Alma Mater Studiorum · Università di Bologna

Scuola di Ingegneria e Architettura
Dipartimento di Informatica · Scienza e Ingegneria · DISI
Corso di Laurea in Ingegneria Informatica

PROGETTO DI SISTEMI BASATI SU DEEP NEURAL NETWORK PER LA RILEVAZIONE DI SIMILARITÀ TRA PASSWORD

Relatore:
Prof. Marco Prandini

Presentata da:
Karina Chichifoi

Correlatore:
Davide Berardi
Andrea Melis

Anno Accademico 2019/2020

Introduzione

L'implementazione di strumenti atti a combattere gli attacchi di tipo denial of service, sono ormai da molto tempo materia di studio e di ricerca. Questi attacchi possono rappresentare una minaccia grave, soprattutto quando si verificano su servizi cruciali e sensibili. Per tale motivo, negli anni sono state sviluppate molte strategie diverse per impedire ai singoli o ai gruppi di computer (noti anche come "zombie armies") di attaccare servizi o reti. Attualmente una delle metodologie di protezione più comuni agli attacchi DoS è il Rate-limiting, che permette di imporre una frequenza massima accettabile per le richieste, al fine di evitare sovraccarichi e congestionamenti delle risorse. In particolare, nei sistemi su larga scala, la limitazione della frequenza rappresenta uno strumento essenziale per garantire la disponibilità e l'integrità delle risorse e dei servizi.

Un altro campo di interesse in cui si sono fatti molti progressi negli ultimi anni, grazie soprattutto all'avvento delle tecnologie Block-chain, è stato quello dell'anonimato in rete. Con il termine anonimato in rete ci si riferisce alla condizione in cui, sulla base di una conoscenza parziale o totale delle interazioni di un utente su una rete, non è possibile risalire all'identità dell'utente stesso, permettendo un'interazione con i servizi offerti dalla rete in assoluta riservatezza. L'anonimato in rete presenta molteplici vantaggi, soprattutto in contesti in cui la privacy costituisce un requisito fondamentale, come ad esempio nelle votazioni o nelle transazioni finanziarie. Inoltre, la possibilità di mantenere l'anonimato può rivelarsi altrettanto utile in ambiti più diffusi come le conversazioni online o sui social network, garantendo la libertà di espressione e la tutela della propria sfera privata.

Tuttavia, l'anonimato in rete presenta anche alcune criticità, tra cui la principale è rappresentata dalla difficoltà di controllo. La ragione è da individuare nel fatto che le metodologie di sicurezza a livello applicazione (pila ISO/OSI) generalmente si fondano sulla sorveglianza del comportamento degli utenti o dei dispositivi, nel corso del tempo, al fine di rilevare i pattern di attività che potrebbero suggerire la presenza di un attacco. In un contesto anonimo, in cui le identità degli utenti non sono tracciabili, ciò diviene notevolmente più arduo. Esistono strumenti di rate-limiting efficaci a livelli inferiori della pila ISO/OSI, come a livello di trasporto o di sessione. Tuttavia, la loro attuazione comporta alcuni svantaggi, come il rischio di limitare o bloccare numerosi indirizzi IP tradotti sotto una NAT, minare la privacy degli utenti disattivando le protezioni come TLS per effettuare DPI o ancora essere elusi da tecniche di mascheramento, come l'IP spoofing. Per tali motivazioni, per un servizio diffuso in ambiente anonimo, non è possibile fare affidamento esclusivamente su questi strumenti.

La presente tesi affronta la discussione di un protocollo chiamato RLN (Rate-Limiting Nullifier) basato sulla tecnologia zk-SNARK, che permette di attuare un rate-limiting in ambiente anonimo. Il protocollo si delinea in tre parti generali che si diversificano nei dettagli in base al dominio applicativo di utilizzo.

Fasi del protocollo:

- **Registrazione:** Durante questa fase, gli utenti che desiderano accedere al servizio devono registrarsi fornendo una prova del possesso di determinati requisiti. Questa prova di possesso è denominata "identity commitment" e viene conservata e utilizzata nelle fasi successive del protocollo. I dati personali che soddisfano i requisiti sono definiti "stake" e possono assumere forme diverse a seconda del contesto applicativo. Ad esempio, possono essere costituiti da un profilo su un social network, dall'indirizzo di un portafoglio di criptovalute o da un'identità digitale come lo SPID o la CIE.

  Gli stake non vengono conservati dal protocollo e, alla fine della fase di registrazione, l'informazione fornita al servizio consiste unicamente nel fatto che un nuovo utente anonimo che soddisfa le specifiche della registrazione è stato aggiunto al servizio.

- **Interazione:** Dopo essersi registrati, gli utenti hanno la possibilità di interagire con il servizio attraverso l'invio di richieste. Ad ogni richiesta, gli utenti sono tenuti a fornire una prova di appartenenza al sistema. Tale prova è generata tramite la tecnologia zk-SNARK, che consente di dimostrare al servizio che l'utente è effettivamente un membro legittimo senza rivelare alcuna informazione sulla sua identità. Inoltre, il protocollo consente di implementare una regola di rate-limiting, che, se non rispettata, porta l'utente a rivelare la propria stake.

  Ciò consente di scoprire e gestire chi attua comportamenti di spam o Dos, e di procedere alla fase successiva.

- **Punizione:** La tipologia e il grado di punizione dipendono molto dal contesto applicativo. Ad esempio, alcune punizioni o misure di "slashing" possono comportare la rimozione del membro dal gruppo, con conseguente impossibilità di interagire con il sistema o la perdita dell'anonimato. In presenza di una stake, è possibile prevedere sanzioni più articolate, come l'inserimento dell'identità virtuale in un registro di esclusione per altri servizi, o la rivelazione dell'indirizzo del portafoglio criptovalute dell'utente e il sequestro dei fondi.

Di seguito si cercherà di guardare a tutto tondo le tecnologie e le idee alla base di zk-Snark e del protocollo RLN. Inoltre si mostrerà l'implementazione di un piccolo prototipo che utilizza questo protocollo per evitare la sovraccarico delle risorse in un servizio basato su API.

1 | Stato dell'arte

1.1 Denial-of-Service (DoS)

Gli attacchi di tipo Denial-of-Service (DoS) sono una tipologia di attacchi informatici estremamente diffusi, la cui frequenza è cresciuta negli anni sia in termini quantitativi che qualitativi. Questo incremento è stato determinato dalla relativa facilità con cui è possibile condurre un attacco di questo tipo, nonché dalle sue conseguenze, che in base alle contromisure adottate e alla struttura del servizio attaccato possono essere più o meno gravi. Nella tesi ci concentreremo solamente, sull'analisi della metodologia di protezione denominate rate-limiting. Queste strategie possono risultare estremamente efficaci in alcune situazioni, come dimostrato dall'evento accaduto ad agosto 2022 a Google [1], che grazie all'attivazione di un rate-limiter è stato in grado di contrastare con successo il più grande attacco di tipo DDoS a livello applicativo mai registrato. Tale attacco, proveniente da oltre 5.256 sorgenti IP dislocate in 132 paesi differenti, ha raggiunto il picco di 46 milioni di richieste al secondo.

In generale, un una regola per il rate limiting consiste in un semplice conteggio delle occorrenze delle richieste in un lasso di tempo. Tuttavia, esistono diverse tecniche per misurare e limitare la frequenza di tali richieste, ognuna con i propri usi e implicazioni.

- **Fixed window :** L'obiettivo del training è combinare le rappresentazioni delle parole limitrofe per prevedere la parola centrale. Tra tutte le strategie che vedremo è la più semplice da implementare, consiste nello stabilire un limite massimo al numero di richieste che possono essere inviate in un determinato intervallo di tempo, detto "finestra". Ad esempio, si potrebbe limitare il numero di richieste a 100 ogni minuto. Questo limite viene applicato in modo uniforme all'interno della finestra temporale. Ciò significa che, una volta raggiunto il limite massimo di richieste consentite, l'utente o l'applicazione deve attendere il termine della finestra temporale per poter inviare nuove richieste. Uno svantaggio significativo di questa strategia è la possibile concentrazione di richieste tutte in una porzione della finestra, rischiando un sovraccarico del servizio. Caso notevole è quello in cui si concentrino tutte le richieste di una finestra al margine della fine e tutte le richieste della finestra successiva al margine dell'inizio questo comporta che il sistema in una durata equivalente a quella della finestra ha ricevuto il doppio delle richieste consentite.

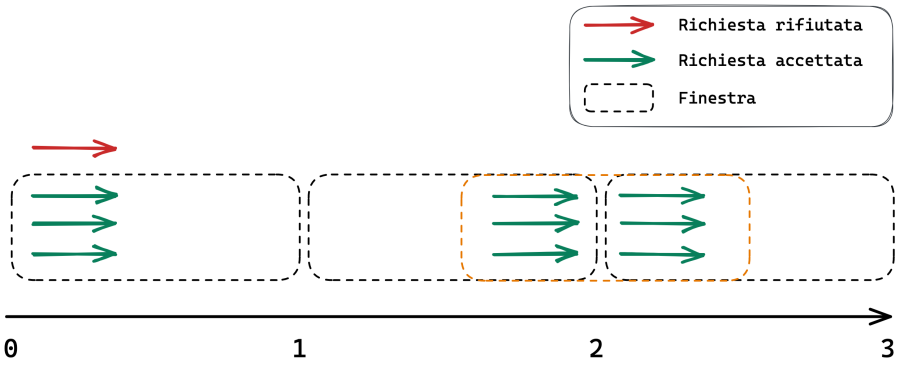

Figura 1.1: diagramma Rate-limiting fixed window

- **Token bucket :** Il funzionamento del token bucket prevede che non tutte le richieste di un servizio vengano mappate 1:1 con la richiesta di risorse, poiché alcune richieste potrebbero richiedere più risorse di altre. Per questo motivo, viene mantenuto un contatore di risorse, che viene scalato per ogni richiesta, del numero di token necessari per portare a termine il lavoro. Il contatore ha una frequenza di riempimento, ovvero a ogni unità di tempo viene reimpostato al suo valore massimo. Quando un utente o un'applicazione invia una richiesta al sistema, il sistema verifica se ci sono abbastanza token disponibili per soddisfare la richiesta. Se non ci sono abbastanza token, la richiesta viene respinta. In questo modo, le richieste vengono accettate solo se c'è abbastanza capacità per soddisfarle.

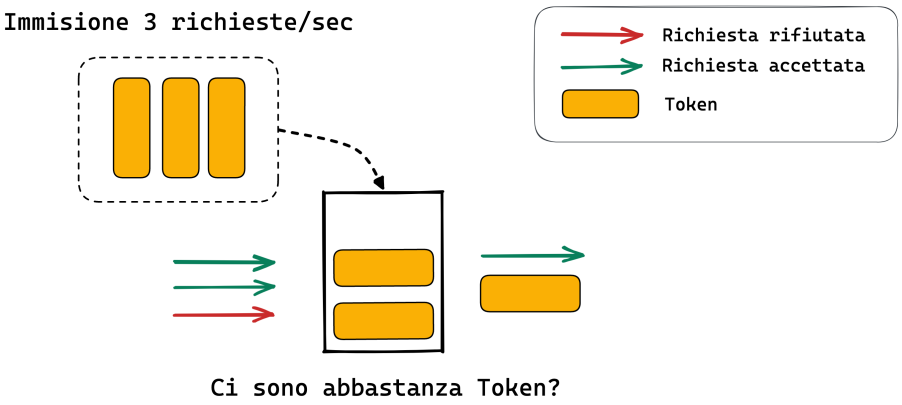

Figura 1.2: diagramma Rate-limiting token bucket

- **Leaky bucket :** L'idea di base è simile a quella del Token bucket ma invece di rispondere al numero di richieste liberando i token necessari fino a esaurimento, il tasso di elaborazione delle richieste viene regolato in modo uniforme. Questo significa che quando un pacchetto di dati arriva al sistema, se il secchio è già pieno, il pacchetto viene scartato. Nel mentre ad ogni unità di tempo viene elaborata una quantità di richieste corrispondente alla velocità di uscita del sistema. In questo modo, il flusso di dati in uscita non supera mai una certa soglia.

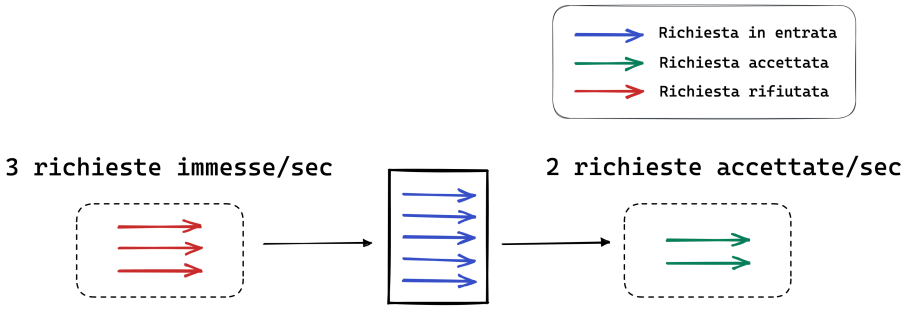

Figura 1.3: diagramma Rate-limiting leaky bucket

- **Sliding window:** La strategia di sliding window adotta un approccio simile alla Fixed Window, ma con una differenza fondamentale: esamina il tasso di richieste effettuate in un periodo di tempo continuo piuttosto che in intervalli fissi. Ad esempio, se il limite di richieste è fissato a 100 al minuto, la strategia prevede di controllare il numero di richieste effettuate nell'ultimo minuto e, nel caso superi il limite, rifiutare la richiesta. Questo approccio evita il potenziale sovraccarico del sistema alla fine di una finestra con un tempo prefissato, ma richiede la gestione di una finestra scorrevole, aumentando la complessità di implementazione.

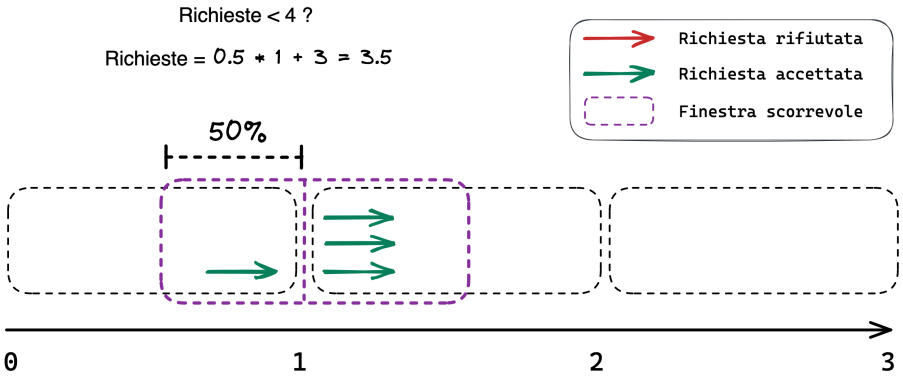

Figura 1.4: diagramma Rate-limiting sliding window

1.2 zk-SNARK

L'acronimo zk-SNARK rappresenta l'espressione completa "Zero-Knowled Succinct Non-Interactive Argument of Knowledge". Questa tecnologia è stata introdotta per la prima volta in un articolo scientifico pubblicato nel 2013 intitolato "Pinocchio: Nearly Practical Verifiable Computation". Da allora, sono state apportate diverse migliorie alla tecnologia e sono emerse numerose applicazioni in svariati settori. Le prime applicazioni significative di zk-SNARK sono state implementate nel contesto della Blockchain, che rimane il settore dove la tecnologia è più conosciuta e applicata, per fare alcuni esempi Etherium una delle Blockchain più accreditate ha iniziato ad implementare la tecnologia nella sua rete dal 2016, e il suo fondatore Vitalik Buterin ha scritto in un articolo sull'argomento: "Perhaps the most powerful cryptographic technology to come out of the last decade is general-purpose succinct zero knowledge proofs, usually called zk-SNARKs" (Forse la tecnologia crittografica più potente emersa nell'ultimo decennio è quella chiamata zk-SNARK).

Zk-SNARK, come suggerisce il nome, rappresenta una tecnologia fondata sul protocollo crittografico zero-knowledge proof, il quale consente a una parte (il dimostratore) di dimostrare a un'altra (il verificatore) la veridicità di un'affermazione senza rivelare nessuna informazione ulteriore. Inoltre questo processo di dimostrazione, viene effettuato in modo da ottenere una prova, in cui sia la dimensione della prova, che il tempo necessario per verificarla crescono molto più lentamente rispetto al calcolo da verificare e senza la necessità di interazione bidirezionale tra le due parti coinvolte.

Di seguito esamineremo meglio le singole parti che compongono la tecnologia zk-SNARK, analizzando i suoi componenti chiave per ottenere una panoramica completa e concisa.

Prima di addentrarci nella descrizione delle componenti che utilizzeremo, è importante introdurre il campo di applicazione dei teoremi e delle proprietà che andremo a utilizzare. In particolare, nei prossimi paragrafi lavoreremo in un dominio di campi finiti, che è un dominio algebrico dove l'insieme dei numeri è finito e rispetta determinate proprietà algebriche come l'addizione, la sottrazione, la moltiplicazione e la divisione. Per le loro proprietà i campi finiti svolgono un importante ruolo in diversi algoritmi crittografici.

1.2.1 Zero-Knowledge

Come accennato precedentemente, Zero-Knowledge proof è un protocollo crittografico in cui una parte dimostra di conoscere una determinata informazione a un'altra parte, senza rivelare alcuna informazione aggiuntiva su di essa. Per esempio, se Alice vuole dimostrare a Bob di conoscere la password del suo account, senza rivelargli la password stessa, può utilizzare un protocollo di tipo Zero-Knowledge. In questo modo, Alice può dimostrare a Bob di sapere qual è la password corretta senza rivelarla, proteggendo così la sua privacy e la sicurezza del suo account. Come è possibile tutto questo? grazie a diverse tecniche crittografiche e tanta matematica.

Formalmente le prove di tipo Zero-Knowledge non sono dimostrazioni di carattere

matematico, ma probabilistiche, il che significa che c'è sempre una probabilità che un dimostratore scorretto riesca a dimostrare la veridicità di un'affermazione a un verificatore onesto. Esistono tuttavia tecniche per ridurre questa probabilità a valori piccoli a piacere.

Il primo articolo che definisce il costrutto è "The Knowledge Complexity of Interactive Proof-Systems" [2] pubblicato nel 1985, dove gli autori introducono il concetto di "Zero-Knowledge proof" come un tipo di "interactive proof systems" (un modello computazionale che simula lo scabio di messagi tra due individui) in cui il verificatore non apprende nulla oltre alla verità dell'affermazione che viene dimostrata.

Per poter costruire un sistema Zero-Knowledge proof per una particolare affermazione abbiamo bisogno che la prova generata soddisfi le seguenti prorpietà :

- **Completezza :** Se l'affermazione da dimostrare è vera, allora un verificatore onesto (cioè che segue correttamente le regole del protocollo) verrà convinto della veridicità da un dimostratore onesto.
- **Correttezza :** Se l'affermazione è falsa, nessun dimostatore disonesto può convincere un verificatore onesto che essa è vera, se non con una piccola probabilità.
- **Zero-knowledge :** Se l'affermazione è vera, nessun verificatore apprende altro se non il fatto che l'affermazione è vera.

Le tre proprietà appena viste descrivono Zero-Knowledge proof da un punto di vista formale ma per avere una visione più intutiva del protocollo è utile vedere il flusso di funzionamento attraverso un esempio. L'esempio in questione è tratto da un famoso articolo di Jean-Jacques Quisquater "How to Explain Zero-Knowledge Protocols to Your Children" [3] ne estrapolerò solo le parti esenziali alla tratazione, ma ne consiglio la lettura.

L'esempio riguarda una caverna a forma di anello, nella quale è posta a metà del percoso una porta che impedisce il completamento del tragitto, a meno di non conoscere una parola segreta. Supponiamo l'esistenza di due parti, Peggy - che conosce il segreto della porta e agirà da dimostratrice - e Victor - che invece non conosce il segreto e sarà il verificatore. Peggy vuole dimostrare a Victor di sapere come superare la barriera senza però rivelare il segreto. Per fare ciò, Peggy propone a Victor di seguire una strategia: dapprima stabiliscono un nome per identificare i due percorsi (ad esempio, A e B), poi Victor rimane fuori dalla caverna mentre Peggy sceglie uno dei due percorsi. Dopo qualche minuto, Victor entra nella caverna e chiama ad alta voce il nome di uno dei due percorsi, a quel punto Peggy dovrà uscire dal percorso chiamato da Victor. Victor accetta la proposta, ma con una condizione: il processo dovrà essere ripetuto più volte. Dopo diversi tentativi, Victor si convince che Peggy conosca effettivamente il segreto per superare la barriera. Possiamo calcolare questa probabilità utilizzando la distribuzione binomiale, dove la probabilità di successo p è del 50% e il numero totale di prove n è 10. La probabilità di indovinare correttamente tutte le 10 scelte di Victor è data dalla seguente formula:

$$P = p^n = 0,5^{10} = 0,0009765625$$

2 | Analisi progettuale

In questo progetto si cerca di sviluppare un sistema per verificare in modo robusto la similarità tra password di un utente, in modo da evitare di utilizzare varianti di password precedenti, e garantire maggiore sicurezza di uno o più account. A tal proposito sono stati condotti esperimenti che, per controllare la similarità tra password, utilizzavano word embedding di parole. In questo lavoro di tesi è stato preso come riferimento il paper di Bijeeta et alii [**bijeeta**].

2.1 Modelli di similarità tra password basati sulle reti neurali

Solitamente le metriche che misurano la robustezza di password non tengono conto della cronologia delle password passate di un utente. Ciò può costituire un problema sia se si effettua un attacco contro un utente, sia per proteggerlo:

- Durante un attacco, si considerano determinate password note grazie a leak, tuttavia non è presente un approccio flessibile che elabori varianti di password di un determinato utente.
- Non è presente un meccanismo che avverta l'utente del potenziale pericolo causato dal riutilizzo di password.

Per questo motivo, nel paper riferimento di Bijeeta et alii [**bijeeta**] sono stati utilizzate due tipologie di reti neurali:

- Per l'attacco è stato sviluppato un modello di rete neurale ricorrente, noto come pass2path.
- Per la difesa sono stati utilizzate tecniche di Natural Language Processing, in particolare modelli di word embedding, i quali generano una corrispondenza tra parole e vettori, mantenendo proprietà semantiche della password originale, come la similarità.

2.2 Prerequisiti

Per la creazione dei due modelli visti precedentemente, è stato utilizzato un leak disponibile sul Deep Web, di dimensione pari a 45 GB e contenente 1.4 miliardi di

coppie mail-password appartenenti ad account su social come LinkedIn, MySpace, Badoo, Yahoo, Zoosk; successivamente è stato filtrato nel seguente modo:

- sono state rimosse le stringhe che contenevano 20 o più caratteri esadecimali;
- sono stati rimossi hash non decodificati;
- sono state rimosse le password più lunghe di 50 caratteri o più corte di 3 che contenevano caratteri non ASCII;
- sono stati rimossi 4528 utenti associati a centinaia di password, poiché è molto improbabile che siano account veri.

Dal risultato del processo di filtraggio sono stati osservati i seguenti punti:

- la password 123456 è stata utilizzata dal 0.9% degli utenti;
- più dell'88% di password hanno una lunghezza compresa tra 6 e 12 caratteri;
- l'80% delle password contiene solo caratteri minuscoli.

Property	Values	% of PWs
Length	$3-5$	2
	$6-8$	48
	$9-12$	40
	$13-50$	10
Composition	Lower case only	80
	Upper case only	3
	Letters only	38
	Digits only	8
	Special characters only	< 0.1
	Letters & digits only	55
	Containing at least one letter, one digit and one special char	5

Figura 2.1: La distribuzione della lunghezza delle password e della loro composizione dopo la pulizia del dataset [**bijeeta**]

2.3 Strategie di attacco: pass2path

2.3.1 Euristiche e appartentenza di password

Una volta ottenuto il dataset, è necessario capire a quali persone appartengono gli account. A tal proposito vengono proposte tre euristiche:

- **Basata su email:** gli utenti vengono identificati solo dalla email. Ciascuna email appartiene a un solo utente.

- **Basata sugli username:** si considera la stringa che precede `@` nell'indirizzo email. Un utente può possedere più mail, tutte identificate dalla stringa che precede `@`.
- **Basata su un metodo misto:** considera sia gli username che le email. Due email sono considerate connesse tra loro se hanno almeno una password in comune e se le mail connesse tra loro sono associate allo stesso username.

Per potere effettuare migliori test sul dataset sono stati rimossi gli utenti che avevano meno di due password associate allo stesso account.

Il dataset successivamente viene diviso in due parti: training set e test set (in rapporto 80%-20%). Nel paper la fase di training utilizza soltanto il dataset relativo all'euristica basata sulle email, mentre per la fase di testing si utilizza sia l'euristica basata sulle email sia quella mista.

2.3.2 Introduzione a pass2path

Pass2path è un modello di rete neurale che consente di creare password in grado di compromettere più del 48% degli utenti in meno di mille tentativi. Il successo di questo modello è dovuto al riutilizzo della stessa password o di varianti di password da parte dell'utente. Per svilupparlo si è tenuto conto della sequenza di trasformazioni $\tau_1 ... \tau_n$, nota come *percorso* che consente, a partire dalla password $\tilde{w}$, di produrre la nuova password w.

Le modifiche sono rappresentate da una unità di misura $\tau \in \mathcal{T}$, che specifica in che posizione e quale tipo di variazione applicare a una password.
τ è composto da una tripletta $\{e, c', l\}$:

- e rappresenta una modifica da apportare;
- c' rappresenta un carattere o una stringa vuota;
- l rappresenta la posizione della modifica della password.

Le modifiche vengono classificate in tre tipologie:

- `sub` (sostituzione);
- `ins` (inserimento);
- `del` (cancellazione).

Se si considera c' sono presenti due situazioni:

- `ins` e `sub`: c' rappresenta il carattere o la stringa vuota;
- `del`: è sempre una stringa vuota.

Per ricavare il percorso tra due password, si considera quello più corto, nel seguente ordine decrescente di preferenza:

1. `del`;
2. `ins`;
3. `sub`.

Le varie trasformazioni del percorso vengono ordinate in base alla posizione della modifica.

Ad esempio, il percorso da `cats` a `kates` (distanza di modifica pari a 2) è:

$$path = \{(sub, k', 0), (ins, e', 3)\}$$

2.3.3 Come allenare pass2path

Prima di allenare pass2path è necessario tradurre ciascuna password come sequenza di tasti premuti su una *tastiera americana ANSI*. La codifica dei tasti premuti è la seguente:

- Per rappresentare una sola lettera maiuscola all'interno di una parola si pone `<s>` (che rappresenta la pressione del tasto `SHIFT`) prima della lettera, che viene lasciata minuscola. Ad esempio `Ciao` viene tradotto come `<s>ciao`.
- se si hanno più lettere maiuscole consecutive seguite da lettere minuscole si pone `<c>` (che rappresenta la pressione del tasto `CAPS LOCK`) prima e dopo la sequenza di lettere maiuscole, che viene lasciata minuscola. Per esempio `PASSword` viene tradotto come `<c>pass<c>word`.
- Nel caso di una sequenza di lettere maiuscole che si conclude alla fine della parola basta porre `<c>` all'inizio della sequenza. Per esempio `PASSWORD` viene tradotto come `<c>password` e `passWORD` come `pass<c>word`.
- Se si hanno caratteri speciali ASCII 128 si pone `<s>` davanti al carattere. Quest'ultimo viene tradotto come il tasto che viene premuto insieme a `SHIFT`. Per esempio `PASSWORD!` viene rappresentata come `<c>password<s>1`, dato che `1` viene premuto insieme a `SHIFT` e `Hello@!!` viene tradotto come `<s>hello<s>2<s>1<s>1`.

Successivamente occorre trovare il percorso di transizioni $\tau_1 ... \tau_n$ che consentano di trasformare la password $\tilde{w}$ in w.

Si è deciso di filtrare le password in base alla lunghezza del percorso, che deve tenere conto anche della modifica di una password basata sulla sequenza di tasti premuti. Sono state eliminate le password con un percorso di lunghezza superiore a δ.

Per allenare pass2path si utilizza il dataset di training basato sulle email. Si utilizzano due reti neurali ricorrenti (RNN) per costruire un auto-encoder [**Sherstinsky_2020**].

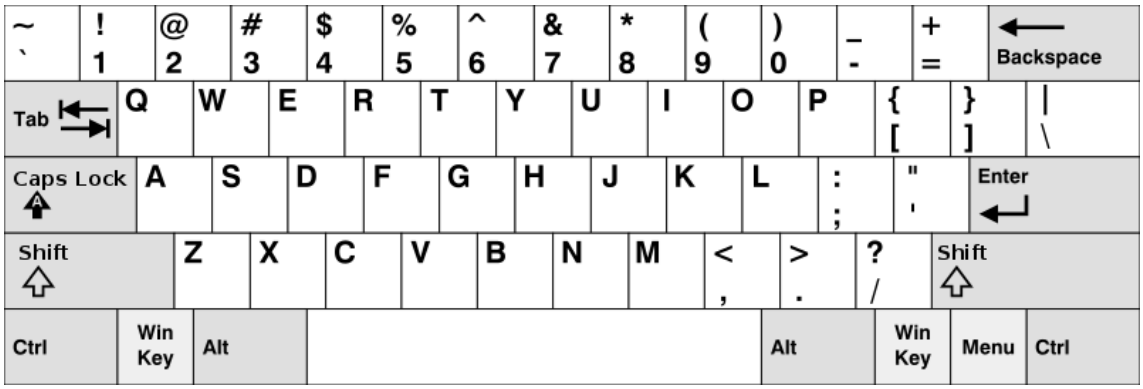

Figura 2.2: Layout della tastiera americana ANSI, fonte: Wikipedia

2.3.4 Efficacia d'attacco con configurazioni non ripetute

Per verificare l'efficacia di attacco della rete si utilizza il dataset di test delle email, con password da indovinare w diverse dalla originale $\tilde{w}$. Con configurazioni non ripetute, pass2path riesce in meno di 100 stime a ricavare il 13% delle password, impiegando 4 ore in tutto (Intel i9 e 128 GB di RAM, su un singolo thread).

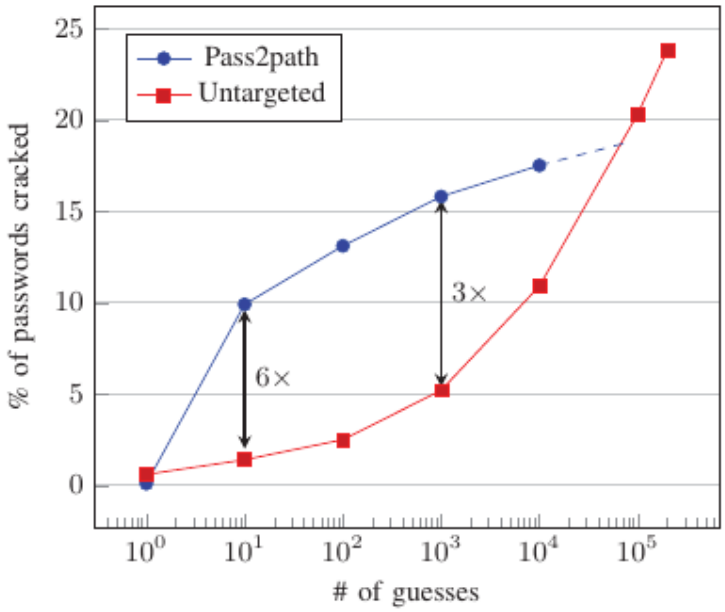

Figura 2.3: Pass2path riesce a indovinare in meno di mille tentativi una percentuale più alta di password, rispetto ad attacchi non mirati. Questi ultimi risultano efficaci soltanto se sono richiesti più di 1000 tentativi [**bijeeta**]

2.3.5 Efficacia d'attacco con configurazioni ripetute

Per verificare l'efficacia di un attacco della rete, in caso di password da indovinare w simile alla originale $\tilde{w}$, si utilizza il test set del dataset misto, in cui si è osservato

che il 40% delle password vengono riutilizzate dagli utenti, rendendoli facili bersagli.

Come primo attacco viene utilizzata la password $\tilde{w}$, in modo da verificare il suo riutilizzo senza varianti, mentre i restanti $q-1$ vengono svolti in accordo alla tecnica di attacco scelta. Confrontando con altri modelli utilizzati, il migliore risultato è stato ottenuto da pass2path, compromettendo metà degli utenti (48.3%) in al massimo mille tentativi. Si è osservato che con password riutilizzate, aumentano le probabilità di indovinare la password con pass2path. Inoltre è importante sottolineare che gli attacchi di tipo non mirato (untargeted) ottengono migliori performance se vengono eseguiti più di 1000 tentativi; al contrario pass2path è il migliore approccio nel caso in cui si proceda con un numero minore.

I ricercatori hanno anche eseguito un vero e proprio attacco all'interno della loro università (Cornell University), in modo da potere testare su password diverse da quelle del dataset, ovvero appartenenti a studenti e professori. Il migliore risultato è stato ottenuto da pass2path, che in meno di 1000 tentativi è riuscito a scoprire la password del 8,4% degli account [**bijeeta**].

2.4 Difesa

2.4.1 Difesa da attacchi a dizionario mirati

Diversi studi mostrano che cambiare password non protegge completamente un utente dagli attacchi [**google**] [**hypr**]. Nel paper viene illustrato il concetto di PPSM (Personalized password strength meters), che sfruttano modelli preallenati di word embedding per dimostrare la sicurezza di una password. In questo modo si possono prevenire attacchi che sfruttano varianti di password presenti in data breach.

2.4.2 Personalized password strength meters

Un PPSM può essere utilizzato con lo scopo di dare un giudizio in tempo reale all'utente sulla sicurezza delle password durante la selezione. Il funzionamento è il seguente:

- vengono considerati in input una potenziale password e un set di password associate all'utente trovate in un data leak;
- vengono utilizzati due possibili criteri come output:
  - **guess rank**, ovvero il numero di tentativi di una tipologia di attacco fatti prima di indovinare la password;
  - **percentuale di similarità**, ovvero la somiglianza tra la potenziale password e la password associata all'utente.

Un possibile modo di ottenere il guess rank è basarsi su pass2path, in modo da evitare che un utente usi una password simile a quella trovata in un data breach; tuttavia prevedere password risulta costoso (dato che si considera un modello di

generazione di password come pass2path) e, nel caso in cui si vogliano inviare i risultati via rete, viene occupata molta banda.

Si è osservato che risulta più efficiente e meno costoso assegnare punteggi che rappresentano la sicurezza di una password rispetto ai guess rank [**bijeeta**].

2.4.3 Realizzazione di un PPSM

Sotto al PPSM si trova un classificatore binario C che prende in input una password candidata w e una password nota da un leak $\tilde{w}$ e restituisce 0 se w è indovinabile in meno di 1000 tentativi a partire da $\tilde{w}$, 1 altrimenti. Per costruire tale classificatore è necessario utilizzare tecniche di word-embedding.

2.4.4 Similarità tra password via word embedding

Si definisce word embedding la tecnica di mappatura di un insieme di parole in uno spazio vettoriale d-dimensionale (solitamente con d che vale 100 o 200 o 300).

In questo modo vengono preservate le proprietà semantiche delle parole, come ad esempio la loro similarità: ad esempio, se due parole compaiono spesso nello stesso contesto, le loro rappresentazioni vettoriali avranno una distanza ridotta.

In particolare, nel problema in esame, la tecnica di word embedding viene applicata alle password: due password risultano simili se vengono scelte spesso dallo stesso utente. Ciò permette di stabilire quanto una password sia sicura, considerando tutte le password precedentemente scelte dall'utente, e di fornire un punteggio (ovvero la percentuale di similarità).

A tal scopo, per costruire un password embedding model, viene utilizzato FastText, che impara la similarità dividendo la parola in una collezione di contesti, definiti come piccole sequenze di parole note come *n-gram*. Le parole che appaiono spesso insieme nello stesso contesto vengono definite simili.

2.4.5 Allenamento di Fasttext

Per l'allenamento del modello di FastText vengono considerati i seguenti parametri:

- **Dimensione del vettore**: impostata a 100, poiché l'allenamento del modello così risulta più rapido rispetto al parametro di default a 300.
- **Subsampling**: ignora le password che ricorrono più frequentemente. Impostato a 10^{-3}, poiché non si vogliono password con più di 1000 occorrenze.
- **Dimensione minima degli ngram**: impostata a 1, in modo che possano essere costruiti embedding per password mai viste durante l'allenamento.
- **Dimensione massima degli ngram**: impostata a 4, dato che le parole presenti nel dataset hanno come minimo 4 caratteri.

2.4.6 Classificazione delle password

Per classificare le password viene utilizzato un classificatore binario che restituisce 0 se le password sono simili tra loro, superando una soglia α di similarità decisa prima della misurazione, 1 altrimenti.

Una password risulta vulnerabile se, data una password w, essa viene indovinata in meno di 1000 tentativi a partire dalla password $\tilde{w}$, utilizzando come euristica pass2Path. Per determinare la soglia α, vengono considerati 10^5 utenti dal dataset di test delle email e per ciascun utente vengono sorteggiate due password dalla collezione di password associati a essi. Una delle due password (la scelta della password è arbitraria) viene considerata come la nuova password da indovinare w, mentre l'altra password $\tilde{w}$ rappresenta la password trovata in un leak.

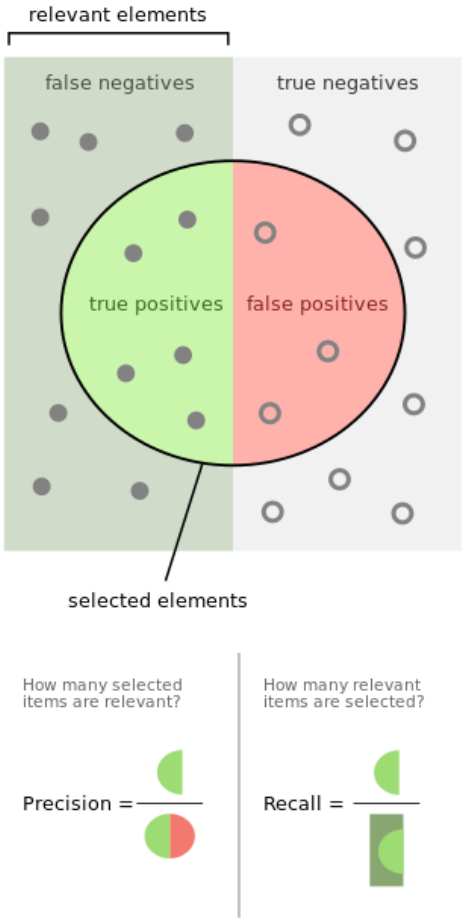

Figura 2.4: Definizione grafica di precision e recall, fonte: Wikipedia

Vengono definiti due parametri per la scelta di α:

- **Precision**: rappresenta quanti veri positivi sono stati rilevati su un totale composto da veri positivi e falsi positivi.
- **Recall**: rappresenta il numero effettivo di elementi positivi che sono stati rilevati su un totale di falsi negativi e veri positivi.

In particolare, nel caso in esame:

- per vero positivo si intende una coppia di password simili correttamente rilevate come tali;
- per falso positivo si intende una coppia di password diverse erroneamente rilevate come simili;
- per falso negativo si intende una coppia di password simili erroneamente non rilevate come tali.

Un valore di precision basso implica una imprecisa distinzione tra password simili e password non simili; un valore di recall basso invece comporta avere molte password simili non rilevate come tali.

Nel paper di riferimento di Bijeeta et alii [**bijeeta**] viene considerato un valore di recall molto alto (99%) e una percentuale di precision nettamente più bassa (60%). Per questo motivo α è stato posto a 0.5.

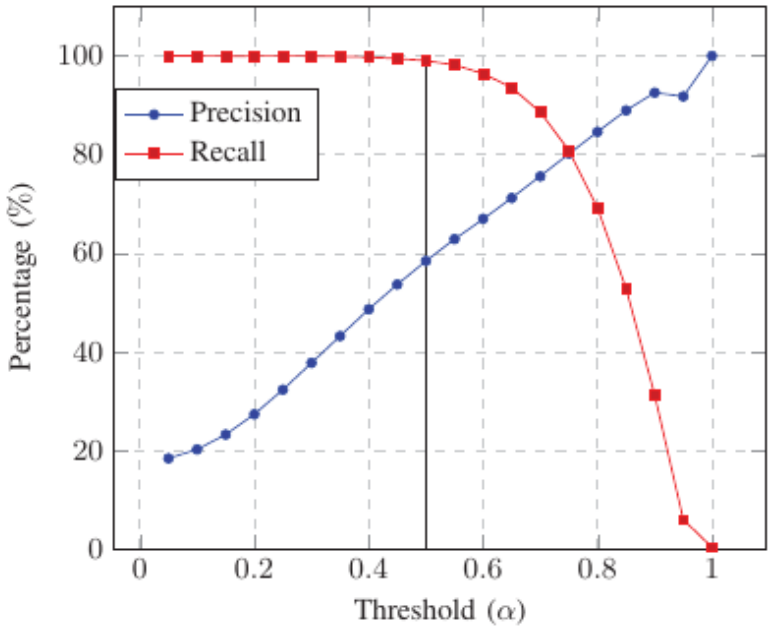

Figura 2.5: Precision e recall di Bijeeta et al. [**bijeeta**]

2.4.7 Modelli compressi di word embedding

Il modello prodotto da Bijeeta et alii [**bijeeta**] è stato successivamente compresso, in modo da ridurre la dimensione da 1.5 GB a 3 MB, senza che la qualità delle performance venisse ridotta, mediante tecniche di quantizzazione. Per la produzione del modello compresso si è tenuto conto del parametro η che determina il rapporto di compressione. Per la valutazione della compressione, sono stati prodotti più modelli in base a differenti valori η; infine è stato scelto il modello con $\eta = 5$ e dimensione complessiva di 3 MB, poiché il valore di recall non ha subito notevoli variazioni.

η	**Size (MB)**	**Precision** (%)	**Recall** (%)
100	50.0	59.1	99.3
10	5.3	48.5	99.0
5	3.0	41.3	98.6

Figura 2.6: Precision e recall in base al rapporto di compressione η del modello di Bijeeta et alii [**bijeeta**]

2.5 Modifiche e analisi del progetto

Per questo progetto sono state effettuate diverse scelte:

- Si è scelto di non implementare pass2path, per motivi di complessità e di costo in termini di testing;
- sono stati modificati i criteri di filtraggio del dataset per allenare il modello di word embedding;
- sono state utilizzate euristiche diverse rispetto a pass2path;
- sono stati modificati i parametri per allenare la rete neurale che ricava la similarità tra password.

Bibliografia

Bibliografia

[1] *Google blocked the largest Layer 7 DDoS attack*. 2022. URL: https://cloud.google.com/blog/products/identity-security/how-google-cloud-blocked-largest-layer-7-ddos-attack-at-46-million-rps.

[2] S Goldwasser, S Micali e C Rackoff. «The Knowledge Complexity of Interactive Proof-Systems». In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. STOC '85. Providence, Rhode Island, USA: Association for Computing Machinery, 1985, 291–304. ISBN: 0897911512. DOI: 10.1145/22145.22178. URL: https://doi.org/10.1145/22145.22178.

[3] Jean-Jacques Quisquater et al. «How to Explain Zero-Knowledge Protocols to Your Children». In: *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. Vol. 435. Lecture Notes in Computer Science. Springer, 1989, pp. 628–631. DOI: 10.1007/0-387-34805-0_60.