Progettazione e sviluppo di sistemi di rilevamento delle intrusioni resilienti agli attacchi avversari

Tesi di Laurea Magistrale in Ingegneria Informatica

Francesco Cucinella

Relatore: Ing. Pierluca Ferraro

Correlatori: Ing. Vincenzo Agate

UNIVERSITÀ DEGLI STUDI DI PALERMO
DIPARTIMENTO DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

PROGETTAZIONE E SVILUPPO DI SISTEMI DI RILEVAMENTO DELLE INTRUSIONI RESILIENTI AGLI ATTACCHI AVVERSARI

TESI DI LAUREA DI
FRANCESCO CUCINELLA

RELATORE
Prof. PIERLUCA FERRARO

CORRELATORE
Prof. VINCENZO AGATE

Sommario

I sistemi di rilevamento delle intrusioni (*IDS - Intrusion Detection Systems*) basati su algoritmi di *Machine Learning (ML)* e *Deep Learning (DL)*, hanno acquisito negli ultimi anni una sempre crescente popolarità. Questi sistemi utilizzano tecniche avanzate per rilevare attività sospette o malevole, attraverso l'analisi di grandi quantità di dati. La loro elevata efficacia, comparata ad altre tecniche tradizionali, si scontra tuttavia con un'intrinseca vulnerabilità rispetto a quelli che vengono definiti Attacchi Avversari. L'obiettivo di questa classe di attacchi è quello di modificare l'input sottoposto agli algoritmi *ML/DL-based*, in modo tale da condurre ad un errore nella corretta rilevazione dell'input stesso. Il presente studio propone quindi la progettazione e realizzazione di un *Network IDS DL-based* che sia robusto rispetto agli Attachi Avversari e che, allo stesso tempo, garantisca un'adeguata accuratezza nella corretta individuazione di input non avversari.

Indice

Introduzione

Negli ultimi anni si è assistito ad una crescita esponenziale della complessità delle reti di computer. La ricerca in questo campo ha prodotto protocolli sempre più sofisticati volti a gestire tale complessità e a favorire una comunicazione efficace attraverso la rete. Ciò di fatto ha aperto alla possibilità di usufruire di numerosi servizi online, precedentemente accessibili solo fisicamente (es. *home banking*, *e-commerce*), con una conseguente crescita della quantità di dati che quotidianamente vengono prodotti e trasferiti attraverso la rete. In molti casi si tratta di dati sensibili che necessitano di essere adeguatamente protetti per garantire la privacy degli utenti e limitare la possibilità di accesso da parte di terzi non autorizzati. Nella ormai maggior parte dei casi, inoltre, i sistemi informatici rappresentano il fulcro delle attività operative di un'organizzazione, per cui eventuali malfunzionamenti, dolosi o accidentali, possono rappresentare un serio fattore di rischio per la continuità operativa e per il business delle organizzazioni. La *cyber sicurezza* mira quindi a mettere in atto un insieme di meccanismi volti a proteggere i sistemi informatici da minacce relative alla confidenzialità, integrità o disponibilità degli stessi. Una delle soluzioni in tal senso è rappresentata dagli **Intrusion Detection Systems (IDS)**, il cui obiettivo è quello di rilevare eventuali accessi non autorizzati, favorendo la possibilità di agire tempestivamente per rimuovere o limitare la minaccia. Questi sistemi possono essere implementati a livello di singolo host (*Host-based IDS*) o possono proteggere più host all'interno di una rete (*Network-based IDS*). Tipicamente si predilige, specie per reti in larga scala, l'utilizzo di *Network-based IDS*, i quali monitorano lo stato dell'intera rete ai fini del rilevamento delle intrusioni, tenendo in considerazione vari parametri derivanti dall'analisi del traffico di rete [1]. Il rilevamento delle intrusioni può essere ottenuto sulla base di diversi paradigmi, tipicamente distinguibili in due grandi tipologie: *classification-based* e *anomaly-based*. Nel primo caso si fa riferimento a tutti quei sistemi che determinano la presenza di intrusioni sulla base di corrispondenze con situazioni note. Risultano quindi rilevabili dall'osservazione di attività anomale simili di cui si ha una conoscenza a priori. Nel secondo caso, l'obiettivo è invece quello di individuare

situazioni che si discostano da quello che dovrebbe essere il normale comportamento del sistema. Il rilevamento di tipo *anomaly-based* risulta più efficace in presenza di attacchi di tipo *zero-day*, ovvero attacchi che sfruttano una o più vulnerabilità del sistema target non note in precedenza. Nel caso *classification-based* si ha un'alta accuratezza nell'individuazione di attacchi già noti ma, di contro, una bassa capacità di rilevamento di attacchi di tipo *zero-day*. Tendenze recenti nella realizzazione di *IDS* prediligono l'adozione di algoritmi di **Deep Learning** ai fini del rilevamento delle intrusioni. Tali algoritmi consentono infatti di trattare dati in grandi quantità, anche caratterizzati da elevate non linearità [2], favorendo un alto tasso di successo nella predizione dei campioni osservati. L'efficacia di questi algoritmi si scontra tuttavia con una loro vulnerabilità intrinseca rispetto a quelli che vengono definiti **Attacchi Avversari**. Si tratta di attacchi il cui obiettivo è quello di perturbare opportunamente l'input da sottoporre al modello *ML/DL-based*, al fine di causare una predizione errata da parte dell'algoritmo [3]. La presenza di tale vulnerabilità ha portato a valutare diversi metodi per limitarla e gestirla, specie in un ambito sensibile alla tematica di sicurezza come quello del rilevamento di intrusioni [4]. La maggior parte delle tecniche di difesa nei confronti di attacchi avversari, è stata incentrata per diverso tempo nel dominio della *Computer Vision (CV)* o del *Natural Language Processing (NLP)*, i quali rappresentano i primi ambiti di applicazione degli algoritmi di *Deep Learning*. Studi in ambito *CV* hanno dimostrato come la generazione di esempi avversarsi nei confronti di un determinato modello di *DL* o *ML*, dia luogo ad esempi avversari validi anche per un altro modello. Questa proprietà viene definita **trasferibilità avversaria** ed è il principio su cui si basano diversi attacchi nei quali l'attaccante non ha alcuna conoscenza del modello target, ovvero scenari *Black-Box (BB)* [5]. Le tecniche di difesa dagli attacchi avversarsi in ambito *CV* o *NLP* non sono però, in genere, direttamente trasferibili nel dominio degli *NIDS*. La trattazione di immagini o sequenze testuali presenta infatti caratteristiche differenti rispetto alla trattazione di sequenze di pacchetti scambiati in rete. Per questo motivo, sono state definite nel tempo tecniche di difesa ad hoc che agiscono direttamente nel dominio degli *NIDS* e, in altri casi, alcune tecniche di difesa esistenti per altri domini sono state adattate per l'applicazione anche in ambito *NIDS*. Nel presente lavoro di tesi, l'obiettivo sarà quello di progettare e realizzare un *NIDS DL-based* robusto rispetto agli attacchi avversari, che risulti altresì efficace nella corretta rilevazione di intrusioni. A tal fine, verrà proposta ed implementata un'architettura basata su modelli di *Deep Learning*, ai quali verranno affiancate delle tecniche di difesa volte a limitare e contrastare l'effetto di attacchi avversari rivolti al sistema.

Capitolo 1

Stato dell'Arte e Nozioni fondamentali

In questo capitolo verranno descritti gli aspetti correlati ai sistemi di rilevamento delle intrusioni, approfondendo ulteriormente la tematica degli attacchi avversari e delle relative difese. Verranno dapprima descritti in linea generale i concetti base correlati al presente studio, quali le caratteristiche dei tipici *NIDS ML/DL-based* e degli attacchi avversari che possono agire su di essi. Verrà descritta in maggior dettaglio la differenza tra attacchi *White-Box* ed attacchi *Black-Box*, presentando alcuni dei principali attacchi riconducibili a ciascuna delle due classi. Verranno infine discusse le principali tecniche di difesa adoperate per contrastare la presenza di input avversari.

1.1 Concetti base

Un *NIDS* monitora lo stato della rete al fine di identificare la presenza di accessi malevoli non autorizzati e, opzionalmente, categorizzare la tipologia di minaccia in atto. Si possono distinguere diversi paradigmi per la rilevazione delle intrusioni, i quali sono riconducibili tipicamente a due differenti tipologie: *Classificazione del Traffico di Rete* e *Rilevamento delle Anomalie*. Nelle sezioni successive verranno discusse le caratteristiche che contraddistinguono i due approcci, a cui seguirà una descrizione della problematica legata agli attacchi avversari cui questi sistemi sono soggetti in ambito *ML* e *DL*.

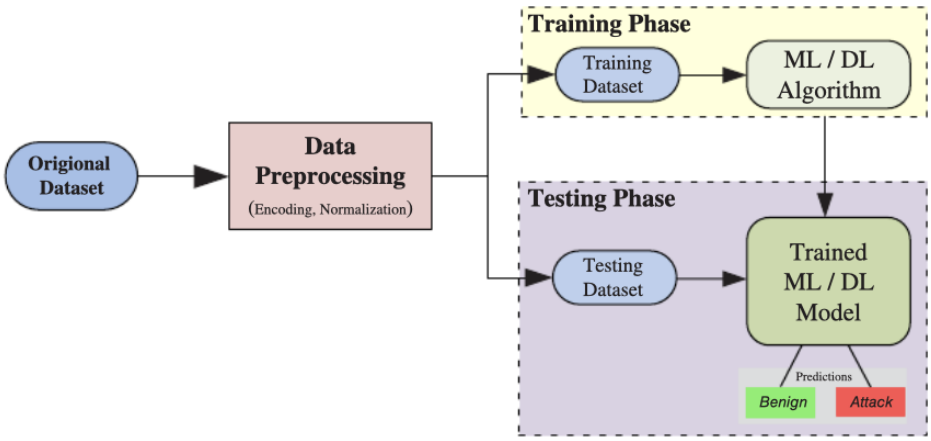

Figura 1.1: Metodologia generica di implementazione di NIDS ML/DL-based. Immagine tratta da [6]

1.1.1 Classification-based NIDS

L'obiettivo di questa famiglia di *NIDS* è individuare e classificare un attacco in atto nella rete. Tipicamente questi sistemi sono basati su tecniche di *Deep Learning* [7], il cui obiettivo è quello di apprendere un modello che, a partire da un certo input, consenta di determinare se questo sia riconducibile ad un possibile attacco noto (e di che tipo). Data la relazione di sequenzialità che lega i dati nel dominio del traffico di rete (ordine dei pacchetti), ai fini del rilevamento alcune tecniche tengono in considerazione questa proprietà, basandosi ad esempio su meccanismi come le *Recurrent Neural Networks (RNN)* [8]. Il vantaggio del paradigma di classificazione è quello di poter individuare, oltre che la presenza di un attacco, anche la sua tipologia, il che agevola nella scelta delle contromisure da intraprendere. Si tratta anche di una tecnica caratterizzata da un basso tasso di falsi positivi, in quanto tipicamente basata sull'addestramento di modelli con dati relativi ad attacchi noti. Risulta però, per lo stesso motivo, vulnerabile rispetto ad attacchi di tipo *zero-day* e, in aggiunta, non sempre è possibile ottenere dataset con un numero adeguato di dati etichettati.

1.1.2 Anomaly-based NIDS

In questo genere di approccio, l'obiettivo è quello di identificare situazioni che si discostano dalle normali attività di rete [9]. È dunque fondamentale in questo caso riuscire a tracciare un profilo di normalità adeguato del traffico di rete, in quanto questo rappresenterà la base di

confronto rispetto ad attività ritenute sospette. Anche per l'implementazione di questo paradigma possono essere adottate tecniche di *Deep Learning*, tipicamente basate sull'utilizzo di *Autoencoders*: un encoder ed un decoder vengono addestrati su dati relativi a profili benigni del traffico di rete, in modo tale da massimizzare l'errore di ricostruzione in corrispondenza di input malevoli [10]. Le tecniche di rilevamento delle intrusioni basate sulla rilevazione delle anomalie sono più adatte per l'individuazione di attacchi di tipo *zero-day* e, inoltre, non sempre necessitano di dati etichettati per la fase di addestramento. Di contro, non è sempre immediato riuscire a definire una nozione generica di 'normale' attività di rete, in quanto questa può dipendere da caso a caso [11].

1.1.3 Caratteristiche di un attacco avversario

Dal punto di vista dell'attaccante, l'obiettivo è tipicamente quello di violare proprietà come l'integrità, la confidenzialità o la disponibilità di risorse all'interno di un sistema informatico. Nel caso di sistemi protetti da un *NIDS ML/DL-based*, l'attacco deve essere tale da riuscire a bypassare tale protezione, per cui il traffico di rete generato deve apparire come benigno pur portando a termine l'obiettivo dell'attacco. Il traffico di rete viene dunque opportunamente modificato, attraverso delle perturbazioni impercettibili, in modo tale da causare la classificazione errata da parte dell'algoritmo *ML/DL-based*. Si parla in questo caso di **attacchi avversari**, dei quali verranno di seguito discusse le caratteristiche fondamentali e le principali tecniche adoperate per metterli in atto. Formalmente, un attacco avversario può essere modellato come un problema di mapping che, a partire dall'input originale, genera l'input avversario. Tipicamente, la funzione di mapping corrisponde ad una somma vettoriale tra l'input originale e la perturbazione che consente di generare l'input avversario. Da [12] si ha:

$$A : \mathbb{R}^d \rightarrow \mathbb{R}^d \; tale \; che \; \vec{x_0}' = A(\vec{x_0}). \; Tipicamente : \vec{x_0}' = \vec{x_0} + \vec{r} \tag{1.1}$$

Il problema si riconduce dunque alla ricerca del vettore di perturbazione $\vec{r}$ da sommare all'input originale $\vec{x_0}$, tale da ottenere un input avversario $\vec{x_0}'$ che il modello target classificherà erroneamente. Nel caso di attacchi avversari **non targettizzati**, l'obiettivo dell'attacco è far in modo che il modello classifichi l'input in una qualsiasi classe purché diversa rispetto alla classe reale. Nel caso di attacchi **targettizzati**, invece, l'obiettivo è fare in modo che l'input avversario venga classificato in una classe specifica, differente rispetto alla classe reale. In entrambi i casi, la

ricerca del vettore di perturbazione ottimo comprende tipicamente due vincoli:

- **Vincolo di confidenza**: nel caso di attacco avversario targettizzato, il vincolo prevede che la confidenza della classe target C_t a cui si intende far ricondurre l'esempio avversario, venga massimizzata rispetto a tutte le altre possibili classi. Formalmente, supponendo un problema di classificazione multi-classe, con classi C_1,... ,C_k e supponendo che la funzione $g_i()$ modelli la confidenza della classe i, il vincolo di confidenza può essere espresso come segue:

$$\max_{j \neq t} \{g_j(\vec{x_0}')\} - g_t(\vec{x_0}') \leq 0 \tag{1.2}$$

- **Vincolo di similarità**: in base a questo vincolo, la perturbazione applicata all'input deve essere tale da garantire che l'input perturbato risulti simile rispetto all'input originale, sulla base di una determinata misura di similarità. Considerando un vincolo di similarità basato sulla norma *Lp* calcolata tra i due input, è possibile modellare il vincolo come segue:

$$\|\vec{x_0}' - \vec{x_0}\| < \varepsilon \tag{1.3}$$

Nel definire un attacco avversario, è necessario dunque tenere in considerazione entrambi i vincoli, al fine di ottenere degli esempi avversari validi e che riescano a raggirare con successo il modello target. A seconda del vincolo a cui viene associata maggiore importanza è possibile distinguere attacchi di tipo **Minimum Norm** (prediligere vincolo di similarità), **Maximum Allowable** (prediligere vincolo di confidenza), **Regularazation** (bilanciamento tra i due vincoli) [13].

La definizione di esempi avversari è inoltre soggetta a vincoli **sintattici** e **semantici**. Nel primo caso, si tratta di far sì che le perturbazioni che vengono applicate all'input garantiscano comunque la validità sintattica del traffico di rete (es. un campo numerico che accetta valori in un range specifico). Nel caso di vincoli semantici, si intende fare in modo che la perturbazione non precluda il raggiungimento dell'obiettivo dell'attacco, dunque l'evasione dei sistemi di rilevazione non deve andare in contrasto con la natura malevola dell'attacco stesso.

1.1.4 Modalità esecuzione attacchi avversari

La generazione di esempi avversarsi che risultino realistici è un problema complesso che può essere affrontato secondo diverse tecniche di manipolazione dell'input originale. Da questo punto di vista è possibile distinguere due principali modalità di manipolazione dell'input:

- **Flow-based**: una delle alternative è quella di agire direttamente sulle *features* utilizzate per classificare l'input. Ciascuna variabile viene opportunamente modificata in modo da ottenere un esempio avversario da sottoporre al modello target. Pur trattandosi di una tecnica semplice, risulta tuttavia poco adatta a casi reali per via della non reversibilità che tipicamente caratterizza i processi di estrazione delle *features* a partire dai dati grezzi.

- **Packet-based**: tecniche più recenti di generazione di esempi avversari si basano invece sulla manipolazione diretta dei **dati grezzi**. In questo caso le modifiche avvengono a livello di pacchetto, il che da luogo ad esempi avversari realistici che possono essere direttamente sottoposti ai sistemi target [14] [15].

Oltre alla modalità di manipolazione dell'input, è possibile distinguere le tecniche adoperate per mettere in atto un attacco avversario sulla base della **conoscenza** che l'attaccante possiede circa il modello target. Distinguiamo anche in questo caso due principali possibili alternative:

- **White Box (WB)**: in questo caso l'attaccante possiede completa conoscenza del modello target, dunque ne conosce parametri ed iperparametri (es. nel caso di un modello basato su reti neurali possono essere funzione di costo, numero di layers della rete, numero di neuroni per ciascun layer, algoritmo di ottimizzazione). Rappresenta lo scenario più semplice in cui può essere attuato un attacco avversario, in quanto l'attaccante può sfruttare la conoscenza sul modello al fine di generare input avversari validi ed efficaci.

- **Black Box (BB)**: in scenari più realistici, l'attaccante possiede una conoscenza limitata o nulla circa il modello target. Non ne conosce dunque parametri o iperparametri e l'obiettivo è, tipicamente, quello di stimare un modello surrogato a partire dal quale verranno generati input avversari da sottoporre successivamente al modello target. Viene sfruttata in questo caso la proprietà di trasferibilità [16], secondo la quale gli esempi avversari generati per un modello risultano avversari anche per un altro modello, anche se quest'ultimo presenta una struttura differente.

In base a come questi aspetti si combinano tra loro, si ottengono dunque diversi possibili metodi di generazione di input avversari, con differente efficacia ed ambito di applicazione.

1.2 Lavori correlati

La tematica riguardante gli attacchi avversari aventi come target modelli di *ML* o *DL* è stata affrontata in diversi studi, sin da quando è stata osservata la vulnerabilità che tali modelli posseggono, per loro natura, rispetto ad input avversari[3]. I principali studi in questo ambito si sono inizialmente focalizzati sull'analisi di attacchi avversari nel dominio della *Computer Vision (CV)* o del *Natural Language Processing (NLP)*, di fatto i domini di maggiore impatto agli albori dell'utilizzo di tecniche di *ML* o *DL* [17] [18]. Naveed Akhtar e Ajmal Mian [19] hanno proposto uno studio che analizza i principali attacchi avversari nei confronti dei task più comuni di *Computer Vision*, mettendo in evidenza i metodi utilizzati per generare l'input avversario a partire dall'input originale. Lo studio propone dei possibili meccanismi di difesa, la maggior parte dei quali sono riferiti al dominio specifico delle immagini e difficilmente applicabili al dominio *NIDS*, per via delle differenze intrinseche che caratterizzano le immagini rispetto alla trattazione di pacchetti di rete. Wang *et al.* [20] propongono uno studio in cui vengono analizzati i fattori che caratterizzano gli attacchi avversari rivolti a modelli di *ML*, descrivendone la tassonomia e la fase di addestramento oggetto dell'attacco. Anche in questo caso vengono descritti i possibili metodi di attacco e relativi meccanismi di difesa, i quali non tengono comunque in considerazione gli aspetti che caratterizzano un dominio di applicazione specifico come quello degli *NIDS*. Gli algoritmi di *ML* e *DL* hanno riscontrato un sempre crescente utilizzo anche nell'ambito della rilevazione delle intrusioni [6], grazie alla loro capacità di apprendere e distinguere patterns di traffico di rete benigni rispetto a patterns anomali. Ai fini dell'addestramento e della valutazione di tali modelli, sono stati pubblicati negli anni vari dataset contenenti sia campioni di traffico di rete benigni che anomali. Alcuni dei più diffusi sono *KDDCup 99*, *NSL-KDD*, *Kyoto*, *UNSW-NB15*, *WSN-DS*, *CICIDS 2017* e *CSECICIDS 2018*. Grazie alla diffusione di questi dataset ed alla sempre crescente disponibilità di risorse hardware per l'addestramento dei modelli di *ML*, sono stati svolti vari studi volti a proporre metodologie sempre più sofisticate ai fini della rilevazione delle intrusioni. Mishra *et al.* [21] propongono uno studio nel quale vengono confrontate le performance di varie tecniche di *ML* applicate al dominio *NIDS* e valutate sulla base del dataset *KDD'99*. Lo studio considera cinque tecniche di classificazione standard quali *Decision tree*, *Neural Network*, *Naive Bayes*, *Support Vector Machine* e *Fuzzy Association*

rules e confronta il tasso di rilevamento garantito da ciascuno di essi. Lo studio affronta anche il problema del *curse of dimensionality* di cui sono oggetto gli algoritmi di *ML*, proponendo alcune soluzioni e confrontando i risultati ottenuti rispetto al considerare l'intera dimensionalità dei dati. Ennaji *et al.* [22] propongono invece un modello ensemble di classificatori per il miglioramento delle performance degli *NIDS*. Lo studio si basa sul dataset *NSL-KDD* e valuta l'adozione di una tecnica di ensemble learning che combina diversi algoritmi come *SVM*, *KNN*, *LR* ed *NB*. Recenti studi hanno inoltre dimostrato l'efficacia di tecniche di *DL* applicate all'ambito *NIDS*. Le più diffuse in tal senso riguardano l'applicazione di *Deep Neural Network (DNN)*, *Convolutional Neural Network (CNN)* e *Recurrent Neural Network (RNN)*. R. Vinayakumar *et al.* [23] propongono un'architettura *DNN* per la rilevazione delle intrusioni addestrata e valutata sulla base di più dataset tra i più diffusi in questo ambito. Lo studio propone il framework *Scale-Hybrid-IDS-AlertNet* ai fini della rilevazione di intrusioni sia *Host-based* che *Network-based*, basato su un modulo di monitoraggio ed un modulo di rilevazione fondato su una rete neurale profonda. Data la sequenzialità che caratterizza i pacchetti in una rete, tecniche come le *Recurrent Neural Networks (RNN)* sono state a loro volta analizzate in ambito *NIDS* [8], anche se risultano poco esplorate rispetto ad altre alternative. Anche l'applicazione di *Convolutional Neural Networks (CNN)* ai fini della rilevazione delle intrusioni è stata analizzata in diversi studi. Si tratta di particolari reti neurali applicate tipicamente all'ambito di immagini o video, data la loro capacità di catturare informazioni locali tramite l'utilizzo di filtri di convoluzione. I pacchetti in un traffico di rete non posseggono informazioni locali per cui l'applicazione di *CNN* in ambito NIDS avviene tipicamente in combinazione con altre tecniche [24] [12]. Il crescente utilizzo di algoritmi di *ML* e *DL* applicati al dominio della rilevazione delle intrusioni, ha conseguentemente generato interesse nell'analizzare gli impatti che gli attacchi avversari comportano anche in questo dominio, considerando gli aspetti che lo contraddistinguono rispetto al dominio *CV* o *NLP*, per i quali il problema era già stato apertamente discusso ed analizzato. Come descritto in [12], vi sono infatti alcune differenze tra dati relativi al traffico di rete e dati relativi ad immagini o sequenze testuali, le quali rendono necessarie delle tecniche ad hoc pensate per contrastare gli attacchi avversari in ambito *NIDS*. Tra queste vi è la differente dimensionalità dello spazio delle *features*, l'elevata correlazione che caratterizza le *features* estratte a partire dal traffico di rete, così come la loro irregolarità dovuta alla presenza di *features* di diverso tipo (categoriche, numeriche, etc.). Appruzzese *et al.* [25] analizzano l'applicazione di attacchi avversari nei confronti di *NIDS ML-based*, proponendo delle tecniche realistiche di modellazione e valutazione di tali attacchi. Lo studio si focalizza in modo particolare sul concetto di 'capacità'

dell'attaccante, ovvero la misura in cui quest'ultimo ha conoscenza circa il modello target dell'attacco, la tecnica di estrazione delle *features* ed il dataset di addestramento, unitamente anche alla capacità di ottenere informazioni utili interrogando direttamente il modello. Lo studio non propone tuttavia possibili tecniche di difesa. Zhang *et al.* [26] propongono uno studio in cui vengono discussi i principali attacchi di tipo Black-box relativi ad *NIDS DL-based* e nel quale viene conseguentemente proposto un framework di difesa chiamato *Tiki-Taka*, basato su diverse tecniche di difesa combinate tra loro.

1.3 Attacchi Avversari

In questa sezione verranno descritte le principali strategie utilizzate ai fini della generazione di input avversari [27]. Queste si possono distinguere in base a vari aspetti, tipicamente riferiti a fattori come **obiettivi**, **conoscenza** circa il modello target e **capacità** di manipolazione dell'input da parte dell'attaccante. Una tipica distinzione è quella tra attacchi *White Box* ed attacchi *Black Box*, i quali si differenziano per le informazioni cui l'attaccante ha accesso circa il modello target. Di seguito verranno descritte le principali tecniche utilizzate per mettere in atto attacchi di entrambe le tipologie.

1.3.1 White Box

In uno scenario *White Box*, si assume che l'attaccante abbia completa conoscenza del modello target, dunque ne conosce la tipologia, gli iperparametri e altre informazioni che lo caratterizzano. Si assume che l'attaccante abbia completa capacità di manipolazione dell'input ai fini della generazione dell'attacco avversario. Tra gli attacchi *White Box* più comuni è possibile distinguere i seguenti:

- **Fast Gradient Sign Method (FGSM)**

L'attacco *FGSM* fa parte della categoria di attacchi di tipo *Maximum allowable*, ovvero che prediligono il vincolo di confidenza rispetto a quello di similarità. In questo attacco, proposto da Goodfellow *et al.* [28], l'input avversario viene generato utilizzando i gradienti della funzione di perdita della rete neurale: a partire da un certo input, questo viene opportunamente modificato in

modo tale da massimizzare la perdita. Formalmente, l'input avversario è ottenuto come segue:

$$adv_x = x + \varepsilon \; sign(\nabla_x \, J(\theta, x, y)) \tag{1.4}$$

in cui ε è la variabile che consente il controllo della perturbazione, y è l'etichetta desiderata e x rappresenta l'input originale. La funzione di segno garantisce che l'entità della perdita sia massimizzata. Il gradiente non viene quindi determinato a partire dai parametri del modello, bensì a partire dall'input, il che consente di definire la modifica necessaria alla generazione dell'input avversario. Lo svantaggio principale di questo attacco, dal punto di vista dell'applicazione in ambito *NIDS*, risiede nel fatto che viene ricercata la perturbazione che massimizza la confidenza della classe target. Le modifiche effettuate alle *feature* in tal senso potrebbero dunque portare a risultati meno pratici in applicazioni reali.

- **DeepFool**

L'attacco *DeepFool* fa parte della famiglia di attacchi di tipo *Minimun Norm*, ovvero che prediligono il vincolo di similarità nella generazione degli esempi avversari rispetto al vincolo di confidenza. Questo attacco è stato proposto da Moosavi-Dezfooli *et al.* [29] e prevede una generazione iterativa di esempi avversari da sottoporre ad un modello di classificazione (binaria o multiclasse). L'input iniziale risiede in una regione delimitata dai confini decisionali del classificatore, dunque all'interno di un poliedro dato dall'intersezione di più iperpiani corrispondenti ai singoli classificatori binari (di fatto una classificazione multiclasse). Ad ogni iterazione, viene definito il vettore di perturbazione tale da spostare l'input verso i confini del poliedro ottenuto linearizzando i confini della regione in cui esso risiede. La perturbazione ad ogni iterazione viene sommata a quella calcolata all'iterazione precedente, sino ad ottenere una classificazione dell'input pari a quella data dai confini decisionali originali del modello. Data la sua natura iterativa, questo attacco richiede tipicamente un tempo maggiore per generare istanze avversarie rispetto ad altre tipologie di attacco.

- **Jacobian Saliency Map Attack (JSMA)**

Anche questo attacco, proposto da Papernot *et al.* [30] rientra nella categoria di attacchi di tipo *Maximum Allowable*. In questo caso, l'input avversario viene generato effettuando iterativamente delle singole modifiche e monitorando gli effetti che tali modifiche comportano rispetto al

risultato della classificazione. Il monitoraggio avviene attraverso una saliency map, definita a partire dai gradienti della rete neurale e basata sulla matrice Jacobiana ottenuta come segue:

$$J_F(x) = \frac{\partial F(x)}{\partial x} = \left[\frac{\partial F_j(x)}{\partial x_i}\right]_{i \in 1...M,\ j \in 1...N} \tag{1.5}$$

In questa mappa, un valore maggiore indica una maggiore probabilità di ingannare la rete nel prevedere l'etichetta target come etichetta dell'input modificato invece dell'etichetta originale. Si tratta pertanto di un algoritmo di generazione di input avversari targettizzati. Una volta calcolata la mappa, l'algoritmo determina l'alterazione dell'input che inganna più efficacemente la rete (es. pixel da modificare in un'immagine). Questo processo viene ripetuto finché viene raggiunta una quantità massima di alterazione consentita (es. numero di pixel alterati) o finché l'inganno viene portato a compimento.

- **Basic Iterative Method (BIM)**

Si tratta anche in questo caso di un attacco di tipo *Maximum Allowable*. È stato proposto da A. Kurakin *et al.* [31] e rappresenta un'estensione dell'attacco *FGSM*: anziché generare l'input avversario in un un'unica iterazione, l'attacco *FGSM* viene applicato più volte, effettuando piccole modifiche in ciascuna iterazione ai fini della massimizzazione della loss. Dopo N cicli, si otterrà dunque l'input avversario che verrà eventualmente modificato ulteriormente per far sì che la perturbazione rientri in un certo limite massimo. Ad ogni iterazione viene dunque effettuato il seguente calcolo:

$$I_\rho^{(i+1)} = Clip_\varepsilon\ \left\{ I_\rho^i + \alpha\ sign\left(\nabla_x J\left(\theta, I_\rho^i, \ell\right)\right)\right\} \tag{1.6}$$

dove I_ρ^{i+1} rappresenta l'input avversario all'i-esima iterazione, *Clip* controlla la quantità di perturbazione in modo che rientri nel limite ε ed α rappresenta la dimensione dello step (tipicamente 1).

- **Carlini & Wagner attack (C&W)**

Si tratta di un set di attacchi introdotti da Carlini e Wagner [32], che fanno riferimento alla famiglia di attacchi avversari *Regularization Based*, ovvero che tentano di bilanciare il vincolo di similarità e confidenza. La generazione dell'input avversario è affrontata come un

problema di ottimizzazione, nel quale si tenta di ricavare la minima perturbazione che causa la misclassificazione dell'input. La funzione obiettivo da ottimizzare è definita come segue:

$$minimize\ D(x,\ x\ +\ \delta) + c\ \cdot\ f(x\ +\ \delta) \tag{1.7}$$

in cui $D(x,\ x\ +\ \delta)$ misura la perturbazione (tipicamente rappresentata dalla norma L_2 o L_∞), $f(x\ +\ \delta)$ rappresenta una funzione definita in modo tale che $f(x\ +\ \delta) \leq 0$ se $C(x\ +\ \delta) = t$, ovvero l'input alterato viene classificato nella classe target. La costante c consente di bilanciare i due obiettivi. Il vantaggio dell'algoritmo risiede nella possibilità di generare input avversari efficaci con perturbazioni minime, nella sua versatilità rispetto a diversi modelli target e nella sua robustezza rispetto a diverse tecniche di difesa.

- **Virtual Adversarial Training (VAT)**

Anche questo attacco, proposto da T. Miyato *et al.* [33], fa riferimento agli attacchi di tipo *Regularization Based*. La generazione dell'input avversario avviene secondo la seguente legge:

$$x_{adv} = x + \ \varepsilon\ \cdot\ d \tag{1.8}$$

in cui x rappresenta l'input originale, ε un parametro che controlla la perturbazione e d definito in modo tale da massimizzare la divergenza *Kullback-Leibler*:

$$KL[F(x)||F(x\ +\ d)] \tag{1.9}$$

La quantità $F(x)$ esprime le probabilità di appartenenza alle classi, ottenuta in genere applicando la funzione softmax alla logits del classificatore. L'applicazione del *VAT* non richiede troppe informazioni (ad esempio, la funzione di perdita o i limiti decisionali del classificatore); in quanto tale, è particolarmente adatto al dominio del rilevamento delle intrusioni poiché l'attaccante ha spesso scarsa conoscenza circa l'IDS target.

1.3.2 Black Box

In scenari di tipo *Black Box* si assume che l'attaccante abbia zero conoscenza circa il modello target [34]. Ai fini della generazione degli input avversari, non è possibile dunque tenere in

considerazione aspetti relativi al gradiente o alla loss del modello, come avviene tipicamente in attacchi di tipo *White Box*. Inoltre, si assume che il modello target venga trattato dall'attaccante come un oracolo al quale vengono effettuate delle interrogazioni al fine di ricavare determinate informazioni sul modello stesso. Verranno descritte di seguito le caratteristiche principali dei più diffusi attacchi di tipo *Black Box*.

- **Substitute Model Attack**:

Questo attacco, proposto da Papernot *et al.* [16], mira a definire un modello F che approssima i confini decisionali dell'oracolo O (modello target). Ciò avviene sottoponendo all'oracolo un insieme di input sintetici, generati tramite una tecnica di *Data Augmentation Jacobian-based*. La risposta fornita dall'oracolo viene utilizzata per associare un'etichetta a ciascun input. Si ottiene in tal modo un dataset di esempi etichettati che possono essere utilizzati per addestrare il modello surrogato che rispecchierà i confini decisionali dell'oracolo O. Avendo a disposizione il modello surrogato, l'attaccante può adottare una delle tecniche di tipo *White Box* per generare input avversari per quel modello. Per la proprietà di trasferibilità avversaria [5], gli input avversari generati a partire dal modello sostituto, saranno validi anche per il modello target, anche se quest'ultimo dovesse presentare caratteristiche strutturali differenti rispetto al modello a partire dal quale gli input avversari sono stati generati.

- **Zeroth Order Optimization**

In questo attacco, proposto da P. Chen *et al.* [35], l'obiettivo è quello di applicare direttamente una strategia di attacco *White Box* sul modello target. Le risposte fornite dal modello target rispetto agli input sentici vengono utilizzate per stimare il gradiente della funzione decisionale del modello. Si tratta dunque di un attacco *score-based*, il quale non richiede l'addestramento di un modello sostituto. La stima del gradiente, nel caso dell'attacco *ZOO*, avviene considerando la differenza simmetrica del quoziente tra le risposte fornite dall'oracolo per input vicini. Gli autori propongono inoltre alcune tecniche che mirano a ridurre il numero di query da sottoporre al modello target. Tra queste vi è il campionamento per importanza, l'attacco gerarchico e la riduzione dello spazio di attacco. Lo studio dimostra inoltre l'efficacia di questa strategia rispetto ad attacchi *Black Box* basati sull'addestramento di un modello sostituto ma, di contro, richiede un maggiore sforzo computazionale.

- **Generative Adversarial Networks**

Una possibile strategia di attacco in scenari Black Box è rappresentata dall'utilizzo di *GANs* [36], particolari reti neurali il cui obiettivo è quello di generare esempi sintetici che rispecchiano quelli reali. L'architettura di queste reti prevede un **Generatore** ed un **Discriminatore**: il primo si occupa di generare gli esempi sintetici mentre il secondo ne calcola la loss rispetto agli esempi reali. L'obiettivo del generatore è quello di minimizzare questa loss, dunque generare esempi che via via diventino sempre più simili a quelli reali. In uno scenario di attacco, le *GAN* possono essere dunque utilizzate per generare input avversari, definendo una perturbazione dell'input tale da minimizzare la loss del Discriminatore [37] [38]. Le *GAN* rappresentano una valida alternativa agli attacchi basati sul modello sostituto o sulla stima del gradiente ma, tipicamente, si basano su una manipolazione dell'input a livello di *features*, il che non sempre si traduce nella possibilità di adottare queste strategia in casi pratici.

- **HopSkipJump**

Questo attacco, proposto da Chen *et al.* [39], fa parte della famiglia di attacchi *Black-Box* di tipo *decision-based*. Non richiede cioè l'addestramento di un modello sostituto o metodi per stimare il gradiente a partire dalle predizioni del modello, bensì si basa esclusivamente sull'output stesso del modello target. Rappresenta una versione ottimizzata dell'attacco *Boundary*, proposto da Brendel *et al.* [40]. In entrambi i casi, si parte inizializzando il campione avversario in modo che appartenga alla classe target. L'attacco procede poi iterativamente: a ciascuno step viene applicata una perturbazione tale da spostare l'input avversario in direzione dell'input originale, calcolando la differenza tra i due. Una volta che il campione di destinazione raggiunge il confine tra la classe originale e quella di destinazione, viene proiettato lungo tale confine. Nel caso dell'attacco *HopSkipJump*, gli autori propongono sia un metodo per il raggiungimento del confine di decisione, sia un metodo di proiezione lungo il confine stesso. Inoltre, viene proposta una strategia per ottimizzare il numero di query necessarie per l'esecuzione del processo iterativo.

1.4 Strategie di difesa

In questa sezione verranno discusse le principali strategie di difesa che possono essere adottate al fine di individuare o mitigare l'effetto di attacchi avversari. Si possono distinguere strategie

che mirano ad individuare la presenza di input avversari, da strategie che mirano invece ad aumentare la **robustezza** del modello rispetto a quest'ultimi. Verranno descritte di seguito alcune tra queste strategie, analizzandone possibili ambiti di applicazione, vantaggi e svantaggi.

- **Gradient Obfuscation**

L'offuscamento del gradiente rientra tra le strategie di difesa il cui scopo è quello di proteggere i parametri del modello dall'accesso esterno. Limitare la quantità di informazioni significative sul modello cui l'attaccante ha accesso, limita di fatto la sua capacità di condurre attacchi di tipo *White Box*. In particolare, nel caso dell'offuscamento del gradiente, il parametro che si intende proteggere è appunto il gradiente della loss del modello. Esistono diversi metodi che consentono di realizzare questo comportamento, tra cui *Shattered Gradient*, *Stochastic Gradient* e *Vanishing & Exploding Gradient* [12]. Nel primo caso l'obiettivo è rendere il gradiente non differenziabile, nel secondo caso il gradiente viene mascherato aggiungendo casualità alla rete o all'input, mentre nel terzo caso la difesa avviene iterando più volte la valutazione della rete, ponendo come input l'output dell'iterazione precedente. La tecnica di mascheramento del gradiente può essere bypassata utilizzando stime del gradiente, attraverso metodi come *BPDA* o la reparametrizzazione [12] e, inoltre, non risulta efficace rispetto ad attacchi di tipo *Black Box*.

- **Input Pre-Processing**

In questo caso, la strategia di difesa consiste nel processare l'input prima di sottoporlo al modello, al fine di rimuovere l'eventuale perturbazione e ricondurlo al suo stato originale. Un esempio di questo tipo è dato da *Mag-Net* [41], un'architettura costituita da un *Detector* ed un *Reformer*. Si tratta in entrambi i casi di autoencoder addestrati su campioni di traffico benigni tali per cui il *Detector* ha l'obiettivo di individuare input avversari che verranno eventualmente scartati, mentre il *Reformer* prende in input i campioni ritenuti benigni dal *Detector* e ne effettua una ricostruzione al fine di rimuovere ogni possibile perturbazione prima di sottoporli di fatto al modello. Si tratta di una tecnica che consente di rilevare attacchi avversari anche abbastanza sofisticati ma che introduce un overhead non indifferente nel tempo di identificazione dell'attacco, per via dell'utilizzo di due Autoencoder.

- **Adversarial Training**

Questa strategia consiste nell'addestrare il modello anche con input avversari, in modo da adattare la funzione di decisione rispetto a possibili perturbazioni [28]. L'obiettivo è dunque quello di rilassare i confini decisionali del modello, portando ad una classificazione corretta anche di input perturbati. Nel dominio *CV* sono stati proposti vari framework per l'addestramento avversario [42] [43] i quali si distinguono tipicamente per il metodo utilizzato per generare gli esempi avversari o nel metodo utilizzato per incrementare la robustezza della funzione di loss. Nel dominio *NIDS*, il vantaggio dell'addestramento avversario è che non comporta ulteriori costi di elaborazione a runtime. Tuttavia, applicare nella pratica l'Adversarial training per gli *NIDS* rappresenta una sfida significativa: non sempre è possibile per il difensore ottenere il traffico rete relativo a possibili attacchi e generare input avversari contro gli *NIDS*.

- **Defensive Distillation**

Questa tecnica, proposta da Papernot *et al.* [44], mira ad aumentare la robustezza del modello rispetto ad input avversari ricorrendo a due reti neurali: una *Teacher Network* ed una *Distilled Network*. La prima è addestrata sul set di dati originale utilizzando procedure di addestramento standard. Le probabilità delle classi date da questa rete vengono utilizzate come target per l'addestramento della seconda rete. La *Distilled Network* può avere la stessa architettura o un'architettura diversa della *Teacher Network*, e viene quindi addestrata considerando come etichette le probabilità di output ottenute nella prima fase di addestramento. Questa rete impara a generalizzare meglio imitando la distribuzione dei risultati della *Teacher Network*, la quale include i livelli di confidenza per ciascuna classe.

- **Adversarial detection**

Si tratta di una strategia di difesa che mira ad individuare la presenza di input avversari a runtime, prima che vengano sottoposti al modello target. Esistono diverse tecniche che consentono di realizzare questo comportamento, le quali si basano tipicamente su una carattersitica intrinseca degli input avversari: essi sono generati in modo sintetico, dunque posseggono proprietà che li distinguono in qualche modo rispetto agli input normali. Una tipica strategia è infatti quella di utilizzare strumenti statistici come la *Principal Component Analysis*, con l'obiettivo di ridurre la dimensionalità dell'input e forzare l'attaccante a modificare solo le componenti principali ai fini della generazione dell'input avversario [45]. Riducendo lo spazio di attacco, aumenta

la probabilità di individuare possibili perturbazioni dell'input. Un'altra soluzione è rappresentata dall'utilizzo di strumenti come la *Maximum Mean Discrepancy*, attraverso la quale viene determinata la misura in cui due set di dati derivano dalla stessa distribuzione sottostante [46]. Alcune tecniche più avanzate di Adversarial Detection prevedono invece un approccio *Mutation – based*: i confini decisionali del modello vengono alterati in modo casuale e viene determinata la misura in cui ciascun input risulti sensibile rispetto a queste alterazioni. Si assume che gli input avversari, che risiedono più in prossimità dei confini decisionali, abbiano una sensibilità maggiore rispetto a queste mutazioni [47].

1.5 Attacchi avversari nel dominio NIDS

Come accennato nei paragrafi precedenti, l'adozione di tecniche di *ML* e *DL* nel dominio *NIDS* ha portato da un lato a definire sistemi di rilevazione più sofisticati e più robusti sia ad attacchi noti che di tipo *zero-day* [48] e, allo stesso tempo, ha esposto questi sistemi alla minaccia di attacchi avversari cui gli algoritmi di *ML* e *DL* sono soggetti per loro natura [28] [3]. In questa sezione verranno analizzate alcune delle principali tecniche per attuare un attacco avversario rivolto ad un *NIDS* ed i relativi meccanismi di difesa proposti dagli autori. Le strategie di attacco più diffuse in ambito *NIDS* si incentrano sul garantire l'evasività dell'attacco stesso, condizionata allo stesso tempo al garantire la sua natura malevola.
Wang [49] ha affrontato il tema degli attacchi avversari in *NIDS DL-based*, verificando l'impatto degli attacchi *FGSM*, *JSMA*, *DEEPFOOL* e *CW* rivolti ad un *MLP-based IDS* addestrato sul dataset *NSLKDD*. I risultati hanno mostrato come ciascun attacco è stato in grado di diminuire le performance del modello, pur dimostrando un'efficacia minore rispetto all'applicazione nel dominio della *Computer Vision*, per il quale questi attacchi sono stati originariamente definiti. Si consideri il fatto che si tratta di attacchi basati sulla manipolazione delle *feature*, dunque difficilmente applicabili in casi reali in cui l'attaccante non ha conoscenza circa il processo di estrazione delle *feature* utilizzato e non può dunque risalire al traffico di rete avversario.
Alhajjar *et al.* [50] hanno proposto uno studio nel quale vengono analizzati gli effetti di attacchi avversari rivolti ad *NIDS ML-based* e basati su tecniche di computazione evolutiva (*PSO* e *GA*) o Deep Learning (*GAN*). Gli autori hanno tenuto in considerazione il vincolo relativo alla difficoltà, per un attaccante, di poter modificare qualsiasi *feature* ai fini della perturbazione dell'input. Inoltre, è stata affrontata la necessità di garantire la natura malevola dell'attacco oltre che l'evasività. Gli esperimenti condotti su sette diversi modelli di Machine Learning, hanno

mostrato come gli algoritmi *SVM* e *DT* risultano maggiormente vulnerabili ad attacchi avversari basati su *GAN* o *GA*. Anche in questo caso si tratta di attacchi basati sulla manipolazione delle *feature*.

Yang *et al.* [51] hanno proposto uno studio nel quale vengono analizzati gli effetti di attacchi avversari in scenari più realistici, in cui cioè l'attaccante ha una conoscenza limitata o nulla circa il modello target. In particolare, sono stati condotti attacchi basati su *WGAN*, *ZOO* e sul metodo del *Subsititute Model*. Ciascun attacco ha comportato una diminuzione delle performance del modello target *DL-based*, dimostrando di fatto che anche in scenari di tipo *Black Box* gli attacchi avversari possono risultare efficaci. La tecnica del modello sostituto, rispetto alle altre due, ha avuto un minor impatto sulla validità del modello target, per via del fatto che non sempre il modello sostituto riesce a rispecchiare i confini decisionali del modello target. L'attacco di tipo *ZOO* ha ottenuto invece i risultati migliori in termini di efficacia nel compromettere la valutazione corretta degli input da parte del modello target, pur presentando una complessità computazionale superiore rispetto agli altri due metodi. L'attacco *GAN* ha a sua volta portato a risultati efficaci anche se gli autori hanno posto in evidenza l'instabilità che caratterizza l'addestramento di reti *GAN*.

Guo *et al.* [52] hanno proposto un attacco basato sul metodo del *substitute model* assumendo quindi uno scenario di tipo *Black Box*. Le fasi dell'attacco hanno previsto dapprima la generazione del modello *MLP* sostituto che rispecchiasse i confini decisionali del modello target e, successivamente, la generazione di input avversari sulla base di un attacco *White Box* rivolto al modello sostituto e basato su un'estensione della tecnica *BIM* adattata al dominio *NIDS*. Gli autori hanno infatti considerato che, nel generare un input avversario per un *NIDS ML* o *DL-based*, è necessario tenere presente che le *features* nel dominio *NIDS* presentano vincoli e correlazioni tra di esse, per cui la modifica di una *feature* può avere impatti anche su altre *features* o sulla validità stessa del traffico di rete. Il metodo proposto è stato condotto su diversi modelli target e valutato sulla base del dataset *KDDCUP99* e *CSECICIDS2018*. I risultati ottenuti da questo studio hanno mostrato la possibilità di generare input avversari efficaci anche in scenari reali in quanto generati a livello di traffico di rete.

Zhang *et al.* [53] hanno proposto un metodo di generazione di input avversari attraverso un metodo a forza bruta (*BFAM*), adatto a scenari di tipo *Black Box* in quanto non richiede la conoscenza della struttura o dei parametri del modello target. In particolare, il processo di generazione dell'input avversario viene affrontato come un problema di ottimizzazione in cui si ricercano le *features* che hanno un impatto maggiore sulla variazione del grado di confidenza dato dal

modello target. Si tratta dunque di verificare la misura in cui ciascuna possibile combinazione delle *features* comporta una classificazione errata da parte del modello. Gli autori hanno posto particolarmente in evidenza la capacità di questo metodo di superare alcuni limiti delle reti *GAN* nella generazione di input avversari, quali ad esempio l'instabilità cui sono soggette nella fase di addestramento o l'impossibilità di intervenire nel processo di generazione dell'input avversario per determinare, ad esempio, quali *features* modificare. I risultati ottenuti dimostrano inoltre l'efficacia di questo metodo nel generare input avversari in grado di garantire sia evasività che natura malevola dell'attacco. Si tratta tuttavia di un metodo che può risultare computazionalmente dispendioso, a causa della necessità di dover verificare ciascuna possibile combinazione di *features* e che può essere rilevato da eventuali sistemi di monitoraggio che controllano il numero di query rivolte al modello target.
Shu *et al.* [54] hanno proposto un metodo per la generazione di esempi avversari in uno scenario di tipo *Black Box*, basato su una *GAN* e un algoritmo di *Active Learning*. Il modello *GAN* prevede come generatore un *Variational Autoencoder* e come discriminatore un *MLP* che rappresenta il modello sostituto. Il processo di *Active Learning* assicura che il modello sostituto approssimi il modello target con un alto grado di accuratezza. La fase di addestramento del modello *GAN* ha l'obiettivo di minimizzare la differenza tra l'input originale e l'input avversario (garantire il vincolo di similarità) e, contemporaneamente, minimizzare la differenza tra l'output fornito dal discriminatore e l'output atteso, ovvero la classificazione dell'input avversario come benigno. I risultati di questo studio mostrano che il metodo proposto raggiunge un tasso di successo di evasione del 98% utilizzando solo 25 istanze etichettate durante l'addestramento. Lu *et al.* [55] hanno proposto due tipologie di attacchi avversari, entrambi in scenario di tipo *Black Box*, riferiti a sistemi di rilevamento delle intrusioni log-based. In entrambi i casi ci si basa su un modello sostituto per condurre un attacco di tipo *White Box*, con l'obiettivo di identificare i punti chiave da modificare per far apparire log relativi a sessioni malevole come benigni. Nel primo caso viene utilizzato un approccio *Attention – based*, in cui dapprima vengono individuate le *features* più importanti dal punto di vista della classificazione dell'input e, successivamente, vengono attuate le perturbazioni su queste *features* ai fini della generazione degli input avversari. Il secondo metodo proposto è invece basato sull'ottimizzazione del gradiente e, in particolare, per adattare questo approccio a input corrispondenti a log sequenziali, l'algoritmo è stato integrato con una *CNN* con modellazione sequenziale di dati testuali. Entrambi i metodi hanno portato a risultati migliori rispetto ad altre tecniche di generazione di input avversari per *IDS log-based.*
Clements *et al.* [56] hanno valutato un *NIDS DL-based*, chiamato *Kitsune*, relativamente sia

alla capacità di difesa rispetto ad attacchi malevoli, sia rispetto alla robustezza relativamente ad input avversari. Il modello per la rilevazione delle intrusioni, chiamato *KitNET*, è basato su un *Autoencoder*, dunque addestrato su input benigni al fine di apprenderne una rappresentazione latente. La valutazione della robustezza di *KitNET* rispetto ad input avversari è stata condotta tramite quattro tipici attacchi, quali *FGSM*, *JSMA*, *CW* ed *ENM*. I risultati ottenuti hanno mostrato la vulnerabilità del modello target rispetto a tutti e quattro gli attacchi avversari.
Hashemi *et al.* [15] hanno proposto dei metodi per la generazione di input avversari rivolti ad *NIDS packet-based* e *flow-based*. Nel primo caso, l'algoritmo proposto dagli autori prevede di agire su tre aspetti: numero di pacchetti scambiati, tempistiche nell'invio dei pacchetti e iniezione di nuovi pacchetti nel traffico di rete. Queste operazioni si combinano dunque tra di loro al fine di generare pacchetti avversari classificati erroneamente da un *NIDS packet-based*. Nel caso di generazione di input avversari rivolti ad *NIDS flow-based*, gli autori evidenziano alcune differenze rispetto al caso *packet-based*, quali: l'impossibilità di agire su determinate *features* che sono estratte dal traffico di rete e la dipendenza tra *features*. Sulla base di queste considerazioni, l'algoritmo proposto prevede di suddividere in gruppi le *features* e generare input avversari considerando le differenze tra ciascun gruppo. I risultati dello studio hanno mostrato la possibilità di generare input avversari per IDS addestrati su *Kitsune* o *CICIDS2017*.

1.6 Sintesi capitolo

Come osservato, i sistemi di rilevamento delle intrusioni *ML/DL-based*, rappresentano una valida alternativa ad altre tecniche tradizionali utilizzate a questo scopo. Va però tenuta in considerazione la loro vulenerabilità rispetto ad attacchi avversari, i quali generano perturbazioni impercettibili dell'input che possono causare una classificazione errata dello stesso da parte del sistema di rilevamento. In questo capitolo sono state presentate le caratteristiche principali di tali attacchi, descrivendone vincoli e modalità di applicazione. A seconda di come questi aspetti si combinano tra loro, è stato possibile descrivere le principali tecniche di attacco utilizzate allo stato dell'arte attuale, differenziandole in tecniche *White Box* e *Black Box*. In entrambi i casi si è osservata la loro fattibilità anche in ambito *NIDS*, riportando diversi studi che hanno svolto delle analisi in tal senso. Oltre ai possibili attacchi avversari, sono state analizzate le tecniche di difesa più diffuse che possono essere adottate per contrastare la presenza di input cui sono state applicate perturbazioni con scopo avversario. Tra queste, si è osservato come alcune siano efficaci esclusivamente in uno scenario *White Box* o *Black Box* o eventualmente

bypassabili da un attaccante che conosce il sistema di difesa messo in atto. Data la larga diffusione di algoritmi *ML/DL-based* ai fini del rilevamento delle intrusioni, la trattazione di attacchi avversari, e relative difese, rappresenta un campo di ricerca ampio e da esplorare. Nel prossimo capitolo, verrà proprosto un sistema di rilevamento delle intrusioni di rete *DL-based*, privo di protezioni da attacchi avversari. Se ne valuteranno le performance circa la capacità di individuazione di tipici attacchi di rete e, successivamente, la vulnerabilità rispetto ad attacchi avversari. Ciò rappresenterà il punto di partenza per valutare in seguito l'efficacia del sistema di difesa che verrà proprosto in questo studio.

Capitolo 2

Implementazione di un NIDS DL-based

Con la crescente complessità delle reti di computer e degli attacchi ad esse mirati, si sono rese necessarie nel tempo tecniche sempre più sofisticate per individuare e contrastare la presenza di intrusioni nei sistemi informatici. A tal scopo, la progettazione di sistemi di rilevamento di intrusioni di rete (*NIDS*), ha incluso metodi avanzati per riconoscere, con sempre maggior accuratezza, la presenza di intrusioni in una rete. In generale, le caratteristiche che un *NIDS* dovrebbe garantire riguardano:

- capacità di **riconoscere**, con alto tasso di successo, attività di rete riconducibili a possibili attacchi in corso;
- capacità di **adattamento** a frequenti mutazioni delle strategie di attacco, dunque abilità nel riconoscere anche attacchi mai osservati in precedenza;
- capacità di processare **grandi quantità** di dati al fine di ricavare informazioni utili in modo efficiente.

I tipici paradigmi messi in atto per la rilevazione di intrusioni possono essere di tipo *classification-based* o *anomaly-based*. Nel primo caso, sulla base di pattern di attacco noti, il sistema di rilevamento analizza il traffico di rete per determinare la misura in cui questo possa essere ricondotto ad uno di questi pattern. Questo tipo di approccio è caratterizzato da alto grado di successo rispetto ai casi noti ma, di contro, risulta poco efficiente rispetto a strategie di intrusioni mai osservate in precedenza (attacchi *zero-day*). Nel caso dei sistemi *anomaly-based*, l'approccio prevede di addestrare il sistema a riconoscere pattern normali di utilizzo della rete, in modo da segnalare i casi che si discostano dal normale comportamento atteso. Questi approcci possono

portare ad un tasso più elevato di falsi positivi rispetto ad approcci basati sulla classificazione ma, di contro, risultano maggiormente efficaci rispetto ad attacchi non noti. Approcci di questo tipo per la progettazione di *NIDS* sono stati messi atto attraverso tecniche di *Machine Learning* o *Deep Learning*, di fatto per la loro capacità di predizione e adattamento a condizioni mutevoli. In particolare, in tempi recenti, la ricerca in questo ambito si è incentrata maggiormente su *NIDS DL-based* per la loro capacità di gestire dati anche ad elevata dimensionalità e garantire predizioni più accurate. Tra le tecniche di *Deep Learning*, una possibile scelta per l'implementazione di un sistema di rilevamento *anomaly-based* è rappresentata dall'utilizzo di *Autoencoder*. Si tratta di particolari reti neurali in cui è possibile distinguere due diverse componenti: un *encoder* ed un *decoder*. L'*encoder* tenta di apprendere una rappresentazione dell'input in uno spazio a più bassa dimensionalità, detto **spazio latente**. Il *decoder*, a partire dalla rappresentazione latente fornita dall'*encoder*, tenta di ricostruire l'input originale. L'uscita di un *Autoencoder* è dunque una copia dell'input [57], per cui l'obiettivo della fase di addestramento è minimizzare l'**errore di ricostruzione** dell'input. L'addestramento è di tipo non supervisionato, in quanto avviene a partire dai dati di input senza necessità di etichette. Questo genere di architettura si presta all'utilizzo in sistemi di rilevamento *anomaly-based*: l'*Autoencoder* viene addestrato su campioni di input corrispondenti a **pattern normali** di utilizzo della rete; in presenza di input che si discostano dal normale pattern appreso, si avrà un elevato errore di ricostruzione che consentirà di rilevare e segnalare l'intrusione. Strutture di questo tipo non necessitano di dati etichettati ai fini dell'addestramento, sono caratterizzati da efficienti fasi di codifica e decodifica dei dati e, di conseguenza, risultano in un metodo efficiente per la rilevazione di attacchi non noti [58]. Di contro, non sempre è possibile generalizzare la condizione di *normale* utilizzo della rete, per cui l'adozione di questi algoritmi ai fini della rilevazione delle intrusioni può portare ad un consistente tasso di *falsi anomali*. Reti neurali più classiche, come un *Multilayer Perceptron*, possono invece garantire un numero inferiore di false segnalazioni. Tali reti prevedono infatti un addestramento di tipo supervisionato, dunque che avviene anche a partire da campioni anomali che la rete apprenderà a riconoscere. L'obiettivo dell'addestramento, in questo caso, è quello di favorire un basso tasso di errore nell'assegnazione dell'etichetta a ciascun input. A partire da associazioni note *input / etichetta* (*training dataset*), i parametri del modello vengono opportunamente modificati in modo da garantire che il maggior numero di predizioni corrispondano alle etichette reali, favorendo anche capacità di generalizzazione rispetto ad input non noti. Dato che il processo di addestramento include anche campioni anomali, si potrebbe avere un errore di predizione, relativamente a quest'ultimi, inferiore rispetto a quello dato da modelli non supervi-

sionati. Di contro, per campioni anomali non noti, si potrebbe invece avere una minore capacità di corretta predizione, in quanto il modello non risulterebbe addestrato a riconoscerli. Date queste considerazioni, il presente studio propone la progettazione e realizzazione di un *NIDS DL-based* basato sulla combinazione di un *Autoencoder* e su una rete neurale profonda classica di tipo *Feed-Forward* corrispondente ad un *Multilayer Pereceptron*. Ciò consentirà anche di valutare con maggior dettaglio l'impatto che avranno gli attacchi avversari (e le conseguenti strategie di difesa) su modelli target di differente natura e combinati tra di loro.

OMISSIS

2.1 Conoscenze correlate

2.1.1 Deep Neural Networks

Le reti neurali profonde rappresentano una delle tecniche maggiormente utilizzate in molte applicazioni dell'intelligenza artificiale, dalla *Computer Vision*, al *Natural Language Processing* e alla *Cybersicurezza*. La particolarità delle reti neurali profonde risiede nella loro capacità di apprendimento anche a partire da dati caratterizzati da **elevata dimensionalità**, il che risulta in modelli non lineari in grado di garantire una predizione efficace. Nel caso di una tipica rete *feed-forward*, si hanno diversi livelli costituiti un certo numero di neuroni, ciascuno dei quali connesso ai neuroni del livello successivo. Le connessioni tra un layer e l'altro sono caratterizzate da pesi, che di fatto rappresentano i parametri dell'apprendimento, descritti dalla matrice W. La struttura di queste reti prevede tipicamente un layer di input, uno o più layers nascosti ed un layer di output. Il layer di output fornisce il risultato al problema di analisi dei dati che si sta affrontando, come ad esempio una classificazione o regressione.L'attivazione di un neurone al layer i è determinata dalla seguente legge:

$$z(i+i) = f(W_i \cdot z_i + b_i) \tag{2.1}$$

in cui f rappresenta una generica funzione di attivazione e b_i il bias al livello i della rete. L'obiettivo della fase di apprendimento è quello di modificare opportunamente la matrice dei pesi W ed i bias b in modo da minimizzare una certa funzione di costo. Ciò avviene attraverso la procedura di *backpropagation*, la quale consiste nel calcolare il gradiente della funzione di costo rispetto ai pesi W ed i bias b, a partire dal layer più esterno e proseguendo iterativamente verso i

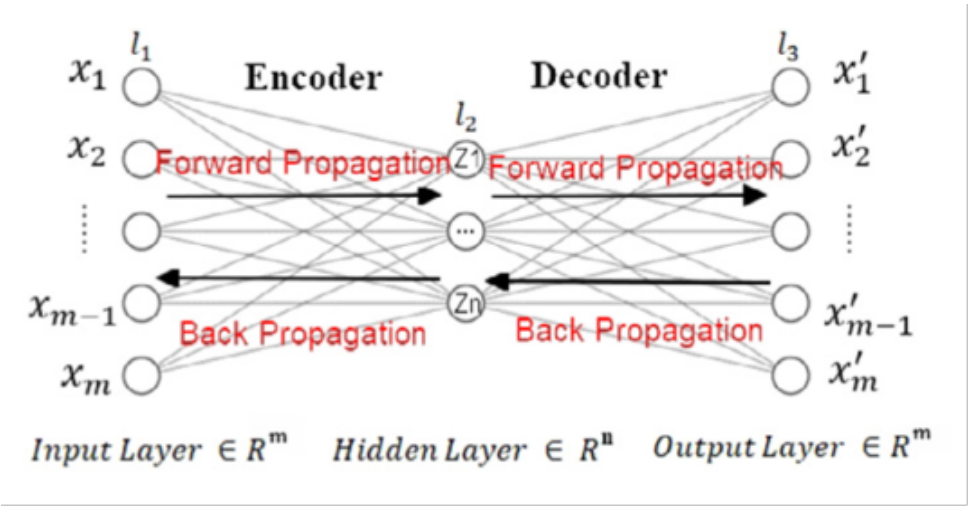

Figura 2.1: Architettura generica di un Autoencoder. Immagine tratta da [60]

livelli più interni [59]. Le reti neurali profonde si presentano in diverse varianti, a seconda del tipo di problema che si intende affrontare e dell'ambito di applicazione. Tra queste è possibile distinguere gli *Autoencoders*, le cui caratteristiche verranno discusse in maggiore dettaglio nel punto successivo.

2.1.2 Autoencoders

Gli Autoencoders sono delle particolari reti neurali in cui è possibile distinguere una componente che effettua la codifica dell'input in uno spazio a più bassa dimensionalità (spazio latente), ed una componente che effettua la decodifica ricostruendo l'input originale a partire dalla rappresentazione latente. L'architettura prevede un layer di input, uno o più layers nascosti ed un layer di output. Il layer di input e quello di output hanno tipicamente la stessa dimensione, mentre i layer intermedi hanno una struttura speculare. La struttura generica di un autoencoder può essere rappresentata come mostrato in Figura 2.1:

L'architettura è di tipo *feed-forward*, per cui anche in questo caso l'obiettivo dell'apprendimento è quello di modificare opportunamente la matrice dei pesi W ed i bias b in modo da minimizzare una certa funzione di costo. Nel caso di un *Autoencoder*, la funzione di costo sarà data dall'**errore di ricostruzione** dell'input, dunque dipenderà dalla misura in cui l'output fornito dalla rete corrisponda all'input originale. Data la loro struttura, e la conseguente capacità di apprendere una rappresentazione latente dell'input, gli *Autoencoder* vengono anche adoperati

ai fini della riduzione della dimensionalità dei dati, in sostituzione a tecniche come *PCA*, con il valore aggiunto di poter apprendere anche trasformazioni non lineari [61]. Formalmente, un *Autoencoder* può essere descritto dalla seguente funzione:

$$g(f(x)) = x' \tag{2.2}$$

in cui x' rappresenta la ricostruzione dell'input x attraverso la funzione $g()$ (*decoder*), quest'ultima applicata alla rappresentazione latente $f(x)$ (encoder) [60].

2.2 Descrizione dataset CICIDS2017

Una componente fondamentale nella definizione di un *NIDS ML o DL-based*, è rappresentata dal dataset a partire dal quale il modello viene addestrato e valutato nelle sue performance, anche rispetto ad attacchi avversari. I dataset possono essere etichettati o non etichettati e contengono tipicamente sia campioni benigni, sia campioni che corrispondono ad attacchi. Tra i dataset che nel corso degli anni sono stati proposti in quest'ambito, nel presente studio si è scelto di basarsi sul *CICIDS2017* (versione improved) [62]. Si tratta di un dataset originariamente prodotto dal *Canadian Institute of Cybersecurity*, con l'obiettivo di superare alcuni limiti dei precedenti dataset utilizzati per la definizione di NIDS [63]. In particolare, i record sono stati generati a partire da file *PCAP* grezzi attraverso il tool *CICFlowMeter*, in cui ciascun record rappresenta un flusso di rete. I flussi corrispondono a scambi bidirezionali di pacchetti tra una sorgente ed una destinazione, aventi in comune le cosiddette cinque tuple (Ip sorgente, Ip destinazione, Porta sorgente, Porta destinazione, Protocollo di livello Applicativo). La raccolta dei dati è avvenuta nel corso di cinque giornate, nelle quali sono stati implementati otto tipologie di attacco quali: *Brute Force FTP*, *Brute Force SSH*, *DoS*, *Heartbleed*, *Web Attack*, *Infiltration*, *Botnet* and *DDoS*. Nel corso della prima giornata sono stati estratti i record relativi unicamente al normale traffico di rete. In studi recenti, il *CICIDS2017* è stato oggetto di revisioni e miglioramenti, sia dal punto di vista dell'estrazione delle *features* a partire dai file *PCAP*, sia dal punto di vista della strategia di etichettatura dei dati. La versione utilizzata nel presente studio comprende 2099976 record, suddivisi come mostrato in Tabella 2.1. Ciascun record comprende 91 *features*, di cui si hanno: 87 che descrivono il flusso, una che rappresenta l'etichetta di classe, due identificativi (uno per il record ed uno del flusso) ed una *feature* che indica se il flusso, pur esibendo un comportamento normale, è parte di un attacco ('*Attempted Category*'). I dati utilizzati sono suddivisi in cinque

file in formato csv, ciascuno dei quali corrisponde al giorno della settimana in cui i relativi record sono stati raccolti (*monday.csv, tuesday.csv...*, *friday.csv*).

2.3 Implementazione MLP ed Autoencoder

OMISSIS

2.4 Sintesi capitolo

In questo capitolo è stato proposto un sistema di rilevamento di intrusioni basato su due modelli *DL-based*, aventi come obiettivo la distinzione tra flussi di rete corrispondenti a normali utilizzi della rete, da flussi relativi a possibili attacchi in atto. A tale scopo, si è scelto di basarsi sul dataset *CICIDS2017* (versione improved), il quale prevede un insieme di record corrispondente a flussi di rete descritti da un insieme di *features*. Avendo a disposizione sia campioni benigni che anomali, con relative etichette, si è scelto di condurre da un lato un addestramento di tipo supervisionato, considerando come target un modello *MLP*, e dall'altro un addestramento non supervisionato basato su un *Autoencoder*. Ciò ha permesso di valutare i differenti risultati ottenuti in entrambi gli scenari e, successivamente, consentirà di valutare gli impatti che input avversari avranno per entrambe le tipologie di modelli. Nel caso dell'*MLP*, sono stati ottenuti risultati migliori in termini di percentuale di campioni correttamente classificati, portando dunque ad un basso margine di errore. Nel caso dell'*Autoencoder*, pur avendo ottenuto un alto indice di accuratezza, si sono osservati un numero superiore di campioni classificati erroneamente rispetto all'*MLP*. I vantaggi e gli svantaggi dei due modelli sono stati combinati, considerando un *Sistema Target* basato sulle predizioni fornite da entrambi i modelli ai fini della classificazione binaria dell'input. Ciò ha permesso di garantire buone performance nella rilevazione di campioni anomali, favorendo un minor numero di *falsi negativi* a discapito di un aumento di *falsi positivi*. Il sistema target è al momento privo di protezioni rispetto ad input avversari, per cui vulnerabile rispetto a quest'ultimi. Nel prossimo capitolo si valuterà la misura in cui alcuni attacchi avversari possono essere condotti con successo sui due modelli, dunque la misura in cui input anomali opportunamente perturbati riescono ad ingannare il *Sistema Target* proposto.

Categoria	**Etichetta**	**Numero di istanze**
Benigni	BENIGN	1582566
Attacchi	Portscan	159066
	DoS Hulk	158468
	DDoS	95144
	Infiltration - Portscan	71767
	DoS GoldenEye	7567
	Botnet - Attempted	4067
	FTP-Patator	3972
	DoS Slowloris	3859
	DoS Slowhttptest - Attempted	3368
	SSH-Patator	2961
	DoS Slowloris - Attempted	1847
	DoS Slowhttptest	1740
	Web Attack - Brute Force - Attempted	1292
	Botnet	736
	Web Attack - XSS - Attempted	655
	DoS Hulk - Attempted	581
	DoS GoldenEye - Attempted	80
	Web Attack - Brute Force	73
	Infiltration - Attempted	45
	Infiltration	36
	SSH-Patator - Attempted	27
	Web Attack - XSS	18
	Web Attack - SQL Injection	13
	FTP-Patator - Attempted	12
	Heartbleed	11
	Web Attack - SQL Injection - Attempted	5
Totale		**2099976**

Tabella 2.1: Struttura dataset CICIDS2017 (improved)

Capitolo 3

Generazione input avversari

In questo capitolo si procederà alla generazione degli input avversari da sottoporre al Sistema Target, al fine di valutarne gli impatti dal punto di vista della decadenza delle performance di classificazione. Nel presente studio, si è scelto di porsi in uno scenario di tipo *Black-Box*: si assume cioè che, nel definire gli input avversari, non si abbia alcuna conoscenza circa il modello target, il che rappresenta una situazione più vicina ai casi reali. In particolare, ai fini dell'esecuzione degli attacchi, si è scelto di basarsi sulla libreria Python *Adversarial Robustness Toolbox* (ART) [64]. Si tratta di una libreria open-source sviluppata da *IBM* che implementa diversi attacchi avversari tra quelli più conosciuti. É incentrata principalmente sulla generazione di input avversari a partire da immagini, ma può agire anche su dati tabulari con successo [65][66]. Nel presente studio, si è scelto di considerare gli attacchi *Boundary* [40], HopSkipJump [39], Zeroth Order Optimization (ZOO) [35] e Query-efficient Black Box [67]. Ciascun attacco è stato condotto a partire da un sottoinsieme di 50.000 campioni anomali scelti in modo random, per ciascun attacco, dal dataset CICIDS2017_improved. Nelle sezioni successive verranno descritti in maggior dettaglio gli attacchi condotti, descrivendone i risultati in termini di *Tasso di Successo* nei confronti dei singoli modelli *DL-based* discussi nel capitolo precedente e sul Sistema Target.

OMISSIS

Capitolo 4

Sistema proposto di difesa da attacchi avversari

OMISSIS

Capitolo 5

Risultati sperimentali

In questo capitolo verranno valutate le performance del sistema proposto di difesa da attacchi avversari, dapprima analizzando i singoli layer che lo costituiscono e, successivamente, relativamente alla loro combinazione. I risultati verranno inoltre confrontati con quelli ottenuti a partire dal sistema di difesa *IDS-Anta*, proposto da *Barik et al.*[83], nell'ottica di fornire una comparazione rispetto ad un differente meccanismo di difesa da attacchi avversari nel dominio *NIDS*.

OMISSIS

Conclusioni

Nella presente tesi è stato analizzato l'impatto degli attacchi avversari nell'ambito dei sistemi di rilevamento delle intrusioni *ML/DL-based*, proponendo un sistema di difesa in grado di limitarne gli effetti.
Nel capitolo [1] è stata inizialmente presentata la problematica relativa agli attacchi avversari, cui gli algoritmi di *Machine Learning* e *Deep Learning* sono esposti per loro natura. Sono state descritte le principali tecniche messe in atto ai fini della generazione di input avversari, classificandole in base a diversi criteri come la conoscenza dell'attaccante circa il modello target (*White Box*, *Black Box*) od al criterio di manipolazione dell'input adoperato. A ciò è stata accompagnata l'analisi delle principali tecniche di difesa da poter mettere in atto per limitare gli effetti di attacchi avversari, descrivendone vincoli e modalità di applicazione e riferite sia al dominio *NIDS* che ad altri ambiti come *Computer Vision* o *NLP*. Un focus aggiuntivo ha riguardato l'ambito *NIDS*, rispetto al quale sono stati approfonditi diversi studi inerenti a questo ambito in termini di attacchi avversari e relative difese.
Nel capitolo [2] è stato descritto il processo di progettazione e implementazione di un *NIDS DL-based*. La realizzazione del sistema è avvenuta combinando due modelli *DL-based*, quali un *MLP* ed un *Autoencoder*. Entrambi i modelli sono stati addestrati a partire dal dataset *CICIDS2017* (versione improved), rispetto al quale sono state descritte anche tutte le fasi di *preprocessing* attuate prima della vera e propria fase di training dei modelli. Il problema è stato affrontato come un problema di classificazione binaria, in cui cioè ciascun modello è stato addestrato a distinguere campioni anomali da campioni benigni con un adeguato indice di accuratezza. Sono state valutate pertanto le performance dei singoli modelli in tal senso e, successivamente, anche della loro combinazione.
Il sistema così proprosto risultava inizialmente sprovvisto di difese da attacchi avversari, per cui nel capitolo [3] l'obiettivo è stato quello di valutare gli impatti che questi avrebbero avuto sul sistema target. Si è provveduto inizialmente alla generazione degli input avversari da sottoporre

al sistema *NIDS*, considerando quattro tipici attacchi *Black Box* quali *HopSkipJump*, *Boundary*, *ZOO* e *Query Efficient*. Gli input così ottenuti sono stati sottoposti sia ai singoli modelli che alla combinazione degli stessi. Si è osservato come l'*Attack Success Rate (ASR)* di ciascun attacco tra quelli considerati, sia risultato superiore nei confronti dei singoli modelli target piuttosto che sulla loro combinazione. Ciò ha suggerito come questo aspetto potesse rappresentare una prima linea di difesa, data dalla sottimissione degli input ad entrambi i modelli. Alcuni input avversari hanno tuttavia avuto successo nell'ingannare entrambi i modelli, il che ha rivelato la necessità di introdurre delle tecniche di difesa che agissero per limitare ulterirmente gli effetti di input avversari caratterizzati da alta trasferibilità.
Nei capitoli [4] e [5] è stata descritta la soluzione proposta in termini di difesa da attacchi avversari. In particolare, il meccanismo messo in atto, ha riguardato la combinazione di due livelli di difesa. Nel primo caso, si è proceduto al riaddestramento dei modelli target, introducendo nel *training dataset* anche input avversari. Ciò ha comportato una maggiore capacità dei singoli modelli nel riconoscere correttamente buona parte degli input avversari, anche se di diversa natura rispetto a quelli considerati per l'*Adversarial Training*. In secondo luogo, è stata introdotta la tecnica di *Mutation Testing*, la quale agisce a runtime sottoponendo l'input a diverse mutazioni del modello *DL-based* al fine di valutarne il *Tasso di Variazione dell'Etichetta*. Sulla base del valore ottenuto viene stabilito se si tratta di input avversario o benigno. Ciò ha portato ad un'ulteriore riduzione dell'*ASR* di ciascuno degli attacchi presi in esame, il che ha rivelato l'efficacia della combinazione delle due tecniche. Un aspetto negativo della soluzione proposta riguarda la presenza di *false segnalazioni* e la limitata quantità di input avversari sui quali è stata valutata l'efficacia del secondo layer di difesa.

Alcune importanti considerazioni vanno effettuate circa la realizzazione del presente studio. In prima analisi, è necessario considerare che gli input avversari sulla base dei quali è stato valutato il sistema proposto, sono stati generati nello *spazio delle features*. Tipicamente, in casi reali, un attaccante non ha la possibilità di modificare direttamente il vettore delle *features*, bensì agisce manipolando il traffico di rete in modo tale che le relative *features* estratte risultino in un input avversario per il sistema target. Un possibile approfondimento potrebbe dunque riguardare la verifica delle performance del *Sistema di Difesa* proposto rispetto ad attacchi avversari basati sulla manipolazione diretta del traffico di rete. In secondo luogo, la valutazione del secondo layer di difesa (*Mutation Testing*) è avvenuta sulla base di un insieme ristretto di campioni. Un'analisi più approfondita a partire da un numero maggiore di campioni avversari è

stata svolta singolarmente, mostrando come questa tecnica possa garantire buone performance nell'individuazione di campioni avversari. Un aspetto da tenere in considerazione in questo contesto, è quello relativo al numero di *false segnalazioni*. Si è osservato come questo possa essere controllato modificando opportunamente i parametri dell'algoritmo di *Mutation Testing* (*e.g. mutation rate, detection sensibility*). Nel nostro caso si è scelto di dare maggiore priorità al garantire un basso numero di *false segnalazioni*, per via del fatto che già al primo layer si ha una consistente quantità di input avversari correttamente individuati. Possibili future implementazioni del sistema di difesa proposto, potrebbero dunque considerare di agire opportunamente sui parametri del secondo layer al fine di stabilire un *trade-off* accettabile tra *falsi positivi* e *falsi negativi*.

Appendice

Source Code

https://github.com/fCucinella/tesi-magistrale

Data Sources

- **CICIDS2017 (improved) preprocessed**:

  https://www.kaggle.com/datasets/francescocucinella21/cicids2017-improved-preprocessed

- **Generated adversarial samples**:

  https://www.kaggle.com/datasets/francescocucinella21/advs-input

Elenco delle figure

Bibliografia

[1] U. H. Rao e U. Nayak. «Intrusion Detection and Prevention Systems». In: *The InfoSec Handbook: An Introduction to Information Security*. Berkeley, CA: Apress, 2014, pp. 225–243. URL: https://doi.org/10.1007/978-1-4302-6383-8_11.

[2] Y. LeCun, Y. Bengio e G. Hinton. «Deep learning». In: *nature* 521.7553 (2015), pp. 436–444.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow e R. Fergus. «Intriguing properties of neural networks». In: *arXiv preprint arXiv:1312.6199* (2013).

[4] O. Ibitoye, R. Abou-Khamis, M. e. Shehaby, A. Matrawy e M. O. Shafiq. «The Threat of Adversarial Attacks on Machine Learning in Network Security–A Survey». In: *arXiv preprint arXiv:1911.02621* (2019).

[5] D. J. Miller, Z. Xiang e G. Kesidis. «Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks». In: *Proceedings of the IEEE* 108.3 (2020), pp. 402–433.

[6] Z. Ahmad, A. Shahid Khan, C. Shiang e F. Ahmad. «Network intrusion detection system: A systematic study of machine learning and deep learning approaches». In: *Transactions on Emerging Telecommunications Technologies* 32 (gen. 2021).

[7] Y. Jia, M. Wang e Y. Wang. «Network intrusion detection algorithm based on deep neural network». In: *IET Information Security* 13.1 (2019), pp. 48–53.

[8] C. Yin, Y. Zhu, J. Fei e X. He. «A deep learning approach for intrusion detection using recurrent neural networks». In: *Ieee Access* 5 (2017), pp. 21954–21961.

[9] V. Agate, S. Drago, P. Ferraro e G. Lo Re. «Anomaly Detection for Reoccurring Concept Drift in Smart Environments». In: *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE. 2022, pp. 113–120.

[10] Y. Mirsky, T. Doitshman, Y. Elovici e A. Shabtai. «Kitsune: an ensemble of autoencoders for online network intrusion detection». In: *arXiv preprint arXiv:1802.09089* (2018).

[11] V. Agate, A. De Paola, S. Drago, P. Ferraro e G. Lo Re. «Enhancing IoT Network Security with Concept Drift-Aware Unsupervised Threat Detection». In: *2024 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2024.

[12] K. He, D. D. Kim e M. R. Asghar. «Adversarial machine learning for network intrusion detection systems: A comprehensive survey». In: *IEEE Communications Surveys & Tutorials* 25.1 (2023), pp. 538–566.

[13] K. Smagulova, L. Bacha, M. E. Fouda, R. Kanj e A. Eltawil. «Robustness and Transferability of Adversarial Attacks on Different Image Classification Neural Networks». In: *Electronics* 13.3 (2024), p. 592.

[14] I. Homoliak, M. Teknos, M. Ochoa, D. Breitenbacher, S. Hosseini e P. Hanacek. «Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach». In: *arXiv preprint arXiv:1805.02684* (2018).

[15] M. J. Hashemi, G. Cusack e E. Keller. «Towards evaluation of nidss in adversarial setting». In: *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*. 2019, pp. 14–21.

[16] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik e A. Swami. «Practical black-box attacks against machine learning». In: *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 2017, pp. 506–519.

[17] Q. Wu, Y. Liu, Q. Li, S. Jin e F. Li. «The application of deep learning in computer vision». In: *2017 Chinese Automation Congress (CAC)*. IEEE. 2017, pp. 6522–6527.

[18] T. P. Nagarhalli, V. Vaze e N. Rana. «Impact of machine learning in natural language processing: A review». In: *2021 third international conference on intelligent communication technologies and virtual mobile networks (ICICV)*. IEEE. 2021, pp. 1529–1534.

[19] N. Akhtar e A. Mian. «Threat of adversarial attacks on deep learning in computer vision: A survey». In: *Ieee Access* 6 (2018), pp. 14410–14430.

[20] X. Wang, J. Li, X. Kuang, Y.-a. Tan e J. Li. «The security of machine learning in an adversarial setting: A survey». In: *Journal of Parallel and Distributed Computing* 130 (2019), pp. 12–23.

[21] P. Mishra, V. Varadharajan, U. Tupakula e E. S. Pilli. «A detailed investigation and analysis of using machine learning techniques for intrusion detection». In: *IEEE communications surveys & tutorials* 21.1 (2018), pp. 686–728.

[22] S. Ennaji, N. El Akkad e K. Haddouch. «A powerful ensemble learning approach for improving network intrusion detection system (nids)». In: *2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS)*. IEEE. 2021, pp. 1–6.

[23] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat e S. Venkatraman. «Deep learning approach for intelligent intrusion detection system». In: *Ieee Access* 7 (2019), pp. 41525–41550.

[24] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao e J. Chen. «DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system». In: *Security and communication networks* 2020 (2020), pp. 1–11.

[25] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti e M. Colajanni. «Modeling realistic adversarial attacks against network intrusion detection systems». In: *Digital Threats: Research and Practice (DTRAP)* 3.3 (2022), pp. 1–19.

[26] C. Zhang, X. Costa-Perez e P. Patras. «Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms». In: *IEEE/ACM Transactions on Networking* 30.3 (2022), pp. 1294–1311.

[27] S. Gaglio, A. Giammanco, G. Lo Re e M. Morana. «Adversarial Machine Learning in e-Health: attacking a Smart Prescription System». In: *International Conference of the Italian Association for Artificial Intelligence (2021 AI*IA)*. Milan, Italy, dic. 2021.

[28] I. J. Goodfellow, J. Shlens e C. Szegedy. «Explaining and harnessing adversarial examples». In: *arXiv preprint arXiv:1412.6572* (2014).

[29] S.-M. Moosavi-Dezfooli, A. Fawzi e P. Frossard. «Deepfool: a simple and accurate method to fool deep neural networks». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[30] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik e A. Swami. «The limitations of deep learning in adversarial settings». In: *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE. 2016, pp. 372–387.

[31] A. Kurakin, I. Goodfellow e S. Bengio. «Adversarial machine learning at scale». In: *arXiv preprint arXiv:1611.01236* (2016).

[32] N. Carlini e D. Wagner. «Towards evaluating the robustness of neural networks». In: *2017 ieee symposium on security and privacy (sp)*. Ieee. 2017, pp. 39–57.

[33] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae e S. Ishii. «Distributional smoothing with virtual adversarial training». In: *arXiv preprint arXiv:1507.00677* (2015).

[34] F. Batool, F. Canino, F. Concone, G. Lo Re e Morana. «A Black-box Adversarial Attack on Fake News Detection Systems». In: *CEUR Workshop Proceedings*. Vol. 3731. IT. 2024.

[35] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi e C.-J. Hsieh. «Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models». In: *Proceedings of the 10th ACM workshop on artificial intelligence and security*. 2017, pp. 15–26.

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville e Y. Bengio. «Generative adversarial networks». In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[37] Z. Lin, Y. Shi e Z. Xue. «Idsgan: Generative adversarial networks for attack generation against intrusion detection». In: *Pacific-asia conference on knowledge discovery and data mining*. Springer. 2022, pp. 79–91.

[38] Q. Cheng, S. Zhou, Y. Shen, D. Kong e C. Wu. «Packet-level adversarial network traffic crafting using sequence generative adversarial networks». In: *arXiv preprint arXiv:2103.04794* (2021).

[39] J. Chen, M. I. Jordan e M. J. Wainwright. «Hopskipjumpattack: A query-efficient decision-based attack». In: *2020 ieee symposium on security and privacy (sp)*. IEEE. 2020, pp. 1277–1294.

[40] W. Brendel, J. Rauber e M. Bethge. «Decision-based adversarial attacks: Reliable attacks against black-box machine learning models». In: *arXiv preprint arXiv:1712.04248* (2017).

[41] D. Meng e H. Chen. «Magnet: a two-pronged defense against adversarial examples». In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2017, pp. 135–147.

[42] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh e P. McDaniel. «Ensemble adversarial training: Attacks and defenses». In: *arXiv preprint arXiv:1705.07204* (2017).

[43] A. Madry, A. Makelov, L. Schmidt, D. Tsipras e A. Vladu. «Towards deep learning models resistant to adversarial attacks». In: *arXiv preprint arXiv:1706.06083* (2017).

[44] N. Papernot, P. McDaniel, X. Wu, S. Jha e A. Swami. «Distillation as a defense to adversarial perturbations against deep neural networks». In: *2016 IEEE symposium on security and privacy (SP)*. IEEE. 2016, pp. 582–597.

[45] A. N. Bhagoji, D. Cullina e P. Mittal. «Dimensionality reduction as a defense against evasion attacks on machine learning classifiers». In: *arXiv preprint arXiv:1704.02654* 2.1 (2017).

[46] K. Grosse, P. Manoharan, N. Papernot, M. Backes e P. McDaniel. «On the (statistical) detection of adversarial examples». In: *arXiv preprint arXiv:1702.06280* (2017).

[47] J. Wang, G. Dong, J. Sun, X. Wang e P. Zhang. «Adversarial sample detection for deep neural network through model mutation testing». In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE. 2019, pp. 1245–1256.

[48] A. De Paola, S. Drago, P. Ferraro e G. Lo Re. «Detecting Zero-Day Attacks under Concept Drift: An Online Unsupervised Threat Detection System». In: *CEUR Workshop Proceedings - ITASEC 2024*. Vol. 3731. 2024.

[49] Z. Wang. «Deep learning-based intrusion detection with adversaries». In: *IEEE Access* 6 (2018), pp. 38367–38384.

[50] E. Alhajjar, P. Maxwell e N. Bastian. «Adversarial machine learning in network intrusion detection systems». In: *Expert Systems with Applications* 186 (2021), p. 115782.

[51] K. Yang, J. Liu, C. Zhang e Y. Fang. «Adversarial examples against the deep learning based network intrusion detection systems». In: *MILCOM 2018-2018 ieee military communications conference (MILCOM)*. IEEE. 2018, pp. 559–564.

[52] S. Guo, J. Zhao, X. Li, J. Duan, D. Mu e X. Jing. «A black-box attack method against machine-learning-based anomaly network flow detection models». In: *Security and Communication Networks* 2021 (2021), pp. 1–13.

[53] S. Zhang, X. Xie e Y. Xu. «A brute-force black-box method to attack machine learning-based systems in cybersecurity». In: *IEEE Access* 8 (2020), pp. 128250–128263.

[54] D. Shu, N. O. Leslie, C. A. Kamhoua e C. S. Tucker. «Generative adversarial attacks against intrusion detection systems using active learning». In: *Proceedings of the 2nd ACM workshop on wireless security and machine learning*. 2020, pp. 1–6.

[55] S. Lu, M. Wang, D. Wang, X. Wei, S. Xiao, Z. Wang, N. Han e L. Wang. «Black-box attacks against log anomaly detection with adversarial examples». In: *Information Sciences* 619 (2023), pp. 249–262.

[56] J. Clements, Y. Yang, A. A. Sharma, H. Hu e Y. Lao. «Rallying adversarial techniques against deep learning for network security». In: *2021 IEEE symposium series on computational intelligence (SSCI)*. IEEE. 2021, pp. 01–08.

[57] I. Goodfellow, Y. Bengio e A. Courville. *Deep learning*. MIT press, 2016.

[58] R. C. Aygun e A. G. Yavuz. «Network anomaly detection with stochastically improved autoencoder based models». In: *2017 IEEE 4th international conference on cyber security and cloud computing (CSCloud)*. IEEE. 2017, pp. 193–198.

[59] P. J. Werbos. «Backpropagation through time: what it does and how to do it». In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[60] X. Li, W. Chen, Q. Zhang e L. Wu. «Building auto-encoder intrusion detection system based on random forest feature selection». In: *Computers & Security* 95 (2020), p. 101851.

[61] Y. Wang, H. Yao e S. Zhao. «Auto-encoder based dimensionality reduction». In: *Neurocomputing* 184 (2016), pp. 232–242.

[62] L. Liu, G. Engelen, T. Lynar, D. Essam e W. Joosen. «Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018». In: *2022 IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2022, pp. 254–262.

[63] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani et al. «Toward generating a new intrusion detection dataset and intrusion traffic characterization.» In: *ICISSp* 1 (2018), pp. 108–116.

[64] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig et al. «Adversarial Robustness Toolbox v1. 0.0». In: *arXiv preprint arXiv:1807.01069* (2018).

[65] H. Jmila e M. I. Khedher. «Adversarial machine learning for network intrusion detection: A comparative study». In: *Computer Networks* 214 (2022), p. 109073.

[66] O. Ibitoye, O. Shafiq e A. Matrawy. «Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks». In: *2019 IEEE global communications conference (GLOBECOM)*. IEEE. 2019, pp. 1–6.

[67] A. Ilyas, L. Engstrom, A. Athalye e J. Lin. «Black-box adversarial attacks with limited queries and information». In: *International conference on machine learning*. PMLR. 2018, pp. 2137–2146.

[68] V. Agate, F. M. D'Anna, A. De Paola, P. Ferraro, G. Lo Re e M. Morana. «A Behavior-Based Intrusion Detection System Using Ensemble Learning Techniques.» In: *CEUR Workshop Proceedings - ITASEC 2022*. 2022.

[69] M. M. Ahsan, M. Mahmud, P. K. Saha, K. D. Gupta e Z. Siddique. *Effect of data scaling methods on machine learning algorithms and model performance. Technologies 9 (3), 52*. 2021.

[70] Y. Lee, S.-H. Oh e M. W. Kim. «An analysis of premature saturation in back propagation learning». In: *Neural networks* 6.5 (1993), pp. 719–728.

[71] D. C. Marcu e C. Grava. «The impact of activation functions on training and performance of a deep neural network». In: *2021 16th International Conference on Engineering of Modern Electric Systems (EMES)*. IEEE. 2021, pp. 1–4.

[72] R.-Y. Sun. «Optimization for deep learning: An overview». In: *Journal of the Operations Research Society of China* 8.2 (2020), pp. 249–294.

[73] A. Shrestha e A. Mahmood. «Review of deep learning algorithms and architectures». In: *IEEE access* 7 (2019), pp. 53040–53065.

[74] T. Dozat. «Incorporating nesterov momentum into adam». In: (2016).

[75] P. D. Lax e M. S. Terrell. *Calculus with applications*. Vol. 4. Springer, 2014.

[76] T. Salimans, J. Ho, X. Chen, S. Sidor e I. Sutskever. «Evolution strategies as a scalable alternative to reinforcement learning». In: *arXiv preprint arXiv:1703.03864* (2017).

[77] K. Ren, T. Zheng, Z. Qin e X. Liu. «Adversarial attacks and defenses in deep learning». In: *Engineering* 6.3 (2020), pp. 346–360.

[78] N. Carlini, G. Katz, C. Barrett e D. L. Dill. «Ground-truth adversarial examples». In: (2018).

[79] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy et al. «Technical report on the cleverhans v2. 1.0 adversarial examples library». In: *arXiv preprint arXiv:1610.00768* (2016).

[80] V. Agate, F. Concone, A. De Paola, P. Ferraro, G. Lo Re e M. Morana. «Bayesian Modeling for Differential Cryptanalysis of Block Ciphers: A DES Instance». In: *IEEE Access* 11 (2023), pp. 4809–4820.

[81] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao et al. «Deepmutation: Mutation testing of deep learning systems». In: *2018 IEEE 29th international symposium on software reliability engineering (ISSRE)*. IEEE. 2018, pp. 100–111.

[82] A. Wald. *Sequential analysis*. Courier Corporation, 2004.

[83] K. Barik e S. Misra. «IDS-Anta: An open-source code with a defense mechanism to detect adversarial attacks for intrusion detection system». In: *Software Impacts* 21 (2024), p. 100664.

[84] A. Alsarhan, M. Alauthman, E. Alshdaifat, A.-R. Al-Ghuwairi e A. Al-Dubai. «Machine Learning-driven optimization for SVM-based intrusion detection system in vehicular ad hoc networks». In: *Journal of Ambient Intelligence and Humanized Computing* 14.5 (2023), pp. 6113–6122.

[85] C. Kalkanlı e A. Özgür. «Asymptotic Performance of Thompson Sampling for Batched Multi-Armed Bandits». In: *IEEE Transactions on Information Theory* (2023).

[86] V. Agate, P. Ferraro, G. Lo Re e S. K. Das. «BLIND: A privacy preserving truth discovery system for mobile crowdsensing». In: *Journal of Network and Computer Applications* (2023), p. 103811. URL: https://www.sciencedirect.com/science/article/pii/S1084804523002308.

[87] L. Dekel, I. Leybovich, P. Zilberman e R. Puzis. «MABAT: A multi-armed bandit approach for threat-hunting». In: *IEEE Transactions on Information Forensics and Security* 18 (2022), pp. 477–490.

Ringraziamenti

Rivolgo un ringraziamento a tutti coloro i quali mi hanno accompagnato in questo percorso universitario e di scrittura della tesi. Al relatore e correlatore, che mi hanno seguito nella fase progettuale e di stesura dell'elaborato, rivolgendomi preziosi suggerimenti. Ai miei colleghi di lavoro, i quali mi hanno dato la possibilità di dedicare maggior tempo allo studio, consentendomi di raggiungere la fine di questo percorso. Ai miei genitori, che hanno sempre supportato e appoggiato le mie scelte, dandomi la forza e la volontà di perseguire i miei obiettivi. A Soraja, il mio punto di riferimento, la persona che mi è stata vicina sin dall'inizio e senza la quale sarebbe risultato più difficile il raggiungimento di questo traguardo. Ringrazio infine le mie sorelle e tutta la mia famiglia, le persone più importanti della mia vita che mi hanno sempre trasmesso coraggio e fiducia in me stesso.