per fare ciò, mette a disposizione, oltre che agli strumenti della CLI, un'interfaccia utente web intuitiva per la gestione degli utenti, dei gruppi e delle risorse della rete. Inoltre, FreeIPA è altamente scalabile e può essere distribuito su più server per gestire grandi reti.

Per l'autenticazione degli utenti, FreeIPA utilizza il protocollo Kerberos: gli utenti possono accedere alle risorse della rete utilizzando le loro credenziali Kerberos, senza dover inserire le password ogni volta. Per archiviare e gestire le informazioni sugli utenti, i gruppi e le risorse della rete, invece, utilizza il server di directory open-source 389 Directory Server, il quale offre funzionalità avanzate di ricerca, replica e sincronizzazione.

FreeIPA supporta l'autenticazione SSO tramite il protocollo SAML (Security Assertion Markup Language), ciò significa che gli utenti possono accedere a più applicazioni utilizzando le stesse credenziali di accesso.

2.2.2 LXC

LXC è l'acronimo di Linux Containers, un sistema di virtualizzazione basato sul kernel Linux che consente di eseguire più sistemi operativi isolati su una singola macchina host. A differenza della virtualizzazione completa, in cui ogni sistema operativo guest ha accesso all'intero hardware dell'host,

Utilizza la virtualizzazione basata sui contenitori, in cui ogni sistema operativo guest condivide le risorse hardware dell'host. La condivisione del kernel fa sì che i container siano molto leggeri e veloci e che abbiano un overhead di risorse molto basso rispetto ad altre tecnologie di virtualizzazione.

LXC fornisce un'interfaccia di riga di comando per la gestione dei container, che consente di creare, avviare, fermare, eliminare e gestire i container in modo semplice ed efficiente. Inoltre, supporta la creazione di immagini di container, che possono essere utilizzate per creare nuovi container in modo rapido e semplice.

Durante l'attività di stage ho utilizzato principalmente container basati su immagini CentOS (Community Enterprise Operating System), una distribuzione Linux basata su Red Hat Enterprise Linux (RHEL) particolarmente adatta all'uso in ambiente server, che offre una vasta gamma di funzionalità e strumenti per gestire un'infrastruttura IT.

In particolare, ho utilizzato l'ultima versione (9) di CentOS Stream, che, a differenza della versione standard, riceve gli aggiornamenti in tempo reale durante lo sviluppo di RHEL, consentendo agli utenti di testare e fornire feedback sulle nuove funzionalità e correzioni di bug in anteprima.

Un'altra differenza tra CentOS e CentOS Stream è la durata del supporto: CentOS ha, storicamente, fornito un supporto a lungo termine per le versioni rilasciate; la versione Stream, al contrario, è progettata per essere una piattaforma di sviluppo in continuo aggiornamento e non offre, dunque, il supporto a lungo termine.

2.2.3 SSH

Secure Shell (SSH) è un protocollo di rete crittografato utilizzato per la gestione sicura di dispositivi di rete e per l'accesso remoto a sistemi informatici. Il protocollo SSH fornisce un canale di comunicazione sicuro tra due dispositivi, garantendo l'integrità, la riservatezza e l'autenticità delle informazioni trasmesse.

L'autenticazione avviene attraverso l'uso di chiavi pubbliche e private: in questo metodo, un'entità che desidera accedere a un sistema remoto genera una coppia di chiavi, una pubblica e una privata; la chiave pubblica viene fornita al sistema remoto, mentre la chiave privata viene conservata dall'entità; quando l'entità si connette al sistema remoto, la chiave privata viene utilizzata per autenticare l'entità.

Inoltre, SSH utilizza la crittografia per proteggere i dati trasferiti tra i dispositivi. In particolare, il protocollo utilizza la crittografia a chiave simmetrica per proteggere i dati durante la trasmissione, e la crittografia a chiave pubblica per autenticare le parti coinvolte.

Capitolo 3

Studio tecnologico

In questo capitolo vengono illustrati nel dettaglio i concetti e le tecnologie utilizzate.

3.1 Single Sign-On

Il Single Sign On (SSO) è una tecnologia che consente agli utenti di accedere a più applicazioni e servizi utilizzando un'unica identità di accesso. In pratica, l'utente inserisce le proprie credenziali di accesso una sola volta e successivamente può accedere a tutte le applicazioni e servizi che supportano l'SSO senza dover inserire nuovamente le credenziali, alternativamente a come accade con l'autenticazione tradizionale. Ciò può risultare particolarmente vantaggioso se l'utente necessita di accedere a molte applicazioni o servizi diversi.

L'SSO funziona attraverso l'utilizzo di un'autorità di autenticazione centralizzata, chiamata Identity Provider (IdP). L'IdP autentica l'utente e fornisce un token di sicurezza che contiene le informazioni sull'utente e sui servizi a cui ha accesso. Questo token può essere utilizzato per accedere a tutti i servizi che supportano tale sistema.

Per utilizzare l'SSO, le applicazioni e i servizi devono supportare uno dei protocolli SSO standard, come SAML (Security Assertion Markup Language) o OpenID Connect. Questi protocolli definiscono il modo in cui le informazioni di autenticazione dell'utente vengono trasmesse tra le diverse applicazioni e servizi.

L'SSO offre, dunque, numerosi vantaggi, tra cui una maggiore comodità per gli utenti, una maggiore sicurezza attraverso l'utilizzo di token di sicurezza a breve termine e una maggiore efficienza nella gestione delle identità e delle autorizzazioni degli utenti. Tuttavia, l'SSO richiede una pianificazione e una configurazione adeguata per garantire la sicurezza e la protezione dei dati degli utenti.

3.2 Self-Sovereign Identity

L'SSI (Self-Sovereign Identity) è un nuovo approccio alla gestione delle identità digitali che consente agli utenti di possedere, controllare e condividere le proprie informazioni di identità in modo sicuro e privato. A differenza dei sistemi di identità tradizionali, in cui le informazioni di identità sono conservate in modo centralizzato da terze parti, l'SSI consente agli utenti di essere i proprietari esclusivi dei propri dati di identità digitali.

L'SSI si basa sulla tecnologia blockchain, che consente di creare registri distribuiti di informazioni sicure e immutabili. In tal modo, le informazioni di identità degli utenti vengono conservate in modo decentralizzato e sicuro, senza la necessità di un'autorità centralizzata di controllo.

Per utilizzare l'SSI, gli utenti creano un'identità digitale che include le informazioni di identità necessarie, come nome, indirizzo e informazioni di contatto. Questa identità digitale viene conservata sulla blockchain e protetta da una chiave privata unica, che solo l'utente possiede.

Gli utenti possono utilizzare la propria identità digitale SSI per accedere a servizi online e condividere le proprie informazioni di identità solo con le parti che desiderano. Questo viene fatto attraverso l'utilizzo di un protocollo di scambio di informazioni sicuro e decentralizzato, chiamato DID (Decentralized Identifier).

L'SSI offre numerosi vantaggi, tra cui un maggiore controllo e privacy per gli utenti rispetto ai sistemi di identità tradizionali, una maggiore sicurezza attraverso l'utilizzo della tecnologia blockchain e una maggiore efficienza nella gestione delle identità digitali. Tuttavia, è ancora una tecnologia emergente e richiede una maggiore adozione e sviluppo per diventare un approccio mainstream alla gestione delle identità digitali.

3.3 OAuth2

OAuth2 è un protocollo di autorizzazione che consente a un'applicazione di accedere alle risorse di un utente senza richiedere le credenziali dell'utente - e, di conseguenza, senza memorizzarle - nato per assicurare l'accesso sicuro e controllato ai dati di un utente da parte di applicazioni di terze parti.

Funziona attraverso una serie di flussi di autorizzazione, in cui l'utente concede l'autorizzazione all'applicazione per accedere alle sue risorse. L'applicazione, a sua volta, ottiene un token di accesso che può essere utilizzato per accedere alle risorse dell'utente.

Il protocollo OAuth2 è utilizzato da molte grandi piattaforme online come Google, Facebook e Twitter ed è diventato, di fatto, uno standard nei servizi cloud, nei social network, nei servizi di pagamento online e in molti altri contesti.

3.4 OpenID Connect

OpenID Connect (OIDC) è un protocollo di autenticazione basato su OAuth2, utilizzato per l'autenticazione degli utenti in applicazioni web e mobile. Progettato per risolvere il problema dell'autenticazione sicura e decentralizzata in applicazioni di terze parti, consente agli utenti di utilizzare l'SSO per accedere a diverse applicazioni, senza, quindi, dover creare un nuovo account per ogni applicazione, bensì delegando l'autenticazione ad un provider esterno (OpenID Provider).

OIDC fornisce un framework standard per l'autenticazione basata su JSON Web Tokens (JWT), in cui l'utente viene autenticato una sola volta e poi viene rilasciato un token di accesso contenente le informazioni di base dell'utente, come l'identificatore univoco, il nome e l'e-mail, che può essere utilizzato per accedere alle risorse protette.

Questo protocollo è stato adottato da molte grandi piattaforme online, tra cui Google, Microsoft e Amazon ed è anche supportato da molte librerie di sviluppo, caratteristica che lo rende semplice da implementare per gli sviluppatori.

3.5 Linux PAM

Linux PAM (Pluggable Authentication Modules) è un framework di autenticazione per i sistemi operativi Linux e UNIX che consente di configurare diversi metodi di autenticazione, come quella tramite password, a due fattori, basata su token, biometrica, ecc.

Utilizzato in una vasta gamma di applicazioni e servizi, tra cui il sistema di login del sistema operativo ed il server SSH, il framework di PAM è composto da una serie di moduli, ognuno dei quali implementa una particolare funzionalità di autenticazione. I moduli PAM sono progettati per essere "pluggable", ovvero possono essere facilmente sostituiti o aggiunti senza dover modificare il codice sorgente del sistema operativo.

L'architettura modulare di PAM consente di creare una catena di moduli, in cui ciascuno dei quali può verificare una parte dell'identità dell'utente. Ad esempio, un modulo può verificare la password dell'utente, mentre un altro può verificare il certificato del client. Se uno qualsiasi dei moduli nella catena fallisce, l'intero processo di autenticazione viene interrotto. Inoltre, è possibile sviluppare dei moduli personalizzati ed integrarli nelle diverse funzioni che richiedono PAM.

3.6 IAM

IAM, acronimo di Identity and Access Management, è un insieme di processi, politiche, tecnologie e strumenti utilizzati per gestire l'identità digitale e il controllo degli accessi agli utenti all'interno di un'organizzazione. L'obiettivo principale di IAM è garantire che le persone giuste abbiano accesso alle risorse appropriate al momento opportuno, in modo sicuro e conforme alle politiche aziendali.

Alcune delle funzionalità chiave di IAM includono la gestione centralizzata delle identità, la gestione dei ruoli e delle politiche di accesso, la gestione delle password, la gestione delle sessioni, la federazione dell'identità per consentire l'accesso a risorse esterne e l'integrazione con sistemi e applicazioni esistenti.

IAM consente anche di implementare il principio del "least privilege", che significa assegnare agli utenti solo i privilegi necessari per svolgere il proprio lavoro, minimizzando così il rischio di abusi o accessi non autorizzati.

L'implementazione di un sistema IAM efficace può portare a numerosi benefici per un'organizzazione, come una maggiore sicurezza dei dati, la conformità normativa, una migliore gestione delle risorse IT, una riduzione dei rischi e un'efficienza operativa migliorata.

3.7 IDaaS

IDaaS, acronimo di Identity as a Service, è un modello di distribuzione dei servizi di gestione delle identità basato su cloud. Con IDaaS, un'organizzazione può affidare la gestione delle identità dei propri utenti a un provider di servizi esterno, eliminando la necessità di mantenere un'infrastruttura locale per la registrazione degli utenti, l'autenticazione e l'autorizzazione degli utenti, la gestione delle password e la gestione delle sessioni.

I servizi IDaaS sono solitamente forniti come un'offerta di software as a service (SaaS), accessibili tramite internet e scalabili in base alle esigenze dell'organizzazione. I provider di servizi IDaaS gestiscono l'infrastruttura necessaria per garantire la sicurezza, la disponibilità e le prestazioni dei servizi di gestione delle identità.

Tra le funzionalità comuni offerte dai servizi IDaaS ci sono l'integrazione con directory aziendali esistenti, la possibilità di supportare l'autenticazione a fattori multipli, l'autenticazione federata per consentire l'accesso a risorse esterne all'organizzazione e la gestione centralizzata delle autorizzazioni degli utenti.

Utilizzando IDaaS, le organizzazioni possono beneficiare di diversi vantaggi, tra cui una maggiore agilità nell'onboarding e nella gestione degli utenti, un'esperienza utente migliorata grazie a un'unica identità digitale per l'accesso a più servizi, una maggiore sicurezza grazie all'implementazione di misure di autenticazione avanzate e un risparmio sui costi e sulla complessità operativa associata alla gestione delle identità.

Capitolo 4

Configurazione dei sistemi

In questo capitolo vengono descritti i procedimenti attuati per configurare i sistemi utilizzati.

4.1 Creazione ambienti virtuali

Avendo familiarità con Ubuntu ho, inizialmente, verificato che il pacchetto del server di FreeIPA fosse presente: ho presto appreso che esso era discontinuo sulle ultime versioni del sistema operativo.

Così, ho deciso di optare per CentOS Stream 9, ultima versione disponibile, per l'ottima compatibilità con FreeIPA e la migliore usabilità in ambienti server.

A questo punto, con l'aiuto del team di Athesys Srl, ho configurato LXC sulla mia macchina Ubuntu 22.04 LTS.

Tramite il comando `lxc-create -t download -n ipa-server`, ho creato un nuovo container con il nome di *ipa-server* indicando di voler scaricare il template dalla lista di quelli disponibili, dalla quale ho scelto l'immagine di CentOS Stream 9 ([Figura 4.1](#)).

Dopo la creazione della macchina ho lanciato i comandi `lxc-start ipa-server` e `lxc-console ipa-server`, rispettivamente per avviare il container e per accedere al relativo terminale.

4.2 Configurazione FreeIPA

Dopo aver configurato il container per il server ed aver eseguito l'aggiornamento dei pacchetti con il comando `yum update`, sono passato all'installazione del server di FreeIPA, disponibile su quella release con il pacchetto *freeipa-server*.

4.3 Configurazione Monokee

Al mio arrivo negli uffici di Athesys Srl, l'azienda aveva già configurato per me un account sulla loro piattaforma di testing, *test.monokee.com*, utilizzando come e-mail il mio indirizzo istituzionale e garantendomi l'accesso a tutte le risorse della piattaforma, oltre che alla documentazione aziendale.

```

root@ubuntivan:~
root@ubuntivan:~# lxc-create -t download -n ipa_server
Downloading the image index
...
DIST  RELEASE ARCH  VARIANT BUILD
...
alinux  8  amd64  default  20230512_00:05
alinux  8  arm64  default  20230512_02:07
alinux  8  ppc64el default  20230512_00:04
alinux  9  amd64  default  20230512_06:28
alinux  9  arm64  default  20230512_02:19
alinux  9  ppc64el default  20230512_00:04
alpine  3.14 amd64  default  20230510_13:01
alpine  3.14 arm64  default  20230510_13:00
alpine  3.14 armhf  default  20230510_13:05
alpine  3.14 i386  default  20230510_13:02
alpine  3.14 ppc64el default  20230510_13:00
alpine  3.14 s390x  default  20230510_13:00
alpine  3.15 amd64  default  20230511_15:54
alpine  3.15 arm64  default  20230511_16:01
alpine  3.15 armhf  default  20230511_17:13
alpine  3.15 i386  default  20230511_15:48
alpine  3.15 ppc64el default  20230511_16:35
alpine  3.15 s390x  default  20230511_15:44
alpine  3.16 amd64  default  20230511_15:57
alpine  3.16 arm64  default  20230511_18:15
alpine  3.16 armhf  default  20230511_20:58
alpine  3.16 i386  default  20230511_15:19
alpine  3.16 ppc64el default  20230511_15:57
alpine  3.16 s390x  default  20230511_15:53
alpine  3.17 amd64  default  20230511_22:07
alpine  3.17 arm64  default  20230512_06:37
alpine  3.17 armhf  default  20230511_21:01
alpine  3.17 i386  default  20230511_15:56
alpine  3.17 ppc64el default  20230511_16:24
alpine  3.17 s390x  default  20230511_15:47
alpine  3.18 amd64  default  20230511_15:45
alpine  3.18 arm64  default  20230511_18:03
alpine  3.18 armhf  default  20230511_21:55
alpine  3.18 i386  default  20230511_15:54
alpine  3.18 ppc64el default  20230511_16:39

```

Figura 4.1: Vista parziale del template download di LXC

Capitolo 5

Ricerca e sperimentazione

In questo capitolo viene descritto il processo di ricerca e sperimentazione di una soluzione efficace per l'implementazione del SSO nativo

5.1 Unix PAM

La prima idea che ho avuto per integrare il Single Sign-On di Monokee via SSH sul container CentOS che ho predisposto è stata quella di creare un nuovo modulo PAM. Ciò perché, studiando i file presenti al percorso `/etc/pam.d/` ho trovato il file `sshd`, che stabilisce i moduli da utilizzare per autenticazione, autorizzazione, sessione e gestione della password. In un primo momento mi sono concentrato sulla parte di autenticazione, controllando il file di configurazione `common-auth`, incluso in `/etc/pam.d/sshd`, che rappresenta l'autenticazione predefinita di UNIX (con password memorizzata localmente).

Tuttavia, l'installazione di FreeIPA sovrascrive il parametro `UsePam yes` del file `/etc/ssh/sshd_config` antepoendo dei parametri relativi a Kerberos, in modo da potersi autenticare con la password dell'utente FreeIPA specificato nel prefisso della macchina nel comando di SSH.

La mia idea era, dunque, quella di rimuovere questi parametri e tornare all'autenticazione via PAM, sostituendo però il modulo predefinito con uno creato appositamente per l'SSO di Monokee.

Il problema restava quello del riconoscimento dell'utente ma avevo già pensato a diversi modi in cui poterlo risolvere, così ho deciso di proseguire e sperimentare con lo sviluppo di un modulo PAM di test, per verificare la fattibilità della mia intuizione.

Dapprima, ho deciso di sviluppare una semplice applicazione PAM-aware, ovvero compatibile con Linux PAM, utilizzando il linguaggio C e facendo riferimento alla documentazione trovata.

5.2 FreeIPA IdP

Dato che la soluzione con il modulo PAM si è rivelata essere più impegnativa del previsto, ho deciso di provare a configurare l'SSO con Monokee da FreeIPA.

Navigando nell'interfaccia web del software, infatti, ho notato che nella sezione *Authentication > Identity Provider Servers* era possibile definire un Identity Provider che utilizzasse OAuth 2.0 come protocollo di autenticazione.

A questo punto, con l'aiuto del team, ed in particolare del CTO di Athesys Srl, mi sono spostato sull'infrastruttura di testing di Monokee per configurare un'applicazione OAuth2 da poter utilizzare come Identity Provider per FreeIPA.

Capitolo 6

Implementazione e documentazione

In questo capitolo viene illustrato il processo di implementazione della soluzione trovata e la stesura della documentazione relativa.

6.1 Configurazione Monokee

All'interno dell'ambiente di test di Monokee, tramite l'interfaccia web, ho creato una nuova applicazione OAuth2 ed un nuovo OpenID Connect provider, che fornisse gli end-point per l'autenticazione via OIDC.

6.2 Configurazione FreeIPA

Da FreeIPA, ho proceduto a configurare un nuovo Identity Provider server, tramite la sezione della GUI di cui sopra, inserendo tutte i metadati richiesti, facendo riferimento all'applicazione OAuth2 creata su Monokee e agli end-point forniti dall'OpenID provider configurato precedentemente.

Successivamente, ho creato un utente FreeIPA che utilizzasse come unico metodo di autenticazione quella tramite Identity Provider esterno (*External IdP*), scegliendo Monokee come IdP e come identificatore il mio indirizzo e-mail istituzionale, già associato al mio account Monokee, per poter eseguire l'accesso via SSO con le mie credenziali.

6.3 Testing

Configurata correttamente l'infrastruttura di autenticazione sia su Monokee che su FreeIPA, ho proceduto a testarla seguendo le indicazioni della documentazione relativa.

6.4 Troubleshooting

6.5 Sviluppi futuri

6.6 Documentazione

L'azienda ha richiesto la redazione di una guida che illustrare il processo di configurazione del server FreeIPA e dei sistemi di Monokey per l'integrazione del Single Sign-On sulle macchine UNIX, per fornire una base documentativa per facilitare le future progressioni e sperimentazioni relative. Ho steso tale documentazione in formato Markdown, versionando il codice sul repository GitHub aziendale fornito da Athesys Srl.

Capitolo 7

Conclusioni

7.1 Consuntivo finale

7.2 Raggiungimento degli obiettivi

L'attività è stata svolta quasi totalmente in linea con la pianificazione prevista: non ci sono stati ritardi di alcun tipo ed il primo periodo, quello di studio delle tecnologie, ha richiesto meno tempo di quanto preventivato, consentendomi, così, di approfondire ulteriormente lo sviluppo delle soluzioni trovate nelle fasi successive. Ho completato tutti gli obiettivi richiesti con successo, ad inclusione di quelli desiderabili e opzionali.

7.3 Conoscenze acquisite

Grazie allo stage con Athesys Srlmi sono addentrato in un campo dell'informatica che poco conoscevo, quello della sicurezza. Ho avuto modo di conoscere i concetti e le tecnologie più significative del momento presente in ambito di identità digitale, come la Self-Sovereign Identity, il Single-Sign On ed alcuni dei protocolli di autenticazione ed autorizzazione più diffusi, apprendendo, anzitutto, cosa significa creare e gestire un'identità digitale e quali sono i rischi di sicurezza legati ad essa. Oltre ad una già ampia formazione teorica, ho avuto anche la possibilità di migliorare le mie competenze in ambito di sistemi UNIX, entrando a contatto con parti di codice che cambiano direttamente il comportamento del sistema operativo, come i moduli PAM e l'SSH. Inoltre, ho imparato a creare dei container LXC dalle immagini messe a disposizione e, successivamente, a configurare un server di Identity and Access Management, quale FreeIPA, modificando anche, in alcuni casi, manualmente dei file di sistema. Infine, ho messo in atto le conoscenze acquisite nella prima fase dell'attività, in particolare quelle riguardanti il funzionamento di OAuth2 ed OIDC, per implementare il PoC richiesto.

7.4 Valutazione personale

Dopo circa trecento ore passate al fianco del team di Athesys Srl sono convinto di aver acquisito delle conoscenze e delle competenze, non strettamente tecniche, fondamentali per il mio ingresso prossimo nell'industria: questa esperienza, che costituisce la mia prima nell'ambito del percorso che mi appartiene, quello dell'informatica, mi ha

permesso di affrontare personalmente e toccare con mano le sfide, i problemi, le metodologie ed i traguardi propri della realtà delle aziende informatiche. A partire dalla comunicazione, dall'organizzazione e dalla gestione del tempo e delle risorse, arrivando poi agli effettivi processi risolutivi e di sviluppo, sento di aver ricevuto un contributo significativo e di essermi messo alla prova, applicandomi al meglio in un ambiente a me quasi del tutto sconosciuto, al di fuori della mia zona di comfort. Ora, al momento della stesura di questo documento, ripercorrendo ciò che ho fatto durante questa attività di stage, mi rendo conto ancora meglio del valore che essa ha avuto ed ha per me e per la mia carriera.

Appendice A

Appendice A

A.1 Integrazione di Monokee Single Sign-On (OIDC) tramite FreeIPA

A.1.1 Indice

- * [Integrazione di Monokee Single Sign-On \(OIDC\) tramite FreeIPA](#)

- [Indice](#)
- [Introduzione](#)
- [Installazione di FreeIPA](#)
 - * [Requisiti di sistema](#)
 - * [Installazione](#)
 - * [Accesso a FreeIPA](#)
- [Setup Monokee](#)
 - * [App OAuth](#)
 - * [OpenID Connect Provider](#)
- [Setup IdP FreeIPA](#)
 - * [Setup Identity Provider Server](#)
 - * [Setup Utenti](#)
- [Autenticazione via CLI](#)

A.1.2 Introduzione

Il seguente documento illustra il procedimento per disporre un ambiente di Single Sign-On che utilizzi Monokee con OpenID Connect come external Identity Provider di FreeIPA.

A.1.3 Installazione di FreeIPA

Requisiti di sistema

Per il server su cui installare FreeIPA è consigliata una macchina con CentOS Stream 9.

È consigliabile anche lanciare i seguenti comandi per installare pacchetti utili: EPEL (Extra Packages for Enterprise Linux) è un insieme di pacchetti di Fedora non presenti nativamente su sistemi RHEL, *bind-utils* contiene dei comandi per ottenere facilmente informazioni riguardanti i DNS, Vim è un editor di testo.

```
sudo yum -y install epel-release
sudo yum -y update
sudo yum install bind-utils vim
```

Installazione

Lanciare il seguente comando per scaricare il pacchetto di FreeIPA.

```
sudo yum -y install ipa-server
```

In questa guida utilizzeremo FQDN, invece che DNS, per comodità. È sufficiente modificare il file */etc/hosts* aggiungendo l'indirizzo IP della macchina su cui si sta effettuando l'installazione.

```
sudo vim /etc/hosts
10.0.3.190 ipa.server.com
```

Configurare l'hostname con lo stesso nome utilizzato nel file */etc/hosts*.

```
sudo hostnamectl set-hostname ipa.server.com
```

Lanciare il comando di installazione.

```
sudo ipa-server-install
```

Appariranno una serie di domande a schermo: è sufficiente premere *Invio* a tutte tranne che all'ultima, di conferma, a cui bisogna rispondere in modo affermativo. Inoltre, verrà chiesto di inserire e confermare delle password. Dopo qualche minuto dovrebbe apparire un messaggio di riuscita installazione.

```
Continue to configure the system with these values? [no]: yes
```

```
The following operations may take some minutes to complete.
Please wait until the prompt is returned.
```

```
Configuring NTP daemon (ntpd)
[1/4]: stopping ntpd
[2/4]: writing configuration
[3/4]: configuring ntpd to start on boot
[4/4]: starting ntpd
Done configuring NTP daemon (ntpd).
Configuring directory server (dirsrv). Estimated time: 30 seconds
.....
Client configuration complete.
The ipa-client-install command was successful
```

```
=====
Setup complete
```

A.1. INTEGRAZIONE DI MONOKEE SINGLE SIGN-ON (OIDC) TRAMITE FREEIPA19

Next steps:

1. You must make sure these network ports are open:

TCP Ports:

- * 80, 443: HTTP/HTTPS
- * 389, 636: LDAP/LDAPS
- * 88, 464: kerberos

UDP Ports:

- * 88, 464: kerberos
- * 123: ntp

2. You can now obtain a kerberos ticket using the command: `'kinit admin'`
This ticket will allow you to use the IPA tools (e.g., `ipa user-add`) and the web user interface.

Be sure to back up the CA certificates stored in `/root/cacert.p12`
These files are required to create replicas. The password for these files is the Directory Manager password
...

È possibile che debbano essere aperte alcune porte in presenza di un firewall attivo, si consiglia, dunque, di lanciare i seguenti comandi. Se il firewall non è attivo, si consiglia comunque di attivarlo e lanciare i comandi seguenti.

```
sudo firewall-cmd --add-service={dns,freeipa-ldap,freeipa-ldaps} --permanent  
sudo firewall-cmd --reload
```

Accesso a FreeIPA

È possibile accedere a FreeIPA da CLI, richiedendo un ticket Kerberos con il comando `kinit admin` oppure dall'interfaccia web, all'indirizzo uguale all'hostname che si è utilizzato (`https://ipa.server.com/`), inserendo come username *admin*; in entrambi i casi la password corrisponde a quella scelta durante il processo di installazione.

A.1.4 Setup Monokee

App OAuth

Su test.monokee.com andare nella sezione *Applications*, creare una nuova applicazione OAuth.

In *Application assets*, aggiungere gli utenti (e/o i gruppi di utenti) ai quali si vuole consentire l'accesso sia nella sezione *Users* (e/o *Groups*) che nella relativa sezione in *Scopes*.

Andare sull'icona della matita per modificare l'applicazione e cliccare su *Next* per passare alle impostazioni del client. Qui scegliere un *Client ID* ed un *Client Secret* (quest'ultimo è possibile anche lasciarlo vuoto) ed impostare gli altri valori come da figura.

Cliccare nuovamente su *Next* ed impostare i valori della pagina come da figura.

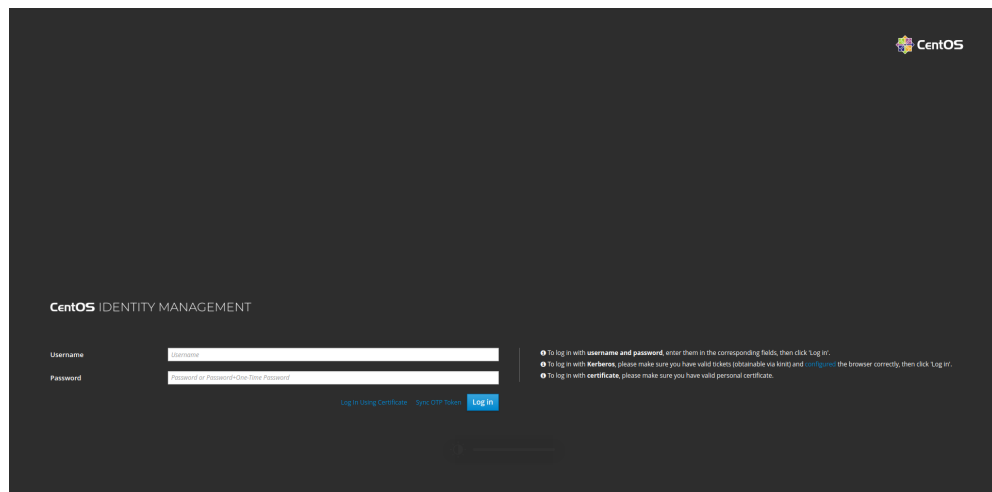


Figura A.1: Vista schermata di login FreeIPA

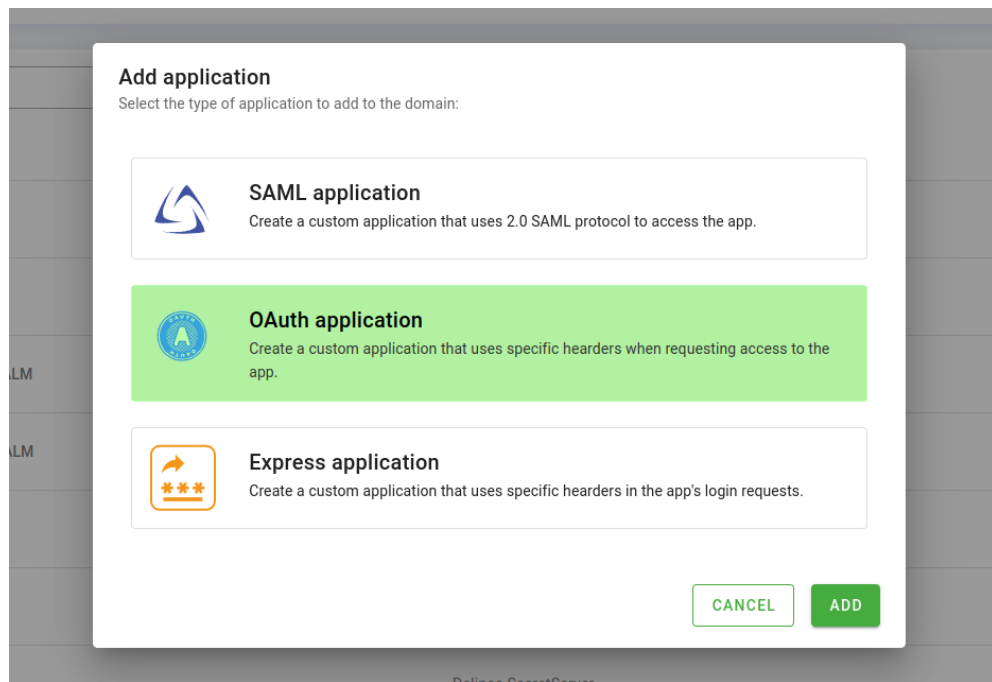


Figura A.2: Vista nuova app OAuth2 da Monokee

OpenID Connect Provider

Spostarsi nella sezione *OAuth Providers* ed aggiungere un nuovo OpenID Provider nell'apposita vista.

Spuntare l'opzione *Display metadata*, scegliere un nome per il provider ed inserire i valori come da figura.

Spostarsi nella sezione *Advanced* ed impostare i valori come da figura.

A.1. INTEGRAZIONE DI MONOKEE SINGLE SIGN-ON (OIDC) TRAMITE FREEIPA21

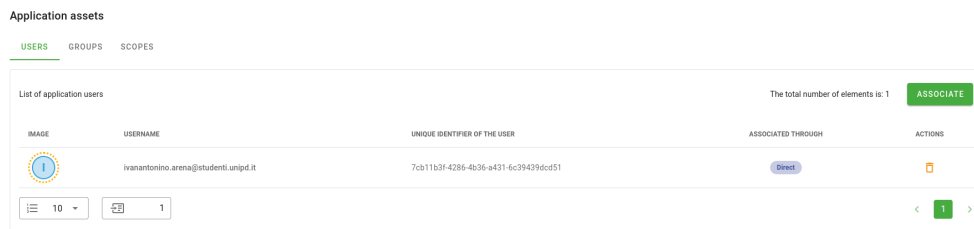


Figura A.3: Vista OAuth users

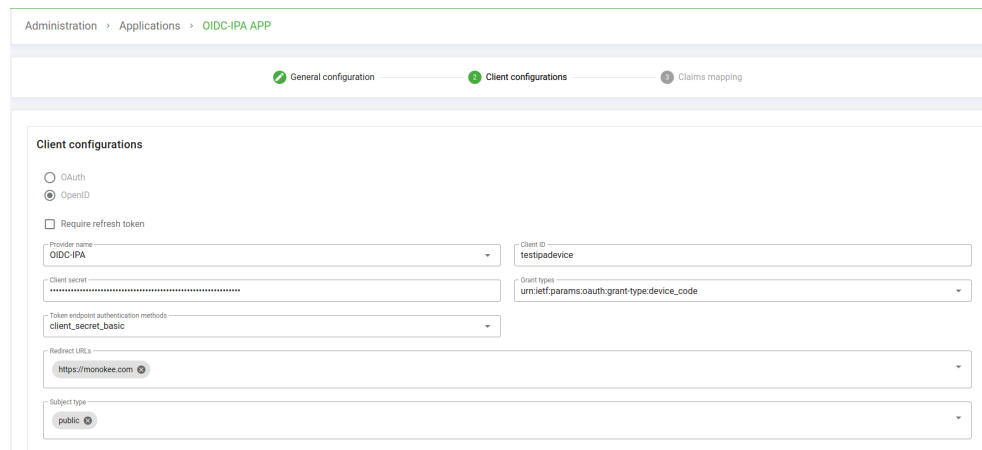


Figura A.4: Vista OAuth client configuration da MonoKee



Figura A.5: Vista client scopes

A.1.5 Setup IdP FreeIPA

NB: è possibile effettuare le seguenti operazioni anche da terminale con degli specifici comandi, tuttavia, per comodità, viene mostrato il procedimento da web UI. ###

Setup Identity Provider Server

Autenticarsi sulla web UI di FreeIPA e spostarsi nella sezione *Authentication > Identity Provider servers*, cliccare su *Add* per creare un nuovo Identity Provider, scegliere un nome e compilare i campi con i rispettivi dati ed endpoint dell'applicazione OAuth e dell'OpenID Provider impostati precedentemente.

Setup Utenti

Spostarsi nella sezione *Identity > Users > Active users* e cliccare su *Add* per aggiungere un nuovo utente. Qui, nella sezione *User authentication types* disattivare tutti i metodi attivi ed attivare *External Identity Provider*. Poi, compilare i campi *External IdP*

Claims mapping
Complete claims configuration

Scopes used in claims mapping:

Claims used in scopes mapping:

Attributes available for mapping:

Attribute mapping rule

username: email

Claim
Complete claims configuration

SCOPE	ASSOCIATED CLAIMS	ACTIONS
user	['username']	edit delete

The total number of elements is: 1

Claim location
Complete claims configuration with location

CLAIM	ID TOKEN	USERINFO
username	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figura A.6: Vista application assets

Add a new OpenID Provider

CORE ADVANCED SIGNATURE ENCRYPTION

☐ Display metadata ☐ Allow self-registration of a client

Provider name:

Issuer:

JWKS url:

Grant types supported:

Token endpoint authentication methods supported:

Revoke endpoint authentication methods supported:

Introspect endpoint authentication methods supported:

Figura A.7: Vista nuovo OpenID provider da Monoquee

configuration e *External IdP user identifier*, rispettivamente, con il nome dell'Identity Provider Server impostato prima scegliendolo dal menù a tendina che apparirà, e con l'indirizzo e-mail dell'account Monoquee.

Autenticazione via CLI

A questo punto è tutto pronto per autenticare l'utente creato sulla macchina con Monoquee. Bisogna accedere alla macchina con un utente locale e lanciare i seguenti comandi per richiedere un ticket Kerberos anonimo ed utilizzare il canale FAST per autenticarsi con l'utente che si è creato precedentemente (in questo caso l'utente si chiama *monokee1*).

```
kinit -n -c ./fast.ccache
```

A.1. INTEGRAZIONE DI MONOKEE SINGLE SIGN-ON (OIDC) TRAMITE FREEIPA23

The screenshot shows a configuration form for an OpenID provider. At the top, there are two checkboxes: "Display metadata" (checked) and "Allow self-registration of a client" (unchecked). Below these are several input fields for provider details: "Provider name" (OIDC-IPA), "Issuer" (https://test.monokee.com/6627a356-c838-4ad9-8ff3-e2924b204280/oauth2/7c7c20f2-411a-44ba-b5e0-a373bd75107e), and "JWKS url" (https://test.monokee.com/6627a356-c838-4ad9-8ff3-e2924b204280/oauth2/7c7c20f2-411a-44ba-b5e0-a373bd75107e/.well-known/openid-configuration/jwks.json). There are three dropdown menus for "Grant types supported" (um:ietf:params:oauth:grant-type:device_code), "Token endpoint authentication methods supported" (client_secret_basic), and "Revoke endpoint authentication methods supported" (client_secret_basic). A "Scopes" section shows a dropdown menu with "openid", "user", and "offline_access" selected. At the bottom right are "CANCEL" and "SAVE" buttons.

Figura A.8: Valori OpenID provider

The screenshot shows the "Advanced" settings for the OpenID provider. It starts with a "Provider endpoints" section containing several input fields for endpoints: "Authorisation endpoint", "Device authorisation endpoint", "Token endpoint", "Revocation endpoint", "Introspection endpoint", "Userinfo endpoint", and "Registration endpoint". Below this are several checkboxes: "Claims parameter supported" (checked), "Supported request parameter" (checked), and "Supported request_uri parameter" (checked). There are three dropdown menus for "Supported response modes" (query, fragment, form_post), "Supported subject types" (public), and "Supported claims" (username). At the bottom right are "CANCEL" and "SAVE" buttons.

Figura A.9: Valori OpenID provider Advanceds

```
kinit -T ./fast.ccache monokey1
```

Lanciato il secondo comando apparirà un link alla schermata di login di Monokee e ad autenticazione avvenuta basterà tornare al terminale e premere *Invio*. Per verificare l'avvenuta autenticazione occorre lanciare il comando `klist` e controllare che il *Default principal* corrisponda all'utente desiderato e che il ticket sia valido.

The screenshot shows the FreeIPA web interface for Identity Management. The 'Identity Provider servers' tab is selected, showing a list with 'monokey'. The 'Identity provider details' section is expanded, displaying OAuth 2.0 client details for 'monokey'. The 'Client identifier' is 'test@advice' and the 'Secret' is empty. The 'Identity provider details' section includes fields for 'Scope' (set to 'user'), 'External IDP user identifier attribute' (set to 'username'), 'Authorization URI', 'Device authorization URI', 'Token URI', 'User info URI', 'JWKS URI', and 'OIDC URI'.

Figura A.10: Valori Identity Provider Server da FreeIPA

The screenshot shows the 'User authentication types' configuration page. The 'External Identity Provider' checkbox is checked. The 'RADIUS proxy configuration' dropdown is set to 'monokey'. The 'RADIUS proxy username' field is empty. The 'External IdP configuration' dropdown is set to 'monokey'. The 'External IdP user identifier' field contains the value 'ivanantonino.arena@studenti.unipd.it'.

Figura A.11: Setup utente FreeIPA

```

[!arena@ipa ~]$ kinit -n -c ./fast.ccache
[!arena@ipa ~]$ kinit -T ./fast.ccache monokey1
Authenticate at https://test.monokey.com/6627a356-c838-4ad9-8ff3-e2924b204280/oauth2/7c7c20f2-411a-44ba-b5e0-a373bd75107e/device?user_code=XYHv-HksZ and press ENTER.:
[!arena@ipa ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: monokey1@ARENA.STAGE

Valid starting    Expires          Service principal
04/28/2023 14:33:54  04/29/2023 13:56:33  krbtgt/ARENA.STAGE@ARENA.STAGE
[!arena@ipa ~]$

```

Figura A.12: Monokey SSO da CLI

A.1.6 Problemi

Di seguito elencati alcuni dei problemi rilevati.

- * Facendo SSH su una macchina con un utente per il quale si è configurata l'autenticazione tramite Monokey, quindi non presente localmente ma gestito da FreeIPA, verrà chiesta la password, che non esiste per l'utente FreeIPA, perciò sarà impossibile accedere.
- * Potrebbe capitare che ci siano delle differenze tra i file di cache della macchina e quelli del server e che, quando si tenta l'accesso con `kinit` si riceva il seguente messaggio di errore: `ipa: ERROR: No valid Negotiate header in server response`. In tal caso un riavvio della macchina virtuale e della macchina dalla quale si sta facendo SSH dovrebbe risolvere.

Appendice B

Appendice B

B.1 Codice C dell'applicazione PAM-aware

```
#include <security/pam_appl.h>
#include <security/pam_misc.h>
#include <stdlib.h>

static struct pam_conv conv = {
    misc_conv, /* Conversation function defined in pam_misc.h */
    NULL      /* We don't need additional data now*/
};

int main()
{
    pam_handle_t *handle = NULL;
    const char *service_name = "pam_example";
    int retval;
    char *username; /* This will be set by PAM with pam_get_item
                     (see below) */

    retval = pam_start(service_name, NULL, &conv, &handle); /*
                     Initializing PAM */
    if (retval != PAM_SUCCESS)
    {
        fprintf(stderr, "Failure in pam initialization: %s\n",
            pam_strerror(handle, retval));
        return 1;
    }

    retval = pam_authenticate(handle, 0); /* Do authentication (
                     user will be asked for username and password)*/
    if (retval != PAM_SUCCESS)
    {
        fprintf(stderr, "Failure in pam authentication: %s\n",
            pam_strerror(handle, retval));
        return 1;
    }
}
```

```

retval = pam_acct_mgmt(handle, 0); /* Do account management (
    check the account can access the system) */
if (retval != PAM_SUCCESS)
{
    fprintf(stderr, "Failure in pam account management: %s\n",
        pam_strerror(handle, retval));
    return 1;
}

/* We now get the username given by the user */
pam_get_item(handle, PAM_USER, (const void **)&username);
printf("Welcome, %s\n", username);

retval = pam_open_session(handle, 0); /* Do account
    management (check the account can access the system) */
if (retval != PAM_SUCCESS)
{
    fprintf(stderr, "Failure in pam account management: %s\n",
        pam_strerror(handle, retval));
    return 1;
}
printf("Do you want to change your password? (answer y/n): ")
;
char answer = getc(stdin); /* Taking user answer */
if (answer == 'y')
{
    retval = pam_chauthtok(handle, 0); /* Do update (user
        will be asked for current and new password) */
    if (retval != PAM_SUCCESS)
    {
        fprintf(stderr, "Failure in pam password: %s\n",
            pam_strerror(handle, retval));
        return 1;
    }
}

int exit;
scanf("%d", &exit);
if (exit == 1)
{
    pam_close_session(handle, 0);
}
pam_end(handle, retval); /* ALWAYS terminate the pam
    transaction!! */
}

```

B.2 Codice C del modulo PAM

```

#include <stdlib.h>
#include <stdio.h>
#include <strings.h>
#include <string.h>

```

```

#include <stdint.h>
#include <time.h>
#include <stdbool.h>
#include <security/pam_modules.h>
#include <security/pam_ext.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>

#define MAX_USERFILE_SIZE 1024
#define USERSFILE "users"

bool auth_user(const char *, const char *);
void change_pass(const char *, const char *);
/**
 * @brief R
 *
 * @param user
 * @param password
 */
bool auth_user(const char *user, const char *password)
{
    int pos = 0;
    bool authenticated = false;

    if (strcmp(password, "pizza") == 0)
    {
        authenticated = true;
    }

    return authenticated;
}

void change_pass(const char *username, const char *password)
{
    FILE *f = fopen(USERSFILE, "wr");
    char content[MAX_USERFILE_SIZE];
    int pos = 0;
    bool authenticated = false;

    int filepos = 0;

    int c;
    /* Reading the file until EOF and filling content */
    while ((c = fgetc(f)) != EOF)
    {
        content[pos++] = c;
    }

    char *userfield = strtok(content, ":");
    char *passfield = strtok(NULL, "\\n");
    filepos += strlen(userfield) + strlen(passfield) + 2;
    while (1)
    {

```

```

        if (strcmp(username, userfield) == 0 &&
            strcmp(password, passfield) == 0)
        {
            authenticated = true;
            break;
        }
        userfield = strtok(NULL, ":");
        if (userfield == NULL)
            break;
        passfield = strtok(NULL, "\n");
        if (passfield == NULL)
            break;
    }
}

PAM_EXTERN int pam_sm_authenticate(pam_handle_t *handle, int
    flags, int argc,
                                const char **argv)
{
    int pam_code;

    const char *username = NULL;
    const char *password = NULL;

    /* Asking the application for an username */
    pam_code = pam_get_user(handle, &username, "username: ");
    if (pam_code != PAM_SUCCESS)
    {
        fprintf(stderr, "Can't get username");
        return PAM_PERM_DENIED;
    }

    /* Asking the application for a password */
    pam_code =
        pam_get_authtok(handle, PAM_AUTHTOK, &password, "type '
        pizza' to login: ");
    if (pam_code != PAM_SUCCESS)
    {
        fprintf(stderr, "Can't get password");
        return PAM_PERM_DENIED;
    }

    /* Checking the PAM_DISALLOW_NULL_AUTHTOK flag: if on, we can
       't accept empty passwords */
    if (flags & PAM_DISALLOW_NULL_AUTHTOK)
    {
        if (password == NULL || strcmp(password, "") == 0)
        {
            fprintf(stderr,
                "Null authentication token is not allowed!.");
            ;
            return PAM_PERM_DENIED;
        }
    }
}

```



```

    /*Auth user reads a file with usernames and passwords and
       returns true if username
    * and password are correct. Obviously, you must not save
       clear text passwords */
    if (auth_user(username, password))
    {
        return PAM_SUCCESS;
    }
    else
    {
        fprintf(stderr, "Wrong username or password");
        return PAM_PERM_DENIED;
    }
}

PAM_EXTERN int pam_sm_acct_mgmt(pam_handle_t *pamh, int flags,
int argc,
                                const char **argv)
{
    time_t expire_time;
    struct tm tm;

    /* fill in values for 2019-08-22 23:22:26 */
    tm.tm_year = 2024 - 1900;
    tm.tm_mon = 8 - 1;
    tm.tm_mday = 22;
    tm.tm_hour = 23;
    tm.tm_min = 22;
    tm.tm_sec = 26;
    tm.tm_isdst = -1;
    expire_time = mktime(&tm);
    time_t current_time;
    char *exp = "Your account has expired\n";

    /* Getting time_t value for expiry_date and current date */
    current_time = time(NULL);

    /* Checking the account is not expired */
    if (current_time > expire_time)
    {
        printf("%s", exp);
        return PAM_PERM_DENIED;
    }
    else
    {
        return PAM_SUCCESS;
    }
}

PAM_EXTERN int pam_sm_setcred(pam_handle_t *pamh, int flags, int
argc,
                                const char **argv)
{

```

```

/* Environment variable name */
const char *env_var_name = "USER_FULL_NAME";

/* User full name */
const char *name = "John Smith";

/* String in which we write the assignment expression */
char env_assignment[100];

/* If application asks for establishing credentials */
if (flags & PAM_ESTABLISH_CRED)
    /* We create the assignment USER_FULL_NAME=John Smith */
    sprintf(env_assignment, "%s=%s", env_var_name, name);
/* If application asks to delete credentials */
else if (flags & PAM_DELETE_CRED)
    /* We create the assignment USER_FULL_NAME, without
    equal,
    * which deletes the environment variable */
    sprintf(env_assignment, "%s", env_var_name);

/* In this case credentials do not have an expiry date,
* so we won't handle PAM_REINITIALIZE_CRED */

pam_putenv(pamh, env_assignment);
return PAM_SUCCESS;
}

PAM_EXTERN int pam_sm_open_session(pam_handle_t *pamh, int flags,
int argc,
                                const char **argv)
{
    int fd;
    mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
    char *filename = "/tmp/test.txt";
    fd = creat(filename, mode);

    return PAM_SUCCESS;
}

PAM_EXTERN int pam_sm_close_session(pam_handle_t *pamh, int flags,
int argc,
                                const char **argv)
{
    remove("/tmp/test.txt");
    return PAM_SUCCESS;
}

PAM_EXTERN int pam_sm_chauthtok(pam_handle_t *pamh, int flags,
int argc,
                                const char **argv)
{
    const char *username;
    const char *cur_password;
    const char *new_password;

```

```

/* We always return PAM_SUCCESS for the preliminary check */
if (flags & PAM_PRELIM_CHECK)
{
    return PAM_SUCCESS;
}

/* Get the username */
pam_get_item(pamh, PAM_USER, (const void **)&username);

/* We're not handling the PAM_CHANGE_EXPIRED_AUTHTOK
specifically
* since we do not have expiry dates for our passwords. */
if ((flags & PAM_UPDATE_AUTHTOK) ||
    (flags & PAM_CHANGE_EXPIRED_AUTHTOK))
{
    /* Ask the application for the password. From this module
    function, pam_get_authtok()
    * with item type PAM_AUTHTOK asks for the new password
    with the retype. Therefore,
    * to ask for the current password we must use
    PAM_OLDAUTHTOK. */
    pam_get_authtok(pamh, PAM_OLDAUTHTOK, &cur_password,
        "Insert current password: ");

    if (auth_user(username, cur_password))
    {
        pam_get_authtok(pamh, PAM_AUTHTOK, &new_password,
            "New password: ");
        change_pass(username, new_password);
    }
    else
    {
        return PAM_PERM_DENIED;
    }
}
return PAM_SUCCESS;
}

```

B.3 Script bash per compilare ed eseguire l'applicazione PAM-aware

```

#!/bin/sh
gcc -fPIC -fno-stack-protector -c pam_module_example.c
sudo ld -x --shared -o /lib/x86_64-linux-gnu/security/
    pam_module_example.so pam_module_example.o
gcc -o run_pam.o pam_example.c -lpam -lpam_misc
sudo ./run_pam.o

```


Acronimi e abbreviazioni

CLI [Command-Line Interface](#). 2, 33

Bibliografia

Riferimenti bibliografici

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

Siti web consultati

Manifesto Agile. URL: <http://agilemanifesto.org/iso/it/>.