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, 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.

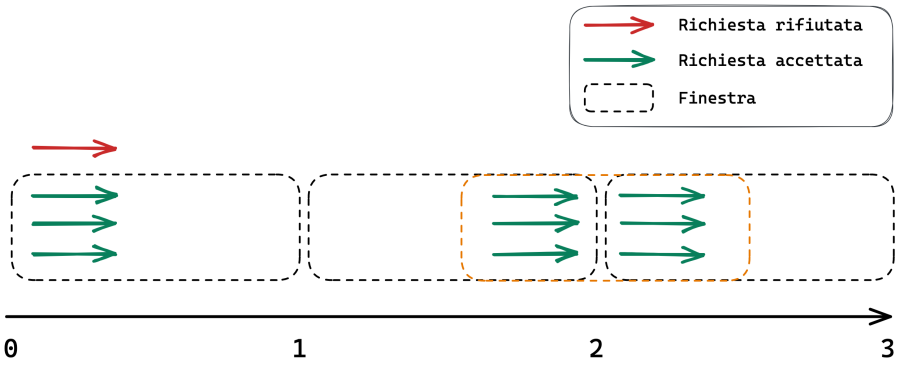

Figura 1.1: diagramma Rate-limiting fixed window

- **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.

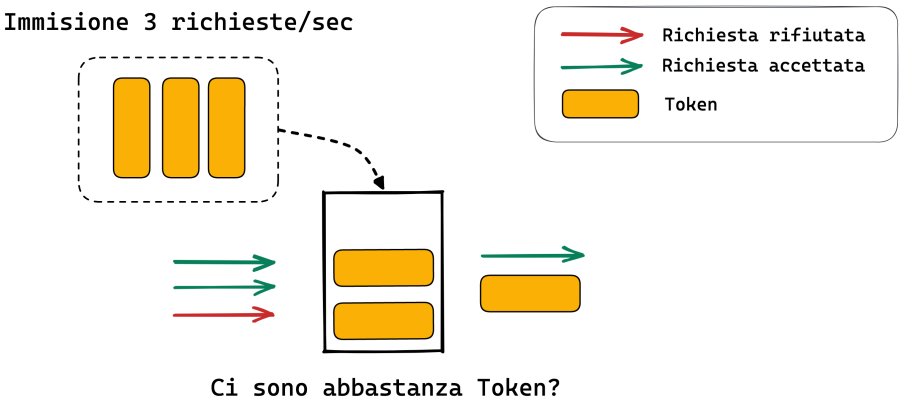

Figura 1.2: diagramma Rate-limiting token bucket

- **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.

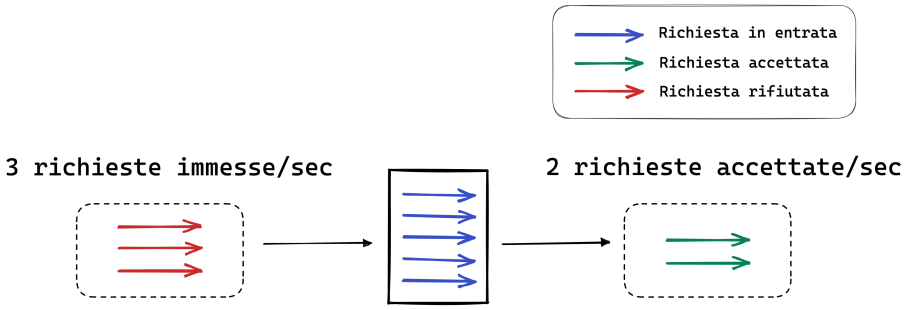

Figura 1.3: diagramma Rate-limiting leaky bucket

- **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.

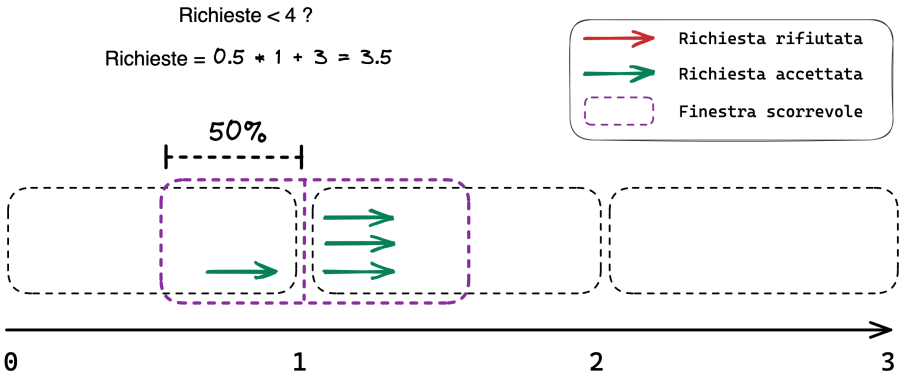

Figura 1.4: diagramma Rate-limiting sliding window

1.2 zk-SNARK

Bibliografia