AUTENTICAZIONE

Marco Console
console@diag.uniroma1.it

NOTA PRELIMINARE

- Il seguente materiale didattico è basato su un insieme di lucidi prepato dal **Prof. Leonardo Querzoni.**

- Ringrazio il **Prof. Leonardo Querzoni** per avermi concesso di usare il materiale da lui prodotto.

IDENTITÀ DIGITALI

IL PROBLEMA

- L’autenticazione è un problema molto intuitivo.

- Gli utenti utilizzano uno strumento informatico con una certa identità digitale mantenendo tutti i loro attributi.
  - Accedo a facebook come my-email@email.com

- Come gestori, vogliamo accertarci che l’identità digitale dichiarata rappresenti davvero l’utente.
  - Accesso alle risorse
  - Possibilità di eseguire determinate azioni.

CONCETTI

- **Entity**: una singola entità fisica che vuole identificarsi
  - Persone o entità astratte come persone giuridiche, aziende…

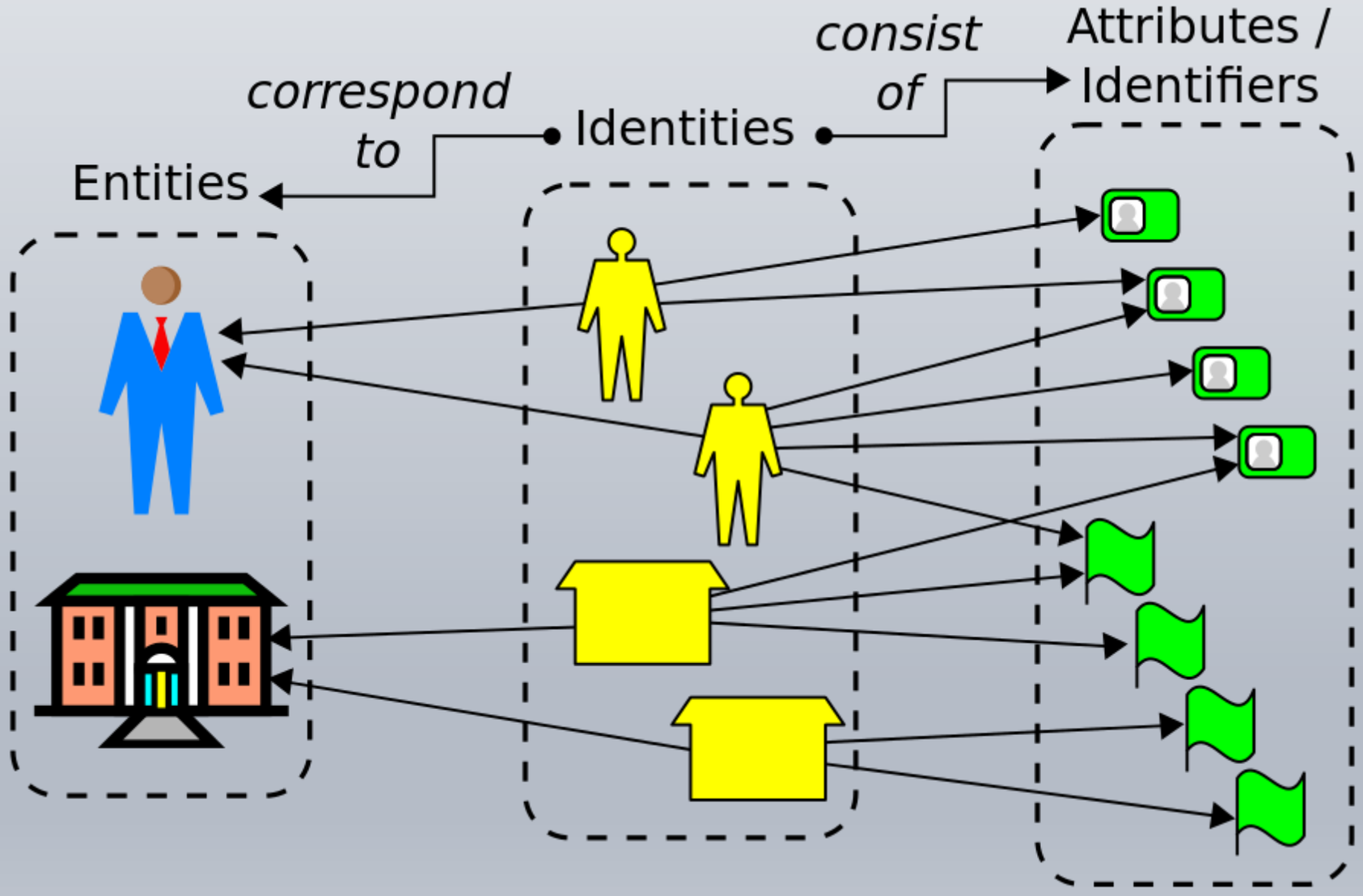
- **(Digital) Identity**: l'identità che l'entità vuole assumere nel contesto dell'applicazione.
  - Identità personali o correlate a concetti astratti come aziende.

- **Attribute:** attributi correlati alle identità.
  - Saldo del conto bancario, lista dei trattamenti medici effettuati…

- Ogni identity deve essere correlata a una singola entity.

IDENTITÀ DIGITALI

Le identità digitali sono il modo in cui ogni individuo è rappresentato nel mondo virtualizzato dei sistemi software.



RELAZIONI FRA I CONCETTI

- **Entity**: una singola entità fisica che vuole identificarsi
  - Può essere collegata a (0, n) Digital Identity

- **(Digital) Identity**: l'identità che l'entità vuole assumere nel contesto dell'applicazione.
  - E' collegata a (1, 1) Entity
  - Può essere collegata a (0, n) Attributes

- **Attribute:** attributi correlati alle identità.
  - Può essere collegata a (0, n) Attributes

IDENTITÀ – ESEMPIO

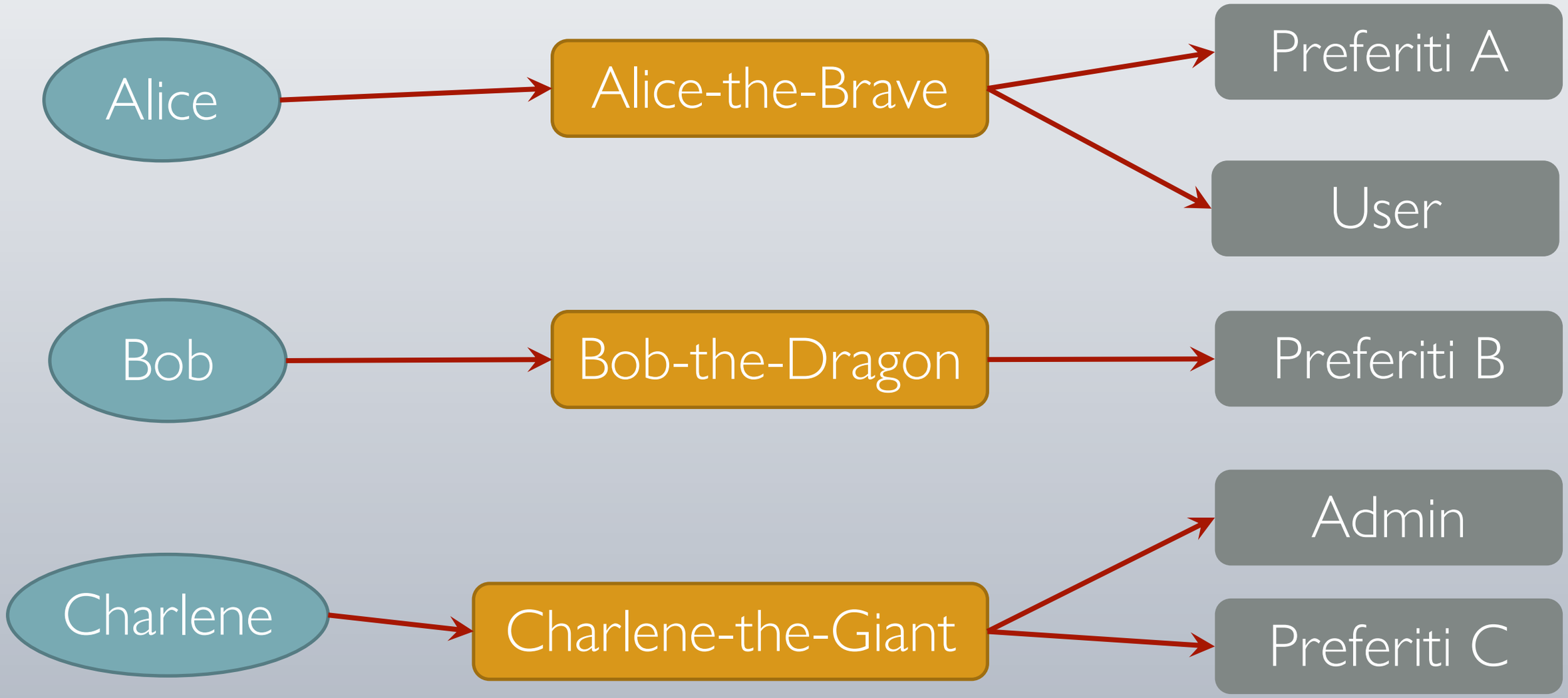
- **Contesto:** una forum online di fumetti.

- **Entity:**
  - Alice, Bob, Charlene

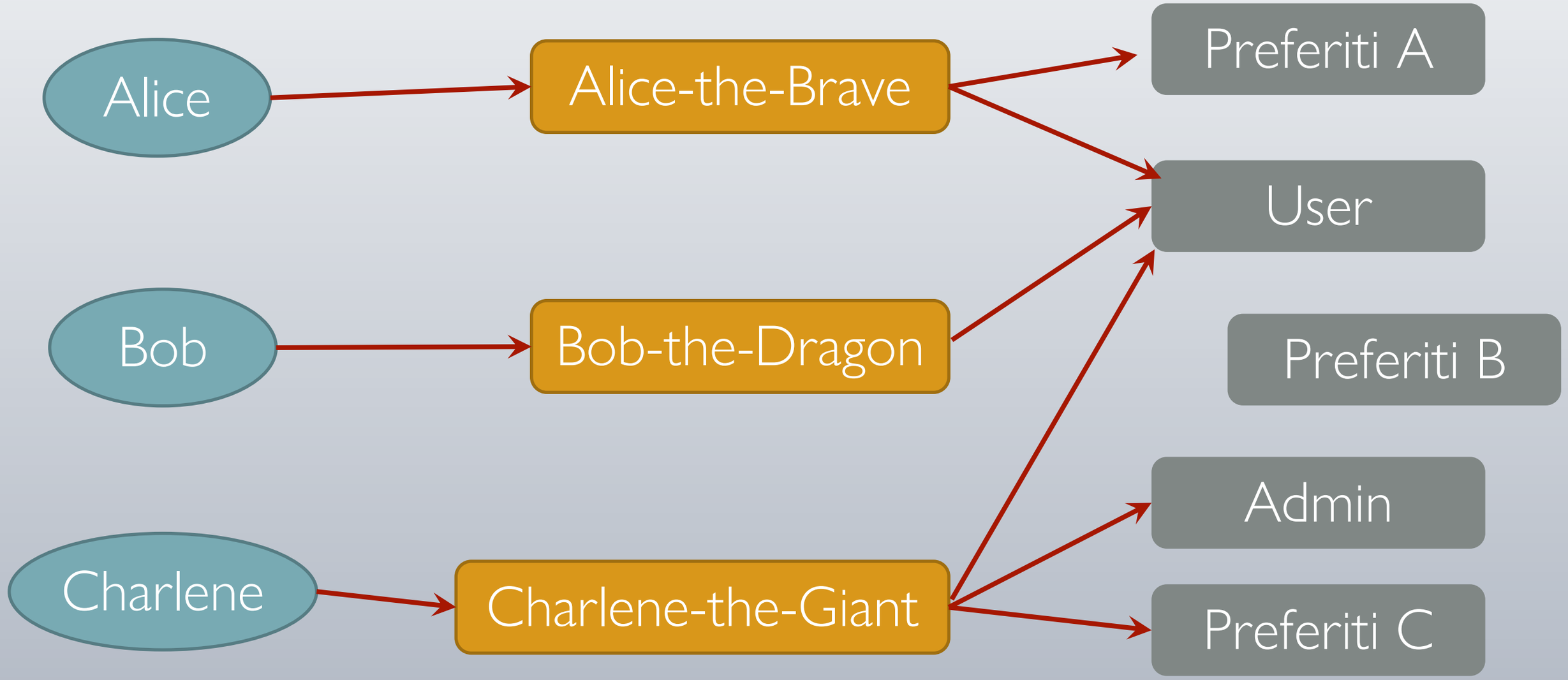
- **Identity**:
  - Alice-the-Brave, Bob-the-Dragon, Charlene-the-Giant

- **Attribute:**
  - Personali: Lista Preferiti A, Lista Preferiti B, Lista Preferiti C.
  - Ruoli: Administrator, User

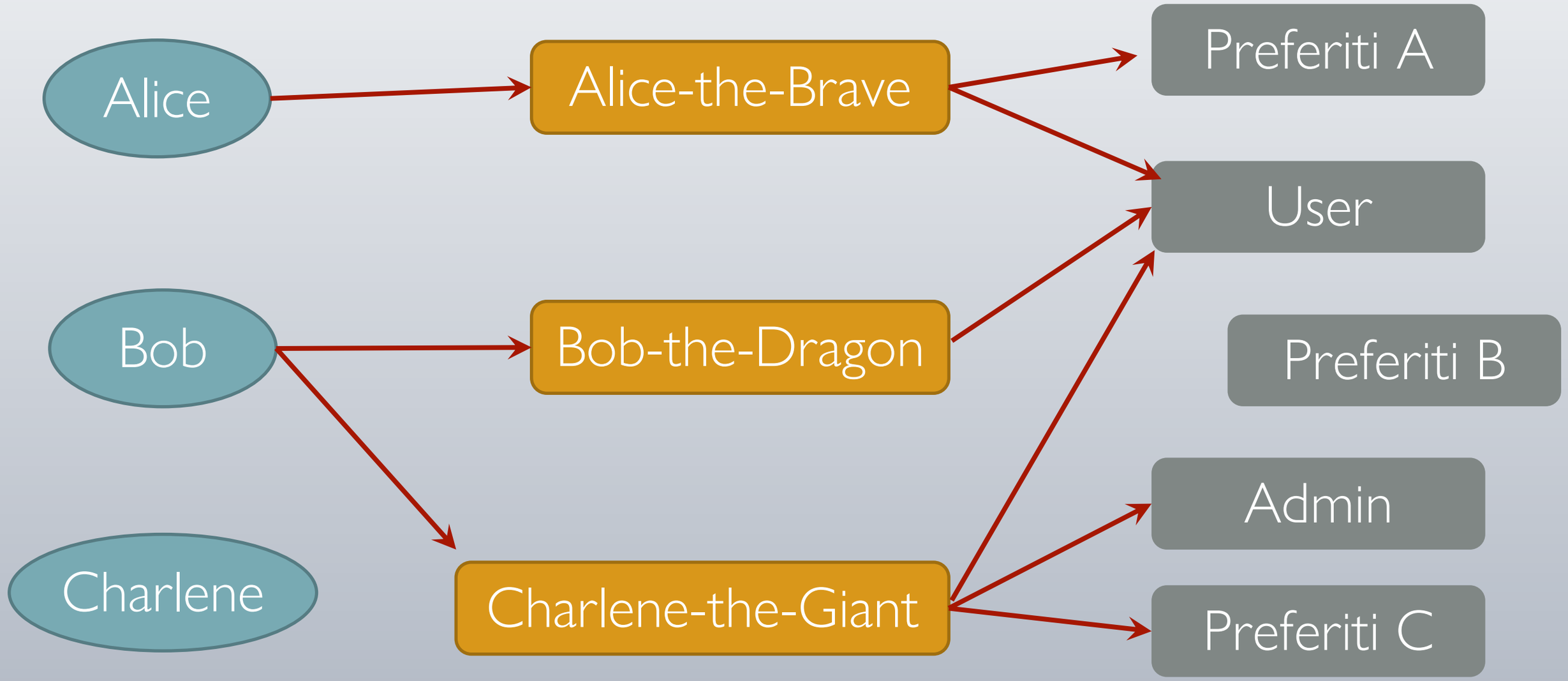
IDENTITÀ – ESEMPIO



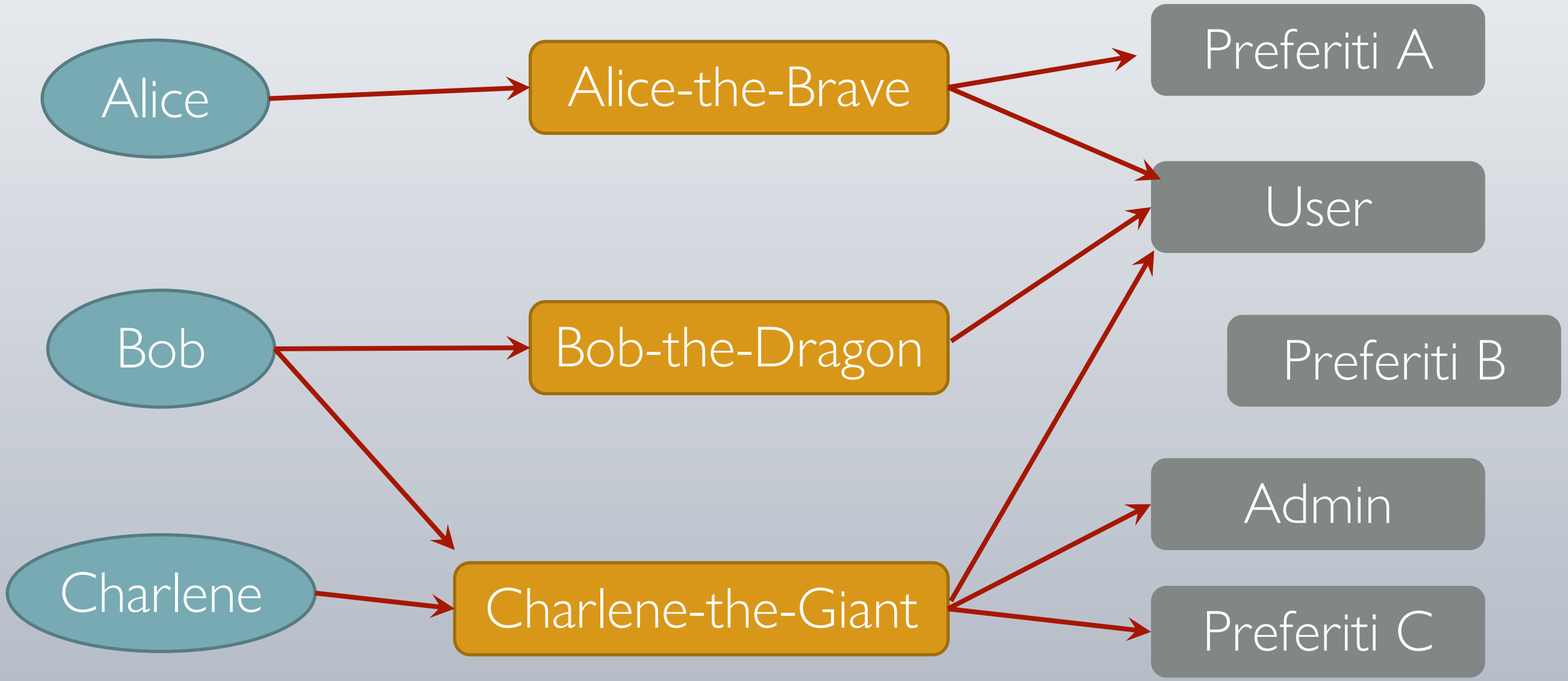
IDENTITÀ – ESEMPIO



IDENTITÀ – ESEMPIO

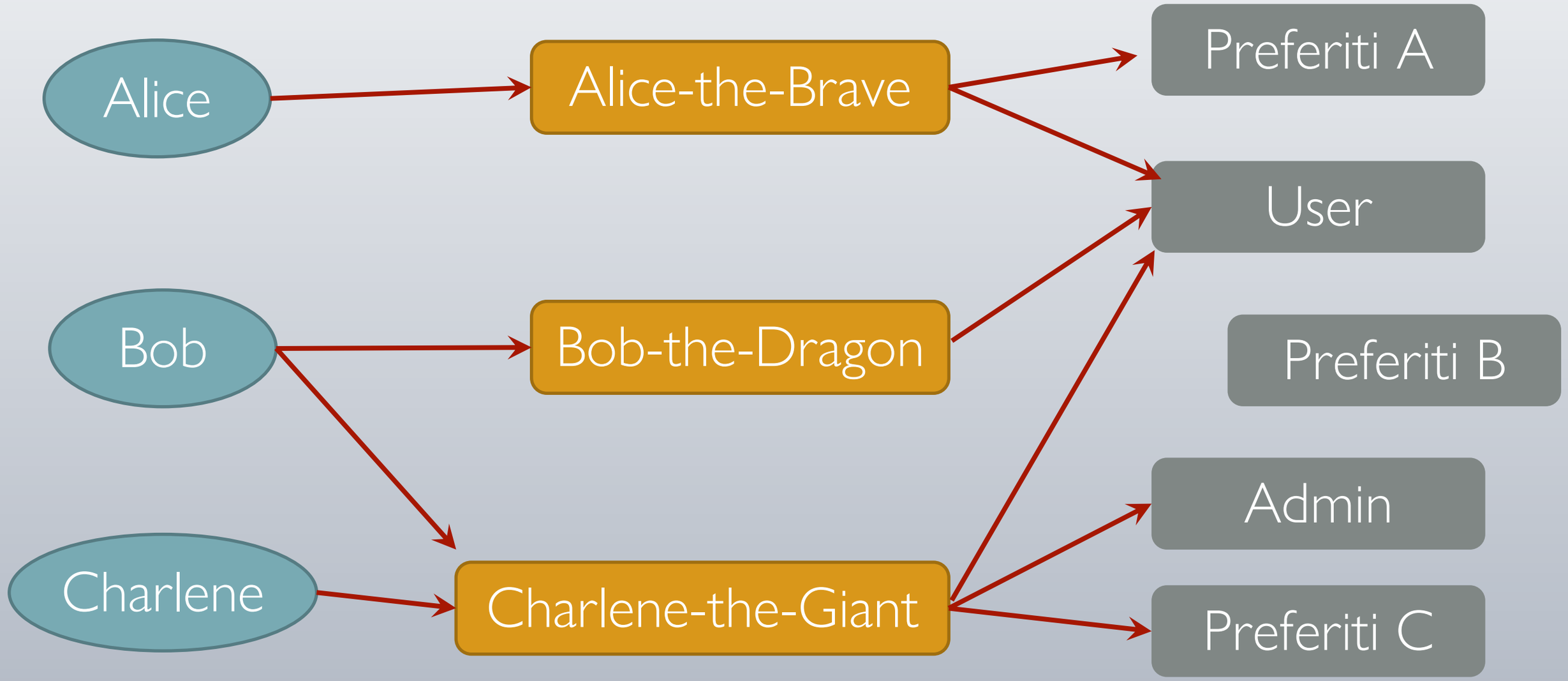


DOMANDA



E’ un sistema di identità valido?

DOMANDA



E’ un sistema di identità valido? **NO!**

AUTENTICAZIONE E AUTORIZZAZIONE

- **Il problema dell'autenticazione.**
  - Assegnare la giusta identità digitale a una entità
  - Il sistema può utilizzare gli attributi dell'identità digitale relativi all'entità.

- **Il problema dell'autorizzazione.**
  - Quali privilegi possiede la singola identità?
  - Solitamente i privilegi sono funzione degli attributi.
  - Vedremo l'autorizzazione nelle lezioni successive.

METODI DI AUTENTICAZIONE

AUTENTICAZIONE

- L'autenticazione è l'operazione con cui una identità digitale viene associata ad un individuo in modo che a questo vengano associati degli attributi.
- Solitamente avviene in due fasi:

1. **Identificazione**: dichiarare in modo univoco l'identità digitale tramite cui si intende interagire con il sistema.
2. **Verifica**: fornire una prova della corrispondenza tra l'identità digitale richiesta e la propria persona.

Autenticazione = Identificazione + Verifica

IDENTIFICAZIONE

- **Identificazione**: dichiarare in modo univoco l'identità digitale tramite cui si intende interagire con il sistema.

- Dipende dal sistema considerato.
  - Solitamente dichiarare il nome dell'identità.

- Per i progetti userete un username o una email.

AUTENTICAZIONE

La fase di verifica tipicamente può avvenire con tre diverse modalità:

- Something you **know**
- Something you **have**
- Something you **are**

AUTENTICAZIONE

La fase di verifica tipicamente può avvenire con tre diverse modalità:

- Something you **know**
  - **Solitamente un segreto come una password**
- Something you **have**
  - **Un oggetto specifico come una smartcard, un dongle, o un certo numero di cellulare.**
- Something you **are**
  - **Caratteristiche biometriche come impronte digitali o forma del volto.**

L’**autenticazione a due fattori** consiste in una verifica con due modalità contemporanee

- Es: password + SIM telefono
- Es: bancomat + PIN

DUE FATTORI – ESEMPIO

- Un esempio classico è il prelievo di denaro all'ATM.

1. Inseriamo la carta nella macchina
   - Something you **have**
2. La macchina ci chiede il pin
   - Something you **know**

PASSWORDS

PASSWORDS

- La tipologia di verifica più comune è tramite password

- Questo potrebbe non essere più vero entro breve.
  - Le password potrebbero essere compromesse dall'utente.
  - Le password più brevi di 10 caratteri sono deboli
  - Le password più brevi di 20 caratteri solo minuscoli sono deboli

- Una buona password dovrebbe contenere:
  - Almeno un numero
  - Almeno un simbolo
  - Almeno una lettera maiuscola
  - Non contenere termini del dizionario

PASSWORDS -- REQUISITI

- A causa della semplicità con cui le password vengono compromesse, i sistemi informatici applicano requisiti sempre più stringenti per le password.

- Lunghezza minima
  - Spesso maggiore di 8 caratteri.
- Alfabeto variegato.
  - «La password deve contenere almeno un numero una lettera maiuscola e un carattere speciale»
- Non deve essere simile a una parola del dizionario.
  - «La password è debole»

HUMAN FACTORS OF SECURITY

- Risultato
  - Gli utenti usano sempre la stessa password per tutti i servizi
    - Le password uniche sono difficili da ricordare
    - La violazione di una delle identità comporta la violazione delle altre
  - Le password non vengono mai cambiate (neanche dopo essere state compromesse!)
    - La compromissione dell’identità garantisce all’attaccante privilegi duraturi nel tempo.
  - Gli utenti si appuntano le password su carta.
    - Facili da rubare se ne vale la pena (conto corrente).
  - Gli utenti dimenticano la password.

HUMAN FACTORS OF SECURITY

2020. (How) Do people change their passwords after a breach?

Sruti Bhagavatula , Lujo Bauer, Apu Kapadia

https://arxiv.org/abs/2010.09853

“Of the 249 participants, 63 had accounts on breached domains; only 33% of the 63 changed their passwords and only 13% (of 63) did so within three months of the announcement.”

IL RISULTATO....

STRATEGIE CONSIGLIATE

- Produrre password complesse facili da ricordare.
  - Sandwiching, padding, substitution

- Layering:
  - password con complessità legata al loro uso

- Mai usare la stessa password due volte.

- Usare un password manager.

PASSWORD COMPLESSI

- Sandwiching + padding
  - Scegliere un elenco di parole dal dizionario (sandwiching)
  - Concatenarle usando una stringa specifica (padding)

- **Esempio:** rosso, tramonto, _-_
  - Rosso_-_Tramonto

- Substitution.
  - Sostituire certi caratteri con altri.

- **Esempio:** password → p455w0rd
  - Semplici sostituzioni non sono più considerate sicure.

QUALCHE LINK

- La mia password è sicura?
  - https://www.my1login.com/resources/password-strength-test/
- Keepass:
  - https://keepass.info/
  - Facile da usare e open-source.
  - Criptografia AES per il database delle password.
  - Genera password random per voi.

PASSWORDS MANAGEMENT

PASSWORD STORAGE

- Problema: come tenere traccia delle identità degli utenti
  - Database
  - Directory

- Come memorizzare la password in modo da garantirne la confidenzialità?

PASSWORD STORAGE

Soluzione 1

- Memorizzare la password “in chiaro” nel database

Svantaggi:

- In caso di compromissione del database tutte le password sono visibili
- **Nota:** anche se il database è sicuro, quella di salvare password in chiaro è una pratica sbagliata.
- L’amministratore del database potrebbe impersonare gli utenti.

FUNZIONI DI HASH

- **One-way hash function**
  - Una funziona che trasforma stringhe in altre stringhe con alcune proprietà

- Non necessariamente univoca ma con bassa collisione
  - $f(a) = f(b)$ con bassa probabilità se $a \neq b$.

- Difficile da invertire.
  - Dato $f(a)$, calcolare a richiede molto tempo.

- Facile da calcolare
  - Dato a, calcolare $f(a)$ richiede un tempo ragionevole.

FUNZIONI DI HASH E PASSWORDS

- Salviamo nel database l'hash delle password.
  - Data a, salviamo $f(a)$

- Quando l'utente richiedere l'accesso ci fornisce la password. Il sistema ne calcola l'hash.
  - Data p, calcoliamo $f(p)$

- Se l'hash è lo stesso l'autenticazione va a buon fine.
  - Se $f(p) = f(a)$ garantiamo l'accesso.

PASSWORD STORAGE

Soluzione 2

- Memorizzare *hash*(password) nel database

Pro:

- Le password non sono immediatamente visibili

Svantaggi:

- Vecchie funzioni di hash come md5 e sha1 non sono più considerate sicure
- Un cluster con 25 GPU nel 2012 generava 348 miliardi di hash al secondo. Oggi ?
- Rainbow tables attack

RAINBOW TABLES

- Una soluzione di trade-off fra spazio e tempo.
  - Precomputiamo una grande quantità di hash.

- Salviamo una lista di hash con relativa password associata.
  - Coppie $< a, f(a) >$

- Dato un hash $f(k)$, controlliamo se nella tabella è presente una coppia $< k', f(k) >$

- Ritorniamo il valore di k'.

- Perché funziona?

RAINBOW TABLES -- NOTE

- Invece di esplorare uno spazio molto grande di password, le Rainbow Table ci permettono di concentrarci su hash di password ricorrenti.
  - Lo spazio è molto meno costoso della capacità computazionale.
- In caso di collisione, avremmo comunque una stringa che il sistema validerebbe come una password.
- Ci sono tecniche per ridurre lo spazio utilizzato.
  - Hashing chains

PASSWORD STORAGE

Soluzione 3

- Memorizzare *sha1*(fixed_secret + password) nel database

Pro:

- Le rainbow tables sono inutile perchè il fixed_secret non è incluso nelle table.

Svantaggi:

- Se il fixed_secret è lo stesso per tutte le password prima o poi qualcuno lo scoprirà.
  - Attacchi statistici o semplice forza bruta.
- Scoperto il fixed_secret, l'attaccante può computare una rainbow table che lavora contemporaneamente su tutte le password

PASSWORD STORAGE

Soluzione 4

- Memorizzare *sha1*(per_password_salt + password) nel database

Pro:

- Ogni attacco può lavorare solo su una singola password alla volta
- Il salt non deve essere necessariamente segreto.

Svantaggi:

- Non fornisce una soluzione al potere computazionale continuamente crescente a prezzi sempre più bassi

PASSWORD STORAGE

Soluzione 5

- Memorizzare *bcrypt*(per_password_salt + password) nel database

Pro:

- Funzioni di hash moderne. Soluzione ad oggi ottimale.

Altre funzioni: Argon2, PBKDF2, scrypt

- Funzioni di hash computazionalmente *inefficienti*
- Inefficiente anche se parallelizzate
- Complessità parametrizzata.

PASSWORD STORAGE

BCrypt

- random salt + fattore di costo + password
- Utilizza il salt per trasformare la password in una chiave
  - La complessità di questa fase è regolata dal fattore di costo
- Utilizza la chiave ottenuta per codificare una stringa nota
- La stringa codificata rappresenta la versione sicura della password

Vantaggi:

- Se le CPU diventano più veloci basta aumentare il fattore di costo
- BCrypt è inerentemente lento
- BCrypt non è ad oggi ottimizzabile per il funzionamento su GPU/ASIC/FPGA

SSO E OAUTH

SINGLE SIGN ON

- Il protocollo Single Sign-On (SSO) permette di autenticarci in un dominio e ottenere un token che prova la nostra identità in tutte le applicazioni associati.

- I token possono avere forme diverse.
  - XML, JSON …

- I token vengono firmati digitalmente dal dominio così che possano essere validati.

VANTAGGI

- Esperienza utente migliorata
  - Più semplice e percepito come più sicuro.
  - Un solo log-in permette l’accesso a molte applicazioni
- Meno lavoro per gli sviluppatori e gli amministratori
  - Lo sviluppo e la manutenzione del controllo delle identità è spesso complesso.
  - Regulatory compliance
- Un singolo punto critico
  - Con sicurezza rafforzata sviluppata da team di esperti.

SVANTAGGI

- Difficile da implementare in applicazioni esistenti
  - Dipende dalle tecnologie usate
- «Unattended Desktop»
  - Una macchina lasciata incustodita potrebbe dare accesso alle risorse protette da SSO
- Un singolo punto critico
  - Se il dominio auntenticatore viene compromesso, tutti le applicazioni che si appoggiano a questo dominio potrebbero essere compromesse.

OAUTH

- Un protocollo che permette di usare account su domini diversi per l’autenticazione.
  - Esempio: account di google per autenticarsi su github
- Non esattamente SSO.
  - In SSO usiamo un unico account per tanti servizi
  - OAUTH ci permette di usare account di specifici servizi per l’autenticazione.
- Chi lo usa? Facebook, Google, Amazon, GitHub …

AUTENTICAZIONE CON OAUTH