UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Studio, progettazione e implementazione di un algoritmo genetico per l'individuazione di problemi di privacy in sistemi IoT

RELATORE

Prof. **Fabio Palomba**

Università degli Studi di Salerno

CANDIDATO

Davide La Gamba

Matricola: 0512106292

Anno Accademico 2021-2022

Alla mia famiglia

Sommario

Al giorno d'oggi miliardi di dispositivi IoT (Internet-of-Things) comunicano costantemente scambiandosi enormi quantità di dati, tra cui informazioni personali degli utenti che ne fanno quotidianamente uso. Per tal motivo, risulta critico porre l'attenzione sulle problematiche relative alla sicurezza e soprattutto alla privacy connesse a questi tipi di dispositivi. Il lavoro proposto, oltre a fornire una panoramica sul contesto applicativo, in particolare sul mondo IoT e i principali attacchi a cui tali dispositivi sono soggetti, si sofferma anche sull'impiego dell'Intelligenza Artificiale (AI) riguardo queste minacce, analizzando le tecniche più utilizzate. Alla luce di ciò, gli algoritmi genetici risultano essere tra le tipologie di soluzioni meno approfondite per contrastare problemi di sicurezza e privacy nel mondo IoT. Per questo motivo, vengono presentate in dettaglio le fasi di progettazione, implementazione e validazione dei risultati di un algoritmo genetico evolutivo che mira a creare un insieme di regole di classificazione per identificare gli attacchi presenti nel dataset KDDCUP99, che raccoglie informazioni su connessioni avvenute tra dispositivi appartenenti ad una rete di stampo militare. L'algoritmo sviluppato utilizza come linea guida un lavoro già esistente di Al-fuhaidi et al. per poi estenderne le capacità di individuazione e migliorarne il tasso di rilevamento modificando fortemente molti aspetti della progettazione, fino ad ottenere prestazioni simili ai migliori classificatori più comuni, come Random Forest.

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

Negli ultimi decenni abbiamo assistito ad una rapida crescita nell'utilizzo di dispositivi intelligenti in ogni ambito delle nostre vite, partendo da oggetti come veicoli e macchinari, fino ad arrivare a smartphone ed elettrodomestici costituiti da software, sensori e attuatori. Dalla comunicazione di questi tramite la rete nasce l'***Internet-of-Things*** (IoT) [4].

1.1 Motivazioni e Obiettivi

Viviamo costantemente immersi in oggetti intelligenti fondamentali per il prosieguo della nostra vita, difatti nel 2021 il numero di dispositivi IoT connessi è stato stimato essere maggiore di 10 miliardi, e si prevede possano superare i 25.4 miliardi entro il 2030, mentre l'ammontare di dati generati da questi si aspetta toccherà i 73.1 ZB (1 zettabyte = 10^9 terabyte) nel 2025 [5]. Difatti molti dispositivi IoT acquisiscono continuamente informazioni dall'ambiente esterno tramite i propri sensori per generare dati da elaborare al fine di fornire servizi. Tra queste informazioni risiedono ovviamente anche dati sensibili e privati, basti pensare al caso degli assistenti intelligenti che sono presenti nelle abitazioni. Per tal motivo, risulta critico porre attenzione alla **sicurezza** e alla **privacy** nei confronti dell'IoT, e in particolare alle minacce e agli attacchi cui sono soggetti questi dispositivi. Sono infatti numerosi gli studi e i diversi approcci verso le soluzioni dei problemi di privacy; in particolare, si può denotare un notevole interesse nell'applicazione dell'**intelligenza artificiale**. Grazie al lavoro di Giordano et al. [6] è stato possibile analizzare la poca attenzione riservata all'utilizzo degli **algoritmi genetici**, di cui è possibile fornire una breve descrizione grazie alla fonte [7]. Questi sono dei particolari algoritmi euristici, ispirati alla teoria dell'evoluzione di Darwin, impiegati per trovare soluzioni a problemi di ottimizzazione. Il modo in cui ciò avviene è tramite la codifica di potenziali soluzioni ad uno specifico problema tramite l'utilizzo di strutture dati simili a cromosomi e l'applicazione di operatori di ricombinazione [7]. L' implementazione di un algoritmo genetico inizia con la generazione di una **popolazione** casuale di **individui**, ognuno valutato tramite una **funzione di valutazione**, che spesso coincide con la **funzione obiettivo** del problema in esame da minimizzare o massimizzare. Ogni iterazione dell'algoritmo consiste nella creazione di una nuova generazione di individui, difatti questi sono sottoposti ad un processo di **selezione**. Il calcolo del grado di sopravvivenza degli individui deriva dalla **funzione di fitness** oltre che dal criterio di selezione scelto. Lo scopo dell'algoritmo è quello di far **evolvere** iterativamente le soluzioni rispetto a questa funzione, fino al raggiungimento di una soluzione soddisfacente o di altri criteri di arresto. Gli operatori che guidano la generazione di nuovi individui, oltre quello di selezione, sono il **crossover** e la

mutazione. Tramite il crossover, o ricombinazione, sono generati gli individui figli a partire dai geni dei genitori. Questi poi saranno presenti nella generazione successiva, rimpiazzando o meno gli individui da cui sono stati generati [7]. L'operazione di mutazione simula quella che realmente avviene negli organismi, andando a modificare dei geni di un nuovo individuo per poter esplorare porzioni maggiori dello spazio delle soluzioni. Le tecniche di crossover e mutazione applicabili dipendono strettamente dalle scelte di codifica degli individui. L'interesse verso l'adozione di algoritmi genetici per l'analisi e la prevenzione di problemi di privacy deriva dalle caratteristiche che li contraddistinguono. Questi sono sicuramente molto flessibili, in quanto permettono di produrre soluzioni per una vasta gamma di problemi anche molto diversi tra loro, e garantiscono risultati accettabili in breve tempo. D'altro canto, spesso risulta difficile codificare le caratteristiche e i parametri delle soluzioni negli individui del problema, e da questi poi estrarre le migliori tecniche di selezione, crossover e mutazione. Per i motivi elencati, l'obiettivo di questo studio è di analizzare l'attuale stato dell'arte per quanto riguarda i problemi di privacy e sicurezza dei sistemi IoT, le soluzioni generalmente adottate e l'impiego degli algoritmi genetici. In seguito verrà presentata la progettazione e l'implementazione di un algoritmo genetico per la classificazione di minacce in un sistema IoT che potrebbero portare a problemi di sicurezza e di privacy, confrontandone i risultati con le altre soluzioni proposte in letteratura, in particolare con il lavoro di Al-fuhaidi et al. [3].

1.2 Risultati

In seguito alle fasi di progettazione e codifica dell'algoritmo, le prestazioni ottenute sono risultate simili a quelle derivanti da alcuni tra gli algoritmi di classificazione più noti, come il Random Forest, presentando come punto di forza una facile comprensibilità dei criteri utilizzati per l'individuazione delle connessioni pericolose. I risultati ottenuti rispetto al lavoro di Al-fuhaidi et al. [3] sono molto simili, tuttavia l'algoritmo proposto in questo lavoro presenta una più vasta capacità di individuazione degli attacchi, derivante da una diversa fase di progettazione.

1.3 Struttura della tesi

Il Capitolo 2 offre una panoramica generale sul background degli argomenti trattati da questo studio, nello specifico l'Internet-of-Things. In particolare nella Sezione 2.1 viene fornita una descrizione dei sistemi IoT, delle loro architetture, i principali problemi di privacy e

l'utilizzo dell'intelligenza artificiale, con un approfondimento nella Sezione 2.2 sui sistemi IoT basati su blockchain. Nella Sezione 2.3 sono illustrate le principali vulnerabilità di questi tipi di sistemi, descrivendone alcuni nel dettaglio tramite l'esposizione di alcuni tentativi di attacchi, con lo scopo di capire le vulnerabilità di questi tipi di dispositivi per poi migliorarne la sicurezza. Il Capitolo 3 offre invece una panoramica sullo stato dell'arte riguardante l'impiego dell'intelligenza artificiale, in particolare degli algoritmi genetici, per problemi di privacy e sicurezza. Nello specifico sono state descritte tecniche per l'individuazione di vulnerabilità e di intrusioni, evoluzione di malware, feature selection ed applicazioni in sistemi blockchain. Infine, viene analizzato nel dettaglio il lavoro di Al-fuhaidi et al. [3], impiegato poi come linea guida nella realizzazione dell'algoritmo genetico. Il Capitolo 4 espone le fasi di progettazione dell'algoritmo proposto, come la scelta di diversi aspetti: l'obiettivo, la codifica, la suddivisione del dataset e gli strumenti da utilizzare, le codifiche delle componenti principali dell'algoritmo come gli individui, la funzione di fitness, la fase di validazione e i parametri utilizzati. Infine viene mostrata una panoramica generale sulla strategia adottata e un confronto con il lavoro di Al-fuhaidi et al. [3]. Nel Capitolo 5 sono riportati i risultati ottenuti dalle diverse versioni dell'algoritmo implementato, confrontandole con le prestazioni dei classificatori tradizionali maggiormente utilizzati e con classificatori casuali. Infine, il Capitolo 6 presenta discussioni sui risultati ottenuti e sui lati positivi e negativi riscontrati nella strategia proposta, proseguendo con delle conclusioni derivanti dai confronti tra le soluzioni ottenute, il lavoro di Al-fuhaidi et al. [3] e i classificatori casuali, terminando con delle conclusioni finali e proposte di sviluppi futuri.

In aggiunta a quanto mostrato in questo lavoro, l'intero codice utilizzato per la realizzazione dell'algoritmo è pubblicamente disponibile in un repository GitHub, al seguente link: `https://github.com/davide-lagamba/GAforPrivacy`.

Al medesimo link è disponibile anche la lista degli strumenti utilizzati come supporto per la realizzazione di diagrammi, figure, grafici e tabelle.

CAPITOLO 2

Background

Per poter progettare una tecnica mirata a contrastare problemi di privacy e di sicurezza in un ambiente IoT tramite un algoritmo genetico, risulta fondamentale analizzare e presentare le caratteristiche principali dei dispositivi IoT, discutendo anche degli impieghi dell'intelligenza artificiale nei confronti delle problematiche ad essi legati.

2.1 Panoramica su Sistemi IoT

2.1.1 Architetture maggiormente utilizzate

Come brevemente illustrato dalla fonte [6], attualmente le tre principali architetture utilizzate nella realizzazione di sistemi IoT sono le architetture three-layer, four-layer e five-layer.

L'architettura three-layer è la più semplice, composta da un Perception Layer, responsabile dell'acquisizione di informazioni dell'ambiente circostante tramite sensori, Network Layer, che si occupa della comunicazione dei dati prodotti dal Perception Layer tramite tecnologie come Wi-Fi e Bluetooth, e Application Layer, dedicato all' interazione con l'utente.

L'architettura four-layer aggiunge un livello denominato Service and Application Support tra l'Application e il Network Layer, per agevolare la comunicazione tra questi.

In seguito, l'architettura five-layer ha introdotto il Transport Layer, il Processing Layer e il Business Layer. Il primo riceve i dati dal Perception Layer e li analizza, fornendo i servizi di connessione alla rete, autenticazione, controllo degli accessi e inoltro delle informazioni ai livelli sovrastanti. Il secondo utilizza infrastrutture cloud per la gestione dei dati ricevuti, effettuando analisi sugli stessi e provocando azioni che sono anche in grado di modificare il contesto esterno. Il terzo invece viene utilizzato per prendere decisioni in base allo stato dell'ambiente. Risulta essere il livello maggiormente impiegato nella preservazione della privacy dell'utente.

Rappresentazioni sommarie delle varie architetture sono riportate nelle Figure 2.1, 2.2 e 2.3.

2.1.2 Problemi di privacy

La medesima fonte [6] sottolinea come, secondo il GDPR[1], il concetto di privacy è composto da Data Prevention, ovvero il mantenimento della sicurezza dei dati da parte di accessi

[1] `https://gdpr.eu/data-privacy/`

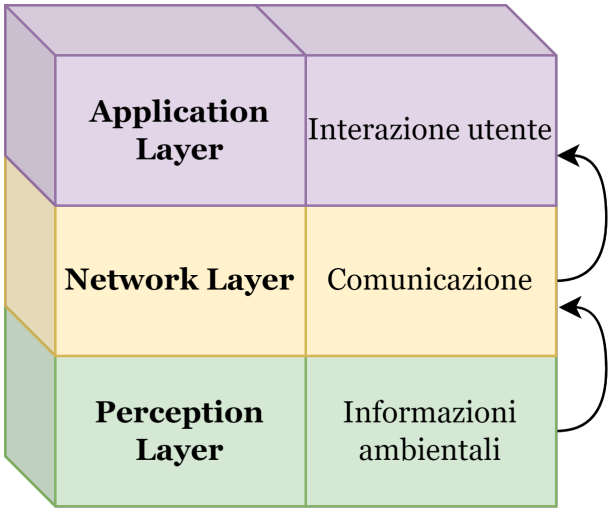


Figura 2.1: Architettura a tre livelli

estranei e non autorizzati, e Privacy Data, che consiste nell'informare l'utente della gestione delle sue informazioni sensibili e private.

Inoltre, come riportato da Giordano et al. [6], le tecniche che mirano ad occultare dati riservati ricadono nella categoria denominata *"Data Perturbation"*, mentre quelle che cercano di impedire gli accessi non consentiti appartengono alla *"Data Restriction"*.

Vengono di seguito riportate le principali minacce per la privacy negli ambienti IoT esposte da Giordano et al. [6]:

- **Identificazione** : Identificazione di un dispositivo attraverso uno dei suoi indirizzi.
- **Tracciamento** : Rilevazione del traffico degli utenti.
- **Profilazione** : Raccolta delle informazioni legate agli utenti.
- **Interazione** : Minacce per la privacy scaturite da interazioni tra più macchine in sistemi IoT.
- **Transizioni del ciclo di vita** : Possibili pericoli per la privacy durante transizioni di stati nel ciclo di vita di un dispositivo, a causa di problemi di memorizzazione.
- **Inventory attacks** : Raccolta non autorizzata di informazioni circa le caratteristiche dei dispositivi personali a seguito di accessi malintenzionati.
- **Linkage** : Collegare dispositivi separati in modo da ottenere informazioni non rivelate direttamente, attraverso la combinazione di più fonti di dati.

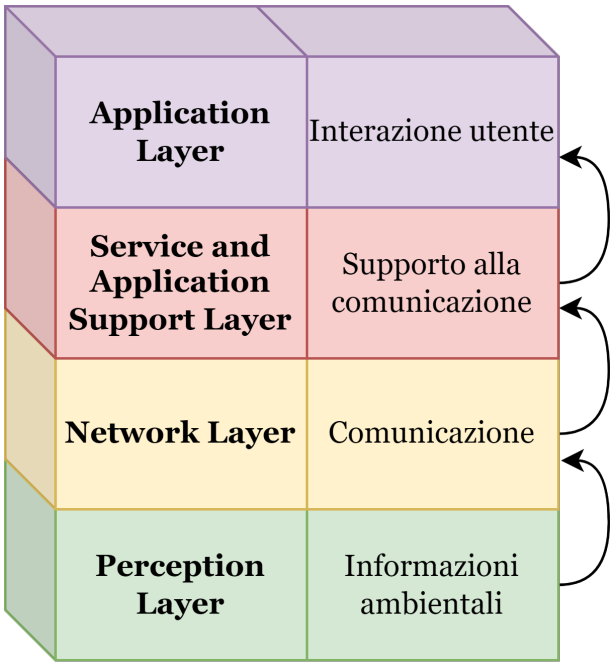


Figura 2.2: Architettura a quattro livelli

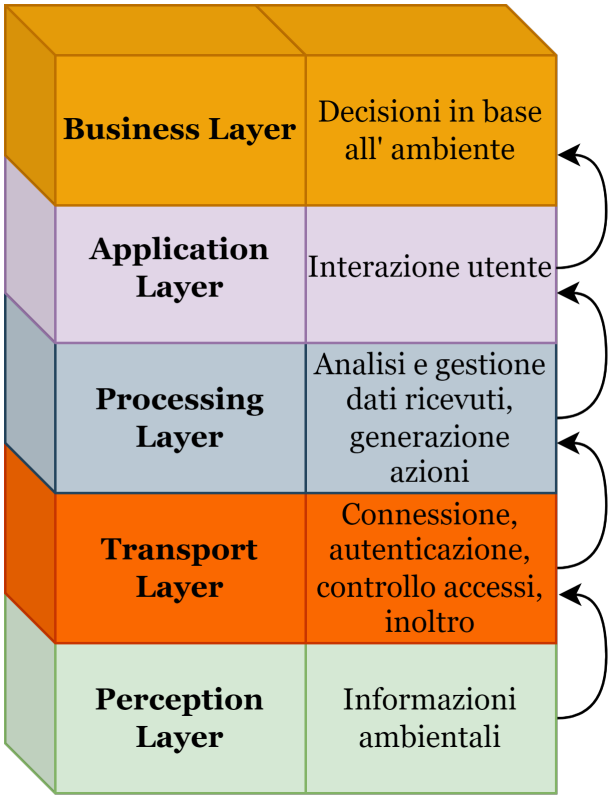


Figura 2.3: Architettura a cinque livelli

L'interesse relativo all'impiego dell'intelligenza artificiale nell'ambito di queste problematiche è in continuo aumento, come testimoniano le statistiche riguardanti le pubblicazioni in letteratura prodotte dalla fonte [6]. I principali obiettivi posti nell'utilizzo di tali tecniche, raccolti da Giordano et al. [6], sono stati:

- **Analisi della rete** : Ricercare la presenza di traffico generato da attività sospette nella rete, principalmente collegate al mancato utilizzo di protocolli sicuri nelle comunicazioni tra sistemi del tipo in analisi. È tipicamente trattato come un problema di classificazione.
- **Individuazione di attacchi** : Ne fanno parte *"Intrusion Detection"*, ovvero la creazione di programmi per l'individuazione di attacchi su dispositivi IoT tramite il confronto del traffico nella rete con attacchi informatici ben conosciuti, e *"Anomaly Detection"*, che invece si concentra sull'identificare le minacce dal traffico prodotto tramite l'utilizzo di modelli matematici, in particolare grazie alla statistica. Anche questo problema è stato trattato principalmente con algoritmi di classificazione.
- **Autenticazione** : Definizione di nuovi meccanismi di autenticazione più sicuri di quelli comunemente adottati, utilizzando generalmente sensori biometrici, allo scopo di produrre identificativi unici per ogni utente.
- **Individuazione di malware** : Creazione di classificatori capaci di analizzare i processi attualmente in esecuzione su una macchina, allo scopo di identificare quelli riconducibili a malware.
- **Sistemi per la preservazione della privacy** : Definizione di nuovi protocolli e meccanismi, come l'adozione della tecnologia blockchain, per migliorare il livello di privacy offerto dai sistemi IoT.

Gli ambienti nel quale sono stati effettuati gli studi analizzati sono stati, quando menzionati, prevalentemente: Smart Home, ambienti sanitari, industriali, produzione di energia elettrica e Smart City. Nella maggioranza dei casi le soluzioni adottate non sono tuttavia dipendenti dallo specifico ambiente in cui viene impiegato l'IoT [6].

Impiego dell'intelligenza artificiale

Grazie allo studio di Giordano et al. [6], è inoltre possibile discutere dei vari tentativi di impiego dell'intelligenza artificiale. Gli approcci più utilizzati nell'ambito del machine learning sono quelli di tipo *shallow*, in particolare quelli di tipo supervisionato. Algoritmi

di classificazione e di regressione, nello specifico quelli maggiormente impiegati risultano essere *Random Forest* e *SVM* (*Support Vector Machine*), creano un modello predittivo tramite la definizione di un insieme di dati di training. Meno popolari in letteratura sono le tecniche non supervisionate, come il clustering, di cui l'algoritmo più utilizzato è risultato il *k-Means*. Queste sono state maggiormente impiegate in combinazione con le tecniche supervisionate, come nel caso della creazione di sistemi per l'individuazione di attacchi dall'analisi del traffico di rete. Un problema di privacy importante relativo all'utilizzo di tecniche di clustering consiste nella diffusione di informazioni sensibili nel processo di creazione dei dati di training, o nella suddivisione in gruppi di traffico anomalo presente in una rete [6]. In merito alle tecniche di tipo *shallow*, grazie al lavoro di Albalawi [1] è possibile brevemente analizzare nello specifico quali problematiche sono state affrontate dai diversi algoritmi (2.1).

Tabella 2.1: Problemi di sicurezza trattati da algoritmi di *shallow* machine learning [1]

Algoritmo	Problemi affrontati
SVM	Identificazione, mitigazione e analisi di intrusioni, malware, attacchi DDOS
K-Nearest Neighbor	Identificazione, mitigazione e analisi di intrusioni, anomalie nella rete, attacchi DDOS
Naive Bayes	Identificazione e analisi di intrusioni e anomalie nella rete
Random Forest	Identificazione di intrusioni, attacchi DDOS, anomalie nella rete, dispositivi non ammessi
Decision Tree	Identificazione di intrusioni, anomalie nella rete, fonti di traffico sospetto
K-Means Clustering	Identificazione di intrusioni, anomalie nella rete, individuazione di attacchi Sybil in Wireless Sensor Network industriali e anonimizzazione di dati privati
Principal Component Analysis	Identificazione, mitigazione e analisi di malware

Più recente è invece l'utilizzo di tecniche di *deep* learning, prevalentemente per confrontare le prestazioni di diversi algoritmi circa le stesse problematiche [6]. Risulta quindi evidente la mancanza di confronti con tentativi facenti utilizzo di diverse tecniche di apprendimento non supervisionato o tecniche di ottimizzazione, come gli algoritmi evolutivi [6].

2.2 Sistemi IoT basati su blockchain

Le principali problematiche relative alla privacy ed alla sicurezza dei sistemi IoT derivano dalla loro natura fortemente centralizzata . Lo scambio di informazioni e i meccanismi di autenticazione e autorizzazione avvengono attraverso un unico server centrale, che si occupa dell'intera logica [8] . Come illustrato dalle fonti [8] e [9], per risolvere queste problematiche diversi sistemi IoT fanno utilizzo della tecnologia blockchain come base per lo scambio di informazioni tra nodi. Questo è un nuovo paradigma che porta una gestione dei dati decentralizzata, poiché si basa su registri distribuiti su di una rete peer-to-peer.

Con il termine "Blockchain" si intende un insieme di tecnologie che utilizzano una catena di blocchi, contenenti transazioni, per la gestione del proprio registro. Viene utilizzato il meccanismo del consenso, distribuito tra tutti i nodi della rete, per accertarsi del corretto ordine delle transazioni validate da includere nei blocchi [8] .

Questa nacque con l'avvento dei Bitcoin, principalmente con lo scopo di validare le transazioni senza dover richiedere l'intervento di un agente esterno, portando quindi alla decentralizzazione. Le soluzioni blockchain si possono catalogare in *"permissionless"*, quando chiunque può diventare un nodo della rete e partecipare alla validazione delle transazioni, *"permissioned"*, quando invece l'accesso è ristretto a partecipanti selezionati, e soluzioni ibride come quelle di tipo *"consortium"* [8] .

2.2.1 Vantaggi

I lavori di Hassan et al. [8] e di Ahmed Teli et al. [9] espongono come la blockchain apporti ai sistemi IoT un gran numero di vantaggi, sia in termini di prestazioni che di sicurezza:

- **Decentralizzazione**: La natura decentralizzata di questi sistemi risolve i problemi legati all'utilizzo di un singolo server centrale per la gestione dei dati, che normalmente rappresenta sia un limite per le prestazioni del sistema, sia un unico possibile punto di fallimento. La compromissione del server centrale potrebbe causare danni irreparabili ad ogni singolo nodo del sistema. L'adozione di un sistema decentralizzato aggiunge sia robustezza che affidabilità all'intera rete. In questo modo, i dati non sono memorizzati in una struttura centrale, ma ogni nodo partecipante possiede una copia completa degli stessi [8].

- **Transazionalità** : Lo scambio di dati tra nodi partecipanti alla blockchain avviene tramite transazioni, in genere adottando meccanismi di crittografia allo scopo di proteggere

le stesse. In sistemi di tipo IoT, quando si parla di transazioni ci si riferisce a qualsiasi azione generata dai partecipanti del sistema [8]. La blockchain registra queste operazioni nei blocchi in modo sia da verificarne la validità sia assicurando la stabilità della "catena" così formata, tramite uno dei possibili meccanismi di consenso, ostacolando tentativi di manomissione [8] [9].

- **Trasparenza** : I blocchi contenenti le transazioni che costituiscono la blockchain sono completamente trasparenti ai partecipanti. In questo modo, è facilmente possibile individuare nodi che mettono a repentaglio la sicurezza o la privacy degli utenti, per poi ostacolarne le azioni [8] .

- **Immutabilità** : La blockchain si configura come una catena di tipo *"append-only"*, rendendo quindi impossibile modificare o eliminare transazioni già eseguite. Questo risolve i problemi di manipolazione e interferenza sui dati, assicurando gli utenti riguardo possibili manipolazioni dei dati prodotti [8] .

- **Mancanza di terze parti** : Uno dei fattori maggiormente connessi a problemi di privacy e di sicurezza è l'utilizzo di componenti di terze parti a cui sono affidate le operazioni di gestione dei dati e validazione delle transazioni [8]. I sistemi IoT basati su blockchain generalmente non si affidano ad alcuna componente di terze parti, diminuendo quindi i rischi legati alla manomissione o al cattivo utilizzo delle stesse [8].

2.2.2 Problemi

Nonostante l'utilizzo della tecnologia blockchain porti a numerosi vantaggi, tali sono anche le problematiche annesse.

Come notato da Ahmed Teli et al. [9] , le blockchain non sono state inizialmente progettate per gestire quantità di dati molto elevate, tipiche dei sistemi IoT che utilizzano sensori. Per questo motivo, spesso si rende necessario agire sui dati, modificandoli e comprimendoli al fine di poterli immagazzinare. Per quanto concerne i meccanismi con cui viene creato il consenso relativo alle nuove transazioni generate dagli utenti, i dispositivi che partecipano al sistema IoT in genere non possono essere utilizzati a causa delle loro caratteristiche tecniche, affidando questo tipo di computazioni generalmente a soluzioni *off-chain*. Inoltre, le differenze di prestazioni e risorse offerte dai dispositivi che partecipano alla rete introducono una disomogeneità nei tempi di risposta e nelle capacità di memorizzazione dei dati [9].

Anche l'immutabilità della blockchain può portare a problemi: i sensori di un sistema potrebbero generare dati invalidi, che però non potranno essere mai cancellati dalla blockchain

a causa della sua natura [9]. I maggiori pericoli legati all'utilizzo di una blockchain sono però relativi alla privacy dei dati. Comunicando in modo pubblico, i nodi possono essere anche soggetti a problemi di data leakage [8], che possono sorgere fin dall'estrazione dei dati, poiché i dispositivi possono essere acceduti da personale non autorizzato [9].

Le fonti [8] e [9] racchiudono anche un'ottima analisi dei rischi di privacy collegati all'utilizzo della blockchain:

- **Riuso degli indirizzi pubblici** : Gli indirizzi pubblici degli utenti che partecipano alla blockchain possono essere ottenuti ed utilizzati da chiunque per diversi scopi. L'utilizzo di pseudonimi per gli utenti non rappresenta una soluzione applicabile, in quanto non è complesso inferire da questi la reale identità degli utenti [8].

- **De-anonimizzazione** : Pur utilizzando tecniche di anonimizzazione nella generazione delle informazioni relative ad una transazione, tramite la combinazione dei dati ottenuti dall' analisi della blockchain e degli indirizzi pubblici è possibile ricavare la reale identità degli utenti [8].

- **Privacy leakage dei wallet** : In molti sistemi blockchain, la gestione delle transazioni è delegata a software chiamati "wallet", i quali si occupano di generare ed eliminare indirizzi non permanenti usati per registrare le transazioni, allo scopo di eliminare i problemi connessi all'utilizzo di indirizzi reali. Tuttavia, l'adozione di questo tipo di software consente a malintenzionati di poter utilizzare tecniche per dedurre informazioni private relative alla blockchain analizzando il comportamento degli wallet [8].

- **Attacchi Sybil** : In questo tipo di attacco, utenti malintenzionati generano una grande quantità di nodi "artificiali" nel sistema IoT, allo scopo di ottenere la maggioranza del potere decisionale nella rete. In questo modo è possibile aggirare il meccanismo di sicurezza del consenso implementato in questi sistemi, portando a possibili gravi problemi di privacy [8].

- **Message Spoofing** : Consiste nell'introduzione di messaggi intenzionalmente errati all'interno della blockchain per creare inconsistenze rispetto al reale stato dell'ambiente. In sistemi IoT di importanza critica, come quelli utilizzati in ambiti ospedalieri o da veicoli, può essere un serio pericolo per la sicurezza [8].

- **Linkage attacks** : Questa tecnica è utilizzata per ottenere informazioni private da dati anonimizzati, utilizzando algoritmi che permettono di trovare collegamenti tra dati presenti su diversi registri distribuiti della blockchain [8].

Un altro problema, non unicamente derivante dall'utilizzo della blockchain ma molto concreto, è il **Denial of Service attack (DOS)**, che consiste nell'ostacolare il normale utilizzo di un servizio [10], che in un ambiente di tipo Internet of Things può risultare in un pericolo critico o addirittura fatale.

Una serie di possibili strategie da adottare per limitare questi rischi sono riportate da Hassan et al. [8]

- **Crittografia** : La gran parte delle reti blockchain utilizza una crittografia che adotta chiavi pubbliche e private per rendere sicura la trasmissione di informazione tra nodi. Garantisce una forte privacy, ma aumenta considerevolmente il carico computazionale, rischiando di rendere inagibile una rete. Inoltre la generazione di chiavi porta ad altri rischi, poiché la manipolazione di questo processo potrebbe causare gravissimi problemi di privacy leakage [8].

- **Anonimizzazione** : Un altro metodo per la preservazione della privacy consiste nell'applicare tecniche che consistono nel modificare i dati sensibili degli utenti generati dai dispositivi, in modo da renderli anonimi. Queste tecniche sono ovviamente suscettibili ad attacchi di tipo Linkage, citati precedentemente. Inoltre, l'anonimizzazione introduce uno strato di complessità per l'analista dei dati, che non ha a disposizione informazioni chiare sugli utenti, ostacolando ricerche ed analisi complesse [8].

- **Smart Contract** : "Contratti" esterni in cui sono racchiusi i dettagli delle transazioni, allo scopo di ridurre il carico di informazioni contenute nei singoli blocchi e di fornire un insieme di operazioni richieste alla registrazione di una nuova transazione. Questi sono strumenti sicuri e autonomi che migliorano la sicurezza e le prestazioni di un sistema IoT, tuttavia necessitano l'utilizzo di componenti di terze parti e possono essere soggetti ad errori algoritmici, in quanto programmabili manualmente. Inoltre, risulta difficile gestire contratti ambigui e determinare le norme giuridiche contrattuali da seguire [8].

- **Mixing** : Protocolli che permettono di interfogliare indirizzi e altri dettagli relativi a diverse transazioni modificando il flusso delle stesse, allo scopo di rendere più difficile il tracciamento degli utenti nelle blockchain. Tuttavia, il livello di protezione offerto

da questa tecnica è piuttosto basso, in quanto è possibile ricostruire interamente una transazione tramite l'analisi del traffico della rete [8].

- **Privacy differenziale** : Protocollo leggero, facile da implementare, che aggiunge incertezza ai dati memorizzati inserendo dati spuri con una certa casualità allo scopo di aumentare il livello di privacy, riducendo però l'accuratezza dei dati così generati [8].

2.2.3 Caratteristiche delle blockchain vulnerabili

Come analizzato nelle Sezioni precedenti, le blockchain possono essere molto diverse tra loro, ad esempio per quanto concerne l'accessibilità, i meccanismi di validazione e consenso, le tecniche di anonimizzazione e molto altro.

Le principali caratteristiche delle blockchain connesse a problemi di privacy leakage sono state identificate da Junejo et al. [11].

Due di queste risultano essere la visibilità e l'accessibilità. Nelle **blockchain pubbliche** infatti le transazioni possono essere viste da chiunque, mentre la possibilità di generare nuove transazioni diversifica le blockchain in permissionless ("senza permesso) e permissioned ("con permesso"). Il tipo di rete che risulta più schermata da attacchi alla privacy degli utenti è quindi sicuramente una **blockchain privata e permissioned** [11].

Un'altra caratteristica analizzata è la totale assenza di servizi di **mixing** o l'utilizzo di servizi poco sicuri, come il **mixing centralizzato**. Utilizzando questo tipo di tecnica, il server centrale che gestisce il processo di mixing, illustrato nella Sezione 2.2.2, risulta essere un *single point of failure*, centralizzando quindi anche i rischi legati a possibili attacchi di tipo denial of service, oltre a risultare in peggiori prestazioni nella generazione di transazioni [11].

Anche l'utilizzo di alcuni protocolli per la validazione e la generazione di consenso possono rendere la blockchain suscettibile ad attacchi alla privacy, mentre altri possono portare a dei vantaggi.

In particolare, è stato ideato ed utilizzato il protocollo **Zero Knowledge Proof**, denominato in questo modo poiché controlla la validità di una transazione nascondendo agli utenti le informazioni sensibili riguardo la transazione in esame. In questo modo si eliminano completamente i possibili rischi di leakage riguardo dati riservati delle transazioni durante il processo di validazione, rendendo la blockchain più sicura [11].

Un altro protocollo adottato dalle blockchain allo scopo di fornire maggiore sicurezza alle proprie transazioni è il ***ring signature***. Questo è un meccanismo utilizzato per garantire l'anonimato all'utente che inizia una transazione. Il cuore di questa tecnica consiste

nel firmare le transazioni con un insieme di chiavi di più utenti, in modo da ostacolare il tracciamento del reale autore. La dimensione delle firme rappresenta però un limite non trascurabile all'utilizzo di questa tecnica in ambienti di grandi dimensioni [11]. Per analizzare con maggior precisione lo stato di una blockchain, Putz et al. [12] illustrano una molteplicità di **metriche** da poter interpretare come indicatrici di possibili minacce. Ad esempio, allo scopo di individuare un attacco DOS, è possibile analizzare le metriche di *throughput* e latenza delle transazioni, mentre per verificare il corretto comportamento del meccanismo di consenso occorre interpretare dati come il numero di blocchi scartati e la frequenza nella creazione delle transazioni. In genere è possibile ottenere queste metriche semplicemente effettuando una collezione di *log* dai dispositivi che partecipano alla rete [12].

2.2.4 Impiego dell'intelligenza artificiale

L'impiego di algoritmi di machine learning si è rivelato utile anche specificamente ai sistemi IoT basati su blockchain. Oltre ai già citati utilizzi per i problemi relativi alla sicurezza e alla privacy, questi sono stati impiegati anche per motivazioni strettamente legate alle blockchain. Sono stati proposti algoritmi basati sul Deep Reinforcement Learning per impostare dinamicamente la produzione dei blocchi, la loro dimensione, gli algoritmi per la generazione di consenso e per aumentare le performance del sistema [13] . I tentativi di utilizzo dell'intelligenza artificiale nel mondo della blockchain consistono quindi nel fornire supporto alla rete, aumentandone le prestazioni.

In generale, il mondo dei sistemi IoT basati su blockchain è ancora in forte espansione e le ricerche e i miglioramenti da effettuare in termini di protezione della privacy sono ancora molteplici.

2.3 Vulnerabilità dei sistemi IoT

2.3.1 Panoramica delle vulnerabilità comuni

Allo scopo di meglio individuare i problemi di sicurezza e di privacy legati a sistemi IoT, risulta necessaria un'analisi più concreta dei principali malware ed exploits in grado di causare i numerosi problemi descritti nelle Sezioni precedenti. A tal motivo, grazie all'analisi di un dataset Europeo relativo ai dispositivi IoT infetti ("Infected and Exposed IoT device statistics"[2]), è stato possibile estrapolare i principali malware ed exploits impiegati attualmen-

[2] `https://data.europa.eu/data/datasets/infected-and-exposed-iot-device-statistics?locale=en`

te, di cui alcuni dei più rilevanti per le tematiche in esame sono riportati nella Tabella 2.2. L'analisi del dataset è stata svolta utilizzando la libreria CSVParser[3] e gli strumenti forniti dal linguaggio Java, andando ad analizzare il file in formato CSV contenente il dataset, creando collezioni di coppie chiave-valore tramite le strutture dati del linguaggio, in particolare mappe e liste, che contenessero il nome dell'attacco e la relativa somma dei dispositivi colpiti da esso presenti nell'insieme dei dati. Le collezioni di coppie così ottenute sono state poi ordinate per valore, in modo da analizzare gli attacchi più presenti.

Tabella 2.2: Principali malware ed exploits utilizzati per attaccare dispositivi IoT

Malware/Exploit	Dispositivi IoT affetti
mirai	63277
telnet-brute-force	27407
ssh-brute-force	18945
ghost-push	9582
unityminer	8418
xcodeghost	8171
vpnfilter	3590
CVE-2017-17215	499

Analizzando gli attacchi più frequenti, è quindi anche possibile individuare quali sono le caratteristiche dei dispositivi IoT che li rendono maggiormente a rischio.

Mirai è un malware utilizzato per manipolare principalmente dispositivi IoT, allo scopo di renderli parte di una *botnet*, in modo da prendere il controllo di una rete di computer per poi lanciare vari tipi di attacchi tramite l'impiego di un server di controllo. Il virus utilizza un dispositivo infetto per cercare altri dispositivi IoT aventi porte aperte da poter infettare tentando di accedervi utilizzando le credenziali di default che sono generalmente associate alla gran parte dei dispositivi [14].

Telnet-brute-force e SSH-brute-force sono tecniche molto semplici che permettono di effettuare l'autenticazione in dispositivi che utilizzano questi protocolli per lo scambio di

[3] `https://commons.apache.org/proper/commons-csv/apidocs/org/apache/commons/csv/CSVParser.html`

dati, semplicemente tentando combinazioni di username e password con una strategia di tipo *brute-force* [15] [16], per poi accedere a dati sensibili o rendere il dispositivo parte di una *botnet* [15]. Anche il malware **unityminer** utilizza la stessa tecnica per accedere ai dispositivi ed installare software allo scopo di effettuare *mining* di criptovalute utilizzando le risorse dei dispositivi [17].

La principale causa di queste vulnerabilità risiede nell'utilizzo di credenziali deboli, spesso quelle di default, che espongono il dispositivo ad attacchi di questo tipo.

Ghost Push è un trojan malware che infetta dispositivi con sistema operativo Android allo scopo di accedere al loro root e manipolarlo per presentare pubblicità all'utente. Questi sono poi in grado di spiare le informazioni private, minando quindi alla privacy degli utenti [18].

XcodeGhost è un malware che consiste in una versione dannosa dell'ambiente di sviluppo Xcode, utilizzata in dispositivi Apple. Le applicazioni manipolate possono essere utilizzate per raccogliere informazioni come il nome del dispositivo e il suo tipo, l' UUID del dispositivo e il tipo di rete, ponendosi come una seria minaccia per la privacy [19].

VPNFilter è un malware progettato per carpire informazioni sensibili come le credenziali di accesso a siti web e rendere inutilizzabili i dispositivi infetti, che sono prevalentemente vulnerabili poiché privi di servizi di sicurezza [20].

CVE(Common Vulnerabilities and Exposures)-2017-17215 è una vulnerabilità[4] che consente l'esecuzione di codice da remoto su dispositivi Huawei. Un utente malintenzionato che ha effettuato l'autenticazione può inviare pacchetti alla porta 37215 per avviare attacchi, facendo eseguire del codice arbitrario.

Altre vulnerabilità comuni estratte da database di exploits[5] [6] sono comuni attacchi di SQL Injection tramite richieste HTTP POST, attacchi Denial of Service che sfruttano vari tipi di overflow o errori nel controllo di puntatori nulli e accesso alla memoria.

2.3.2 Device Spoofing

Uno degli attacchi più pericolosi per una rete IoT consiste nel Device Spoofing, già trattato precedentemente. Ling et al. [21] illustrano i passaggi per lanciare un attacco di

[4] `https://nvd.nist.gov/vuln/detail/CVE-2017-17215`

[5] `https://nvd.nist.gov/`

[6] `https://www.exploit-db.com/`

questo tipo, allo scopo di prendere il controllo dell'identità di una *Smart Plug* (presa/spina intelligente), strumento altamente utilizzato in ambienti di tipo IoT domestici e quindi tendenti a conservare molti dati riservati. L'autore crea un bot software che imita una presa ed esegue il processo di autenticazione HTTP in modo da ottenere le credenziali del controller, ovvero il dispositivo (in genere uno smartphone) con il quale comunica. L'utente seleziona una presa tramite il suo indirizzo MAC, assumendo che questa sia online e non utilizzi credenziali di default, e tramite l'indirizzo si registra al servizio di autenticazione, in questo caso in un server remoto, inviando due pacchetti UDP consecutivi al server: il primo per informare il server delle registrazione alla rete, e il secondo contenente le informazioni del dispositivo che sta tentando di registrarsi (indirizzo MAC, tipo, alias, indirizzo IP, numero di porta e versione del firmware). Considerato che il server non utilizza un metodo di autenticazione per autenticare le prese, registra la presa *spoofed* ed invia un pacchetto UDP per la conferma. Da questo momento, il bot risulta essere online nella rete IoT. Quando l'utente vittima dell'attacco aprirà dal suo smartphone (il controller) l'applicazione per comunicare con la presa selezionata dall'attaccante, invierà una richiesta UDP al server di autenticazione, contenente nel campo "auth value" le sue credenziali di username e password, che il server inoltrerà successivamente alla presa *spoofed*. In questo modo, un utente malintenzionato può utilizzare le credenziali del controller così derivate per qualsiasi scopo, caratterizzando quindi il Device Spoofing come un grande pericolo per la privacy.

2.3.3 Deanonimizzazione in blockchain

Come già riportato nella Sezione 2.2.2, gli attacchi di deanonimizzazione risultano tra i più pericolosi per la privacy in ambienti che utilizzano transazioni per lo scambio di dati tra gli utenti. Conti et al. [22] hanno esposto le tecniche maggiormente utilizzate per la deanonimizzazione delle transazioni in una rete blockchain, in particolare Bitcoin. Questa rende anonimi i reali indirizzi degli utenti fornendo una lista di possibili indirizzi pubblici ad ogni dispositivo, per cui un utente malintenzionato ha come scopo quello di creare un mapping uno-a-molti tra un utente e gli indirizzi a lui associati. Per fare ciò è possibile effettuare un'analisi della blockchain, che però necessita di tre passi di pre-processing:

- **Grafo transazionale** : La blockchain può essere vista come un grafo aciclico delle transazioni in cui i nodi, collegati da archi unidirezionali, rappresentano le transazioni memorizzate nei blocchi.

- **Grafo degli indirizzi** : Attraversando il grafo transazionale è possibile dedurre le relazioni tra i vari indirizzi di input e output, dalle quali è possibile costruire il grafo degli indirizzi, in cui i nodi sono gli indirizzi degli utenti.

- **Grafo delle entità** : Combinando il grafo degli indirizzi così creato con delle **euristiche** dedotte dalle tecniche e dai protocolli usati per generare l'anonimato nella rete è possibile creare dei grafi di entità in cui i nodi sono degli insiemi di indirizzi che sembrano appartenere allo stesso utente, connessi tra loro in base alle transazioni effettutate. Le euristiche utilizzate dalla fonte accomunano gli indirizzi usati come input per le stesse transazioni poiché generalmente queste sono create da un singolo utente, mentre gli indirizzi di output delle transazioni che non compiono ulteriori movimenti sono accomunati a quelli di input. In base all'utilizzo di euristiche diverse, che possono risultare anche molto complesse, il grafo delle entità può generare un numero variabile di **falsi positivi** in seguito al clustering degli indirizzi.

Raggiunta una precisione soddisfacente è poi possibile collegare queste informazioni con identità del mondo reale, tramite il confronto con gli indirizzi utilizzati per accedere ad altri servizi. Diverse compagnie hanno creato e reso disponibili dei tool per poter analizzare facilmente delle blockchain, in particolare quelle Bitcoin, allo scopo di poter rintracciare gli utenti che compiono attività illecite nella rete. Ad esempio, se un utente effettua un acquisto tramite Bitcoin è possibile collegare il suo indirizzo con quello di altre transazioni nella blockchain, e dai dettagli delle transazioni arrivare ai cookie dell'utente, dai quali è possibile inferire la sua reale identità. In blockchain utilizzate in ambienti IoT, queste tecniche di deanonimizzazione possono essere usate per collegare un indirizzo IP ad uno specifico utente semplicemente analizzando le transazioni della rete.

2.3.4 Linkage attack

Come illustrato nella Sezione 2.1.2, tra i problemi di privacy principali legati al mondo IoT troviamo l'identificazione, il tracciamento, la profilazione e il linkage.

Lu et al. [23] illustrano come sia possibile associare la reale identità degli utenti di un sistema IoT ai loro dispositivi. I ricercatori hanno utilizzato un semplice dispositivo, un Raspberry Pi (RPi) 3 dotato di microfono e videocamera, per carpire volti e voci degli utenti di un ufficio, mentre gli indirizzi MAC dei dispositivi della rete WiFi sono stati acquisiti tramite un processo di *sniffing*. L'obiettivo dello studio è dimostrare come sia semplice associare, tramite tecniche di clustering, volti e voci dei frequentatori abituali di un ambiente

di lavoro con gli indirizzi "anonimi" dei loro dispositivi, combinando informazioni da più fonti. Dopo una fase iniziale di raccolta dei dati biometrici (volti e voci dei volontari) e degli indirizzi MAC, avvenuta in molteplici sessioni di incontri in spazi pubblici, è stato eseguito un processo di *filtering*, allo scopo di rimuovere dal dataset così creato i dispositivi di poco interesse, come gli access point WiFi, e dispositivi i cui pacchetti sono stati intercettati ma che non si trovavano nel luogo di interesse dell'indagine, analizzando l'intensità del segnale ricevuto da essi. In seguito è stato possibile applicare una *feature extraction* dei dati biometrici tramite delle reti neurali (utilizzando facenet per i volti e x-vector per le voci), per poi andare a creare un *linkage tree* da utilizzare per eseguire un **clustering gerarchico** dei dati biometrici, allo scopo di individuare immagini e voci relative ad una singola persona. I nodi foglia dell'albero sono i singoli campioni, mentre i nodi intermedi rappresentano i cluster di tutti i nodi foglia discendenti. Ad ogni nodo è inoltre associato un *linkage score* che indica il grado di somiglianza delle feature estratte dai campioni così aggregati. A queste informazioni è stato poi aggiunto un *context vector* per ogni nodo, contenente le informazioni circa la presenza o l'assenza dei campioni del cluster alle varie sessioni di raccolta dati eseguite. Lo stesso vettore è associato agli indirizzi MAC ottenuti dalla fase di filtering, allo scopo di poter poi collegare i cluster creati all'indirizzo MAC che presenta la maggior somiglianza relativamente ai valori del *context vector*. In seguito all'associazione tra nodi ed indirizzi, è necessario selezionare i nodi da considerare con un processo di *node selection*. In particolare, il valore di ogni nodo è dato da una somma pesata tra il grado di somiglianza delle feature dei campioni e il valore di somiglianza del context vector di un nodo con quello dell'indirizzo MAC a lui associato. Vengono così selezionati i migliori K cluster, dove in questo caso K equivale al numero di volontari che hanno partecipato allo studio. La possibile tecnica di mitigazione per questo attacco illustrata dalla fonte consiste nell'adozione della randomizzazione degli indirizzi MAC da parte dei dispositivi partecipanti al sistema IoT, allo scopo di aggiungere un grado di incertezza maggiore alle rilevazioni del dispositivo attaccante.

CAPITOLO 3

Stato dell'arte

In seguito all'analisi del background circa le tematiche in esame, è possibile concentrarsi ora sull'impiego delle tecniche di intelligenza artificiale di interesse, in particolare degli algoritmi genetici, in merito alle problematiche esposte. Nonostante la poca presenza in letteratura di tecniche per la rilevazione di problemi di privacy in ambienti IoT utilizzanti algoritmi genetici, questi sono stati notevolmente esplorati per problemi più generici di individuazione di vulnerabilità.

3.1 Individuazione di vulnerabilità

Le fonti [24] e [25] illustrano il modo in cui questi sono stati impiegati per la generazione di input potenzialmente pericolosi nelle interazioni con Web Application, allo scopo di generare vulnerabilità di tipo **XSS** (**cross-site scripting**), che permettono l'introduzione o l'esecuzione di codice in siti web tramite l'utilizzo di form non correttamente sanificati, allo scopo di generare gravi problemi di sicurezza e di privacy, ottenendo o manipolando dati sensibili. Le due fonti, pur utilizzando approcci diversi ([24] white box, [25] black box), impiegano gli algoritmi genetici per lo stesso scopo. Queste esercitano diversi percorsi nell'esecuzione dell'applicazione web generando input potenzialmente dannosi tramite un approccio evolutivo. La codifica degli individui consiste in una serie di parametri letterali e/o numerici, concatenati seguendo diversi criteri per comporre una richiesta verso il sito da testare. In particolare, i valori da inserire nei parametri non vengono generati in modo completamente casuale, ma sono scelti da un insieme di input potenzialmente dannosi raccolti dagli autori in seguito all'analisi delle vulnerabilità già note e più comuni. Una volta generata la popolazione iniziale, gli individui sono valutati in base ad una funzione fitness, che considera diversi fattori come il numero di caratteri correttamente iniettati e la singolarità dell'input, ed indica quanto un individuo sia vicino a provocare un XSS. L'analisi dei risultati per la rilevazione dei valori di fitness avviene comparando gli output prodotti dagli individui con dei pattern di XSS ben noti. Nel lavoro di Duchene et al. [25], ad esempio, ciò avviene analizzando la somiglianza del Document Object Model del sito prodotto dalla richiesta con delle espressioni regolari che identificano una vulnerabilità di questo tipo. Gli individui che superano una certa soglia di valore fitness sono poi selezionati per effettuare le operazioni di crossover e mutazione (quest'ultima avviene generalmente con probabilità minore), allo scopo di generare nuovi individui. L'operazione di mutazione consiste nel modificare gli attributi della richiesta presenti nell'individuo e/o nel cambiare i valori degli stessi. Il crossover invece genera individui figli che hanno valori degli attributi combinati

dai genitori. La creazione di nuove generazioni di individui viene interrotta quando risulta trovata una vulnerabilità per quel cammino o quando si raggiunge un numero definito di generazioni. Tramite gli algoritmi genetici è quindi possibile creare dei veri e propri casi di test in grado di mettere alla prova la sicurezza di applicazioni web.

3.2 Evoluzione di malware

Altri utilizzi degli algoritmi genetici consistono nell'evoluzione di software, in particolare di malware. Noreen et al. [26] mostrano come è possibile **evolvere virus** appartenenti alla famiglia dei **Bagle**, che vengono propagati tramite allegati di email allo scopo di creare una backdoor nel sistema operativo. Gli individui sono codificati da una serie di parametri che costituiscono le caratteristiche ad alto livello dei virus, come il nome dell'applicazione in cui si nascondono, l'oggetto e il contenuto delle email che inviano e altri, che vengono poi tradotti in codice a basso livello da un *code generator*. Gli operatori di crossover utilizzati sono quelli classici, come il *single-point* e l' *uniform crossover*. Il valore di fitness tramite il quale vengono selezionati gli individui migliori consiste nel grado di somiglianza del virus generato con i virus presenti nell'insieme dei malware di training. Lo scopo della ricerca compiuta in [26] potrebbe essere quello di creare virus di allenamento per un antivirus, in grado quindi di migliorare la sicurezza dei dispositivi.

3.3 Feature Selection

Un altro impiego utile degli algoritmi genetici risulta essere quello studiato dalla fonte [27], che li combina a tecniche di classificazione. Lo scopo in questo lavoro è quello di classificare correttamente intrusioni in una rete analizzando le caratteristiche del traffico. In questo caso, gli algoritmi genetici risultano particolarmente utili nella fase di feature selection, ovvero nella selezione delle caratteristiche del dataset da prendere in considerazione per la creazione e l'allenamento del modello di classificazione. Gli individui di questo algoritmo genetico sono quindi delle semplici stringhe di bit, in cui ogni bit corrisponde alla presenza/assenza di una feature del dataset. Il modo in cui gli individui sono valutati è tramite il numero di errori nella classificazione del modello creato e allenato sulle feature indicate dall'individuo. Selezione, crossover e mutazione avvengono poi utilizzando i metodi *rank selection*, *two point crossover* e *bit flip mutation*. Gli individui migliori sono poi utilizzati per la creazione di modelli di classificazione utilizzanti l'algoritmo Decision Tree, già dimostrato essere uno dei più

efficienti per questo tipo di problematiche. I risultati ottenuti da Stein et al. [27] mostrano un notevole miglioramento nelle percentuali di errori sulla classificazione degli attacchi alla rete (Probe, DOS, R2L e U2R) rispetto ai corrispettivi modelli che non utilizzano GA per la selezione delle caratteristiche. Oltre ai vantaggi in termini di prestazioni, gli algoritmi genetici si dimostrano utili anche perché permettono di ottenere soluzioni sub-ottime senza la necessità di analizzare precisamente le possibili configurazioni, ma basando le valutazioni su osservazioni empiriche, come avvenuto in questo caso.

3.4 Individuazione di intrusioni

Nel lavoro di Li [28], l'autore invece utilizza gli algoritmi genetici per la creazione di regole in grado di individuare intrusioni potenzialmente pericolosi in una rete, analizzandone le connessioni. A differenza della fonte [27], i GA sono qui impiegati in modo diretto e non in combinazione con altri algoritmi. Gli individui in questo caso rappresentano delle regole di tipo *if(condizione) then{azione}*, dove i singoli individui presentano differenti valori dei parametri nella *condizione*. I parametri considerati sono molteplici, come l'indirizzo di origine della connessione, l'indirizzo di destinazione, il protocollo utilizzato, la durata della connessione, il numeri di byte trasmessi e molti altri. L'azione indicata dalle regole è invece sempre quella di bloccare la connessione sospetta. Ai parametri considerati sono poi associati dei pesi, i quali influiscono sulla valutazione assegnata ad ogni individuo. La funzione di valutazione difatti si basa sulle prestazioni della regola indicata dall'individuo nell'identificare correttamente intrusioni da un dataset di connessioni già classificate. Allo scopo di correttamente individuare più intrusioni, è necessario ricavare un insieme di regole che siano il più diverse possibile. Nel processo di selezione, crossover e mutazione vengono quindi favoriti gli individui che più differiscono dalla popolazione, basandosi su delle metriche che analizzano sia il genotipo che il fenotipo degli individui.

Un approccio simile è stato applicato da Paliwal et al. [29] per l'individuazione di sei tipologie di attacchi ad una rete, che ricadono nelle categorie di attacchi DOS, Probing attack e R2L (Remote to Local attack). Anche in questo caso il dataset di partenza, contenente connessioni alla rete etichettate dal tipo di attacco in atto (se presente), è stato suddiviso in un insieme di allenamento e uno di testing. L'insieme di dati di allenamento è stato sottoposto ad una fase di feature extraction, ed è stato utilizzato per la creazione di regole *if(condizione) then{azione}* per l'individuazione di diversi attacchi. Questo differisce dal lavoro di Li [28] poiché considera un minor numero di feature per la creazione delle regole, ed inoltre non si

limita ad individuare le connessioni pericolose ma mira a classificarle. Anche la funzione di fitness risulta differente, difatti in questo caso questa è misurata come una somma pesata del rapporto tra la quantità di intrusioni classificate correttamente sul numero di connessioni dell'insieme di allenamento e il rapporto tra il numero di intrusioni individuate correttamente sul numero di connessioni classificate come intrusioni.

Anche Al-fuhaidi et al. [3] hanno applicato gli algoritmi genetici al problema della classificazione di attacchi in una rete, in particolare usando il medesimo dataset utilizzato dalla fonte [29], generando allo stesso modo delle regole *if(condizione) then{azione}*. In questo caso però la funzione di fitness risulta essere diversa. In particolare, questo studio mostrava come le prestazioni ottenute dall'utilizzo di GA per individuare e classificare le intrusioni erano simili a quelle derivanti dalla creazione di modelli di classificazioni con algoritmi tradizionali, come ad esempio Decision Tree.

Un differente utilizzo dei GA in merito all'individuazione di intrusioni in una rete è quello esposto da Zhang et al. [30]. In questo caso, un algoritmo genetico viene applicato per determinare il numero di strati e di neuroni di un **DBN** (***Deep Belief Network***), ovvero un modello di classificazione, usato in questo caso per classificare le intrusioni in sistemi IoT basati su architetture four-layer. Gli individui dell'algoritmo genetico adottato sono sequenze di bit che indicano il numero di neuroni presenti in ogni strato del modello. La selezione avviene tramite *Roulette Wheel* applicando **elitismo**, ovvero conservando il miglior individuo ad ogni generazione. La funzione di fitness tramite la quale sono valutati gli individui indica l'accuratezza nella classificazione delle intrusione ottenuta da un modello DBN che presenta la configurazione indicata dall'individuo. In particolare, questa è una funzione pesata che favorisce gli individui meno complessi, allo scopo di trovare soluzioni più efficienti. La tecnica di crossover applicata è stata progettata in modo da far incrociare le caratteristiche di strati diversi di due individui, in modo da meglio navigare lo spazio delle soluzioni e non convergere prematuramente. I risultati ottenuti per l'identificazione di specifici attacchi sono superiori alle tecniche tradizionali, pur utilizzando soluzioni meno complesse.

3.5 Applicazioni in sistemi blockchain

Gli algoritmi genetici sono stati utilizzati anche per risolvere problemi di ottimizzazione in sistemi IoT basati su blockchain. Klaokliang et al. [31] mirano ad ottimizzare i parametri di configurazione dell'architettura decentralizzata che si occupa della gestione del consenso. In questo caso gli individui consistono nelle diverse possibili combinazioni di parametri. Le

caratteristiche da ottimizzare sono la dimensione dei blocchi, il tempo di attesa prima della creazione di un nuovo blocco e il limite di transazione per blocco. La funzione di fitness valuta gli individui sulla base delle prestazioni della blockchain con la configurazione indicata su 5000 transazioni di test, in particolare sul rapporto tra il numero di transazioni al secondo della configurazione in esame e quello della configurazione di default, moltiplicato per il tasso di successo del trasferimento della transazione. Ad ogni generazione sono selezionati i migliori 5 individui per partecipare alle operazioni di crossover e mutazione. La tecnica di crossover applicata è la *two-point crossover*, mentre quella di mutazione è la *resetting mutation*. I risultati conseguiti dall'utilizzo della configurazione ottenuta utilizzando questo GA (Genetic Algorithm) sono notevolmente superiori alle prestazioni di default, seppur in ambienti reali queste sono soggette a cambiamenti.

3.6 Analisi approfondita del lavoro scelto

A seguito dell'accurata analisi della letteratura esposta in questo Capitolo riguardante i sistemi IoT, i loro punti di debolezza, i principali attacchi e l'impiego degli algoritmi genetici, il lavoro condotto da Al-fuhaidi et al. [3] è risultato particolarmente utile poiché espone chiaramente le possibilità e le prestazioni dell'applicazione di algoritmi genetici nella classificazione di intrusioni in un ambiente IoT. Per tal motivo è stato scelto di analizzare approfonditamente l'approccio degli autori, cercando poi di realizzare un algoritmo genetico simile, per poi confrontare ed analizzare i risultati ottenuti. Ripetendo ed estendendo quanto detto nella Sezione 3.4, il lavoro di Al-fuhaidi et al. [3] si concentra sul dataset KDDCUP99 10% [32] per la creazione di un insieme di regole di classificazione delle intrusioni in un ambiente di tipo IoT di stampo militare, specializzandosi esclusivamente nell'individuazione di attacchi di tipo DOS. Il dataset citato è stato quindi suddiviso in due insiemi, uno utilizzato per effettuare il training delle regole di classificazione, e uno per il testing. Nella fase di pre-processing, gli autori hanno effettuato una conversione tra valori simbolici e valori numerici, e poi in seguito una normalizzazione dei dati, portando tutti i valori in un range compreso tra -1 e 1. Per la feature selection sono state attuate due strategie, che hanno poi portato ad altrettante soluzioni proposte: la prima utilizzava 18 feature sulla base del sistema operativo utilizzato e la seconda altre 31 feature basandosi sull'*Information Gain* delle stesse. La funzione di fitness utilizzata si configura come la differenza tra il rapporto costituito dal numero di attacchi classificati correttamente sul numero totale di attacchi nell'insieme di allenamento e quello costituito dal numero di connessioni classificate erroneamente come pericolose sul

numero totale di connessioni non pericolose nell'insieme di allenamento. Le metriche di valutazione scelte dagli autori sono state l'Accuracy, il Detection Rate e il tasso di falsi allarmi, che verranno approfondite nella Sezione 5.1.1, mentre per la Selezione sono stati utilizzati gli algoritmi tournament selection, roulette-wheel, ranking selection lineare ed esponenziale [3], anche se non è stato approfondito il modo in cui sono applicati. Per gli operatori di Crossover e Mutazione non sono stati riportati dettagli specifici. Per l'implementazione è stata utilizzata la libreria GALIB del linguaggio Java, ed inoltre sono stati effettuati confronti con altri classificatori per valutarne le prestazioni.

CAPITOLO 4

Design

In seguito all'analisi del background e dello stato dell'arte, è possibile trattare la progettazione dell'algoritmo proposto, analizzando le motivazioni e le scelte.

4.1 Obiettivo dell'algoritmo

Come anticipato nella Sezione 3.6, l'obiettivo dell'algoritmo genetico da sviluppare è quello di creare un insieme di regole *if(condizione) then{azione}* per la classificazione di attacchi all'interno di un ambiente IoT. Per farlo, è stato utilizzato il lavoro di Al-fuhaidi et al. [3] come linea guida, modificandone diversi aspetti per poi confrontare i risultati e trarre delle conclusioni circa l'efficacia degli algoritmi genetici e i migliori aspetti di questi.

4.2 Dataset analizzato

Il lavoro in esame si focalizza sull'individuazione dei soli attacchi di tipologia DOS all'interno del dataset KDDCUP99 [32], molto analizzato in letteratura per problematiche di privacy dell'IoT, come descritto da Giordano et al. [6]. Questo dataset è il risultato di una fase di analisi ed elaborazione di un altro insieme di dati, il DARPA 1998[1], che contiene informazioni riguardo connessioni in un network di tipo militare, terminata con l'aggiunta e l'accorpamento di alcune feature. Il dataset di partenza, creato dal Lincoln Lab del MIT, è stato formato utilizzando 7 settimane di simulazioni di interazioni verso 3 dispositivi connessi alla rete, il cui traffico è stato intercettato da uno *sniffer* [2]. I dati ottenuti sotto forma di file TCP dump sono stati poi elaborati per la creazione del dataset, le cui feature possono essere suddivise in quattro categorie [2]: le feature di base, ottenute dagli header dei pacchetti, le feature di contenuto, derivanti dal payload, le feature del traffico basate sul tempo come il numero di connessioni verso la stessa destinazione in un determinato arco temporale, e le feature del traffico basate sull'host, costruite come quelle basate sul tempo ma relative invece ad un insieme di un numero specifico di connessioni [2]. Grazie al lavoro di Kayacık et al. [2] è possibile classificare le feature del dataset in base alla loro rilevanza nell'individuazione di diversi tipi di attacchi tramite il calcolo dell'*Information Gain* di ogni feature. Questa operazione è stata svolta anche in questo studio con l'ausilio del software Weka[2]. In particolare, questo valore è stato calcolato sia sul dataset contenente ogni tipo di attacco, sia singolarmente per ogni attacco, come svolto dalla fonte in esame [2]. Gli esiti di queste analisi svolte in autonomia sono riportate nelle Tabelle 4.1 e 4.2, mentre quelli prodotti

[1] `https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset`
[2] `https://www.cs.waikato.ac.nz/ml/weka/`

da Kayacık et al. [2] sono nella Tabella 4.3. A seguito di questi confronti, è stato possibile riscontrare gli stessi risultati per la gran parte degli attacchi, tranne che per alcune tipologie (guess_passwd, teardrop, satan, nmap) di cui la migliore feature riportata da Kayacık et al. [2] risulta essere tra le 4 più influenti, con poca differenza di Information Gain, e per altre (buffer_overflow, loadmodule, ftp_write, phf, multihop, rootkit) le feature calcolate risultano essere diverse, probabilmente a causa dell'esigua popolazione di questi attacchi nel dataset, nell'ordine delle decina di connessioni sul totale di quasi 500000. Tuttavia, sia per rimanere fedeli alle fonti utilizzate dal lavoro in esame [3] sia per l'autorevolezza della fonte originaria [2], sono stati presi in considerazione i risultati elaborati dal lavoro di Kayacık et al. [2], anche poiché la discrepanza tra i risultati ottenuti risulta essere davvero minima e limitata ad attacchi poco rappresentati nel dataset e quindi non in grado di migliorare significativamente le prestazioni del classificatore. In ogni caso, le feature più rilevanti per ogni tipo di attacco sono state prese in considerazione tra le diverse strategie adottate nella realizzazione dell'algoritmo genetico, come descritto nella Sezione 4.4.6.

Il primo esperimento condotto da Al-fuhaidi et al. [3] si focalizza su un sottoinsieme di 18 feature delle 41 totali. Queste sono basate sul sistema operativo di destinazione della connessione (feature usate in Windows) [3], e sono:

{duration, protocol_type, service, flag, src_bytes, dst_bytes, land, wrong_fragment, urgent, count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate} [3], mentre il secondo esperimento utilizza 31 feature sulla base dell'*Information Gain*.

4.3 Strumenti utilizzati

L'algoritmo genetico di questo lavoro è stato realizzato utilizzando il linguaggio di programmazione Java, poiché l'esistenza di molte librerie utilizzabili riduce l'impegno necessario alla creazione di molte componenti dell'algoritmo. La libreria CSVParser[3] è stata utilizzata per estrapolare i dati delle connessioni dal dataset contenuto in un file di formato CSV. Il cuore dell'algoritmo risiede però nella libreria Jenetics[4], che agevola la realizzazione di algoritmi genetici in Java tramite una serie di classi e interfacce.

[3] `https://commons.apache.org/proper/commons-csv/apidocs/org/apache/commons/csv/CSVParser.html`

[4] `https://jenetics.io/`

Information Gain	Numero feature	Nome feature
1.4384029	5	src_bytes
1.3902034	23	count
1.3552348	3	service
1.0977182	24	srv_count
1.0961848	36	dst_host_same_src_port_rate
1.0159265	2	protocol_type
0.9054253	33	dst_host_srv_count
0.8997895	35	dst_host_diff_srv_rate
0.8713826	34	dst_host_same_srv_rate
0.7844647	30	diff_srv_rate
0.7773695	29	same_srv_rate
0.7645689	4	flag
0.5820167	6	dst_bytes
0.5662843	38	dst_host_serror_rate
0.5449353	25	serror_rate
0.5438835	39	dst_host_srv_serror_rate
0.521429	26	srv_serror_rate
0.4364079	12	logged_in

Tabella 4.1: 18 migliori feature per Information Gain calcolate tramite Weka sul dataset contenente tutti gli attacchi

In particolare, questa libreria permette di implementare gli individui degli algoritmi genetici tramite le classi *Population*, *Phenotype*, *Genotype*, *Chromosome* e *Gene*. La libreria mette a disposizione una serie di operatori di crossover, mutazione e selezione già implementati, e permette di creare funzioni di fitness da zero.

4.4 Soluzioni proposte

4.4.1 Classe "Connection"

In primo luogo si è reso necessario codificare le connessioni del dataset in analisi per poterle immagazzinare e utilizzare. Per fare questo è stata creata una classe "*Connection*", contenente come variabili d'istanza ogni feature presente nel dataset, inclusa la categoria

Tipo attacco	Numeri feature	Migliori feature
buffer_overflow	10, 3, 14	hot, service, root_shell
loadmodule	3, 17, 33	service, num_file_creations, dst_host_srv_count
perl	16, 14	num_root, root_shell
neptune	30, 29	diff_srv_rate, same_srv_rate
smurf	5, 23, 24	src_bytes, count, srv_count
guess_passwd	11, 5, 6	num_failed_logins, src_bytes, dst_bytes
pod	5, 8, 3	src_bytes, wrong_fragment, src_bytes
portsweep	4, 35, 41	flag, dst_host_diff_srv_rate, dst_host_srv_rerror_rate
ipsweep	37, 5, 3	dst_host_srv_diff_host_rate, src_bytes, service
land	7, 26, 38	land, srv_serror_rate, dst_host_serror_rate
ftp_write	3, 32, 9	service, dst_host_count, urgent
back	6, 5, 10	dst_bytes, src_bytes, hot
imap	3, 26, 39	service, srv_serror_rate, dst_host_srv_serror_rate
satan	29, 35, 23, 27	same_srv_rate, dst_host_diff_srv_rate, count, rerror_rate
phf	14, 10, 19	root_shell, hot, num_access_files
nmap	5, 3, 4	src_bytes, service, flag
multihop	3, 33, 13	service, dst_host_srv_count, num_compromised
warezmaster	6, 1	dst_bytes, duration
warezclient	5, 3	src_bytes, service
spy	39, 38	dst_host_srv_serror_rate, dst_host_serror_rate
rootkit	3, 33, 13	service, dst_host_srv_count, num_compromised

Tabella 4.2: Migliori feature per Information Gain calcolate tramite Weka per singoli attacchi

di appartenenza ("normal" o il nome dell'attacco). Per semplificare la progettazione degli individui dell'algoritmo genetico, le soluzioni impiegate utilizzano solo valori di tipo Integer, per cui i valori immagazzinati nel dataset sotto forma di stringhe sono stati convertiti in interi tramite l'utilizzo di un apposita struttura dati che associa ad ogni stringa presente nel dataset un valore intero. Anche i valori memorizzati come decimali sono stati trasformati in numeri interi semplicemente moltiplicandoli per 100. L'insieme di *Connection* ottenute estrapolando le informazioni dal file CSV viene conservato in una lista, poi analizzata per valutare le soluzioni.

Tabella 4.3: Lista delle feature per le quali è stata selezionata la classe più rilevante per Kayacık et al. [2]

Classe della connessione	Feature rilevanti
normal	1, 6, 12, 15, 16, 17, 18, 19, 31, 32, 37
smurf (dos)	2, 3, 5, 23, 24, 27, 28, 36, 40, 41
neptune (dos)	4, 25, 26, 29, 30, 33, 34, 35, 38, 39
land (dos)	7
teardrop (dos)	8
ftp_write (r2l)	9
back (dos)	10, 13
guess_pwd (r2l)	11
buffer_overflow (u2r)	14
warezclient (r2l)	22

4.4.2 Codifica degli individui

Il singolo individuo della popolazione è una regola *if(condizione) then{azione}* volta a rilevare connessioni dannose. In particolare, esso rappresenta la *condizione* della regola generata, mentre nella parte *azione* della regola potrebbe essere implementata una contromisura da adottare per contrastare la connessione o a fini statistici e di analisi. Le condizioni relative alle singole feature sono tutte legate dall'operatore logico AND, mentre i versi delle diseguaglianze da verificare per ogni caratteristica sono codificati come parte dell'individuo, quindi soggetti ad evoluzione, tranne per i geni che si riferiscono a feature di tipo categorico, per cui viene utilizzata sempre l'uguaglianza. Ogni individuo consiste di un *Phenotype*, contenente a sua volta un *Genotype*. Un singolo *Genotype* contiene un vettore di *IntegerChromosome*, i cui elementi contengono i geni delle soluzioni, ovvero i veri e propri valori delle feature degli individui, codificati in oggetti della classe *Gene*. I geni necessitano di essere memorizzati in cromosomi differenti in base ai loro valori di minimo e massimo. In particolare, sono state utilizzate 3 differenti codifiche degli individui, ognuna mirata ad individuare una diversa categoria di attacchi e quindi ognuna con un diverso numero di geni, collegati a diverse caratteristiche delle connessioni. Le feature considerate dalle diverse codifiche sono scelte sulla base dell'*Information Gain* calcolato per ogni attacco, come spiegato nella Sezione 4.2.

Di seguito sono riportate le 3 differenti codifiche in dettaglio:

- **Codifica DOS** : La prima soluzione implementata è costituita da 11 cromosomi, di cui 7 contengono esclusivamente un gene poiché questi sono collegati a caratteristiche con valori di minimo e massimo unici rispetto alle altre. Questi valori sono stati estrapolati in seguito ad uno studio dei rispettivi limiti delle caratteristiche nel dataset, in modo da non esplorare regioni dello spazio delle soluzioni potenzialmente irrilevanti. Tuttavia, a causa di questo approccio le regole create in tal modo potrebbero avere difficoltà nel classificare connessioni con valori delle caratteristiche che non rientrano nei limiti estrapolati dal KDDCUP99. Tutti i valori per le feature utilizzate nella prima soluzione proposta sono racchiuse nella Tabella 4.4. Come è possibile notare, le uniche feature che presentano uguali valori sono: wrongFragment e urgent, count e srvCount, e le feature che esprimono valori percentuali. L'ultimo cromosoma contiene i geni relativi ai versi delle disuguaglianze da verificare. Anche in questo caso sono stati utilizzati valori interi, in particolare 1 equivale a $\leq$ mentre 2 a $\geq$. I segni di uguaglianza sono stati tralasciati poiché nella fase iniziale di tentativi non si sono rivelati in grado di ben classificare feature continue e quindi avrebbero rallentato il processo di evoluzione delle regole. Le feature utilizzate nella prima soluzione sono le stesse considerate dalla fonte [3], e permettono di ben classificare gli attacchi di tipo DOS.

- **Codifica Probe** : La seconda codifica implementata si concentra sulla classificazione degli attacchi di categoria Probe, che mirano ad acquisire informazioni potenzialmente sensibili sui dispositivi della rete, ponendosi sia come pericoli per la privacy che per la sicurezza [33]. Grazie alle analisi condotte da Kayacık et al. [2], sono state selezionate le feature più collegate a questi tipi di attacchi, riportate nella Tabella 4.5 insieme ai rispettivi limiti di valori. Per alcune caratteristiche sono stati scelti limiti superiori leggermente maggiori di quelli nel dataset, per poter prevenire la presenza di nuove connessioni con valori diversi. Anche in questo caso si è adottata la stessa strategia per quanto riguarda i versi delle diseguaglianze.

- **Codifica U2R** : Per la terza e ultima codifica implementata è stata adottata la stessa strategia mostrata per la seconda soluzione, ma mirata a trovare attacchi di tipo U2R, poiché risultavano la classe di attacchi meno rappresentata dalla lista di quelli individuati dalle regole prodotte con le precedenti due codifiche. Le feature selezionate e i loro range di valori sono riportati nella Tabella 4.6.

Tabella 4.4: Valori estremi feature prima soluzione

Feature/(Min, Max)	
duration	(0, 58329)
protocolType	(1, 3)
service	(1, 66)
flag	(1, 11)
srcBytes	(0, 999)
dstBytes	(0, 9999)
land	(0, 1)
wrongFragment	(0, 3)
urgent	(0, 3)
count	(0, 99)
srvCount	(0, 99)
serrorRate	(0, 100)
srvSerrorRate	(0, 100)
rerrorRate	(0, 100)
srvRerrorRate	(0, 100)
sameSrvRate	(0, 100)
diffSrvRate	(0, 100)
srvDiffHostRate	(0, 100)

Non si è reso necessario l'utilizzo di un'ulteriore codifica mirata ad individuare gli attacchi di tipo R2L poiché questi risultano sufficientemente individuati dalle codifiche Dos e Probe.

4.4.3 Funzione di fitness

Per la funzione di fitness utilizzata per valutare gli individui generati è stata scelta la misura di performance indicata dal lavoro in analisi [3], la cui formula è riportata di seguito.

$$fitness = \frac{ab}{A} - \frac{a}{B}[3]$$

Le *ab* indicano attacchi correttamente individuati, le *a* indicano le connessioni erroneamente classificate come dannose, mentre A e B corrispondono rispettivamente alla somma degli attacchi e al totale delle connessioni normali, ed il valore risultante varia tra 0 e 1.

Tabella 4.5: Valori estremi feature seconda soluzione

Feature/(Min, Max)	
duration	(0, 58329)
dstBytes	(0,9999)
loggedIn	(0,1)
suAttempted	(0,2)
numShells	(0,2)
numRoot	(0, 993)
numFilesCreations	(0, 28)
numAccessFiles	(0, 8)
srvDiffHostRate	(0, 100)
dstHostSrvDiffHostRate	(0, 100)
rerrorRate	(0, 100)
dstHostCount	(0, 260)
flag	(1, 11)

In particolare, valori maggiori indicano regole con una migliore capacità di individuare minacce nel dataset, per cui la misura di fitness risulta da massimizzare. Il tipo di classificazione che è possibile ottenere con questa tecnica consiste solo in una separazione in attacchi e connessioni normali, senza ottenere una distinzione tra le varie minacce.

Alla luce di ciò, il problema in esame è di tipo mono-obiettivo, poiché la misura da massimizzare risulta essere esclusivamente quella descritta.

4.4.4 Parametri utilizzati

Nello sviluppo dell'algoritmo sono state tentate diverse configurazioni di parametri, quali il numero di individui, il criterio di terminazione, la probabilità di crossover e mutazione e i relativi operatori. Inizialmente i valori scelti si sono rivelati inadeguati al problema in esame, ovvero 100 individui, un limite massimo di 10 minuti di esecuzione, probabilità di mutazione e crossover pari al 30% e all'80%, con SinglePointCrossover e RouletteWheelSelector. In seguito a numerosi tentativi mirati a testare l'efficacia dei diversi parametri, le percentuali legate a mutazione e crossover sono state cambiate al 10% e al 90%, l'operatore di selezione

Tabella 4.6: Valori estremi feature terza soluzione

Feature/(Min, Max)	
duration	(0, 58329)
srcBytes	(0, 999)
dstBytes	(0,9999)
loggedIn	(0,1)
suAttempted	(0,2)
numShells	(0,2)
numRoot	(0, 993)
numFilesCreations	(0, 28)
numAccessFiles	(0, 8)
srvDiffHostRate	(0, 100)
dstHostCount	(0, 260)

utilizzato in definitiva è stato l'EliteSelector, mentre l'operatore di mutazione utilizzato è quello fornito di default dalla libreria Jenetics, che muta i geni degli individui ottenendo nuovi valori per i range prestabiliti, in modo da esplorare meglio le possibili soluzioni garantendo maggiore diversità. Inoltre, la taglia della popolazione di individui è stata innalzata a 1000 e a 5000, in base alla dimensione del dataset di training, con un criterio di arresto fissato a 1000 generazioni. La scelta del nuovo operatore di selezione è stata effettuata per garantire una forma di **elitismo** alla popolazione, ovvero permettere alle soluzioni migliori di sopravvivere alla generazione successiva senza cambiamenti. Questa scelta si è rivelata fondamentale per le prestazioni dell'algoritmo, poiché la generazione di regole capaci di classificare correttamente una o più connessioni è un evento poco probabile data la vastità dello spazio delle soluzioni, e a tal motivo è importante non rischiare di perderle. Questi parametri hanno permesso di trovare soluzioni con ottimi risultati, ma impiegando anche decine di ore per la computazione. Di seguito viene riportata la sezione di codice utilizzata per la creazione delle soluzioni con la prima codifica degli individui presentata. La struttura delle restanti codifiche si diversifica per la costruzione del genotipo, ed è disponibile al seguente link: `https://github.com/davide-lagamba/GAforPrivacy`.

```
final Factory<Genotype<IntegerGene>> GTF = Genotype.of(
      IntegerChromosome.of(0, 58329),//duration (1)
      IntegerChromosome.of(1, 3),//protocolType (2)
      IntegerChromosome.of(1,66),//service (3)
      IntegerChromosome.of(1, 11),//flag (4)
      IntegerChromosome.of(0,999),//srcBytes (5)
      IntegerChromosome.of(0,9999),//dstBytes (6)
      IntegerChromosome.of(0,1),//land (7)
      IntegerChromosome.of(0,3,2),//wrongFragment (8) e urgent (9)
      IntegerChromosome.of(0, 99, 2),//count (23) e srvCount (24)
      IntegerChromosome.of(0, 100, 7),//serrorRate (25),
         srvSerrorRate (26), rerrorRate (27),
      // srvRerrorRate (28), sameSrvRate (29), diffSrvRate (30) e
         srvDiffHostRate (31)
      IntegerChromosome.of(1, 2, 14)//segni disuguaglianze
);

final Engine<IntegerGene, Double> engine;
engine = Engine.builder(
            RunnerConfig10p::fitness,
            GTF).populationSize(population)
      .selector(new EliteSelector<>())
      .optimize(Optimize.MAXIMUM)
      .alterers(
            new Mutator<>(0.1),
            new SinglePointCrossover<>(0.9)
      ).build();

final EvolutionStatistics<Double, ?>
      statistics= EvolutionStatistics.ofNumber();

final Phenotype<IntegerGene, Double> best= engine.stream()
      .limit(generations)
      .peek(statistics)
      .collect(toBestPhenotype());
```

Listing 4.1: Struttura prima codifica degli individui (codifica DOS)

4.4.5 Suddivisione del dataset e fase di validazione

Avendo a disposizione un dataset di grandi dimensioni come il KDDCUP99, è risultato significativo analizzare le differenze nelle prestazioni tra l'utilizzo del dataset parziale, composto dal 10% del totale e scelto dalla fonte in esame [3], ed il dataset intero, composto da circa 5 milioni di connessioni. A seguito di questa suddivisione, risulta importante sottolineare che ogni operazione relativa alla scelta della codifica degli individui descritta nelle Sezioni 4.2 e 4.4.2 è stata effettuata sul dataset KDDCUP99 10%. In particolare, da questa diversificazione nascono le due soluzioni proposte, ovvero la **soluzione 10%** e la **soluzione 100%**. La soluzione 10% utilizza come dataset di training il 10% del dataset KDDCUP99 10%, composto da meno di 50000 connessioni, e la restante parte del dataset per la validazione delle regole ottenute, mentre la soluzione 100% utilizza il KDDCUP99 10%, che corrisponde a circa il 10% dell'intero dataset, per effettuare il training, e le rimanenti connessioni del KDDCUP99 100% per la validazione dei risultati. La prima soluzione risulta essere quindi più specifica e simile a quanto svolto da Al-fuhaidi et al. [3], anche se nel loro lavoro il dataset di validazione presentava una quantità di connessioni circa pari a quelle del dataset di training, mentre in questo caso si è scelto di utilizzare la parte restante dei dati, ovvero il 90% del dataset ridotto. Questo confronto mira quindi a verificare l'importanza delle dimensioni del dataset di training durante l'applicazione dell'algoritmo genetico, in particolare analizzando le differenze nelle misure di prestazioni adottate.

4.4.6 Strategia adottata

Individuare correttamente la quasi totalità degli attacchi, non limitandosi solo agli attacchi DOS come invece scelto dalla fonte [3], risulta improbabile da realizzare con una sola regola o con una sola codifica. Per questo motivo, l'obiettivo è stato quello di realizzare un insieme di regole, anche costituite da codifiche diverse, per poter classificare al meglio le diverse minacce. A tal scopo, per la creazione di ogni regola è stato utilizzato il dataset ridotto composto dai soli attacchi non individuati dall'insieme di regole precedentemente creato. Questa tecnica è stata fondamentale per permettere di individuare attacchi poco presenti nel dataset, che risulta essere pesantemente sbilanciato in favore di attacchi DOS. Oltre a questa strategia, per poter rilevare tipologie specifiche di minacce, è stato anche tentato l'utilizzo di alcuni filtri sul dataset che permettessero di considerare solo le connessioni normali e quel sottoinsieme degli attacchi da voler riconoscere, ma le prestazioni ottenute non si sono rivelate vantaggiose. Un altro aspetto importante da considerare è stato sicuramente l'ordine

di applicazione dei diversi algoritmi con le 3 codifiche riportate nella Sezione 4.4.2. In seguito a diversi tentativi, la combinazione migliore di codifiche da utilizzare in cascata è stata: 11 regole con codifica DOS, 4 regole con codifica Probe e 2 regole con codifica U2R. In particolare è risultata interessante la scelta di applicare prima le codifiche Probe, tuttavia questa strategia portava ad un considerevole aumento dei falsi positivi, per cui è stata in definitiva scartata. Nonostante la caratteristica di sbilanciamento nella composizione del dataset in esame, la corretta identificazione degli attacchi di tipo Denial of Service risulta in ogni caso critica per la sicurezza e la privacy degli utenti di sistemi IoT. Questi attacchi, anche se di piccola entità, vengono infatti spesso utilizzati come distrazione per le reti, con lo scopo di deviare l'attenzione e gli sforzi degli addetti alla sicurezza mentre in realtà un altro attacco, più pericoloso anche per la privacy, è in atto. Spesso anche i pochi secondi di vulnerabilità causati dagli attacchi Dos possono rivelarsi fatali per le intrusioni nelle reti [34]. Anche nella Sezione 2.3.1 è possibile notare come alcuni dei malware più comuni per i dispositivi IoT hanno lo scopo di causare questi tipi di attacchi. La motivazione per l'interesse destinato agli altri tipi di attacchi presenti nel dataset, ovvero gli **R2L** (Remote to Local), **U2R** (User to Root) e **Probe** derivano dalla definizione stessa degli attacchi. La prima categoria è volta ad ottenere permessi di un utente registrato alla macchina senza effettivamente esserlo, la seconda mira ad ottenere i vantaggi di ruoli superiori a quello concesso, e la terza punta ad ottenere dettagli circa i dispositivi connessi, per poi sfruttarne le debolezze [2]. I pericoli legati alla privacy derivanti da questi tipi di attacchi sono ben chiari, poiché ottenere maggiori poteri spesso permette di accedere ad informazioni riservate, e inoltre anche le caratteristiche dei dispositivi potrebbero ricadere nei dati considerati sensibili, oltre che rivelare potenziali debolezze dei sistemi di sicurezza [33].

Una breve e generica panoramica che descrive la metodologia usata per la creazione dell'algoritmo è riportata nella Figura 4.1.

4.4.7 Differenze con il lavoro in esame

Alla luce delle soluzioni descritte, è possibile elencare le differenze tra il lavoro di Alfuhaidi et al. [3] e quanto implementato. Il principale miglioramento nell'algoritmo proposto è che questo non si limita all'individuazione dei soli attacchi DOS ma è esteso ad ogni tipo di attacco presente nel dataset, con particolare attenzione agli attacchi Probe e U2R. La suddivisione in training e validation set è stata approcciata sia in modo molto similare sia diversamente, rispettivamente nelle soluzioni 10% e 100%, allo scopo di confrontare i risultati ottenuti e stabilire la migliore strategia. Anche la scelta delle feature da considerare è stata

modificata, utilizzando sia le 18 feature della prima soluzione proposta da Al-fuhaidi et al. [3] e poi utilizzando anche altre caratteristiche sulla base dell' *Information Gain*. Inoltre, i valori delle caratteristiche non sono stati normalizzati e la funzione di fitness utilizzata non è stata esattamente la stessa della fonte, in particolare non sono stati applicati pesi ai termini della funzione e il valore soglia per la selezione non è stato utilizzato. Queste modifiche sono state attuate perché questi elementi portavano a prestazioni generalmente peggiori dell'algoritmo genetico, rallentando l'evoluzione degli individui. Risulta quindi importante sottolineare che l'unica operazione di pre-processing sui dati effettuata è stata la feature selection, come ben descritto nella Sezione 4.4.2, proprio come nel lavoro in esame.

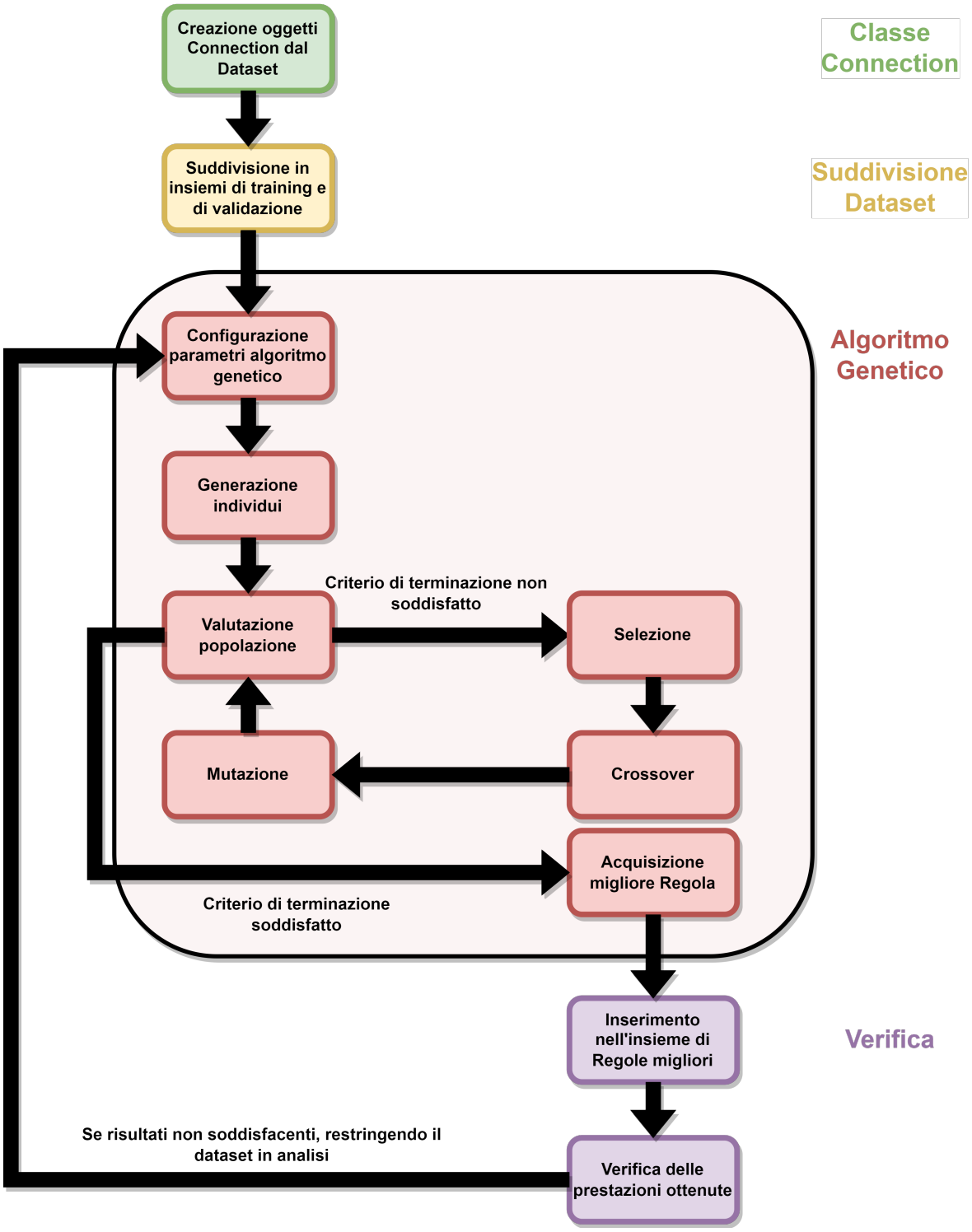


Figura 4.1: Metodologia generale utilizzata per la creazione dell'algoritmo

CAPITOLO 5

Validazione

Dopo aver presentato nel dettaglio le fasi di progettazione dell'algoritmo genetico proposto, risulta fondamentale analizzare i risultati ottenuti per poter trarre delle conclusioni circa l'approccio utilizzato.

5.1 Metodologia di valutazione

5.1.1 Metriche di valutazione

In seguito alla creazione degli insiemi di regole relative alle due soluzioni indicate nella Sezione 4.4.5, sono state impiegate diverse metriche di valutazione per poter analizzare le prestazioni delle regole ottenute. Queste sono state calcolate tramite la **matrice di confusione** derivante dall'impiego di ogni insieme di regole. La matrice indica il numero di connessioni dannose e normali correttamente individuate, rispettivamente chiamate **True Positive (TP)** e **True Negative (TN)**, e quelle incorrettamente classificate come dannose, **False Positive (FP)**, e incorrettamente classificate come normali, **False Negative (FN)** [35]. Da queste misure è stato possibile ricavare delle informazioni più significative, poi utilizzate come criteri per il confronto delle soluzioni proposte con gli altri classificatori. Oltre alla misura di fitness esposta nella Sezione 4.4.3, le altre metriche utilizzate sono l'accuracy, la precisione, la F-Measure, il tasso di rilevamento, la specificity, il tasso di falsi allarmi e il Matthews correlation coefficient (MCC).

- **Accuracy** $= \frac{TP+TN}{TP+FP+FN+TN}$ **[35]** : La percentuale delle connessioni correttamente classificate sul totale, meno rilevante per dataset sbilanciati come il KDDCUP99 [35].
- **Precision** $= \frac{TP}{TP+FP}$ **[35]** : Il rapporto tra le connessioni dannose correttamente classificate e il totale delle connessioni classificate come dannose [35].
- **F-Measure** $= \frac{2*Precision*Detection\ Rate}{Precision+Detection\ Rate}$ **[35]** : La media armonica tra la metrica Precision e il tasso di rilevamento [35].
- **Tasso di rilevamento** $= \frac{TP}{FN+TP}$ **[3]** : Il tasso di attacchi correttamente individutati sul totale degli attacchi del dataset [3].
- **Specificity** $= \frac{TN}{TN+FP}$ **[35]** : Il tasso di connessioni normali correttamente identificate [35].
- **False Alarm** $= \frac{FP}{TN+FP}$ **[36]** : Il tasso di connessioni erroneamente classificate come positive sul totale di connessioni non dannose [36].

- **MCC** $= \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}}$ **[37]** : Valore che varia tra -1 e 1, cresce se la regola classifica correttamente sia le connessioni dannose che quelle normali [37].

Nella creazione delle regole da inserire negli insiemi poi usati per la classificazione, l'obiettivo è stato quello di mantenere alti valori di Accuracy, Precision, Detection Rate, F-Measure ed MCC, mantenendo valori molto bassi di False Alarm. Per fare ciò, sono stati tentati diversi approcci di concatenazione delle regole derivanti dalle diverse codifiche, come esposto nella Sezione 4.4.6. In particolare, si è preferito migliorare il Detection Rate (tasso di rilevamento) anche a costo di ottenere valori di False Alarm leggermente superiori, poiché nel contesto di un ambiente IoT di stampo militare, quindi di importanza critica, per un classificatore con il compito di rilevare attacchi risulta fondamentale scongiurare il maggior numero di pericoli possibili anche a costo di ottenere più Falsi Positivi.

5.1.2 Altri classificatori

A seguito della realizzazione dell'algoritmo genetico descritto nel Capitolo 4, è risultato cruciale confrontare le prestazioni e i risultati ottenuti con altri tipi di classificatori. Il primo confronto è stato effettuato con un classificatore casuale, per assicurare che l'algoritmo genetico sia più efficace di uno che utilizza poche o nessuna informazione specifica dei dati per individuare attacchi. In particolare, sono stati realizzati quattro diversi classificatori casuali, sempre utilizzando il linguaggio Java.

Questi si distinguono per due caratteristiche: effettuare una classificazione generica o specifica rispetto alle categorie di attacchi (Probe, U2R, Dos e R2L) e l'essere pesati o non pesati. I classificatori generici miravano semplicemente a catalogare le connessioni in "attacchi" e non, mentre quelli specifici puntavano anche ad identificare la categoria di attacco. Gli algoritmi pesati effettuavano le scelte casuali in proporzione alle percentuali dei tipi di connessioni nel dataset, quindi avendo delle semplici informazioni effettive sui dati, mentre quelli non pesati sceglievano la categoria da associare ad ogni connessione con la stessa probabilità. Nel caso del classificatore generico non pesato, questo utilizza le stesse proporzioni della controparte specifica, ovvero: con 1 probabilità su 5 classifica una connessione come normale e con la restante probabilità come dannosa, derivante dall'iniziale suddivisione in 5 categorie di connessioni (una normale e le quattro tipologie di attacco); per tal motivo, seleziona con la stessa probabilità una delle 5 classi di connessioni, accomunando le 4 classi di attacchi. Data la casualità prevalente in questi tipi di classificatori, i risultati ottenuti sono relativi ad una media misurata su 100 esecuzioni dell'algoritmo.

Per quanto concerne i classificatori non pesati, questi non hanno necessitato una suddivisione in training e validation set, poiché non utilizzano informazioni specifiche dei dati se non la quantità di diverse categorie degli attacchi, mentre i classificatori specifici sono stati prima "allenati" sul 10% del dataset in esame, al fine di estrarre le percentuali delle categorie di attacchi da poter poi utilizzare sull'insieme di validazione. In questo caso, la suddivisione tra training e validation set è avvenuta casualmente, per cui la partizione delle connessioni che compongono i due insiemi non è esattamente la medesima utilizzata dalla rispettiva soluzione 10% o 100% proposta. Inoltre, la classificazione "specifica" di questi tipi di classificatori non mira a classificare i singoli attacchi poiché l'individuazione più dettagliata delle connessioni dannose sarebbe risultata molto difficoltosa data la quantità di attacchi diversi presenti nel dataset, soprattutto per i classificatori non pesati.

Confronti più rilevanti sono stati invece eseguiti con dei classificatori ben noti ed utilizzati comunemente. In particolare, per fare ciò è stato impiegato il software Weka. Anche in questo caso, i classificatori scelti sono stati allenati sul 10% del dataset scelto e poi validati sul restante 90%, verificandone i risultati al variare delle diverse versioni create del dataset di allenamento e dell'obiettivo da raggiungere: training set con e senza bilanciamento delle classi di connessioni da classificare, mirando ad una classificazione generica o specifica rispetto ai singoli attacchi ed utilizzando tutte le feature del dataset o solo quelle considerate in qualche soluzione dell'algoritmo genetico. Risulta importante far notare che tutte le variazioni, tranne la *feature selection*, sono state eseguite esclusivamente al dataset di training, e non anche all'insieme di validazione, per non influenzare i risultati della fase di validazione. Diversamente rispetto ai classificatori casuali, in questo caso è stato possibile valutare la classificazione specifica rispetto ai singoli attacchi invece che alle loro categorie grazie agli algoritmi messi a disposizione, permettendo di avere quindi maggiori informazioni.

I classificatori scelti per effettuare questi confronti sono stati: ZeroR, Naive Bayes, Decision Table e Random Forest, poiché risultano essere classificatori diversi tra loro e rappresentativi delle loro categorie. Anche il classificatore SMO è stato preso in considerazione poiché impiegato anche da Al-fuhaidi et al. [3], tuttavia le risorse richieste per effettuare il training risultavano eccessivi e i risultati ottenuti su porzioni minori dei dataset erano inferiori rispetto ad altri classificatori come il Random Forest e il Decision Table. Per quanto concerne i confronti con il classificatore Naive Bayes, le precondizioni legate al suo utilizzo non sono state verificate, per cui le prestazioni riportate circa questo classificatore potrebbero essere sottoposte a future analisi e sperimentazioni. I parametri modificabili degli algoritmi impiegati sono stati in ogni caso quelli predefiniti dal software Weka. Inoltre, i confronti

condotti sia per i classificatori casuali che per quelli forniti da Weka sono stati eseguiti sia sul dataset KDDCUP99 10% che sul KDDCUP99 100%, proprio come nell'algoritmo genetico proposto. Oltre che da Al-fuhaidi et al. [3], confronti tra classificatori sono stati presentati anche dalla fonte [33], tuttavia concentrandosi esclusivamente sugli attacchi di tipo Probe piuttosto che come presentato in questo lavoro.

5.2 Risultati dell'algoritmo genetico

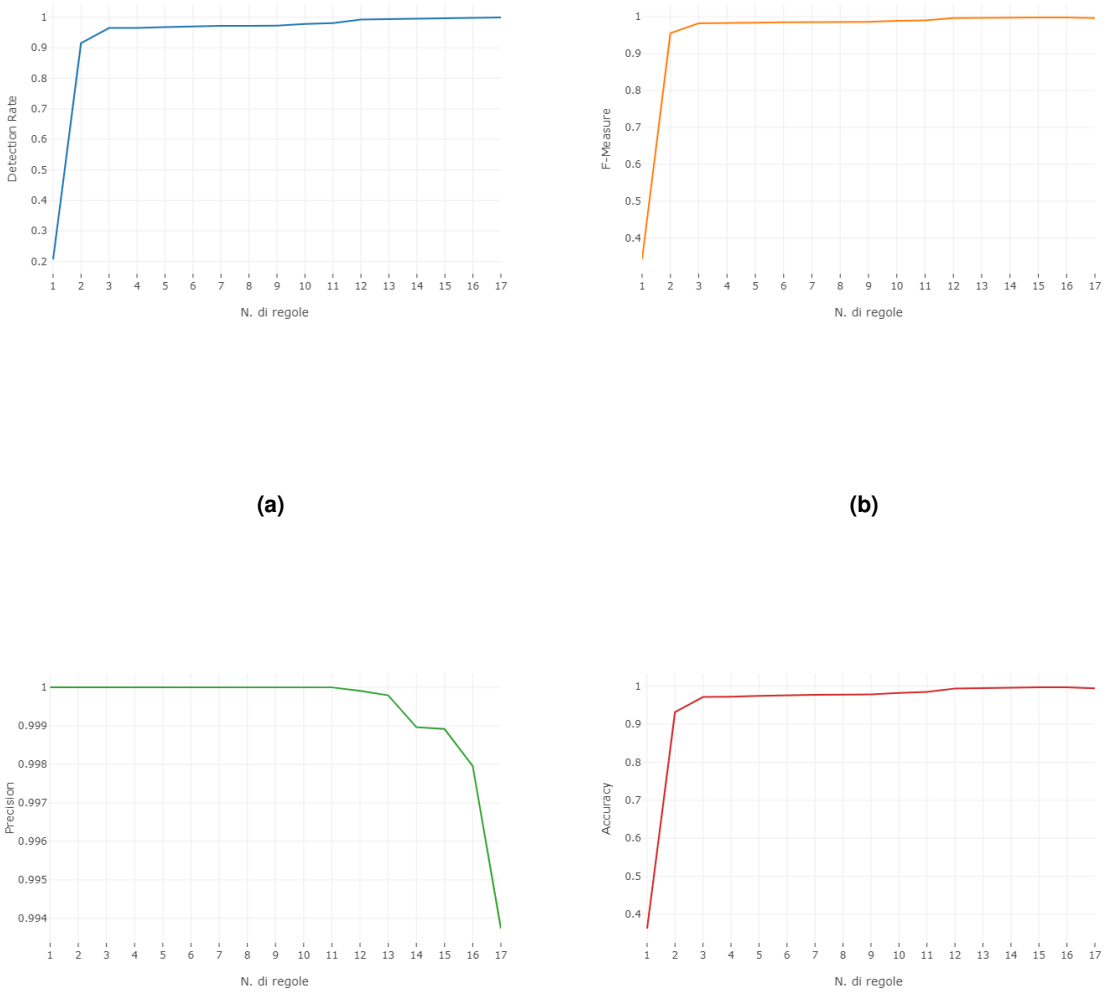
Di seguito sono riportate le Tabelle 5.1 e 5.2, che presentano i valori precisi dei risultati ottenuti, arrotondati per difetto, per le metriche descritte nella Sezione 5.1.1, e i grafici che ne descrivono l'andamento all'aumentare delle regole dell'insieme adoperato per la classificazione generica degli attacchi, sia per la soluzione 10% che per la soluzione 100%.

N. di regole	Detection Rate	F-Measure	Precision	Accuracy	Specificity	MCC	False Alarms
1	0.20662	0.34248	1.0	0.36272	1.0	0.22077	0.0
2	0.91497	0.95560	1.0	0.93170	1.0	0.82414	0.0
3	0.96466	0.98201	1.0	0.97161	1.0	0.91816	0.0
4	0.96500	0.98218	1.0	0.97188	1.0	0.91889	0.0
5	0.96784	0.98365	1.0	0.97416	1.0	0.92494	0.0
6	0.97052	0.98504	1.0	0.97632	1.0	0.930746	0.0
7	0.97172	0.98566	1.0	0.97729	1.0	0.93336	0.0
8	0.97200	0.98580	1.0	0.97751	1.0	0.93397	0.0
9	0.97257	0.98609	1.0	0.97797	1.0	0.93522	0.0
10	0.97771	0.98873	0.99999	0.98209	0.99998	0.94666	0.00001
11	0.98090	0.99035	0.99999	0.98465	0.99998	0.95390	0.00001
12	0.99249	0.99619	0.99990	0.99390	0.99963	0.98109	0.00036
13	0.99439	0.99708	0.99979	0.99533	0.99915	0.98544	0.00084
14	0.99608	0.99752	0.99896	0.99602	0.99578	0.98750	0.00421
15	0.99705	0.99798	0.99891	0.99676	0.99558	0.98981	0.00441
16	0.99822	0.99809	0.99795	0.99693	0.99165	0.99029	0.00834
17	0.99938	0.99655	0.99373	0.99444	0.97427	0.98236	0.02572

Tabella 5.1: Andamento preciso delle statistiche nella soluzione 10% all'aumentare delle regole

N. di regole	Detection Rate	F-Measure	Precision	Accuracy	Specificity	MCC	False Alarms
1	0.00037	0.00075	0.99924	0.19907	0.99999	0.00861	1.14E-6
2	0.20953	0.34646	0.99999	0.36665	0.99999	0.22372	1.14E-6
3	0.20996	0.34706	0.99999	0.36700	0.99999	0.22399	5.71E-6
4	0.26027	0.41304	0.99999	0.40731	0.99999	0.25566	6.85E-6
5	0.26298	0.41644	0.99999	0.40948	0.99999	0.25734	6.85E-6
6	0.26302	0.41650	0.99999	0.40952	0.99999	0.25737	6.85E-6
7	0.26342	0.41700	0.99998	0.40984	0.99998	0.25761	0.00001
8	0.26383	0.41750	0.99998	0.41016	0.99998	0.25786	0.00001
9	0.26444	0.41827	0.99998	0.41065	0.99998	0.25824	0.00001
10	0.26444	0.41827	0.99997	0.41065	0.99997	0.25823	0.00002
11	0.26485	0.41878	0.99997	0.41097	0.99997	0.25848	0.00002
12	0.98090	0.98603	0.99121	0.97773	0.96496	0.93149	0.03503
13	0.99121	0.99120	0.99120	0.98591	0.96454	0.95577	0.03545
14	0.99444	0.99228	0.99014	0.98761	0.96009	0.96091	0.03990
15	0.99639	0.99324	0.99012	0.98914	0.95995	0.96572	0.04004
16	0.99976	0.99420	0.98870	0.99066	0.95397	0.97057	0.04602
17	0.99989	0.98925	0.97884	0.98259	0.91288	0.94500	0.08711

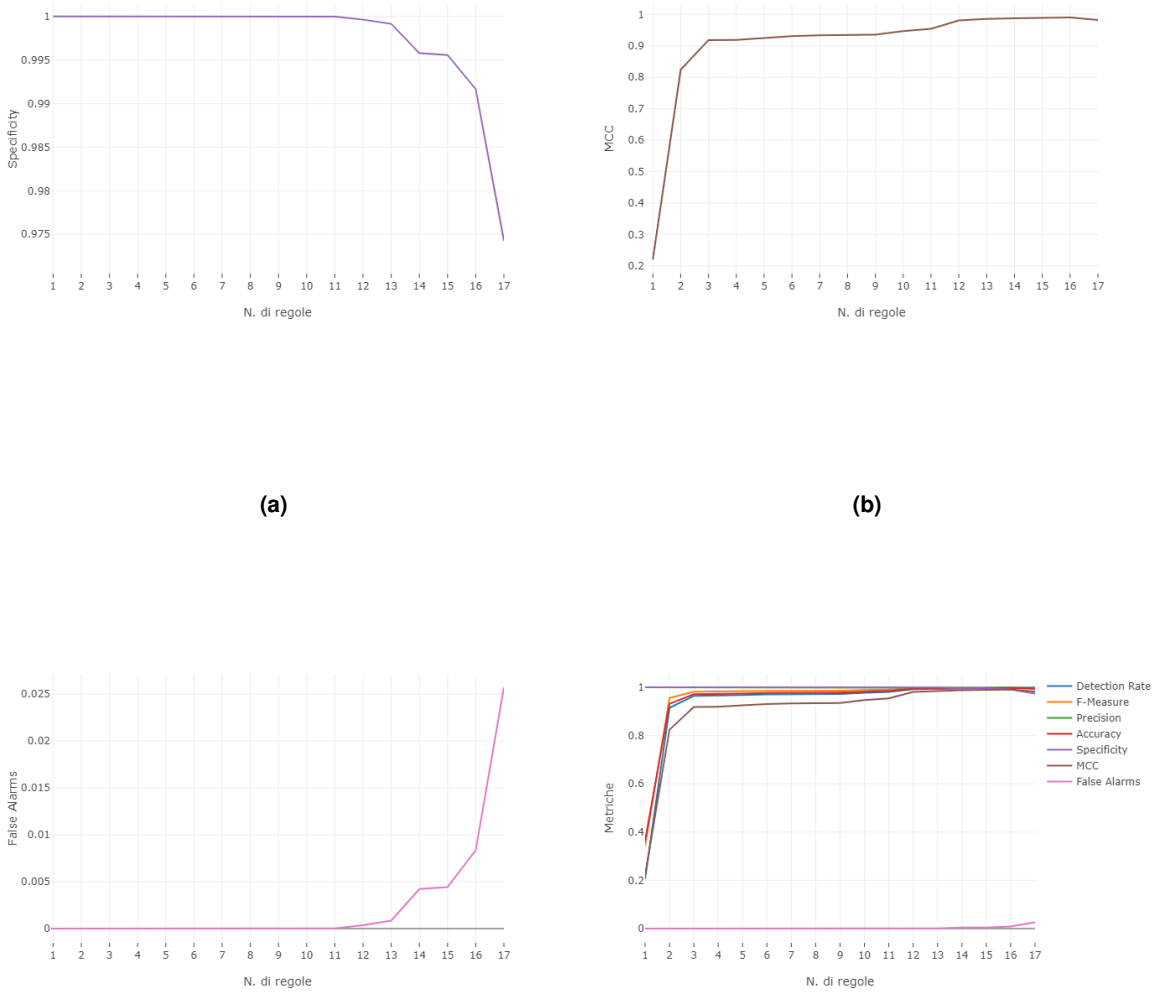
Tabella 5.2: Andamento preciso delle statistiche nella soluzione 100% all'aumentare delle regole



(a) **(b)**

(c) **(d)**

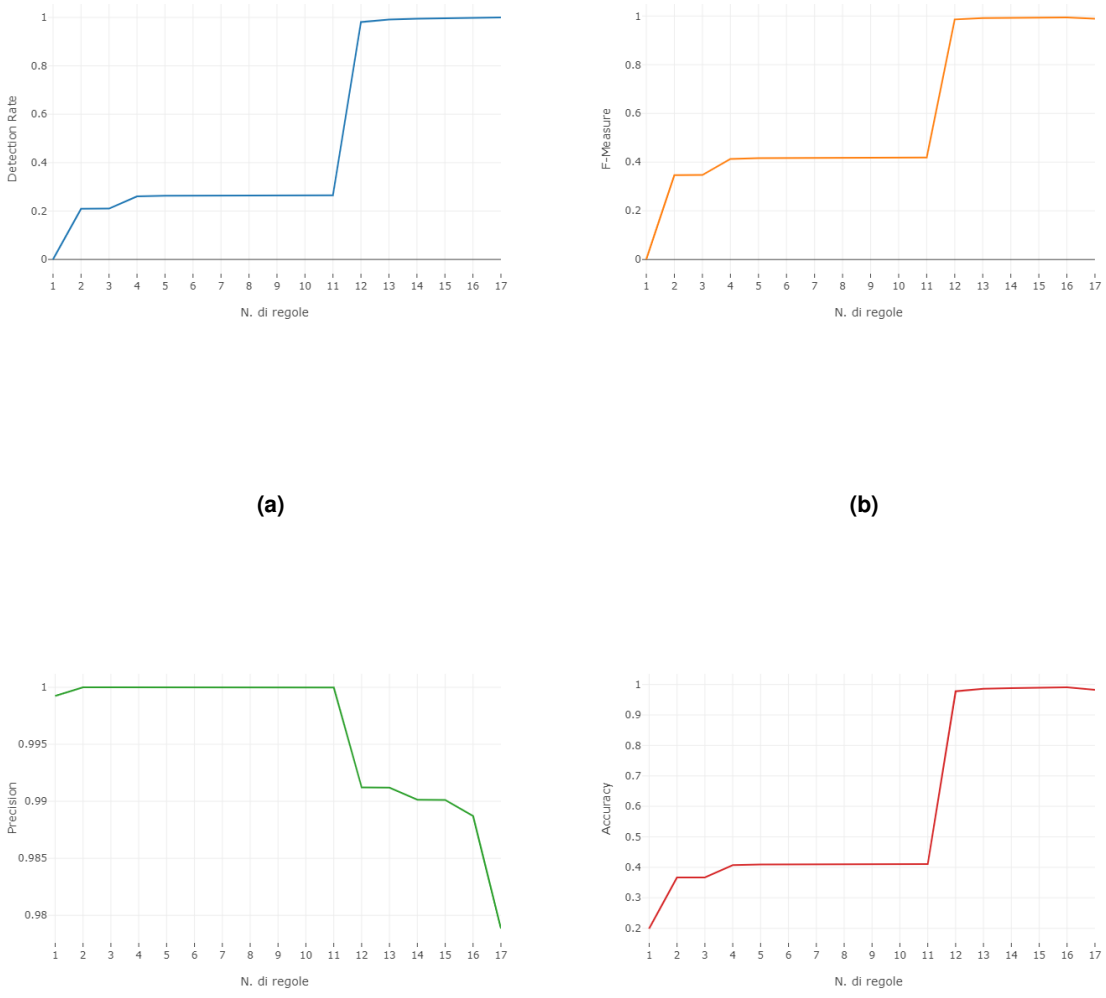
Figura 5.1: Andamento Detection Rate (a), F-Measure (b), Precision (c) e Accuracy (d) nella soluzione 10% all'aumentare delle regole



(a) **(b)**

(c) **(d)**

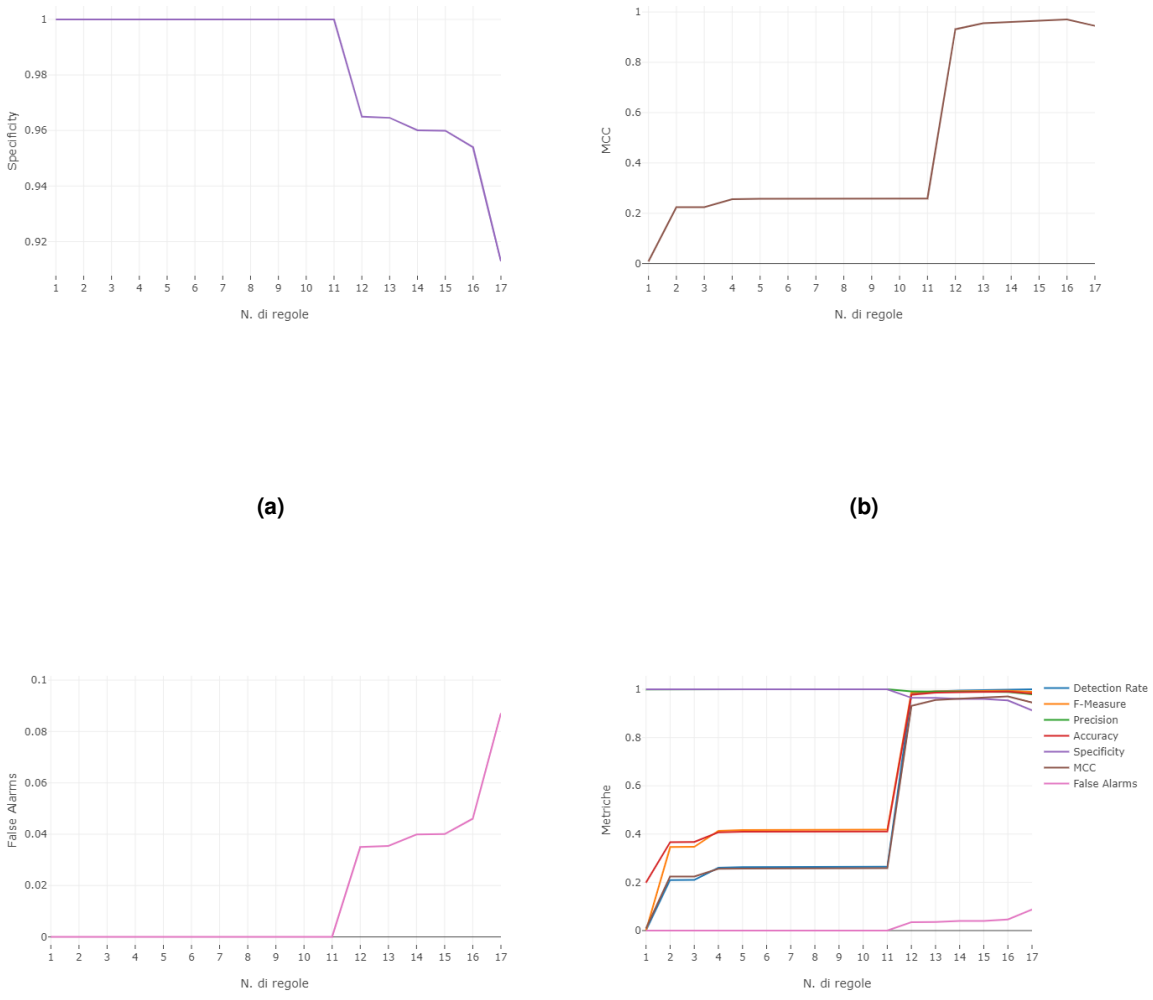
Figura 5.2: Andamento Specificity (a), MCC (b), False Alarms (c) e di ogni metrica (d) nella soluzione 10% all'aumentare delle regole



(a) **(b)**

(c) **(d)**

Figura 5.3: Andamento Detection Rate (a), F-Measure (b), Precision (c) e Accuracy (d) nella soluzione 100% all'aumentare delle regole



(a) **(b)**

(c) **(d)**

Figura 5.4: Andamento Specificity (a), MCC (b), False Alarms (c) e di ogni metrica (d) nella soluzione 100% all'aumentare delle regole

Oltre alle metriche utilizzate per la valutazione dell'insieme di regole, sono riportati di seguito anche i risultati integrali delle matrici di confusione ottenute dalla fase di validazione, mostrati nelle Figure 5.5 e 5.6 e nelle Tabelle 5.3 e 5.4.

N. di regole	True Positive	True Negative	False Positive	False Negative
1	73793	87479	0	283345
2	326773	87479	0	30365
3	344517	87479	0	12621
4	344639	87479	0	12499
5	345653	87479	0	11485
6	346612	87479	0	10526
7	347041	87479	0	10097
8	347140	87479	0	9998
9	347344	87479	0	9794
10	349180	87478	1	7958
11	350317	87478	1	6821
12	354459	87447	32	2679
13	355136	87405	74	2002
14	355739	87110	369	1399
15	356087	87093	386	1051
16	356505	86749	730	633
17	356917	85229	2250	221

Tabella 5.3: Andamento preciso delle matrici di confusione della soluzione 10% all'aumentare delle regole

Le statistiche relative alle singole tipologie di attacchi presenti nel dataset sono mostrate nelle Tabelle 5.5 e 5.6. Questi risultati sono stati prodotti filtrando il dataset di training e di validazione in esame, al fine di contenere esclusivamente le connessioni normali e quelle relative alla tipologia di attacco da considerare. Anche in questo caso, i risultati sono stati riportati arrotondandoli per difetto, ed è possibile notare come nell'insieme di validazione della soluzione 100% non è stato possibile riportare risultati relativi agli attacchi U2R e R2L, proprio a causa della mancanza di essi nel dataset di validazione utilizzato, ricavato dal KDDCUP99 100% eliminando le connessioni presenti nel KDDCUP99 10%, utilizzato come insieme di training.

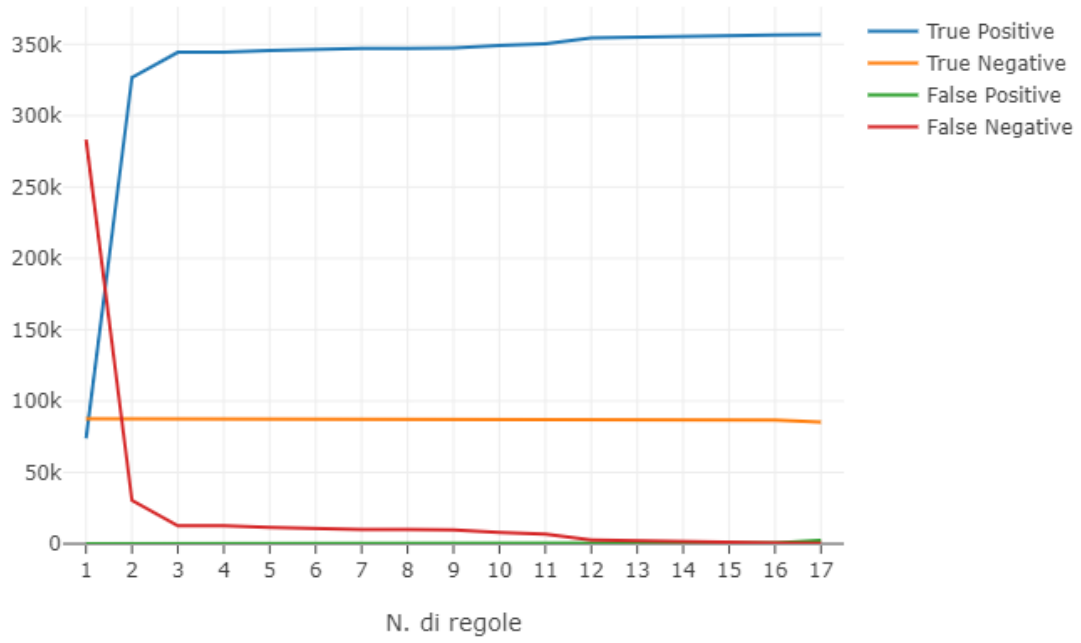


Figura 5.5: Andamento della matrice di confusione nella soluzione 10% all'aumentare delle regole

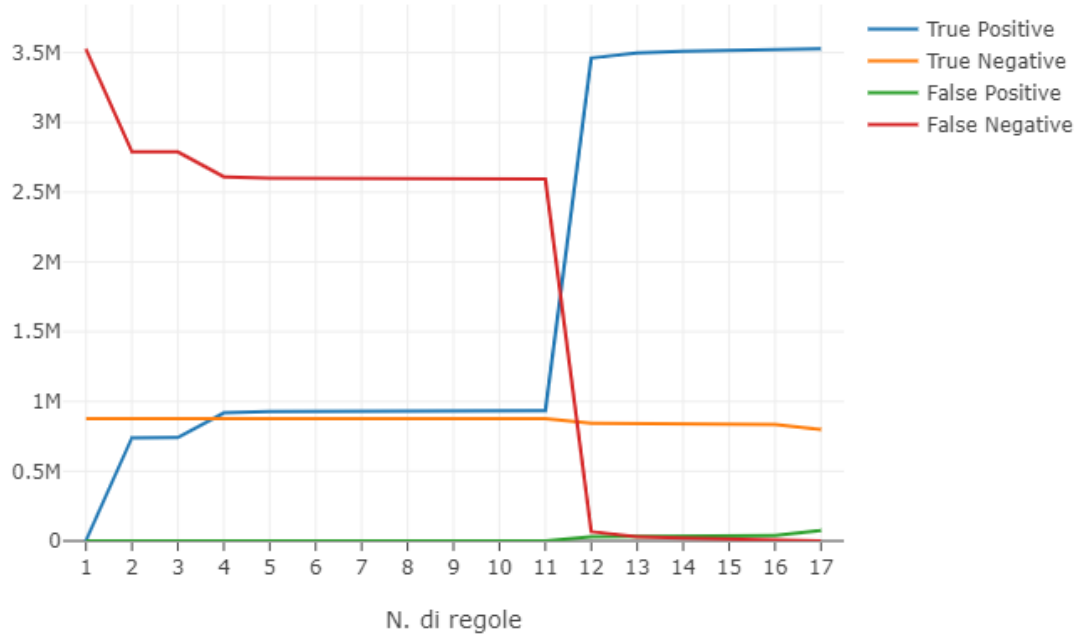


Figura 5.6: Andamento della matrice di confusione nella soluzione 100% all'aumentare delle regole

N. di regole	True Positive	True Negative	False Positive	False Negative
1	1327	875503	1	3527580
2	739412	875503	1	2789495
3	740950	875499	5	2787957
4	918476	875498	6	2610431
5	928033	875498	6	2600874
6	928207	875498	6	2600700
7	929615	875492	12	2599292
8	931036	875492	12	2597871
9	933192	875492	12	2595715
10	933192	875482	22	2595715
11	934636	875482	22	2594271
12	3461525	844827	30677	67382
13	3497906	844464	31040	31001
14	3509293	840567	34937	19614
15	3516176	840443	35061	12731
16	3528068	835208	40296	839
17	3528525	799237	76267	382

Tabella 5.4: Andamento preciso delle matrici di confusione della soluzione 100% all'aumentare delle regole

In seguito alla descrizione dei risultati ottenuti, è possibile iniziare ora la comparazione delle prestazioni dell'algoritmo genetico in fase di validazione con i modelli di classificazione casuale testati. Risulta da evidenziare come le tecniche più simili come strategia all'algoritmo in esame siano quelle di tipo generico; per tal motivo, le Tabelle relative ai confronti con i classificatori di tipo specifico sono raccolte nel repository GitHub, al seguente link: `https://github.com/davide-lagamba/GAforPrivacy`. Anche in questo caso sono riportati i risultati relativi ad entrambe le soluzioni proposte, con i risultati approssimati per difetto.

Come prevedibile, è possibile notare dalle Tabelle 5.7 e 5.8 e da quelle riportate nel repository GitHub che l'algoritmo genetico implementato, in entrambe le soluzioni proposte, ha prestazioni nettamente migliori dei classificatori casuali presentati. I valori forniti da Weka e riportati nelle Tabelle si riferiscono alle medie pesate delle metriche rispetto alle diverse classi di attacchi. Sono analizzate ora le prestazioni dei classificatori di Weka elencati nella Sezione

Metrica	Dos	Probe	U2R	R2L
Training Accuracy	0.99509	0.97605	0.97521	0.97547
Training Detection Rate	1.0	0.98976	0.57142	0.97247
Training False Alarms	0.02449	0.02449	0.02449	0.02449
Training Precision	0.99389	0.61722	0.01639	0.30635
Training Specificity	0.97550	0.97550	0.97550	0.97550
Training MCC	0.98465	0.77161	0.09377	0.53863
Training F-Measure	0.99694	0.76031	0.03187	0.46593
Testing Accuracy	0.99483	0.97480	0.97389	0.97327
Testing Detection Rate	0.99993	0.98708	0.22222	0.88692
Testing False Alarms	0.02572	0.02572	0.02572	0.02572
Testing Precision	0.99365	0.61980	0.00442	0.28616
Testing Specificity	0.97427	0.97427	0.97427	0.97427
Testing MCC	0.98375	0.77156	0.02808	0.49526
Testing F-Measure	0.99678	0.76146	0.00867	0.43271

Tabella 5.5: Statistiche relative alle diverse tipologie attacchi della soluzione 10%

5.1.2 e confrontati i risultati disponibili con quelli derivanti dalla validazione dell'algoritmo genetico. Notiamo inoltre che i risultati per alcune metriche non sono disponibili in quanto, per alcune categorie, i valori restituiti dalla matrice di confusione sono uguali a 0, impedendo di calcolare determinate metriche. Per questo motivo, nei classificatori specifici è possibile confrontare esclusivamente il Detection Rate. Questo deriva dalla casualità con cui avviene la suddivisione negli insiemi di training e validation: data la bassa popolazione di alcuni tipi di attacchi, questi potrebbero essere presenti solo in uno dei due insiemi e quindi non è possibile classificarli tutti per poi calcolare un valore medio. Per questo motivo, nei confronti tra le soluzioni proposte e quelle implementate in Weka le partizioni delle connessioni presenti negli insiemi di training e validazione non sono le medesime, tuttavia sono mantenute le proporzioni delle dimensioni degli insiemi.

Anche in questo caso, le configurazioni più simili a quelle utilizzate per l'algoritmo genetico sono quelle dei classificatori che utilizzano solo una parte delle feature del dataset sbilanciato per classificare attacchi generici. Tuttavia, allo scopo di analizzare meglio tutte le possibili tecniche, sono riportati anche i confronti tra i classificatori composti dalle

Metrica	Dos	Probe	U2R	R2L
Training Accuracy	0.98289	0.91718	0.91386	0.91408
Training Detection Rate	0.99999	0.99074	0.51923	0.91474
Training False Alarms	0.08591	0.08591	0.08591	0.08591
Training Precision	0.97909	0.32743	0.00322	0.10971
Training Specificity	0.91408	0.91408	0.91408	0.91408
Training MCC	0.94602	0.54395	0.03568	0.30006
Training F-Measure	0.98943	0.49219	0.00640	0.19592
Testing Accuracy	0.98253	0.91600	0.91288	0.91288
Testing Detection Rate	0.99999	0.98970	N/A	N/A
Testing False Alarms	0.08711	0.08711	0.08711	0.08711
Testing Precision	0.97862	0.32435	0.0	0.0
Testing Specificity	0.91288	0.91288	0.91288	0.91288
Testing MCC	0.94518	0.54067	N/A	N/A
Testing F-Measure	0.98919	0.48859	0.0	0.0

Tabella 5.6: Statistiche relative alle diverse tipologie attacchi della soluzione 100%

diverse configurazioni e l'algoritmo proposto, che invece presenta sempre la medesima configurazione illustrata nel Capitolo 4. Inoltre, per la medesima motivazione riguardante i confronti con i classificatori casuali, le Tabelle relative alle comparazioni con i classificatori specifici implementati in Weka sono disponibile nel repository GitHub, al seguente link: `https://github.com/davide-lagamba/GAforPrivacy`.

Metrica	Alg. Gen. 10%	Class. generico casuale 10%	Class. generico pesato 10%
Accuracy	0.99444	0.68187	0.68370
Detection Rate	0.99938	0.80003	0.80302
False Alarms	0.02572	0.80005	0.80296
Precision	0.99373	0.80308	0.80311
Specificity	0.97427	0.19994	0.19703
MCC	0.98236	-0.00001	0.00006
F-Measure	0.99655	0.80156	0.80307

Tabella 5.7: Confronto statistiche soluzione 10% algoritmo genetico con classificatori generici casuali

Metrica	Alg. Gen. 100%	Class. gener. casuale 100%	Class. gener. pesato 100%
Accuracy	0.98259	0.68084	0.68169
Detection Rate	0.99989	0.79997	0.80136
False Alarms	0.08711	0.79993	0.80128
Precision	0.97884	0.80141	0.80143
Specificity	0.91288	0.20006	0.19871
MCC	0.94500	0.00004	0.00008
F-Measure	0.98925	0.80069	0.80140

Tabella 5.8: Confronto statistiche soluzione 100% algoritmo genetico con classificatori generici casuali

Metrica	Alg. Genetico 10%	ZeroR	Naive Bayes	Decision Table	Random Forest
Precision	0.993	N/A	0,984	0,998	1,000
Det. Rate	0.999	0,803	0,984	0,998	1,000
F-Measure	0.996	N/A	0,984	0,998	1,000
MCC	0.982	N/A	0,950	0,995	0,998

Tabella 5.9: Confronto statistiche soluzione 10% algoritmo genetico con classificatori Weka che considerano le stesse feature dell'algoritmo genetico, generici, su dataset non bilanciato

Metrica	Alg. Genetico 10%	ZeroR	Naive Bayes	Decision Table	Random Forest
Precision	0.993	N/A	0,985	0,997	0,999
Det. Rate	0.999	0,803	0,984	0,997	0,999
F-Measure	0.996	N/A	0,984	0,997	0,999
MCC	0.982	N/A	0,951	0,990	0,998

Tabella 5.10: Confronto statistiche soluzione 10% algoritmo genetico con classificatori Weka che considerano le stesse feature dell'algoritmo genetico, generici, su dataset bilanciato

Metrica	Alg. Genetico 10%	ZeroR	Naive Bayes	Decision Table	Random Forest
Precision	0.993	N/A	0,985	0,998	1,000
Det. Rate	0.999	0,803	0,984	0,998	1,000
F-Measure	0.996	N/A	0,984	0,998	1,000
MCC	0.982	N/A	0,951	0,995	0,999

Tabella 5.11: Confronto statistiche soluzione 10% algoritmo genetico con classificatori Weka che considerano tutte le feature, generici, su dataset non bilanciato

Metrica	Alg. Genetico 10%	ZeroR	Naive Bayes	Decision Table	Random Forest
Precision	0.993	N/A	0,985	0,997	1,000
Det. Rate	0.999	0,803	0,984	0,997	1,000
F-Measure	0.996	N/A	0,984	0,997	1,000
MCC	0.982	N/A	0,950	0,992	0,998

Tabella 5.12: Confronto statistiche soluzione 10% algoritmo genetico con classificatori Weka che considerano tutte le feature, generici, su dataset bilanciato

Metrica	Alg. Genetico 100%	ZeroR	Naive Bayes	Decision Table	Random Forest
Precision	0.978	N/A	0,992	1,000	1,000
Det. Rate	0.999	0,801	0,992	1,000	1,000
F-Measure	0.989	N/A	0,992	1,000	1,000
MCC	0.945	N/A	0,975	0,999	1,000

Tabella 5.13: Confronto statistiche soluzione 100% algoritmo genetico con classificatori Weka che considerano le stesse feature dell'algoritmo genetico, generici, su dataset non bilanciato

Metrica	Alg. Genetico 100%	ZeroR	Naive Bayes	Decision Table	Random Forest
Precision	0.978	N/A	0,992	0,999	1,000
Det. Rate	0.999	0,801	0,992	0,999	1,000
F-Measure	0.989	N/A	0,992	0,999	1,000
MCC	0.945	N/A	0,975	0,997	1,000

Tabella 5.14: Confronto statistiche soluzione 100% algoritmo genetico con classificatori Weka che considerano le stesse feature dell'algoritmo genetico, generici, su dataset bilanciato

Metrica	Alg. Genetico 100%	ZeroR	Naive Bayes	Decision Table	Random Forest
Precision	0.978	N/A	0,990	0,999	1,000
Det. Rate	0.999	0,801	0,990	0,999	1,000
F-Measure	0.989	N/A	0,990	0,999	1,000
MCC	0.945	N/A	0,969	0,998	1,000

Tabella 5.15: Confronto statistiche soluzione 100% algoritmo genetico con classificatori Weka che considerano tutte le feature, generici, su dataset non bilanciato

Metrica	Alg. Genetico 100%	ZeroR	Naive Bayes	Decision Table	Random Forest
Precision	0.978	N/A	0,990	0,999	1,000
Det. Rate	0.999	0,801	0,990	0,999	1,000
F-Measure	0.989	N/A	0,990	0,999	1,000
MCC	0.945	N/A	0,969	0,997	1,000

Tabella 5.16: Confronto statistiche soluzione 100% algoritmo genetico con classificatori Weka che considerano tutte le feature, generici, su dataset bilanciato

CAPITOLO 6

Discussione e conclusioni

Al termine della presentazione e della esaustiva fase di validazione dell'algoritmo elaborato, è ora possibile trarre delle conclusioni circa la sua efficacia e discutere dei possibili sviluppi futuri.

6.1 Discussione e conclusioni sulle prestazioni dell'algoritmo

6.1.1 Confronto tra le soluzioni proposte

Come primo aspetto, è analizzabile la differenza di risultati ottenuti tra le due soluzioni proposte: la soluzione 100% risulta essere leggermente superiore in termini di Detection Rate rispetto alla controparte 10%, presentando tuttavia prestazioni peggiori nelle restanti metriche. Nonostante ciò, proprio grazie al tasso di rilevamento migliorato, la soluzione 100% potrebbe risultare la più efficace da integrare in un sistema reale, poiché permetterebbe generalmente di classificare una percentuale maggiore di attacchi, seppur segnalando più frequentemente delle connessioni normali erroneamente come pericolose. Tuttavia, come mostrato nella Tabella 5.6, alcuni attacchi non erano presenti nell'insieme di testing della Soluzione 100%, per cui le prestazioni nei confronti degli attacchi U2R e R2L sono deducibili esclusivamente dai risultati sull'insieme di training.

Analizziamo ora anche le prestazioni riscontrate in fase di testing rispetto al lavoro di Al-fuhaidi et al. [3] nella Tabella 6.1.

Metrica	Alg. Gen. 100%	Alg. Gen. 10%	Alg. Gen. di Al-fuhaidi et al. [3]
Testing Accuracy	98.259%	99.444%	99.971%
Detection Rate	99.989%	99.938%	99.87%
False Alarm	08.711%	02.572%	0.003%

Tabella 6.1: Confronto statistiche algoritmo genetico presentato con algoritmo genetico di Al-fuhaidi et al. [3]

Come è possibile notare, entrambe le soluzioni proposte riescono nell'intento di migliorare il Detection Rate ottenuto dal lavoro di partenza, aumentando anche la varietà di minacce rilevate, come descritto nella Sezione 4.4.7. Tuttavia, proprio a causa dei diversi obiettivi prefissati in questo lavoro, le altre metriche di valutazione risultano essere leggermente peggiorate. Le motivazioni dei risultati così ottenuti potrebbero risiedere nei diversi approcci alla progettazione dell'algoritmo, in particolare alla concatenazione delle regole ottenute tramite diverse codifiche e alla restrizione del dataset di partenza a seguito della genera-

zione di ogni regola di classificazione. Attraverso le Tabelle 5.5 e 5.6 è possibile analizzare nel dettaglio le prestazioni dell'algoritmo proposto nei confronti delle diverse tipologie di attacchi. Le migliorie apportate risultano notevoli soprattutto rispetto agli attacchi Probe, di grande interesse per le problematiche di privacy, ma anche per gli attacchi R2L. Queste sono probabilmente derivanti dalle diverse codifiche utilizzate e quindi dalla fase di feature selection. In merito agli attacchi U2R invece è possibile notare come i risultati ottenuti non siano ai livelli delle altre tipologie di attacchi, ciò a causa della esigua popolazione di questi all'interno del dataset, nell'ordine delle decine sul totale delle connessioni presenti. Per tal motivo, come già descritto, nell'insieme di validazione utilizzato per la soluzione 100% non è stato possibile analizzare le prestazioni per gli attacchi di questo tipo, poiché non presenti. Nonostante l'utilizzo di una codifica degli individui mirata proprio a classificare attacchi U2R, l'inserimento delle regole generate in questo modo innalzano notevolmente la quantità di falsi positivi, come è possibile notare dai risultati ottenuti.

A seguito dell'analisi dell'algoritmo proposto, è possibile anche mostrare come sono strutturate alcune delle regole generate.

Regola prodotta con codifica DOS nella soluzione 10%:

$if(duration <= 5855\ AND\ protocolType == icmp\ AND\ service == ecr\_i\ AND\ flag == SF\ AND\ srcBytes >= 462\ AND\ dstBytes <= 8249\ AND\ land == 0\ AND\ wrongFragment <= 2\ AND\ urgent <= 0\ AND\ count >= 1\ AND\ srvCount >= 1\ AND\ serrorRate <= 0.81\ AND\ srvSerrorRate <= 0.27\ AND\ rerrorRate <= 0.88\ AND\ srvRerrorRate <= 0.31\ AND\ sameSrvRate <= 1.00\ AND\ diffSrvRate <= 0.90\ AND\ srvDiffHostRate <= 1.00)\{attacco\ trovato\}$

Struttura del genotipo della regola:

$$[5855, 3, 10, 1, 462, 8249, 0, 2, 0, 1, 1, 81, 27, 88, 31, 100, 90, 100, 1, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1]$$

Regola prodotta con codifica Probe nella soluzione 10%:

$if(duration <= 3290\ AND\ dstBytes <= 5389\ AND\ loggedIn >= 0\ AND\ suAttempted <= 2\ AND\ numShells <= 0\ AND\ numRoot <= 184\ AND\ numFilesCreations <= 2\ AND\ numAccessFiles <= 6\ AND\ srvDiffHostRate <= 1.00\ AND\ dstHostSrvDiffHostRate <= 0.53\ AND\ rerrorRate <= 1.00\ AND\ dstHostCount >= 13\ AND\ flag == S0)\{attacco\ trovato\}$

Struttura del genotipo della regola:

$$[3290, 5389, 0, 2, 0, 184, 2, 6, 100, 53, 100, 13, 5, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2]$$

Regola prodotta con codifica U2R nella soluzione 10%:

$if(duration <= 26427\ AND\ srcBytes <= 6\ AND\ dstBytes <= 2731\ AND\ loggedIn <= 1\ AND\ suAttempted <= 0\ AND\ numShells <= 1\ AND\ numRoot <= 648\ AND\ numFilesCreations <= 4\ AND\ numAccessFiles <= 2\ AND\ srvDiffHostRate <= 57\ AND\ dstHostSrvDiffHostRate <= 2)\{attacco\ trovato\}$

Struttura del genotipo della regola:

$$[26427, 6, 2731, 1, 0, 1, 648, 4, 2, 57, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

Le restanti regole prodotte sono riportate nella repository GitHub `https://github.com/davide-lagamba/GAforPrivacy`.

Come è possibile notare dalle regole prodotte, alcuni parametri si sono rivelati essere ininfluenti dati i segni delle disuguaglianze generate dall'evoluzione caratteristica dell'algoritmo. Per tal motivo, un possibile sviluppo futuro potrebbe essere esplorare approcci diversi alla scelta delle feature rilevanti rispetto a quelli presentati.

6.1.2 Confronto con i classificatori casuali

Attraverso il confronto con classificatori di tipo casuale, è possibile stabilire se l'algoritmo implementato garantisca prestazioni che giustifichino le fasi di studio, progettazione e codifica. Difatti si denota una grande discrepanza tra le prestazioni dei classificatori proposti. In particolare, dalle Tabelle 5.7 e 5.8 e da quelle riportate nel repository GitHub si evince la bontà dell'algoritmo genetico rispetto ad un classificatore che non utilizza particolari informazioni circa la struttura degli elementi da classificare.

6.1.3 Confronto con i classificatori tradizionali

Nella Sezione 5.2 sono riportati i confronti dell'algoritmo genetico sviluppato con i classificatori tradizionali presenti in Weka. Analizzando le comparazioni sia con le configurazioni più simili degli algoritmi tradizionali sia più differenti rispetto all'algoritmo genetico, di cui alcune consultabili nel repository GitHub, è possibile giudicare positivamente le prestazioni delle soluzioni proposte, soprattutto in termini di Detection Rate. In particolare, il tasso di rilevamento ottenuto risulta generalmente migliore rispetto agli algoritmi ZeroR e NaiveBayes, ed ha prestazioni simili a Decision Table e Random Forest. Tuttavia quest'ultimo è considerabile leggermente più efficace dalle comparazioni proposte.

Di notevole importanza risultano anche i confronti tra le diverse configurazioni utilizzate per valutare i risultati degli algoritmi tradizionali: generalmente le prestazioni migliori sono

state ottenute dagli algoritmi applicati con la medesima configurazione adottata dall'algoritmo genetico, per cui le diverse possibilità di pre-processing sui dati esplorate in questa fase non risultano avere un grande impatto sulle prestazioni della classificazione. Anche in merito alla scissione in classificatori generici e specifici, i risultati migliori sono stati ottenuti da quelli generici, come possibile prevedere.

6.2 Conclusioni finali e proposte di sviluppi futuri

Al termine di queste analisi, è possibile discutere dei pro e dei contro relativi all'impiego degli algoritmi genetici per la classificazione. Analizzando esclusivamente le prestazioni ottenute, queste risultano molto simili a quelle ottenibili dai migliori classificatori tradizionali; tuttavia l'implementazione dell'algoritmo genetico necessita di una fase di progettazione più lunga e ponderata, soprattutto a causa della selezione delle codifiche e della funzione di fitness. D'altro canto, le regole generate dall'algoritmo proposto risultano facilmente comprensibili, a differenza dei criteri utilizzati da molti classificatori tradizionali, fattore che potrebbe risultare fondamentale da massimizzare in alcuni ambiti di ricerca. Tra i possibili sviluppi futuri sarebbe degno di nota implementare un algoritmo genetico che invece mira ad effettuare una classificazione specifica degli attacchi, con lo scopo di confrontarne i risultati con quanto ottenuto. Inoltre, l'implementazione presentata risulta fortemente legata alla struttura del dataset KDDCUP99, per cui potrebbe essere utile sviluppare una soluzione più generica, da poter poi adottare anche nel mondo reale per valutarne le prestazioni. Infine, potrebbe essere esplorato l'utilizzo delle funzioni multi-obiettivo per poter generare regole di classificazione capaci di individuare diverse categorie di attacchi secondo più criteri.

Ringraziamenti

In conclusione di questo lavoro, ritengo doveroso ringraziare tutti i Docenti che mi hanno accompagnato in questo percorso di laurea, e in particolare il Professore Fabio Palomba per la competenza, il supporto, la dedizione e la disponibilità fornite, così come il Dottor Giammaria Giordano. Ritengo doveroso aggiungere che il Professor Palomba ha rappresentato per me un punto di riferimento fondamentale in questa esperienza.

Ringrazio i miei genitori, Sara e Nello, mio fratello Alessandro, i miei nonni e i miei zii, poiché sono sempre stati un modello da seguire.

Questo percorso universitario mi ha anche permesso di conoscere colleghi, diventati poi amici, che mi hanno molto supportato da un punto di vista umano; per questo ringrazio anche loro.

Ringrazio infine anche le persone conosciute al di fuori di questo ambiente, come Maria Martina, Rebecca, Mirko, Marco e Fabrizio, per essere sempre stati al mio fianco ed avermi sostenuto.

Bibliografia

[1] U. Albalawi, "A comprehensive analysis on intrusion detection in iot based smart environments using machine learning approaches," *International Journal of Scientific & Technology Research*, vol. 9, pp. 1646–1652, 2020. (Citato alle pagine vi e 10)

[2] G. Kayacık, A. Zincir-Heywood, and M. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on kdd 99.," 01 2005. (Citato alle pagine vi, 30, 31, 34, 35 e 41)

[3] B. Al-fuhaidi, I. Abd-Alghafar, G. Salama, and A. Abd-Alhafez, "Performance evaluation of a genetic algorithm based approach to network intrusion detection system," 05 2009. (Citato alle pagine vii, 3, 4, 26, 27, 28, 30, 31, 35, 36, 40, 41, 42, 45, 47, 48 e 63)

[4] L. Strous, S. von Solms, and A. Zúquete, "Security and privacy of the internet of things," *Computers & Security*, vol. 102, p. 102148, 2021. (Citato a pagina 2)

[5] B. Jovanovic, "Internet of things statistics for 2022 - taking things apart." `https://dataprot.net/statistics/iot-statistics/`, 2022. (Citato a pagina 2)

[6] G. Giordano, F. Palomba, and F. Ferrucci, "Smart detection and preservation of privacy concerns in iot systems: A systematic literature review," *Available at SSRN 3979390*, 2021. (Citato alle pagine 2, 6, 7, 9, 10 e 30)

[7] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, pp. 65–85, Jun 1994. (Citato alle pagine 2 e 3)

[8] M. U. Hassan, M. H. Rehmani, and J. Chen, "Privacy preservation in blockchain based iot systems: Integration issues, prospects, challenges, and future research directions,"

Future Generation Computer Systems, vol. 97, pp. 512–529, 2019. (Citato alle pagine 11, 12, 13, 14 e 15)

[9] T. Ahmed Teli, F. Masoodi, and R. Yousuf, "Security concerns and privacy preservation in blockchain based iot systems: Opportunities and challenges," 2020. (Citato alle pagine 11, 12 e 13)

[10] M. Mirkin, Y. Ji, J. Pang, A. Klages-Mundt, I. Eyal, and A. Juels, *BDoS: Blockchain Denial-of-Service*, p. 601–619. New York, NY, USA: Association for Computing Machinery, 2020. (Citato a pagina 14)

[11] A. Z. Junejo, M. A. Hashmani, and A. A. Alabdulatif, "A survey on privacy vulnerabilities in permissionless blockchains," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 9, 2020. (Citato alle pagine 15 e 16)

[12] B. Putz and G. Pernul, "Detecting blockchain security threats," in *2020 IEEE International Conference on Blockchain (Blockchain)*, pp. 313–320, 2020. (Citato a pagina 16)

[13] P. Sandner, R. Joas, and J. Gross, "Blockchain, iot and ai — a perfect fit." https://philippsandner.medium.com/blockchain-iot-and-ai-a-perfect-fit-c863c0761b6#:~:text=A%20possible%20connection%20between%20these,full%20potential%20if%20applied%20combined., 2020. (Citato a pagina 16)

[14] C. Beek, "Mirai botnet creates army of iot orcs." https://www.mcafee.com/blogs/other-blogs/mcafee-labs/mirai-botnet-creates-army-iot-orcs/, 2017. (Citato a pagina 17)

[15] S. Stahie, "New iot botnet finds open telnet ports and brute-forces entry and installation." https://www.bitdefender.com/blog/hotforsecurity/new-iot-botnet-finds-open-telnet-ports-brute-forces-entry-installation, 2020. (Citato a pagina 18)

[16] T. Luft, "Stories from the soc -ssh brute force authentication attempt tactic." https://cybersecurity.att.com/blogs/security-essentials/stories-from-the-soc-ssh-brute-force-authentication-attempt-tactic, 2021. (Citato a pagina 18)

[17] C. Osborne, "Unityminer cryptocurrency malware hijacks qnap storage devices." https://www.zdnet.com/article/

unityminer-cryptocurrency-malware-hijacks-qnap-storage-devices/, 2021. (Citato a pagina 18)

[18] PandaSecurity, "'ghost push' malware threatens android users." https://www.pandasecurity.com/en/mediacenter/mobile-security/ghost-push-malware-android/, 2017. (Citato a pagina 18)

[19] J. Rossignol, "What you need to know about ios malware xcodeghost." https://www.macrumors.com/2015/09/20/xcodeghost-chinese-malware-faq/, 2015. (Citato a pagina 18)

[20] W. Largent, "New vpnfilter malware targets at least 500k networking devices worldwide." https://blog.talosintelligence.com/2018/05/VPNFilter.html, 2018. (Citato a pagina 18)

[21] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu, and X. Fu, "Security vulnerabilities of internet of things: A case study of the smart plug system," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 05 2017. (Citato a pagina 18)

[22] M. Conti, E. Sandeep Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3416–3452, 2018. (Citato a pagina 19)

[23] C. X. Lu, Y. Li, Y. Xiangli, and Z. Li, "Nowhere to hide: Cross-modal identity leakage between biometrics and devices," *Proceedings of The Web Conference 2020*, Apr 2020. (Citato a pagina 20)

[24] A. W. Marashdih, Z. Zaaba, and H. Khalid, "Web security: Detection of cross site scripting in php web application using genetic algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 8, 01 2017. (Citato a pagina 23)

[25] F. Duchene, S. Rawat, J.-L. Richier, and R. Groz, "Kameleonfuzz: Evolutionary fuzzing for black-box xss detection," in *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy*, CODASPY '14, (New York, NY, USA), p. 37–48, Association for Computing Machinery, 2014. (Citato a pagina 23)

[26] S. Noreen, S. Murtaza, M. Shafiq, and M. Farooq, "Evolvable malware.," pp. 1569–1576, 01 2009. (Citato a pagina 24)

[27] G. Stein, B. Chen, A. S. Wu, and K. A. Hua, "Decision tree classifier for network intrusion detection with ga-based feature selection," in *ACM-SE 43*, 2005. (Citato alle pagine 24 e 25)

[28] W. Li, "Using genetic algorithm for network intrusion detection," *Proc. United States Department of Energy Cyber Security Group 2004 Training Conference, Kansas City, Kansas*, May 24-27 2004. (Citato a pagina 25)

[29] S. Paliwal and R. Gupta, "Article: Denial-of-service, probing & remote to user (r2l) attack detection using genetic algorithm," *International Journal of Computer Applications*, vol. 60, pp. 57–62, December 2012. Full text available. (Citato alle pagine 25 e 26)

[30] Y. Zhang, P. Li, and X. Wang, "Intrusion detection for iot based on improved genetic algorithm and deep belief network," *IEEE Access*, vol. 7, pp. 31711–31722, 2019. (Citato a pagina 26)

[31] N. Klaokliang, P. Teawtim, P. Aimtongkham, C. So-In, and A. Niruntasukrat, "A novel iot authorization architecture on hyperledger fabric with optimal consensus using genetic algorithm," in *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, pp. 1–5, 2018. (Citato a pagina 26)

[32] Information and C. S. U. of California Irvine, "Kdd cup 1999 data." `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, 1999. (Citato alle pagine 27 e 30)

[33] C. Ambedkar and V. K. Babu, "Detection of probe attacks using machine learning techniques," *International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)*, vol. 2, no. 3, pp. 25–29, 2015. (Citato alle pagine 35, 41 e 48)

[34] J. Reo, "How can a ddos attack be part of a security breach?." `https://www.corero.com/blog/how-can-a-ddos-attack-be-part-of-a-security-breach/`, 2017. (Citato a pagina 41)

[35] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015. (Citato a pagina 45)

[36] H. Om and A. Kundu, "A hybrid system for reducing the false alarm rate of anomaly intrusion detection system," in *2012 1st international conference on recent advances in information technology (RAIT)*, pp. 131–136, IEEE, 2012. (Citato a pagina 45)

[37] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020. (Citato a pagina 46)