

ALMA MATER STUDIORUM - UNIVERSITÀ DEGLI STUDI DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

PROGETTAZIONE DI UN INTRUSION DETECTION SYSTEM SU PIATTAFORMA BIG DATA

Tesi di Laurea Magistrale in
DATA MINING

Relatore
Prof. MATTEO GOLFARELLI

Presentata da
ALESSIO ADDIMANDO

Co-relatore
Dott. ENRICO GALLINUCCI

Sessione di Laurea II
Anno Accademico 2017 – 2018

A mio nonno Salvatore Antonio

Indice

Introduzione	1
1 Stato dell'arte	5
1.1 Evoluzione del data management: i Big Data	5
1.2 Analisi di Data Stream	17
1.3 Cybersecurity e Intrusion Detection System su piattaforma Big Data	20
1.3.1 Metodi e criteri di valutazione	22
1.3.2 L'utilizzo di streaming di dati	26
1.3.3 L'introduzione di tecniche di Data Mining e Machine Learning	29
1.3.4 Il ruolo dei Big data nella Network Intrusion Detection	30
2 Apache Spark	35
2.1 Architettura	37
2.1.1 Driver program	39
2.1.2 Executor	40
2.1.3 Cluster manager	40
2.2 Linguaggi supportati	40
2.3 Astrazioni fondamentali: RDD e DAG	41
2.4 Spark SQL	44
2.4.1 Dataset	45
2.4.2 Dataframe	45
2.5 Spark Streaming	45

2.5.1	Modelli di elaborazione	46
2.5.2	Conseguenze dei micro-batch	48
2.6	Spark MLlib	48
2.7	Prestazioni e benchmarking	50
3	Definizione del problema	53
3.1	Descrizione architettura di monitoraggio attuale: <i>Suricata</i> e <i>Alien Vault</i> . .	53
3.2	Problematiche	56
3.3	Requisiti	57
4	Analisi e prototipo: Styx	59
4.1	Dataset e sorgenti	59
4.2	Architettura generale e componenti	63
4.2.1	Componente DSP e scenari d'attacco considerati	64
4.2.2	Componente ML e algoritmi di mining utilizzati	70
4.3	Scelte tecnologiche e hardware	74
4.4	Implementazione	76
4.4.1	Raccolta dati, preprocessing e ricostruzione delle sessioni	77
4.4.2	Integrazione delle sorgenti e arricchimento	79
4.4.3	Resampling e bilanciamento del dataset	81
4.4.4	GDPR e anonimizzazione delle informazioni sensibili	82
4.5	Risultati	83
	Conclusioni	89
	Bibliografia	91
	Ringraziamenti	97

Elenco delle figure

1.1	Dualità tra il semplice Data Management e il post processo di Analytics . . .	9
1.2	Componenti core dell'architettura di Hadoop 2.0	13
1.3	Design e i vari tipi di nodo della piattaforma Hadoop.	15
1.4	Aspetti cardine di un sistema DSP.	18
1.5	Struttura generale di un'intrusione	25
1.6	Rappresentazione di un Real-Time Intrusion Detection System	32
1.7	Possibile scelta tecnologica per un'architettura di un Real-Time Intrusion Detection System	33
2.1	Logo Apache Spark	36
2.2	Moduli principali Apache Spark	38
2.3	Modello architetturale di Spark	38
2.4	Principali linguaggi usati in Spark	41
2.5	Grafo degli stage	42
2.6	Stage, task e il processo di submitting di un job in Apache Spark	43
2.7	Ciclo di vita di un RDD in Spark	43
2.8	Creazione di un DStream in Apache Spark	46
2.9	Gestione <i>batch-oriented</i> di uno stream di dati prima di essere elaborato dallo Spark Engine	46
2.10	Comparazione delle prestazioni tra scenari di esecuzione (problema di re- gressione lineare) in Apache Hadoop e Apache Spark	51
3.1	Componenti principali sistema USM™Alien Vault	56

4.1	Struttura sorgente 1 - NetFlow	60
4.2	Struttura sorgente 2 - Suricata	62
4.3	Struttura sorgente 3 - Alien Vault	62
4.4	Architettura generale Styx	63
4.5	Port Scan versione TCP SYN	66
4.6	Port Scan versione TCP FIN	67
4.7	HTTP Flood - DDoS a livello applicativo	68
4.8	SYN Flood - DDoS a livello di protocollo	69
4.9	Amplification DNS - DDoS volumetrico	70
4.10	Random Forest composto da 2 soli alberi	71
4.11	Apache Spark UI	75
4.12	Asset tecnologico di Styx	76
4.13	Stima della quantità (in byte) di header generati per la sorgente 1 - NetFlow	77
4.14	Stima della quantità (in byte) di dati scambiati dalla rete in base alle informazioni fornite da NetFlow	78
4.15	Ricostruzione delle sessioni di comunicazione sorgente 1 - NetFlow	79
4.16	Schema logico per l'integrazione delle tre sorgenti Styx	79
4.17	Tabelle di arricchimento dell'informazione per il dataset finale	80
4.18	Tabella dataset finale componente ML	81
4.19	Distribuzione delle sessioni in base alla durata (in secondi)	84
4.20	Distribuzione delle sessioni in base al volume di dati scambiato (in byte)	84
4.21	Sessioni insolite riconducibili a tentativi di port-scan	85
4.22	Statistiche in merito alla localizzazione delle sessioni analizzate e i tentativi di intrusione rilevati	86
4.23	Generalizzazione di un percorso decisionale in Styx per l'individuazione di una possibile sessione sospetta. L'accuratezza del modello è di circa l'80%.	88

Introduzione

Negli ultimi anni, nel panorama digitale, è stato rilevato un ingente aumento del numero di dispositivi e utenti con accesso ad Internet. L'introduzione degli smartphone, dei numerosi servizi online e del mondo dell'Internet of Things, grazie al quale sensori intelligenti e macchine risultano essere agenti nella rete, ha portato per ogni utente un numero maggiore di dispositivi, direttamente collegati tra loro e alla rete globale. Proporzionalmente a questi fattori ogni giorno vengono generati continuamente, e in qualsiasi contesto, grandi quantità di dati difficili da gestire. Nasce la necessità di riorganizzare gli asset aziendali, ma in generale le infrastrutture, per far fronte ad un calibro di informazione del genere. Affiancato a questo non si esclude la possibilità che all'interno dei giganteschi dataset possano essere presenti informazioni che, a livello strategico, rappresentano un vero e proprio strumento non solo per migliorare il rapporto con il cliente o la sua gestione interna ma anche per avere un vantaggio verso aziende avversarie. Questi dati, elaborati correttamente, permettono di aiutare il processo di analisi decisionale aziendale a qualsiasi stadio, dalla produzione di un prodotto fino all'azione di marketing che gli concerne. L'insieme di queste motivazioni da vita al concetto dei *Big Data*.

Affiancato a questo, la grande quantità di macchine e utenti in rete ha esponenzialmente aumentato, come diretta conseguenza, anche il numero di attacchi informatici, atti nella stragrande dei casi all'impropriazione illecita di dati sensibili o a provocare disservizi nelle reti private. Un esempio evidente è il campus universitario di Forlì-Cesena che stima costantemente e simultaneamente attive circa 3000 macchine interconnesse tra di loro e con la rete esterna (comprensivo di server, stampanti, videocamere, cluster etc).

La grande quantità di risorse connesse quotidianamente in rete assume una certa importanza visti i dati sensibili che gestiscono e immagazzinano. Per questo motivo, l'intera rete ha la necessità primaria di essere costantemente monitorata per garantirne il funzionamento e la sicurezza in termini di affidabilità e integrità contro attacchi informatici. Tuttavia, nonostante l'architettura di monitoraggio venga continuamente aggiornata ed è soggetta a costante controllo e manutenzione, questa presenta colli di bottiglia evidenti e limitazioni nell'elaborazione dell'intero traffico di rete a causa della grande quantità di dati (perdita di informazioni e analisi a livelli di aggregazione limitati). Per far fronte a questa problematica di sicurezza informatica lo scopo di questa tesi è stato quello di unire due ambiti informatici ben distinti (Big Data e sicurezza informatica) integrando al processo di sicurezza di questa rete privata una piattaforma di analisi e monitoraggio per il rilevamento di intrusioni (intrusion detection system), su piattaforma Big Data. Il progetto, a tal fine, affronta un'analisi quantitativa (mediante report e statistiche) e qualitativa (individuazione di nuova conoscenza) dei flussi di dati scambiati nella rete del campus universitario di Forlì-Cesena. Dalle comunicazioni tra terminali interni, e verso l'esterno, sono state individuate sessioni di comunicazione dove, in questa particolare applicazione, il processo analitico si impone, come primo scopo, l'individuazione, di eventuali situazioni "sospette" (attacchi DDos, botnet, malware). Il prototipo realizzato (denominato *Styx*), come spiegato dettagliatamente nel corso del documento, sfrutta tecniche di data stream processing (elaborazione di dati real-time) e di machine learning (tecniche di apprendimento per estrazione di modelli predittivi) per potenziare l'attuale sistema di monitoraggio della rete universitaria cercando di individuare pattern utili per il controllo dei dati in ingresso rispetto a potenziali minacce e scenari di attacco. Il documento è diviso in quattro capitoli: il primo (1) capitolo fornisce una panoramica compilativa su quello che attualmente offre lo stato dell'arte in merito ai due ambiti che convergono in questo progetto. Il secondo (2) capitolo presenta una descrizione asettica della tecnologia usata, quindi del framework *Apache Spark*, i suoi moduli e i dettagli più tecnici. Il terzo (3) capitolo fornisce una descrizione dettagliata della definizione del problema, dell'architettura di partenza e dei vari requisiti emersi in fase di analisi e progettazione pre-implementativa. Nell'ultimo

e quarto (4) capitolo è presente una descrizione della realizzazione del progetto *Styx*; dalla descrizione dei dataset e del design architettonico fino all'effettiva implementazione del prototipo, le attività di test e validazione e i risultati ottenuti.

Capitolo 1

Stato dell'arte

1.1 Evoluzione del data management: i Big Data

La ricostruzione storica dell'evoluzione del data management riporta il concetto di dato agli anni '60, quando quest'ultimo non rappresentava nient'altro che una semplice serie di caratteri e cifre, memorizzati in forma statica. Le uniche tecnologie disponibili, pertanto, permettevano di immagazzinare dati grazie all'ausilio di supporti magnetici che le aziende e le banche, uniche realtà dove era sensato poter parlare di data storage, utilizzavano per un'analisi sommaria limitata alla loro semplice estrazione e primitiva elaborazione, pur sempre non automatizzata. Con l'introduzione dei primi DBMS (cioè *database management system*), il data management incontra la sua evoluzione sempre più veloce che presenta come milestone più significativa il progetto di prima Base di Dati relazionale della Relational Software Inc. (cioè oggi famosa come *Oracle Corp.*)[1]. Con l'avvento dei database relazionali e del linguaggio di interrogazione *SQL* (cioè *Structured Query Language*), negli anni '80, l'analisi dei dati assume una certa dinamicità: estrarre in maniera semplice i dati grazie all'esecuzione di una query permette la gestione di quest'ultimi sia in maniera più aggregata, sia a livello di massimo dettaglio; sempre più aderente, quindi, alle esigenze dell'utente che interagisce con la base dati. Le attività di analisi avvengono su basi di dati operazionali, ovvero sistemi di tipo OLTP (cioè *On Line Transaction Processing*) caratterizzati e ottimizzati prevalentemente per operazioni di tipo transazionale

e che quindi mettono a disposizione funzioni semplici come l'inserimento, la cancellazione e la modifica dei record. In merito a ciò è possibile sancire che la maggior parte dei sistemi OLTP si contraddistingue per una limitata possibilità di storicizzazione dei dati. Con l'arrivo della rete Internet e della distribuzione su vasta scala dei personal computer, infatti, nel giro di pochi anni, l'insorgenza di nuovi bisogni aziendali diventa inevitabile: aumentano esageratamente le sorgenti di dati eterogenee e il gran numero di applicazioni B2B produce dati non replicati su macchine diverse; non garantendo quindi uniformità tra le informazioni. Per far fronte a questa differenza e alla necessità di estrarre un nuovo calibro di informazione dai propri dati, il primitivo concetto di Database Relazionale si evolve trasversalmente nel moderno *Data Warehouse*. La differenza sostanziale rispetto ai primi si basa sull'assunzione che i database sono progettati in maniera tale da avere un'informazione estremamente strutturata e priva di ridondanze dovuta a più processi di normalizzazione. Il *Data Warehouse*, al contrario, è caratterizzato da un tipo di informazione meno normalizzata e sintetica, memorizzata ad un livello di dettaglio prettamente minore dove è perfino consentita la presenza di ridondanze; tant'è vero che, partendo da uno schema logico relazionale, per la loro definizione iniziale si fa ricorso ad un processo di "denormalizzazione" del classico database. Mentre un database relazionale contiene dati relativi ai processi operativi aziendali in una struttura rigida, fornendo funzionalità semplici come letture, inserimenti, modifiche e cancellazioni, il *Data Warehouse* è progettato in modo da fornire funzionalità per individuare relazioni tra i dati che possono essere utilizzate per definire delle tecniche strategiche (cioè operazioni di *Drill-down*, *Roll-up*, etc). La grande mole di dati oggi prodotti e le necessità sempre più emergenti di gestirli al fine di estrarne valore, implica, a partire dal nuovo millennio, la nascita di un nuovo fenomeno oggi diffusissimo nel panorama della Business Intelligence, e in generale della Computer Science: il *Big Data management*.

Definizione, principali sorgenti e applicazioni

Con il termine Big Data, secondo la definizione di *Gartner Group*, si intende una grande quantità di dati e informazioni che è possibile trovare in formati strutturati e destrutturati,

generati ad alta velocità con risorse informative estremamente eterogee.

I dataset, inoltre, richiedono tecniche e tecnologie avanzate di elaborazione delle informazioni per consentirne: cattura, stoccaggio, distribuzione, gestione e analisi. Il processo analitico avviene, principalmente, per assistere il processo decisionale aziendali che sfruttano i risultati ottenuti per rimodellare i processi e gli indicatori di business (cioè *KPI*) in maniera sempre più specializzata e mirata.

Dalla canonicità della definizione prima espressa, inoltre, è semplice estrapolare tre concetti cardine che caratterizzano questo tipo di dati:

- Volume
- Varietà
- Velocità

Le tre chiavi di definizione, anche dette comunemente *The Three V's* [2] riescono a descrivere le caratteristiche la “struttura” con la quale i Big Data si identificano. Di seguito una spiegazione sistetica di ognuno di esse.

Il *Volume*, naturalmente, si riferisce alla grandezza dei dati. Insita anche nella denominazione, l'enorme quantità di memoria che possono occupare parte dall'ordine di grandezza del Terabyte (cioè 1024 Gbyte) sfociando senza difficoltà nel Petabyte (1M Gbyte) e così via. E' riportato, ad esempio, che Facebook riesca ad elaborare fino a un milione di fotografie/s e poiché un petabyte equivale a 1024 terabyte, le stime suggeriscono che il social network detenga un database che occupa uno spazio di oltre 30 petabyte. Nel corso degli ultimi anni, con l'avvento del Web 2.0, dei sensori nella ormai nota *Internet of Things* e la grande diffusione dei sistemi gestionali, la crescita dei dati prodotti è esponenziale. Inoltre, poiché il valore spesso deriva proprio dall'aver a disposizione grandi quantità di dati, la propensione allo "scarto", quindi a mantenere la maggior parte dell'informazione, è praticamente ridotta all'osso.

La *Varietà* si riferisce alla eterogeneità strutturale in un insieme di dati. I progressi tecnologici permettono alle imprese di utilizzare vari tipi di dati in forma strutturata, semi strutturata o anche non strutturata. I dati strutturati, che costituiscono solo il 5% di tutti

i dati esistenti, sono i dati tabulari più comuni, presenti solitamente in fogli di calcolo (cioè *Excel*) o database tradizionali. Un linguaggio testuale per lo scambio di dati Web, come può essere XML o la notazione JSON, ad oggi più utilizzata come standardizzazione, sono invece tipici esempi di dati semistrutturati. Testi, immagini, audio e video sono invece esempi di dati non strutturati, ossia posseggono un'organizzazione strutturale non compatibile al calcolatore e non utilizzabile immediatamente ai fini di analisi. La varietà dei dati rappresenta decisamente un'ulteriore vantaggio al processo di “digitalizzazione” per le aziende che ricercano sempre di più attività di profilazione ad-hoc dei proprio clienti e una gestione intelligente delle proprie risorse.

La *Velocità* è il terzo aspetto da considerare nella definizione e vive in concomitanza con l'elevato volume sopracitato. Usualmente sono generati con una rapidità tale da accendere la sfida tra le aziende nell'analizzare le informazioni con altrettanta rapidità, estrapolare le informazioni utili per il business e identificare un problema, una tendenza o un'opportunità di guadagno. Il tutto necessariamente rientrando, in ogni caso, in un intervallo di tempi di elaborazione minimizzato rispetto alla possibile concorrenza del mercato. E' necessario far ricadere le proprie scelte tecnologiche su componenti applicativi che permettono di lavorare in tempo, approssimativamente, reale. Da qui nasce appunto il concetto di *data streaming*, successivamente analizzato (vedi sezione 1.2). Affiancate a quest'ultime, sono state successivamente introdotte[3] altre tre "V" che ne descrivono a maggior dettaglio la natura onde evitare possano essere banalmente confusi con semplici grandi quantità di dati immagazzinati nel tempo.

- **Valore:** un concetto che assume una dimensione prettamente economico-strategica. Basandosi sulla definizione di Oracle, i Big Data sono spesso caratterizzati da un "*valore a bassa densità*". Ciò significa che i dati ricevuti nella forma originale, di solito, hanno un valore bassissimo rispetto al volume in entrata. Questo implica che, proporzionalmente parlando, un valore elevato può essere ottenuto soltanto analizzando grandissimi volumi di tali dati.
- **Veridicità:** concetto abbastanza relativo e difficilmente quantificabile, rappresenta l'inattendibilità di alcune sorgenti di dati. Un esempio triviale è dato dai dati raccolti

dai social media dove la natura dei dati è incerta in quanto riconducibile a componenti umane facilmente condizionabili e poco deterministiche. Nonostante ciò, chiaramente, anche queste assumono la loro importanza tanto da rendere l'analisi di dati prodotti dai social un altro aspetto dei Big Data rivolto allo sviluppo di strumenti e algoritmi per la "normalizzazione" forzata di quest'ultimi in maniera consistente.

- Variabilità (e complessità): sono due ulteriori dimensioni di dati di grandi dimensioni. La variabilità si riferisce alla variazione della frequenza di generazione dei flussi di dati in termini di grandezza capacità. Spesso, la velocità dei Big Data non è costante nel tempo, presentando picchi e naturalmente grandi rallentamenti.

Affiancata alla definizione diviene logico fornire una, seppur breve, carrellata su quelli che attualmente sono i principali usi non escludendo un'ulteriore evoluzione degli stessi. Il Data Management rappresenta solo parte integrante del processo applicativo Big Data nella sua interezza. Il loro "ciclo di vita", a tal proposito, parte da un copioso e rapido input con acquisizione iniziale technology-oriented fino al raggiungimento finale con un output decisamente più business-oriented. In sostanza questo lascia intendere che il valore potenziale dell'input venga effettivamente sbloccato soltanto quando viene concretamente sfruttato per guidare il processo decisionale aziendale (cioè azioni di marketing, realizzazione di report dinamici). Per far ciò, le aziende sono molto orientate alla ricerca continua di elevate capacità di rapida trasformazione di grandi volumi di dati.

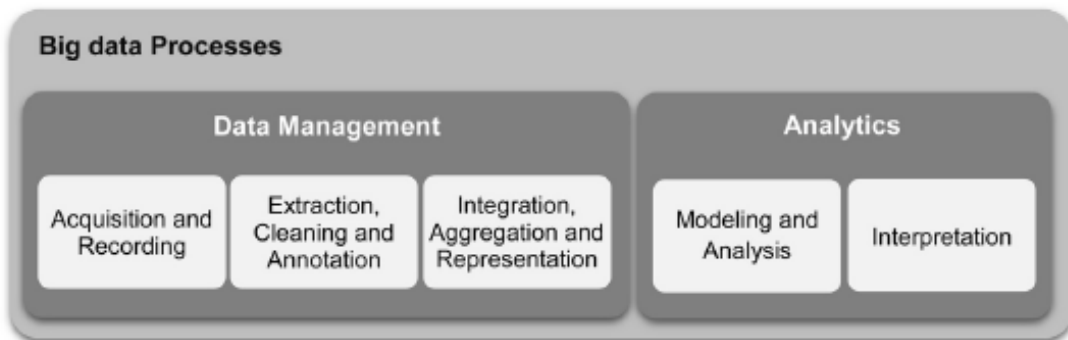


Figura 1.1: Dualità tra il semplice Data Management e il post processo di Analytics

Il processo generale può essere suddiviso in cinque fasi, illustrate in figura 1.1, che convergono nei due principali sotto-processi: *la gestione e l'analisi*. La prima coinvolge i processi e le tecnologie di supporto per acquisire e memorizzare gli elementi poi oggetto di analisi nella fase successiva. La seconda, nella parte destra, si riferisce alle tecniche utilizzate per analizzare e estrarre conoscenza ad alto valore aggiunto. A tal merito, l'analisi dei dati può essere vista come un sotto-processo nel processo globale Big Data. Sempre dalla figura 1.1 si evince l'unione concettuale ma non applicativa tra il tradizionale Data Management e un più avanzato stadio di analitica indipendente dotato di sue metriche, modelli e campi di azione. Ad approfondimento di quanto appena presentato, si riporta di seguito solo una possibile, ma non unica, divisione delle principali specializzazioni che l'analisi dei Big Data offre ad oggi nel panorama tecnologico attuale:

- Multimedia Analytics.
- Social Media Analytics.
- Descriptive, prescriptive and predictive Analytics.
- User Profiling.

La sorgente dei dati multimediali è facilmente intuibile; si tratta principalmente di analisi di documenti testuali, tracce audio, video e immagini, Tutte posseggono stampo prettamente statistico. Il text mining, che avviene tramite tecniche supervisionate e non di *machine learning* (cioè *classificazione, clustering, etc*) e algoritmi di cosiddetta *information extraction* (i.e *IE*), consente alle aziende di convertire grandi volumi di testo human-generated (cioè dati non strutturati) in una sintesi significativa che supporta il processo decisionale. Per esempio, analisi del testo può essere utilizzata per prevedere il mercato azionario sulla base delle informazioni estratte da notizie finanziari. Oppure analizzare un gran numero di telefonate effettuate da un centro di customer care può migliorarne sensibilmente la qualità poiché permette di individuarne vincoli e attrito nei processi di assistenza più complessi. In generale, si possono riassumere tre tipi di analisi dei Big Data: descrittive, prescrittive, predittive.

- Le analisi *descrittive* fanno esattamente ciò che il nome suggerisce, sintetizzano o descrivono i dati grezzi e ne fanno qualcosa che è interpretabile dagli esseri umani. Nello specifico vengono analizzati gli eventi passati, dove per eventi passati ci si riferisce a qualsiasi punto del tempo che si è verificato un evento, sia che sia un minuto fa o sia un mese fa. Le analisi descrittive sono utili in quanto consentono alle organizzazioni di imparare dai comportamenti passati e di aiutarli a capire come potrebbero influenzare i risultati futuri. Le statistiche descrittive sono utili per mostrare per esempio, il totale dei prodotti stock presenti in magazzino o la spesa media per cliente. Le organizzazioni devono quindi utilizzare le analisi descrittive quando vogliono capire, a livello aggregato, cosa sta succedendo nella loro società.
- Le analisi *prescrittive* facilitano gli utenti a “prescrivere” diverse azioni possibili per implementare e guidare l’attività verso una soluzione. L’analisi prescrittiva è tutta sulla consulenza. Tenta di quantificare l’effetto delle future decisioni per consigliare i possibili risultati prima che esse siano effettivamente adottate. L’analisi prescrittiva non solo prevede che cosa accadrà, ma spiega anche perché accadrà e fornisce le raccomandazioni in merito alle azioni che sfruttano queste previsioni. Le analisi prescrittive sono complesse da amministrare e la maggior parte delle aziende non le utilizza ancora. Tuttavia, quando vengono implementate correttamente, possono avere un grande impatto su come le imprese adottano decisioni e in tal modo, aiutarle a fornire i prodotti giusti al momento giusto, ottimizzando così l’esperienza del cliente grazie anche ad una profilazione adeguata dell’utente stesso (cioè User Profiling) monitorizzandone caratteristiche, preferenze d’ogni genere e attitudini.
- Le analisi *predittive* comprendono una varietà di tecniche finalizzate a “prevedere”, come è intuibile dalla denominazione, i risultati futuri sulla base di dati storici e attuali. Basandosi su vecchi modelli, dati storici certi e/o survey si cercano di scoprire nuovi approcci alle situazioni aziendali e non, prevedendo quindi errori e cali. Una sorta di “contabilità direzionale” basandosi però su moli di dati semi strutturati. Al contrario se l’origine dei dati appare completamente illogica, si cercano di scoprire

nuovi pattern e relazioni che hanno alta probabilità di ripresentarsi in futuro. Muovendoci in ambito statistico è necessario sempre considerare un margine di errore che i nuovi modelli analitici cercano sempre più di minimizzare. Nella pratica, il concetto può essere applicato a quasi tutte le discipline; dal prevedere la rottura di una lavatrice in base al flusso di dati provenienti dalle diverse migliaia di sensori (cioè *prognostic maintenance*) alla previsione delle prossime mosse di acquisto dei clienti in base a quello che comprano (cioè *recommendation system*) fino a individuare anomalie in una rete di macchine aziendale, o universitaria, facendo riferimento a scenari di cybersecurity noti ma anche addestrando sistemi appositi alla continua analisi e al continuo monitoraggio (per esempio *Intrusion Detection System*) dei flussi di dati scambiati basandosi su dataset storici (cioè *logs, alerts, etc*).

In quest'ultimo è fondamentalmente racchiuso il fulcro applicativo del progetto di tesi, esplicito nel dettaglio più avanti, nel capitolo 4. Al suo interno, le tecniche di analisi predittiva sono suddivise in due gruppi ben divisi a seconda dell'origine che il campione dei dati possiede.

La prima piattaforma di sviluppo per i BigData: Apache Hadoop

Finora è stato affrontato un discorso generale sul fenomeno dei Big Data evidenziando caratteristiche, importanti applicazioni e tecnologie di storage dove è necessità di lavorare con database non convenzionali e infrastrutture scalabili è sempre più evidente. Le tecnologie tradizionali come basi di dati relazionali faticano sensibilmente se applicate nell'uso dei Big Data e l'architettura hardware parallela. Anzi, queste ne mettono in luce una importante complicazione che risiede fondamentalmente nella gestione e nel calcolo di grandi moli di dati in tempo, approssimativamente, reale aggiungendoci anche la presenza di dati non strutturati. Tale esigenza, ha portato alla definizione di nuovi modelli di gestione dei dati ma anche a nuove logiche architetturali necessarie per l'elaborazione di quest'ultimi in modo efficiente e, soprattutto, scalabile. A tal merito, la più diffusa nel panorama attuale è la piattaforma ApacheTM Hadoop[®] [4]. Hadoop è il più conosciuto framework open-source

finalizzato alla memorizzazione e la gestione distribuita di grandi quantità di dati su cluster di macchine solitamente *commodity hardware*, la sua ultima release è *Hadoop 2.X*. Anche detto *commodity cluster computing*, il commodity hardware è l'uso di un gran numero componenti hardware, ovvero dispositivi a prezzi accessibili che sono generalmente compatibili con altri dispositivi di questo tipo, spesso collegati in rete per avere più potenza di calcolo e di elaborazione a un basso costo. La piattaforma presenta, a livello logico, una struttura prettamente modulare con alto fattore di fault-tolerance. Tutti i moduli, infatti, sono progettati in modo tale che il software del framework gestisca automaticamente gli eventuali eventi di danneggiamento dell'hardware, molto più frequenti in un sistema di calcolo parallelo e distribuito. La struttura minimizzata presenta un file system distribuito chiamato Hadoop Distributed File System (cioè *HDFS*), un componente di elaborazione *MapReduce*, un resource manager detto YARN e una serie di altri moduli per la processazione dei dati seguenti paradigmi differenti dal MapReduce prima citato (figura 1.2).

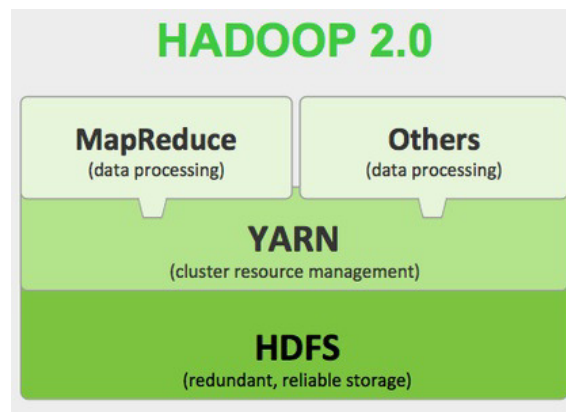


Figura 1.2: Componenti core dell'architettura di Hadoop 2.0

Il processo inizia con la divisione dei file in blocks (cioè *128MB* di sizedefault) distribuiti lungo i nodi del centro di calcolo (cioè *cluster*). Successivamente, per processare i dati, HDFS procede con il trasferimento del codice nei nodi per poter processare in parallelo e con tempi ridotti (cioè *Streaming Data Access*), sulla base dei dati che ogni nodo avrà il compito di elaborare. L'architettura base di Hadoop, si compone dei seguenti elementi:

- HDFS.

- Hadoop Common.
- MapReduce.
- YARN.

HDFS, come riportato nella documentazione ufficiale e già accennato prima, è il filesystem distribuito di Hadoop, progettato appositamente per essere eseguito su commodity hardware. Quando la mole di dati va oltre i canoni classici di stoccaggio di una singola macchina, diventa appunto necessario partizionare gli stessi su un certo numero di macchine separate e interconnesse da una rete, rendendo il filesystem distribuito più complesso rispetto ai tradizionali.

Hadoop Common rappresenta il livello software comune a tutto lo stack che fornisce le funzioni di supporto agli altri moduli.

MapReduce è l'implementazione dell'omonimo paradigma di programmazione. Operando secondo il principio “divide et impera” riesce a suddividere un problema complesso, assieme ai relativi dati, in piccole parti processate in modo autonomo procedendo, una volta concluse tutte le sotto-elaborazioni, al merge dei risultati parziali producendo un unico risultato finale.

YARN (cioè *Yet Another Resource Negotiator*) si occupa della schedulazione dei task. Questo viene fatto secondo il principio della *data locality*, ossia ogni nuova attività è assegnata al nodo più vicino al fine di minimizzare il costo di spostamento dei dati. I task sono intesi come sottoattività che compongono le procedure *map* e *reduce* eseguite nel processo di calcolo.

- *Map*: Il nodo master prende i dati di ingresso, li suddivide in piccoli sotto problemi, e distribuisce il lavoro ai nodi slaves. Il singolo nodo mapper produce il risultato intermedio della funzione di `map()` sotto forma di coppie *[chiave, valore]* memorizzate su un file distribuito, la cui locazione è notificata al master alla fine della fase di *map*.
- *Reduce*: Il nodo master colleziona le risposte, combina le coppie *[chiave, valore]* in liste di valori che condividono la stessa chiave e li ordina per chiave.

Si evince quindi che HDFS, YARN e MapReduce rappresentano il cuore del framework Hadoop. Affinché la computazione possa essere portata a termine, i due componenti devono collaborare fra loro.

E' presentata di seguito una panoramica interna relativa ai nodi, elementi cardine dell'intera architettura, e alle procedure (figura 1.3). In un cluster Hadoop, come è possibile

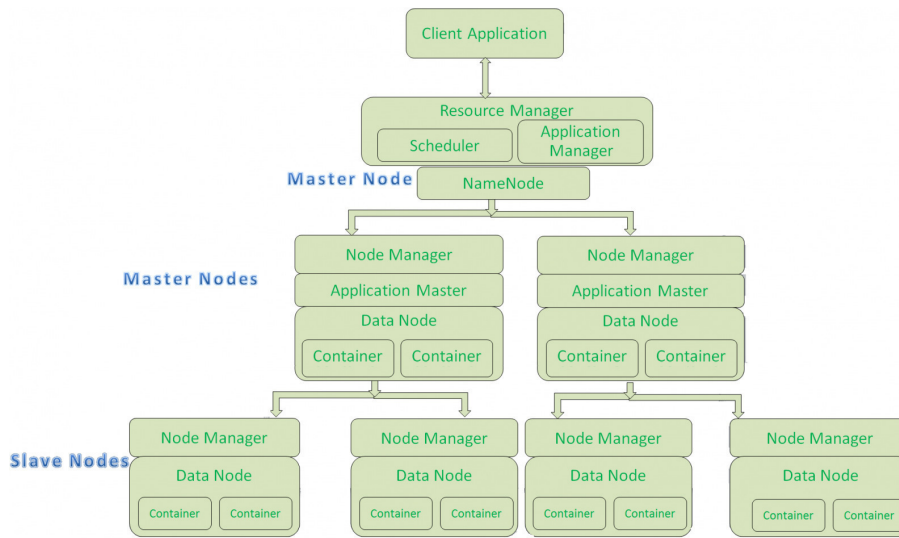


Figura 1.3: Design e i vari tipi di nodo della piattaforma Hadoop.

notare dalla figura, non tutti i nodi sono uguali, ma si dividono in due tipologie [5]. Il nodo master, che esegue i processi di coordinamento di HDFS e MapReduce e, analogamente, il nodo slave, utilizzato per la memorizzazione dei risultati e per l'esecuzione delle operazioni di calcolo. Schematicamente la struttura è organizzata secondo i seguenti elementi per quanto riguarda YARN:

- Un Resource Manager (RM) globale (uno per cluster) È l'autorità ultima che arbitrerà le risorse tra tutte le applicazioni. Questo è a sua volta composto da due elementi. Lo Scheduler, responsabile dell'allocazione delle risorse alle varie applicazioni in esecuzione, in base ai requisiti del sistema e l'Applications Manager, responsabile dell'accettazione degli invii di lavoro, negoziare il primo contenitore per eseguire l'AMP e fornire il servizio per il riavvio di quest'ultimo in caso di errore.

- Per ogni nodo slave, un Node Manager (NM). È responsabile per i *container*; monitora il loro utilizzo delle risorse (quindi CPU, memoria, disco, rete, ecc.) e li segnala all'RM. Un contenitore è un'entità astratta, utilizzata per eseguire un processo specifico dell'applicazione con un insieme vincolato di risorse.

Su richiesta di un client, il RM trova un NM che può lanciare l'*Application master process* (AMP) in un contenitore. L'AMP può quindi richiedere più contenitori per eseguire un calcolo distribuito; la richiesta di allocazione/disallocazione può avvenire in modo dinamico. Per quanto invece riguarda HDFS:

- Il *NameNode* (master-side) mantiene la struttura della directory di tutti i file presenti nel file system e tiene traccia della macchina in cui essi sono memorizzati. È importante inoltre tenere in considerazione che questo memorizza solo le strutture e non i dati veri e propri. Le applicazioni client possono dialogare con il NameNode quando vogliono informazioni per individuare un file, o quando vogliono aggiungere/-copiare /spostare/cancellare un file. Il NameNode risponde alle richieste restituendo un elenco di server DataNode dove si trovano memorizzati i dati richiesti. Quando il NameNode cessa la sua esecuzione ne risente l'intero file system. Per ovviare a questo inconveniente è presente un SecondaryNameNode opzionale che può essere ospitato su una macchina separata, il quale tuttavia non fornisce alcuna ridondanza reale.
- Il *DataNode* (slave-side): memorizza i dati nel file system che per essere funzionale ha solitamente più DataNode che contengono dati replicati. Le applicazioni client possono comunicare direttamente con un DataNode, una volta che il NameNode ha fornito la posizione dei dati. Allo stesso modo, le operazioni di MapReduce affidate alle istanze TaskTracker nei pressi di un DataNode, possono parlare direttamente ai DataNode per accedere ai file. Istanze TaskTracker possono, anzi devono, essere "schierate" sugli stessi server delle istanze DataNode, in modo che le operazioni di MapReduce vengano eseguite vicino ai dati. Le istanze DataNode possono comunicare tra loro, specialmente quando si tratta di dati replicati.

1.2 Analisi di Data Stream

L'esigenza sempre più grande di lavorare in tempo reale ha dato adito al fenomeno Big Data di assumere una nuova accezione: l'analisi e la gestione di data stream. Il data stream processing rappresenta quella particolare pratica che si occupa di elaborare flussi continui di dati, raccolti da sorgenti varie (per esempio sensori, web, social network, log manager) geograficamente distribuite, prodotti con data rate non predicibili, al fine di ottenere risposte tempestive [6]. I contesti applicativi di questo tipo di elaborazione sono molto variegati, spaziano dal mercato finanziario al rilevamenti bancari di frodi passando per cyber attacchi a reti informatiche. La continuità che i flussi di dati assumono lascia intendere che in intervalli di tempo notevolmente ridotti la quantità di dati da analizzare cresca esponenzialmente. Una quantità di dati così vasta, routine se si parla di Big Data, aggiunta ad una velocità di trasmissione e trasferimento altrettanto alta, rende macchinosa e spesso inadempibile la memorizzazione completa dei dati, tantomeno se le tecniche di storage risultano ricadere nei tradizionali DBMS. Una soluzione relazionale classica permetterebbe di processare dati solo su richiesta, producendo significative latenze e inevitabile perdita di frazioni di dataset, impensabile in alcuni contesti. Occorre, perciò, far ricorso a nuove tecnologie e strumenti consoni alle esigenze nell'immediato specificate. A tal merito, una prima possibile alternativa è presentata dall'evoluzione "customizzata" dei classici DBMS: il DSMS (cioè *data stream management system*).

Questa particolare soluzione fornisce una interessante possibilità per gestire data stream in quanto mette a disposizione:

- Esecuzione continua delle query (cioè *continuous query*, o anche *CQ*).
- Interrogazioni con sintassi SQL-like.
- Elaborazione di flussi senza ricorrere alla memorizzazione del dato.

Di seguito, si elencano due ulteriori soluzioni esistenti che rendono possibile l'analisi di flussi secondo le regole prima definite:

DSP (cioè *data stream processing*): è il modello che deriva dalla generalizzazione del DSMS quindi che opera su dati non persistenti. Permette l'elaborazione di stream di dati provenienti da sorgenti differenti, producendo, a sua volta, nuovi stream in uscita dove è possibile riapplicare il procedimento al fine di stratificare concettualmente l'analisi.

CEP (cioè *complex event processing*): si sviluppa in parallelo al DSMS, come evoluzione del modello publish-subscribe procede con l'elaborazione di notifiche di eventi e non dati generici provenienti da sorgenti diverse avvantaggiando l'identificazione di pattern di eventi (o eventi complessi) di interesse.

Il progetto di tesi è basato su Apache Spark (vedi capitolo 2) che comprende un componente per l'analisi dati in streaming, successivamente descritto nel dettaglio. Gli aspetti caratterizzanti un sistema di data stream processing sono diversi e sono presentati in figura 1.4. Lo schema ha come primo elemento la definizione delle applicazioni che può avvenire

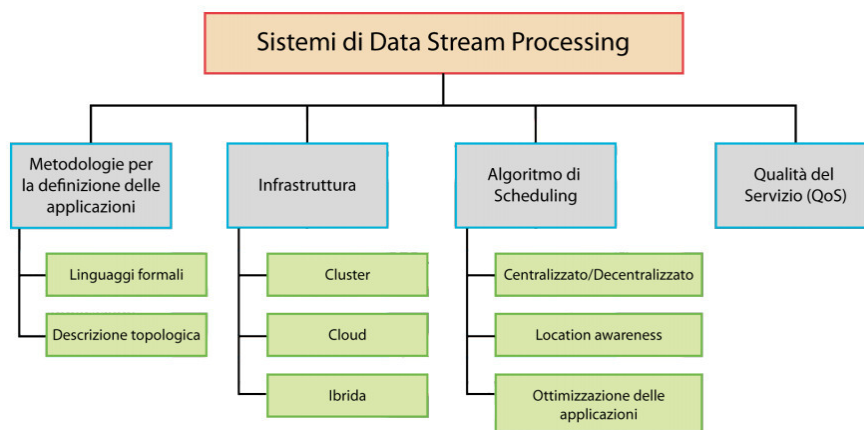


Figura 1.4: Aspetti cardine di un sistema DSP.

mediante due diverse metodologie:

- I *linguaggi formali*, dotati di maggiore rigosità ma alta espressività possono essere dichiarativi o imperativi, e specificano la composizione degli operatori primitivi o il risultato vero e proprio (SQL-like).
- La *descrizione topologica* permette maggiore flessibilità in quanto è possibile procedere ad una definizione esplicita dei componenti che compongono il sistema e delle

interconnessioni per mezzo di un grafo diretto aciclico, chiamato, appunto, *topologia*. Una topologia è composta da nodi (componenti che processano i dati) e archi (data stream trasmessi da un nodo all'altro).

Un altro aspetto molto importante da considerare è l'infrastruttura. Dove, per l'appunto, risiedono le risorse computazionali.

Le scelte possibili sono tre:

- Far uso di un cluster dedicato con un numero di nodi omogenei, “vicini” ed in numero staticamente definito.
- Dipendere totalmente dai recenti servizi cloud che permettono l'allocazione dinamica con conseguente migliore assorbimento delle fluttuazioni versatili nell'arrivo dei dati. Avendo inoltre, nodi geograficamente distribuiti diviene nuovo interesse per il DSP, ma anche scatenatore di nuove problematiche riguardo latenza tra i nodi, scala etc.
- Mediare tra le due precedenti con delle soluzioni ibride considerando un insieme statico di nodi, estendibili con risorse on-demand nel cloud pur valutando bilanciamento del carico, overhead e costi generali.

Ad ogni modo, la prima soluzione è la scelta tradizionale perché più comoda, accessibile e soprattutto facilmente gestibile in termini di organizzazione delle risorse di calcolo. Per il progetto di tesi si è optato per la medesima soluzione, vista la disponibilità di un cluster da parte del gruppo di ricerca, più avanti descritto.

A seguire, ogni sistema DSP, possiede uno *Scheduler*, componente fondamentale responsabile dell'assegnamento dei task delle applicazioni da eseguire alle risorse computazionali a disposizione. La sua efficienza è un aspetto critico che influenza fortemente le performance del sistema e delle applicazioni eseguite. Un algoritmo di scheduling predefinito è sviluppato in relazione a diversi fattori ben individuati:

- La scelta ponderata tra un algoritmo centralizzato o algoritmo distribuito che portano ad un'obbligatoria conoscenza intera rete valutando eventuali problemi di scalabilità.

- La valutazione delle metriche da ottimizzare circa latenza, utilizzo della rete, importanza degli operatori e risorse.
- La capacità adattativa che una particolare architettura può offrire.
- Ove necessario, la capacità di ottimizzare il grafo applicativo generalmente con definizione di applicazioni con linguaggi formali (per esempio, merging, splitting e load shedding)

Terminando, un buon sistema DSP garantisce una elevata Quality of Service (*QoS*) passando dalla gestione ottimizzata degli stream ad un elevato tasso di fault-tolerance nelle applicazioni. Per processare grossi flussi di dati, inoltre, è utile aumentare il grado di parallelismo con tecniche di:

- Task parallelism: stesso dato utilizzato da diversi nodi in parallelo.
- Data parallelism: diversi dati appartenenti allo stesso flusso sono processati da diverse repliche dello stesso nodo.

1.3 Cybersecurity e Intrusion Detection System su piattaforma Big Data

Le problematiche di sicurezza delle reti e le news correlate a un attacco informatico avvenuto su larga scala sono all'ordine del giorno. Si stima che gli attacchi informatici abbiano di gran lunga superato le minacce di attacco terroristico negli Stati Uniti e anche in Europa la situazione non differisce particolarmente [7].

In questo contesto, è assolutamente necessario essere a conoscenza dei vari possibili scenari che possono delineare la presenza di un cyber attacco.

L'analisi delle minacce informatiche, infatti, può essere sensibilmente migliorata correlando gli eventi di sicurezza di numerose fonti eterogenee [8].

Software anti-virus e firewall attualmente non sono sufficientemente affidabili per fornire, da soli, una protezione completa per i sistemi informatici più articolati. Nasce, infatti, la

necessita di implementare sistemi di rilevamento e prevenzione dalle intrusioni nelle reti private (cioè *IDS*, *Intrusion Detection System*) al fine di proteggere le informazioni sensibili contro vari tipi di attacchi [4]. I sistemi di rilevamento delle intrusioni possono essere classificati in tre tipi:

- Sistemi di rilevamento delle intrusioni basato sulla rete (cioè *NIDS*).
- Sistemi di rilevamento delle intrusioni basato su host (cioè *HIDS*).
- Sistemi di rilevamento delle intrusioni basato su ibridi (cioè *HybridIDS*).

La differenza sostanziale sta nel fatto che un *HIDS* rileva attività dannose su un singolo computer mentre un *NIDS* identifica le intrusioni monitorando più host contemporaneamente ed esaminando il traffico di rete. In un *NIDS*, i *sensori* sono situati nei punti di strozzatura della rete per eseguire il monitoraggio, spesso nella zona demilitarizzata (cioè *DMZ*) o sui confini della rete al fine di catturarne tutto il traffico.

In quelli ibridi, invece, le possibili intrusioni avvengono, oltre ad analizzare il traffico di rete, analizzando i registri delle applicazioni, le chiamate di sistema, eventuali modifiche non autorizzate al file system (per esempio file delle password, binari, accesso a database di informazioni sensibili, etc.).

Gli *IDS* sono spesso usati in concomitanza con altri componenti (per esempio router e firewall). Un sistema di prevenzione delle intrusioni, in particolare, può essere visto come una combinazione di software antivirus, firewall locali e *IDS*, il cui obiettivo non è solo quello di rilevare gli attacchi, ma anche di fermarli rispondendo automaticamente. Il "contrattacco", a tal proposito, può avvenire con procedure di difesa come disabilitare le connessioni, segnalare gli alert al sysadmin, terminare processi o arrestare il sistema istantaneamente in casi più gravi, ecc. Analogamente anche i sistemi di prevenzione possono essere classificati allo stesso modo in due distinte categorie:

- Sistemi di prevenzione dalle intrusioni network-based.
- Sistemi di prevenzione dalle intrusioni host-based.

In questo progetto di tesi, in particolare, partendo da un studio dello stato dell'arte dei metodi e delle tecniche di rilevamento delle intrusioni classiche viene fornita una valutazione rispetto ad una integrazione di questo contesto con un panorama orientato alle piattaforme Big Data analizzandone nuove tecniche analitiche.

1.3.1 Metodi e criteri di valutazione

Esistono tre tipi di metodi per il rilevamento delle intrusioni:

- Signature-based detection (anche detto misuse detection).
- Anomaly-based detection.
- Hybrid detection.

Il più comunemente usato e più accurato tra questi è il primo. Dopo che un nuovo attacco viene "definito", in base alle risorse a cui punta e altri indicatori di rete specifici, viene delineato il suo modello di attacco (cioè *attack pattern*) e quindi la sua, appunto "firma". Gli specialisti di sicurezza della rete, a questo punto, possono progettare una difesa contro un nuovo assalto riconosciuto proprio in base al ripresentarsi delle stesse condizioni e quindi della stessa firma identificativa. Per la difesa proposta, quindi, l'IDS è aggiornato di conseguenza al fine di rilevare il nuovo modello di attacco e rispondere prontamente. Questo metodo è molto efficace nel rilevare attacchi noti e produce fondamentalmente un numero limitato di falsi positivi (cioè FP), ossia alert che classificano il traffico normale come dannoso. Tuttavia, se il modello di attacco è leggermente modificato (per esempio target spostato su risorse differenti, specifiche di comunicazione diverse), questo metodo non è totalmente in grado di identificare le versioni modificate dell'attacco stesso e può risultare inefficace e facilmente valicabile. [9][12].

Normalmente un IDS anomaly-based, invece, possiede come riferimento un set di modelli di comportamento "safe" nel suo sistema di difesa. Dato un flusso di dati in arrivo, una situazione anomala viene classificata come un attacco quando quest'ultimo presenta delle differenze dal modello comportamentale classico di riferimento. Questa tipologia di rilevamento può essere utilizzata con il sussidio di tecniche di apprendimento non supervisionate

per rilevare nuove tipologie di attacchi senza la necessità, quindi, di pattern etichettati; tuttavia, anche quest'ultimo presenta la possibilità di generare falsi allarmi. In particolare questo approccio presenta una buona precisione nel rilevamento attacchi a livello di rete come *SYN flood*, *teardrop*, e *Denial of Service* (cioè DOS); ma non nel riconoscere anomalie a livello di applicazione come *Remote to Local* (R2L) e *User-to-Root* (U2R, cioè *unauthorized privilege escalation*).

Questo perché vengo considerati soltanto i campi di intestazione del pacchetto come flag, numeri di porta e indirizzi IP, etc; quindi, appunto, funzionano bene solo se un attacco riguarda solo i campi correlati a livello di rete. Sfortunatamente, non hanno modo di rilevare gli attacchi dove l'esperienza utente è direttamente coinvolta. Ad esempio, un attacco che induce gli utenti a scaricare un file di script dannoso poiché non coinvolge campi di intestazione del pacchetto, non attiva alcun allarme. Il metodo di rilevamento intrusioni ibrido è stato sviluppato per migliorare le prestazioni e le capacità di sistemi di rilevamento e prevenzione delle intrusioni (IDPS) combinando il metodo basato sulla firma (cioè signature-based) e il metodo basato sull'anomalia (cioè anomaly-based) [10]. Alcuni criteri di valutazione che possono essere utilizzati per confrontare le prestazioni degli algoritmi in un IDS includono [11]:

- Precisione.
- Tasso di falsi negativi (FNR).
- Tasso di falsi positivo (FPR).
- Tempi impiegati per la rilevazione.
- Consumo di memoria.

Tra i cinque criteri di valutazione, tre sono spesso usati per l'IDS:

- *Accuracy* (cioè precisione): misurazione della percentuale di fallimento, rilevamento corretto e il numero di falsi allarmi generato dall'IDS.
- *FNR* (cioè false negative rate): la percentuale dei campioni che vengono segnalati come normale quando in realtà sono effettivamente situazioni anomale.

- *FPR* (cioè false positive rate): la proporzione di istanze normali che sono erroneamente sono classificate come anomale.

In sintesi, la tabella 2.1 mostra un confronto tra i due approcci evidenziando caratteristiche e svantaggi di entrambi. La tabella 2.2, a sua volta, confronta i tre metodi di rilevazione in base a diversi criteri prestazionali. [13].

	Anomaly Detection	Misuse/Signature Detection
Caratteristiche	Usa la deviazione dal normale utilizzo come modello per identificare le intrusioni.	Utilizza i modelli di attacchi noti (firme) per identificare le intrusioni.
Svantaggi	- Deve studiare l'interrelazione sequenziale tra le transazioni - Falsi positivi	- Gli attacchi noti devono essere codificati a mano - Impossibile rilevare nuovi attacchi (no evoluzione) - È necessario aggiornare le firme - Falsi negativi

Tabella 2.1 Un confronto tra Anomaly Detection e Misuse Detection

Tecniche di detection	Alarm Rate	Velocità	Consumo di risorse	Flessibilità	Affidabilità	Scalabilità	Robustezza
Anomaly	Alto	Basso	Alto	Alto	Moderato	Alto	Alto
Signature	Basso	Alto	Basso	Basso	Alto	Basso	Basso
Ibrido	Moderato	Moderato	Alto	Alto	Alto	Alto	Alto

Tabella 2.2 Un confronto tra i diversi metodi di detection

Sofisticati attacchi hacking sono in costante aumento nel panorama informatico e i bersagli più ambiti risultano essere aziende, agenzie governative, enti pubblici proprio per la quantità di dati sensibili che detengono dei proprio database. Questo tipo di attacco è comunemente chiamato *Advanced Persistent Threat* (cioè APT) [23][24][25].

APT si rivolge a un sistema specifico e analizza le vulnerabilità del sistema a lungo prima di concretizzarsi. Pertanto è difficile prevenirlo e soprattutto rilevarlo rispetto agli attacchi tradizionali. L'attacco APT si svolge generalmente in quattro fasi (figura 1.5[cita il paper realltimewithbig]):

- Intrusione
- Ricerca

- Raccolta, anche detta *leaking*
- Attacco

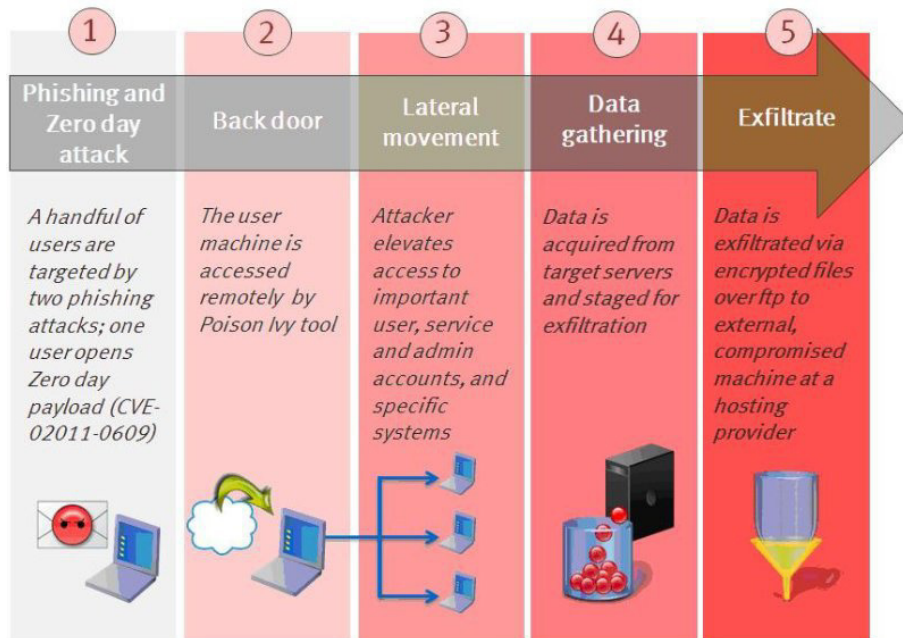


Figura 1.5: Struttura generale di un'intrusione

Nella fase di intrusione, l'hacker cerca informazioni sul sistema di destinazione e prepara il attacco.

Nella fase di ricerca, si analizzano i dati di sistema come il registro di sistema per ottenere informazioni preziose e cercare vulnerabilità di sicurezza che possono essere sfruttate per successive azioni invasive.

Nella fase di raccolta, si installano malware quali *Trojan horse*, trap doors e backdoor per raccogliere dati di sistema e mantenere attivo l'accesso al sistema per il futuro.

Nella fase finale, si distrugge il sistema di destinazione utilizzando le informazioni acquisite e i permessi di accesso raggiunti.

Esistono molte tecnologie utilizzate per prevenire l'intrusione mediante l'utilizzo di un processo di big data management. In generale le tecniche per mantenere la sicurezza informatica all'interno di una rete privata si riducono a due categorie [26]:

- *Firewall*: un firewall è un sistema di sicurezza di rete, basato su hardware o software che, secondo una serie di regole (cioè *default-deny* o *default-allow*), funge da barriera tra reti affidabili e reti non attendibili.
- *Intrusion Detection System*: un IDS ispeziona tutta la rete in entrata e in uscita identificando tentativi di intrusione sospetti notificandoli al sysadmin.

Anche se entrambi si riferiscono alla sicurezza della rete, un IDS differisce da un firewall in quanto oltre a rilevare delle intrusioni cerca di impedirne la concretizzazione. Il firewall invece limita l'accesso tra reti per prevenire intrusioni, possiede una sorta di *blacklist* ma non segnala un attacco dall'interno della rete. Le tecnologie convenzionali spesso non possono supportare analisi a lungo termine e su larga scala principalmente perché producono un grande volume di dati e spesso non è economico, fattibile o utile mantenerli. Perciò, la maggior parte dei registri di eventi e log vengono cancellati dopo un periodo di conservazione fisso che non va oltre il mese. In secondo luogo, è troppo inefficiente per condurre analisi e query complesse su dati non strutturati e grandi set di dati caratterizzati dalla presenza di incompletezze e inconsistenze (cioè *noise*).

1.3.2 L'utilizzo di streaming di dati

In generale, l'analisi e la gestione efficace dei flussi di dati rappresenta una grande sfida dal momento che i dati stessi sono generalmente non memorizzati in nessun tipo di repository o container. Il modello di query continuo è un modello di query tipico in un flusso sistema di gestione di questo tipo di dati dove vengono eseguire le query predefinite periodicamente e costantemente al fine di raccogliere dati aggregati, rispondere alle modifiche (cioè *refreshing*) e segnalare il loro stato attivo. A questo proposito il data mining implica cambiamenti dinamici ed efficienti per la scoperta di modelli generali all'interno degli stream. Nel panorama della Cybersecurity è centro di interesse identificare dinamicamente le intrusioni in base all'anomalia del flusso di messaggi in ingresso per mezzo della ricostruzione di modelli di flusso con tecniche di clustering, o confrontando gli attuali pattern più frequenti con quelli specifici estratti nelle volte precedenti. I principali problemi associati al data mining su streaming di dati sono:

- Evolution
- Drift
- Lunghezza (idealmente) infinita

Il concetto di *evolution*, quindi di evoluzione, è definito dalla possibilità di riscontrare continuamente nuove classi a causa del continuo aggiornamento del dataset.

Il concetto di *drift*, banalmente *tendenza*, si riferisce analogamente alla logica conseguenza che i dati possano cambiare nel tempo. Nel modello di apprendimento (cioè *learning model*) il concetto di drift è introdotto a causa della velocità dei componenti che analizzano grandi stream di dati. Essendo questa altamente variabile, rende il modello predittivo di determinate "target variable" instabile e decisamente nondeterministico. Il tutto porta ad un processo di predizione sempre meno accurato con lo scorrere del tempo.

La lunghezza infinita, invece, implica sicuramente una privazione dell'attività di storage e una conseguente e continua fase di training.

Il concetto di *real-time intrusion detection* quindi *c* si presenta quindi come un task tedioso e poco accurato a causa del grande volume di dati coinvolti e l'instabilità con cui sono prodotti e ricevuti. Lo squilibrio dei dati è infatti uno degli ostacoli principali da tenere in considerazione. Se il livello di squilibrio nei dati è elevato, i classificatori avranno una precisione e un'affidabilità inversamente proporzionale, quindi decisamente inferiore. Lo squilibrio nel carico di lavoro, tuttavia, è un problema inevitabile nei dati in tempo reale. Le tecniche di campionamento, tuttavia, (cioè *sampling*) sono approcci comuni alla riduzione dell'errore nel utilizzo di classificatori [14]. Bisogna considerare che gli attacchi e le intrusioni malevoli sono fenomeni esageratamente dinamici, quindi lavorare in real-time risulta una vera e propria necessità. Un evento potrebbe essere normale se preso in considerazione da solo ma decisamente dannoso se è considerato come parte di un sequenza di eventi (per esempio port scanning). L'analisi dei dati provenienti da un flusso viene utilizzata perché proprio per aiutare a identificare le intrusioni in questo tipo di situazioni. Può essere molto utile per identificare sequenze di eventi che frequentemente si

verificano assieme, scoprendo quindi pattern sequenziali dotati di una certa periodicità.

In generale si distinguono tre principali tipi di anomalie [15]:

- *Point anomalies*: dove solo alcuni data sample sono rilevati come anomalie rispetto all'intero dataset di riferimento.
- *Collective anomalies*: dove collection di data sample sono rilevate come anomalie soltanto se assieme.
- *Contextual (conditional) anomalies*: dove le anomalie sono rilevate solo se appartenenti adun determinato constesto con determinate situazioni che in un altro caso rappresenterebbero la normalità.

Un esempio di framework per lo stream data mining è *Massive on-line analysis* (cioè MOA) che include strumenti per la raccolta e valutazione di algoritmi di apprendimento automatico. Può implementare clustering, classificazione, regressione, mining grafico. Contiene funzioni per un'analisi online e offline per clustering e classificazione e strumenti di valutazione e visualizzazione [16]. I dati provenienti dagli stream, tuttavia, possono essere raccolti da varie fonti ed elaborati in un unico motore di elaborazione in modo che i risultati vengano scritti su un sistema di destinazione. *Flink, Storm e Spark Streaming* sono tre principali piattaforme open-source per l'elaborazione degli stream. Storm è migliore in efficienza e throughput rispetto agli altri, ma Spark Streaming è robusto nei guasti dei nodi (cioè fault tolerance) e fornisce un ripristino affidabile senza eventuali perdite [18]. Un altro problema, inoltre, sono i "falsi allarmi" dovuti a grandi volumi in input. Dato che i dati di input possono contenere milioni di oggetti, una bassa percentuale di falsi allarmi rendere l'analisi complessa. Dati etichettati corrispondenti ad un comportamento normale sono chiaramente spesso disponibili, ma le stesse etichette in un dataset dedicato alle intrusioni spesso non lo sono, come sarà poi spiegato nella fase di analisi del progetto di tesi. Pertanto, *unsupervised* o *semi-supervised* sono spesso i preferiti.

1.3.3 L'introduzione di tecniche di Data Mining e Machine Learning

La selezione delle feature (cioè principal component analysis, *PCA*) è importante nell'apprendimento automatico e nel processo di data mining. Questa può migliorare le prestazioni di previsione dei vari modelli attraverso la riduzione dei dati in termini di dimensione accelerando il processo di apprendimento. In generale questa pratica avviene, quando si parla di analisi in contesti offline, dove il dataset è fornito *a priori*.

Tuttavia, in un contesto real-time sarebbe costoso raccogliere e analizzare i dati man mano. La selezione di feature online (cioè *OFS*) si basa infatti sullo sviluppo di classificatori online che utilizzano solo un numero limitato di feature [23]. Estrazione di big data un importante argomento di ricerca. I big data sono definiti come "set di dati la cui dimensione va notevolmente oltre la capacità degli strumenti software e dbms convenzionali di acquisire, archiviare, gestire e analizzare. Il volume dei big data in molti settori va da poche decine di terabyte a più petabyte [17]. Tecnologie in merito includono infatti attività di machine learning, data mining, crowd sourcing, natural language processing, stream processing, time series analysis, cluster computing, cloud computing, parallel computing, visualization, e perfino graphics processing unit (GPU) computing.[19] Sfortunatamente, gli attributi e i record ridondanti rendono l'operazione di intrusion detection un compito molto impegnativo ma che può sensibilmente essere semplificato con l'utilizzo di tecniche di PCA; molto importante per l'efficienza del processo e aiuta a ridurre la complessità computazionale migliorando le prestazioni dei classificatori.

Tecniche di data mining come clustering, classificazione, e l'estrazione di regole associative vengono spesso utilizzate per ricavare informazioni di possibili intrusioni di rete attraverso l'analisi dei dati relativi alle sessioni di comunicazione. Nello specifico, inoltre, c'è differenza tra le applicazioni di classificazione e clustering nei sistemi IDS [20]:

- *Clustering*: un vantaggio del clustering sul metodo di classificazione è che non ha bisogno di usare dataset etichettati. Questo è utile negli IDS perché l'attività dannosa ha senso che venga raggruppata insieme, separandola logicamente dalla normale attività di rete.

- *Classificazione*: un IDS basato sulla classificazione tenta di classificare tutto il traffico in record malevoli o normali che siano. La sfida sta proprio nel minimizzare il numero di false negative e false positive.

In letteratura le tecniche generalmente più utilizzate per eseguire una classificazione della rete sono support vector machine (cioè *SVM*), alberi decisionali, generazione di regole associalitive, reti neurali e algoritmi genetici. La Tabella 4 [21] descrive e confronta questi metodi.

Nelle applicazioni di data mining applicate sugli eventi generati nella rete di comunicazione gli aspetti principali dell'analisi includono:

- Classificazione dello stato della rete, tipi di applicazioni e tipi di utenti, ecc. al fine di attuare l'azione più idonea per ciascuna situazione.
- Raggruppamento di utenti in gruppi in base a elementi comuni come applicazioni, risorse hardware, specifiche software, posizione, ruolo.
- Sviluppo di modelli di regressione per prevedere il carico del traffico, qualità del canale di comunicazione, il numero di richieste di assistenza, eventuali danni e la qualità del servizio [22].

1.3.4 Il ruolo dei Big data nella Network Intrusion Detection

A questo proposito i sistemi di elaborazione di Big Data stanno diventando parte della gestione della sicurezza perché aiutano a preparare, pulire e interrogare (processo di *ETL*) dati eterogenei anche se composti da record incompleti e/o "rumorosi" [27]. Sostanzialmente il fenomeno dei Big Data per il rilevamento delle intrusioni illecite nelle reti risulta calzante alle necessità e ai requisiti di sicurezza in quanto promuove l'utilizzo di grandi volumi di dati complessi e con vari formati provenienti da fonti eterogenee. Possono essere creati, infatti, modelli ultra-dimensionali per profilare i dati dello stream in modo accurato e, soprattutto, online.

Tecnologie Big Data come l'ecosistema Apache Hadoop e, per lo stream, Apache Spark Streaming, permettono l'elaborazione e l'archiviazione di dataset grandi ed eterogenei

ad alta velocità. Grazie a queste tecnologie è possibile ottenere una visione generale di una rete informatica includendo nell'analisi più livelli di astrazione (cioè *netflow*, *firewall*, *IDS* e *file di registro*, *host*, *rules*). Inoltre, a questo si affianca un approccio globale per sviluppare strumenti, tecniche e infrastrutture che si adattano alla necessità di ottenere un'interrogazione continua su dati in streaming comprendendo ambiti di analisi statistica, ragionamento deduttivo (*inferenza*), ragionamento induttivo (*machine learning*) e calcolo ad altre prestazioni (*parallelizzazione* e *cluster computing*). Inoltre la previsione e il rilevamento delle intrusioni di rete sono attività sensibili al fattore tempo e quindi necessitano di tecnologie Big Data altamente efficienti e con una *fault tolerance* elevata (cioè *Apache Spark*). Una minaccia APT, precedentemente descritta, è un attacco mirato contro un bene di alto valore o un sistema fisico e rappresenta uno dei più gravi problemi di sicurezza informatica.

L'uso dei Big Data è sicuramente un approccio appropriato all'identificazione dell'APT [28] poiché permette:

- Il rilevamento dell'anomalia basato sulla correlazione di eventi storici e recenti. Ad esempio, un aumento volume del traffico DNS (cioè *Domain Name System*) in breve tempo può essere dovuto a legittimi comportamenti degli utenti. Ma un tale alert indica un tentativo di data exfiltration se viene rilevata anche nel traffico storico per un periodo di giorni. Inoltre, questo tipo di correlazione, poiché eseguita su una casistica molto ampia, aiuta a ridurre il tasso di falsi positivi (FPR).
- L'acquisizione e l'integrazione dinamica da fonti di dati eterogenee come traffico di rete, eventi (per esempio IDS, dispositivi di rete, logs) che aiutano il sistema a correlare eventi sporadici di bassa gravità come diretta conseguenza di una intrusione latente.

Rispetto ai sistemi tradizionali (cioè *SIEM*[29]), l'analisi dei Big Data non ha un finestra a tempo limitato ma può evolversi nel tempo in maniera performante senza rallentamenti nonostante l'alto carico di lavoro.

E' riportato di seguito un modello introdotto dalla *SSM Arts Science, Periyar University*

per la possibile realizzazione di un sistema di Intrusion Detection che fa ricorso sia a un approccio prettamente batching che uno real-time. Il tutto partendo con, in input, dei log di rete provenienti dalle macchine tenute sotto controllo.

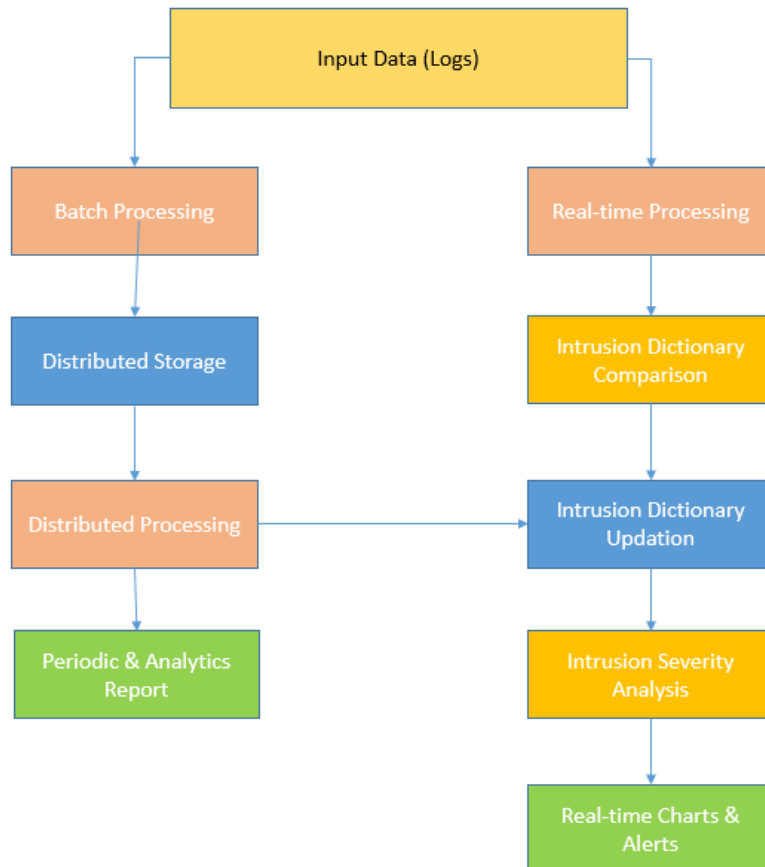


Figura 1.6: Rappresentazione di un Real-Time Intrusion Detection System

In figura 1.6, è possibile notare come il processo di controllo della rete si parallelizzi completamente nelle due componenti batch e realtime.

Seguendo il primo filone, viene effettuato un processing dei dati a livello distribuito che permette l'elaborazione di quest'ultimi con fine ultimo la creazione di report analitici e periodici per l'utente.

Il processo analogo, real-time, permette, interfacciandosi con il precedente, l'aggiornamento continuo del "dizionario" contenente le regole per l'analisi comparativa dei log al fine

di identificarne un modello sempre aggiornato. In questo caso il risultato è la generazione di alert e chart che, tramite una dashboard intuitiva per l'utente, riescono a fornire informazioni in tempo reale sullo stato della rete e le anomalie rilevate.

A livello di tecnologie e architettura, lo schema proposto può essere sviluppato come mostrato in figura 1.7.

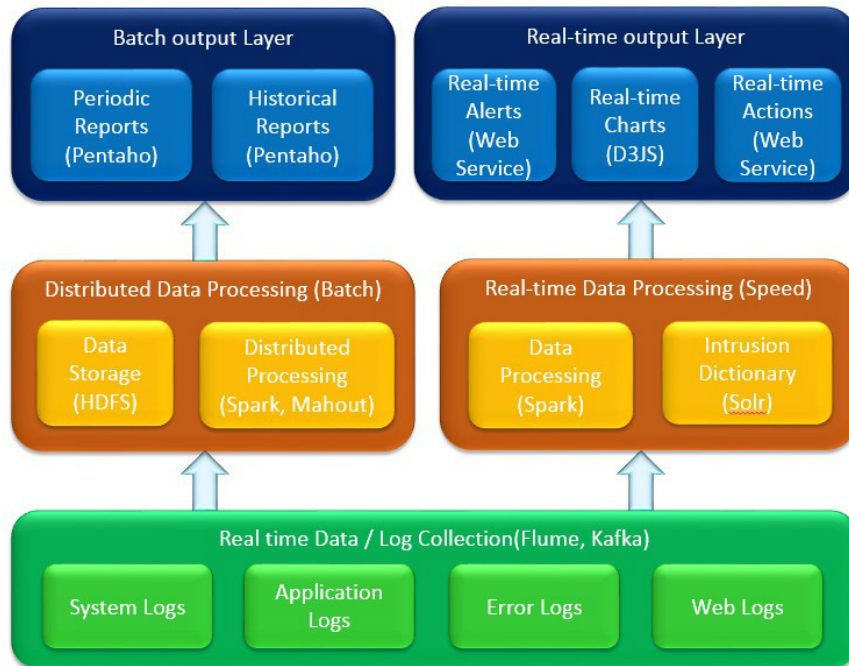


Figura 1.7: Possibile scelta tecnologica per un'architettura di un Real-Time Intrusion Detection System

La parte di log collection, può essere affidata a piattaforme per la gestione di streaming in maniera affidabile, distribuita ed efficiente come *Apache Flume* o *Apache Kafka*[31][32], usato anche per la realizzazione del prototipo di tesi.

Sistemi di questa classe di software permettono il raccoglimento di dati mediante "canali di comunicazione dedicati" favorendo una discriminazione topic-based utile per smistare in maniera bilanciata il traffico dati, molto copioso nella generazione in questo ambito.

Successivamente, la parte di processing può essere commissionata, per il batch, con relativo processo di data storage, ad HDFS in combinazione con il più famoso framework

distribuito per big data Apache Spark, successivamente presentato nel dettaglio (vedi Capitolo 2). Analogamente, nel componente realtime la combinazione può prevedere, data la non necessità di storage, nuovamente Apache Spark in concomitanza con Apache Solr, utile per l'indexing delle regole di sicurezza estratte e continuamente aggiornate.

Per la presentazione dei contenuti user-readable la scelta delle tecnologie è abbastanza ampia e spazia dall'utilizzo di software per la BI (cioè Business Intelligence) come Pentaho[30], per la creazione di report dinamici e navigabili, ai tradizionali *web service* per la visualizzazione delle informazioni costantemente aggiornate della rete e degli alert.

Capitolo 2

Apache Spark

Apache Spark™^[33] (*Apache Software Foundation, Maggio 2014*) è un framework di calcolo distribuito scritto in Scala, open source e basato sul paradigma MapReduce, compatibile con l'ecosistema complementare Apache Hadoop (maggiori dettagli nella sezione 1.1).

Il framework si impone nel panorama dei Big data fondamentalmente con due grandi sfide:

- Permettere un'esecuzione più efficiente di applicazioni di data analysis che prevedono deployment su piattaforma cluster di commodity hardware.
- Ottimizzare le performance lato hardware.
- Abilitare l'utilizzo di nuove tipologie di analytics .

In merito al secondo caso, si distinguono due macrocategorie:

- *Algoritmi iterativi*, in cui le stesse routine sono richiamate più volte sullo stesso dataset.
- *Analisi interattive*, in cui, con lo scopo di analizzare un determinato dataset per estrarne informazioni e conoscenza, vengono eseguite più query.

Al fine di poter essere eseguito Apache Spark necessita di due componenti "core":

- Un cluster resource manager.



Figura 2.1: Logo Apache Spark

- Un sistema di storage distribuito.

Come cluster manager Spark supporta sia la modalità standalone quindi affidandosi al proprio cluster manager (cioè standalone scheduler) piuttosto che l'utilizzo di Hadoop YARN (vedi 1.1 o Apache Mesos[35]).

Per quanto riguarda lo storage distribuito, Spark è versatile nella scelta. Questo infatti può essere integrato a framework come *HDFS*, *Amazon S3* o *Apache Cassandra*[36]. Spark implementa inoltre una modalità di esecuzione locale, principalmente per testing e prototipazione affiancata da un processo di memorizzazione dati in grado di utilizzare il file system locale e quindi non distribuita. Il progetto Apache Spark è composto dai seguenti moduli (figura 2.2):

- *Spark Core*: è il modulo principale e fornisce i servizi per l'esecuzione di task, le operazioni di I/O, lo scheduling, il recupero dal fallimento di un nodo del cluster (cioè *disaster recovery*). Il core di Spark fornisce un API-set accessibile tramite i linguaggi di programmazione più diffusi (cioè *Java*, *Python*, *Scala* e *R*, vedi 2.2).
- *Spark SQL*: fornisce supporto per l'elaborazione di dati semi-strutturati basati sul modello relazionale con interrogazioni SQL (cioè *HIVEQL*). Si possono esplorare anche file JSON, file di testo e qualunque formato di file supportato da *Hive*[36], tra cui *Parquet*[37], *CSV* (cioè *comma-separated values*).

Spark SQL si integra perfettamente con il resto del sistema: i risultati delle query

possono essere trasformati e analizzati nell'ambiente Spark e, analogamente, è possibile analizzare dati con un qualsiasi motore SQL dopo avergli associato uno schema tabellare.

- *Spark Streaming*: comprende una serie di funzionalità per l'analisi di stream di dati, ad esempio log di errori, un flusso di tweet, *analisi di reti informatiche*. Anche questo componente si integra perfettamente con il resto. I flussi di stream possono arrivare da fonti come MoM (Apache Kafka), o filesystem distribuiti (HDFS), e vengono ingeriti in piccoli lotti (cioè *micro-batch*) per essere analizzati. Si possono svolgere tutte le operazioni fattibili sui dati statici, ma in più sono presenti funzionalità specifiche relative a necessità temporali, come le *sliding windows*.
- *MLlib*: è il framework utilizzato per il machine learning distribuito altamente ottimizzato, che sfrutta il fatto che i dati di Spark possono essere memorizzati in memoria. Molti degli algoritmi di machine learning sono infatti iterativi. Ovviamente MLlib offre la possibilità di usare solo gli algoritmi che sono per loro natura parallelizzabili (cioè regressione lineare, clustering, alberi decisionali, etc).
- *GraphX*: è il modulo utilizzato per il graph processing distribuito, quindi per l'analisi di grafi di grande ampiezza che non potrebbero essere analizzati su una singola macchina (per esempio *social graph*). GraphX unifica ETL, analisi esplorativa e calcolo iterativo su grafi in un unico sistema. I grafi sono gestiti come gli altri dataset, con cui si possono effettuare anche dei join. Tramite l'API *Pregel*, ad esempio, si possono implementare algoritmi iterativi personalizzati per l'analisi di grafi.

2.1 Architettura

Come già detto, Spark può lavorare sia in un singolo nodo che in cluster. Nel caso generale sono presenti un serie di processi in esecuzione per ogni applicazione Spark. Nello specifico, seguendo come modello architetturale quello del *master/worker*, questo presenta, due tipologie principali di componenti:

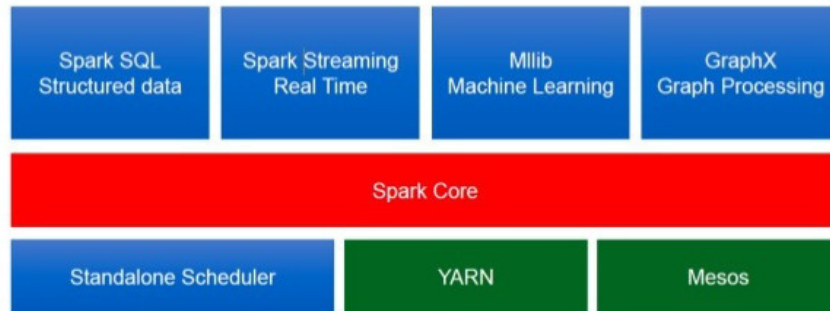


Figura 2.2: Moduli principali Apache Spark

- Un driver program (cioè *master*)
- Uno o più executor (cioè *worker*)
- Un cluster manager

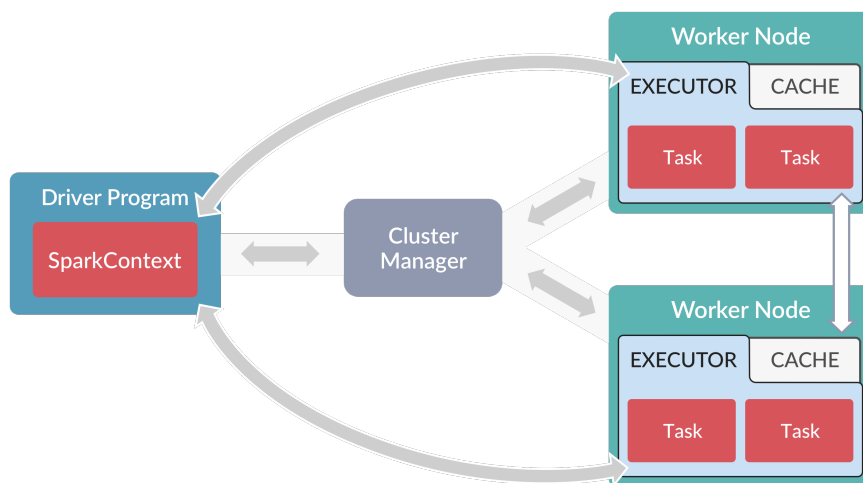


Figura 2.3: Modello architetturale di Spark

Come si evince dalla figura 2.3 il driver program è il processo che esegue la funzione `main()` e che genera lo *SparkContext*. Affiancato a questo ogni executor è un processo lanciato su un nodo worker, in grado di eseguire task, mantenere i dati in memoria (cioè *RAM* o *cache*) o su disco e restituire risultati della computazione al driver. In generale il driver

coordina il lavoro ma non svolge nessuna attività; le computazioni vengono svolte dagli executor, che aggiornano quest'ultimo sul loro stato di avanzamento e comunicano l'eventuale stato finale.

In tutto ciò il cluster manager è un servizio esterno integrato che gestisce e ridistribuisce le risorse permettendo il deployment in modalità cluster. In breve, le applicazioni Spark diventano un insieme di processi indipendenti sul cluster, coordinati dallo *SparkContext*. Prima dell'esecuzione, quest'ultimo si connette al cluster manager (per esempio *YARN*), per richiedere l'allocazione delle risorse necessarie all'applicazione per eseguire la computazione. Una volta connesso, il driver identifica gli executor disponibili sui nodi e manda a ognuno di questi sia il codice dell'applicazione che i task da eseguire. Ogni executor eseguirà poi le operazioni necessarie sulla partizione di dati a propria disposizione estratta solitamente da una file system distribuito, (*HDFS*). Segue una descrizione più accurata di ogni main component appartenente alla struttura logica di esecuzione.

2.1.1 Driver program

Il driver è il processo principale, quello in cui è presente il metodo main contenente il codice utente. Quest'ultimo, che contiene operazioni sugli *RDD* (cioè dataset distribuiti, vedi 2.3), viene eseguito in parallelo dai processi executor distribuiti nel cluster. Il driver può essere eseguito sia all'interno del cluster che nella macchina local che manda in esecuzione l'applicazione. Le seguenti due funzioni sono svolte:

- *Suddividere il programma utente in un insieme di task*, cioè la più piccola unità di lavoro in Spark. In ogni programma Spark si leggono dati da disco in uno o più RDD, si "trasformano" e si recupera il risultato della computazione. Le operazioni di trasformazione vengono effettuate solo quando si richiede un risultato.
- *Effettuare lo scheduling dei task sui nodi executor*. Lo scheduling dei task viene fatto in base a dove sono memorizzati i file, per evitare il più possibile di trasferirli in rete. Se un nodo fallisce, lo scheduling in un altro nodo viene effettuato automaticamente dalla piattaforma, ricalcolando logicamente solo i dati persi fino a quel momento.

2.1.2 Executor

Gli executor, o esecutori, sono i processi che svolgono i compiti (cioè *task*) sotto direttiva del driver. Ogni applicazione ha i propri executor, cioè i propri processi, ognuno dei quali può avere più *thread* in esecuzione. Gli executor hanno una certa quantità di memoria assegnata configurabile che permette di memorizzare i dataset in memoria se richiesto dall'applicazione utente. Gli esecutori sono attivi per tutto il tempo di esecuzione di un'applicazione; se un esecutore fallisce lo SparkContext riesce comunque a continuare l'esecuzione del programma ricalcolando solamente i dati persi in quel determinato task.

2.1.3 Cluster manager

I cluster manager si occupano di gestire le risorse all'interno di un cluster. Ad esempio quando più applicazioni richiedono risorse della rete di nodi, il cluster manager effettua lo scheduling nei nodi in base alla memoria e ai core della CPU liberi in quel momento mediante lo snapshot computazionale corrente. Alcuni cluster manager permettono anche di dare priorità diverse a diverse applicazioni. Spark supporta i seguenti cluster manager:

- YARN, il resource manager proveniente dall'ecosistema Hadoop
- Mesos
- Standalone cluster manager
- Deployment su cluster regolato da servizi *Amazon EC2*

2.2 Linguaggi supportati

Spark offre anche molte opzioni per il linguaggio di programmazione: Scala, Java, Python e R. Lo Spark Survey del 2017[38] (figura 2.4) che ha sondato la comunità Spark mostra una crescita particolarmente rapida su Python e R. In particolare, il 58% degli intervistati utilizza Python (un aumento del 49% rispetto a 2014) e il 18% ricade su R. Addirittura il 71% ritiene Scala il più idoneo visto l'approccio prettamente funzionale che un'applicazione Spark richiede.

In generale Scala, rispetto a Python, risulta 10 volte più veloce e, sebbene la verbosità e la learning curve (cioè curva di apprendimento) decisamente più a sfavore dell'utente, supporta funzioni avanzate di concorrenza e typing che ottimizzano decisamente la qualità del codice prodotto. Per questo motivo il progetto di tesi (*Styx*, vedi capitolo4) è stato integralmente realizzato con l'ausilio del medesimo linguaggio.

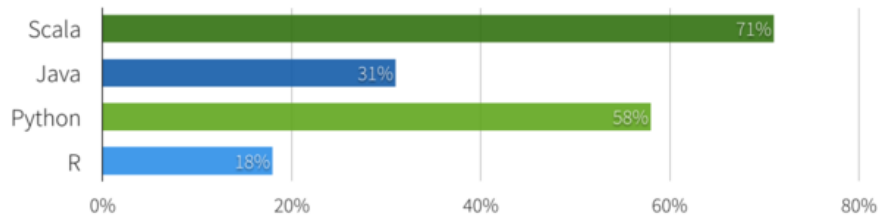


Figura 2.4: Principali linguaggi usati in Spark

2.3 Astrazioni fondamentali: RDD e DAG

Un RDD (cioè *Resilient Distributed Dataset*)[7] si definisce come una collezione fault-tolerant e *non modificabile* di elementi sulla quale si può operare in parallelo memorizzata in maniera distribuita. La struttura dati è divisa logicamente in parti chiamate *partizioni*, ognuna delle quali è presa in carico per un determinato job da un executor di Spark. Per processare i dati, il sistema creerà, infatti, un task per ogni partizione (in HDFS questa coincide con un blocco). Gli elementi di un RDD non devono necessariamente esistere nella memoria fisica e possono essere ricostruiti in caso di fault di un nodo, assicurando così tolleranza ai guasti(cioè fault-tolerance).

Il meccanismo è molto ingegnoso: si memorizza un grafo aciclico diretto (cioè *DAG*) (figura 2.5) delle operazioni da fare per ottenere, e riottenere, il contenuto di un RDD. Le operazioni di trasformazione o di salvataggio/recupero di dati vengono trasformate in una serie di stage eseguiti in sequenza, ognuno dei quali è composto da un insieme di task che vengono eseguiti dagli executor.

Spark permette di creare un RDD in tre differenti modi.

- Partendo da un file, salvato su HDFS o in locale in qualche nodo.

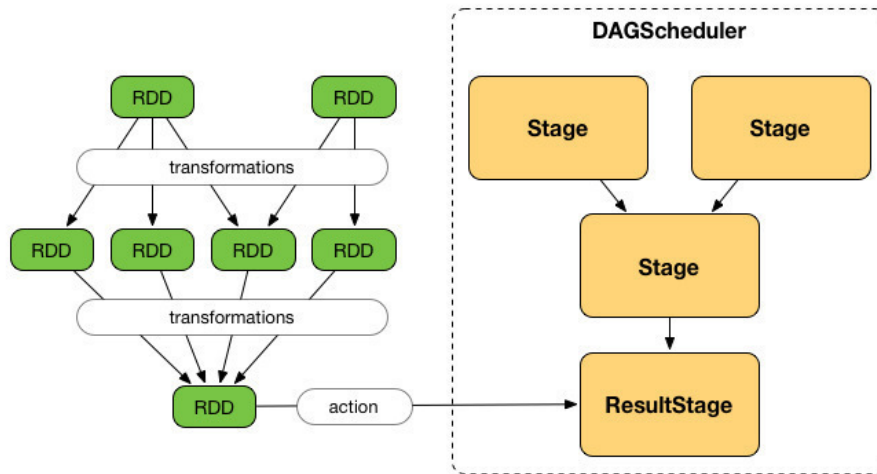


Figura 2.5: Grafo degli stage

- Parallelizzando l'accesso ad una collezione dati memorizzati in quel momento nello heap (cioè `array`, `list`, `sequence`), distribuendola cioè tra più nodi del cluster.
- Trasformando un RDD esistente, cioè applicando funzioni di trasformazione sugli oggetti dell'RDD stesso. Il risultato sarà, naturalmente, un nuovo RDD.

Operazioni sugli RDD

Sull'astrazione di storage appartenente all'ambiente Spark possono essere eseguite vari tipi di operazioni ad alto livello[40]:

- Trasformazione.
- Azione.

Per ogni RDD, viene effettuato un mapping logico-fisico tra partizioni e task ogni qualvolta venga sottomesso un job. Questo avviene tramite gli *stage* (figura 2.6). Uno stage è un'unità fisica di esecuzione. È un passo atomico in un piano di esecuzione fisica composto da un insieme di attività parallele: ogni attività viene assegnata univocamente ad una partizione di un RDD che calcola i risultati parziali di una funzione eseguita come parte di un lavoro delegato a Spark. Come è ben comprensibile dalla figura 2.7 le operazioni di

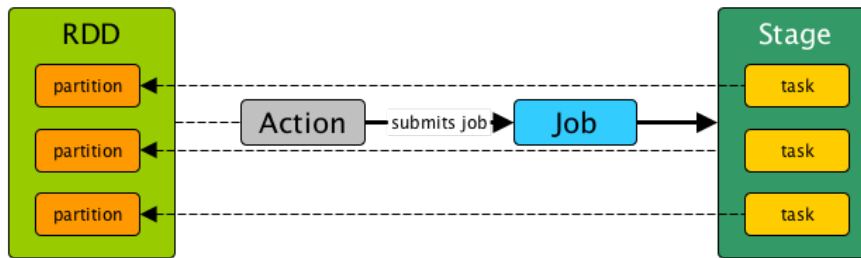


Figura 2.6: Stage, task e il processo di submitting di un job in Apache Spark

transformation creano un nuovo dataset a partire da quello esistente mentre le operazioni di action ritornano un risultato solo dopo che termina la computazione sull'insieme dei dati. Tutte le operazioni di trasformazione sono effettuate con approccio di *lazy evaluation*, quindi vengono eseguite solo nel momento in cui un'operazione di azione richiede un risultato; questo approccio rende il sistema decisamente più efficiente. Normalmente ogni RDD trasformato viene ricalcolato ogni volta che viene applicata un'azione. Onde evitare l'esecuzione ripetuta delle stesse operazioni è possibile salvare in memoria l'RDD trasformato, rendendo così il processo di azione più performante. Inoltre, gli RDD sono, come anticipato, resistenti ai guasti poiché si tracciano le proprie *data lineage information*[41] per ricostruire automaticamente i dati persi in caso di errore.

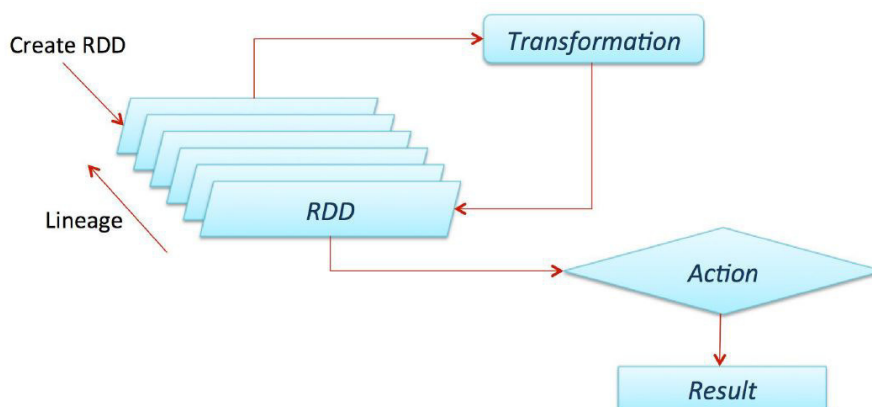


Figura 2.7: Ciclo di vita di un RDD in Spark

Le operazioni di trasformazione possono essere di due tipi:

- *Narrow*: l’RDD in output ha partizioni originate da una singola partizione dell’RDD da cui deriva (per esempio `map`, `filter`, `distinct`).
- *Wide*: quando i dati necessari per calcolare una singola partizione in un RDD possono essere sparsi in più partizioni dell’RDD padre. È il caso di trasformazioni come `groupByKey` o `reduceByKey` o anche `mapValues`. In questo caso Spark dovrà effettuare un’operazione di *shuffling*, trasferendo dati attraverso il cluster.

Esempi di azioni, invece, possono essere `reduce`, `collect`, `take`. Quindi quei processi che producono valori che tornano al programma driver di Spark o che vengono memorizzati su file.

Persistenza degli RDD

Una delle funzionalità più importanti di Spark è la memorizzazione di un insieme di dati in memoria. Normalmente ogni RDD viene ricalcolato ogni volta che è applicata su di esso un’azione. Persistendo un RDD, ogni nodo memorizza le partizioni del dataset a cui è associato per riutilizzarle nel momento in cui vengono applicate altre azioni sul medesimo dataset. Ciò consente a Spark di essere molto più efficiente e veloce nel caso di algoritmi iterativi o di analisi interattive. Ogni RDD può essere salvato utilizzando diverse modalità di memorizzazione: i dati possono ad esempio essere salvati in memoria o su disco.

2.4 Spark SQL

Spark SQL è il modulo di Spark per l’elaborazione di dati strutturati. Diversamente dalle API fornite da Spark RDD, l’interfaccia di Spark SQL è in grado di eseguire ulteriori ottimizzazioni grazie al maggior numero di informazioni in possesso sui dati e sulla computazione da eseguire. Sia Spark Core che Spark SQL utilizzano comunque lo stesso motore computazionale, dunque lo sviluppatore è libero di usare differenti API, anche nello stesso programma. Spark SQL permette di esprimere interrogazioni in linguaggio SQL (cioè HQL, un dialetto custom di Hive) e di interfacciarsi con Hive per il management in forma tabellare. Elementi fondamentali in Spark SQL sono *Dataset* e *Dataframe*.

2.4.1 Dataset

Un Dataset è una collezione distribuita di dati. L'interfaccia Dataset è stata introdotta a partire da Spark 1.6 e mantiene tutti i benefici degli RDD (cioè strong typing, functional programming), a cui si aggiungono le ottimizzazioni di Spark SQL. Un Dataset può essere creato a partire oggetti JVM e poi manipolato attraverso funzioni di trasformazione come ad esempio `map` o `filter`.

2.4.2 Dataframe

Un DataFrame è un dataset, ma organizzato in colonne; ognuna delle quali è associata ad una label. È concettualmente equivalente a una tabella di un database relazionale e può essere creato a partire da file dati strutturati (per esempio file csv,JSON, etc), tabelle Hive, database esterni o RDD stessi. L'interfaccia Dataframe è disponibile per Scala, Java, Python e R. A differenza degli RDD, non fornisce le funzionalità di type-checking a tempo di compilazione infatti nella conversione opposta, quindi da RDD a Dataframe, è necessario definire un `TypeSchema` per permettere la definizione delle tabelle.

2.5 Spark Streaming

Spark sta guadagnando una grande popolarità perché risolve molti dei limiti intrinseci di Hadoop in quanto estende il modello MapReduce per renderlo più veloce e abilitare più scenari di analisi, come le query interattive e, in particolare, *l'elaborazione continua degli streaming di dati in tempo reale con Spark streaming*.

Spark Streaming[42] può importare dati da un'ampia varietà di fonti, compresi stream provenienti da Apache Kafka, Apache Flume, Amazon Kinesis o anche semplicemente Twitter, nonché da sensori e dispositivi collegati tramite socket TCP. È anche possibile elaborare i dati memorizzati in sistemi di file come HDFS o Amazon S3. Spark Streaming può elaborare i dati utilizzando una varietà di algoritmi e funzioni come *map*, *reduce*, *join* e *sliding window*. Una volta elaborati, i dati vengono inviati ai file nei file system o mostrati in apposite dashboard per la visualizzazione in tempo reale. In linea generale, ciò che Spark

Streaming fa è prendere un flusso continuo di dati e convertirlo in uno stream, chiamato DStream, costituito da RDD, quindi batch, che contengono dati solo appartenenti ad un determinato arco di tempo (figura 2.8). Internamente, ciò che accade è che Spark Streaming memorizza ed elabora questi dati come una sequenza di RDD (Resilient Distributed Data). Per questo motivo, in Spark Streaming, un flusso nello Spark Core è analizzato in modo normale, senza sapere che si sta effettivamente elaborando un flusso di dati, perché il lavoro di creazione e coordinamento degli RDD partizionati è delegato esclusivamente a Spark Streaming (figura 2.9)



Figura 2.8: Creazione di un DStream in Apache Spark



Figura 2.9: Gestione *batch-oriented* di uno stream di dati prima di essere elaborato dallo Spark Engine

Al fine di limitare le possibilità di guasti o perdita dati, Spark Streaming supporta un concetto chiamato "*checkpointing*" che garantisce che tutti i dati e i metadati associati agli RDD che formano i flussi di dati vengano continuamente salvati nella memoria. In caso di questo ciò consente al driver di recuperare il flusso di dati possa essere recuperato ed elaborato nuovamente e facilmente.

2.5.1 Modelli di elaborazione

Spark Streaming supporta diversi modelli corrispondenti alla semantica solitamente utilizzata per l'elaborazione dei flussi. Ciò garantisce che il sistema fornisca risultati affidabili,

anche in caso di guasti del nodo. I flussi di dati possono essere elaborati secondo i seguenti modelli:

- Esattamente una volta (cioè *exactly once*). Ogni elemento viene elaborato una sola volta.
- Al più una volta (cioè *at most once*). Ogni elemento può essere elaborato una volta e non può essere rielaborato.
- Almeno una volta (cioè *at least once*): ogni elemento deve essere elaborato almeno una volta. Ciò aumenta la possibilità che i dati non vengano persi, ma è anche possibile che vengano generati duplicati (cioè inconsistenza e ridondanze nell'analisi).

Dal punto di vista del processo, il modello più semplice da costruire è "al massimo una volta", quindi il secondo. Ciò che questo scenario implica accettabile l'occasionale perdita di dati, poiché ciò che conta di più è mantenere la continuità del flusso a discapito di una perdita poco significativa. Un esempio è uno streaming video: di volta in volta i pacchetti di informazioni vengono persi e la qualità viene ridotta un po', ma l'importante è che la comunicazione venga mantenuta e che non debba ricominciare dall'inizio. Rispetto al modello "almeno una volta", quindi il terzo, è scontato far notare che anche se può accadere qualche danno in qualche nodo i dati non saranno persi; anche perché quando il nodo recupera, o il suo carico di lavoro viene riassegnato ad un altro, elaborerà tutti i dati per garantire che nessuno di loro vada perso. Sebbene il modello "esattamente una volta" sia ciò che sembrerebbe logico da scegliere in ogni caso, in realtà non è necessario in ogni momento e il suo utilizzo non è così favorevole. E' importante, infatti, far notare che questo modello è il più dispendioso in termini di risorse e può causare problemi di prestazioni a causa di tutta l'ulteriore elaborazione necessaria per garantire che ogni dato non venga perso o duplicato. Può infatti portare ritardi e inconsistenza nei risultati delle operazioni svolte.

2.5.2 Conseguenze dei micro-batch

Come si è visto, Spark Streaming non funziona prettamente sulla base di flussi continui di dati ma su *micro-batch*, quindi mirco-lotti di dati, che hanno un intervallo tra di loro (cioè in genere *meno di 5 secondi*).

Tuttavia, è importante comprendere le conseguenze che ciò può succedere quando si fa ricorso ad un approccio del genere. Da un lato, è possibile impostare e ridurre l'intervallo fino a tempi minori di *1sec*, il che fornirebbe prestazioni in tempo praticamente reale, ma ad un costo elevato in termini di risorse di elaborazione e scheduling. Inoltre, i dati potrebbero non essere ricevuti nell'ordine esatto in cui sono stati generati. Questo può essere rilevante a seconda dell'applicazione specifica. Ad esempio, in una timeline di Twitter potrebbe non essere essenziale che i tweet vengano elaborati esattamente nello stesso ordine in cui sono stati pubblicati. Tuttavia, in un contesto di cybersecurity, in cui si concretizza il progetto di tesi, il fattore tempo è essenziale per monitorare consistentemente il comportamento di una rete al fine di rilevarne anomalie e ricostruirne eventi da notificare. In ogni caso l'uso di macro-batch renderebbe tediosa e poco affidabile l'esecuzione e il processamento a causa di una onerosa e continua gestione delle memoria.

2.6 Spark MLlib

MLlib [43][44][45] è la libreria di machine learning (cioè *ML*) appartenente alla suite applicativa di Spark. Il suo obiettivo è rendere il processo di Machine Learning pratico e scalabile. Ad un livello elevato, fornisce strumenti come:

- Algoritmi di machine learning: algoritmi più diffusi come classificazione, regressione, clustering e filtraggio collaborativo
- Caratterizzazione: estrazione delle caratteristiche, trasformazione, riduzione della dimensionalità e selezione
- Pipelines: strumenti per la costruzione, la valutazione e l'ottimizzazione delle pipeline ML

- **Persistenza:** salvataggio e caricamento di algoritmi, modelli e pipeline
- **Utilità:** algebra lineare, statistiche, gestione dei dati, ecc.

Dall’inizio del progetto Apache Spark, MLlib è stato considerato fondamentale per il suo successo e la sua diffusione. Il vantaggio principale di MLlib è che consente al data scientist di concentrarsi sui requisiti del problema e sulle strutture dati da considerare invece di risolvere le complessità che circondano il dover lavorare con dati distribuiti (cioè infrastruttura, configurazioni, fault tolerance). Altrettanto importante considerare che Spark MLlib è una libreria generica che fornisce algoritmi per la maggior parte dei casi d’uso e allo stesso tempo consente alla comunità di svilupparla ed estenderla per nuovi casi d’uso specializzati. In linea di massima i vantaggi MLlib includono:

- *Semplicità:* semplici API interfacciabili tramite linguaggi più comuni come Scala, R e Python. E’ possibile fin da subito implementare un algoritmo di ML e parametrizzarlo, se si è utenti esperti.
- *Scalabilità:* possibilità di eseguire lo stesso codice ML in locale piuttosto che mediante l’utilizzo di cluster.
- *Streamlined end-to-end:* sviluppare con MLlib su Spark consente di affrontare queste esigenze distinte con un unico strumento invece di molti disaccoppiati. I vantaggi sono le curve di apprendimento decisamente più basse, gli ambienti di sviluppo e produzione meno complessi e, in definitiva, i tempi più brevi per fornire modelli affidabili ad alte prestazioni.
- *Compatibilità:* spesso si dispone di flussi di lavoro sviluppati in strumenti analoghi, come *Scala*, *Python*, *pandas*[46] e *scikit-learn*[47]. Spark MLlib fornisce strumenti che facilitano l’integrazione di questi flussi di lavoro già esistenti lavorando in maniera interoperabile.

Ci sono una serie di casi d’uso aziendali comuni che possono comprendere l’uso di Spark MLlib. Di seguito qualche esempio più comune[48]:

- Ottimizzazione del marketing e della pubblicità: *in base al comportamento dell'utente quali prodotti dovrebbero essere raccomandati a ciascun utente per massimizzare le entrate?*
- Monitoraggio della sicurezza (cioè sicurezza informatica), compresa la valutazione del rischio e il monitoraggio della rete: *in base allo storico dei log di rete quali utenti mostrano comportamenti anomali e quali potrebbero essere dannosi?*
- Ottimizzazione operativa della supply chain o anche manutenzione preventiva: *in un determinato sistema è possibile individuare quali componenti possono incorrere in guasti che richiedono controlli preventivi?*

2.7 Prestazioni e benchmarking

Spark risulta notevolmente più veloce rispetto ad Hadoop[49] nell'esecuzione di algoritmi iterativi, grazie alla possibilità di memorizzare i dati in memoria. Se infatti la prima iterazione viene eseguita in tempi paragonabili a quelli di Hadoop, dalla seconda in poi il tempo necessario per ogni iterazione cala notevolmente, grazie appunto al meccanismo di *caching*. Vale lo stesso discorso per le interrogazioni interattive: la prima query ha lo stesso tempo di esecuzione su Spark e Hadoop, ma dalla seconda in poi Spark risulta nettamente più rapido rispetto a quest'ultimo. E' stato dimostrato che Spark possa risultare da 10 fino a 100 volte più veloce rispetto ad Hadoop MapReduce [34] eseguendo un semplice caso di esecuzione di regressione lineare (figura 2.10); questo principalmente grazie a specifiche ottimizzazioni tra cui l'uso primario della memoria cache.

In sintesi questo rappresenta, probabilmente, la scelta tecnologica più vantaggiosa nel panorama della Data Analytics.

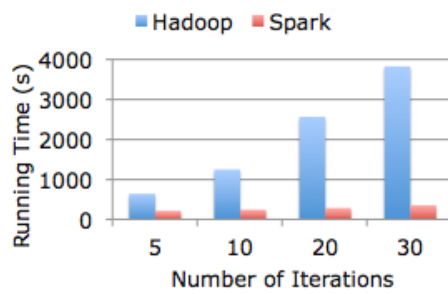


Figura 2.10: Comparazione delle prestazioni tra scenari di esecuzione (problema di regressione lineare) in Apache Hadoop e Apache Spark

Capitolo 3

Definizione del problema

In questo capitolo è fornita una panoramica del problema affrontato nello svolgimento del progetto di tesi comprendendo una descrizione della situazione attuale e delle problematiche iniziali, un approfondimento sulle sfide proposte e, infine, gli obiettivi del progetto stesso.

3.1 Descrizione architettura di monitoraggio attuale: *Suricata* e *Alien Vault*

Il campus Unibo di Forlì-Cesena comprende numerosi corsi di Laurea e, di conseguenza, numerose sedi didattiche e dipartimenti. Si stima che simultaneamente in media siano attive costantemente circa 3000 macchine interconnesse tra di loro e con la rete esterna (comprensivo di server, stampanti, videocamere, cluster etc).

La grande quantità di risorse connesse quotidianamente in rete assume una certa importanza visti i dati sensibili che gestiscono e immagazzinano. Per questo motivo, l'intera rete ha la necessità primaria di essere costantemente monitorata per garantirne il funzionamento, ma specialmente la sicurezza in termini di affidabilità e integrità contro attacchi informatici. Per far fronte a questa problematica di sicurezza informatica vengono adottati numerosi strumenti certificati e costantemente aggiornati che monitorano il traffico di rete verso l'interno e verso l'esterno persistentemente.

A questo proposito, affiancati ad un Firewall esteso a tutta la rete interna (Fortinet Fortigate[50]) si individuano i due componenti principali usati per lo scopo:

- L'IDS *Suricata*[51]
- *Alien Vault*[52], un sistema di Unified Threat Management(UTM)[53]

Il primo è un motore di rilevamento delle minacce di rete gratuito, open source, veloce, sempre aggiornato e robusto. Si basa sull'individuamento di sessioni anomale in base ad un set di regole che può essere esteso ad un maggior numero di possibili attacchi (*signature-anomaly based*, vedi capitolo 1). Il secondo, invece, introdotto recentemente e a pagamento, va ad integrare completamente Suricata al fine di aumentare la qualità e l'affidabilità del monitoraggio della rete non solo in chiave di rilevamento di eventuali anomalie ma anche di prevenzione di minacce e difesa dagli attacchi.

Un UTM infatti, oltre a includere un modulo IDS fornisce un set di funzionalità ad alto livello molto numeroso. Questo genere di componente infatti comprende di base:

- Firewall di rete
- Rilevamento delle intrusioni (IDS)
- Prevenzione delle intrusioni (ISP)
- Gateway anti-virus
- Firewall e controllo del livello dell'applicazione (livello 7 ISO-OSI[54]) e deep packet inspection.
- Web proxy and content filtering
- Prevenzione della perdita di dati (sistemi di Data Loss Prevention[55]) e sistema di gestione degli eventi di sicurezza (SIEM)
- Rete privata virtuale (VPN) e Tarpit di rete

Alien Vault in particolare è una piattaforma all-in-one e viene classificato come un sistema di Unified Security Management™(USM™), ossia una specializzazione dell'UTM. Questo comprende:

- *Asset Discovery*: scoperta della rete attiva e passiva.
- *Vulnerability Assessment*: scansione di rete attiva, monitoraggio continuo delle vulnerabilità.
- *Rilevamento delle minacce*: IDS di rete e host, monitoraggio dell'integrità dei file.
- *Monitoraggio del comportamento della rete*: analisi del flusso di rete, normalizzazione del registro, monitoraggio della disponibilità del servizio.
- *SIEM*: gestione dei log, correlazione degli eventi SIEM, analisi e reporting. Mette a disposizione del sysadmin una UI intuitiva con dashboard descrittive che presentano grafici, indicatori e utili alert in tempo reale rispetto all'operato continuo che effettua il sistema nell'intera rete.

Alien Vault fornisce una gestione unificata della sicurezza ed è composto da tre componenti principali disponibili come hardware o dispositivi virtuali (figura 3.1) :

- Sensore USM: distribuito in tutta la rete per raccogliere i registri per fornire le funzionalità di sicurezza essenziali necessarie per una visibilità completa del comportamento della rete.
- USM Server: aggrega e correla le informazioni raccolte dai sensori e fornisce un'unica gestione con attività reporting e amministrazione.
- USM Logger: archivia in modo sicuro i dati del registro eventi non elaborati per indagini forensi e requisiti di conformità.

Oppure da degli USM All-in-One che combinano i componenti Server, Sensor e Logger su un singolo sistema.

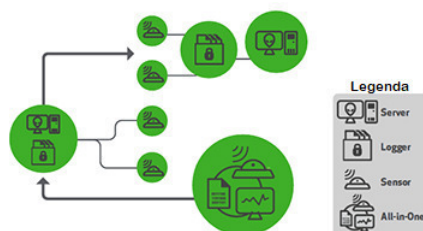


Figura 3.1: Componenti principali sistema USM™ Alien Vault

3.2 Problematiche

L'architettura di monitoraggio presentata viene continuamente aggiornata ed è soggetta a costante controllo e manutenzione. Nonostante ciò questa presenta colli di bottiglia evidenti e limitazioni nell'elaborazione dell'intero traffico di rete. Con la significativa mole di dati scambiati nella rete interna e verso l'esterno (circa 1GB/s) l'intera architettura, prettamente centralizzata, è soggetta all'inevitabile perdita di eventi. Ciò rende l'analisi e il controllo complessivo incompleto (fino al 40% dell'intero carico di lavoro è perso). Oltre a questa limitazione, un altro aspetto fondamentale da considerare è che l'intero sistema di controllo agisce soltanto a livello di singolo pacchetto, o comunque aggregando a livello di singola sessione di comunicazione. Questo, tuttavia, non permette l'individuamento di situazioni assimilabili a pattern di attacchi e minacce che solo a maggiori livelli di aggregazione sarebbe possibile estrarre (per esempio il *DDoS*[56] o un tentativo di *Portscanning*[57])

Queste limitazioni sono superabili grazie ad un'architettura di supporto per l'analisi di dati su piattaforme per il calcolo parallelo e distribuito. Acquisire un supporto del genere che si impone di fornire un set di funzioni atte al rendere il monitoraggio della rete più efficiente con due importanti innovazioni:

- Aumenta sensibilmente la complessità introducendo nuove tecniche più avanzate di analisi dati su piattaforme apposite, estendendo le capacità e le feature del sistema corrente.

- Riduce la perdita di dati che potrebbero generare alert e informazioni utili incrementando l'affidabilità del monitoraggio.

A tal scopo sono stati definiti un elenco di requisiti che andranno a validare lo sviluppo di un sistema di monitoraggio integrato al modello esistente, basato su Big Data Analytics e tecniche di machine learning in un contesto real-time (cioè di data stream processing).

3.3 Requisiti

Per far fronte alle problematiche presentate dal committente del progetto, ossia il centro di sicurezza informatica ACCF, è riportata una lista di desiderata che porteranno alla definizione del nuovo modello.

- Il sistema dovrà far riferimento ad un'architettura distribuita che supporti il calcolo parallelo. L'utilizzo di un'infrastruttura del genere garantirà continuità operativa in tempo reale grazie a meccanismi di fault-tolerance e disaster recovery. Per permettere ciò è buona pratica che le scelte tecnologiche ricadano sull'utilizzo di framework specifici per l'analisi dati con esecuzione su cluster (per esempio Apache Spark, capitolo2).
- Il sistema includerà moduli integrativi al modello AS-IS di supporto incrementando la complessità di quest'ultimo ed espandendone il feature set. Il componente, a tal proposito, includerà l'utilizzo di tecniche di machine learning e data mining per l'estrazione di nuove differenti forme di conoscenza avvalorando per intero processo di monitoraggio e protezione di rete.
- Vista la grande quantità di dati generati, l'alta frequenza con cui questi si presentano e l'eterogeneità dell'informazione accessibile, il progetto comprenderà aspetti di *Big Data Management*. Un approccio del genere permetterà di lavorare senza limitazioni a livello di data processing evitando ritardi nell'esecuzione o perdite significative dell'informazione.

- Il sistema dovrà integrare l'attività dell'attuale in termini di interoperabilità di rete senza vincolare o ostacolarne la completa esecuzione.
- Il sistema farà ricorso ad un approccio completamente real-time.
- Le nuove funzionalità del sistema permetteranno di non solo replicare alcune feature già esistenti in un contesto più efficiente e affidabile, ma estenderanno quest'ultime in base alle nuove necessità presentate dal committente non previste nell'architettura di monitoraggio attuale (per esempio la ricerca di nuovi scenari di cyberattacco)
- Il sistema dovrà generare report intuitivi e facilmente fruibili dal committente e i sysadmin al fine di segnalare situazioni anomale o semplici aggiornamenti sullo stato della rete in base ad alcune misure e dimensioni predefinite.
- Vista la natura ancora molto sperimentale del progetto questo dovrà essere progettato tenendo in considerazione la possibilità di una successiva espansione evolutiva con l'introduzione di nuovi moduli applicativi nelle successive release.
- Per facilitare la fruibilità, l'analisi delle informazioni e la valutazione dei risultati, il sistema dovrà far uso di dati in formati standard rispettando le principali notazioni (cioè *JSON*, *CSV*).
- Per ottemperare alla regolamentazione europea per la protezione dei dati (cioè *General Data Protection Regulation, GDPR*[58][59]) dovendo lavorare con informazioni personali e sensibili, univocamente riconducibili ad un utente, tutti gli indirizzi *IPv4* [60] dovranno essere sottoposti ad un processo di anonimizzazione[61] prima di essere elaborati all'intero nell'infrastruttura e/o storicizzati.
- Per uno sviluppo corretto e consistente già dalla prima release tutti i risultati ottenuti verranno valutati dal committente con l'eventuale possibilità di estendere i requisiti e i test di validazione.

Rispetto a questa collezione di requisiti viene proposto il progetto di tesi: ***Styx*** (dettagli nel capitolo 4).

Capitolo 4

Analisi e prototipo: Styx

In questo capitolo viene introdotto e spiegato nel dettaglio il progetto di tesi *Styx* per la realizzazione di un intrusion detection system su piattaforma Big Data, con l'ausilio di tecniche di data mining e machine learning. Il nome del progetto deriva da *Stige*, uno dei cinque fiumi presenti negli Inferi secondo la mitologia greca e romana. La sua acqua era utilizzata dal dio Zeus per scovare i traditori tra gli dei.

4.1 Dataset e sorgenti

Per la progettazione e lo sviluppo del sistema sono state messi a disposizione tre sorgenti fondamentali a cui è possibile accedervi sia in tempo reale sia immagazzinandone campioni per successive analisi. I dati provenienti da queste sorgenti hanno reso possibile un'analisi del problema identificando quali siano gli aspetti di particolare interesse. Inoltre, dopo una fase di pre-processing delle varie sorgenti, queste hanno presentato per la gran parte il carico di lavoro della prima release presentata nel prototipo.

Netflow

Il flusso NetFlow è stato direttamente reindirizzato dal firewall Fortigate del campus. Il firewall Fortinet, modello Fortigate, utilizzato dal centro di sicurezza informatica del campus si occupa, come mansione principale, del filtraggio dei vari pacchetti che circolano tra le

macchine collegate alla rete universitaria e quelle appoggiate su ISP esterni, e quindi LAN private. Si tratta di un flusso di dati che forniscono log rispetto alle singole comunicazioni di rete tra un indirizzo della rete interna e uno esterno o semplicemente tra due interni. L'informazione, a livello di singola comunicazione, è strutturata come rappresentato dalla tabella in figura 4.1. Come è possibile evincere dalla figura, sono fornite informazioni su:

The image shows a screenshot of a table titled "Source 1 - NetFlow". The table lists various fields related to network traffic. The fields are: IP SORGENTE, IP DESTINAZIONE, PORTA SORGENTE, PORTA DESTINAZIONE, FIRST_SWITCH, LAST_SWITCH, BYTE SCAMBIATI, PROTOCOLLO DI RETE, and INTERFACCIA DI RETE. The table is presented in a grid-like format with a light gray background.

Source 1 - NetFlow
IP SORGENTE
IP DESTINAZIONE
PORTA SORGENTE
PORTA DESTINAZIONE
FIRST_SWITCH
LAST_SWITCH
BYTE SCAMBIATI
PROTOCOLLO DI RETE
INTERFACCIA DI RETE

Figura 4.1: Struttura sorgente 1 - NetFlow

- Indirizzo IP sorgente: il mittente della comunicazione.
- Indirizzo IP destinazione: il destinatario della comunicazione.
- Porta sorgente: la porta applicativa utilizzata dal mittente per trasmettere.
- Porta destinazione: la porta applicativa utilizzata dal destinatario per ricevere.
- Inizio della trasmissione (*first_switch*) sotto forma di timestamp.
- Fine della trasmissione (*last_switch*) sotto forma di timestamp.
- Byte scambiati durante la trasmissione.
- Il protocollo di rete utilizzato (per esempio TCP[62], UDP[63]).

- L'interfaccia di rete (per esempio tramite rete Wi-Fi o ethernet).

Suricata

L'informazione fornita da Suricata rappresenta un arricchimento di ciò che è racchiuso in Netflow. Il flusso dell'IDS d'Ateneo, infatti, presenta quelli che sono i casi anomali, quindi le sessioni che, a seguito di un match con delle regole specifiche denotano un comportamento atipico di alcune macchine fornendone una prima etichettatura classificativa (per esempio, un uso non autorizzato del dns d'Ateneo, o un malware). Rispetto a Netflow, inoltre, l'IDS effettua un processo di aggregazione individuando le sessioni di comunicazione univoche e non solo le singole trasmissioni. I dati presentano i seguenti attributi (figura 4.2):

- I due IP e le rispettive porte che identificano la sessione.
- Un timestamp come riferimento temporale alla segnalazione dell'evento.
- Il tipo di evento.
- Il protocollo di rete utilizzato.
- L'interfaccia di rete (per esempio tramite rete Wi-Fi o ethernet).

Alien Vault

Infine, l'ultima sorgente dati che è stata presa in considerazione è generata da Alien Vault. Il componente funge da validatore degli eventi segnalati da Suricata aggiungendone conoscenza. In più, svolgendo un'attività di monitoraggio completa e più approfondita, può identificare eventi di altro genere o comunque non riportati nei log della sorgente 2. La sua struttura è composta come in figura 4.3:

- I due IP e le rispettive porte che identificano la sessione.
- Un timestamp come riferimento temporale alla segnalazione dell'evento.



Figura 4.2: Struttura sorgente 2 - Suricata



Figura 4.3: Struttura sorgente 3 - Alien Vault

- Il tipo di evento classificato secondo un dizionario interno del componente (per esempio *User Privilege Escalation*[64], *Backdoor*[65], *Reverse-tunnel SSH*[66]).
- La priorità della segnalazione. Alien Vault, infatti, distingue i log generati in un range di 5 classi di priorità, dalla meno importante alla più critica.

- Il protocollo di rete utilizzato.

4.2 Architettura generale e componenti

L'intera architettura del progetto è descritta in figura 4.4. Come già precedentemente

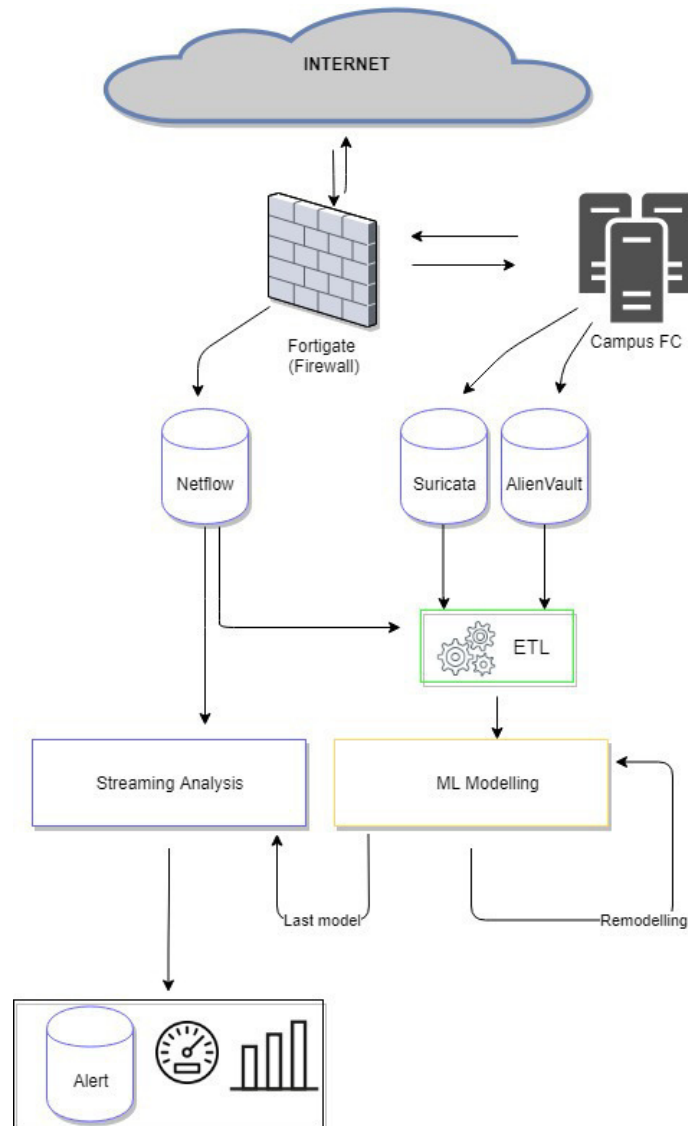


Figura 4.4: Architettura generale Styx

anticipato, questa si divide sostanzialmente in due componenti ben distinti affrontando due livelli applicativi differenti:

- Componente di Data stream processing.
- Componente di modellazione predittiva basato su tecniche di mining e machine learning.

4.2.1 Componente DSP e scenari d'attacco considerati

Il componente di data stream processing si occupa di ricevere il flusso di dati da NetFlow, come si evince dallo schema. Le comunicazioni della rete interna verso l'esterno e viceversa, infatti, sono completamente monitorate dal firewall d'Ateneo (Fortigate) dal quale è possibile ricevere l'intero traffico sotto forma di *header* in notazione *JSON*. I task designati a questo componente sono i seguenti:

- In base alle informazioni a livello di singola trasmissione individuare le sessioni di comunicazione rilevando le anomalie in termini di durata (sessioni di una durata atipica) o "pesantezza" (alto numero di byte scambiati/scaricati/caricati in rete) e generarne dei report illustrativi. A tal merito, da una precedente analisi su un campione di dati raccolti offline (vedi 4.4.1) è stato possibile individuare delle *soglie* poi utilizzate nell'ambito di streaming sia per quanto riguarda la durata sia per i dati complessivamente scambiati in ogni sessione.
- Aggregare le informazioni generali sul traffico di rete rispetto a determinati attributi (per esempio IP interni, durate, etc) al fine di scovare situazioni malevole che fanno riferimento a determinati scenari di cyber-attacco e generarne alert in tempo reale.
- Per mezzo dei modelli predittivi costantemente aggiornati, cercare di individuare, a fronte di un processo di apprendimento automatico, possibili casi riconducibili a situazioni malevole di diversa classificazione. Il modello di apprendimento viene fornito dal componente ML e si basa su un campione di dati precedentemente raccolti ed propriamente etichettati in base alla criticità della situazione (vedi 4.2.2).

In merito alla seconda voce, gli scenari di attacco presi in considerazione inizialmente sono stati di 4 tipologie:

- *Sessioni lunghe*: identificano anomalie riconducibili a scenari di intrusione e fuga di dati.
- *Sessioni pesanti*: identificano anomalie riconducibili a possibile uso improprio della rete, download di file pirata, applicazioni P2P per il traffico di materiale illecito.
- *Port Scan*.
- *Distributed Denial of Service*.

Port Scan

Un port scan [57] è un attacco che invia richieste client a un determinato intervallo di indirizzi di porta di un host, con l'obiettivo di trovare una porta attiva e sfruttare una vulnerabilità nota di quel servizio assegnato a quella porta. Convenzionalmente una scheda di rete gestisce fino a 65536 numeri di porta distinti e utilizzabili; anche se la maggior parte dei servizi utilizzano una gamma limitata (per esempio HTTPS la 443, SSH la 22). La scansione, come metodo per scoprire canali di comunicazione sfruttabili, è un tentativo di intrusione ormai ben noto. L'idea è di sondare quanti più ascoltatori possibile e tenere traccia di quelli che sono ricettivi o utili. Una particolare evoluzione del port scan è rappresentato dal *port sweep*. Il portsweep differisce da portscan, perché più host vengono scansionati per una specifica porta di ascolto. Ad esempio, se l'utente malintenzionato vuole scoprire tutti i server Web in esecuzione nella rete di destinazione, il portsweep viene lanciato verso la porta 80 e 443 soltanto ma verso tutti gli host nella rete. Di seguito sono riportati alcuni dei tipi più comuni di port scan:

- *Scansione TCP Connect*: la scansione TCP connect è la forma più semplice di scansione TCP. La chiamata di sistema connect() fornita dal sistema operativo viene utilizzata per aprire una connessione (socket) a qualsiasi porta sul computer di destinazione. Se la porta è in ascolto, l'operazione connect() avrà esito positivo, in caso contrario la porta non risulterà raggiungibile.

Come vantaggio si ha che l'utente/attaccante non richiede alcun privilegio speciale per la scansione. La scansione può essere eseguita molto più rapidamente poiché la

velocità data dagli script automatizzati permette di verificare la validità della porta molto rapidamente. Un suo svantaggio invece è la facile rilevabilità dai sistemi IDS e IPS, come descritto anche nel progetto di tesi.

- *Scansione TCP SYN*: la scansione TCP SYN è anche conosciuta come scansione "semiaperta", in quanto il mittente/utente malintenzionato non apre una connessione TCP completa. L'utente malintenzionato invia un pacchetto SYN e attende una risposta. Una risposta SYN/ACK dalla destinazione indica che la porta è in ascolto e non appena viene ricevuto un SYN/ACK, il mittente o l'utente malintenzionato invia una risposta RST (figura 4.5).

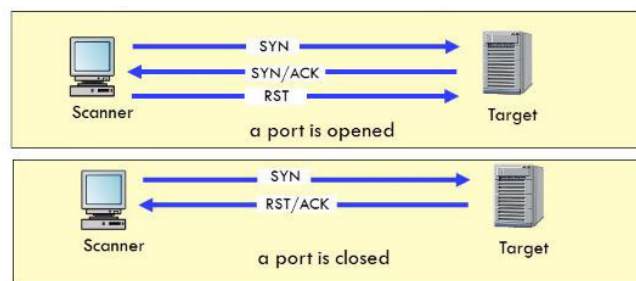


Figura 4.5: Port Scan versione TCP SYN

- *Scansione TCP FIN* : la maggior parte dei firewall e dei dispositivi IPS monitor, rileva e filtra la scansione basata su SYN. Ma molti firewall e dispositivi IPS consentono l'invio di pacchetti FIN. Quando i pacchetti FIN vengono inviati dall'attaccante alla destinazione, le porte chiuse rispondono con in risposta con un pacchetto RST, mentre le porte aperte ignorano i pacchetti FIN inviati dall'attaccante (figura 4.6). Mentre questo metodo funziona nei sistemi unix-like, i sistemi operativi basati su Microsoft Windows inviano le risposte al pacchetto RST all'attaccante in entrambe le condizioni, sia che la porta sia chiusa o aperta. Quindi potrebbe essere usato anche per distinguere host windows da host unix. Questo attacco è sicuramente il miglior candidato perché può essere utilizzato per eludere molti firewall e sistemi IPS.
- *Fragmentation scanning* : la scansione frammentata viene eseguita per eludere i firewall e altri dispositivi di filtraggio dei pacchetti inviando le intestazioni TCP in

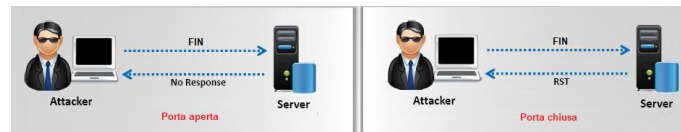


Figura 4.6: Port Scan versione TCP FIN

piccoli frammenti, che potrebbero confondere i firewall e altri dispositivi durante l'elaborazione dei pacchetti stessi. Mentre molti firewall possono accodare i pacchetti per l'assemblaggio, con questo tipo di scansione si può consumare totalmente la potenza di elaborazione dell'host della vittima rendendo accessibile un servizio apparentemente non consentito.

In particolare, in Styx, il port scan è segnalato quando vengono rilevate una serie di connessioni da un IP esterno su un IP interno su porte differenti in tempi corti.

Distributed Denial of Service

Un (DDoS) [56] attacco denial-of-service distribuito è un tentativo dannoso di perturbare il normale traffico di un server, un servizio o di una rete intera al fine di generare un disservizio per il bersaglio o le sue infrastrutture circostanti, con un flusso di traffico Internet. Gli attacchi DDoS raggiungono l'efficacia utilizzando molteplici sistemi informatici compromessi come fonti di traffico di attacco. Le macchine sfruttate possono includere computer e altre risorse di rete come i dispositivi IoT, dove la componente sicurezza è ancora poco presa in considerazione. In altre parole, un attacco DDoS è come paragonabile ad un ingorgo nella strada principale che impedisce il traffico regolare di arrivare a destinazione desiderata. Un attacco DDoS richiede a un utente malintenzionato di assumere il controllo di una rete di macchine online per effettuare un attacco. Computer e altre macchine (come i dispositivi IoT) sono infettati da malware, trasformando ognuno in un bot (o zombi). L'utente malintenzionato ha quindi il controllo remoto sul gruppo di bot, che è chiamato botnet. Una volta stabilita una botnet, l'attaccante è in grado di dirigere le macchine inviando istruzioni aggiornate a ciascun bot tramite un metodo di controllo remoto. Quando l'indirizzo IP di una vittima è preso di mira dalla botnet, ciascun bot risponderà inviando richieste alla destinazione, causando potenzialmente un sovraccarico della rete o del server di destinazione, con conseguente negazione del servizio al traffico

normale. Poiché ogni bot è un dispositivo Internet legittimo non sospetto, la separazione logica del traffico di attacco dal traffico normale può essere difficile da effettuare. Esistono diverse tecniche di attacco DDOS:

- *Attacchi di livello applicazione*: a volte indicato come attacco DDoS di livello 7 (in riferimento al settimo livello del modello ISO-OSI), l'obiettivo di questi attacchi è di esaurire le risorse della vittima. Gli attacchi hanno come obiettivo il livello in cui le pagine Web sono generate sul server e consegnate in risposta alle richieste HTTP. Una singola richiesta HTTP è a basso costo per l'esecuzione sul lato client ma può essere costosa per il server di destinazione rispondere in quanto il server spesso deve caricare più file ed eseguire query di database per creare una pagina web o parti di quest'ultima. Gli attacchi di livello 7 sono difficili da difendere in quanto il traffico può essere difficile da contrassegnare come dannoso. Un esempio è l'HTTP Flood (figura 4.7).

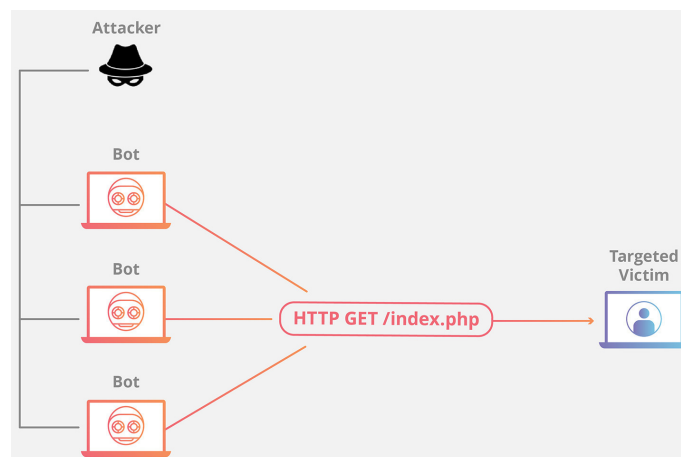


Figura 4.7: HTTP Flood - DDoS a livello applicativo

- *Attacchi a livello di protocollo*: gli attacchi di protocollo, noti anche come attacchi di esaurimento dello stato, causano un'interruzione del servizio consumando tutta la capacità disponibile della tabella di stato dei server di applicazioni Web o di risorse intermedie come i firewall. Gli attacchi al protocollo utilizzano punti deboli nel livello 3 e nel livello 4 dello stack del protocollo per rendere la vittima inaccessibile.

Un esempio è il SYN Flood (figura 4.8). Un SYN Flood è analogo a un lavoratore in una stanza delle forniture che riceve richieste dalla parte sul retro del negozio. Il lavoratore riceve una richiesta, va a prendere il pacco e attende la conferma prima di portare il pacco davanti. L'operatore riceve quindi molte più richieste di pacchetti senza conferma fino a quando non è in grado di trasportare altri pacchetti le richieste iniziano a non ricevere risposta. Questo attacco sfrutta l'handshake TCP inviando un numero elevato di pacchetti SYN TCP (Initial Connection Request) con indirizzi IP di origine fittizia. La macchina vittima risponde a ciascuna richiesta di connessione e quindi attende il passaggio finale nell'handshake, che non avviene mai, esaurendo le risorse molto presto.

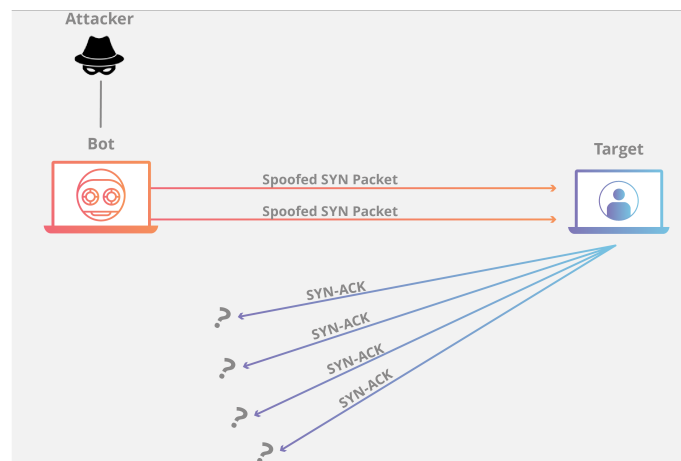


Figura 4.8: SYN Flood - DDoS a livello di protocollo

- *Attacchi volumetrici* Questa categoria di attacchi tenta di creare una congestione consumando tutta la larghezza di banda disponibile. Grandi quantità di dati vengono inviate a un target utilizzando una forma di amplificazione o un altro mezzo per creare un traffico enorme, come le richieste da una botnet. Un esempio famoso è l'amplificazione del DNS (figura 4.9). Effettuando una richiesta a un server DNS aperto con un indirizzo IP falsificato (l'indirizzo IP reale della destinazione), l'indirizzo IP di destinazione riceve una risposta dal server. L'utente malintenzionato struttura la richiesta in modo tale che il server DNS risponda all'obiettivo con una

grande quantità di dati. Di conseguenza, la vittima riceve un'amplificazione della query iniziale dell'aggressore creando un ingorgo applicativo.

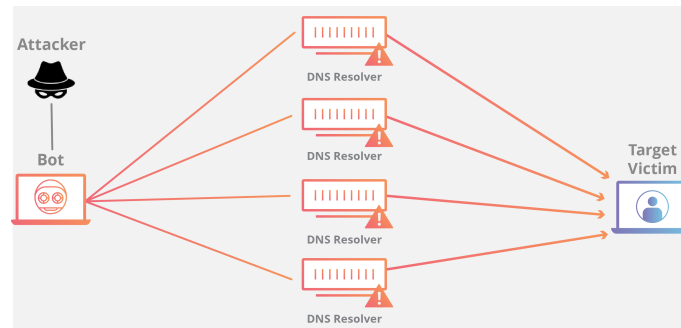


Figura 4.9: Amplification DNS - DDoS volumetrico

In particolare, per il progetto di tesi il componente di DSP procede all'identificazione di un DDoS al verificarsi della presenza di un gran numero di indirizzi differenti (presunta bot net) che si collegano in un range di tempo molto ridotto ad una determinata macchina (presunta vittima) scambiando un volume elevato di byte.

4.2.2 Componente ML e algoritmi di mining utilizzati

Il componente per la modellazione predittiva si basa su algoritmi di classificazione e clustering, quindi prende in considerazione sia un approccio supervisionato mediante apposite etichette derivate dal processo di integrazione ed ETL delle tre sorgenti (vedi 4.4.2) che uno non supervisionato in ogni caso effettuato sullo stesso dataset riconciliato. Gli algoritmi messi a disposizione dal framework utilizzati sono numerosi; per questa prima release, il prototipo fa uso di due in particolare:

- *Random Forest* (classificazione supervisionata).
- *K-Means* (clustering non supervisionato).

Random Forest

Random Forest è un algoritmo di apprendimento automatico flessibile e facile da usare che produce, anche senza troppo tuning iperparametrico, un ottimo risultato per la maggior parte dei casi. È anche uno degli algoritmi più utilizzati per la sua semplicità e la

propensione per le attività di classificazione e di regressione che costituiscono i problemi principali per la maggior parte dei sistemi di apprendimento automatico. Random Forest è un algoritmo di apprendimento supervisionato. Come si può intuire dal suo nome, crea una "foresta" e la rende in qualche modo casuale. La "foresta" che costruisce è un insieme di *Decision Trees*, la maggior parte delle volte addestrato con il metodo "bagging", cioè di "insacco". L'idea generale del metodo di insacco si basa su una combinazione di modelli di apprendimento che aumenta il risultato complessivo. In altre parole, la foresta casuale crea più alberi decisionali e li unisce per ottenere una previsione più accurata e stabile (un esempio in figura 4.10).

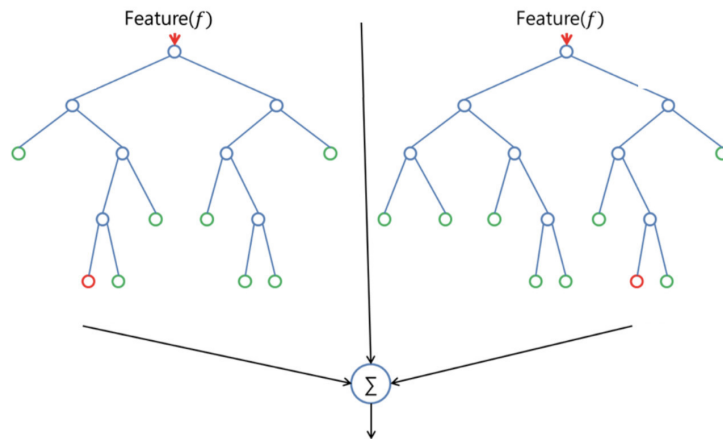


Figura 4.10: Random Forest composto da 2 soli alberi

Random Forest ha quasi gli stessi iperparametri di un albero decisionale ma aggiunge ulteriore casualità al modello perché al crescere degli alberi. Invece di cercare la caratteristica più importante mentre divide un nodo, cerca la migliore funzionalità tra un sottoinsieme casuale di funzionalità. Ciò si traduce in un'ampia diversità che generalmente si traduce in un modello migliore. Pertanto, in Random Forest, solo un sottoinsieme casuale delle funzionalità viene preso in considerazione dall'algoritmo per la divisione di un nodo. È anche possibile rendere gli alberi più casuali, aggiungendo soglie casuali per ciascuna

caratteristica piuttosto che cercare le migliori soglie possibili (come fa un normale albero delle decisioni). Random Forest è una raccolta di Decision Trees, come già accennato, ma ci sono alcune differenze. Se si immette un set di dati di training con caratteristiche ed etichette in un albero decisionale, verrà formulato un gruppo di regole, che verranno utilizzate per effettuare le previsioni. Ad esempio, se si vuol pronosticare se una persona farà clic su un annuncio online, si potrebbe raccogliere l'annuncio sulla persona che ha fatto clic in passato e alcune caratteristiche che descrivono la sua decisione. Se si inseriscono le caratteristiche e le etichette in un albero decisionale, genererà alcune regole sui vari comportamenti della persona. Quindi è possibile pronosticare se l'annuncio verrà cliccato o meno. In confronto, l'algoritmo della foresta casuale seleziona in modo casuale le osservazioni e le caratteristiche per costruire diversi alberi decisionali e quindi calcola la *media* dei risultati massimizzando l'accuratezza. Un'altra differenza è che gli alberi decisionali "profondi" potrebbero soffrire di sovralimentazione. La foresta casuale impedisce il sovralfollamento la maggior parte del tempo, creando sottoinsiemi casuali delle caratteristiche e costruendo alberi più piccoli utilizzando questi sottoinsiemi. Successivamente, combina i sottoalberi. Random Forest è anche considerato un algoritmo molto maneggevole e facile da usare, perché gli il numero di iperparametri non è così elevato e sono intuibili. In primo luogo, c'è l'iperparametro "*n_estimators*", che è solo il numero di alberi che l'algoritmo costruisce prima di prendere il voto massimo o di fare delle medie delle previsioni. In generale, un numero maggiore di alberi aumenta le prestazioni e rende le previsioni più stabile, ma rallenta il calcolo. Un altro iperparametro importante è "*max_features*", ovvero il numero massimo di funzionalità che Random Forest può provare in un singolo albero. L'ultimo iper-parametro importante in termini di velocità è "*min_sample_leaf*". Questo determina, come dice il nome, il numero minimo di foglie necessarie per suddividere un nodo interno.

K-means

Il problema del K-means clustering può essere formalmente riassunto in "dati un intero e un insieme di n punti dati in R, l'obiettivo è scegliere i k centri in modo da ridurre la

distanza totale tra ciascun punto dati e il suo centro più vicino."

$$\frac{1}{n} \sum_j \sum_{x_i \in C_j} d(x_i, c_j)^2$$

È anche importante notare che non è richiesto che c_j debba essere compreso tra x_i .

Tuttavia, risolvere questo problema è *NP-hard*; ma ci sono algoritmi che possono localmente cercare delle soluzioni.

L'algoritmo standard fu proposto per la prima volta da Stuart Lloyd nel 1957, e si basa su un processo iterativo. Nel K-means si distinguono due elementi principali:

- **Centroids:** la costruzione dei cluster è semplice: ogni punto al suo centroide più vicino, altrimenti l'errore aumenta.
- **Clusters:** la ricerca dei centroidi si riduce nel minimizzare:

$$\sum_{x_i \in C} d(x_i, c)^2$$

forzando la derivata a zero; ogni centroide è impostato al centro del suo cluster, altrimenti l'errore aumenta.

In ogni caso, l'idea generale è, a partire da k centroidi iniziali *candidati*:

- Ricalcola i cluster;
- Ricalcola i centroidi;
- ... e si ripete.

Per la scelta dei candidati iniziali, tuttavia, ci sono più alternative:

- Prenderli semplicemente in modo casuale;
- Partendo da uno casuale per poi esplorare il più possibile la "nuvola di punti" intorno a quest'ultimo.

Chiaramente l'efficienza dei risultati dipende anche da quanto i dati sono effettivamente dispersi. È stato dimostrato che partendo da punti casuali e allontanandosi da essi fornisce un'interessante accuratezza nella validazione e nel test del modello. Con questo semplice insieme di passaggi, iterato per il tempo necessario, è possibile ridurre l'errore ed eseguire una sorta di divisione concettuale basata sulla somiglianza dei punti appartenenti al cloud.

Inoltre, è interessante capire come il risultato dell'analisi può variare in base al valore assunto dalla "K", racchiuso nel nome dell'algoritmo.

È probabilmente, tra i *hyper-parameters*, il più importante sia rappresentato dal numero effettivo di cluster esclusivi o classi che effettivamente divideranno il set di dati.

In breve, la scelta di k è decisiva. Ma come sceglierlo nel miglior modo possibile può essere complicato. Può essere scelto in base al numero di cluster esaminando visivamente i punti dati, rischiando però molta ambiguità in questo processo se non si lavora con dataset semplici. Questo tuttavia non è sempre negativo, perché trattandosi di apprendimento non supervisionato c'è una certa soggettività inerente nel processo di etichettatura.

In realtà, avere una precedente esperienza con quel particolare problema o qualcosa di simile aiuta molto a scegliere il giusto valore.

Inoltre, un metodo più formale dato per determinare il numero di cluster che potrebbe essere è il metodo *Elbow* [71]: questo si basa sull'*errore quadratico medio* (cioè SSE) per alcuni valori di k (ad esempio 2, 4, 6, 8, ...). L'SSE è definito come la somma della distanza quadrata tra ciascun membro del cluster e il suo centroide.

Formalmente:

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} d(x, c_i)^2$$

4.3 Scelte tecnologiche e hardware

Per lo sviluppo dell'applicazione è stato necessario anzitutto configurare un centro di elaborazione dati. Tutto il progetto, pertanto, è stato sviluppato su un cluster da 11 nodi di commodity hardware messo a disposizione dall'Ateneo, in particolare dal Business Intelligence Lab (BILab), che si occupa giornalmente di tali tematiche. Di seguito sono riportate le caratteristiche tecniche di ogni nodo:

- CPU: Intel i7-4790, 4 Core, 8 threads (Hyper-Threading), 3.6Ghz
- RAM: 32 GB
- HARD-Drive: 3x2TB HDD
- Ethernet: Gigabit
- Operative System: CentOS 6.6 (Linux)

Sul cluster è stato installato il modulo di analisi Apache Spark versione 2.1.3. All'indirizzo IP locale del driver, sulla porta 4040, in aggiunta, per facilitare la gestione e la supervisione di tutti gli executor, i job e gli stage a tempo di esecuzione delle varie applicazioni, Spark mette a disposizione una UI (User Interface) apposita e user-friendly (figura 4.11). L'utiliz-

The screenshot shows the Apache Spark UI interface. At the top, there are navigation tabs for Jobs, Stages, Storage, Environment, Executors, SQL, and Streaming. Below the tabs, there's a summary section for 'Spark Jobs (7)' with statistics like 'User: admin@...', 'Total Operator: 16 min', 'Active Jobs: 2', and 'Completed Jobs: 53'. The main part of the image is a table with columns: Job Id, Description, Submitted, Duration, Stages: Succeeded/Total, and Tasks (for all stages): Succeeded/Total. The table is divided into 'Active Jobs (2)' and 'Completed Jobs (50)'. The 'Active Jobs' section shows a single job with a 'Streaming job running operator' stage. The 'Completed Jobs' section lists multiple jobs, each with several stages and tasks, showing their completion status and counts.

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
54	save at SessionIdentifier.scala:494	2018/09/28 17:43:55	9 s	4/137	905/16333
0	Streaming job running operator	2018/09/28 17:28:13	18 min	0/1	0/1
Completed Jobs (50)					
53	Streaming job from [output operation 0, batch time 17:28:10]	2018/09/28 17:43:55	3 ms	1/1	1/1
52	Streaming job from [output operation 0, batch time 17:28:06]	2018/09/28 17:42:57	1.0 min	32/32 (100 skipped)	30052505 / 10120 (skipped)
51	Streaming job from [output operation 0, batch time 17:28:05]	2018/09/28 17:42:56	7 ms	1/1	1/1
50	Streaming job from [output operation 0, batch time 17:28:05]	2018/09/28 17:41:59	57 s	31/31 (96 skipped)	34856495 / 3724 (skipped)
49	Streaming job from [output operation 0, batch time 17:28:05]	2018/09/28 17:41:58	4 ms	1/1	1/1
48	Streaming job from [output operation 0, batch time 17:28:04]	2018/09/28 17:41:02	56 s	30/30 (82 skipped)	32063209 / 10120 (skipped)
47	Streaming job from [output operation 0, batch time 17:28:04]	2018/09/28 17:41:01	9 ms	1/1	1/1
46	Streaming job from [output operation 0, batch time 17:28:03]	2018/09/28 17:40:04	57 s	29/29 (88 skipped)	30089009 / 3812 (skipped)
45	Streaming job from [output operation 0, batch time 17:28:03]	2018/09/28 17:40:03	9 ms	1/1	1/1
44	Streaming job from [output operation 0, batch time 17:28:03]	2018/09/28 17:39:09	53 s	29/29 (84 skipped)	28954809 / 3574 (skipped)
43	Streaming job from [output operation 0, batch time 17:28:03]	2018/09/28 17:39:09	3 ms	1/1	1/1

Figura 4.11: Apache Spark UI

zo di Apache Spark permette lo sviluppo sia del componente real-time (*Spark Streaming*) sia di quello per il ML (*MLlib*). Ciò permette la generazione di un unico eseguibile compilato su cluster effettuandone un deployment distribuito. Per la creazione degli stream e la raccolta dati dalle tre sorgenti è stato adottato un message-oriented middleware (MoM)[72] chiamato Apache Kafka. Kafka viene utilizzato per la creazione di pipeline di dati in tempo reale e applicazioni di streaming. È scalabile orizzontalmente, fault-tolerant, veloce e compatibile con la gran parte dei framework per la data analysis. Un'idea generale sulle

varie tecnologie adottate nei vari componenti dell'architettura viene riportata in figura 4.12.

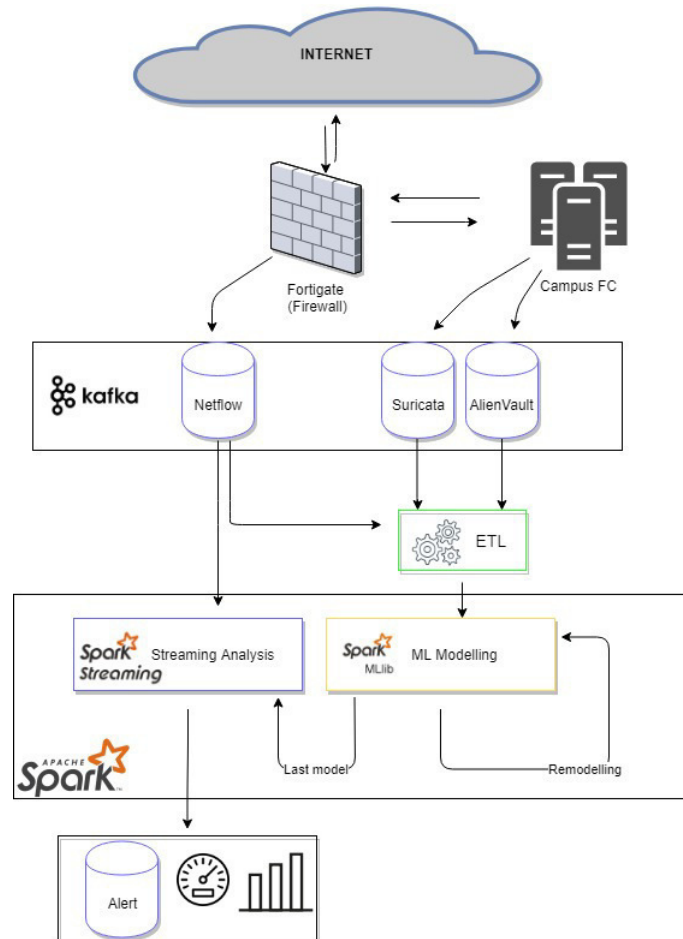


Figura 4.12: Asset tecnologico di Styx

4.4 Implementazione

L'implementazione di Styx si divide in più fasi che possono riguardare esclusivamente un componente o entrambi. Di seguito vengono spiegate nel dettaglio.

4.4.1 Raccolta dati, preprocessing e ricostruzione delle sessioni

La prima fase riguarda la raccolta dati necessaria per un'analisi preliminare delle sorgenti. Sono stati immagazzinati, a tal proposito, i dati generati dalle varie sorgenti nell'arco di una giornata infrasettimanale (da mezzanotte a mezzanotte). Da questi è stato possibile risalire ad una stima della quantità di dati e della frequenza con cui essi sono generati. Grazie all'informazione contenuta nell'intestazione dei vari record ricevuti, inoltre, è stato possibile estrarre informazioni utili anche in merito al traffico effettivamente scambiato. La figura 4.13 fa riferimento alla prima sorgente (cioè NetFlow). Quest'ultima lascia intuire come la frequenza di generazione dei record raggiunga anche picchi di 9 MB/s. Il grafico, infatti, indica sull'asse delle ascisse la componente temporale (in secondi) e sull'asse delle ordinate la quantità di dati ricevuta (in byte) di informazione sotto forma di header. Il

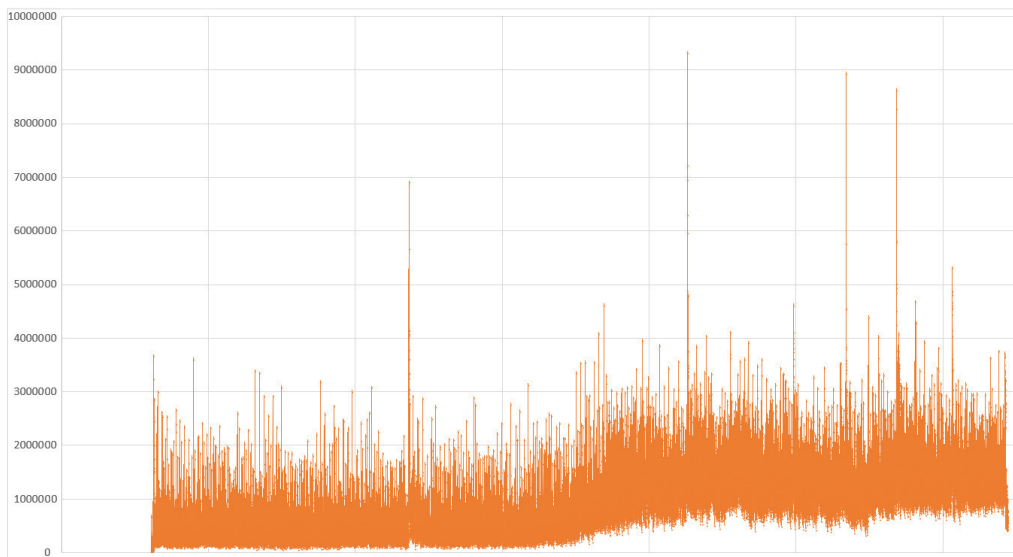


Figura 4.13: Stima della quantità (in byte) di header generati per la sorgente 1 - NetFlow

secondo grafico (in figura 4.14), analogamente, fa riferimento sempre alla stessa sorgente, soltanto che in questo caso non si parla di volume header ma di byte trasferiti complessivamente verso l'esterno o tra macchine interne. E' possibile notare in questo caso come l'intera banda sia spesso quasi totalmente occupata raggiungendo picchi di quasi 1 GBit/s (cioè circa 125MB/s). Il processo successivo alla raccolta di dati, effettuata tramite un

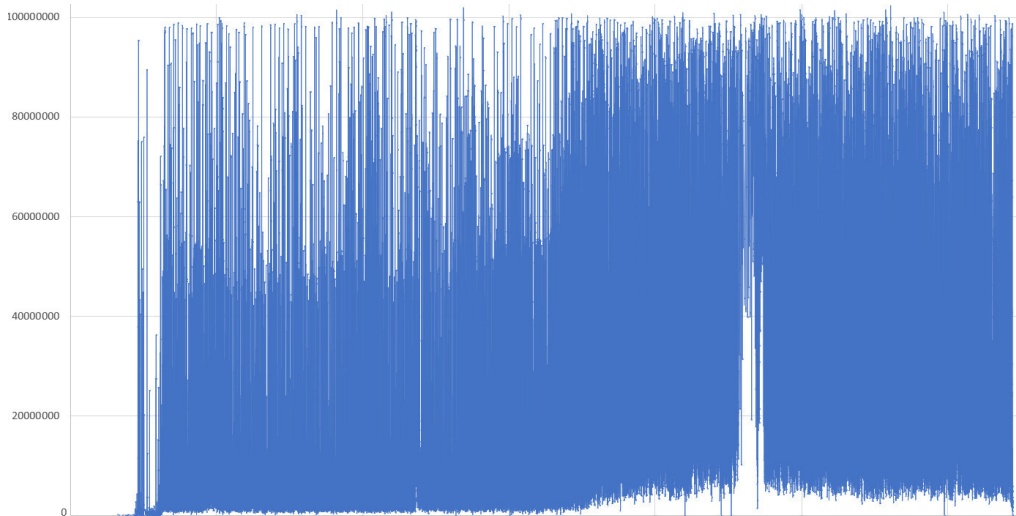


Figura 4.14: Stima della quantità (in byte) di dati scambiati dalla rete in base alle informazioni fornite da NetFlow

consumer-client Kafka, è stato quello di pre-processing dei dati. In riferimento alla prima sorgente il traffico ricevuto si presenta sotto forma di singola trasmissione appartenente ad una sessione di comunicazione non definita. E' stato necessario, a tal proposito, ricostruire le sessioni di comunicazione aggiungendo informazioni derivate dalla raccolta dei vari record univocamente collegati a quest'ultime. In figura 4.15 è possibile notare come ogni record appartenga univocamente ad una sessione, che a sua volta è formata da N record. Sono stati, inoltre, derivati tre nuovi attributi:

- *La durata*: intesa come la somma di tutte le singole durate.
- *Il traffico in ingresso*: inteso come i byte ricevuti.
- *Il traffico in uscita*: inteso come i byte trasmessi.

Ogni sessione è univocamente identificata da: IP interno, IP esterno (nel caso di comunicazione sono soltanto ordinati per grandezza), porta sorgente, porta destinazione e timestamp. Il timestamp è necessario per distinguere univocamente sessioni che possono ripetersi ed essere assimilate ad una soltanto. Questo tipo di preprocessing è direttamente effettuato a livello di streaming (componente DSP) e permette poi di effettuare controlli per l'identificazione degli scenari precedentemente elencati.

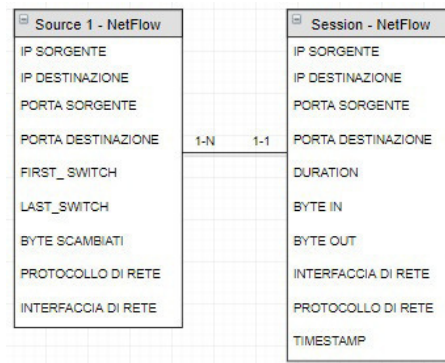


Figura 4.15: Ricostruzione delle sessioni di comunicazione sorgente 1 - NetFlow

4.4.2 Integrazione delle sorgenti e arricchimento

La fase di integrazione comprende le tre sorgenti principali più una fase di arricchimento che costituisce poi il tabellone finale, nonché dataset per l'elaborazione del componente ML. In figura 4.16 è riportato lo schema logico dell'integrazione delle tre sorgenti. Come si può evincere dalle cardinalità ad ogni sessione ricostruita è associato uno (in caso di positivi) o nessun alert di Suricata (in caso di negativi). Analogamente ad ogni record di Suricata è associato uno o nessun alert di Alien Vault. Al processo di integrazione è

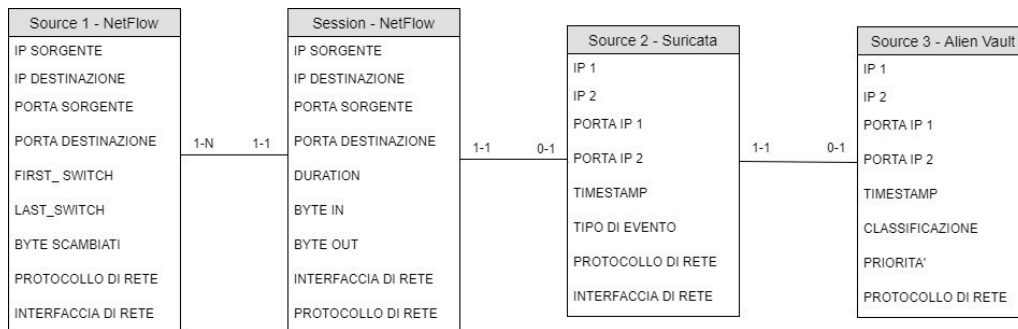


Figura 4.16: Schema logico per l'integrazione delle tre sorgenti Styx

stato anche incluso un processo di arricchimento dell'informazione sotto 3 aspetti principali (figura 4.17):

- Localizzazione: ogni IP address appartiene ad un range di indirizzi catalogati per

stato di appartenenza. Grazie all'utilizzo di Maxmind Geolite[73], un database costantemente aggiornato e provvisto di un servizio di API, è stato possibile risalire allo stato di appartenenza di ogni IP esterno in comunicazione con uno interno al fine di individuare possibili correlazioni tra anomalie e attributi di localizzazione (per esempio fuori-dentro UE, continente, etc).

- Tipo di porta: nelle reti i range di porta sono limitati a tre intervalli: le Well-known ports (da 0 a 1023), le porte registrate (1024 a 49151) e le porte dinamiche (chiamate anche porte private, da 49152 a 65535). Grazie a questa distinzione è possibile individuare possibili collegamenti tra le porte utilizzate e la criticità di una sessione (per esempio porte non associabili a servizi canonici come HTTPS, etc).
- Ruolo della macchina: grazie a delle tabelle fornite dal centro di sicurezza informatica d'Ateneo è stato possibile individuare il ruolo delle macchine, la propria criticità (in base alle risorse che gestiscono) e il sistema operativo. A tal merito macchine con un determinato ruolo che, per esempio, iniziano un'attività diversa sono sintomo di una possibile anomalia e/o una possibile intrusione. Analogamente, è possibile fare delle analisi su quali sistemi operativi sono maggiormente esposti a determinate minacce informatiche.

Ruolo	Localizzazione	TIPO-PORTA
HOSTNAME	INDIRIZZO IP	NUMERO PORTA
RUOLO	STATO	RANGE
CRITICITA'	CONTINENTE	
SISTEMA OPERATIVO	UNIONE_EUROPEA (S/N)	

Figura 4.17: Tabelle di arricchimento dell'informazione per il dataset finale

La tabella finale (figura 4.18), contenendo attributi categorici, il cui valore indica l'appartenza ad una categoria in un dominio finito, sono stati serializzati per agevolare il processo di machine learning con l'ausilio di tecniche di *indexing* di cui il framework è provvisto.

StyxRecord
IP INTERNO
IP EST. LOCALIZZATO
TIPO_PORTA SORG.
TIPO_PORTA DEST.
DURATION
BYTE IN
BYTE OUT
INTERFACCIA DI RETE
PROTOCOLLO DI RETE
TYPE
RUOLO
CRITICITA*
SISTEMA OPERATIVO
INDIRIZZO IP
STATO
UNIONE_EUROPEA (S/N)
CONTINENTE

Figura 4.18: Tabella dataset finale componente ML

4.4.3 Resampling e bilanciamento del dataset

Prima dell'utilizzo di algoritmi di ML messi a disposizione dalla piattaforma di analisi è stato eseguito, vista la copiosità del dataset di riferimento e il forte sbilanciamento tra le classi con cui erano etichettati i dati (rapporto positivi/negativi $1:100000$), un processo di resampling, in particolare di *undersampling*. Quest'ultimo consiste nel ridurre le occorrenze della classe presente in maggiore percentuale (quindi i negativi) rendendo possibile un'analisi complessiva e equilibrata del dataset. In generale, infatti, la presenza di un dataset sbilanciato durante una processo di machine learning può risultare un vero limite per le analisi. Un dataset sbilanciato può portare il sistema ad ignorare completamente la classe presente in minore percentuale e questo può portare a quello che viene definito in letteratura *accuracy paradox*[69]. Per risolvere il problema i possibili approcci possono essere schematizzati in sette strategie[70]:

1. *Raccogliere più dati*: banalmente un set di dati più grande potrebbe esporre una

prospettiva diversa e forse più equilibrata sulle classi.

2. *Cambiare metrica delle prestazioni*: la precisione non è la metrica da utilizzare quando si lavora con un set di dati sbilanciati. E' utile prendere in considerazione diverse misure di prestazione che possono fornire una visione più approfondita del modello rispetto alla tradizionale precisione di classificazione (per esempio l'indice di *recall*, o le *curve ROC*).
3. *Ricampionare il set di dati*: Questo cambiamento è chiamato campionamento del set di dati e ci sono due metodi principali che è possibile utilizzare per uniformare le classi. È possibile aggiungere copie di istanze dalla classe sottorappresentata denominata *sovracampionamento* (o *oversampling*) o eliminare le istanze dalla classe sovrarappresentata, chiamata *sottocampionamento* (o *undersampling*).
4. *Generare campioni sintetici*: un modo semplice per generare campioni sintetici è quello di campionare casualmente gli attributi da istanze nella classe di minoranza.
5. *Provare diversi algoritmi con diverse specifiche*: ogni problema ha un algoritmo più adatto e consigliato per l'analisi.
6. *Provare i modelli penalizzati*: usare gli stessi algoritmi, ma dare loro una prospettiva diversa sul problema. La classificazione penalizzata comporta un costo aggiuntivo sul modello per fare errori di classificazione sulla classe di minoranza durante l'allenamento. Queste sanzioni possono spingere il modello a prestare maggiore attenzione alla classe di minoranza (esempio *CostSensitiveClassifier*).
7. *Rimodellare il problema in istanze più piccole e gestibili*.

4.4.4 GDPR e anonimizzazione delle informazioni sensibili

È noto che il regolamento generale sulla protezione dei dati (GDPR) afferma che gli indirizzi IP dovrebbero essere trattati come dati personali perché possono essere utilizzati per individuare individui per trattamenti diversi, anche se non per identificarli effettivamente. Per condurre delle analisi di ogni genere infatti è necessario che questi ultimi siano

anonimizzati o richiesto il consenso attivo all'utente. Per ottemperare al regolamento gli indirizzi IP con cui lavora Styx sono già nella fase di preprocessing criptati mediante cifratura simmetrica a 256bit (Advanced Encryption Standard). La chiave è posseduta solo da personale autorizzato alla decifratura in caso di alert da esplorare.

4.5 Risultati

Per il componente DSP sono stati estratti numerosi report che fanno riferimento ad un intervallo di esecuzione di circa 1 giorno. Questi hanno permesso di individuare, inizialmente, due aspetti fondamentali rispetto al concetto di sessione "lunga" e sessione "pesante". Come si evince dal grafico in figura 4.19 la maggior parte delle sessioni presentano una durata complessiva minore di 10 secondi, seguite da una durata di circa 1500 secondi e 60 secondi, quindi meno di mezzora o di un minuto nella stragrande delle rimanenti. Queste statistiche sono state utili per individuare degli *outlier*, quindi delle situazioni in cui si può parlare di sessioni "lunghe". Come è possibile notare sul grafico una quantità molto ridotta di sessioni ha una durata di più di 1 giorno. Una durata così prolungata può essere attribuita ad una possibile intrusione (per esempio malware che trasmettono dati sensibili all'esterno o delle backdoor costantemente attive). Definire una soglia del genere ha reso possibile la creazione di report descrittivi che sono stati consegnati al centro di sicurezza informatica per un processo di validazione, il quale ha messo in risalto alcune situazioni da verificare manualmente da un amministratore di sistema.

Analogamente per il volume, è stato individuata (figura 4.20) una distribuzione tendenzialmente normale. La maggior parte delle sessione, infatti, non scambiano più di 100 KB, o comunque si aggirano sempre intorno a quella cifra. Questo è molto indicativo perché permette di individuare casi "insoliti" dove il traffico supera l'ordine del gigabyte, riconducibili a download illeciti, o in combinazione con altre condizioni, ad attacchi di tipo DDoS. La validazione dei seguenti casi è avvenuta allo stesso modo dei precedenti; sono state infatti individuate sessioni sospette stabilite con indirizzi di rete extra-europei identificabili come grandi download di file, con buona probabilità, non autorizzati o coperti da copyright, mediante reti P2P (per esempio torrent).

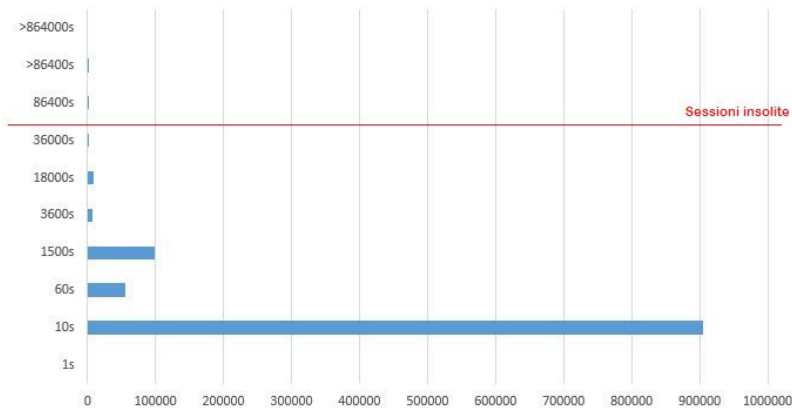


Figura 4.19: Distribuzione delle sessioni in base alla durata (in secondi)

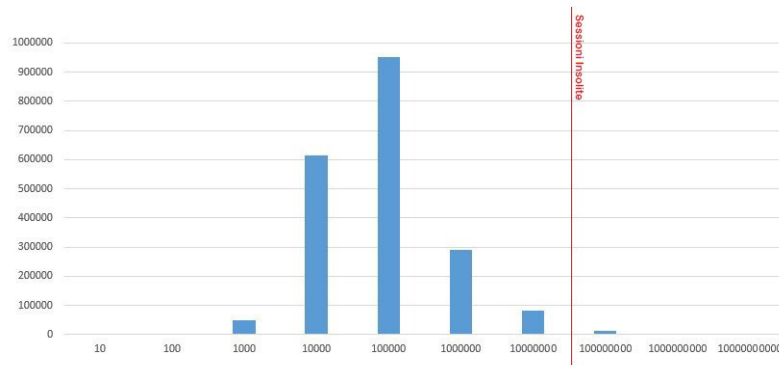


Figura 4.20: Distribuzione delle sessioni in base al volume di dati scambiato (in byte)

Inoltre, ad un livello di aggregazione più alto, in relazione ai due scenari di attacco presi in considerazione (port-scan e ddos), dai report generati sono emerse situazioni interessanti dove è stato necessario approfondirne la natura e i dettagli. In figura 4.21 si mostra una serie di record in cui situazioni che presentano un gran numero di connessioni verso un host della rete interna effettuate in tempi corti, possono essere ricondotte ad un tentativo di port-scan. In merito a ciò alcuni record più critici sono stati validati dal centro di sicurezza informatica in cui si è notato un continuo tentativo di intrusione in uno dei server DNS d'Ateneo. Un server DNS è un componente che permette la traduzione di indirizzi IP in hostname, quindi rende possibile la corretta navigazione in rete. La sua natura compromessa vista la sua funzionalità strategica può risultare un problema di grosso calibro in quanto comprometterebbe l'intero traffico di rete. Situazioni meno importanti

sono state ricondotte a tentativi di intrusione in webserver o simili dove una corretta configurazione del firewall può risolvere rapidamente la criticità. In ogni caso le segnalazioni hanno portato l'intero report ad essere punto di riferimento per approfondimenti sulla vulnerabilità di determinate macchine della rete prese di mira.

IP interno	IP esterno	PORTA	NUMERO CONNESSIONI	TEMPO CONNESSIONE
41e69da8b8596f0872b948a74a907c20	1990b9ed35183bae4e193b030ea0f16	10051	23954	1840
1990b9ed35183bae4e193b030ea0f164	95db4232e1f09ed092f7b7f61cf110c0	10051	11069	74
cc04efa08cc0a28bdf2a096996e42ec3	4ba3a438bb5127b91407cb321658704f	88	9145	46
51d1b01aefe278fb4703985b5520f825	275f5b6f854dc5d7069f77afd5ea8590	3052	6664	20

Figura 4.21: Sessioni insolite riconducibili a tentativi di port-scan

Allo stesso modo sono stati individuate quelle sessioni potenzialmente riconducibili a scenari DDoS, dove sono state individuate un gran numero di connessioni, anche da IP differenti (possibile botnet), su un'unica macchina in tempi ridotti e con un traffico trasmesso ingente. In questo caso i risultati sono stati molto limitati e rappresentati da alcuni casi la cui validazione non ha portato alert significativi o comunque non effettivamente diagnosticabili come attacchi di disservizio (per esempio è stato rilevato il server che distribuisce aggiornamenti OS a buona parte delle macchine della rete). Questo è motivato dal fatto che un attacco del genere, sebbene possa avvenire in ogni momento, è più raro che si verifichi. Impostare l'analisi sotto questo punto di vista può, tuttavia, preparare la rete ad una tempestiva rilevazione di questo tipo di attacchi, portando beneficio alla sicurezza complessiva.

Per quanto riguarda il componente ML, l'analisi ha portato risultati sostanzialmente molto descrittivi. Innanzitutto, partendo dal dataset di riferimento, arricchito dal processo di localizzazione, si denota una propensione dei positivi (possibili tentativi di intrusione) verso indirizzi, e quindi host, extra-europei. Il grafico in figura 4.22 mostra come il 64% dei positivi sia attribuito ad attaccanti non europei; di questi, in particolare, ben l'89% soltanto proveniente da indirizzi IP del continente asiatico e l'11% americano.

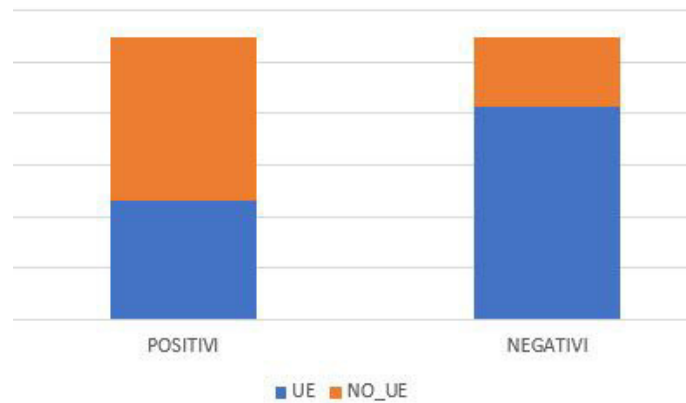


Figura 4.22: Statistiche in merito alla localizzazione delle sessioni analizzate e i tentativi di intrusione rilevati

Successivamente è stato fatto ricorso, come già accennato, a vari algoritmi di ML per l'estrazione di un modello decisionale composto da possibili regole atte a determinare la presenza di una possibile intrusione, quindi di un record positivo. Le tecniche supervisionate (random forest) hanno individuato uno sbilanciamento delle condizioni per la discriminazione del tipo di sessione (anomala o standard) costituendo percorsi decisionali che coinvolgono i seguenti attributi (figura 4.23):

- Localizzazione: come già detto, la gran parte degli tentativi di intrusione sono riconducibili ad indirizzi extra-europei. L'equivalente è emerso mediante l'estrazione di percorsi decisionali per mezzo dell'algoritmo random forest. Allo stesso modo per l'approccio non supervisionato dove si è notato che la definizione dei cluster in base ai criteri di somiglianza tra le sessioni, spesso coincide con questo attributo.

- Byte scambiati: una grande quantità di dati implica una possibile intrusione, o comunque una situazione poco usuale. Sono stati discretizzati, in base alle soglie definite precedentemente, le quantità di dati scambiati identificando tre classi di criticità in base al volume (standard, medio, alto). Anche in questo caso entrambi gli approcci hanno posto il fuoco su questo attributo.
- Porte di rete utilizzate: è stato verificato che la gran parte dei positivi utilizzi come porta di destinazione (quindi porta interna) servizi tradizionali come HTTPS/SSH, a discapito delle porte sorgenti dove l'attaccante può chiaramente scegliere il valore da utilizzare e cambiarlo continuamente per depistare i sistemi di sicurezza. Ad ogni modo questo permette di capire che la maggior parte dei tentativi di intrusione/attacco cercano "falle" nel sistema partendo dai servizi canonici (dns, server web, database), che dovrebbero essere quelli tenuti sotto maggiore monitoraggio.

I modelli risultati da un approccio non supervisionato, invece, sebbene abbiano individuato "cluster di sessioni" dove l'indice di somiglianza è molto orientato verso i medesimi valori sopra elencati, non hanno portato a risultati o ragionevoli sovrapposizioni con le etichette reali. Questo è probabilmente probabilmente attribuibile al gran numero di campi e alla varietà strutturale che i dati presentano. In ogni caso, i risultati complessivi sono preliminari e derivanti da un'analisi fatta su una porzione ridotta dei dati disponibili. Questa, tuttavia, riesce a fornire una visione di insieme orientando l'analisi su attributi significativi piuttosto che su informazioni poco utili a fini analitici.

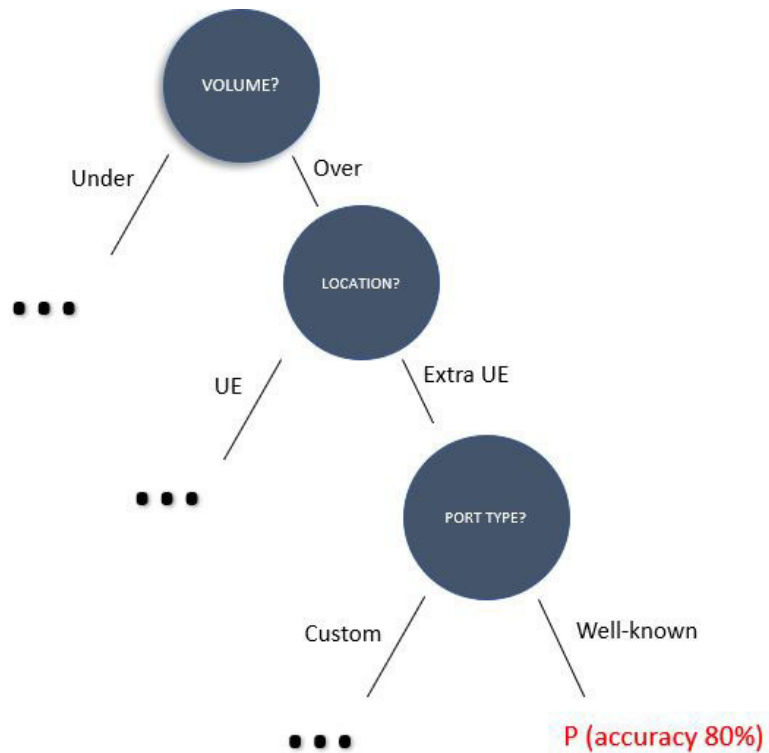


Figura 4.23: Generalizzazione di un percorso decisionale in Styx per l'individuazione di una possibile sessione sospetta. L'accuratezza del modello è di circa l'80%.

Conclusioni

Il fenomeno dei Big Data è ormai entrato in circolo nei processi business ed è sfida continua nel mondo aziendale cercare, e ottimizzare al massimo, sempre nuovi scenari di analisi. Il progetto di tesi si muove proprio in questa nuova concezione della data analysis, introducendo un nuovo mezzo per il supporto all'analisi e al monitoraggio di una rete informatica composta da migliaia di utenti.

Dopo una prima fase di progettazione e studio dello stato dell'arte, è stato rilasciato un primo prototipo conforme all'analisi dei requisiti che il dominio applicativo ha comportato. E' stato possibile far uso delle tecnologie principali nella realtà dell'analisi dati, dando modo di capire al meglio cosa vuol dire non solo progettare un sistema per l'elaborazione di dati complesso e distribuito, ma approfondirne una sua specializzazione (data stream processing) in chiave di sicurezza informatica e apprendimento automatico (machine learning). Il progetto, sebbene ancora in una fase preliminare, ha fornito fin da subito, in fase di test, risultati abbastanza soddisfacenti, soprattutto per il componente di streaming. Con il supporto del centro di sicurezza (ACCF), infatti, modellando i requisiti e implementando le conoscenze acquisite in fase di studio della letteratura si è arrivati a riscontri positivi con l'individuazione di modelli di rete (possibili attacchi, intrusioni) inizialmente solo teoricamente ipotizzati. Sicuramente l'applicazione, visti i tempi di sviluppo non troppo estesi, necessita ancora di rimodulazioni, azioni di manutenzione sia evolutiva che correttiva, fattorizzazioni e nuove valutazioni. La componente ML, in particolare richiede ancora molto approfondimento sia per quanto riguarda l'algoritmistica sia per la modellazione dei dati da utilizzare. Per come è stato ideato il progetto si muove in un dominio molto critico protagonista oggi nel panorama dell'informatica.

L'idea di far convergere due branche apparentemente concettualmente poco conciliabili come possono essere la sicurezza informatica e i sistemi di elaborazione di dati, lascia intendere, pertanto, l'utilità in ogni possibile dominio dei sistemi di Data Analytics. In sostanza, ovunque ci sia una quantità di informazioni mediamente elevata è probabilmente presente una possibilità di analisi ed estrapolazione di valore concreto per il committente.

Apache Spark, in aggiunta, si è rivelato un sistema per il real-time intuitivo, molto potente e ben congeniato. La sua facilità nell'uso permette di progettare in poco tempo sistemi complessi ma concettualmente ben definiti grazie anche ai numerosi moduli applicativi e alle numerose librerie per affrontare problemi di machine learning e analisi di stream. Il sistema porterà indubbiamente in futuro nuove evoluzioni estendendo l'attuale set di funzioni con nuovi algoritmi di ML, per esempio. Nel complesso la piattaforma, compatibile con l'ecosistema Hadoop, si è presentata dal primo momento affidabile e oltremodo performante rendendo possibile l'ottenimento di risultati dal primo momento sensati e, soprattutto, interessanti e attinenti ai fini stabiliti in partenza con il centro di sicurezza del campus.

Sono presenti, naturalmente, tanti aspetti che il prototipo per ora non comprende completamente; come per esempio la possibilità di individuare scenari di attacco diversi dai pochi presi in considerazione (DDos, sessioni lunghe, etc). In futuro, se il progetto avesse la possibilità di proseguire, si potrebbe espandere la struttura in modo da risolvere le problematiche che solo un'analisi dei requisiti più approfondita e dei tempi di sviluppo e progettazione più estesi possono far emergere. Ad oggi, la presentazione del primo prototipo può ritenersi, nel complesso, preliminarmente soddisfacente e già di supporto effettivo all'architettura di monitoraggio correntemente adottata dall'Ateneo.

Bibliografia

- [1] Pescatore F., *La Storia dei Database: le origini*, AppuntiDigitali.it, Luglio 2011
- [2] Gandomi A., Haider M., *Beyond the hype: Big data concepts, methods, and analytics*, International Journal of Information Management, December 2014
- [3] Torrey M., *The 5 Vs of Big Data*, Immoore-Oracle, October 2015.
- [4] White T., *Hadoop: The Definitive Guide* O'Reilly Media Inc., 2009.
- [5] Hadoop Official Documentation, *High-level Architecture of Hadoop*
<https://opensource.com/life/14/8/intro-apachehadoop-big-data>
- [6] Nardelli M., *Distributed Data Stream Processing summary* October 2015.
- [7] Conteh, N. Y., & Schmick, P.J., *Cybersecurity: risks, vulnerabilities and countermeasures to prevent social engineering attacks*. International Journal of Advanced Computer Research, 6(23), 31-38, 2016.
- [8] Zuech, R., Khoshgoftaar, T. M., Wald, R., *Intrusion detection and big heterogeneous data: a survey*. Journal of Big Data, 2(3), 1-41, 2015.
- [9] Kabiri, P., Ghorbani, A. A., *Research on intrusion detection and response: A survey.*, IJ Network Security, 1(2), 84-102, 2005.
- [10] Youssef, A., Emam, A., *Network intrusion detection using data mining and network behaviour analysis*, International Journal of Computer Science Information Technology, 3(6), 87-98.,2011.
- [11] Faisal, M. A., Aung, Z., Williams, J. R., Sanchez, A., *Securing advanced metering infrastructure using intrusion detection system with data stream mining* In Pacific-Asia Workshop on Intelligence and Security Informatics (pp. 96-111). Springer Berlin Heidelberg, May 2012.
- [12] Najafian, Z., Aghazarian, V., Hedayati, A., *Signature-Based Method and Stream Data Mining Technique Performance Evaluation for Security and Intrusion Detection in Advanced Metering Infrastructures (AMI).*, International Journal of Computer and Electrical Engineering, 7(2), 128-13, 2015.
- [13] Patel, A., Taghavi, M., Bakhtiyari, K., JúNior, J. C., *An intrusion detection and prevention system in cloud computing: A systematic review.*, Journal of

- network and computer applications, 36(1), 25-41, 2013.
- [14] Balasubramanian, R., Joseph, S.J.S.A., *Intrusion Detection on Highly Imbalanced Big Data using Tree Based Real Time Intrusion Detection System: Effects and Solutions.*, International Journal of Advanced Research in Computer and Communication Engineering, 5(2), 27-32, 2016.
- [15] Kicanaoglu, B. *Unsupervised Anomaly Detection in Unstructured Log-Data for Root-Cause-Analysis*, Master's Thesis, Computing and Electrical Engineering, Tampere University of Technology ,2015.
- [16] Parikh, D., Tirkha, P., *Data Mining Data Stream Mining – Open Source Tools.*, International Journal of Innovative Research in Science, Engineering and Technology, 2(10), 5234-5239, 2013.
- [17] Merelli, I., Pérez-Sánchez, H., Gesing, S. and D'Agostino, D., *Managing, Analyzing, and Integrating Big Data in Medical Bioinformatics: Open Problems and Future Perspectives*, BioMed Research International, Volume 2014, Article ID 134023, 1-13, 2014.
- [18] George, G, Haas, M.R., Pentland, A., *Big Data and Management*, Academy of Management Journal, 57(2): 321–326, 2014.
- [19] Zhang, D., *Granularities and Inconsistencies in Big Data Analysis*, International Journal of Software Engineering and Knowledge Engineering, 23(6): 887–893, 2013.
- [20] Lappas, T. and Pelechrinis, K., *Data Mining Techniques for (Network) Intrusion Detection Systems*, Technical Report, Department of Computer Science and Engineering, UC Riverside, Riverside CA 92521, May 10, 1-13., 2010.
- [21] Singh, J., Nene, M. J., *A survey on machine learning techniques for intrusion detection systems*, International Journal of Advanced Research in Computer and Communication Engineering, 2(11), 4349-4355.,2013.
- [22] Zuech, R., Khoshgoftaar, T. M., Wald, R., *Intrusion detection and big heterogeneous data: a survey*, Journal of Big Data, 2(1), 3., 2015.
- [23] Sremack J. *Big Data Forensics – Learning Hadoop Investigations*,

-
- [24] Stonebraker, M. and J. Hong. *Researchers Big Data Crisis; Understanding Design and Functionality*, Communications of the ACM, 55(2):10-11, 2012.
- [25] Colins M., *Network Security Through Data Analysis*
- [26] Denning DE, *An intrusion-detection model*, IEEE Trans Softw Eng 13(2):222–232. doi:10.1109/TSE.1987.232894, (1987).
- [27] Denning D, *An intrusion-detection model*, IEEE computer society Symposium on research security and privacy, pp 118–131, (1986).
- [28] Cárdenas, A. A., Manadhata, P. K., Rajan, S., *Big data analytics for security intelligence*, University of Texas at Dallas@ Cloud Security Alliance. 1-22, 2013.
- [29] Security information and event management (SIEM), *Wikipedia Official Page*, https://en.wikipedia.org/wiki/Security_information_and_event_management.
- [30] Pentaho, Hitachi Vantara, *Pentaho official website*, <https://www.hitachivantara.com/go/pentaho.html?source=pentaho-redirect>.
- [31] Apache Flume, *Apache Flume official website*, <https://flume.apache.org/>
- [32] Apache Kafka, *Apache Kafka official website*, <https://kafka.apache.org/>
- [33] Apache Spark, *Apache Spark official website*, <https://spark.apache.org/>
- [34] Matei Zaharia et al., *Spark: Cluster Computing with Working Sets*, 2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud’10, Boston, MA, USA, June 22, 2010.
- [35] Apache Mesos, *Apache Meson official website*, <https://mesos.apache.org/>
- [36] Apache Hive, *Apache Hive official website*, <https://hive.apache.org/>
- [37] Apache Parquet, *Apache Parquet official website*, <https://parquet.apache.org/>
- [38] Databricks, *Apache® Spark™ Survey 2016 Report*, <https://pages.databricks.com/2016-spark-survey.html>
- [39] *Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In Memory Cluster Computing*, http://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf
- [40] Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... Ghodsi, A., *Apache spark: a unified engine for big data processing*, Communications

- of the ACM, 59(11), 56-65, 2016.
- [41] *Data lineage definition*, https://en.wikipedia.org/wiki/Data_lineage
- [42] Chintapalli, S., Dagit, D., Evans, B., Farivar, R., Graves, T., Holderbaugh, M., ... Poulosky, P., *Benchmarking streaming computation engines: Storm, flink and spark streaming.*, In Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International (pp. 1789-1792). IEEE, 2016.
- [43] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... Xin, D., *Mllib: Machine learning in apache spark.*, The Journal of Machine Learning Research, 17(1), 1235-1241, 2016.
- [44] Bullock, J., Momeni, A., *ML lib: robust, cross-platform, open-source machine learning for max and pure data.*, In NIME (pp. 265-270), Maggio 2015.
- [45] Apache Spark MLlib , *MLlib official page*, <https://spark.apache.org/mllib/>
- [46] Pandas Library, *Official website*, <https://pandas.pydata.org/>
- [47] Scikit-learn Library, *Official website*, <http://scikit-learn.org/stable/>
- [48] Bradley J., Meng X., and Lee D., *Why you should use Spark for machine learning*, <https://www.infoworld.com/article/3031690/analytics/why-you-should-use-spark-for-machine-learning.html>
- [49] Pan, S., *The performance comparison of hadoop and spark.*, 2016,
- [50] *Fortigate Firewall official website*, <https://www.fortinet.com/products/next-generation-firewall.html>
- [51] *Suricata IDS official website*, <https://suricata-ids.org/>
- [52] *Alien Vault*, <https://www.alienvault.com/>
- [53] *Unified Threat Management*, https://en.wikipedia.org/wiki/Unified_threat_management
- [54] Zimmermann, H., *OSI reference model–The ISO model of architecture for open systems interconnection*, IEEE Transactions on communications, 28(4), 425-432, 1980.
- [55] Liu, Simon, Kuhn R., *Data loss prevention*, IT professional 12.2, 2010.
- [56] Douligieris, C., Mitrokotsa, A., *DDoS attacks and defense mechanisms: classification and state-of-the-art*, Computer Networks, 44(5), 643-666, 2004.

- [57] Lee, C. B., Roedel, C., Silenok, E., *Detection and characterization of port scan attacks*, University of California, Department of Computer Science and Engineering, 2003.
- [58] *IP address*, https://en.wikipedia.org/wiki/IP_address
- [59] *GDPR personal data definition*, <https://eugdprcompliant.com/personal-data/>
- [60] *GDPR: requisiti di pseudonimizzazione*, <https://www.infogdpr.eu/pseudonimizzazione-gdpr-58.html>
- [61] *GDPR: requisiti di pseudonimizzazione*, <https://www.infogdpr.eu/pseudonimizzazione-gdpr-58.html>
- [62] *TCP Protocol*, https://en.wikipedia.org/wiki/Transmission_Control_Protocol
- [63] *UDP Protocol*, https://en.wikipedia.org/wiki/User_Datagram_Protocol
- [64] *Privilege Escalation*, https://en.wikipedia.org/wiki/Privilege_escalation
- [65] *Backdoor*, [https://en.wikipedia.org/wiki/Backdoor_\(computing\)](https://en.wikipedia.org/wiki/Backdoor_(computing))
- [66] Knafo J. *What is Reverse SSH Port Forwarding*, <https://blog.devolutions.net/2017/3/what-is-reverse-ssh-port-forwarding>
- [67] Donges N., *The Random Forest Algorithm*, <https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffd>
- [68] Jain, A. K., *Data clustering: 50 years beyond K-means.*, Pattern recognition letters, 31(8), 651-666, 2010.
- [69] Afonja T., *Accuracy Paradox: what is it?*, <https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b>
- [70] Brownlee J., *8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset*, <https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/>
- [71] Bholowalia, P., Kumar, A. E., *BK-means: A clustering technique based on elbow method and k-means in WSN*, International Journal of Computer Applications, 105(9), 2014.
- [72] *Message-oriented middleware*, https://en.wikipedia.org/wiki/Message-oriented_middleware

- [73] MaxMind – GeoLite Database, *MaxMind official website*,
<https://www.maxmind.com/en/home>

Ringraziamenti

Ringrazio il Prof. Matteo Golfarelli e il Dott. Enrico Gallinucci che mi hanno seguito nel percorso di tesi ad ogni passo con grande professionalità, preparazione e pazienza. Un ringraziamento speciale va, inoltre, a tutto il team del BI Lab e al Dott. Ciro Barbone, responsabile del centro di sicurezza informatica del Campus, senza i quali l'intero progetto non sarebbe stato possibile.

Un grande grazie va al mio gruppo di amici e colleghi di Cesena e dintorni, il cui nome è talmente variabile che non ha senso riportarlo. Siete stati preziosi fin dall'inizio di questo lungo ma soddisfacente percorso, dimenticarvi sarà impossibile.

Gracias por todo ai miei cari *Desperados*, che hanno reso fantastico il mio periodo all'estero facendomi sentire a casa, anche se a chilometri di distanza dalla Madre Patria.

In particolare, a chi non ha mai smesso di stare *allmio* fianco: ni antes ni después.

Un grazie all'Università di Bologna, per avermi dato la possibilità di crescere e formarmi nella tranquillità più assoluta y también a la Universitat Politècnica de Catalunya, por la posibilidad de enriquecer mucho mi carrera universitaria: moltes gràcies!

Un ringraziamento generale, infine, è destinato alla mia famiglia e a chiunque abbia rasserenato e arricchito, anche soltanto in un'occasione, le mie giornate universitarie.