Adversarial Machine Learning per la generazione di attacchi ai sistemi di rilevamento delle intrusioni

Tesi di Laurea Magistrale in Ingegneria Informatica

Giuliano Solaro

Relatore: Prof. Giuseppe Lo Re

Correlatori: Prof. Alessandra De Paola, Ing. Pierluca Ferraro, Ing. Vincenzo Agate

NDS LAB - Networking and Distributed Systems
http://www.diid.unipa.it/networks/

Adversarial Machine Learning per la generazione di attacchi ai sistemi di rilevamento delle intrusioni

TESI DI LAUREA DI
GIULIANO SOLARO

RELATORE
Ch.mo Prof. GIUSEPPE LO RE

CORRELATORI
Prof. ALESSANDRA DE PAOLA
Ing. PIERLUCA FERRARO
Ing. VINCENZO AGATE

Sommario

Gli attacchi informatici, nell'ultimo decennio, hanno registrato, a livello globale, un incremento significativo sotto il profilo della frequenza, dell'impatto e delle dimensioni, al punto che risulta necessario sviluppare costantemente nuove tecniche di difesa che riescano a contrastarli. Proprio per questo, è pratica comune utilizzare delle tecniche basate su machine learning, in modo da addestrare dei sistemi capaci di rilevare repentinamente possibili attacchi. Tuttavia, anche questi sistemi hanno delle vulnerabilità, che possono essere sfruttate creando degli input specifici, detti campioni o esempi avversari, che hanno lo scopo di causare un output errato.

In questo lavoro verrà sviluppata una tecnica di attacco per la generazione di campioni avversari, con lo scopo di superare il controllo dei sistemi di rilevamento delle intrusioni basati su machine learning. Il punto cardine consiste nell'inserire una perturbazione in alcuni attributi di un campione malevolo e modificare gli altri in modo da mantenere intatta la caratteristica dell'attacco. Per fare ciò, si imporranno dei vincoli durante la fase di modifica. L'introduzione della perturbazione, invece, seguirà una tecnica basata sul confine delle classi, in modo da generare campioni che siano il più possibile vicini a quello originario, ma appartengano alla classe obbiettivo. Inoltre, verrà utilizzato un modello sostituto con lo scopo di rimpiazzare

il sistema da attaccare durante la fase di creazione dei campioni avversari.

Per l'implementazione dell'attacco è stato utilizzato il dataset CICIDS2017, dal quale sono stati selezionati sia i campioni necessari ad addestrare il modello sostituto che quelli malevoli da perturbare. I risultati indicano come il metodo sia in grado di trovare dei buoni campioni avversari utilizzando un numero quasi irrisorio di query da rivolgere al sistema vittima.

Indice

Introduzione

Nel presente lavoro verranno descritte e studiate le principali tecniche di adversarial machine learning sui sistemi di rilevamento delle intrusioni. Verranno inoltre presentate alcune tecniche di generazione di esempi avversari esistenti in letteratura scientifica dalle quali è stata presa ispirazione per la creazione del nuovo metodo oggetto della tesi.

L'ambito nel quale si sviluppa la tesi è abbastanza innovativo. Sebbene l'adversarial machine learning esista già da molto tempo, presenta ancora molte sfaccettature che rendono difficile la sua applicazione sui sistemi di rilevamento delle intrusioni. Questo è dovuto al fatto che l'approccio è stato sviluppato per l'addestramento delle reti neurali convoluzionali, le quali lavorano su immagini, notevolmente più semplici da gestire rispetto a file binari o statistiche di flusso di comunicazioni di rete. Di conseguenza la loro applicazione sugli IDS risulta ancora oggi un campo nuovo e, per la maggior parte, inesplorato.

Il cardine del lavoro svolto è lo sviluppo di un algoritmo di attacco agli IDS che consiste nella generazione di campioni speciali, detti campioni o esempi avversari, che portano il sistema vittima ad effettuare una classificazione errata. Più nello specifico verrà introdotta una perturbazione a un campione malevolo in modo che il sistema vittima lo classifichi come benevolo e, di conseguenza, non crei mai una allerta sull'attacco in corso. Per garantire che il campione generato risulti ancora malevolo anche dopo l'introduzione della perturbazione, sono stati imposti dei vincoli nella modifica. Inoltre, si utilizza un modello sostituto che rimpiazza l'IDS vittima durante la fase di creazione del campione avversario riducendo la probabilità di allertare il sistema.

La tesi è suddivisa in cinque capitoli.

Il capitolo 1 ha lo scopo presentare i sistemi di rilevamento delle intrusioni, di indicare le varie tipologie esistenti, di descrivere i rispettivi funzionamenti, di elencare i principali metodi di implementazione e di mostrare alcune tecniche, dette tecniche di evasione, che creano problemi nel rilevamento delle anomalie.

Il capitolo 2 ha lo scopo di descrivere l'Adversarial Machine Learning, la sua applicazione contro gli IDS e le diverse tecniche di attacco e di difesa conosciute in letteratura scientifica.

Il capitolo 3 entrerà più nel dettaglio riguardo all'adversarial machine learning e verranno analizzati alcuni dei principali approcci.

Nel capitolo 4 verrà presentato un nuovo sistema di generazione degli esempi avversari, si descriveranno nel dettaglio i vincoli imposti nell'introduzione della perturbazione e la particolare tecnica di implementazione dell'attacco.

Infine, nel capitolo 5 verranno descritti i dettagli dell'implementazione in riferimento al modello sostituto e all'applicazione dei vincoli. Verranno inoltre presentanti tutti i valori associati ai parametri dell'attacco e i risultati sperimentali ottenuti tramite il metodo proposto.

Capitolo 1. Sistemi di rilevamento delle intrusioni

Questo capitolo ha lo scopo di presentare i sistemi di rilevamento delle intrusioni, di indicare le varie tipologie esistenti, di descrivere i rispettivi funzionamenti, di elencare i principali metodi di implementazione e di mostrare alcune tecniche che creano problemi nel rilevamento delle anomalie.

Un sistema di rilevamento delle intrusioni, detto Intrusion Detection System o, più semplicemente, IDS, è un sistema di monitoraggio che rileva attività anomale, come le intrusioni (qualsiasi tipo di attività non autorizzata che causa danni a un sistema informativo), e che genera degli allarmi ogni volta che queste vengono rilevate. Il suo compito quindi, a differenza dei firewall, non è quello di bloccare qualsiasi connessione o codice considerati malevoli, ma quello di generare degli avvisi che verranno esaminati dai responsabili della sicurezza di quel sistema, i quali investigheranno il problema e sceglieranno le azioni da svolgere per rimediare alla minaccia.

Ad oggi il loro utilizzo è considerato necessario in quanto le nuove tecniche di attacco sono diventate talmente sofisticate da rendere obsoleti i normali firewall.

1.1 Tipologie di IDS

Esistono diverse tipologie di IDS [41]. In riferimento al modo in cui lavorano si distinguono i **Signature-Based IDS** o **SIDS** e gli **Anomaly-Based IDS** o **AIDS**. Ragionando invece sulla loro disposizione si individuano gli **Host-Based IDS** o **HIDS** e i **Network-Based IDS** o **NIDS**.

1.1.1 SIDS e AIDS

Un **SIDS** si basa su tecniche di riconoscimento di pattern al fine di rilevare una tipologia di attacco conosciuto [43]. Un componente cardine di questa architettura è il **Database delle Signature**: questo contiene le firme di tutte le intrusioni conosciute. La firma di una intrusione è una sequenza continua di bytes, spesso tradotta in stringa alfanumerica, che la identifica univocamente. Ogni volta che viene rilevata una nuova intrusione se ne calcola la firma e si inserisce nel database in modo da mantenerlo

aggiornato. Il compito di un SIDS è quello di analizzare un comportamento, calcolarne la firma e ricercarne delle corrispondenze all'interno del Database delle Signatures, trovate le quali viene generato un allarme.

Per quanto riguarda la loro efficienza, i SIDS normalmente hanno un eccellente grado di rilevamento di intrusioni già conosciute e un basso tasso di segnalazione di falsi positivi, cioè comportamenti normali erroneamente rilevati come malevoli. Tuttavia, hanno molte difficoltà nel riconoscere nuove tecniche di intrusione, come gli attacchi Zero-Days, in quanto, per queste, non esiste nessun signature all'interno del database (verrebbe inserita per la prima volta solo dopo che questa intrusione è stata commessa e rilevata) [44]. Inoltre, un ulteriore svantaggio è la necessità di aggiornare costantemente il database dei signature.

Oggi questo approccio al rilevamento delle intrusioni è stato quasi totalmente abbandonato in favore degli AIDS. Questo perché gli attacchi più comuni sono gli Zero-Days, sui quali non si ha alcuna conoscenza a priori, e i malware polimorfi che, cambiando spesso forma, modificano la loro firma rendendone impossibile l'identificazione.

Un **AIDS** si basa sull'idea che un comportamento normale differisca da uno anomalo. La soluzione è allora quella di predisporre dei modelli di comportamento degli utenti, cosa che in passato era svolta tramite euristiche. Data la grandissima difficoltà nel trovare quelle migliori, differendo il comportamento degli utenti da sistema a sistema, si è fatta strada l'idea di utilizzare delle tecniche di Machine Learning [45] [46] che, autonomamente, sono in grado di apprendere la differenza tra i vari tipi di comportamento. Proprio grazie a questo approccio, oggi gli AIDS sono in grado di risolvere la maggior parte dei problemi dei SIDS, come quello di rilevare gli attacchi Zero-Day o le intrusioni relativamente nuove. Inoltre, sono difficili da superare dal momento che un attaccante non può conoscere quale comportamento è classificato come anomalo senza prima allertare il sistema.

Gli AIDS presentano però diversi svantaggi. Tra i principali vi è la necessità di addestrare il modello di Machine Learning, per il quale bisogna disporre di una grandissima quantità di dati in modo che l'IDS sia in grado di approssimare, in maniera più accurata possibile, i modelli di comportamento. Inoltre, l'addestramento è computazionalmente oneroso e richiede molto tempo, il che è poco adatto al mondo estremamente dinamico della sicurezza informatica. Sarebbe allora necessario ottenere periodicamente nuovi campioni da utilizzare per riaddestrare il modello di Machine Learning. Un ulteriore svantaggio è l'elevato tasso di falsi positivi dovuto al

fatto che non esiste una linea di confine netta tra comportamento usuale e comportamento malevolo; infatti, esistono comportamenti al limite che tendono ad essere classificati erroneamente dall'AIDS. La figura 1.1 traccia i profili di comportamento tra intrusi e utenti autorizzati. I due comportamenti appaiono quasi del tutto differenti e, di conseguenza, facilmente distinguibili. Solo la parte grigia centrale, dove le due campane si sovrappongono, indica i comportamenti al limite, ovvero quelle parti dove un intruso e un utente autorizzato si comportano allo stesso modo: proprio in questo punto l'IDS ha scarsa capacità di classificazione comportando un elevato tasso di falsi positivi.

In generale, oggi, SIDS e AIDS sono utilizzati insieme, uno a complemento dell'altro, in modo da aggiungere strati di protezione, migliorare l'efficienza nel rilevamento di anomalie e, di conseguenza, incrementare il livello di sicurezza del sistema.

1.1.2 HIDS e NIDS

Un **HIDS** [47] è un IDS interno all'host. Viene distribuito su un particolare endpoint e il suo compito è quello di proteggerlo dalle minacce interne ed esterne.

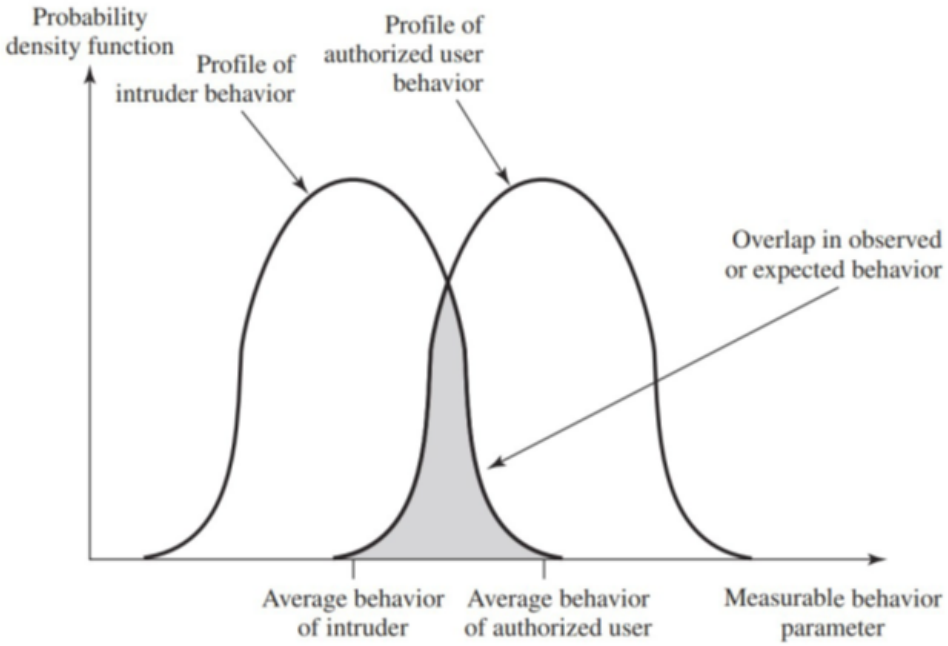

Figura 1.1: Profilo dei comportamenti di intrusi e utenti autorizzati

Fonte: [34] Cap 22

Gli HIDS hanno in generale una efficienza molto elevata in quanto godono di molti privilegi quale l'accesso al sistema operativo, alla memoria o ai registri. Proprio per questo possono avere la capacità di monitorare il traffico di rete da e verso la macchina, osservare i processi in esecuzione, ispezionare i registri del sistema e i file di log. Inoltre, dato che sono interni all'host, sono in grado di esaminare anche i dati cifrati.

Il problema è che la visibilità di un IDS basato su host è limitata solamente alla macchina sul quale viene distribuito, anche se, come detto, questa è molto approfondita. Un ulteriore svantaggio è dato dal fatto che la sua azione è ritardata: a causa del suo posizionamento entra in azione solamente quando l'intruso, o il malware, si trova già all'interno dell'host; quindi, ha già avuto a disposizione del tempo per entrare in esecuzione ed effettuare azioni potenzialmente dannose.

Un **NIDS** [48], al contrario, è una soluzione IDS basata sulla rete ed è progettata per monitorarla e, di conseguenza, proteggere tutti i dispositivi a essa connessi. Ha visibilità su tutto il traffico che scorre attraverso questa ed effettua il controllo basandosi sui contenuti, sui metadati dei pacchetti, quando non cifrati, e sulle statistiche del flusso di ogni comunicazione. Questo punto di vista è quindi molto più ampio di quello degli HIDS e fornisce la capacità di rilevare minacce diffuse. Inoltre, data la loro posizione, possono rilevare una anomalia prima che questa arrivi all'host vittima.

Tuttavia, questi sistemi hanno anche degli svantaggi: non hanno visibilità all'interno degli endpoint che proteggono e godono di pochi permessi rispetto agli HIDS. Inoltre, esistono numerose **tecniche di evasione**, come la cifratura, la frammentazione dei pacchetti e il flooding della rete, che impediscono ai NIDS di effettuare correttamente la propria analisi.

1.2 Disposizione di un NIDS

Il posizionamento di un IDS nella rete è strategico e dipende da molti fattori [34]. In base a questo l'IDS sarà soggetto a carichi di lavoro diversi, avrà scopi differenti e scoprirà attacchi di tipo differente.

Una prima opzione è quella di disporlo **all'esterno del firewall** principale dell'organizzazione, indicato nel punto **2** nella figura 1.2. In questo caso l'IDS viene raggiunto dall'intero traffico di rete; quindi, deve analizzare la totalità di pacchetti.

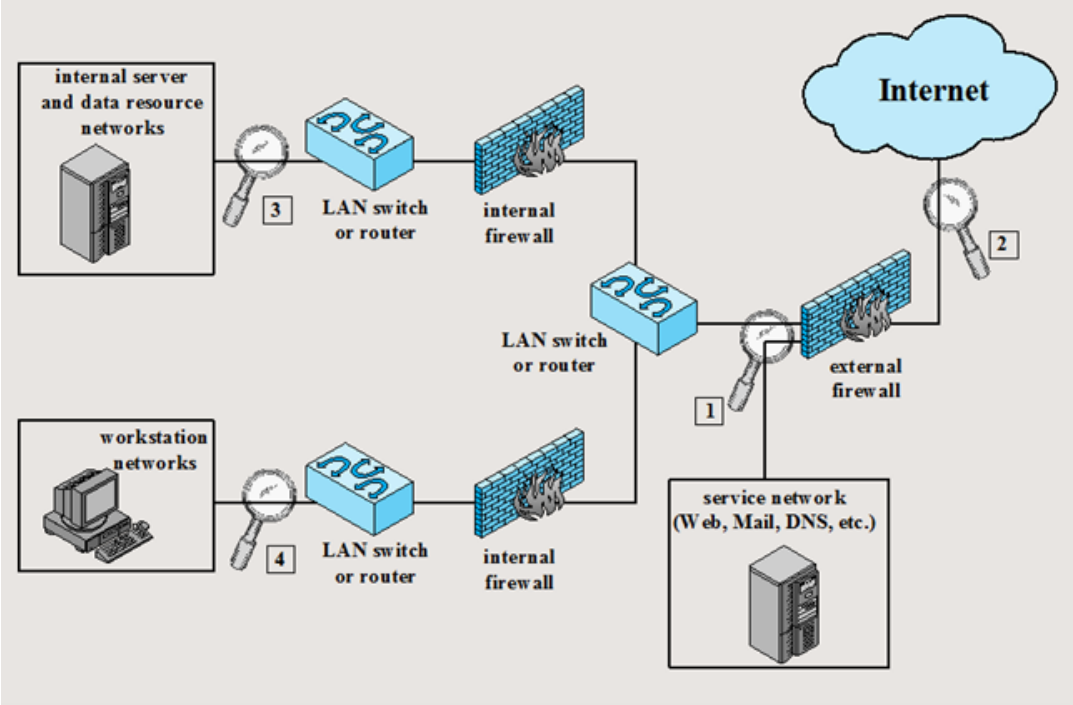

Figura 1.2: Posizionamento di un NIDS nella rete

Fonte : *https://slidetodoc.com/cse-4905-firewalls-intrusion-detection-systems-8-1/*

Questo comporta ovviamente un elevato carico di lavoro che, anche a causa della lentezza nell'elaborazione, lo rende vulnerabile alle tecniche di evasione di tipo Flooding. Inoltre, ricordando che il compito di un firewall è quello di scartare dei pacchetti di rete basandosi su determinate politiche, l'IDS analizzerà anche tutto ciò che verrà successivamente scartato. Di conseguenza il posizionamento esterno è poco conveniente ai fini della sicurezza.

Questo approccio ha anche delle utilità: consente di documentare lo sforzo di sicurezza necessario per difendere un sistema, in quanto vengono analizzati tutti i tentativi; consente di ottenere maggiori dati necessari per la creazione di statistiche sugli attacchi o per l'addestramento di altri IDS basati su Machine Learning.

Una seconda opzione, indicata dal punto **1** in figura 1.2, è posizionarlo **nella zona demilitarizzata**, cioè la parte della rete compresa tra il firewall esterno e i firewall interni che espone servizi a una rete esterna non sicura. In questa posizione arriveranno all'IDS solamente i pacchetti già filtrati dal firewall esterno, ma non ancora dai firewall interni. Lo scopo di questo posizionamento è quello di rilevare gli attacchi che sono diretti ai servizi offerti dalla zona demilitarizzata, ma non quelli

diretti agli endpoints, in quanto si cadrebbe negli stessi problemi di sovraccarico descritti precedentemente.

Una terza opzione, indicata nei punti **3** e **4** in figura 1.2, è quella **dietro i firewall interni**. Questa è la posizione più interna possibile ed è anche la più riparata in quanto tutti i pacchetti che arrivano sono già stati filtrati da più firewall. Lo scopo di questo posizionamento è quello di rilevare le intrusioni dirette ai server interni e ai database (punto 3), oppure quello di proteggere ogni singola workstation connessa alla rete (punto 4).

1.3 Implementazione degli AIDS

I metodi di implementazione degli AIDS possono essere divisi in tre macrogruppi, come mostrato in figura 1.3:

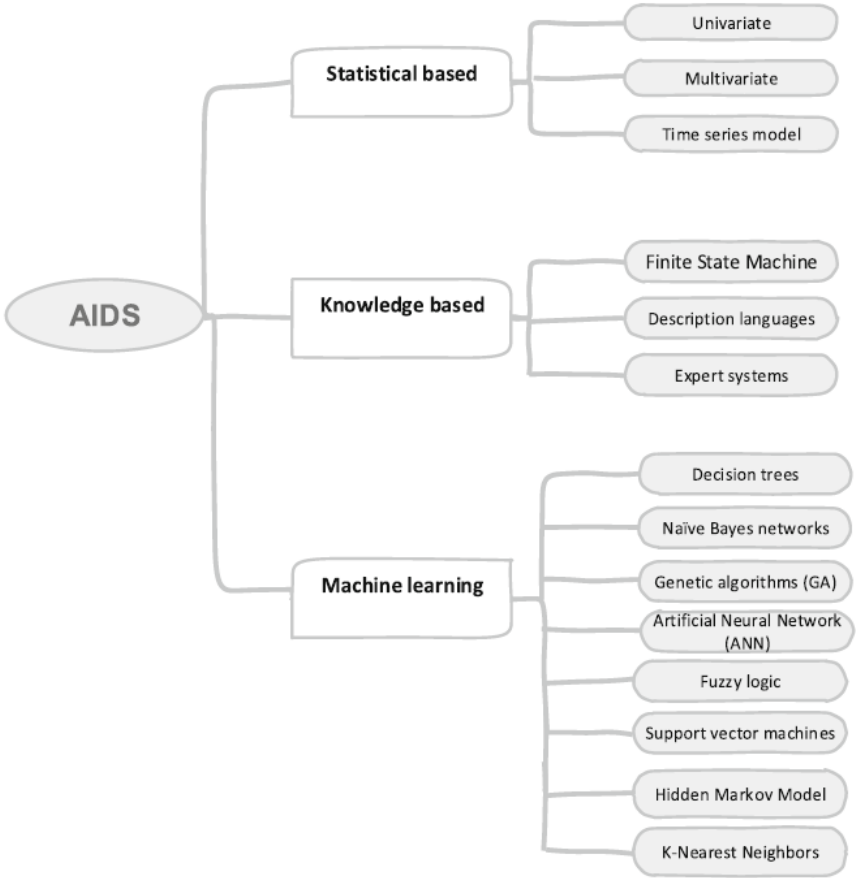

Figura 1.3: Metodi di implementazione degli AIDS

Fonte: [33]

- **Metodi basati su approcci statistici** [49]: consistono nel raccogliere ed esaminare una serie di dati necessari alla creazione di modelli statistici che riguardano il normale comportamento degli utenti
- **Metodi basati su basi di conoscenza** [50] [51]: cercano di identificare il tipo di azione in base a delle regole scritte nelle basi di conoscenza
- **Metodi basati su Machine Learning** [45] [46]: acquisiscono dei pattern di variazione dei dati di input per poi effettuare delle predizioni

1.3.1 Tecniche Statistiche

Come accennato precedentemente, le tecniche statistiche [49] consistono nella creazione di un modello probabilistico delle azioni normali degli utenti, in modo da individuare potenziali intrusioni interpretabili come eventi a bassa probabilità. Si basano infatti su modelli statistici come la media, mediana, moda e deviazione standard dei pacchetti. In altre parole, queste tecniche ispezionano ogni singolo pacchetto e lo valutano secondo il modello probabilistico creato.

Esistono tre approcci differenti:

- **Univariate** [52]: i dati hanno una sola variabile. L'approccio è utilizzato quando il profilo statistico creato si basa su una sola misura di comportamento.
- **Multivariate** [52]: lo scopo è quello di comprendere le relazioni che esistono tra le variabili. L'approccio è utilizzato quando è possibile effettuare una migliore classificazione lavorando sulle combinazioni piuttosto che su dati singoli.
- **Time Series Model** [58]: una serie temporale è una sequenza di osservazioni ottenute in istanti di tempo differenti. Una osservazione è malevola se la probabilità che si verifichi in quell'istante di tempo è bassa, altrimenti è benevola.

1.3.2 Tecniche basate su basi di conoscenza

L'approccio [50] [51] richiede la creazione di basi di conoscenza che riflettono il normale (legittimo) traffico nella rete. Le azioni che differiscono rispetto a questo profilo standard sono considerate intrusioni.

Il principale vantaggio è dato dal fatto che, essendo le basi create da esperti, hanno un basso tasso di falsi positivi in quanto il sistema conosce bene tutti i comportamenti normali. Tuttavia, nei sistemi dinamici, come quelli relativi alla cybersecurity, la base di conoscenza dovrebbe essere aggiornata regolarmente, il che risulta complicato data la sua complessità.

Anche per queste tecniche esistono tre approcci differenti:

- **Macchine a Stati Finiti** [59]: le macchine a stati finiti consentono di rappresentare un flusso di esecuzione. In queste vi sono una serie di stati e transizioni. In base al valore in input, derivato dal comportamento dell'utente, si effettua una transizione da uno stato all'altro. Se la transizione ci porta nello stato di allerta significa che l'input è da segnalare come una intrusione.
- **Description Language** [60]: il linguaggio descrive, attraverso una particolare sintassi, le regole che possono essere usate per specificare le caratteristiche di un attacco definito.
- **Sistemi Esperti** [61]: un sistema esperto conosce una serie di regole che definiscono un attacco. Queste regole vengono spesso scritte a mano.

1.3.3 Tecniche basate su Machine Learning

Il machine learning è il processo di estrazione di informazioni da una grande quantità di dati nella forma di set di regole o complicate funzioni matematiche. Il principale vantaggio degli approcci basati su Machine Learning [45] [46] è una efficienza maggiore rispetto ai precedenti e, soprattutto, una minore conoscenza apportata da persone. Tra i principali algoritmi si hanno: clustering, reti neurali, regole di associazione, alberi decisionali, algoritmi genetici. Tutti questi modelli possono essere addestrati usando dei dataset contenenti dati relativi a intrusioni.

Ai fini della creazione di un IDS, in letteratura, sono stati utilizzati diversi algoritmi di machine learning. Per quanto riguarda gli algoritmi basati su **addestramento supervisionato**, ovvero quegli algoritmi in cui i dati in input sono etichettati e la rete apprende le relazioni tra un input e l'output indicato dall'etichetta, i principali metodi sono:

- **Alberi Decisionali**: l'algoritmo crea un albero in cui i nodi interni effettuano dei test sugli attributi e i nodi foglia specificano il risultato della classificazione (True o False). Nella scelta degli attributi di test si sfrutta il

concetto di informazione (o incertezza): più nello specifico, a ogni passo si sceglie di effettuare un test su quell'attributo che fornisce una quantità di informazione elevata. Nell'ambito degli IDS, ogni pacchetto in input viene testato sull'albero, che predice se è malevolo oppure normale.

Il vantaggio degli alberi decisionali è la loro semplicità di costruzione e l'alta accuratezza nella classificazione. Lo svantaggio riguarda invece l'information gain che è spesso sbilanciato in favore dei rami più profondi: gli alberi poco profondi hanno elevata capacità di generalizzazione, inoltre, dato che sono piccoli, è facile per un esperto estrarre delle regole che possono rendere la classificazione più intuitiva; gli alberi molto profondi, invece, hanno una elevata accuratezza nella classificazione di dati conosciuti, ma perdono la capacità di generalizzazione, necessaria per classificare correttamente quelli sconosciuti. Proprio per questo gli alberi troppo profondi vengono spesso potati in modo da lasciare un margine di errore che migliora la capacità di generalizzazione.

Esempi di applicazione sono [3] e [2] in cui vengono studiati i server DNS per poter comprendere quali domini sono coinvolti in attività malevole. Il sistema si basa su 5 componenti principali e i dati utilizzati hanno 15 attributi, tutti usati nella costruzione dell'albero decisionale. Grazie a degli esperimenti si è notato che per garantire prestazioni ottimali il primo addestramento per il sistema deve durare sette giorni; poi il classificatore viene riaddestrato ogni giorno sui nuovi dati raccolti dai server. L'accuratezza finale del sistema è del 98.5% e il rate di falso allarme FAR (paragrafo 1.4) è di 0.9%. Ancora oggi è considerato uno dei migliori approcci per il riconoscimento di domini malevoli.

- **Naive Bayes**: il metodo utilizza il teorema di Bayes per calcolare la probabilità di appartenenza di un dato in input a una certa classe. Il calcolo delle probabilità congiunte degli attributi in input risulta però troppo complicato, quindi, è necessario applicare una forte ipotesi Naive, ovvero una ipotesi di indipendenza statistica di tutti gli attributi. Si può facilmente comprendere che questa ipotesi è una forzatura, in quanto gli attributi di un dato sono spesso statisticamente dipendenti tra loro, ma è necessario farla per semplificare l'addestramento. Il risultato è la creazione di un modello con scarse prestazioni, poco migliore della scelta casuale. Proprio per questo motivo oggi Naive Bayes viene utilizzato come benchmark per la valutazione di altri modelli.

In ambito IDS, la tecnica si basa sul fatto che certi attributi hanno differenti valori di probabilità negli attacchi e nei comportamenti normali: preso un dato in input, se un suo attributo ha un valore di probabilità completamente differente rispetto a ciò che è considerato normale, il dato viene classificato come attacco.

- **Evolutionary Computation**: comprende la programmazione genetica GP e gli algoritmi genetici GA. Sono approcci euristici applicati ad algoritmi di ottimizzazione basati sul principio di evoluzione. Gli algoritmi puntano a modificare i membri di una generazione applicando operazioni di selezione, crossover e mutazione sugli elementi. Inizialmente la popolazione è generata casualmente e, tramite una funzione di fitness, si testa ogni individuo: quelli con valore di fitness elevato vengono selezionati nella generazione successiva (sono circa il 10% degli individui che costituiranno la generazione successiva), gli altri invece saranno combinati tramite l'operazione di crossover, in cui parti casuali di elementi vengono scambiate tra loro, questi andranno a formare circa il 90% della generazione successiva. Infine, molto raramente (meno dell'1% delle volte), viene applicata una mutazione in cui una parte di un elemento è sostituita da un valore generato casualmente.

  La principale differenza tra GP e GA consiste nel fatto che nel GP gli individui sono rappresentati come programmi, ad esempio programmi LISP, mentre nel GA sono rappresentati come sequenze di bit; quindi, le operazioni di crossover o mutazione sono molto semplici.

  Lo scopo della evolutionary computation in ambito IDS è la creazione di un individuo capace di classificare comportamenti normali e anomali.
- **Reti Neurali Artificiali (ANN**): classica tecnica di Machine/Deep Learning che utilizza le reti neurali. Si danno in input degli esempi e si effettua l'addestramento tramite gli algoritmi di back-propagation e di discesa lungo il gradiente. Ad oggi è l'approccio basato su machine learning più apprezzato in quanto, grazie alla presenza di molti layer nascosti, la rete neurale è in grado di creare funzioni altamente non lineari che comportano una elevata efficienza e capacità di generalizzazione. Lo svantaggio è che sono strettamente dipendenti dai dati con cui vengono addestrati. Proprio per questo, sebbene siano in grado di riconoscere attacchi frequenti, non hanno una efficienza elevata nel riconoscere quelli meno frequenti a causa della mancanza di record nei dataset di

addestramento. Questo problema è molto grave in quanto, in sicurezza informatica, gli attacchi più rari sono quelli che causano danni maggiori. Degli esempi di applicazione sono [6] e [4]. In [6], Cannady ha utilizzato un classificatore multi-label per identificare anomalie. Vengono selezionati nove attributi: ID protocollo, porta sorgente, porta di destinazione, indirizzo sorgente, indirizzo di destinazione, tipo di ICMP, codice ICMP, lunghezza complessiva dei dati e dati. La rete è stata addestrata su dati normali e dati anomali. L'errore quadratico medio (RMSE) raggiunto è stato di circa 0.058 e 0.07 rispettivamente per training e testing e una accuratezza del 93%.

In [4] si hanno invece 5 fasi: preprocessing, clustering del normale traffico, normalizzazione, fase di addestramento della ANN, fase di decisione della ANN. La prima fase usa una Self-Organizing Map (SOM), un metodo di addestramento non supervisionato che consente di imparare dei normali schemi di traffico. In questo modo, nel primo stage, gli attributi in input vengono quantizzati in bins. Il secondo stage è un percettrone multistrato MLP i cui pesi e iperparametri vengono stabiliti dalla SOM. Una volta addestrata la MLP si continua con la fase di predizione. L'approccio si è dimostrato funzionare in maniera ottima tanto da ottenere una efficacia del 100% per comportamento normale; viceversa, per alcuni attacchi, il valore di FAR raggiunge anche il 76%.

- **Support Vector Machines (SVM)**: è una normale tecnica di classificazione lineare in cui si cerca di trovare un iperpiano di separazione tra le classi, lasciando però un margine di errore più elevato possibile. Più nello specifico, si comincia cercando i vettori di supporto, ovvero gli elementi più estremi appartenenti alle varie classi e, per ogni classe, si crea un iperpiano passante per questi vettori. Gli iperpiani di classi differenti devono essere paralleli tra loro e lo spazio compreso tra questi è detto margine: la particolarità è che questo non deve contenere alcun dato in modo da garantire la miglior classificazione possibile. Conclusa l'operazione, si posiziona l'iperpiano di separazione tra le due classi, che sarà parallelo agli iperpiani che costituiscono il margine e passerà per il suo centro.

  Le SVM presentano un problema: lavorano con iperpiani, quindi non riescono a ottenere dei buoni risultati per classi linearmente non separabili. Proprio per questo si ricorre spesso al **Kernel Trick**, un espediente

matematico che fa uso di una funzione di kernel per mappare i dati in uno spazio di dimensione maggiore rispetto a quello iniziale, in modo da renderli linearmente separabili. Per i principi matematici infatti, ogni funzione, anche altamente non lineare, diventa lineare quando viene proiettata in uno specifico spazio di dimensioni elevate. In base al tipo di kernel possono quindi essere realizzati tanti tipi di separatori, come ad esempio lineari, polinomiali, gaussiani o tangente iperbolica.

- **Association Rules** e **Fuzzy Association Rules**: l'obbiettivo è quello di scoprire delle regole di associazione traendole dai dati. Le regole di associazione stabiliscono in quale misura degli elementi sono collegati tra loro in modo da poter affermare che quando ce n'è uno ce ne sarà anche un altro. Si usano quindi misure come il supporto e la confidenza.

  Il problema principale di questo approccio è che funziona solamente per dati booleani, mentre in molte applicazioni reali i dati sono principalmente di tipo categorico o quantitativo. Proprio per questo è stata sviluppata una modifica detta **Fuzzy Association Rules** in cui una regola di associazione è descritta come

$$\boldsymbol{IF\ (X\ is\ A) \rightarrow (Y\ is\ B)} \tag{1.1}$$

  Dove X e Y sono variabili e A e B sono dei fuzzy set che caratterizzano rispettivamente X e Y.

  Una implementazione dell'Association Rule è il **Frequent Pattern Mining**: l'idea è quella di lavorare su una serie di items che appartengono a uno stesso gruppo e, in base alle relazioni tra questi, creare degli itemset frequenti, ovvero insiemi di oggetti che spesso si trovano negli stessi gruppi in input. Per definire gli itemset frequenti ci si basa sul concetto di supporto minimo, cioè un valore calcolato come rapporto tra il numero di transazioni in cui questo è presente e il totale delle stesse: quando il supporto di un itemset supera questo valore minimo può essere considerato frequente. Lo scopo del Frequent Pattern Mining è quello di trovare tutti gli itemset frequenti massimi per poter stabilire tutte regole di associazione tra l'universo di items possibili.

  Esempi di applicazioni in ambito sicurezza sono [5] e [1]. In [5] il Frequent Pattern Mining è stato utilizzato per trovare le relazioni tra i parametri TCP/IP e i tipi di attacchi. L'applicazione delle regole di associazione prevedeva operazioni del tipo

$$IF(service\ AND\ src_{port} AND\ dst_{port} AND\ num_{conn})\ THEN\ attack\_type \quad (1.2)$$

In questo modo, in base al fatto che vi sono particolari servizi attivi in determinate porte di sorgente e destinazione, si riesce a risalire al tipo di attacco. Le performance migliori si sono ottenute con regole di 6 elementi in attacchi come DoS, Probe and Scan, U2R e R2L. Proprio per l'elevata efficienza si è compreso che l'association Rule Mining è un ottimo metodo per poter trovare le firme degli attacchi.

In [1], l'algoritmo consente di utilizzare l'association rule generalizzato per trovare anomalie e identificare pattern ricorrenti. Vengono utilizzate tre classi di regole: *Traffic Flow rules* che coinvolgono gli indirizzi di sorgente e destinazione; *Provided Security rules* che utilizzano la porta e indirizzo di destinazione; *Service Usage rules* che sfruttano la porta di destinazione e l'indirizzo sorgente.

Per quanto riguarda **l'addestramento non supervisionato**, il principale obbiettivo è quello di trovare pattern e strutture che differenziano i dati in quanto non sono etichettati. Di conseguenza non si conosce il valore che il sistema debba calcolare, ma deve impararlo autonomamente. I dati in input sono solitamente trattati come delle variabili aleatorie sulle quali viene creato, attraverso l'addestramento, un modello di densità. In altre parole, i dati, durante il processo di apprendimento, vengono raggruppati automaticamente in varie classi basandosi su misure di similarità o distanza, come mostrato, in riferimento ai sistemi di rilevamento delle intrusioni, in figura 1.4 a sinistra.

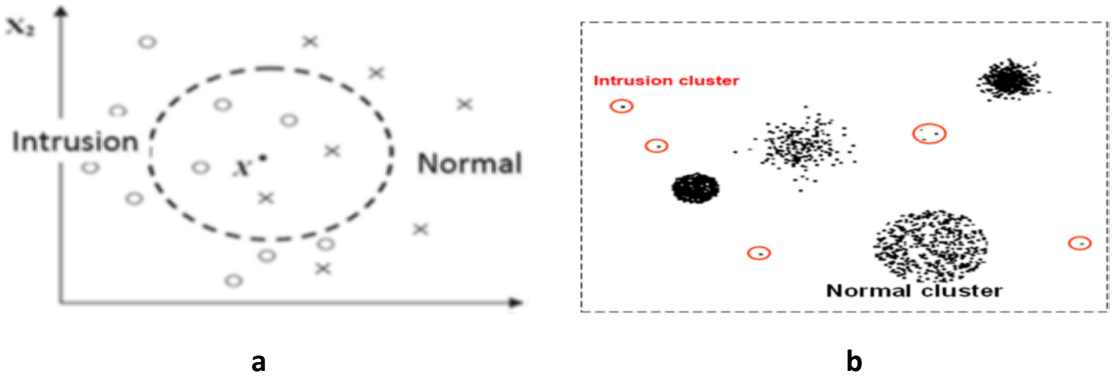

Figura 1.4: Clustering

Fonte: [33]

Una volta raggruppati, come mostrato in figura 1.4a, tutti i dati appartenenti ai clusters piccoli sono classificati come intrusioni in quanto hanno una minore probabilità di verificarsi rispetto al normale comportamento. Inoltre, dato che comportamenti normali e malevoli sono dissimili tra loro, i rispettivi clusters saranno sempre separati.

I principali metodi di machine learning applicati agli IDS che fanno uso di addestramento non supervisionato sono:

- **K-Means**: consente di raggruppare *n* oggetti in *k* gruppi. Per farlo si crea uno spazio geometrico nel quale valutare i dati e ogni dato viene inserito nel cluster col centroide più vicino. Per le misure di distanza si usa spesso la distanza euclidea, che crea clusters sferici, o la distanza di Mahalanobis che consente di classificare clusters con densità differente. Il valore di *k* è un iperparametro che viene fornito in input dall'utente: per scegliere il migliore si procede per tentativi o si utilizza l'elbow method.

  Il K-Means è oggi considerato il miglior metodo di classificazione non supervisionato. Il vantaggio principale è che i malware con signature simili vengono sempre raggruppati nello stesso cluster anomalo e di conseguenza bloccati.
- **Clustering Gerarchico**: tecnica di clusterizzazione che tenta di creare clusters in maniera gerarchica. Esistono *Clustering Agglomerativo* o *Bottom Up* e *Clustering Divisivo* o *Top Down*. Nel *Clustering Agglometarivo* si parte da *n* cluster, uno per ogni dato in input, e, a ogni iterazione, si uniscono i due cluster più simili tra loro allo scopo di crearne uno nuovo che li sostituisce. Si procede finché non rimangono *k* clusters.

Viceversa, nel *Clustering Divisivo* si parte da un singolo cluster con *n* elementi e si separano da questo dei sotto cluster differenti rispetto alla media. Si procede finché non si ottengono *k* clusters.

1.3.4 Considerazioni computazionali

Nella figura 1.5 sono mostrati i costi di addestramento dei metodi precedentemente descritti. Quelli con costo pari a $O(n)$ o $O\big(n\ log(n)\big)$si utilizzano per lavorare in tempo reale; $O(n^2)$sono costi già elevati, ma comunque accettabili; da $O(n^3)$in su sono invece costi molto elevati da preferire solamente per approcci

Algorithm	Typical Time Complexity	Streaming Capable	Comments
ANN	O(emnk)	low	Jain et al. [107] e: number of epochs k: number of neurons
Association Rules	>> $O(n^3)$	low	Agrawal et al. [108]
Bayesian Network	>> O(mn)	high	Jensen [41]
Clustering, k-means	O(kmni)	high	Jain and Dubes [46] i: number of iterations until threshold is reached k: number of clusters
Clustering, hierarchical	$O(n^3)$	low	Jain and Dubes [46]
Clustering, DBSCAN	O(n log n)	high	Ester et al. [109]
Decision Trees	$O(mn^2)$	medium	Quinlan [54]
GA	O(gkmn)	medium	Oliveto et al. [110] g: number of generations k: population size
Naïve Bayes	O(mn)	high	Witten and Frank [89]
Nearest Neighbor k-NN	O(n log k)	high	Witten and Frank [89] k: number of neighbors
HMM	$O(nc^2)$	medium	Forney [111] c: number of states (categories)
Random Forest	O(Mmn log n)	medium	Witten and Frank [89] M: number of trees
Sequence Mining	>> $O(n^3)$	low	Agrawal and Srikant [92]
SVMs	$O(n^2)$	medium	Burges [112]

Figura 1.5: Costi computazionali

Fonte: [32]

offline. Negli approcci online si deve considerare anche velocità di input/output dei dati o buffering.

In generale, affinché una rete sia adatta per approcci online deve tener conto di tre fattori:

- **Complessità computazionale**: come detto prima, un metodo di machine learning dovrebbe avere una complessità di $O(n)$ o $O\big(n\ log(n)\big)$ affinché la sua applicazione in uno streaming di dati sia ottimale. Nonostante questo, anche le ANNs vengono spesso usate per approcci online.
- **Capacità incrementale**: capacità di riaddestrare la rete in caso di nuovi dati in input. Tra i principali metodi si hanno gli algoritmi di clustering e metodi statistici, come Bayes e HMM.
- **Capacità di generalizzazione**: questa consente al sistema di non deviare troppo dal modello iniziale ogni volta che vede un nuovo dato. In altre parole, si adatta gradualmente ai dati in input.

1.3.5 Migliori approcci per l'implementazione di IDS

L'applicazione dei normali modelli di Machine Learning nella cyber security è peculiare rispetto agli altri domini. Come già accennato, quello della cyber security

è un mondo molto dinamico dove giornalmente devono essere create nuove strategie. Proprio per questo, le reti dovrebbero essere riaddestrate frequentemente e, preferibilmente, in maniera incrementale, a differenza degli approcci classici in cui verrebbe fatto, solo se necessario, a distanza di lunghi periodi di tempo. Un'altra caratteristica legata alla cyber security è la difficoltà di ottenere dati per l'addestramento. Sebbene i dati siano facili da raccogliere, la loro mole è talmente elevata da renderne difficile l'organizzazione.

Nella scelta dei possibili metodi per l'implementazione degli IDS possono essere fatte delle osservazioni in base ciò che è più utilizzato in letteratura.

Sicuramente un ruolo importante è ricoperto dai *dati* utilizzati per l'addestramento. Esistono pochi dataset pubblici e molti di questi sono abbastanza arretrati. Tra i principali vi sono il DARPA 1998, DARPA 1999, DARPA 2000, KDD 1999 e, tra i più recenti il CICIDS 2017. Nella maggior parte dei casi, vengono raramente utilizzati gli interi datasets, ma sono filtrati in modo da lavorare solamente sui dati utili ai propri fini.

Altri parametri di confronto tra i diversi metodi sono:

- **Accuratezza**: l'accuratezza è un parametro fondamentale. Si punta ad avere sistemi con un valore di accuratezza vicini al 100% e con un FAR pari a 0%.
- **Tempo di addestramento**: la rete dovrebbe essere veloce da addestrare. Infatti, in ambito cyber security, ogni giorno vengono trovate nuove vulnerabilità che portano gli attacchi ad essere molto dinamici. La rete deve essere capace di adattarsi, quindi deve essere riaddestrata con la stessa velocità di generazione degli attacchi.
- **Tempo di classificazione**: tempo di reazione del modello, già addestrato, ogni volta che si trova a dover classificare qualcosa che non ha mai visto. Se questo è un attacco, il tempo impiegato per la segnalazione deve essere molto breve.
- **Chiarezza del risultato finale**: è necessario fornire un output più chiaro possibile in modo da aiutare l'amministratore del sistema sotto attacco a comprendere bene la causa dell'anomalia e, di conseguenza, reagire con prontezza. Ricordare infatti che un IDS ha il solo scopo di allertare possibili comportamenti anomali e non quello di bloccarli.

1.4 Metriche di valutazione

Le metriche di valutazione [40] [41] sono le formule utilizzate per valutare le performance di un IDS. Ogni IDS viene valutato in base ai parametri mostrati nella seguente matrice di confusione.

Predizione → Reference ↓	C_i	Altre Classi
C_i	TP_i	FN_i
Altre Classi	FP_i	TN_i

Tabella 1.1: Matrice di confusione

dove **True Positive TP_i** indica il numero di elementi appartenenti alla classe C_i che sono stati predetti correttamente come appartenenti a quella classe; **False Positive FP_i** indica il numero di elementi non appartenenti alla classe C_i che sono stati predetti erroneamente come appartenenti a quella classe; **False Negative FN_i** indica il numero di elementi appartenenti alla classe C_i che sono stati predetti erroneamente come appartenenti ad altre classi; **True Negative TN_i** indica il numero di elementi non appartenenti alla classe C_i che sono stati predetti correttamente come appartenenti a classi differenti. Nel caso ideale la matrice di confusione è diagonale, ovvero tutti i dati in input sono classificati correttamente.

Tutti questi valori vengono utilizzati nel calcolo degli indici di valutazione.
Tra i principali:

- **Accuracy**: indica quanto un classificatore è accurato. Si calcola come il rapporto tra le istanze classificate correttamente e tutte le istanze in input.

$$\boldsymbol{Acc = \frac{TP+TN}{TP+TN+FP+FN}} \tag{1.3}$$

  Se le classi sono poco bilanciate, ovvero vi sono molti elementi appartenenti solo ad alcune classi, l'indice non è molto affidabile.
- **Precision P_i**: è la frazione di elementi predetti correttamente come appartenenti alla classe C_i diviso il numero di elementi totali predetti come appartenenti, correttamente o no, alla classe i-esima. Il miglior valore

possibile è 1 nel caso in cui tutti gli elementi della classe i-esima sono stati classificati correttamente.

$$\boldsymbol{P_i = \frac{TP_i}{TP_i+FP_i}} \quad (1.4)$$

- **Recall R_i** o **True Positive Rate TPR_i**: è il rapporto tra gli elementi predetti correttamente come appartenenti alla classe C_i e il numero di elementi totali che hanno come etichetta (appartengono veramente) la suddetta classe. Il miglior valore possibile è 1 nel caso in cui tutti gli elementi della classe i-esima sono stati classificati correttamente.

$$\boldsymbol{R_i = \frac{TP_i}{TP_i+FN_i}} \quad (1.5)$$

- **F1 score**: è il prodotto di Precision e Recall diviso la loro somma. Il miglior valore possibile è 1 e corrisponde al caso in cui P_i e R_i sono entrambi pari a 1.

$$\boldsymbol{F_1 = 2\ \frac{P_i * R_i}{P_i+R_i}} \quad (1.6)$$

- **False Positive Rate FPR_i o False Alarm Rate FAR**: è la frazione di elementi predetti erroneamente come appartenenti alla classe C_i rispetto agli elementi appartenenti ad altre classi.

$$\boldsymbol{FPR_i = \frac{FP_i}{FP_i+TN_i}} \quad (1.7)$$

Gli indici TPR e FPR vengono utilizzati per la creazione delle curve **Receiver Operating Characteristic** o **ROC Curve**. Indicando l'FPR sull'ascissa e il TPR sull'ordinata la curva mostra graficamente la performance di un IDS. Un test con discriminazione perfetta ha una curva ROC che passa per l'angolo in alto a sinistra.

1.5 Tecniche di evasione

Le tecniche di evasione [41] comprendono tutti quei metodi che un utente malevolo può sfruttare per evitare di essere rilevato da un IDS.

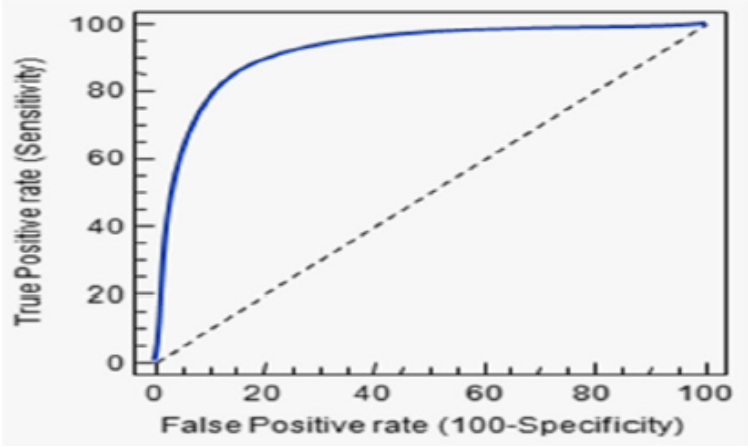

Figura 1.6: Esempio di una curva ROC

Fonte: [33]

Le tecniche più utilizzate sono:

- **Frammentazione**. Si basa sul funzionamento tipico delle normali reti: quando un pacchetto è troppo grande per essere trasferito attraverso un determinato collegamento, viene frammentato; questi frammenti viaggiano singolarmente per il resto della rete finché, arrivati al destinatario, vengono ricomposti nell'ordine corretto per ricreare il pacchetto originario.

  Gli attacchi che sfruttano questa tecnica consistono nella forzatura della frammentazione di un pacchetto malevolo. L'IDS, soprattutto se NIDS, non ha la capacità di ricostruire il pacchetto originario, ma analizza ogni frammento singolarmente. Dato che questo contiene solamente parte del codice malevolo è molto probabile che non venga segnalato come anomalo; quindi, che non venga generato alcun allarme.

  Gli IDS che invece sono in grado di analizzare l'intero pacchetto frammentato, come gli HIDS, sono vulnerabili ad un attacco basato sul timeout: ogni IDS conserva i frammenti per un certo intervallo di tempo, oltre il quale li scarta. Gli attacchi consistono nell'invio di frammenti in ritardo in modo da superare questo intervallo.
- **Flooding**. L'attaccante invia un numero elevato di pacchetti che causano un sovraccarico nell'IDS e lo rendono inattivo per un certo periodo di tempo. Quando il detector è inattivo qualsiasi traffico in entrata non è controllato; quindi, è possibile inviare il codice malevolo. Tra i protocolli più usati per il flooding vi sono UDP e ICMP.

- **Offuscamento**. Si tenta di nascondere il codice malevolo o l'attacco rendendolo difficile da riconoscere. La tecnica viene utilizzata spesso nei comuni malware: il programma che li contiene viene reso funzionalmente utile, in modo da rendere difficile scovarlo attraverso una semplice analisi statica o reverse-engineering. Per questo motivo, l'offuscamento è in grado di sopraffare molti IDS. La maggior parte delle tecniche di offuscamento tentano di sfruttare delle limitazioni dei database delle signatur: un esempio è l'uso di signature codificate in esadecimale o in particolari formati non sono supportati dagli IDS.
- **Cifratura**. I pacchetti cifrati non possono essere letti dagli IDS e soprattutto non può essere riconosciuta la firma per la quale ricercare una corrispondenza nel database dei signature. Più nello specifico gli HIDS sono in grado di esaminare una comunicazione cifrata, in quanto si trovano dentro l'host. Viceversa, gli NIDS risultano vulnerabili in quanto non possono leggere il contenuto dei pacchetti che intercettano. Dal momento che oggi la maggior parte dei pacchetti nella rete sono cifrati, gli IDS vengono progettati per lavorare su statistiche del traffico piuttosto che sulla lettura diretta del loro contenuto.

Capitolo 2. Adversarial Machine Learning negli IDS

Questo capitolo ha lo scopo di descrivere l'Adversarial Machine Learning, la sua applicazione contro gli IDS e le diverse tecniche di attacco e di difesa conosciute in letteratura scientifica.

L'Adversarial machine learning è una tecnica volta a compromettere il corretto funzionamento di un sistema informatico che faccia uso di algoritmi di apprendimento automatico, tramite la costruzione di input speciali in grado di ingannarli: più nello specifico, lo scopo di tali tecniche è quello di causarne una classificazione errata. Nella maggior parte dei casi, l'obbiettivo finale è quello di costruire dei campioni atti a riaddestrare il sistema e, di conseguenza, migliorarne l'efficienza. L'avversario è spesso una intelligenza artificiale che viene addestrata di pari passo al modello vittima: mentre questo cerca di migliorare l'accuratezza della classificazione di tutti i dati in input, sia normali che avversari, l'avversario si addestra per generare dei campioni sempre più difficili da riconoscere.

L'applicazione dell'adversarial machine learning per l'addestramento di sistemi di sicurezza basati su intelligenza artificiale, come gli IDS [7][8], sta sempre più prendendo campo in quanto rappresenta lo scenario reale in cui questi sistemi dovrebbero operare: un attaccante cerca di sfruttare delle vulnerabilità per ottenere dei vantaggi. Gli attacchi ai sistemi di sicurezza sono però molto più complicati rispetto a quelli classici rivolti alle reti neurali convoluzionali (CNN) a causa della difficoltà di mantenere intatta la funzionalità del campione modificato. Nel caso delle CNN basta modificare dei pixel e renderli di un colore simile; viceversa, nella cybersecurity la modifica di un bit, in un eseguibile o in un protocollo, potrebbe portare anche all'arresto del sistema. Inoltre, anche se il dato viene modificato correttamente, non è possibile verificare in alcun modo che la sua funzionalità sia stata mantenuta (si lavora con sequenze di bit o statistiche sul flusso di pacchetti), a differenza delle CNN in cui basta una semplice ispezione visuale.

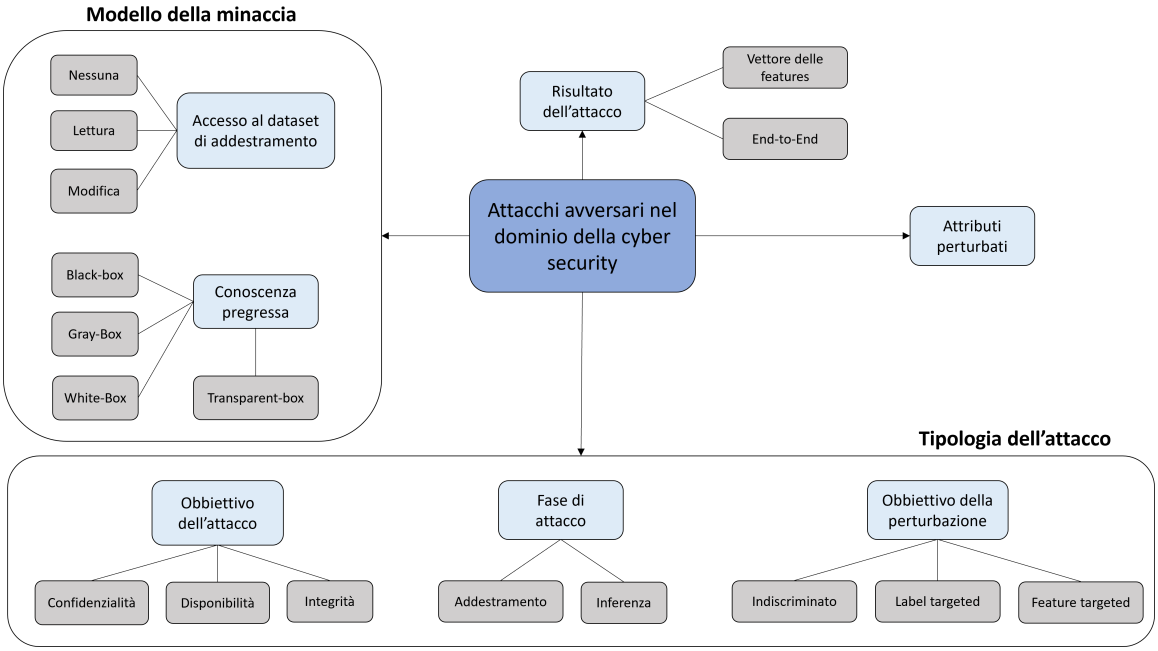

Figura 2.1: Tassonomia degli attacchi

Fonte: [8]

2.1 Tassonomia degli attacchi

Gli attacchi sono categorizzati sulla base di quattro caratteristiche, come mostrato dalla figura 2.1 [47]:

1. **Modello della minaccia**
2. **Tipologia di Attacco**
3. **Attributi** da perturbare nell'attacco.
4. **Risultato** dell'attacco.

Il modello della minaccia può essere ulteriormente scomposto in *Accesso al dataset di addestramento* e *Conoscenza pregressa*, e la Tipologia di Attacco in *Obbiettivo dell'attacco*, *Fase di attacco* e *Obbiettivo della perturbazione*.
Le ultime due caratteristiche sono particolarmente importanti nel campo della cyber security in quanto, a causa della difficoltà di mantenere invariata la funzionalità del campione in input, vengono imposti numerosi vincoli che specifichino quali attributi possono essere modificati e come è possibile farlo.

2.1.1 Modello della minaccia

Il modello della minaccia rappresenta tutto ciò che l'attaccante conosce del modello vittima prima di iniziare l'attacco. Più informazioni possiedono, maggiore sarà la sua probabilità di successo. Come già accennato questa caratteristica può essere scomposta in *Accesso al dataset di addestramento* e *Conoscenza pregressa*.

- **Accesso al dataset di addestramento**: indica la capacità dell'attaccante di accedere al dataset utilizzato per addestrare il modello, in modo da **leggere** i dati o **modificare** quelli esistenti al fine di introdurre delle vulnerabilità nel sistema manipolandone l'addestramento. Può anche capitare che l'attaccante non possieda **nessuna capacità** di accesso al training set.

  È facile capire che negli attacchi più comuni un attaccante ha solo la capacità di lettura, mentre quella di modifica è poco probabili. Questi ultimi però, dato che consentono all'attaccante di manipolare a proprio piacimento l'addestramento del sistema, sono i più pericolosi.
- **Conoscenza pregressa**: indica la conoscenza a priori che l'attaccante ha del sistema. Si distinguono: gli **attacchi Black-Box** che non richiedono alcuna conoscenza pregressa del sistema, quindi l'attaccante vede il modello come un oracolo al quale può fare delle domande e ottenere delle risposte; gli **attacchi Gray-Box** che richiedono una conoscenza limitata del sistema, come gli attributi su cui lavora, gli output di alcuni layer nascosti o il tasso di errore sulla predizione; gli **attacchi White-Box** che richiedono una conoscenza elevata del sistema, come l'architettura del modello e i suoi iperparametri; gli **attacchi Transparent-Box** in cui l'avversario ha una conoscenza totale del sistema, quindi oltre a conoscere l'architettura e gli iperparametri del modello conosce anche il sistema di difesa utilizzato dalla rete.

  È semplice comprendere che gli attacchi di tipo White-Box e Transaprent-Box sono poco comuni nella realtà in quanto è molto difficile che un attaccante abbia una conoscenza tanto elevata del sistema. Spesso chi pratica questi attacchi è lo stesso che ha creato il modello vittima. I più comuni sono invece quelli di tipo Black-Box o Gray-Box.

2.1.2 Tipologia di attacco

Ogni attacco possiede delle caratteristiche specifiche, come l'obbiettivo o gli attributi perturbati, in base alle quali è possibile effettuarne una classificazione. Come già accennato precedentemente si distinguono *Obbiettivo dell'attacco*, *Fase di attacco* e *Obbiettivo della perturbazione*.

- **Obbiettivo dell'attacco**: rappresenta l'obbiettivo dell'attacco che potrebbe essere quello di minacciare la **confidenzialità** di alcune informazioni private (come gli iperparametri del modello) effettuando richieste al modello vittima, l'**integrità** del sistema forzandolo a predire determinati input in maniera errata o la sua **disponibilità**, ovvero renderlo indisponibile bloccandone il funzionamento. Nel caso degli IDS una indisponibilità consentirebbe a tutto il traffico di rete di accedere al sistema senza alcun controllo.
- **Fase di attacco**: consiste nella fase in cui l'attacco viene svolto e dipende dall'influenza che l'attaccante riesce ad esercitare sul sistema. Si possono distinguere gli attacchi in fase di **addestramento** e quelli in fase di **inferenza**. Quelli in fase di addestramento vengono effettuati durante la fase di addestramento del modello e puntano a introdurre vulnerabilità nel sistema manipolando i dati di training; si parla di attacchi **causativi**. Un esempio è l'attacco di **poisoning** in cui l'attaccante inserisce nel training set dei dati a proprio piacimento.

  Gli attacchi in fase di inferenza sono invece gli attacchi effettuati durante la fase di predizione, ovvero quando il sistema è stato già addestrato, e puntano a trovare e sfruttare vulnerabilità nella fase di classificazione; si parla di attacco **esplorativo** o di **evasione** in quanto si evade dal controllo del sistema.

  Gli attacchi in fase di inferenza sono solitamente più complessi di quelli effettuati durante l'addestramento in quanto l'attaccante ha un minor margine di azione.
- **Obbiettivo della perturbazione**: rappresenta gli attributi che sono obbiettivo dell'attacco. Si distinguono gli **attacchi indiscriminati** in cui si punta a minimizzare la probabilità che il sistema effettui una classificazione corretta su tutti gli input, **attacchi Label-Targeted** che puntano a minimizzare la probabilità che il sistema classifichi correttamente una specifica classe e **attacchi Feature-Targeted** in cui il

sistema ha difficoltà a classificare specifici dati in input o i valori dei singoli attributi.

Nel caso della classificazione binaria, come spesso accade nel dominio della cyber security, le prime due tipologie di attacco sono uguali in quanto si hanno solamente due classi.

2.1.3 Attributi perturbati

Normalmente gli attaccanti hanno necessità di modificare più di una feature sui dati per avere successo. Proprio questo consente di effettuare una classificazione degli attacchi in base agli attributi che vengono aggiunti, modificati o eliminati. La modifica delle feature ha un peso differente in base al dominio di applicazione: l'alterazione di una stringa in una e-mail è molto più semplice e meno dannosa rispetto a quella di una sequenza di bit in un eseguibile. Inoltre, una modifica indiscriminata di tutti gli attributi porterebbe alla perdita della funzionalità del dato, come ad esempio, un malware perturbato potrebbe perdere la sua caratteristica malevola diventando una sequenza di bit senza alcun significato. Sebbene non esista una soluzione che garantisca che il dato perturbato rimanga ancora funzionale, una pratica comune è quella di introdurre dei vincoli nella modifica.

Un'altra soluzione utilizzata nel modello proposto, oggetto della tesi, riguarda l'applicazione di vincoli, tramite l'utilizzo di modelli di machine learning, al fine di mantenere invariate le correlazioni tra gli attributi. L'approccio verrà approfondito nel Capitolo 4.

2.1.4 Risultato dell'attacco

La classificazione può riguardare l'output dell'attacco da mandare in input al sistema vittima. Esistono due tipi di risultati possibili:

- **Vettore dei feature**: l'attacco consiste nel perturbare un vettore dei feature per ottenerne uno nuovo. In questo tipo di attacco la perturbazione genera spesso campioni che l'attaccante non può riutilizzare. L'unica soluzione, allora, sarebbe quella di applicare dei vincoli, ma anche questi sono complicati da valutare.

- **End-to-End**: l'attacco genera un output funzionale. Il metodo è utilizzato realmente in quanto ha una elevata funzionalità per l'attaccante.

  Un esempio nell'ambito cyber security è quello di aggiungere chiamate di sistema agli eseguibili: il risultato è sempre un eseguibile che se costruito in maniera corretta potrebbe eludere un sistema di controllo.

2.2 Metodi di attacco

La stragrande maggioranza di attacchi agli IDS viene fatta attraverso metodi basati su gradiente i quali puntano ad introdurre perturbazioni, ottimizzate per certe metriche di distanza, tra i campioni originali e perturbati. In altre parole, osservando lo spazio degli attributi, si cerca di muovere il dato originale lungo una direzione ottenuta tramite il calcolo di un gradiente cercando di introdurre la minima perturbazione possibile o, detto in altri termini, spostare il campione il minimo possibile.

Le tre principali metriche di distanza sono [30]:

- $\mathbf{L_{inf}}$: ha lo scopo di minimizzare il massimo ammontare di perturbazione introdotta in ogni attributo.
- $\mathbf{L_0}$: ha lo scopo di minimizzare il numero di attributi perturbati.
- $\mathbf{L_2}$: ha lo scopo di minimizzare la distanza euclidea tra il campione originale e quello perturbato.

Il concetto di metrica di distanza è quello che crea i maggiori problemi nell'adversarial machine learning. Nella realtà, infatti, la distribuzione delle perturbazioni dei campioni realmente malevoli potrebbe essere molto differente da quella ottenuta utilizzando queste metriche nella creazione del campione avversario.

Negli attacchi agli IDS una proprietà cardine riguarda la **trasferibilità tra modelli** [13]. Questa si basa sul **Transfer Learning**, una tecnica in cui un modello pre-addestrato su certi dati viene riutilizzato per svolgere compiti simili a quelli per i quali era stato preparato. Se due modelli sono stati creati per svolgere compiti simili allora esiste una conoscenza generale comune a entrambi e, di conseguenza, questi sono, almeno alla base, simili tra loro. Sulla base del Transfer Learning è stato creato un nuovo approccio detto **Multitask Learning**, oggi molto utilizzato per le reti neurali profonde. Il metodo utilizza un modello generale come base comune di altri

modelli, i quali effettuano operazioni specifiche aggiungendo ulteriori layer nascosti che lo specializzano verso un task. È stato dimostrato che tramite un addestramento iniziale generico che viene specializzato successivamente, il modello finale possiederà una capacità di generalizzazione elevata.

Tornando adesso al concetto di trasferibilità tra modelli, se un modello viene addestrato sui dati etichettati da un altro, ne sta carpendo le conoscenze di base, di conseguenza, un campione dovrebbe essere classificato allo stesso modo da entrambi. Questo consente di semplificare gli attacchi di tipo black-box. L'attaccante adesso è in grado di generare un sostituto del modello vittima e addestrarlo su dati etichettati da quest'ultimo, in modo da copiarne le linee di confine per la classificazione. In questo modo può effettuare un attacco di tipo transparent-box sul sostituto e, trovato un campione funzionante, può riutilizzarlo su quello vittima. È però necessario che questo venga imitato alla perfezione affinché la sua efficienza sia elevata, ovvero che tutti i campioni avversari generati sul modello sostituto funzionino anche sul modello finale.

Le tecniche di attacco possono essere divise nei seguenti macrogruppi:

- **Poisoning**: è un attacco all'integrità del sistema. Mira a inserire dati etichettati erroneamente nel training set in modo da poter sfruttare l'errore di addestramento che ne scaturirà. Il problema è che la loro aggiunzione porterebbe il modello vittima a classificare in maniera errata molti altri dati in fase di test destando quindi sospetti.
- **Esaurimento di risorse**: esistono due varianti. La prima consiste nell'ingannare il sistema facendo in modo che classifichi il traffico normale come anomalo e generino degli allarmi per tutte le comunicazioni. La seconda variante consiste invece in attacchi di flooding in cui vengono inviati tanti pacchetti al sistema vittima in modo da fargli esaurire le risorse e impedire il normale traffico di rete. Ad oggi gli attacchi di flooding sono molto utilizzati contro gli IDS.
- **Attacchi alla confidenzialità**: consistono nell'utilizzo di reverse engineering per ottenere informazioni sul modello vittima. Spesso sono finalizzati allo svolgimento di attacchi di tipo white-box.

  Oggi vengono raramente utilizzati in ambito cyber security in quanto risulta complicato carpire informazioni senza destare sospetti.
- **Exploit di vulnerabilità**: si basano su vulnerabilità trovate nei maggiori framework di creazione e addestramento di sistemi intelligenti, come

Tensorflow o scikit-learn. In base al tipo di vulnerabilità un attaccante è in grado di compromettere il corretto funzionamento del sistema di sicurezza. Oggi vengono raramente utilizzati in ambito cyber security.

Esaminando adesso le tecniche di attacco, tra le principali si hanno:

- **Box constrained limited-memory Broyden Fletcher-Goldfarb-Shanno (L-BFGS)** [17]: basandosi sulla metrica L_2 (distanza euclidea tra campione perturbato e originale), punta a generare un campione simile al primo aggiungendo una perturbazione. Si cerca di minimizzare la quantità di perturbazione aggiunta al campione originale.

  Sebbene il metodo riesca a produrre campioni ottimi è computazionalmente troppo oneroso nell'esecuzione dell'ottimizzazione.
- **Fast Gradient Sign Method (FGSM)** [13]: la tecnica consiste nel valutare il segno del gradiente della funzione di Loss, usata dal classificatore obbiettivo, rispetto alla label reale per generare un nuovo campione che ne massimizzi il valore.
- **Jacobian Based Saliency Map Attack (JSMA)** [15]: usa la feature selection con l'obbiettivo di minimizzare il numero di attributi perturbati dall'algoritmo (metrica L_0). Il metodo fa uso di una **saliency map** per ogni campione di input, la quale contiene un valore di importanza associato a ogni suo attributo: questo valore specifica quanto la modifica di ogni feature influenzerà il processo di classificazione. Le feature vengono poi selezionate in ordine decrescente per valore di importanza.

  Il metodo è computazionalmente più oneroso rispetto a FGSM, ma riduce drasticamente il numero di features perturbate creando un campione avversario molto simile a quello originale.
- **Deepfool** [14]: minimizzare la distanza euclidea tra il campione perturbato e quello originale (metrica L_2). L'attacco consiste nel calcolare una linea retta di confine che separa i campioni di classi differenti, a cui si aggiunge una perturbazione perpendicolare. Nelle reti neurali queste linee di confine sono spesso non lineari; quindi, l'attacco viene ripetuto iterativamente finché non si trova un candidato adatto (campione avversario).

  Il metodo è computazionalmente oneroso (più di JSMA), ma introduce una perturbazione inferiore rispetto a quelli precedenti.

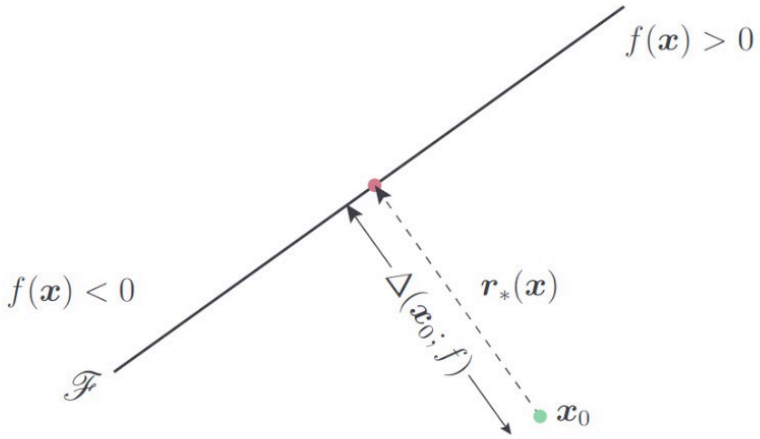

Figura 2.2: Deepfool

Fonte: [7]

- **Carlini & Wagner attack (C&W)** [10]: l'attacco diventa un problema di ottimizzazione. È molto simile al L-BFGS, ma, piuttosto che utilizzare la cross entropy come funzione di Loss, si utilizza la Hinge Loss e viene aggiunta una nuova variabile che rende la risoluzione più efficiente.
- **Generative Adversarial Network (GAN)** [12]: due sistemi basati su machine learning vengono addestrati per competere tra loro: uno agisce da generatore di campioni, l'altro da classificatore. Il primo cerca di produrre campioni che portano il classificatore a sbagliare, mentre il secondo si addestra a riconoscerli in modo da classificarli correttamente. Idealmente si raggiunge un equilibrio dove nessuno dei due riesce più a migliorare il proprio comportamento.

  Nel caso si utilizzino delle reti neurali, un grande problema è rappresentato dallo **svanimento del gradiente** [16] in cui il gradiente della funzione di Loss diventa talmente piccolo da non consentire più alcun addestramento. Il problema è tipico della GAN in quanto il campione generato dalla rete avversaria ha una perturbazione bassa, pertanto l'errore finale diventa piccolo.
- **Wasserstein GANs (WGAN)** [19]**:** è una modifica della GAN in cui si utilizza, piuttosto che il gradiente della funzione di Loss, la distanza di Wasserstein, che quantifica la divergenza tra due distribuzioni. È dimostrato che la tecnica elimini molti problemi delle GAN e produca un addestramento più diretto.

  L'approccio, però, come avviene nella GAN, introduce nel campione avversario delle perturbazioni meno predicibili rispetto agli altri metodi basati su gradiente, che riescono a farlo rispettando una determinata metrica [18].

- **Zeroth-order optimization attack (ZOO)** [11]**:** consente di stimare il gradiente di un classificatore senza averne accesso, per questo è ideale per effettuare attacchi di tipo black-box. Il metodo utilizza la discesa stocastica di ordine nullo che consiste nell'aggiungere iterativamente delle perturbazioni a ogni attributo, facendone stimare il gradiente e l'hessiana al classificatore vittima al fine di ottimizzare i campioni malevoli. Questo rende la tecnica indipendente dall'addestramento di modelli sostituti o dalla conoscenza di dettagli relativi al classificatore stesso.

  Il metodo ottiene risultati simili al C&W nella generazione di campioni malevoli. È però necessario effettuare una grandissima quantità di query al classificatore e questo potrebbe destare sospetti.

2.3 Metodi di difesa

Le tecniche di difesa dagli attacchi si possono distinguere in due sottocategorie:

- **Metodi di rilevamento**: cercano di rilevare i dati potenzialmente avversari prima che arrivino al modello.
- **Metodi di robustezza**: cercano di migliorare l'accuratezza del classificatore in modo da renderlo resistente agli esempi avversari.

Per quanto riguarda le tecniche, si distinguono:

- Metriche che misurano la robustezza di un classificatore contro esempi avversari. Uno dei metodi più apprezzati consiste nel misurare la robustezza di un sistema di machine learning approssimando la soglia minima di perturbazione richiesta a un attaccante per avere successo. Bisognerebbe però tenere conto della difficoltà che si ha nell'inserire la perturbazione nei dati perché risulta complicato trovare metriche che consentono di farlo.
- Metodi resistenti ad attacchi sconosciuti e a Transparent-box. Ancora non sono sviluppati metodi che possiedano queste capacità in quanto si riscontrano due problemi. Il primo è la difficoltà di creare un sistema in grado di resistere ad attacchi Transparent-box in cui l'attaccante conosce tutto della rete. Il secondo è la difficoltà di creare sistemi capaci di resistere a tutti i possibili attacchi, anche sconosciuti, in quanto una pratica comune è quella di renderli resistenti solo ad attacchi specifici.
- Unione di tecniche specifiche e agnostiche al dominio di applicazione.

La difesa consiste nell'unire tecniche di addestramento normalmente usate in ambiti differenti, come ad esempio l'unione di tecniche di computer vision nell'abito degli IDS. Tra queste tecniche vi sono:

- **Adversarial training** [13]: addestrare la rete con dati generati da un avversario, oppure, piuttosto che generare dati validi, si potrebbe utilizzare una funzione in grado di introdurre una perturbazione nella funzione di Loss del classificatore, in modo da emulare l'effetto di un dato perturbato.
- **Gradient Masking** [28]: assumendo che un modello non sia differenziabile oppure il suo gradiente sia nullo, qualsiasi attacco basato sul gradiente diventa inefficace. L'approccio, allora, sarebbe quello di usare modelli non differenziabili come alberi decisionali o random forest. Ovviamente si tratta di una assunzione, in quanto anche questi modelli sono vulnerabili a tipologie di attacchi simili.
- **Feature squeezing** [29]: i dati vengono compressi e successivamente classificati. La compressione ha lo scopo di ridurre la perturbazione nel dato. Se la classificazione su un campione compresso è differente da quella del campione originale, allora si tratta di un campione avversario. Il metodo, però, dipende troppo dai dati di addestramento e dall'algoritmo di compressione, e questo è uno svantaggio in quanto ogni metodo performa in maniera differente in dipendenza dai dati.
- **Universal Perturbation Defence** [20]: il metodo consiste nel piazzare una rete neurale, detta **Perturbation Rectifying Network (PRN)**, di fronte al classificatore da proteggere, con lo scopo di intercettare i campioni avversari. La rete consente di ridurre in maniera sostanziale il rumore dei dati in input, come la perturbazione di un campione avversario, e di mandare in input, al sistema che protegge, solamente le caratteristiche principali, che sono le uniche necessarie alla classificazione.
- **MagNet** [21]. Il metodo è costituito da due componenti principali: un **detector** e un **reformer**. Il detector ha il compito di filtrare i dati in input al classificatore effettuando una prima semplice classificazione: se i dati deviano troppo da un comportamento considerato normale, verranno bloccati. Il reformer, invece, si occupa di ridurre il rumore di ogni campione che superi il controllo del detector, in modo da eliminarne la perturbazione residua che potrebbe portare a una classificazione errata.

2.3.1 MANDA

Viene adesso presentato un recente approccio alla difesa degli IDS.

MANDA [9], acronimo di Manifold-Decision Bondary-Adversarial, è uno schema di rilevamento di esempi avversari per IDS basati su machine learning. Il suo compito è quello di assistere un normale IDS nella difesa da esempi avversari (AE) che lo porterebbero a effettuare classificazioni errate. MANDA è quindi un metodo di robustezza (paragrafo 2.3), ausiliario a un IDS, che intercetta tutti i dati prima che arrivino a quest'ultimo e verifica se si tratta di campioni avversari.

L'algoritmo consiste inizialmente nel generare dei campioni avversari da utilizzare per addestrare dei modelli di machine learning, allo scopo di distinguerli dai campioni normali. Si tratta di Manifold e Decision Boundary (DB), modelli che verranno descritti successivamente.

Per quanto riguarda la **generazione dei campioni avversari**, si definisce lo **spazio del problema**, per un IDS, come tutte le possibili istanze di traffico nella rete sotto forma di pacchetti. Lo **spazio dei feature** è composto invece da tutti i possibili feature-vector che rappresentano i pacchetti. È necessario che ogni AE sia leggibile dall'IDS, quindi il generatore e l'IDS devono lavorare sullo stesso spazio del problema. In altre parole, AE deve essere un pacchetto di rete ben formato.

Per fare questo, un primo passo è quello di prendere un valore di input pulito, che chiamiamo x, e calcolarne il feature vector x' (x traslato nello spazio dei feature). Infine, x' viene proiettato nello spazio del problema ottenendo z'. Per farlo è possibile usare delle normali tecniche di mapping inverso. Alcuni feature, però, non sono differenziabili, quindi non è possibile invertirne la trasformazione. Proprio per questo, in MANDA, si è deciso di non introdurre alcuna perturbazione sui feature non differenziabili. Normalmente, per gli IDS, queste features sono rappresentate da quelle categoriche.

È necessario, inoltre, che z' mantenga la sua funzione originaria anche dopo queste trasformazioni. Risulta molto complicato generare un AE che possieda tale proprietà e, come già accennato precedentemente, l'unica soluzione è quella di imporre dei vincoli nella trasformazione. In MANDA sono stati applicati i seguenti vincoli:

- Differenziare le features funzionali, che sono quelle che hanno diretta importanza nella funzionalità dell'istanza, da quelle non funzionali, ovvero quelle di contorno. Le features funzionali non dovrebbero essere alterate, in

quanto, data la loro importanza, una modifica porterebbe alla perdita delle proprietà intrinseche dell'istanza.
- Fare in modo che anche dopo la perturbazione il valore della feature ricada nel dominio corretto.
- Mantenere un livello di perturbazione basso per i feature non funzionali.
- Aggiungere delle perturbazioni consistenti nei feature correlate tra loro.

Una volta generati i campioni avversari si passa ad addestrare i modelli che si occuperanno della loro classificazione. Per semplicità, il funzionamento dell'algoritmo è spiegato tramite un esempio. Osservando la figura 2.3, in rosso è indicato il manifold malevolo e in blu quello benevolo. Si nota che questi si trovano rispettivamente a sinistra e a destra della linea di classificazione dell'IDS, ma a causa della sua forte non linearità, questa si inserisce all'interno dei manifold. Si possono verificare due casi: il caso *A* in cui i due manifold sono completamente separati tra loro e il caso *B* in cui i due manifold sono sovrapposti.

La componente Manifold di MANDA ha risultati ottimi solamente sul caso A, mentre quella DB sul caso B. Le due componenti sono infatti completamente autonome e potrebbero essere usate separatamente; MANDA le usa contemporaneamente sui pacchetti in input per ottenere migliori performance.

Il concetto di **manifold** si basa sul Manifold Learning, una tecnica di machine learning in cui si assume che i dati in input possano essere raggruppati secondo strutture, chiamate Manifold, a minore dimensione rispetto ai dati stessi. Il Manifold Learning è il processo che impara autonomamente le proprietà geometriche e topologiche di un dato manifold, in pratica gli dà la forma. Il risultato è la creazione di un modello M dove $M(x)$ restituisce le probabilità che x si trovi in tutti i possibili Manifold conosciuti dal modello. In MANDA si usano due manifold: uno benevolo e l'altro malevolo.

Tornando all'esempio, si esamina il caso A. Il campione avversario x' si trova dentro il manifold malevolo ed è molto lontano da quello benevolo, ma, trovandosi dal lato opposto della linea di separazione dell'IDS, questo lo classifica

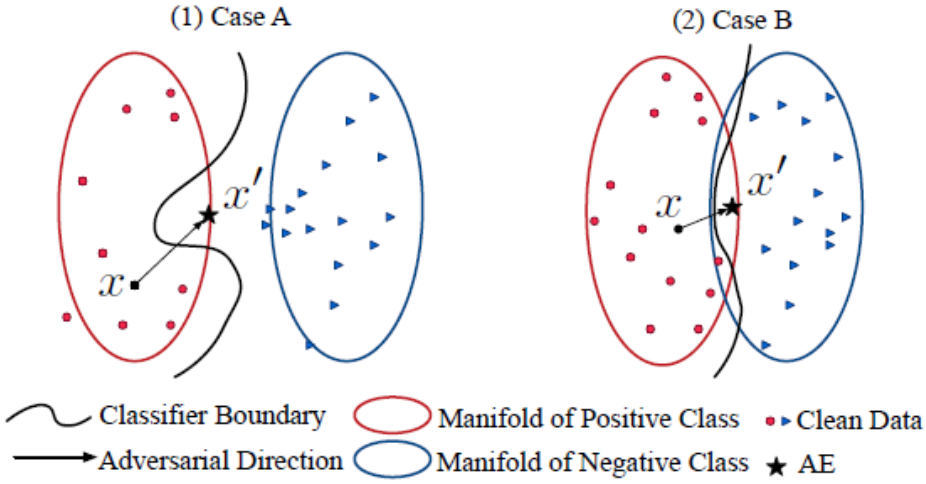

Figura 2.3: Esempio di funzionamento MANDA

Fonte: [9]

come benevolo. Ne deriva che, su x', l'output del Manifold e dell'IDS sono discordi. Supponendo adesso che il classificatore sia in grado di riconoscere i dati in input puliti (non avversari) con una accuratezza elevata, lo stesso attacco descritto sopra, fatto su un pacchetto pulito, porta il manifold e l'IDS a dare la stessa risposta.

Si può riassumere quanto detto dicendo che punti vicini che si trovano nello stesso manifold appartengono molto probabilmente alla stessa classe. Da questo possiamo ottenere una metrica di valutazione dei dati in input: se l'output tra IDS e Manifold è inconsistente si tratta di un esempio avversario, viceversa è un campione normale.

Per esaminare la componente di **Decision Boundary** si analizza il caso B.
Si assume che:

- se due manifold non sono completamente separabili (si intersecano), ma la maggior parte delle istanze è distinguibile;
- se l'IDS ha una accuratezza elevata nella classificazione;
- se x è un dato in input pulito e x' è il suo corrispondente a cui è stata aggiunta una perturbazione;
- se x' mantiene le proprietà essenziali di x (se x è malevolo anche x' lo sarà);

allora x' si trova, molto probabilmente, vicino alla linea di classificazione.

DB deve valutare se un valore in input si trovi vicino alla linea di classificazione. L'idea è quella di applicare a x' delle piccole perturbazioni, spesso

rumore gaussiano a media nulla e con una certa varianza. È molto probabile che, data la vicinanza alla linea di classificazione, questo valore perturbato cada adesso dall'altro lato. Provando con tanti valori casuali, si ha quindi una elevata instabilità nel risultato della classificazione dell'IDS. Al contrario, se x' fosse stato un valore lontano dalla linea, una piccola perturbazione non avrebbe modificato il risultato della classificazione. La seconda metrica utilizzata da MANDA è: se un input che viene leggermente perturbato porta a una elevata instabilità nella classificazione, questo è un esempio avversario.

I due metodi, Manifold e DB, vengono utilizzati contemporaneamente sul campione in input e, se almeno uno dei due lo classifica come avversario, viene immediatamente scartato.

Capitolo 3. Attacchi agli IDS

In questo capitolo verranno analizzati alcuni dei principali approcci riguardo l'adversarial machine learning pubblicati in letteratura scientifica.

Riprendendo quanto detto nei capitoli precedenti, lo scopo dell'adversarial machine learning è quello di costruire dei dati perturbati che portano il sistema vittima ad effettuare una classificazione errata. Oggi questa scienza sta diventando sempre più matura, gli attacchi sono sempre più complessi e questo garantisce prestazioni migliori, sia per quanto riguarda la qualità del campione perturbato, sia per la capacità di mantenere l'attaccante nascosto agli occhi del sistema vittima.

Alcuni degli approcci citati in questo capitolo costituiscono un punto di riferimento per lo sviluppo del metodo proposto, discusso nel capitolo 4.

Le differenti tecniche di attacco sono cinque e vengono analizzate a partire dalla più semplice, che è anche la meno efficace:

- **Brute-force Black-Box Attack (BFAM)** [22]: è un approccio a forza bruta che consiste nel modificare iterativamente la perturbazione introdotta negli attributi e nel valutare quale modifica porta alla generazione di esempi avversari sempre migliori.
- **Adversarial Attack against DoS IDS** [23]: **An improved Boundary Based Method**: implementa una particolare tecnica di ricerca della perturbazione basata sul confine di separazione tra dati malevoli e benevoli e che riduce il numero di query da fare al classificatore vittima.
- **Black-Box Adversarial ML Attack on Network Traffic Classification** [24]: si utilizza un modello sostituto che simula il comportamento del classificatore vittima e, su questo, si effettua un attacco transparent-box.
- **Black-Box Attack on Deep Anomaly Detectors** [25]: molto simile al metodo precedente. La differenza sta nella tipologia del modello sostituto utilizzato e nell'applicazione di una tecnica di riduzione del numero di query da effettuare al classificatore vittima.
- **Generative Adversarial Attacks Against IDS Using Active Learning** [26]: viene implementata una *generative adversarial network* (GAN) che apprende autonomamente la quantità di perturbazione da introdurre nel dato per renderlo avversario.

3.1 Brute-force Black-Box Attack (BFAM)

L'attacco, come indicato dal titolo, è un approccio a forza bruta in cui non vengono applicate particolari tecniche di ricerca del risultato, ma, in maniera quasi casuale, per ogni iterazione viene incrementato il valore di perturbazione finché non viene trovato un esempio avversario, oppure vengono superati dei limiti imposti dagli sviluppatori. Inoltre, sebbene venga definito un attacco di tipo black-box, l'algoritmo richiede, oltre all'output del classificatore (malevolo o benevolo), anche il *confidence score*, ovvero il valore di probabilità che quel dato appartenga alla classe specificata.

OMISSISS

3.2 Adversarial Attack against DoS IDS: An improved Boundary Based Method

OMISSISS

3.3 Black-Box Adversarial ML Attack on Network Traffic Classification

OMISSISS

3.4 Black-Box Attack on Deep Anomaly Detectors

OMISSISS

3.5 Generative Adversarial Attacks Against IDS Using Active Learning

OMISSISS

Capitolo 4. Sistema Proposto

In questo capitolo viene presentato un nuovo sistema di generazione degli esempi avversari, si descrivono nel dettaglio i vincoli imposti nell'introduzione della perturbazione e la particolare tecnica di implementazione dell'attacco.

Nei capitoli precedenti si è discusso degli IDS e di alcuni specifici attacchi che questi potrebbero subire. Tutto questo è necessario per comprendere il modo in cui lavora il sistema proposto, che consente di effettuare attacchi agli IDS tramite la generazione di esempi avversari. L'attacco proposto unisce molte tecniche precedentemente descritte nei capitoli 2 e 3, e crea un sistema altamente affidabile e generalista, capace di generare in poco tempo molti esempi avversari per qualsiasi tipologia di intrusione.

OMISSISS

4.1 Vincoli

OMISSISS

OMISSISS

4.1.1 Stima dei valori mancanti

OMISSISS

4.2 Modello sostituto

OMISSISS

4.3 Metodo di attacco

OMISSISS

Capitolo 5. Valutazioni Sperimentali

In questo capitolo verranno descritti i dettagli dell'implementazione in riferimento al modello sostituto e all'applicazione dei vincoli. Verranno inoltre presentanti tutti i valori associati ai parametri dell'attacco e i risultati sperimentali ottenuti tramite il metodo proposto.

5.1 Dataset utilizzato

OMISSISS

5.2 Creazione del modello sostituto

OMISSISS

5.3 Stima dei valori mancanti e applicazione dei vincoli

OMISSISS

5.4 Parametri

OMISSISS

5.5 Risultati

OMISSISS

Capitolo 6. Conclusioni

Con il presente lavoro viene descritto l'adversarial machine learning sui sistemi di rilevamento delle intrusioni, una tecnica di addestramento che, oltre ai tantissimi vantaggi, presenta diverse sfaccettature che, in alcuni casi, lo rendono un sistema poco efficace. Sicuramente simulare un attacco ad un sistema aiuta a comprendere quali sono le sue vulnerabilità e come risolverle. Questo vale ancora di più se si tratta di sistemi che hanno il compito di proteggere le reti da potenziali attacchi. Attraverso l'adversarial machine learning è possibile addestrare un IDS sugli esempi avversari che sfruttano proprio queste vulnerabilità, in modo da eliminarle, rendendo il sistema più robusto. Differenti problematiche, però, riducono le capacità dell'approccio. Una di queste è la difficoltà di comprendere se il campione avversario, ottenuto dopo la modifica, sia ancora malevolo oppure abbia perso la sua caratteristica. Non esiste ancora un metodo assoluto per determinarlo. Questo perché l'adversarial machine learning è nato nell'ambito delle reti neurali convoluzionali, che lavorano sulle immagini, per le quali è facile effettuare una valutazione sul campione finale tramite una semplice ispezione visiva.

OMISSISS

Per quanto riguarda le prospettive future, si potrebbe pensare di formulare una dimostrazione sulla correttezza della modifica del campione, in modo da poter affermare con certezza che il campione avversario generato rimanga malevolo.

Si potrebbe inoltre proseguire ricreando i pacchetti originari a partire dal vettore dei feature avversario creato dall'algoritmo. Questo lavoro risulta abbastanza complesso in quanto esistono attributi, soprattutto quelli riferiti ai tempi di attesa, difficilmente imitabili.

Infine, si potrebbe considerare la possibilità di automatizzare l'intero attacco a partire dalla scelta dei vincoli, in modo da fargli predisporre autonomamente tutto ciò che è necessario per creare il campione avversario. La difficoltà maggiore sta nella fase di stima dei valori degli attributi vincolati, per la quale è necessario stabilire un ordine di addestramento dei modelli di machine learning al fine di massimizzarne le prestazioni.

Per concludere, il presente lavoro ha consentito di approfondire un ambito di ricerca recente ancora oggi in via di sviluppo e di capire in che modo muoversi quando si affronta uno studio così complesso.

Riferimenti Bibliografici:

[1] Daniele Apiletti et al. «Characterizing network traffic by means of the NetMine framework». In: Computer Networks 53.6 (2009), pp. 774–789.

[2] Leyla Bilge et al. «Exposure: A passive dns analysis service to detect and report malicious domains». In: ACM Transactions on Information and System Security (TISSEC) 16.4 (2014), pp. 1–28.

[3] Leyla Bilge et al. «Exposure: Finding malicious domains using passive DNS analysis.» In: Ndss. 2011, pp. 1–17.

[4] Alan Bivens et al. «Network-based intrusion detection using neural networks». In: Intelligent Engineering Systems through Artificial Neural Networks 12.1 (2002), pp. 579–584.

[5] Hanen Brahmi, Imen Brahmi e Sadok Ben Yahia. «OMC-IDS: at the cross-roads of OLAP mining and intrusion detection». In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer. 2012, pp. 13–24.

[6] James Cannady. «Artificial neural networks for misuse detection». In: National information systems security conference. Vol. 26. Baltimore. 1998, pp. 443–456.

[7] Nuno Martins et al. «Adversarial machine learning applied to intrusion and malware scenarios: a systematic review». In: IEEE Access 8 (2020), pp. 35403–35419.

[8] Ishai Rosenberg et al. «Adversarial machine learning attacks and defense methods in the cyber security domain». In: ACM Computing Surveys (CSUR) 54.5 (2021), pp. 1–36.

[9] Ning Wang et al. «Manda: On adversarial example detection for network intrusion detection system». In: IEEE Transactions on Dependable and Secure Computing (2022).

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

[11] Pin-Yu Chen et al. «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.

[12] Ian Goodfellow. «Nips 2016 tutorial: Generative adversarial networks». In: arXiv preprint arXiv:1701.00160 (2016).

[13] Ian J Goodfellow, Jonathon Shlens e Christian Szegedy. «Explaining and harnessing adversarial examples». In: arXiv preprint arXiv:1412.6572 (2014).

[14] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi e Pascal Frossard. «Deepfool: a simple and accurate method to fool deep neural net- works». In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, pp. 2574–2582.

[15] Nicolas Papernot et al. «The limitations of deep learning in adversarial settings». In: 2016 IEEE European symposium on security and privacy (EuroS&P). IEEE. 2016, pp. 372–387.

[16] Tim Salimans et al. «Improved techniques for training gans». In: Advances in neural information processing systems 29 (2016).

[17] Christian Szegedy et al. «Intriguing properties of neural networks». In: arXiv preprint arXiv:1312.6199 (2013).

[18] Qiao Yan et al. «Automatically synthesizing DoS attack traces using generative adversarial networks». In: International Journal of Machine Learning and Cybernetics 10.12 (2019), pp. 3387–3396.

[19] Martin Arjovsky, Soumith Chintala e Léon Bottou. «Wasserstein generative adversarial networks». In: International conference on machine learning. PMLR. 2017, pp. 214–223.

[20] Naveed Akhtar, Jian Liu e Ajmal Mian. «Defense against universal adversarial perturbations». In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 3389–3398.

[21] Dongyu Meng e Hao 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.

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

[23] Xiao Peng, Weiqing Huang e Zhixin Shi. «Adversarial attack against dos intrusion detection: An improved boundary-based method». In: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). IEEE. 2019, pp. 1288–1295.

[24] Muhammad Usama et al. «Black-box adversarial machine learning attack on network traffic classification». In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC). IEEE. 2019, pp. 84–89.

[25] Aditya Kuppa et al. «Black box attacks on deep anomaly detectors». In: Proceedings of the 14th International Conference on Availability, Reliability and Security. 2019, pp. 1–10

[26] Dule Shu et al. «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.

[27] Didong Li, Minerva Mukhopadhyay e David B Dunson. «Efficient manifold and subspace approximations with spherelets». In: arXiv preprint arXiv:1706.08263 (2017)

[28] Vasisht Duddu. «A survey of adversarial machine learning in cyber warfare». In: Defence Science Journal 68.4 (2018), p. 356.

[29] Weilin Xu, David Evans e Yanjun Qi. «Feature squeezing: Detecting adversarial examples in deep neural networks». In: arXiv preprint ar Xiv:1704.01155 (2017).36

[30] Maria Rigaki. Adversarial deep learning against intrusion detection classifiers. 2017.

[31] Agate, V., Paola, A. D., Lo Re, G., & Morana, M. (2016). A simulation framework for evaluating distributed reputation management systems. In Distributed Computing and Artificial Intelligence, 13th International Conference (pp. 247-254). Springer, Cham.

[32] Agate, V., De Paola, A., Gaglio, S., Lo Re, G., & Morana, M. (2016, June). A framework for parallel assessment of reputation management systems. In Proceedings of the 17th International Conference on Computer Systems and Technologies 2016 (pp. 121-128).

[33] Gaglio, S., Giammanco, A., Lo Re, G., & Morana, M. (2022). Adversarial Machine Learning in e-Health: Attacking a Smart Prescription System. In International Conference of the Italian Association for Artificial Intelligence (pp. 490-502). Springer, Cham.

[34] Agate, V., De Paola, A., Lo Re, G., & Morana, M. (2016). Vulnerability Evaluation of Distributed Reputation Management Systems. In VALUETOOLS.

[35] Concone, F., Lo Re, G., Morana, M., & Das, S. K. (2022). SpADe: Multi-Stage Spam Account Detection for Online Social Networks. IEEE Transactions on Dependable and Secure Computing.

[36] Agate, V., De Paola, A., Lo Re, G., & Morana, M. (2018, October). A platform for the evaluation of distributed reputation algorithms. In 2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT) (pp. 1-8). IEEE.

[37] Agate, V., Paola, A. D., Lo Re, G., & Morana, M. (2021). A simulation software for the evaluation of vulnerabilities in reputation management systems. ACM Transactions on Computer Systems (TOCS), 37(1-4), 1-30.

[38] Agate, V., De Paola, A., Lo Re, G., & Morana, M. (2020). DRESS: A Distributed RMS Evaluation Simulation Software. International Journal of Intelligent Information Technologies (IJIIT), 16(3), 1-18.

[39] Marco Barreno et al. «The security of machine learning». In: Machine Learning 81.2 (2010), pp. 121–148.

[40] Anna L Buczak e Erhan Guven. «A survey of data mining and machine learning methods for cyber security intrusion detection». In: IEEE Communications surveys & tutorials 18.2 (2015), pp. 1153–1176

[41] Ansam Khraisat et al. «Survey of intrusion detection systems: techniques, datasets and challenges». In: Cybersecurity 2.1 (2019), pp. 1–22.

[42] William Stallings. Cryptography and network security, 7th ed. Pearson Education, 2016.

[43] Ansam Khraisat, Iqbal Gondal e Peter Vamplew. «An anomaly intrusion detection system using C5 decision tree classifier». In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer. 2018, pp. 149–155.

[44] Symantec, "Internet security threat report 2017," April, 7017 2017, vol. 22 Available: https://www.symantec.com/content/dam/symantec/docs/reports/ istr-22-2017-en.pdf

[45] Anna L Buczak e Erhan Guven. «A survey of data mining and machine learning methods for cyber security intrusion detection». In: IEEE Communications surveys & tutorials 18.2 (2015), pp. 1153–1176.

[46] Ankush Meshram e Christian Haas. «Anomaly detection in industrial networks using machine learning: a roadmap». In: Machine Learning for Cyber Physical Systems. Springer, 2017, pp. 65–72.

[47] Gideon Creech. «Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks». Tesi di dott. UNSW Sydney, 2014.

[48] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya e Jugal K Kalita. «Network anomaly detection: methods, systems and tools». In: Ieee communications surveys & tutorials 16.1 (2013), pp. 303–336.

[49] Wei-Chao Lin, Shih-Wen Ke e Chih-Fong Tsai. «CANN: An intrusion detection system based on combining cluster centers and nearest neighbors». In: Knowledge-based systems 78 (2015), pp. 13–21.

[50] Okan Can e Ozgur Koray Sahingoz. «A survey of intrusion detection systems in wireless sensor networks». In: 2015 6th international conference on modeling, simulation, and applied optimization (ICMSAO). IEEE. 2015, pp. 1–6.

[51] Salma Elhag et al. «On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems». In: Expert Systems with Applications 42.1 (2015), pp. 193–202.

[52] Nong Ye et al. «Multivariate statistical analysis of audit trails for host-based intrusion detection». In: IEEE Transactions on computers 51.7 (2002), pp. 810–820.

[53] Agate, V., Concone, F., & Ferraro, P. (2018, June). Wip: Smart services for an augmented campus. In 2018 IEEE International Conference on Smart Computing (SMARTCOMP) (pp. 276-278). IEEE.

[54] Timilsina, A., Khamesi, A. R., Agate, V., & Silvestri, S. (2021). A reinforcement learning approach for user preference-aware energy sharing systems. IEEE Transactions on Green Communications and Networking, 5(3), 1138-1153.

[55] Agate, V., De Paola, A., Ferraro, P., Lo Re, G., & Morana, M. (2021). SecureBallot: A secure open source e-Voting system. Journal of Network and Computer Applications, 191, 103165.

[56] Agate, V., Concone, F., & Ferraro, P. (2021, November). A Resilient Smart Architecture for Road Surface Condition Monitoring. In The Proceedings of the International Conference on Smart City Applications (pp. 199-209). Springer, Cham.

[57] Agate, V., D'Anna, F. M., De Paola, A., Ferraro, P., Re, G. L., & Morana, M. (2022). A Behavior-Based Intrusion Detection System Using Ensemble Learning Techniques.

[58] Jouni Viinikka et al. «Processing intrusion detection alert aggregates with time series modeling». In: Information Fusion 10.4 (2009), pp. 312-324.

[59] Neil Walkinshaw, Ramsay Taylor e John Derrick. «Inferring extended finite state machine models from software executions». In: Empirical Software Engineering 21.3 (2016), pp. 811–853.

[60] Ivan Studnia et al. «A language-based intrusion detection approach for automotive embedded networks». In: International Journal of Embedded Systems 10.1 (2018).

[61] Gisung Kim, Seungmin Lee e Sehun Kim. «A novel hybrid intrusion detection method integrating anomaly detection with misuse detection». In: Expert Systems with Applications 41.4 (2014), pp. 1690–1700.

[62] De Paola A., Favaloro S., Gaglio S., Lo Re G., Morana M., Malware detection through low-level features and stacked denoising autoencoders, (2018) CEUR Workshop Proceedings, 2058

[63] De Paola A., Gaglio S., Lo Re G., Morana M. A hybrid system for malware detection on big data, (2018) INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, pp. 45 – 50, DOI: 10.1109/INFCOMW.2018.8406963

[64] Agate V., Curaba M., Ferraro P., Lo Re G., Morana M., Secure e-voting in smart communities, (2020) CEUR Workshop Proceedings, 2597, pp. 1 – 11

[65] Concone F., De Paola A., Lo Re G., Morana M., Twitter analysis for real-Time malware discovery, (2017) 2017 AEIT International Annual Conference: Infrastructures for Energy and ICT: Opportunities for Fostering Innovation, AEIT 2017, 2017-January, pp. 1 – 6, DOI: 10.23919/AEIT.2017.824055

[66] Bordonaro A., De Paola A., Lo Re G., Morana M., Smart Auctions for Autonomic Ambient Intelligence Systems, (2020) Proceedings - 2020 IEEE International Conference on Smart Computing, SMARTCOMP 2020, art. no. 9239687, pp. 180 – 187, DOI: 10.1109/SMARTCOMP50058.2020.00043