Università
Ca'Foscari
Venezia

Corso di Laurea magistrale in Informatica

Tesi di Laurea

Strumenti per il data mining di log ed il riconoscimento dei falsi allarmi in sistemi di rilevamento e prevenzione delle intrusioni

Ca' Foscari
Dorsoduro 3246
30123 Venezia

Relatore
Ch. Prof. Riccardo Focardi

Laureando
Marino Pagot
Matricola 808739

Anno Accademico
2011 / 2012

Abstract

Negli ultimi decenni, per proteggere i computer e le reti dall'aumento degli attacchi, sono emerse nuove tecnologie come i sistemi di rilevamento e prevenzione delle intrusioni, i quali gestiscono le insidie anche tramite la registrazione dei relativi allarmi in log. Quest'ultimi possono contenere enormi quantità di dati, perciò la base di questa Tesi di Laurea consiste nell'analisi di tali log, in modo da raggiungere gli obiettivi centrali quali l'estrazione delle informazioni più significative e la limitazione della pecca maggiore che contraddistingue queste tecnologie, ovvero la difficile distinzione tra i molto probabili falsi allarmi ed i veri attacchi. In questa Tesi di Laurea si introducono questi sistemi di rilevamento e prevenzione, descrivendone contesto, principi, peculiarità e funzionamento. Si considerano le caratteristiche dei dati presenti nei log, per capire quali sono le tecniche di mining più efficaci da applicare e l'importanza del riconoscimento delle false minacce. Inoltre, si implementano queste metodologie tramite lo sviluppo di applicativi, estendendole anche con tecniche proposte recentemente per l'individuazione dei falsi allarmi. Infine, si testa l'efficacia degli strumenti implementati sfruttando log significativi e analizzando i risultati ottenuti.

Sommario

Indice delle figure 5
Indice delle tabelle 7
1 Introduzione 9
1.1 Contenuti 10
1.2 Motivazioni ed obiettivi 11
1.3 Rilevamento e prevenzione delle intrusioni 15
2 IDPS: Descrizione, principi, peculiarità 23
2.1 Introduzione ai sistemi IDPS 23
2.1.1 Descrizione IDPS e relativa importanza 23
2.1.2 Utilizzi 24
2.2 Classificazione 26
2.2.1 Sorgente monitorata 26
2.2.2 Approccio all'analisi degli eventi 29
2.2.3 Tempo delle operazioni di riconoscimento 30
2.2.4 Modalità di reazione 31
2.3 Analisi degli IPS per combattere i falsi allarmi 32
2.3.1 Categorie 33
2.3.2 Utilizzi 35
2.4 Analisi degli NIDPS per combattere i falsi allarmi 36
2.4.1 Collocazione di un NIDPS 37
2.4.2 Il sistema NIDPS per eccellenza: Snort 40
3 Considerazioni preliminari per le tecniche di data mining 45
3.1 Proprietà dei dati di log di allarmi 45
3.2 Caratteristiche generali delle tecniche di data mining implementate 46
3.2.1 Introduzione e struttura generale 46
3.2.2 Mining specifico dei log 48
4 Mining degli itemset frequenti 51
4.1 Necessità di una nuova tecnica e formulazione iniziale del problema 51
4.2 Descrizione dettagliata dell'algoritmo 52
4.2.1 Mining dei singoli item frequenti 53
4.2.2 Creazione delle strutture per il mining 54
4.2.3 Creazione degli insiemi di dipendenza 56

4.2.4 Creazione dei prefissi di dipendenza 57
4.2.5 Creazione dell'albero delle transazioni 58
4.2.6 Creazione dell'albero dei pattern 61
4.2.7 Derivazione dei pattern 63
4.2.8 Gestione dei risultati 65
5 Mining mediante clustering 67
5.1 Necessità di una nuova tecnica e formulazione iniziale del problema 67
5.2 Problemi dei dati nei file di log 67
5.3 Descrizione dettagliata dell'algoritmo 68
5.3.1 Generazione delle 1-regioni dense 69
5.3.2 Generazione dei cluster candidati 70
5.3.3 Selezione dei cluster effettivi 73
5.3.4 Perfezionamento dei cluster 75
5.3.5 Gestione dei risultati 76
6 Tecniche per il rilevamento dei falsi allarmi 77
6.1 Prima metodologia di classificazione: "Itemset" 78
6.1.1 Prima versione 78
6.1.2 Seconda versione 78
6.1.3 Terza versione 80
6.2 Seconda metodologia di classificazione: "Cluster" 84
6.2.1 Prima versione 84
6.2.2 Seconda versione 84
6.2.3 Seconda versione 86
6.2.4 Terza versione 87
6.3 Gestione dei risultati 89
7 Analisi dei risultati 91
7.1 File di log utilizzati e comandi 91
7.2 Risultati legati alle tecniche di data mining 92
7.3 Risultati legati al rilevamento dei falsi allarmi 94
7.3.1 Configurazione adeguata delle metodologie 94
7.3.2 Sfruttamento dei casi di studio 96
7.3.3 Regole rilevate 97
7.3.4 Risultati ottenuti dalla classificazione 100
8 Conclusioni e considerazioni finali 107
9 Bibliografia 109

Indice delle figure

Figura 7.1 Parte finale dei risultati (con supporto del 2%) contenuti in “nomeSettimana_Itemset_Mining” 93
Figura 7.2 Parte finale dei risultati (con supporto del 2%) contenuti in “nomeSettimana_Cluster_Mining”. 93
Figura 7.3 Screenshot dei risultati finali riassuntivi ottenuti tramite la metodologia"Itemset" 101
Figura 7.4 Screenshot dei risultati finali riassuntivi ottenuti tramite la metodologia"Cluster" 102

Indice delle tabelle

Tabella 7.1 Confronto tra le due tecniche di data mining variando supporto/densità 92
Tabella 7.2 Risultati del clustering al variare del supporto e in modalità "unico cluster" 94
Tabella 7.3 Classi di attacco (e relative specifiche minacce) presenti nel dataset DARPA del 1999 95
Tabella 7.4 Verifica della bontà dei risultati ottenuti 96
Tabella 7.5 Regole relative alla classe di attacco attempted-recon (Attempted Information Leak) 97
Tabella 7.6 Regole relative alla classe di attacco web-application-activity (Access to a potentially vulnerable web application) 98
Tabella 7.7 Regole relative alla classe di attacco attempted-dos (Attempted Denial of Service) 98
Tabella 7.8 Regole relative alla classe di attacco attempted-admin (Attempted Administrator Privilege Gain) 99
Tabella 7.9 Regole relative alla classe di attacco web-application-attack (Web Application Attack) 99
Tabella 7.10 Regole relative alla classe di attacco misc-attack (Misc-Attack) 99
Tabella 7.11 Regole relative alla classe di attacco misc-activity (Misc-activity) 99
Tabella 7.12 Regole relative alla classe di attacco bad-unknown (Potentially Bad Traffic) 99
Tabella 7.13 Regole relative alla classe di attacco protocol-command-decode (Generic Protocol Command Decode) 100
Tabella 7.14 Regole relative alla classe di attacco attempted-user (Attempted User Privilege Gain) 100
Tabella 7.15 Regole relative alla classe di attacco non-standard-protocol (Detection of a non-standard protocol or event) 100
Tabella 7.16 Regole relative alla classe di attacco trojan-activity (A Network Trojan was detected) 100
Tabella 7.17 Regole relative alla classe di attacco default-login-attempt (Attempt to login by a default username and password) 100
Tabella 7.18 Confronto tra i risultati della classificazione rilevati dalle pubblicaz.scientif. e quelli relativi alle due metodologie 100
Tabella 7.19 Classificazione relativa alla classe di attacco attempted-recon 103
Tabella 7.20 Classificazione relativa alla classe di attacco web-application-activity 103
Tabella 7.21 Classificazione relativa alla classe di attacco attempted-dos 104
Tabella 7.22 Classificazione relativa alla classe di attacco attempted-admin 104
Tabella 7.23 Classificazione relativa alla classe di attacco web-application-attack 104
Tabella 7.24 Classificazione relativa alla classe di attacco misc-attack 105
Tabella 7.25 Classificazione relativa alla classe di attacco misc-activity 105
Tabella 7.26 Classificazione relativa alla classe di attacco bad-unlnown 105
Tabella 7.27 Classificazione relativa alla classe di attacco protocol-command-decode 105
Tabella 7.28 Classificazione relativa alla classe di attacco attempted-user 105
Tabella 7.29 Classificazione relativa alla classe di attacco non-standard-protocol 105
Tabella 7.30 Classificazione relativa alla classe di attacco trojan-activity 105
Tabella 7.31 Classificazione relativa alla classe di attacco default-login-attempt 106

1 Introduzione

L'aumento esponenziale della possibilità di connessione locale e globale di computer e reti e l'interdipendenza dei sistemi informatici, porta alla conseguente maggiore esposizione di questi stessi strumenti a possibili attacchi malevoli. Perciò, sorge la necessità di proteggere tali entità, e i dati in esse contenuti, da possibili minacce, che negli ultimi decenni sono in continuo aumento. Tale problema si evidenzia in particolar modo quando si tratta il caso Internet; in quanto rete globale, è accessibile da chiunque, consente di collegare qualsiasi individuo ad ogni altro, e questo può avvenire senza particolari restrizioni e protezioni. Di conseguenza, le possibilità di attacco aumentano in modo drastico, e perciò la necessità di sicurezza diventa ancora più evidente.

Una tecnologia che fornisce supporto per cercare una soluzione a questo bisogno di protezione è rappresentata dalla categoria dei sistemi di rilevamento e prevenzione delle intrusioni. I primi, contraddistinti dalla sola individuazione delle minacce vengono definiti IDS (Intrusion Detection System), mentre quelli che, a questa peculiarità, aggiungono la possibilità di reagire ad attacchi, vengono identificati come IPS (Intrusion Prevention System). Per trattare l'intera categoria generale di questi strumenti, in questa Tesi di Laurea viene scelto l'utilizzo della sigla abbreviata IDPS, in modo da racchiudere concettualmente entrambi le possibilità.

Per ciascuna tipologia, inoltre, il trattamento delle minacce può basarsi su due distinte possibilità. Un'analisi a livello di singolo sistema, dove vengono considerate le varie attività ed interazioni interne al sistema stesso; in questo caso gli strumenti vengono definiti HIDS (Host Intrusion Detection System) e HIPS (Host Prevention System) e possono essere identificati globalmente come HIDPS. La seconda opzione consiste in un'analisi a livello di rete (analizzando il traffico in transito); in questo caso vengono nominati NIDS (Network Intrusion Detection System) e NIPS (Network Intrusion Prevention System), mentre la categoria che li accorpa può essere definita NIDPS.

Nella Tesi di Laurea viene posta maggiore attenzione (soprattutto riguardo la parte applicativa) alla categoria specifica dei NIDPS, in quanto viene preso in considerazione il software Snort; si tratta infatti del sistema di rilevamento e prevenzione delle intrusioni a livello di rete maggiormente sofisticato, conosciuto ed utilizzato al mondo.

L'intera categoria dei sistemi di protezione IDPS è comunque caratterizzata dalla gestione delle insidie anche mediante la memorizzazione dei relativi allarmi generati in log. Questi log possono contenere quotidianamente notevoli quantità di allarmi che devono venire sottoposti ad una continua ed attenta revisione da parte di esperti. Inoltre, in questa enorme mole di dati, sono presenti per lo più allarmi che non consistono in minacce effettive, ovvero rappresentano dei falsi positivi che possono venire ignorati.

Di conseguenza, il primo obiettivo principale della Tesi è quello di fornire un metodo efficace per analizzare ed estrarre le informazioni importanti dai dati contenuti nei log, e questo lo si ottiene tramite l'applicazione di opportune tecniche di data mining. Il secondo scopo della Tesi è la naturale conseguenza di tali metodologie, le quali vengono infatti estese con tecniche proposte di recente in modo da porre un limite al problema inevitabile che contraddistingue questi sistemi di rilevamento e di prevenzione, ovvero distinguere efficacemente le vere minacce dai falsi allarmi e dagli eventi di scarsa importanza.

1.1 Contenuti

Questa Tesi di Laurea si articola fondamentalmente in due aree principali. La prima è costituita da un approfondito studio di ricerca sulla categoria dei sistemi di rilevamento e prevenzione delle intrusioni, ovvero gli IDPS; la seconda parte, invece, consiste nella descrizione della progettazione e dell'implementazione dei due applicativi realizzati in grado di mettere in pratica determinati approcci e raggiungere gli obiettivi principali della Tesi già accennati nell'introduzione.

Nella prima parte, costituita dal capitolo 2 ([1], [2], [3], [4]) vengono analizzati dettagliatamente i sistemi IDPS. Vengono inizialmente introdotti descrivendone i principi, le caratteristiche fondamentali ed il loro utilizzo. Successivamente, vengono classificati in base ai diversi aspetti ed approcci che li contraddistinguono e su cui si basa il loro funzionamento. Infine vengono analizzate determinate categorie che soffrono particolarmente del problema dei falsi allarmi. Tale capitolo presenta quindi lo scopo di ricavare una conoscenza ed una visione esaurienti delle tecnologie che producono i dati che verranno analizzati dettagliatamente in seguito.

La seconda parte è costituita da tre sezioni, ciascuna delle quali può essere considerata come la diretta conseguenza della precedente; le prime due sono accomunate dal fatto che il loro scopo è quello di descrivere come sono stati pensati, realizzati e come operano gli applicativi finali, mentre l'ultima consiste nell'analisi e nel confronto dei risultati ottenuti applicando gli stessi strumenti.

Nel capitolo 3 ([5]) vengono effettuate delle essenziali considerazioni preliminari per poter raggiungere il primo obiettivo di questa Tesi di Laurea: il data mining degli allarmi presenti nei log. Infatti, è opportuno effettuare una breve analisi delle caratteristiche dei dati memorizzati nei log, per capire quali sono le tecniche di data mining più efficaci da attuare per ottenere buone analisi.

Fatto ciò, si procede con la descrizione di come vengono messe in pratica tali metodologie per l'estrazione delle informazioni importanti dai dati presenti nei log. Ciò equivale alla descrizione di come avviene il data mining nei rispettivi due applicativi. Il capitolo 4 ([6]) affronta la prima tecnica, consistente nel mining degli itemset frequenti, mentre il capitolo 5 ([7]) quella del clustering.

Nel capitolo 6 si passa all'affrontare il secondo obiettivo su cui è incentrata la Tesi, ovvero il rilevamento dei falsi allarmi, il quale sfrutta il lavoro realizzato nel capitolo precedente, estendendolo ulteriormente. Vengono descritte come vengono implementate tali metodi; in particolare, la prima tecnica adotta esclusivamente il mining degli itemset frequenti ([8]) (definita "Itemset"), mentre la seconda anche quella del clustering ([9]) (definita "Cluster").

In definitiva, il primo applicativo realizzato attua il mining degli allarmi contenuti nei log utilizzando la prima tecnica di data mining studiata, e per il rilevamento dei falsi allarmi si basa sulla prima metodologia "Itemset" di questo capitolo. Per quanto riguarda il secondo applicativo, invece, l'estrazione della informazioni dai log avviene tramite la seconda tecnica di data mining, mentre per la distinzione tra falsi allarmi e quelli importanti sfrutta il secondo procedimento "Cluster" introdotto sempre in questo capitolo.

Nel capitolo 7 rientra la fase di test dell'efficacia degli strumenti implementati. Vengono introdotti i log molto significativi a disposizione e generati dal software Snort. Viene effettuato un confronto dei risultati ottenuti dai due applicativi riguardo le tecniche di data mining.

Successivamente viene descritto come viene raggiunta una configurazione adeguata delle due metodologie di classificazione e vengono tenuti in considerazione degli opportuni casi di studio. Infine viene svolta una comparazione significativa dei dati generati applicando le due diverse tecniche.

Infine, il capitolo 8 tratta le opportune considerazioni e conclusioni finali, comprendendo anche la possibilità di ulteriori sviluppi futuri.

1.2 Motivazioni ed obiettivi

Data mining degli allarmi nei log

Il principio di un IDPS consiste essenzialmente nel cercare di rilevare i segni di intrusione e limitarli prima che il danno sia fatto, un servizio sia reso vietato, o i dati vengano persi e questo avviene tramite diversi meccanismi. In generale, i sistemi configurati opportunamente generano log che tengono traccia dei servizi, degli utenti e dei dati; in questo caso mostrano le tracce di attività sospette.

Il punto di forza degli IDPS è la loro capacità di osservazione senza sosta dei pacchetti presenti sulla rete, della loro interpretazione in binario, e della notifica di quando qualcosa di sospetto sta accadendo. Un IDPS ben configurato si trasforma in una sorta di moltiplicatore di forze per i responsabili della sicurezza (amministratore di rete, di sistema, analisti e tutti quelli utenti addetti alla gestione del sistema o della rete) perché, in questo modo, riescono a monitorare molteplici dati da diversi sistemi.

La velocità nella rilevazione permette di generare gli allarmi e le reazioni in modo pressoché immediato e a qualsiasi ora, a differenza di quanto accadrebbe se, per tale scopo, venissero impiegati invece dei responsabili della sicurezza. Di conseguenza, il numero di eventi rilevati è sempre in continuo aumento: un IDPS è in grado di produrre gigabyte di dati (sottoforma di log) quotidianamente a causa dell'enorme aumento nella potenza di elaborazione dei computer e di conseguenza dell'analisi). A tutto questo si aggiunge anche la crescita esponenziale delle reti di comunicazione e delle velocità che le caratterizzano, fattori fondamentali che influiscono in tal senso.

Un IDPS crea (sfortunatamente ma inevitabilmente per sua natura, sebbene ormai utilizzato da un decennio) spesso e volentieri grandi quantità di allarmi, e la maggior parte di questi si rivelano essere falsi positivi oppure eventi di poca importanza. Vengono generati enormi moli di dati anche se l'IDPS è ben configurato e nel caso peggiore (nessuna configurazione iniziale) vengono prodotti talmente tanti dati che il tempo per l'analisi può diventare talmente elevato da rendere ormai inutile la risposta agli relativi attacchi. Essere veloci e precisi nell'individuare quando si è sotto attacco permette invece di prevenire o perlomeno mitigare potenziali compromissioni.

Di conseguenza, il lavoro del responsabile nel trattare gli effettivi allarmi con maggior attenzione viene reso più difficoltoso, perché può essere sommerso da questa enorme quantità di dati. Ciò gli Impedisce di individuare gli eventi che rappresentano un reale minaccia. L'analisi manuale del responsabile dei dati così come vengono prodotti dall'IDPS si rivela un compito pressoché impossibile da perseguire e può sopraffarlo.

Il problema fondamentale perciò consiste nell'impossibilità di analizzare normalmente questi allarmi che spesso contengono molte più informazioni di quelle a cui è veramente interessato un responsabile della

sicurezza. La revisione dei log però è inevitabile: deve essere considerata un procedimento basilare e una prima linea di difesa in termini del rilevamento delle intrusioni, che va fatta regolarmente (altrimenti genera più log che possono venire facilmente ignorati) perché porta grandi benefici riguardo la conoscenza di come il sistema funziona, della sua salute e del suo corretto funzionamento.

Quindi, per rimediare a queste diverse problematiche, devono essere applicate delle tecniche in modo da rendere veloce l'elaborazione degli allarmi degli IDPS. La soluzione naturale, che viene adottata in questa Tesi di Laurea, è quella di attuare dei metodi basati su tecniche di data mining per facilitare al responsabile della sicurezza il task della revisione nell'estrazione efficiente di conoscenza dai log di eventi. Tali log vengono infatti elaborati con l'obiettivo di ottenere e scoprire in modo immediato e preciso nuove regolarità (e irregolarità), i cosiddetti pattern, altrimenti difficilmente visibili e magari fino a quel momento sconosciuti. I pattern, siccome mostrano gli elementi comuni che si riscontrano all'interno di un log, possono facilmente rivelare i falsi allarmi o quelli irrilevanti celati nella marea di dati.

Le informazioni risultanti vengono sfruttate dal responsabile della sicurezza per avere immediatamente una visione concisa delle caratteristiche più comuni degli allarmi che si verificano. In base a queste può adottare delle decisioni, come l'apporto di modifiche alla configurazione dell'IDPS o all'elemento monitorato, la gestione di problematiche di sicurezza e la definizione di regole di correlazione e filtraggio per i nuovi allarmi.

Rilevamento dei falsi allarmi

L'applicazione delle tecniche di data mining permette di ottenere una sorta di resoconto riassuntivo immediato e conciso degli aspetti comuni riguardanti l'enorme quantità di allarmi nei log. Quindi, già di per se, tale procedimento apporta un notevole aiuto al responsabile della sicurezza nella gestione e amministrazione di un sistema o di una rete.

Tuttavia, una volta ottenute queste nuove informazioni, devono venire a loro volta analizzate in maniera dettagliata; il responsabile deve saper interpretare i risultati in modo da prendere delle determinate decisioni per migliorare il controllo sul funzionamento del sistema IDPS e sulla sicurezza dell'elemento in analisi. Uno di questi miglioramenti consiste nel perfezionare la configurazione del sistema IDPS affinché nell'immediato futuro questo riesca a ridurre gli allarmi che produce; ciò avviene eliminando progressivamente la quantità di falsi allarmi (prodotti dalle rispettive regole all'interno del sistema IDPS) che consistono, molto probabilmente, nella maggior parte degli allarmi totali che un IDPS genera.

Il problema dei falsi allarmi, unito a quello degli allarmi effettivi invece non rilevati, è inevitabile, consiste in una sorta di piaga che sfortunatamente caratterizza tutti gli IDPS, anche quando questi vengono sottoposti ad una laboriosa regolazione per il sistema o rete specifici a cui sono stati applicati. Ciò può manifestarsi in modo ancora più evidente, in termine di danni che può provocare, quando si ha a che fare in modo specifico, come vedremo poi, con gli IPS.

Innanzitutto è necessario dare la definizione di cosa sia un falso positivo e un falso negativo. Un falso positivo consiste nell'inappropriata generazione di un evento o di un allarme in risposta ad un'identificazione errata di un tentativo di intrusione o attacco. Un falso negativo, invece, indica la mancata produzione di un evento o di un allarme quando un'intrusione avviene effettivamente, ovvero si sta verificando nell'elemento monitorato.

E’ opportuno specificare che il termine falso positivo è sinonimo di falso allarme, e perciò in questa Tesi viene utilizzato preferibilmente quest’ultimo. Inoltre, la Tesi si focalizza esclusivamente su questo tipologia di allarmi, perché le tecniche applicate si basano appunto sulla loro identificazione, tralasciando invece i falsi negativi. Anch’essi comunque si rivelano un aspetto importante per quanto riguarda gli IDPS: rimanere ignari della presenza di una minaccia quando invece questa si verifica effettivamente rappresenta un alto livello di esposizione ai pericoli.

Il problema dei falsi allarmi assume diversa rilevanza in base alla tipologia di IDPS che si utilizza.

Gli IDPS di tipo IDS sono completamente passivi: dopo aver rilevato la minaccia, non viene intrapresa nessuna azione per modificare le caratteristiche delle interazioni a livello di sistema o del traffico di rete. Il loro scopo esclusivo è quello di avvertire il responsabile della sicurezza quando avviene un attacco, senza però cercare di mitigarlo o fermarlo totalmente. In questo caso, la presenza dei falsi positivi porta con se diversi aspetti negativi:

- crea grande carico di lavoro non necessario per il responsabile della sicurezza. Deve infatti analizzarli e all’occorrenza definire degli opportuni filtri. Perciò può dedicare minor tempo e attenzione all’analisi approfondita degli allarmi che in teoria invece rilevanti
- rende un IDPS inutile se la quantità di dati è troppo grande: per una rete estesa, tale strumento può creare quotidianamente anche un milione di allarmi. Quindi è difficile distinguere in questa marea di allarmi quelli effettivi (corrispondenti ad attacchi o compromissioni che hanno avuto successo) dal resto
- se una minaccia relativamente inoffensiva inizia a propagarsi nella rete, un IDPS può generare un allarme per ogni comunicazione, nascondendo eventi invece quelli veramente pericolosi
- attività assolutamente benigne e legittime possono esibire alcune stesse caratteristiche di azioni malevole e quindi essere trattate erroneamente come tali. Se non si conosce alla perfezione l’elemento monitorato e l’ambiente circostante (cosa che non si può ottenere praticamente causa l’estrema complessità delle interazioni all’interno di un sistema o di una rete), si va facilmente incontro ad identificazioni e interpretazioni errate. Traffico criptato può generare molti falsi positivi in modo anche casuale a causa della parte di dati (payload) criptato
- possono venire volutamente creati da un attaccante per nascondere tentativi veri di compromissione, deviando quindi l’attenzione
- Possono essere collegati a delle minacce che inoffensive per l’elemento monitorato perché sono legate a caratteristiche no presenti nello stesso elemento e non si è vulnerabili da quel punto di vista
- da un punto di vista pratico a livello business, possono indurre ad optare per strumenti alternativi, in quanto le aziende solitamente mettono a disposizione risorse limitate per la sicurezza; se il costo di realizzazione e di gestione di un IDPS è molto più grande rispetto al valore delle informazioni che devono essere protette, è opportuno riconsiderare la strategia per la sicurezza

La situazione si complica drasticamente quando si considerano gli IDPS di tipo IPS caratterizzati da capacità più sofisticate. Invece di essere attori passivi nel sistema o nella rete sotto esame, questi hanno la facoltà di agire automaticamente per limitare le interazioni dei processi all’interno di un sistema o agire sul traffico di rete. Impiegare un IPS non comporta quindi semplicemente ricevere enormi quantità di allarmi, bensì permette anche di controllare e fermare classi di attacchi che possono essere bloccati solo da dispositivi ben equipaggiati. Perciò è richiesta la massima attenzione (mirata alla perfezione) sui dati che produce un IDPS.

Per prevenire un attacco, questo deve essere prima rilevato e, di conseguenza, i meccanismi per il rilevamento sono presi in prestito dagli IDS, e quindi sono soggetti alle stesse limitazioni riguardo la generazione dei falsi positivi. In quel caso gli aspetti negativi riguardavano essenzialmente l'alto carico di lavoro per il responsabile che deve controllare gli allarmi perché non possono essere implicitamente verificati come validi. In questo caso invece è l'IPS che essenzialmente controlla le minacce da lui rilevate, e, in base alla sua configurazione, mette in pratica le relative contromisure. La presenza di falsi allarmi può quindi alterare o addirittura bloccare direttamente e inevitabilmente attività legittime, e questo deve essere assolutamente evitato. Di conseguenza, il rilevamento dei falsi allarmi per cercare di ridurne drasticamente il numero assume quindi una rilevanza maggiore rispetto al caso degli IDS.

Un IPS non può essere configurato in modo da essere libero nel reagire in modo arbitrario agli allarmi, specialmente quelli che invece sono legittimi. L'obiettivo è quello minimizzarli scegliendo un accettabile livello di rischio, un giusto compromesso, perché purtroppo la soluzione perfetta non esiste. La soluzione migliore consiste nel regolare e limitare le possibilità delle azioni e delle risposte ad un piccolo insieme ben definito di attività malevoli, ovvero utilizzare le contromisure solo per eventi per i quali si ha alta confidenza per il fatto che non si generino falsi positivi e dove c'è l'evidente necessità di prevenire il tentativo di attacco dall'essere portato a compimento.

Per limitare la quantità di falsi allarmi, sia per quanto riguarda gli IDS che per gli IPS quindi è imprescindibile adottare per lungo tempo un'intensa, precisa, attenta, granulare e incrementale regolazione del sistema IDPS, tenendo conto delle caratteristiche, potenzialità e regole relative a tale sistema. Deve essere svolto un lavoro significativo sulla gestione, orientato ai dettagli e che richiede tempo: una volta installato e configurato il sistema deve essere continuamente gestito, soprattutto perché l'ambiente in cui opera è in continuo aggiornamento. Il sistema IDSP adotta spesso delle firme relative alle minacce per la loro rilevazione; tali firme devono subire un aggiornamento periodico per combattere le inside di nuova generazione.

Se non si adottano queste accortezze, il valore dell'impiego di un IDPS diminuisce sensibilmente; il procedimento di configurazione e regolazione del sistema deve essere supportato quindi da un'efficiente fase di rilevamento dei falsi allarmi. Il problema è decidere il grado di specificità dell'IDPS nel rilevare i falsi allarmi. Una scelta estrema consiste nel marchiare ogni attività come sospetta , ma non è realistico: non verrebbe tralasciato nessun attacco potenziale, ma il flusso di falsi allarmi sarebbe opprimente. Un'opzione totalmente opposta può essere quella di ignorare la maggior parte degli allarmi e delle potenzialità di un IDPS: la quantità di falsi allarmi sarebbe minima, verrebbero ignorati molti pericoli effettivi.

La fase di rilevamento dei falsi allarmi perciò deve venire assolutamente applicata ed in modo efficiente, ma se viene svolta esclusivamente in modalità manuale richiede notevoli risorse in termini di tempo, elaborazione e impiego di personale esperto. La procedura può invece essere applicata in maniera automatica; la soluzione che viene adottata in questa Tesi è infatti quella di sfruttare le tecniche di data mining accennate precedentemente estendendole con delle metodologie che, interfacciandosi con le precedenti, riescono ad applicare l'automazione dell'analisi degli allarmi e la loro conseguente classificazione. Lo scopo è quello di ottenere come risultato finale una significativa ed accurata distinzione tra i falsi allarmi e di poca importanza, e quelli invece veramente rilevanti.

I risultati ottenuti dalla classificazione forniscono un importante aiuto al responsabile, soprattutto perché:

- la quantità di allarmi rilevati viene sensibilmente ridotta, perché la maggior parte degli allarmi consiste in falsi allarmi e di poca importanza, e quindi viene ridotto il carico di lavoro della revisione.
- Può prendere delle decisioni riguardo quali regole generanti i falsi positivi possono essere effettivamente eliminate o meno, e riguardo le eventuali modifiche da apportare ai differenti aspetti della sicurezza nell'elemento monitorato e nell'ambiente circostante.

Il responsabile deve comunque osservare attentamente i risultati ottenuti dalla classificazione in quanto ogni allarme, ogni regola, ogni log (l'appropriata configurazione di un IDPS) è altamente dipendente sia dalle caratteristiche del sistema o della rete monitorati che dalle aspettative del responsabile della sicurezza che deve attuare delle azioni in base agli allarmi rilevati.

1.3 Rilevamento e prevenzione delle intrusioni

In questa sezione viene descritto dove gli argomenti della Tesi possono trovare applicazione. Il contesto in questione è da ricercarsi naturalmente nell'ambito dei sistemi IDPS; tali sistemi vengono utilizzati per migliorare la sicurezza informatica, e, per tal motivo, è opportuno descrivere il loro ruolo all'interno di tale sicurezza. Per prima cosa, viene trattato la loro collocazione in termini generali, per poi passare ad una visione più specifica quale la sicurezza multistrato.

Inserimento nella sicurezza informatica globale

Innanzitutto è opportuno definire quando un sistema informatico (sistema singolo o rete) può considerarsi sicuro da eventuali minacce ed intrusioni, perciò bisogna specificare cosa s'intende per sicurezza, in particolare di sicurezza informatica.

E' opportuno comunque ricordare che in tutti le possibili circostanze, la sicurezza non deve essere considerata un prodotto, bensì un processo, in quanto non esiste una soluzione già preconfezionata adatta a tutte le esigenze, ma devono essere valutati di volta in volta svariate sfaccettature. Un aspetto importante è quello di definire dei concetti fondamentali per una buona politica di sicurezza:

- qualsiasi definizione di sicurezza è una visione idealistica, perché, sebbene vengano rispettate tutte le diverse misure e raccomandazioni per proteggere qualsiasi computer ad uso domestico oppure aziendale, risulta impossibile ottenere un sistema informatico perfettamente invulnerabile
- è possibile cercare di avvicinarsi all'obiettivo di una sicurezza perfetta tenendo presente però che possono subentrare degli aspetti negativi quali la creazione di forti disagi per l'utente, l'impiego di notevoli risorse finanziarie, e la nascita di sistemi altamente isolati che possono facilmente ridurre la produttività
- è necessario che vi sia un equilibrio tra la possibilità di proteggere le informazioni particolarmente sensibili dall'essere lette, modificate o persino eliminate da utenti non autorizzati, e quella di conservare comunque una certa flessibilità, ovvero mantenere una certa facilità d'utilizzo delle risorse a disposizione

Per tutti questi motivi, è ragionevole prendere in esame anticipatamente tutti gli aspetti necessari per ottenere, come risultato finale, un soddisfacente grado di sicurezza. Infatti, l'applicazione di buone regole di sicurezza, unita ad un'oculata analisi dei rischi, rende il sistema o la rete più resistenti alle intrusioni.

Ritornando alla definizione di sicurezza informatica, viene utilizzata una definizione in termini dell'operatività del sistema informatico, ovvero può essere definito sicuro solo se rispetta questi tre principi fondamentali:

- Confidenzialità
- Integrità
- Disponibilità

La *confidenzialità* assicura che le risorse ed i dati presenti all'interno di un sistema informatico possono essere acceduti esclusivamente dalle persone autorizzate, ovvero solo da chi ha il diritto di fruirne a seconda delle modalità stabilite da una certa politica di sicurezza. In questa circostanza rientra anche "l'oscuramento" dell'esistenza di particolari informazioni. Per ottenere tutto ciò, si ricorre, nella maggior parte dei casi, all'utilizzo di tecniche di crittografia per cifrare e decifrare dati.

In questo caso, quando la minaccia riesce a compromettere questa peculiarità della sicurezza informatica, significa che è riuscita purtroppo ad ottenere l'accesso ad informazioni confidenziali.

L'*integrità* garantisce che le risorse ed i dati contenuti in un sistema informatico possono venire modificati solo dagli utenti autorizzate, e soltanto entro le limitazioni a loro assegnate. I dati devono quindi risultare corretti e validi sia dal punto di vista del contenuto (le operazioni di modifica, se autorizzate, devono essere consistenti ed avvenire solo tramite procedimenti adatti) che della loro origine (la modalità del loro ottenimento e la correttezza della fonte).

Per realizzare questa importante caratteristica, vengono utilizzate tecniche di crittografica per calcolare la firma collegata ad un certa entità per un certo file e per un certo tempo, oppure tecniche di identificazione ed autenticazione, come ad esempio il controllo degli accessi.

In questa circostanza, l'intrusione riesce a violare questo aspetto legato alla sicurezza informatica quando avviene con successo la modifica delle informazioni confidenziali o l'alterazione degli aspetti legati alla loro origine.

La *disponibilità* assicura che i servizi, le risorse ed i dati fruibili da un sistema informatico siano accessibili ed utilizzabili da utenti autorizzati, e in tempi e modi soddisfacenti. Le misure che possono essere adottate per mettere in pratica ciò consistono principalmente nell'utilizzo di sistemi operativi e applicativi stabili e di buona qualità, e nell'adozione di adeguate politiche di backup.

In questo caso, quando la minaccia riesce a compromettere questa caratteristica, significa che è riuscita purtroppo a trasformare in inutilizzabili alcune risorse, rendendo il sistema bloccato e quindi non fruibile secondo il suo usuale funzionamento.

A questo punto, può essere data una definizione di minaccia, o meglio ancora di intrusione, in quanto è il termine che caratterizza la descrizione dei sistemi IDPS (**Intrusion** Detection/Prevention System):

Un intrusione è qualsiasi genere di operazione che riesce a compromettere uno dei tre aspetti su cui si basa il concetto di sicurezza informatica, ovvero la confidenzialità, l'integrità e/o la disponibilità di alcuni dati e/o risorse.

Per riuscire a soddisfare i tre principali concetti che contraddistinguono la sicurezza informatica (la disponibilità può essere evitata perché riguardante in particolar modo l'aspetto strutturale), il sistema

informatico, a sua volta, deve riuscire a fornire diversi servizi; in particolare, questi possono essere suddivisi in tre classi funzionali fondamentali (nella maggior parte dei casi vengono realizzate a livello di sistema operativo, ma esistono anche applicativi che sono in grado di adottarle):

- Identificazione
- Autenticazione
- Autorizzazione

L'*identificazione* consiste in servizi con lo scopo di identificare l'utente quale entità nel sistema; vengono sfruttati sia per l'accesso al sistema da parte dell'utente che per l'associazione di permessi e privilegi utilizzati dai successivi servizi (ad esempio l'associazione utente-username). In definitiva, tali servizi vengono utilizzati per rispondere alla domanda: "Chi sei?".

L'*autenticazione* racchiude servizi per validare l'identità di chi vuole accedere al sistema, e quindi se rappresenta effettivamente un utente autorizzato. (ad esempio l'associazione username-password con uso di crittografia o chiavi asimmetriche). Questi servizi vengono sfruttati per rispondere alla domanda: "Sei veramente tu?".

L'*autorizzazione* rappresenta l'insieme di servizi per definire le azioni permesse a ciascun utente quale entità nel sistema. Questo aspetto spesso è collegato al meccanismo del controllo degli accessi.

Il meccanismo basilare è definito DAC (Discrectionary Access Control), ed è caratterizzato dal fatto che il proprietario di un file può decidere le modalità di accesso allo stesso file da parte degli altri utenti. Purtroppo però, il proprietario non ha nessuna restrizione su quel file, e quindi non si possono limitare le operazioni allo stesso.

Più avanzato è il meccanismo MAC (Mandatory Access Control), in cui non solo vi è il controllo dell'accesso al contenuto dei file, ma anche delle operazioni specifiche che possono venire eseguite.

Inoltre è presente un altro meccanismo, l'RBAC (Role-Based Access Control) dove i permessi e i privilegi non vengono assegnati direttamente a ciascun utente, ma attraverso dei "ruoli" creati appositamente e assegnati quindi ad un particolare insieme di utenti.

Tutti e tre i meccanismi, e quindi la classe dei servizi relativi all'autorizzazione, rispondono idealmente alla domanda: "Cosa sei in grado di fare?".

Tenendo presente la definizione data prima di sicurezza informatica, basata sul soddisfare i tre principi definiti precedentemente, quest'ultimi possono essere raggiunti sfruttando le classi di servizi appena descritte. Queste, in definitiva, vengono utilizzate per definire un'opportuna relazione tra utenti e risorse del sistema attraverso il meccanismo dell'identificazione (per l'accesso degli utenti), dell'identificazione (per la validazione dell'identità) e quello dell'autorizzazione (per la concessione agli utenti di determinate operazioni), chiamato anche controllo degli accessi.

Detto questo, può essere data una nuova definizione di intrusione:

Un'intrusione consiste nello sfruttare la debolezza o le pecche nell'architettura del sistema informatico operando come il normale utente dello stesso sistema, bypassando i meccanismi di identificazione, autenticazione e/o autorizzazione.

Come già detto, garantire la sicurezza è ancora più problematico quando si ha a che fare con sistemi in rete, ed in particolare Internet, dove, a causa della presenza del protocollo fondamentale IP, nella maggior parte dei casi non esistono meccanismi per l'accesso, per la validazione e per il controllo degli accessi. Tecniche come IPSEC, VPN, SSL riescono ad utilizzare tali meccanismi, ma questo avviene con particolari difficoltà in termini di complessità di utilizzo e di gestione.

A questo punto entrano in scena i sistemi di rilevamento e prevenzione delle intrusioni, ovvero gli IDPS, in quanto riescono a oltrepassare i limiti imposti dalla sicurezza basata su questi meccanismi; essi infatti si basano sul fatto che l'intrusione, o violazione della sicurezza, avviene quando si verifica un comportamento anomalo del sistema. L'efficacia del funzionamento di questi strumenti sta proprio nel saper individuare tutti quelle azioni e utilizzi differenti certamente da quelli legittimi e rientranti nella normalità, e che possono essere facilmente sintomo di un'eventuale intrusione. Per tal motivo, questi strumenti devono essere sostenuti da una politica di sicurezza adeguata, attenta e caratterizzata da una gestione in continua evoluzione.

Quindi se il meccanismo dell'identificazione è equivalente a porsi la domanda "Chi sei?", quello dell'identificazione a "Sei veramente tu?", quello dell'autorizzazione a "Cosa sei in grado di fare?", gli IDPS concettualmente rispondono alla questione "Perché stai facendo questo?".

Gli IDPS sono quindi strumenti molto utili in quanto non sono sostitutivi ai precedenti meccanismi, bensì complementari, perché riescono a controllarne l'efficacia ed individuare un eventuale loro fallimento.

Per concludere questa sezione, è opportuno dare una definizione di quale sia il compito principale di questi sistemi, ovvero il rilevamento delle intrusioni:

Il rilevamento dell'intrusione (Intrusion Detection) consiste nel processo di monitoraggio degli eventi che si verificano all'interno di un sistema o di una rete, con lo scopo di analizzarli ed individuare ed isolare possibili problemi di sicurezza. In maggior dettaglio, ogni evento rappresenta un'intrusione intenzionale non voluta e non autorizzata da parte di un computer a risorse computerizzate quali un singolo sistema oppure una rete. Questo accesso non autorizzato consiste nella volontà di compromettere, o altrimenti danneggiare, i dispositivi vittima della minaccia.

Inserimento nella sicurezza informatica multistrato

Gli IDPS possono venire collocati adeguatamente anche all'interno di un altro concetto di sicurezza informatica, ovvero la sicurezza informatica multistrato. Questa tipologia di sicurezza ai tempi odierni è sempre più richiesta, soprattutto nell'ambito dei sistemi informativi per contrastare il continuo aumento delle insidie a livello informatico.

Con il trascorrere del tempo, gli attacchi di tipo digitale sono sempre in continuo aumento ed evoluzione. Per tal motivo, i metodi classici di sicurezza non riescono a garantire un sufficiente standard di protezione. Perciò è fondamentale progettare un sistema informatico che rispecchi pari pari le varie misure di sicurezza che vengono adottate, e il concetto di sicurezza multistrato ne è la naturale conseguenza.

Il concetto consiste nell'applicare una sovrapposizione di misure multiple di difesa, una sorta di sicurezza in profondità in quanto, non avendo mai hai la certezza assoluta di poter fermare gli attacchi, almeno si tenta di rendergli la vita il più difficile possibile. Il vantaggio di tale scelta è la sensibile limitazione dell'esposizione

agli attacchi: nel caso una minaccia riesca comunque ad oltrepassare gli strumenti di sicurezza di un dato livello, sono comunque presenti ulteriori difese più sofisticate ai livelli sottostanti che la costringono a dover studiare ulteriori strategie.

L'aspetto principale da ricordare è di non utilizzare un singolo elemento per tutta la sicurezza, ma cercare di applicare diversi sistemi di sicurezza in modo che siano ciascuno il più adatto possibile agli elementi che si sta cercando di proteggere. In questo modo, è risultato che l'80% degli attacchi può essere fermato, mentre il restante 20% può essere catturato dal sistema IDPS. Perciò, quest'ultimo deve essere trattato come un singolo componente, una parte della difesa multistrato (che incorpora la sicurezza informatica in tutti i suoi aspetti), e non deve essere assegnatali l'intera responsabilità della sicurezza. Può essere considerato un supplemento, un complemento alle altre difese, e non un rimpiazzo, in quanto può operare cooperativamente e attuare diverse funzioni per attuare una strategia di sicurezza multistrato.

La sicurezza informatica è caratterizzata da tre punti principali:

- Efficacia: deve avere come scopo finale l'obiettivo iniziale previsto
- Efficienza: non deve creare disagi per l'utente e per l'infrastruttura già esistente
- Contenimento dei costi: non deve impiegare notevoli risorse finanziarie che siano superiori alla soluzione dei problemi

La sicurezza stratificata a livelli è l'opzione migliore per far si che questi tre punti vengano rispettati; procedere per passi è diventato quasi indispensabile per contrastare l'enorme tipologia di attacchi esistenti (sia dall'esterno che dall'interno di una rete).

Il fatto di concentrarsi sulla sicurezza livello per livello migliora la precisione e la completezza delle misure difensive, ed inoltre permetter di isolare eventualmente solo lo strato interessato direttamente dall'intrusione, senza coinvolgere e minare la sicurezza globale dell'intero sistema informatico.

I livelli presi in considerazione sono quelli che caratterizzano l'ambiente dell'IT:

- Livello Perimetro
- Livello Rete
- Livello Sistema
- Livello Applicazioni
- Livello Dati

Per essere coerenti con il tema di questa Tesi di Laurea, ora vengono trattati solamente i livelli che possono contenere al loro interno i sistemi IDPS, indicandone l'importanza e descrivendo alcuni strumenti di pari livello.

Livello Rete

Questo livello comprende qualsiasi aspetto riguardante le tipologie di rete LAN (Local Area Network) e WAN (Wide Area Network). Essendo entrambi composte sia da sistemi clienti che serventi, le regole di sicurezza imposte devono essere differenziate rispetto alla tipologia che assume la risorsa come parte del sistema informatico globale.

Tale livello è importante e va progettato con maggior attenzione rispetto a quello precedente, ovvero al livello Perimetro. Infatti, il pericolo maggiore è che quest'ultimo venga configurato in base a condizioni precise ed affidabili, mentre il livello in esame abbia la facoltà di permettere la totale uscita delle informazioni provenienti dall'interno della rete. Questo comporta che la presenza di una minima falla all'interno riesca a compromettere comunque il sistema dall'esterno.

Ora segue una breve descrizione degli strumenti adatti per ottenere la protezione a questo livello.

NIDS e NIPS

Questi strumenti sono talmente importanti che sono contraddistinti da un'apposita sezione a loro dedicata. Rappresentano la versione a livello rete dei sistemi IDPS; viene omessa la loro descrizione in quanto affrontata con dettaglio nel capitolo 2.

Meccanismi dell'autenticazione e del controllo degli accessi

Questo aspetto è già stato accennato nel paragrafo 1.3. Ogni utente che necessita si avere accesso alla rete, deve precedentemente procedere con l'autenticazione attraverso determinati protocolli sicuri, in modo da garantire solo agli utenti autorizzati l'utilizzo (secondo le limitazioni concesse) delle risorse nella rete. L'intruso, sebbene sia eventualmente riuscito a oltrepassare le misure di sicurezza attuate al livello precedente (Perimetro) risulta quindi impossibilitato nello sfruttare le risorse.

Applicativi per la gestione delle vulnerabilità a livello di rete

Questi strumenti, aggiornandosi in modo continuo, individuano automaticamente le varie tipologie di vulnerabilità esistenti per determinati sistemi operativi o applicativi. Successivamente, procedono con la scansione dell'intera rete per controllare se tali software sono realmente aggiornati e privi di vulnerabilità; se necessario si occupano, di conseguenza, della gestione e della distribuzione delle rispettive contromisure.

Vantaggi degli strumenti a questo livello

I diversi strumenti appena analizzati sono molto sofisticati e analizzano accuratamente quello che accade nella rete, individuando, o addirittura prevenendo, l'attacco di un intruso che é riuscito ad oltrepassare le misure del livello precedente. Inoltre, cercano di verificare e porre rimedio a possibili vulnerabilità, oppure riescono ad impedire l'accesso non autorizzato alla rete e alle risorse. In questo modo, è possibile prevenire attacchi come trojan, exploit, backdoor, rootkit che possono provenire indirettamente dall'interno della rete.

Svantaggi degli strumenti a questo livello

E' probabile che vengano generati molti falsi allarmi, e questo può portare anche alla conseguente negazione di connessioni che invece sono legittime. Oppure si può presentare il rischio che gli stessi numerosi allarmi vengano ignorati quando invece si ha a che fare con una reale insidia. Inoltre, richiedono una continua gestione e in alcuni casi non sono perfettamente scalabili perché non applicabili alla totalità di software presenti in circolazione. Infine richiedono risorse non indifferenti dal punto di vista progettuale, strutturale, e dei costi per la gestione ed il mantenimento.

Livello Sistema

Il livello riguarda tutte le metodologie realizzate per un'intera rete ma decentralizzate ed adattate ad un singolo sistema della stessa rete. Tale livello è caratterizzato da un'importanza pari a quella del primo livello (Perimetro), perché, se in quel caso indicava l'ultima difesa della rete dall'esterno, qui rappresenta l'ultima difesa di un sistema della rete.

Ora si procede con l'affrontare le tecnologie che si addicono a questo strato della sicurezza informatica multistrato.

HIDS e HIPS

Questi strumenti sono la versione a livello sistema degli IDPS, e quindi analoghi rispettivamente ai NIDS e NIPS, solo che monitorano tutto ciò che accade su una scheda di rete appartenente ad un singolo sistema, invece che occuparsi dell'intera rete. Ovviamente, anche qui viene omessa la loro descrizione in quanto affrontata nel capitolo 2.

Meccanismo dell'autenticazione e del controllo degli accessi

Anche a livello di sistema un utente che necessita l'accesso, deve precedentemente procedere con meccanismi di autenticazione attraverso singole regole sulle password, e tutto quello che può essere collegato ed influenzato da account di tipo locale.

Applicativi per la gestione delle vulnerabilità a livello di sistema

A differenza della loro controparte a livello di rete, questi strumenti si occupano solamente della scansione del singolo sistema. Spesso consistono nella versione client e quindi si sincronizzano ai loro corrispettivi a livello di rete per adottare le giuste contromisure e applicarle direttamente al sistema.

Antivirus

Tale strumento è divenuto ormai essenziale, soprattutto per combattere minacce quali worm o rootkit. E' opportuno che venga collocato su ciascun sistema in modo da diminuire il carico di lavoro degli eventuali serventi utilizzati come antivirus globali della rete; quest'ultimi invece si occupano principalmente e solamente di particolari insidie.

Vantaggi degli strumenti a questo livello

Questo tecnologie garantiscono un altro livello di protezione e sono molto precise perché è possibile imporre regole ad hoc e specifiche per ciascun sistema che riveste il ruolo di obiettivo finale degli attacchi. Dato che il loro obiettivo è mirato, queste metodologie si prestano maggiormente per ambienti le cui caratteristiche variano minimamente nel tempo, in modo da limitare gli aspetti della gestione e dei costi.

Svantaggi degli strumenti a questo livello

In questo caso valgono le stesse considerazioni date per il caso riguardante il livello rete. A queste si aggiunge il fatto che sono necessarie diverse misure distinte in base allo specifico sistema al quale vengono applicate, e questo è causato dalla facile presenza di entità eterogenee. In ambienti più dinamici, perciò, è preferibile applicare meccanismi di controllo e impartire regole invece a livello globale.

2 IDPS: Descrizione, principi, peculiarità

2.1 Introduzione ai sistemi IDPS

2.1.1 Descrizione IDPS e relativa importanza

Nel mondo della sicurezza delle reti, l'abilità di conoscere effettivamente quando un intruso sta operando tentativi di intrusioni in sistemi, o altre attività malevoli, può fare la differenza tra la compromissione o meno del sistema informatico.

Nessun singolo sistema è troppo irrilevante per necessitare di sicurezza, come nessuna rete è troppo piccola per rimanere non protetta: se un attaccante può usare un computer o la rete, sicuramente lo farà. Inoltre Internet è uno degli ambienti più ostili, e, sebbene venga implementata una forte e preparata difesa, questa non fa desistere la persone ad attaccare.

Tutto ciò denota l'importanza dell'utilizzo di un IDPS, come un'indispensabile applicativo per un responsabile della sicurezza. Un IDPS, come già accennato in precedenza, è progettato ed utilizzato per l'Intrusion Detection, ovvero il rilevamento di attacchi o di utilizzi non autorizzati di sistemi, reti, e relative risorse, con lo scopo di limitarli e persino annullarli quando possibile.

Un IDPS è l'equivalente di un allarme altamente tecnologico configurato in modo tale da monitorare informazioni che risiedono ai confini con l'ambiente circostante, come attività ostili e intrusi ben noti. Permette di analizzare il funzionamento interno della rete, paragonabile ad una scansione ai raggi X o le analisi del sangue: infatti analizza il traffico per determinare l'esistenza di virus e invasori, altrimenti si è vulnerabili a tali minacce, e si accorge della loro presenza solo quando si presentano malfunzionamenti o corruzioni. In definitiva, tiene informato il responsabile della sicurezza riguarda la salute del e la sicurezza del sistema informatico monitorato identificando attacchi e attività sospette, vulnerabilità di sicurezza e debolezze.

In altri termini, si tratta di una tecnologia specializzata in grado di analizzare ed interpretare le attività di un singolo sistema o il traffico di una rete. Queste attività possono consistere nell'analisi delle chiamate di sistema, del contenuto di log relativi a diversi strumenti (firewall, server, sistemi locali) del flusso dei pacchetti di rete, ecc.. E' da ritenersi essenzialmente un packet sniffer automatizzato che allerta quando avviene qualcosa di interessante o sospetto.

Per analogia, un IDPS fa per un singolo sistema o una rete quello che un software antivirus fa per i file entranti in un computer: ispeziona il contenuto dei dati in transito per osservare e deviare possibili minacce, come un software antivirus ispeziona il contenuto di file in entrata, gli allegati di e-mail, le informazioni attive di pagine Web, e così via, per individuare firme di virus (pattern che combaciano con minacce ben conosciute) o possibili azioni malevole (pattern relativi a comportamenti che si rivelano, come minimo, sospetti).

Un IDPS rappresenta un'importante linea di difesa contro gli attacchi; il loro obiettivo fondamentale è rilevare i tentativi di compromissione il più velocemente possibile, prima che venga provocato qualsiasi danno, in modo da mitigare le vulnerabilità e procedere con l'assicurare la rete da tentativi di attacco

futuri. Se si riesce a intercettare un attacco, possibilmente nel modo migliore, un IDPS può rivelarsi l'unico modo per mostrare cosa è avvenuto e prevenire che questo accada nuovamente. Un altro aspetto significativo è venire a conoscenza del fatto che la compromissione sia avvenuta con successo, ovvero che alcuni sistemi siano stati compromessi.

I benefici nell'utilizzo di un IDPS sono evidenti, ma è importante impiegarlo con aspettative realistiche, ovvero tenendo presenti gli aspetti come l'installazione, il mantenimento, la gestione e l'interpretazione dei risultati. Richiede infatti buona conoscenza dell'ambiente (e delle sue diverse caratteristiche) nel quale viene introdotto e ; nel caso particolare dei sistemi a IDPS a livello di rete questo si traduce nella struttura della rete, che deve essere ben nota in modo da determinare la migliore posizione di inserimento del sistema IDPS, delle sue componenti e anche quali modalità operativa utilizzare.

Per quanto riguarda la struttura che costituisce un IDPS, generalmente lo strumento può ritenersi suddiviso in quattro elementi principali:

- sensore: si pone in ascolto su una determinata interfaccia per intercettare e raccogliere i dati in transito. Possono essere gestiti anche molteplici sensori simultaneamente.
- filtro: insieme di istruzioni restrittive (regole, pattern, ecc.) o di dati statistici per individuare i comportamenti anomali e utilizzati quindi per il confronto con i dati rilevati dalla sorgente
- motore: insieme dei file di configurazione dove avviene l'interfacciamento tra il sensore e il generatore di output, ed effettua l'analisi dei dati applicando il filtro
- generatore di output: dopo le operazioni svolte dal filtro, avviene la produzione del resoconto delle possibili insidie attraverso la produzione di diversi tipi di output

A questi si aggiungono, per completezza, gli elementi dell'ambiente esterno con i quali lo strumento interagisce:

- sorgente per il la raccolta dei dati: la locazione dove avviene la collezione delle informazioni da analizzare (un sistema singolo, componenti singoli di un sistema, un segmento di rete)
- output: visione dei risultati ottenuti, tipicamente tramite console a video

2.1.2 Utilizzi

Gli IDPS ormai sono riconosciuti come una parte principale delle sicurezza all'interno di un'azienda; infatti esistono diverse motivazioni che portano ad optare per l'utilizzo di tali strumenti:

- aiutare a rilevare tentativi o effettive intrusioni, permettendo al responsabile della sicurezza di prendere appropriate decisione per mitigare l'effetto conseguente e rimediare tramite diverse possibilità prima che avvengano danni ulteriori
- monitorare gli attacchi provenienti dall'esterno, ma anche quelli che giungono dall'interno (consistenti nella maggior parte delle minacce) con possibilità di rintracciamento per correlare le attività malevole
- prevenire eventuali futuri minacce dissuadendo gli intrusi: la conoscenza dell'applicazione di tecnologie sofisticate può scoraggiare la loro volontà di attacco, riducendola o addirittura annullandola. Questa può rivelarsi però un'arma a doppio taglio in quanto l'intruso, se particolarmente intraprendente, può essere invece ulteriormente spinto a risolvere sfide sempre più complicate

- individuare violazioni non rilevate da altri strumenti: è in grado di rilevare intrusioni collegate a vulnerabilità di applicativi e sistemi operativi; in questo modo può informare del pericolo il responsabile della sicurezza, il quale può effettuare gli opportuni backup del caso per evitare la corruzione delle informazioni. Infatti, molte vulnerabilità, sebbene conosciute, non vengono o non possono essere opportunamente corrette per svariati motivi. Quindi il sistema IDPS può informare che qualcosa non sta funzionando secondo le modalità previste e può porre rimedio a situazioni facilmente verificabili quali:
    - configurazioni errate
    - mancanza di tempo per sovraccarico di risorse o di lavoro da parte del responsabile della sicurezza, in particolar modo in presenza di reti vaste ed eterogenee
    - applicativi proprietari
- rilevare la parte iniziale dell'attacco consistente nella ricerca iniziale di un qualche punto debole, ovvero la fase di scansione iniziale. Riesce quindi ad identificare applicativi di scansione in grado di rilevare exploit noti (ad esempio su server), che spesso sono il preambolo all'arrivo a breve di veri attacchi molto più ostili
- identificare accessi fisici diretti, come connessioni non autorizzate (login, accessi non protetti a risorse condivise, ecc.) ed indiretti come attacchi di tipo DoS (Denial of Service), scansione di porte, funzionalità ping, ecc.
- rilevare attacchi di tipo buffer overflow, per sfruttare le debolezze exploit presenti nei sistemi operativi e negli applicativi specifici, producendo compromissioni ed attacchi di tipo DOS o DDOS (Distributed Denial of Service)
- rilevare minacce come backdoor e trojans (programmi per accesso e controllo remoto non autorizzato) per attacchi quali spam, invio di dati privati, o rendere il sistema vittima (inconsapevole) un supporto per l'attacco o un per la distribuzione di materiale illegale (trasformandolo in un cosiddetto "zombie")
- verificare l'integrità dei dati e delle applicazioni con codice personalizzato che può causare violazione nell'accesso fisico
- rilevare infezioni causate da virus attraverso l'utilizzo delle firme identificanti in modo preciso ed univoco la minaccia. Le informazioni del "contagio" vengono estratte dai log della vittima, in quanto l'infezione manifesta un comportamento anomalo dell'elemento monitorato facilmente riconoscibile. Dai log si evidenza inoltre per quali sistemi è necessario prestare particolare attenzione perché potenzialmente infetti e quindi si rivela una funzionalità molto utile in caso di "contagi" di massa

Se ci focalizza con particolare attenzione alla sicurezza globale di dove gli IDPS vengono inseriti, allora possono presentarsi ulteriori specifici vantaggi nel loro impiego, in quanto permettono di:

- identificare vulnerabilità e debolezze nei dispositivi di protezione perimetrale (firewall, switch, router). Si rivela utile nella loro gestione quando devono essere riconfigurati e per verificare regolarmente la funzionalità, ad esempio, delle regole presenti nel firewall e delle ACL (Access Control List, liste per il controllo degli accessi) dei router
- proteggere attività interne in aggiunta alle difese di perimetro, ovvero monitorare funzioni e applicazioni interne, quindi risorse molto importanti e delicate, quali l'accesso a database, funzioni DNS, Server mail, politiche dell'azienda, ecc..
- fornire informazioni sui pericoli per l'azienda: è possibile effettuare un resoconto della situazione in cui si trova l'azienda in merito al suo grado di sicurezza e ai possibili rischi, in modo da prendere

eventuali decisioni sulle risorse che dovrebbero essere impiegate nell'infrastruttura di rete. Inoltre, può indicare la probabilità di ricevere un determinato attacco o se questo è già avvenuto, informazioni utili nel caso siano presenti in azienda delle politiche di sicurezza oppure un piano per la gestione della stessa sicurezza; in questo modo è possibile rilevare violazioni di tali politiche e di altre norme specifiche dell'azienda (tentativi di login falliti, accessi critici di applicazioni, falle nella sicurezza) e forzare l'applicazione della stessa politica

2.2 Classificazione

Esistono diverse modalità di classificazione di un IDPS, in quanto esistono differenti suddivisioni gerarchiche e di categoria; infatti questi strumenti vengono distinti in base a diversi principali aspetti quali:

- la sola sorgente di informazioni che monitorano
- il tipo di approccio all'analisi degli eventi
- il tempo in cui avvengono le operazioni di riconoscimento
- la modalità di reazione

2.2.1 Sorgente monitorata

Questa caratteristica rappresenta la principale distinzione che classifica un IDPS, in quanto indica la sorgente nel sistema (in base a dove vengono localizzati) dalla quale vengono prelevate le informazioni sulle quali viene effettuata la rilevazione degli eventi.

Le informazioni da analizzare vengono acquisite tramite diversi modi, quali ad esempio lo sniffing e l'analisi dei log. I dati che tratta un IDP possono consistere in tre tipologie:

- dati specifici di un'applicazione: il corretto flusso dei dati. E' una tipologia di dati poco analizzata rispetto alle successive e quindi viene omessa la descrizione
- dati specifici del singolo sistema: chiamate a sistema utilizzate, contenuto di log, permessi al file system. Gli strumenti che si basano sull'analisi di questo tipo di dati vengono definiti HIDPS (Host Intrusion Detection/Prevention System)
- dati specifici della rete: contenuto dei pacchetti in transito per osservare quali sono i sistemi potenzialmente malevoli. Gli IDPS che monitorano questa tipologia di dati sono identificati con la sigla NIDPS (Network Intrusion Detection/Prevention System)
- dati che rientrano nelle precedenti categorie: consistono in informazioni di diversa natura, e quindi i relativi strumenti consistono in soluzioni definite ibride

HIDPS

Questi strumenti sono la versione a livello di singolo sistema degli IDPS e rappresentano il primo tipo di IDPS progettato ed implementato storicamente. Il controllo avviene su un singolo sistema (dove lo strumento risiede) relativamente alle attività rilevate dall'HIDPS su quello stesso sistema. Opera sui dati acquisiti e provenienti dall'interno di quest'ultimo, relativi principalmente al sistema operativo, agli applicativi e al file system, e catturati sfruttando la modalità non promiscua di una scheda di rete NIC (Network Interface Controller). Viene eseguito in pratica un controllo dettagliato sulle azioni eseguite nel

sistema, analizzando i file di log, le chiamate a sistema, le modifiche al file system, quelle allo stack e altre componenti.

L'HIDPS rappresenta l'ultima difesa dove le altre hanno fallito, in modo da prevenire la compromissione totale. Sebbene venga prestata più attenzione sui sistemi a livello di rete (che vedremo poi), anche questo tipo di strumento è molto potente, perché aggiunge un importante strato di sicurezza. Questi sono i principali aspetti positivi e negativi legati al suo utilizzo:

Vantaggi

- permette di analizzare le attività attuali con grande precisione, determinando esattamente quali utenti e/o processi sono coinvolti in un possibile attacco all'interno del sistema
- la modalità non promiscua è ben supportata e non richiede particolare costo computazionale da parte del processore del sistema in cui risiede
- il controllo a livello di singolo sistema produce ulteriori e nuove informazioni a livello locale riguardanti le modifiche del file system, le chiamate di sistema, i log di sistema, che, combinati con le comunicazioni di rete, forniscono una considerevole quantità di dati da analizzare alla ricerca più approfondita di possibili insidie
- favorisce la possibilità di impostare un'insieme di regole molto specifiche ed incentrate ad una migliore protezione individuale, quindi mirata; ciò provoca anche una diminuzione del numero di regole e quindi riduce il carico computazionale di ogni singolo sistema
- permette di conoscere, quasi immediatamente, se l'attacco ha avuto successo o meno, in quanto può individuare direttamente il risultato di un tentativo di intrusione accedendo e monitorando in modo diretto e locale i dati e i processi del sistema in questione, a differenza della versione a livello di rete
- molto potente per gli attacchi di tipo buffer overflow, e si rivela utile anche per correlare attacchi rilevati da differenti sensori dislocati nell'ambiente circostante
- riesce ad operare su dati criptati perché può analizzare quelli in uscita prima della criptatura, e quelli in entrata una volta effettuata la decriptatura.

Svantaggi

- soffre del problema dei numerosi falsi allarmi, quindi è necessaria una continua gestione per migliorare i risultati del riconoscimento
- essendo localizzati e mirati per ciascun sistema, devono essere installati, configurati e gestiti singolarmente, con un corrispondente impiego elevato di risorse in termini di tempo e di costi. Inoltre, utilizzano le medesime risorse del sistema dove risiedono, con conseguente influenza sulle prestazioni
- sono molto vulnerabili rispetto ad attacchi a livello di rete (in quanto prendono in considerazione solamente i pacchetti rivolti a loro) e rispetto ad attacchi di tipo Denial Of Service perché direttamente influenzati dalla situazione e dal comportamento del singolo sistema

NIDPS

Tali strumenti effettuano il monitoraggio dell'intera rete dal punto di vista della locazione dove vengono impiegati, ovvero l'intero ramo di una rete. Quindi, mettendosi in ascolto su un segmento, o collegamento, della rete, sono in grado di monitorare e proteggere l'intero traffico riguardante i diversi sistemi che sono

connessi a quello stesso tratto di rete. Il controllo avviene sul traffico di una rete catturando e analizzando i pacchetti in transito alla ricerca di tracce di intrusione.

A differenza dei precedenti strumenti HIDPS che utilizzano il funzionamento normale delle schede di rete NIC, questi sfruttano la modalità promiscua, in modo da permettere il monitoraggio non solo delle connessioni destinate all'indirizzo MAC della propria scheda di rete, ma alla totalità del traffico che transita in quello specifico segmento di rete.

Questa tipologia di IDPS è la più diffusa, utilizzata e sviluppata, in quanto la configurazione è generale per l'intero segmento nel quale essi operano. Ora vengono elencati i principali vantaggi e svantaggi relativi all'impiego di un NIDPS:

Vantaggi

- possiede un minimo impatto sulla rete esistente, in quanto spesso rimane nello stato passivo non interferendo con le normali attività della stessa rete; inoltre, se configurato, può essere reso invisibile, aumentando la sua sicurezza e quella complessiva della rete
- se ben posizionato nell'ambiente dove opera, è in grado di monitorare una vasta rete composta da molteplici sistemi, riuscendo ad analizzare la totalità del traffico

Svantaggi

- oltre agli headers dei pacchetti, viene analizzato anche il loro contenuto; di conseguenza il carico di lavoro è notevole, in particolar modo in presenza di reti estese o di forte traffico, con la possibilità ulteriore di tralasciare alcuni pacchetti che possono nascondere insidie. Una possibile futura soluzione è l'implementazione dei NIDPS completamente a livello hardware, con conseguente incremento della velocità
- riesce a rilevare solo se l'intrusione è avvenuta oppure no, ma non se ha avuto successo o meno; di conseguenza, il responsabile della sicurezza deve verificare personalmente, per ogni singolo sistema, le eventuali conseguenze della minaccia
- il problema dei falsi negativi è molto evidente, a causa della configurazione generale per l'osservazione del comportamento dell'intera rete. Perciò, non riesce ad analizzare in profondità le insidie specifiche relative ad un sistema considerato nella sua individualità
- non riesce ad accedere e ad analizzare i dati criptati e quindi il loro contenuto, in particolare quando viene utilizzato il sistema di criptatura a livello di rete tramite IPsec (Internet Protocol Security); ciò può essere risolto tramite l'uso di una politica di sicurezza più flessibile, ad esempio attraverso IPSec in modalità tunnel
- alcuni pacchetti accettati dai NIDPS possono venire scartati dai singoli sistemi, in quanto l'implementazione dello stack dei protocolli di rete può differire da quella dello stack dei sistemi, e quindi alcuni sistemi possono non seguire certi aspetti degli standard TCP/IP

Soluzioni ibride

Effettuano una combinazione dei due approcci precedenti, e cioè i dati vengono raccolti dagli strumenti in funzione nei singoli sistemi che li ospitano, e poi vengono integrati con i dati recuperati dalla rete.

2.2.2 Approccio all'analisi degli eventi

Questo aspetto riguarda la modalità grazie alla quale vengono riconosciute le anomalie del sistema, e sono presenti fondamentalmente due tecniche:

- Signature Detecion (o Misuse detection): rilevamento attraverso l'utilizzo di firme o regole ben definite
- Anomaly Detection: rilevamento, tramite euristiche, del discostamento del comportamento attuale da quello usuale

Signature Detection

Rappresenta la tecnica primaria, maggiormente utilizzata e più diretta perché rileva una potenziale minaccia attraverso una qualche forma di rappresentazione, di solito un insieme di regole (da qui deriva il termine Signature Detection), grazie alle quali è possibile definire i comportamenti anomali da rilevare.

Anomaly Detection

Approccio che cerca di classificare, di solito utilizzando la statistica e tramite diversi meccanismi e modelli, il comportamento normale del sistema, trattando di conseguenza come sospette tutte quelle azioni che si distaccano da esso. Utilizza concetti predefiniti (euristiche) per le normali (legittime) attività di sistema per distinguere le anomalie caratterizzate da devianze rispetto al normale comportamento del sistema, e monitorarle, riportarle o bloccarle non appena accadono.

Le misurazioni utilizzate possono essere:

- statiche o euristiche: il superamento di una soglia limite su determinati attributi del comportamento
- statistiche di tipo parametrico dove la distribuzione degli attributi del profilo rientrano in un certo pattern, oppure non parametriche dove questa distribuzione è appresa da valori storici osservati nel tempo

Alcune tipologie implementano profili di utente, di sistemi o di connessioni di rete, che rappresentano le linee guida dell'attività normale; possono essere create utilizzando campioni statistici o approcci basati su regole o reti neurali, a sua volta nati dai dati storici collezionati durante le normali operazioni.

Un IDPS basato su questa tecnica è caratterizzato da questi aspetti:

Vantaggi

- elevata generalità, perché è in grado automaticamente di adattarsi per rilevare nuovi schemi di minacce, quindi rilevarli seppure non abbia conoscenze specifiche
- produce nuove info per definire nuovi pattern per le definizioni delle firme

Svantaggi

- è necessaria la scelta di quali modelli e quali elementi, come l'oggetto della misurazione e altri parametri d'impostazione, in modo da selezionare adeguatamente le possibili insidie da sottoporre al controllo del responsabile della sicurezza

- alcuni risultati possono rivelarsi non attendibili a causa dell'imprevedibilità del comportamento umano

2.2.3 Tempo delle operazioni di riconoscimento

Questa distinzione viene effettuata in base a quale periodo viene scelto per effettuare l'effettiva analisi dei dati rilevati e quindi il riconoscimento di possibili minacce; la tipologia di strumenti definiti in base a questo parametro può essere:

- Online: il tempo d'analisi è pressoché identico a quando avviene il rilevamento dei dati
- Offline: l'analisi dei dati rilevati avviene in un tempo futuro

Online

Vengono rilevate le intrusioni in tempo reale, ovvero vi è una risposta immediata non appena viene riconosciuto un tentativo di intrusione. Nella maggior parte dei casi, vengono esaminati gli eventi che accadono entro un certo lasso di tempo; quando vengono rilevate possibili intrusioni in base a certe situazioni, vengono generati gli opportuni allarmi, ed eventualmente ulteriori procedure e reazioni, comprese contromisure automatiche.

Un IDPS, che si basa su questa peculiarità, presenta questi aspetti:

Vantaggi

- possibilità di reagire in modo quasi istantaneo alle minacce, limitando, o addirittura impedendo, ulteriori complicazioni

Svantaggi

- se l'intruso riesce a riconoscere la presenza di contromisure automatiche (ad esempio il blocco delle operazioni) può sfruttare la situazione attraverso un attacco di tipo Denial Of Service, portando al fallimento di un principio della sicurezza informatica quale la disponibilità
- vengono impiegate sostanziose risorse in termini di computazione per i controlli in tempo reale, e di spazio per l'archiviazione dei dati

Offline

Processano i dati raccolti con un certo ritardo, ovvero effettuano il riconoscimento eseguendo la verifica sui dati rilevati solo in un successivo momento. I principali vantaggi e svantaggi relativi all'utilizzo di tale caratteristica sono i seguenti:

Vantaggi

- analisi molto completa, dettagliata, senza restrizioni sul tempo e sulla quantità di dati, che può essere anche notevole
- alleggerimento del carico computazionale dovuto ad un'analisi a posteriori

Svantaggi

- l'intrusione può essere già avvenuta e aver provocato le relative negative conseguenze; inoltre è necessaria l'archiviazione di grandi quantità di dati

2.2.4 Modalità di reazione

Questo aspetto si verifica quando sono stati analizzati gli eventi e l'intrusione è stata rilevata. A questo punto l'IDS deve reagire in una qualche modalità, che può essere:

- passiva: semplice rilevamento e raccolta delle informazioni
- attiva: rilevamento, analisi dei dati raccolti e attuazione di misure di reazione alle minacce rilevate

Passiva

La reazione consiste esclusivamente nell'informare tipicamente il responsabile della sicurezza che qualcosa di errato sta avvenendo dai dati ottenuti dalle risorse che l'IDPS monitora, senza la peculiarità di proteggere, effettuando la semplice registrazione dell'avvenimento secondo molteplici possibilità di memorizzazione dell'output. La reazione vera e propria, se necessaria, viene messe in pratica da chi attinge a queste informazioni.

Gli IDS rientrano in questa categoria in quanto permettono solamente di ottenere informazioni dalle diverse risorse che monitorano, ovvero notificano ad esempio la presenza dell'intrusione, chi l'ha effettuata e in che modo.

Vantaggi

- grande fonte di informazioni utili per identificare e tenere traccia delle intrusioni, in modo da aiutare a prendere successivamente le decisioni più adatte per proteggere il sistema e stabilire una corretta politica di sicurezza

Svantaggi

- lo scopo è prettamente informativo, senza l'obiettivo di proteggere

Attiva

Non appena viene rilevata l'intrusione, il sistema attua automaticamente delle contromisure per proteggere e agire in modo effettivo, attraverso delle azioni quali:

- la collezione di ulteriori dati, tramite l'incremento della sensitività del sensore, in modo da ottenere più informazioni attinenti all'intrusione
- la terminazione dell'attacco secondo diverse modalità come il bloccaggio di un particolare indirizzo IP tramite filtraggio oppure l'invio di pacchetti per la chiusura forzata della connessione malevola.

In questa categoria sono presenti gli IPS e i firewall perché entrambi sono in grado di mettere in atto reali misure di protezione. Per quanto riguarda i primi strumenti, le varie azioni in risposta agli attacchi vengono descritte nel paragrafo successivo **Errore. L'origine riferimento non è stata trovata.**, mentre i secondi dispositivi riescono a filtrare l'accesso solo per specifiche connessioni, servizi, ecc., in modo da impedire l'accesso agli intrusi.

Vantaggi

- reazione automatica senza necessità di intervento da parte del personale

Svantaggi

- alcuni comportamenti, prodotti dall'attuazione delle contromisure automatiche, possono rivelarsi persino controproducenti e soggetti ad attacchi, e quindi devono essere opportunamente gestiti e controllati con particolare attenzione

2.3 Analisi degli IPS per combattere i falsi allarmi

Fin'ora sono stati trattati i sistemi IDPS in termini generali, senza concentrarsi particolarmente su determinati aspetti. In questa sezione, viene analizzata con maggior dettaglio la categoria dei sistemi IPS, cioè gli IDPS caratterizzati dalla funzione di prevenzione delle intrusioni, oltre che dal loro semplice rilevamento. Viene svolta quest'analisi approfondita in quanto sono tipologie più sofisticate rispetto agli IDS e soffrono maggiormente del problema dei *falsi allarmi*, che va perciò trattato con grande attenzione.

In questo caso, la protagonista principale è la prevenzione delle intrusioni, l'*Intrusion Prevention*, che consiste fondamentalmente nel processo di attuazione della risposta attiva in presenza di rilevamento di attacchi. Quindi, innanzitutto bisogna dare una definizione di risposta attiva:

La riposta attiva consiste in una misura reattiva in risposta ad attacchi tramite contromisure automatiche per prevenire che le minacce riescano a compromettere la sicurezza di un sistema singolo o di una rete, dunque qualsiasi funzione che altera o blocca il traffico rilevato da un IDS. Ovviamente, si attua solo su minacce effettivamente rilevate.

L'obiettivo di un IPS, che si basa appunto sull'Intrusion Prevention, è quindi quello di rispondere in modo automatico ad un attacco rilevato e minimizzare, o idealmente annullare, gli effetti dannevoli di un tentativo di intrusione senza richiedere l'intervento di un responsabile della sicurezza. Un IPS è essenzialmente un IDS provvisto della funzionalità aggiuntiva di poter attuare un'azione in risposta ad un attacco; l'IDS è solo un attore passivo che genera migliaia di allarmi ma non risponde in nessun modo.

Gli IPS si possono distinguere concettualmente in due sottocategorie, ovvero i dispositivi con risposta attiva semplice, e quelli con una risposta attiva avanzata.

I dispositivi con *risposta attiva semplice* attuano contromisure quali la riconfigurazione dinamica o alterazione dei meccanismi del controllo dell'accesso, delle sessioni, o anche di pacchetti singoli sulla base di allarmi prodotti da un IDS. Purtroppo, tali dispositivi non riescono a fermare l'attacco prima che questo raggiunga la vittima perché reagiscono dopo che il dato fa il suo ingresso nel sistema singolo o nella rete, dando vita ad una sfida contro il tempo tra l'attacco e la reazione allo stesso; spesso non è chiaro se la contromisura abbia avuto effetto, e spesso vince il primo di questi.

Gli strumenti con *risposta attiva avanzata*, considerati i veri e propri IPS, invece, riescono anche ad attuare la modifica o lo scarto dei dati che rappresentano un attacco non appena questi attraversano lo strumento di prevenzione. Di solito, questa tipologia di dispositivi si riferisce all'utilizzo di strumenti situati dove avviene l'effettivo transito delle informazioni, in modo da interagire direttamente con i dati da osservare.

Infatti, l'unico modo per l'effettiva prevenzione è l'evitare che l'attacco riesca ad accedere al primo passo e solo un dispositivo di questa tipo può farlo. Questi dispositivi riescono a modificare o eliminare i dati non appena quest'ultimi attraversano tali strumenti e prima che giungano al sistema singolo o alla rete, con lo scopo di annullare effettivamente e sin dalla fase iniziale gli attacchi.

Gli IPS, soprattutto quelli rientrati nella categoria della risposta attiva avanzata, non hanno comunque ancora eguagliato l'utilizzo degli IDS, soprattutto per il fatto che necessitano di una costante regolazione, revisione, e monitoraggio per assicurare l'attuazione delle corrette operazioni. Inoltre, portano con se un prezzo da pagare non indifferente, come i fondamentali problemi di prestazioni e stabilità all'interno del sistema singolo o della rete che proteggono. Le azioni di analisi infatti richiedono tempo e provocano ritardi nella reattività e nel throughput dell'elemento sotto osservazione.

2.3.1 Categorie

Ora è opportuno dare una veloce descrizione delle sottocategorie che costituiscono gli IPS, ovvero quelli a livello di sistema (HIPS) e a livello di rete (NIPS), con particolare attenzione verso le misure che mettono in atto in risposta agli attacchi. A loro volta, ciascuno di questi insiemi è formato dalla tipologia con riposta attiva semplice e da quella con riposta attiva avanzata.

2.3.1.1 HIPS

Risposta attiva semplice

A livello di singolo sistema, i dispositivi con risposta attiva semplice, di solito, vengono implementati a livello software e direttamente nel sistema stesso. Un evento sospetto può essere rilevato tramite diversi modalità, come l'analisi dei file di log, delle chiavi di registro e dei file specifici associati con exploit conosciuti, o dei server sospetti che operano su porte caratterizzate da numero alto.

Dopo aver rilevato possibili intrusioni, tali sistemi reagiscono secondo determinate azioni, che non necessariamente prevengono l'attacco dall'aver successo, ma cercano di mitigarlo, riducendone gli effetti ed i danni.

I meccanismi che caratterizzano questi dispositivi consistono nell'attuazione di modifiche al sistema, come l'alterazione del permessi di accesso al file system, la modifica dei controlli per gli utenti, la rimozione di virus e worm, la riconfigurazione dell'insieme di regole del firewall locale dopo il rilevamento dell'attacco, ecc.. I software antivirus possono quindi rientrare in questa categoria in quanto scandiscono e sfruttano il file system e le informazioni sul registro per determinare la presenza di specifiche infezioni, per poi rimuovere il codice corrispondente attraverso normali operazioni sullo stesso file system.

Risposta attiva avanzata

Attraverso questa tipologia di strumenti, esiste la possibilità concreta di bloccare dall'inizio l'exploit prima che abbia successo; per questo motivo si rivelano molto potenti perché sfruttano le API (Application Program Interfaces) del sistema operativo e del kernel stesso.

Le contromisure a livello di sistema che riguardano la risposta attiva avanzata sono principalmente:

- le Application Shim, ovvero pezzi specializzati di codice specifici per l'applicazione e strettamente integrati in essa in quanto operano all'interno dello spazio degli indirizzi del processo dell'applicazione. Intercettano i dati in ingresso, li ispezionano alla ricerca di attività malevole ed eventualmente li validano. Se i dati sono malevoli, possono venire opportunamente modificati e convertiti in una forma innocua, oppure possono venire direttamente scartati in modo da non passare ad un alto livello ed interagire pericolosamente con la parte rimanente dell'applicazione
- tecniche offerte dal kernel del sistema operativo quali l'intercettazione di chiamate a sistema e i meccanismi forzati (LIDS, DAC, MAC). La procedura di prevenzione viene realizzata direttamente all'interno del kernel in modo da permettere o impedire che l'applicazione esegua chiamate a sistema ed utilizzi le risorse in modalità compromettenti. Gli attacchi, infatti, una volta ottenuta la compromissione dell'applicazione, possono sfruttare l'interfaccia delle chiamate a sistema e i relativi servizi forniti
- le misure a tempo di compilazione e tempo di esecuzione per la protezione dello stack, come l'utilizzo di canarini o di uno stack non eseguibile, per prevenire gli attacchi di tipo buffer overflow

2.3.1.2 NIPS

Risposta attiva semplice

A livello di rete, le azioni che rientrano nella categoria della risposta attiva semplice vengono applicate, in particolare, ad attacchi che si propagano in modo automatico, piuttosto che a quelli individuali che si sviluppano con diverse comunicazioni sulla distanza e non al primo passo. Infatti, i dispositivi che sfruttano questo tipo di azioni sono impotenti nel prevenire attacchi malevoli che necessitano solo di un singolo pacchetto per raggiungere con successo il loro obiettivo prefissato.

Gli attacchi possono essere rilevati tramite diversi meccanismi, accomunati comunque dal fatto che devono interagire con un dispositivo esterno ed in modo indiretto sui dati; generalmente esistono tre differenti strategie di risposta attiva semplice a livello di rete in base ai corrispondenti primi tre diversi livelli del modello ISO/OSI (non viene preso in considerazione il livello Fisico in quanto le risposte automatiche non dovrebbero riguardare le caratteristiche fisiche della rete):

- a livello Collegamento Dati vengono disabilitate le porte dello switch dove è avvenuto l'attacco; è opportuno solo per attacchi generati da sistemi locali. E' applicabile, inoltre, la possibilità di temporizzare il bloccaggio, perché è probabile che le porte non debbano essere bloccate in modo indefinito
- a livello Rete: vengono modificate le politiche di un firewall o le ACL di un router attraverso l'aggiunta di regole in modo da bloccare tutti i pacchetti in entrata ed in uscita corrispondenti all'IP dell'attaccante. Anche qui è importante l'uso della temporizzazione siccome le regole o le modifiche potrebbero venire rimosse dopo un lasso di tempo configurabile
- a livello Trasporto vengono generati messaggi di spoofing contenenti codice di errore per resettare la connessione (messaggi RST di reset sfruttando il protocollo TCP) oppure per segnalare porta non raggiungibile (sfruttando ICMP tramite protocollo UDP). Quindi consistono in azioni per terminare forzatamente le connessioni malevoli o notificare la non raggiungibilità. In questo caso la temporizzazione non è applicabile in quanto le contromisure avvengono per pacchetto o per sessione.

In alcuni casi può essere presente la funzionalità della whitelist, ovvero la possibilità di scegliere che determinati indirizzi IP o reti, o addirittura protocolli, non debbano mai essere soggetti a risposte automatiche.

Risposta attiva avanzata

La principale differenza con la categoria precedente consiste nell'utilizzo diretto dei dispositivi definiti "in linea", ovvero residenti direttamente sul percorso che i pacchetti, da monitorare, seguono quando attraversano la rete. La loro capacità di interazione diretta consiste nel riuscire ad attuare la modifica o lo scarto dei pacchetti (che rappresentano un attacco) durante il loro tragitto non appena attraversano l'IPS e prima che giungano ai dispositivi interni collegati allo stesso sistema IPS. Infatti l'IPS, in questo caso, non ha l'obbligo di inoltrare al passo successivo i pacchetti che vengono visti come malevoli.

I principali dispositivi che vengono sfruttati sono:

- i router, che si occupano di far attraversare i pacchetti IP tra reti connesse
- i firewall, che aggiungono uno strato essenziale di sicurezza e impongono delle precise politiche alle normali comunicazioni di rete
- i bridge, che connettono due segmenti ethernet insieme, senza però dar vita ad un nodo ulteriore nel percorso di rete (il campo TTL dei pacchetti che lo attraversano rimane invariato)

I dispositivi a livello di rete che attuano una riposta attiva avanzata, ovvero i veri IPS, possono mettere in atto le medesime contromisure relative di quelli con risposta attiva semplice, con l'aggiunta però che possono agire anche a livello Applicazione; infatti, sono in grado di alterare il vero contenuto informativo dei dati dei singoli pacchetti in modo da essere resi innocui e l'attacco corrispondente diventi inutile prima che raggiunga il suo obiettivo finale. Lo svantaggio è rappresentato dal fatto che il checksum a livello Trasporto deve essere ricalcolato. Inoltre, non può essere applicata la temporizzazione, perché l'effetto della modifica dei dati è transitorio e non persiste una volta che il pacchetto viene fatto passare.

Contromisure ibride

Possono avvenire anche combinazioni delle varie misure che rompono gli schemi delle normali convenzioni sistema singolo e rete. Un esempio sono le Applicazioni Shim che si interfacciano con un firewall remoto per aggiungere una regola in modo da bloccare un IP attaccante un'applicazione. Tuttavia, siccome richiedono comunicazioni tra diversi sistemi, rientrano maggiormente nella categoria delle metodologie con risposta attiva semplice.

2.3.2 Utilizzi

Un sistema IPS, grazie alle sue caratteristiche e possibilità complesse di funzionamento, viene impiegato per diversi obiettivi:

- rappresenta una delle poche soluzioni in caso di software vulnerabile non aggiornabile o sottoponibile a patch. Quando viene annunciata la vulnerabilità di un applicazione, può trascorrere molto tempo (se non addirittura l'impossibilità) per la disponibilità di un aggiornamento, di una patch che ripari la falla, o del rilascio di un'intera nuova versione dello stesso software. Questo può

essere causato dal fatto che la patch è di difficile distribuzione, causa più problemi di quelli per cui è stata progettata per risolvere, è contenuta in un pacchetto completo e non disponibile singolarmente, oppure è legata ad un software proprietario. Se l'applicazione è critica per il business e opera su un server che deve rimanere globalmente accessibile (per esempio un mail server), quest'ultimo non può essere posto in modalità offline. Nel frattempo, l'unica soluzione è il ricorso ad un sistema preventivo in grado di legittimare o meno le funzionalità dell'applicazione per continuare ad operare pur essendo presente l'exploit. Perciò, nel frattempo, un IPS permette di prendere decisioni granulari per assicurare la sicurezza attraverso diverse possibilità d'azione

- permette di evitare sin dall'inizio anche la minima traccia di compromissione del sistema; infatti, se questa malauguratamente avviene, tutto il software infetto deve venire installato nuovamente da sorgenti immacolate e ben conosciute. Siccome è difficile determinare in modo preciso quali elementi sono stati infetti, il modo migliore è quello di reinstallare l'intero sistema operativo e la totalità delle applicazioni. Anche il kernel può essere stato compromesso, e quando questo accade qualsiasi aspetto, anche quello più basilare, non può più essere considerato affidabile. Per ottenere informazioni utili sulla compromissione o meno del disco montato si può applicare al file system un'analisi di tipo forense.
- rileva e blocca worm ed exploit automatici e ben definiti. Spesso e volentieri, gli autori di elementi malevoli, non si curano di cambiare le caratteristiche della loro creazione, che rimane quindi invariata; in aggiunta, non sfruttano tecniche di evasione come lo spezzettamento della sessione, la frammentazione dei pacchetti, e la tecnica dello shell code sotto varie forme. Di conseguenza, un worm può venire frequentemente rilevato senza provocare molti falsi allarmi, perché caratterizzato da un alto grado di unicità che favorisce la configurazione di un IPS. Inoltre, alcuni worm anche molto dannevoli possono lasciarsi dietro programmi backdoor o installare logger dei pulsanti digitati sulla tastiera, e richiedere comunicazione bidirezionale tra attaccante e vittima, quindi discretamente riconoscibili.

2.4 Analisi degli NIDPS per combattere i falsi allarmi

In questa sezione viene presa in maggiore considerazione la categoria dei NIDPS, in quanto, come accennato nell'introduzione, la parte applicativa analizza i log prodotti dallo strumento Snort, un sistema NIDPS che opera esclusivamente a livello di rete.

Per prima cosa viene trattato un aspetto di questa tipologia di sistemi, ovvero dove avviene il loro inserimento. Nell'ottica globale di una rete, è fondamentale studiare una buona metodologia per scegliere la collocazione di un NIDPS all'interno della stessa rete, con l'obiettivo di *ridurre il numero di falsi allarmi* e di falsi negativi.

Secondariamente viene introdotto il sistema NIDPS per eccellenza, Snort, che verrà sfruttato per creare i log da sottoporre ad analisi.

2.4.1 Collocazione di un NIDPS

I sistemi NIDPS, similmente ai firewall, possono essere realizzati totalmente a livello software, o possono sfruttare una combinazione di aspetti software e hardware sottoforma di tecnologie indipendenti, preinstallate e preconfigurate.

Possono operare sui medesimi dispositivi o server dove operano firewall, proxy, o altri servizi perimetrali, anche se è molto più diffuso l'uso di sensori separati. Sono comunque preferibili NIDPS che non operano sugli stessi elementi dove vengono installati i firewall o altri servizi da monitorare; in questo modo permettono un monitoraggio più attento e accurato.

Esistono diverse possibilità di inserimento di un NIDPS e dei suoi sensori, ciascuna caratterizzata, allo stesso tempo, da vantaggi e svantaggi. La scelta migliore è quella di ottenere un buon compromesso tra le peculiarità volute ed i costi in base alle necessità dell'azienda.

2.4.1.1 Prima problematica

Innanzitutto, bisogna tener presente che quando si ha a che fare con uno strumento che funge da analizzatore impiegato in una rete switchata, il sensore purtroppo riesce a rilevare solo il traffico rivolto a ad un particolare segmento di rete. Quindi sorge un primo problema, ovvero come poter osservare il traffico totale; per tale aspetto esistono diverse soluzioni.

Collegamento alle porte SPAN di uno switch

La prima opzione è quella di collegare l'IDS ad una porta SPAN (chiamate anche di mirror) di uno switch (se dotato di tale porta), ovvero sfruttando il cosiddetto mirroring. In ciascuna di queste porte, infatti, viene replicato tutto il traffico che attraversa lo switch.

FIGURA

Vantaggi

- è considerata la metodologia più sicura e ben vista nell'ambito dell' ambiente IT
- non sconvolge la topologia dell'infrastruttura già presente, e quindi i costi sono bassi
- non occorre l'utilizzo di hub con relativi problemi in reti complesse e con notevole traffico
- non esistono elementi singoli che portano al fallimento dell'intero sistema dove sono presenti, i cosiddetti SPOF (Single Point of Failure)
- è possibile gestirlo direttamente dalla rete interna
- può diventare un NIPS

Svantaggi

- le porte SPAN sono presenti solo in switch molto sofisticati, e spesso sono uniche
- il traffico totale risulta di molto superiore alla capacità trasmissiva di una singola porta SPAN, con conseguente sovraccarico e perdita di prestazioni e di pacchetti, soprattutto in presenza di forte traffico
- sotto carico eccessivo, può essere anche controproducente se operativo da NIPS

Collegamento ad un hub

La seconda opzione consiste nell'utilizzare un hub, a cui collegare i sistemi del segmento di rete da controllare, ovvero il router ed il firewall, e la rete con il NIDS. Questa soluzione è possibile in quanto l'hub è in grado di replicare il traffico su tutti i segmenti collegati prima di inviarlo al coretto destinatario.

FIGURA

Vantaggi

- utile per reti di media-grande dimensione
- riesce a rilevare sicuramente tutto il traffico senza perdite
- buona alternativa se si utilizzano switch semplici sprovvisti di porte SPAN
- non sconvolge la topologia dell'infrastruttura già presente, e quindi i costi sono bassi
- è possibile gestirlo direttamente dalla rete interna
- può diventare un NIPS

Svantaggi

- in presenza di forte traffico, l'ampliamento dei domini di collisioni (dovuto alla presenza dell'hub) può portare alla generazione di molte collisioni e al degrado delle prestazioni in quel punto della rete, fino alla saturazione dell'hub
- aggiunta di un SPOF
- il risparmio iniziale in termini di costi, può portare ad una spesa eccessiva futura al verificarsi di probabili problemi
- sotto notevole traffico, può essere anche controproducente se operativo da NIPS

Collocazione all'interno di macchine dedicate

In questa opzione il NIDPS viene implementato direttamente all'interno della macchina dedicata alle funzionalità di firewall o di proxy.

FIGURA

Vantaggi

- utile per reti di media-piccola dimensione
- riesce a rilevare sicuramente tutto il traffico, in entrata ed in uscita, senza perdite
- è possibile gestirlo direttamente dalla rete interna
- non è necessario l'acquisto e l'impiego di macchine aggiuntive dedicate esclusivamente alla funzionalità di NIDPS
- può diventare facilmente un NIPS attraverso la modalità "in linea", in quanto può interagire in modo diretto con il firewall

Svantaggi

- può essere applicato solo nei casi in cui il collegamento verso l'esterno è unico, e se accostato a particolari tipologie di macchine
- provoca un impatto sull'infrastruttura, in quanto deve essere opportunamente integrato con le macchine già presenti

- operando come NIPS può portare alla terminazione non voluta di connessioni

2.4.1.2 Seconda problematica

Dopo aver risolto il primo problema, ovvero aver trovato la soluzione migliore per riuscire a rilevare il traffico totale, rimane la questione di scegliere in quale posizione esatta della rete porre il NIDPS; anche in questo caso esistono più alternative. La scelta ottimale consiste, se possibile, nella combinazione di varie opzioni, anche se questo richiede buone risorse; in ogni caso, bisogna assolutamente tener presente che il piazzamento deve essere tale in modo da consentire poi alle persone di competenza di esaminare adeguatamente i dati rilevati.

A monte del firewall

Il primo punto nodale si trova nel collegamento tra l'esterno ed il firewall che protegge la rete interna gestita dal router, quindi a diretto contatto con l'esterno.

FIGURA

Vantaggi

- cattura di tutto il traffico sia in entrata che in uscita
- possibilità di osservare la totalità degli attacchi destinati alla rete interna, quindi una visione effettiva del numero e del tipo di insidie che si possono verificare

Svantaggi

- il numero di falsi allarmi può rivelarsi molto elevato e ciò richiede una configurazione del NIDPS con bassa sensitività. Inoltre, può indurre un'analisi poco accurata da parte dell'amministratore e, di conseguenza, i veri pericoli possono venire ignorati
- la quantità di traffico può essere notevole, dunque c'è la possibilità della saturazione e che molti pacchetti vengano ignorati
- se il firewall funge anche da NAT (Network Address Translation), non è possibile ricavare informazioni su chi ha inviato le insidie dalla rete interna
- non riesce a rilevare minacce caratterizzate dalla criptatura dei dati

A valle del firewall

Il secondo punto critico risiede tra il firewall e la rete interna governata dal router.

FIGURA

Vantaggi

- rileva solo gli attacchi che hanno superato il firewall, quindi quelli che probabilmente sono da considerare veramente insidiosi e da analizzare subito con attenzione
- il traffico da rilevare e il numero di falsi allarmi è minore, quindi meno carico di lavoro per il NIDPS e per chi deve controllare gli allarmi
- è possibile rilevare facilmente gli attacchi molto insidiosi provenienti dalla rete interna, in quanto vengono individuati direttamente i sistemi che generano traffico malevolo

- gli attacchi più comuni vengono scartati dal filtraggio dello stesso firewall

Svantaggi

- non si ha la visione dei veri pericoli provenienti dall'esterno e della loro origine
- può crearsi un falso senso di protezione
- vulnerabile ad attacchi provenienti dalla rete interna

2.4.2 Il sistema NIDPS per eccellenza: Snort

Caratteristiche

Snort consiste in un applicativo open source e rappresenta un'evoluzione dei comuni sistemi IDPS. In particolare, è un IDS che opera a livello di rete, ovvero è un NIDPS. E' caratterizzato da un'architettura modulare che porta a notevoli vantaggi e queste sono alcune delle sue peculiarità principali che lo contraddistinguono:

- opera un'analisi del traffico in tempo reale e la registrazione di pacchetti provenienti da reti caratterizzate da diversi protocolli
- è caratterizzato da un potente motore di analisi per la rilevazione del traffico sospetto, anch'esso modulare, che permette di aggiungere nuove modalità di rilevamento tramite l'utilizzo di plugin. Alcuni esempi di queste estensioni sono la possibilità di riconoscere la frammentazione malevola dei pacchetti (per sfuggire ai controlli) tramite il loro riassemblamento, la rilevazione dei portscan, ecc..
- attua un'analisi di protocollo, di ricerca e utilizza delle rules (regole di rilevazione delle intrusioni) anche sofisticate (create tramite un semplice e apposito linguaggio di scripting), che permettono l'analisi ed il confronto con i pacchetti in transito alla ricerca di possibili intrusioni (il cosiddetto matching). Le regole vengono ottenute soprattutto sulla base di database contenenti le cosiddette signature: firme di attacchi ben conosciuti che essenzialmente descrivono i tipi di pacchetti indicanti che il sistema è sotto attacco o un attacco è già avvenuto con successo
- gestisce gli allarmi tramite molteplici meccanismi di registrazione, utilizzando estensioni per le diversi tipologie di output

E' in grado di operare secondo molteplici modalità operative (tutte passive tranne l'ultima):

- Sniffer: medesima modalità di funzionamento del tool tcpdump, ovvero intercetta, in modalità passiva, il traffico in transito sull'interfaccia impiegata per il rilevamento delle intrusioni e mostra i dati raccolti a video
- Logger di pacchetti: opera secondo la modalità precedente, ed in aggiunta registra il traffico catturato memorizzandolo secondo diverse possibilità
- NIDS: analizza il traffico rilevato effettuando il confronto tra ciascun pacchetto e le regole. Se il confronto fornisce esito positivo significa che è stata rilevata una possibile intrusione; di conseguenza vengono prodotti particolari avvisi ed eventualmente memorizzati gli allarmi
- NIPS: se viene rilevata un intrusione, agisce mettendo in pratica alcune delle contromisure di risposta attiva (descritte nel paragrafo **Errore. L'origine riferimento non è stata trovata.**), sia semplice che avanzata (tramite modalità "in linea")

In definitiva, Snort (rispetto alla classificazione dei sistemi nel paragrafo 2.2) è un sistema NIDPS che si basa sul riconoscimento delle minacce, sfruttando la tecnica Signature Detection, tramite un analisi di tipo online, e può operare sia in modalità passiva (NIDS) che in modalità attiva (NIPS).

Struttura

Snort può essere ritenuto un'evoluzione se paragonato a prodotti analoghi, e questo lo si denota in particolar modo analizzando la sua struttura, costituita da:

- sensore: possono essere impiegati diversi sensori, ciascuno annesso ad una corrispondente interfaccia. Viene utilizzato un modulo per la raccolta dei pacchetti, basato sulle librerie LibPcap o WinPcap
- decoder dei pacchetti: effettua l'interfacciamento tra il sensore ed il resto della struttura; si occupa della decodifica iniziale dei dati, distinguendoli seconda del protocollo sfruttato e cercando di garantire la massima velocità possibile
- preprocessori: aggiuntivi rispetto alla struttura tipica standard; possono integrarsi quindi a questa con lo scopo di rilevare tipi di insidie che altrimenti la struttura centrale non riuscirebbe a farlo, oppure per dilazionare il costo computazionale e quindi le prestazioni del sistema generale. Consistono in filtri che operano sui pacchetti (possono essere chiamati una sola volta per pacchetto) e svolgono particolari funzioni complesse per ricercare pattern di insidie
- plugin di rilevamento: elementi ulteriori che analizzano uno o più parametri di un pacchetto per determinare se rispettino alcuni criteri stabiliti (ad esempio i flag dei pacchetti TCP). Possono essere chiamati anche diverse volte per pacchetto, ricevendo in ingresso argomenti diversi.
- regole: insieme di istruzioni, con specifiche, relative alle insidie da tenere in considerazione e applicate ai dati raccolti per effettuarne il confronto. Si differenziano per tipo di attacco, applicativo, ecc., e devono venire esplicitamente abilitate nella configurazione di Snort per risultare attive
- motore di rilevamento: cuore centrale per il rilevamento del traffico sospetto, che riceve i pacchetti analizzati dai preprocessori, e collabora con i plugin di rilevamento per la loro dettagliata analisi e decodifica. Contemporaneamente, interpreta le regole, le applica ai dati che riceve e, in base alle direttive, compie determinate azioni
- plugin per l'output: estensioni aggiuntive per interfacciarsi correttamente con la destinazione in modo da generare l'output desiderato. Vengono invocati quando deve essere notificato qualche evento o registrare un allarme (ad esempio memorizzarlo in un database piuttosto che in un file di testo)

A questi elementi si aggiungono, per precisione, quelli relativi all'ambiente esterno con i quali Snort interagisce:

- sorgente dei dati: traffico di rete
- output: console, allarmi, log, syslog, database

Analisi del funzionamento tramite descrizione delle regole

Avendo introdotto i vari elementi che costituiscono Snort, il naturale passo successivo è quello di indicare come queste si coordinano fra loro in modo da descrivere come Snort effettivamente opera. Per far questo, è utile indicare brevemente com'è strutturata una regola nella sua forma basilare; in sostanza, è divisa concettualmente in due sezioni principali: la direttiva ed il corpo della regola.

La direttiva indica quale azione deve essere svolta in caso di confronto con esito positivo, e può essere di cinque tipi (elencati in ordine di priorità di analisi da parte di Snort):

- Alert: produce un allarme in base alle specifiche della regola, e poi registra il pacchetto nell'output desiderato
- Pass: ignora i pacchetti che non si vuole vengano ulteriormente analizzati
- Log: registra il pacchetto nell'output, senza generare l'allarme corrispondente
- Activate: produce un allarme, e poi attiva una regola di tipo dinamico
- Dynamic: rende la regola attiva solo quando viene attivata da un'altra regola tramite la direttiva Activate; a quel punto diventa una regola di tipo Log

A seguire, si incontra il corpo della regola, costituito da diversi campi quali:

- il protocollo utilizzato
- l'indirizzo IP della sorgente
- la porta della sorgente
- il simbolo di direzione della trasmissione
- l'indirizzo IP della destinazione
- la porta della destinazione

Per maggior completezza, c'è da dire che è possibile di definire anche variabili, intervalli di valori per vari campi, la totalità di valori assumibili da un campo attraverso la parola chiave "any", ecc..

Oltre alla forma essenziale di una regola formata dalla direttiva e dal suo corpo, esiste anche la possibilità di definire delle opzioni aggiuntive (racchiuse globalmente tra parentesi tonde) che in realtà sono il cuore della funzionalità di Snort. Infatti, vengono utilizzate per analizzare con maggior precisione e profondità diverse informazioni: permettono, ad esempio, di controllare il contenuto del pacchetto, la dimensione dei dati effettivi dal pacchetto, il valore del campo riservato al TTL (Time To Live), riportare specifici messaggi, sfruttare le opzioni relative ai preprocessori, ecc..

A questo punto, considerando anche gli elementi che costituiscono una regola, si procede con la descrizione dei passi che contraddistunguono il funzionamento di Snort.

Il pacchetto in arrivo dalla sorgente dei dati, viene rilevato e catturato dal sensore corrispondente; questo lo invia al decoder, che lo pone in coda informando che quel pacchetto è caratterizzato da un determinato protocollo. Successivamente viene inviato ai preprocessori, le cui funzioni principali sono:

- riassemblaggio di flussi TCP, in modo da rendere inutile un eventuale tentativo di nascondere un'intrusione sfruttando la tecnica della frammentazione dei pacchetti
- controllo dello stato delle connessioni
- eliminazione dei caratteri Unicode per facilitare l'applicazione successiva delle regole da parte del motore vero e proprio

Se non sono stati individuati tipologie di attacco, il pacchetto viene passato al motore che lo processa; questo, supportato dai plugin di rilevamento, procede con la decodifica e l'interpretazione delle regole (in particolar modo della parte riservata alle opzioni). Se il confronto ha successo viene applicata la direttiva della regola, e, tramite le estensioni per l'output, la notifica dell'avvenuta rilevazione viene registrata nella destinazione corrispondente.

Rappresentazione delle minacce

Esistono diverse modalità con le quali Snort produce gli eventi da notificare e memorizza gli allarmi. Quella che viene utilizzata principalmente nella Tesi è relativa alla registrazione degli allarmi in file di testo (i cosiddetti log) e sfruttando la modalità estesa di descrizione degli allarmi. L'opzione per ottenere ciò è rappresentata dal parametro "-A full".

Di conseguenza, i log utilizzati nella Tesi sono quelli prodotti e memorizzati da Snort secondo tale modalità, nella quale ogni allarme viene inserito nel log con una descrizione testuale dettagliata e completa di tutti i possibili elementi che rappresentano la minaccia rilevata dalla regola abilitata nella configurazione di Snort che ha prodotto quell'allarme. Questo è un esempio delle descrizioni di due allarmi che possono essere presenti in un log:

FIGURA

Come si può osservare, gli allarmi vengono memorizzati in ordine di tempo crescente (in base al timestamp) e ogni descrizione è separata dalle altre da una riga vuota.

3 Considerazioni preliminari per le tecniche di data mining

Gli allarmi prodotti dalla maggior parte dei sistemi IDPS (nel caso specifico della Tesi il sistema è Snort) possono venire registrati, in base ai desideri del responsabile della sicurezza in specifici file di log. La registrazione degli allarmi, con la corrispondente creazione di file di log, sta assumendo sempre più importanza riguardo la gestione di un sistema o di una rete, in quanto rappresentano un eccezionale fonte per determinare lo stato di salute di questi elementi. Molte aziende adottando persino un'infrastruttura riservata alla gestione di tale procedura.

Come descritto nel paragrafo 1.2, l'espansione sia delle caratteristiche e delle potenzialità riguardo un singolo sistema che della quantità e della tipologia di traffico rispetto ad una rete, ha prodotto un'enorme mole di dati che devono essere fruiti dal responsabile della sicurezza, e quindi sottoposti alla sua analisi tramite svariate modalità.

Di conseguenza, è opportuna l'applicazione di tecniche per rendere più efficace lo studio ed interpretazione dei dati e scoprire anche informazioni nascoste; le metodologie di data mining si rivelano sono la scelta più popolare per risolvere il problema di come ottenere questa conoscenza.

3.1 Proprietà dei dati di log di allarmi

Le tecniche di data mining più adatte da applicare devono venire opportunamente scelte, studiate e modificate per adattarsi nel modo migliore alle proprietà dei dati contenuti nei log. Perciò, è necessario innanzitutto analizzare le caratteristiche che contraddistinguono le informazioni all'interno dei log generati da Snort, che valgono anche nella maggior parte dei sistemi IDPS, e in termini ancora più generali, anche nella quasi totalità dei log esistenti.

In questa sezione, perciò, vengono discusse le proprietà speciali che devono essere considerate affinché le tecniche possono sfruttarle nel migliore dei modi, in modo da progettare accuratamente le stesse tecniche.

La natura dei dati da sottoporre a tali metodologie assume un ruolo di fondamentale importanza nel scegliere i giusti algoritmi; la maggior parte di essi sono progettati per data set generici dove non vengono effettuate assunzioni specifiche riguardo la natura dei dati. Nonostante ciò, quando si analizza il contenuto di tipici file di log a livello mondiale (e anche in questo caso di log prodotti da Snort), è stato dimostrato che si possono riscontrare delle caratteristiche che distinguono i dati di questi file di log da quelli di un generico data set.

In molteplici studi ([7], [9]), sono stati effettuati esperimenti su diversi log provenienti da differenti domini, e, sebbene sia impossibile dimostrare che queste proprietà si verifichino in un qualsiasi file di log, è facile prevedere che siano aspetti comuni ad una grande vastità di tali file. In questa Tesi viene testato se queste peculiarità si verificano anche per i log generati da Snort, e si è riscontrato che:

- la maggior parte delle parole occorre solo poche volte, ovvero la maggioranza dei termini è molto infrequenti, e una significativa percentuale di queste compare esattamente una sola volta (vale non

solo per i il campo relativo al timestamp ma anche per il resto delle parole). Inoltre, solo una piccola parte sono relativamente frequenti
- esistono molte correlazioni forti tra parole che occorrono frequentemente. Questo è causato dal fatto che l'allarme viene generalmente formattato in base ad un certo formato stringa prima di venir memorizzato; quindi, quando vengono memorizzati allarmi dello stesso tipo più volte, le parti costanti del formato stringa diventano parole frequenti che occorrono molte volte nel log. Si possono comunque verificare forti correlazioni anche tra campi corrispondenti alle parti variabili del formato di stringa (ad esempio gli indirizzi, le porte, ecc..)
- molte combinazioni di parole che occorrono frequentemente sono costituite da molti elementi

Da uno studio ancora più specifico riguardo gli elementi tipici che caratterizzano i dai all'interno di un log di Snort, i risultati ottenuti mostrano che:

- solo poche regole (tra quelle abilitate nella configurazione di Snort) generano la maggior parte degli allarmi
- le regole più prolifiche sono caratterizzate dalla tendenza di produrre allarmi per lunghi periodi di tempo; da questo si può ipotizzare che la quasi totalità degli allarmi legati a tali regole siano molto probabilmente delle false minacce oppure eventi di poca rilevanza

3.2 Caratteristiche generali delle tecniche di data mining implementate

3.2.1 Introduzione e struttura generale

Il riscontro delle proprietà appena descritte indica che possono facilmente esistere dei pattern comuni che occorrono nei log, e che spesso corrispondono a falsi allarmi o allarmi irrilevanti. Pertanto, l'applicazione di tecniche di data mining produce conoscenze preziose per la successiva futura classificazione degli allarmi. Per ciascuna di queste tecniche viene implementato un algoritmo per scoprire in modo efficiente i pattern, che deve quindi essere progettato opportunamente in modo da sfruttare e basarsi su queste proprietà speciali.

Le tecniche di data mining applicate nella Tesi consistono nel mining degli itemset frequenti e nel clustering. Si rivelano adatte nel ricercare regolarità in diversi data set, compresi i log di Snort, in quanto aiutano a rilevare i diversi tipi di pattern in un file di log che caratterizzano aspetti comuni, ma sono anche in grado di rivelare eventi anomali che rappresentano comportamenti inaspettati.

Gli algoritmi invece basati sulle regole di associazione vengono spesso usati per rilevare associazioni temporali tra tipi di eventi; ma spesso non possono essere applicati direttamente ai file di log, perché quest'ultimi non hanno un formato comune, oppure mancano alcuni attributi necessari all'algoritmo come il tipo di evento.

I due processi di data mining consistono ciascuno in tre passi principali:

- Preprocessamento dei dati
- Ricerca dei pattern

- Analisi dei pattern

Ciascun processo, all'occorrenza, può diventare di tipo iterativo; in base ai risultati dell'analisi, infatti, possono essere regolati o i parametri della fase di preprocessing (per esempio scegliendo i campi da mantenere per ogni allarme) oppure quelli corrispondenti agli algoritmi di mining (per esempio la minima soglia del supporto o della densità) con lo scopo di ottenere dei risultati in base alle aspettative desiderate.

Preprocessamento dei dati

L'input del processo globale consiste nei dati dei log riguardanti gli allarmi prodotti da Snort secondo la modalità descritta nel paragrafo 2.4.2; tali dati non possono venire utilizzati per scopi di data mining nella forma in cui sono memorizzati. Per questo motivo, devono venire opportunamente trasformati in questa fase in modo che l'output della conversione risulti appropriato e possa venire utilizzato come input degli algoritmi. I due algoritmi non richiedono diversi formati di input tra loro, e quindi la fase di preprocessamento fornisce un unico tipo di dato in output.

La procedura di preprocessamento consiste in due fasi:

- la rimozione delle parti non significative quali i diversi simboli e tutti quei campi che non rappresentano informazioni necessarie per l'algoritmo. Gli attributi, i cui valori devono rimanere nella descrizione di ogni allarme in quanto fondamentali per la fase di ricerca dei pattern, sono:
    - l'identificatore
    - l'indirizzo IP della sorgente
    - la porta della sorgente
    - l'indirizzo IP della destinazione
    - la porta della destinazione
    - il protocollo utilizzato
- la modifica di alcuni dei campi rimanenti per separare alcuni valori che sono accorpati in un'unica stringa (indirizzo IP e relativa porta) e il riordino dei vari elementi secondo l'elenco indicato nel punto precedente in modo da avere una rappresentazione significativa

E' stato implementato un semplice programma in python che svolge questa funzione; di seguito viene mostrato un esempio di una descrizione di un allarme in input al programma ed il risultato corrispondente in output.

FIGURe

Ricerca dei pattern

La ricerca dei pattern consiste nell'applicare le tecniche di data mining i metodi per la ricerca degli aspetti comuni nei dati dei log. Oltre ai dati in input, sono necessari diversi parametri, in particolare il valore della soglia minima del supporto nel caso della prima tecnica o quello della soglia minima della densità riguardo la seconda metodologia. La scelta appropriata di tali valori rappresenta uno dei problemi chiave; la soluzione migliore è quella di procedere con le diverse iterazioni del processo globale da parte del responsabile della sicurezza fino a quando viene trovato il valore più adatto. In questa fase, è necessaria quindi l'interazione del responsabile, e quindi è consigliabile l'esecuzione iterativa degli algoritmi di ricerca dei pattern su una parte relativamente ristretta dell'intero log. Scegliendo la giusta dimensione da prendere come campione, il tempo di risposta dell'algoritmo rimane piccolo, mentre il campione

rappresenta comunque, in modo significativo e accurato, l'intero log. Settare il valore del parametro relativo a queste soglie non è una procedura da sottovalutare, e richiede molta pratica e attenzione da parte del responsabile.

Al termine di questa fase, vengono generati i cosiddetti pattern, ciascuno rappresentato da una serie di valori situati in una determinata posizione del pattern in quanto relativi al corrispondente attributo (i valori sono quelli riferiti agli attributi rimasti dopo la fase di preprocessamento). Alcuni valori possono non riferirsi ad un singolo specifico elemento, ma a molteplici, e quindi, per riassumerli, viene utilizzato il simbolo asterisco.

Analisi dei pattern

La fase di analisi dei pattern consiste nel comprendere i risultati ottenuti dagli algoritmi e trarne le conclusioni, in modo da aiutare il responsabile della sicurezza ad adottare le giuste decisioni. Queste informazioni vengono sfruttate anche dalle metodologie implementate successivamente per la classificazione degli allarmi ed il rilevamento di quelli considerati false insidie.

3.2.2 Mining specifico dei log

In generale, le tecniche di data mining, se applicate ad un log, assumono che ogni voce di un log (in questo caso gli allarmi di Snort caratterizzati dallo loro descrizione) sia caratterizzato da due attributi: il tempo di occorrenza e il tipo. Per entrambi ci sono due possibilità e quindi, in base a queste due proprietà, si possono ottenere quattro combinazioni finali.

Per quanto riguarda il primo le due opzioni sono le seguenti:

- il log viene mantenuto nella sua unicità, senza subire nessuna suddivisione interna. Ciascun allarme che lo compone ha il suo periodo singolo di occorrenza e non viene inserito in un'insieme di allarmi appartenenti ad un determinato lasso di tempo
- il log viene suddiviso in un fissato numero di finestre, ciascuna contenenti determinati allarmi in base a certi criteri.

Nella Tesi viene utilizzata la prima opzione, che, è la più semplice ed adatta per il compito di data mining. La seconda opzione viene utilizzata nella parte di rilevamento dei falsi allarmi, in quanto, essendo una soluzione più sofisticata, si presta meglio per uno scopo più delicato e preciso.

Nel log, come già accennato nel paragrafo 2.4.2, gli allarmi sono presenti in ordine crescente secondo il tempo di occorrenza; quelli che presentano lo stesso tempo di occorrenza vengono ordinati poi in base ad altri attributi. L'ordine degli eventi viene spesso preso in considerazione quando si applica il mining perché può rivelare relazioni casuali tra tipi di eventi. Tuttavia, anche il mining, senza la considerazione di questo aspetto, è caratterizzato dalla medesima importanza; ciò è motivabile dal fatto che gli eventi memorizzati possono non rispecchiare la realtà dal punto di vista cronologico, a causa di latenze presenti nella rete, clock di sistema non perfettamente sincronizzati oppure gli eventi vicini tra loro possono non essere collegati da un legame di tipo causa-effetto.

A riguardo del secondo parametro, ci sono due corrispondente alternative:

- trattare il log come un insieme di elementi semplici (definiti di tipo “raw”), ovvero senza essere contraddistinti da particolari tipi di evento; perciò il mining effettuato su questi elementi viene definito come il mining orientato alle righe che compongono il log
- assumere che il log rappresenti un insieme di eventi (ad esempio gli eventi di una sessione ftp di un utente). Di conseguenza, il mining corrispondente viene indicato come il mining orientato ai tipi di evento

Nella Tesi viene scelta la prima possibilità, siccome i log prodotti da Snort contengono dati quali allarmi che, sia dal punto di vista concettuale che come struttura di formato, si prestano favorevolmente a tale assunzione e non ad essere trattati come eventi. Inoltre, è spesso difficile attuare la seconda opzione siccome le voci in log difficilmente contengono codici espliciti che descrivono il tipo di evento, bensì bisogna derivarli riconoscendo ad esempio dei pattern comuni che si ripetono.

Con la selezione della prima opzione (mining sulle righe) il termine riga, d’ora in poi, è sinonimo di allarme, sebbene la descrizione di quest’ultimo consista in diverse righe nel log.

A questo punto, se viene considerato un file di log L, dove ogni riga rappresenta un allarme A, allora L diventa una sequenza finita di m allarmi $L = (A^1,\ldots,A^m)$

Ogni allarme A può essere definito come una tupla formata dai valori che assumono determinati attributi in base a quell’allarme. Un attributo viene definito dal simbolo α, mentre il valore di un determinato attributo relativo ad un certo allarme A come A_α. Ad esempio Id è l’attributo relativo al codice identificativo di un allarme, e A_{id} è il valore assunto dall’allarme A per tale attributo

Se si tiene presente che gli attributi da considerare utili per la fase di mining (quelli i cui valori rimangono dopo la fase di preprocessamento dei dati) sono:

- l’identificatore (id)
- l’indirizzo IP della sorgente (srcIP)
- la porta della sorgente (srcPort)
- l’indirizzo IP della destinazione (dstIP)
- la porta della destinazione (dstPort)
- il protocollo utilizzato (proto)

allora ogni allarme A può essere rappresentato dalla tupla:

$A = (A_{Id}, A_{srcIP}, A_{scrPort}, A_{destIP}, A_{destPort}, A_{proto})$

Per distinguere però valori identici appartenenti ad attributi differenti, è opportuno assegnare ad ogni attributo il numero della posizione che esso assume nella riga; quindi ogni allarme A diventa una tupla formata da una serie di elementi ciascuno costituito dalla coppia valore attributo-numero posizione:

$A = \{(A_{Id},1),(A_{srcIP},2),\ldots\}$

Allo stesso modo, al termine della ricerca dei pattern, ogni pattern P può essere considerato una tupla costituita da una successione di elementi ognuno rappresentato dalla stessa tipologia di coppia:

$P = \{(P_{id},1),(P_{scrIP},2),\ldots\}$

Da ricordare che ad un attributo per un pattern P possono essere associati diversi elementi ed in quel caso tali elementi vengono indicati tramite il carattere asterisco (A_{attr} = '*'). Se invece il valore riguardo un attributo è unico, l'elemento corrispondente viene indicato: in questo caso si dice che il pattern P possiede l'attributo alfa, ed il suo valore è P_{α}.

Tali definizioni vengono sfruttate nei capitoli successivi.

4 Mining degli itemset frequenti

4.1 Necessità di una nuova tecnica e formulazione iniziale del problema

Innanzitutto è opportuno definire il problema del mining degli itemset frequenti: dato il database delle transazioni D e soglia minima del supporto pari a s, lo scopo è quello di cercare tutti gli itemset frequenti ed il supporto corrispondente a ciascuno di essi.

Le tecniche di data mining proposte di recente si basano per la maggior parte sull'utilizzo dell'algoritmo Apriori per il mining degli itemset frequenti, ed in particolare per quelli orientati ai tipi di evento. Tali scelte, però, non sono consone ad essere applicate ai log di Snort in quanto contraddistinte da diversi aspetti negativi (perché contrari alle proprietà e scelte descritte nel capitolo precedente):

- spesso ignorano il mining orientato alle righe
- sono inefficienti dal punto di vista computazionale quando operano su elementi frequenti di discreta dimensione
- non riescono a rilevare gli itemset infrequenti, utili in certi casi per rilevare anomalie soprattutto nel campo della sicurezza di rete

Quindi, è necessario progettare un algoritmo di mining degli itemset frequenti che riesca a sfruttare invece tutte le peculiarità che contraddistinguono i dati presenti nei log di Snort.

Se si considera il log di Snort come un database di transazioni, è possibile formulare il problema del mining orientato alle righe come il problema del mining degli itemset frequenti. Quindi, il mining degli itemset frequenti può essere attuato effettuando il mining orientato alle righe.

Per prima cosa, viene preso in considerazione un file di log L per essere considerato opportunamente come un database di transazioni, ovvero D (più specificatamente D(L)), in modo da poter applicare a questo dataset l'algoritmo di mining degli itemset frequenti.

Dalla definizione finale data nel capitolo precedente, ogni allarme A deve essere considerato una tupla costituita da una serie di elementi (in questo caso vengono definiti item) ognuno formato dalla coppia valore attributo-numero posizione:

$A = \{(A_{Id},1),(A_{srcIP},2),\ldots\}$

Considerando il file di log L come un database di transazioni D(L), tutte le parole distinte (senza ripetizioni) che lo compongono rappresentano l'insieme di tutti i possibili n item $I = \{i_1,\ldots,i_n\}$. Quindi, ciascuna singola riga (che consiste in un allarme definito A) è una transazione, ovvero una tupla (t_{id}, X) dove:

- t_{id} è l'identificatore della transazione (corrispondente al numero di riga nel log)
- X è un itemset $X = \{x_1,x_2,\ldots,x_n\}$ formato da tutte le parole che costituiscono la riga, ed è un sottoinsieme dell'insieme di tutti gli item I.

Se l'allarme viene trattato come l'i-esima riga, allora l'i-esimo allarme A rappresenta la i-esima transazione, ovvero la tupla (m,X) dove X = {(A_{Id},1),(A_{srcIP},2),…}.

Andando più in dettaglio riguardo gli elementi tipici quando di parla di data mining, bisogna trattare anche altre definizioni:

- Cover di un itemset X: insieme di identificatori di transazioni che contengono al loro interno lo stesso itemset X, ovvero:
  $cover(X) = \{ t_{id} \mid (t_{id},Y) \in D, X \subseteq Y\}$
- supporto di un itemset X: numero di elementi della sua cover
  $supp(X) = |cover(X)|$
- X è itemset frequente se s è la soglia minima del supporto e $supp(X) \geq s$. Il valore del supporto spesso consiste in una percentuale p e quindi $s = |D|*p/100$
- X itemset di tipo closed: non esiste nessun superset di X con lo stesso supporto
- FClosed(D,s): insieme degli itemset frequenti di tipo closed per il database D e con soglia minima del supporto s

4.2 Descrizione dettagliata dell'algoritmo

L'algoritmo di mining degli itemset frequenti presente nella Tesi combina le caratteristiche vantaggiose sia degli algoritmi basati su una ricerca che di quelli fondati su una ricerca in profondità, tenendo inoltre in considerazione le importanti proprietà dei dati presenti nei log discusse nel capitolo 3.

Innanzitutto, si basa su una strategia di ricerca in ampiezza e mantiene gli itemset frequenti in una struttura dati ad albero; in questo modo risolve la dipendenza dalla quantità di memoria che caratterizza gli algoritmi basati su una ricerca invece in profondità. Inoltre, vengono adottate delle tecniche che riducono la quantità di informazioni da tenere in considerazione ed analizzare, diminuendo così sensibilmente il consumo di memoria, l'efficienza e la velocità l'esecuzione (evitando quindi le debolezze di Apriori).

Inoltre, si concentra sugli itemset frequenti di tipo closed, in quanto consistono in una versione compatta ma che mantiene le stesse informazioni dell'insieme degli itemset frequenti. In questo modo, vengono descritti i pattern degli allarmi frequenti che non solo identificano itemset frequenti, ma rivelano anche associazioni forti tra tutti i tipi dei valori degli attributi.

E' opportuno ricordare che l'algoritmo di mining si basa sulla ricerca degli itemset frequenti all'interno di log. Questi sono costituiti da un insieme di righe, le quali, a loro volta, sono formate da un insieme di parole. Come detto precedentemente, l'insieme di righe deve essere trattato come un insieme di transazioni e l'insieme di parole come un insieme di item. Perciò, d'ora in avanti il termine transazione viene utilizzato per definire una riga, e la stessa cosa con il termine item per identificare una parola. L'item, però, è qualcosa di realmente più complesso e la parola è solamente una sua componente; perciò, è importante definire cosa s'intende per item, e cioè un tipo di dato costituito principalmente da:

- la stringa che rappresenta la parola stessa rilevata nel log
- la tipologia dell'item, che consiste essenzialmente nella posizione che assume la parola all'interno della riga. Tale campo è molto importante in quanto ad una determinata posizione corrisponde una determinata tipologia di attributo, e quindi di item.

- il codice identificativo dell'item, definito da una stringa formata dalla posizione che l'item assume all'interno della riga concatenata con la stringa che rappresenta la parola
- il contatore del numero di occorrenze rilevate all'interno del log
- l'indice che contraddistingue l'item all'interno di una struttura dati (definita successivamente) che rappresenta tutti gli item frequenti

Ora viene analizzato l'algoritmo descrivendo le fasi che lo costituiscono e quindi analizzandolo passo per passo.

4.2.1 Mining dei singoli item frequenti

Come ogni algoritmo che si basa sulla ricerca in ampiezza, il primo passo è il mining degli item frequenti, in modo da creare la base di partenza per il mining più complesso.

Inizialmente viene effettuata una prima scansione delle transazioni presenti nel log per contare quante volte ogni singolo differente item compare all'interno del log, ovvero il numero di occorrenze. Questo procedimento può impiegare un grande consumo di memoria in quanto il numero di item differenti può essere facilmente elevato e quindi mantenere i diversi contatori nella memoria principale potrebbe rivelarsi un problema. Inoltre, se esistono molti item poco frequenti, la memorizzazione dei corrispondenti contatori si rivela uno spreco di spazio. Tuttavia, la scansione completa del log è una scelta obbligata, altrimenti è praticamente impossibile affermare con certezza quali sono gli item frequenti.

La soluzione è quella di prendere in considerazione solo gli item potenzialmente frequenti, cioè quelli che potrebbero rivelarsi come tali, ed escludere subito quelli che sicuramente che non possono esserlo. Per attuare ciò, l'algoritmo effettua una prima scansione in modo da costruire il cosiddetto wordsVector, una struttura dati di tipo vettore con un numero m configurabile di celle, ciascuna contraddistinta da due campi:

- una lista di item che rappresenta l'insieme di item accomunati da una certe caratteristica
- il contatore delle occorrenze associato a quell'insieme (inizializzato al valore zero)

L'aggiornamento di questi campi viene effettuato sfruttando una funzione hash. Durante la prima scansione infatti, ad ogni item rilevato viene applicata una funzione hash opportunamente realizzata che genera un valore intero compreso i tra 0 e m-1. A questo punto la il valore presente nel contatore alla pozione i del vettore wordsVector, ovvero l'i-esimo contatore, viene incrementato di un'unità; per il momento non viene comunque aggiunto anche l'item corrispondente alla lista di quella posizione, la quale quindi rimane vuota.

Per maggior precisione, la funzione hash per ogni item viene applicata al codice identificativo dell'item; questo viene fatto per identificare in modo preciso un determinato item senza possibilità di incorrere a delle interpretazioni errate che possono verificarsi quando una stessa stringa appare in differenti posizioni all'interno di diverse righe.

Detto ciò, si può prendere in considerazione l'insieme i_1, i_2,…, i_j, composto da tutti gli item caratterizzati dal fatto che la funzione hash applicata a ciascuno di essi genera il medesimo valore i. Dunque, una volta terminata la creazione del vettore, alla posizione i-esima il contatore contiene la somma totale di tutte le occorrenze di ciascun item appartenente a quell'insieme.

Questo avviene anche grazie all'utilizzo di un'efficiente funzione hash che distribuisce i valori in ingresso in modo uniforme rispetto alla gamma di valori in uscita; di conseguenza, se sono presenti n differenti parole nel log, ogni contatore corrisponde approssimativamente a n/m item.

Terminata la prima scansione e quindi l'aggiornamento di tutti i contatori del vettore, si procede con una seconda scansione. Ad ogni item letto dal log viene applicata nuovamente la funzione hash che ritorna il valore i. Se alla posizione i del vettore wordsVector il valore contenuto nel contatore è pari o superiore alla soglia minima del supporto impostata, l'item può essere un potenziale item frequente; perciò, viene aggiunto nella lista relativa alla posizione i-esima del wordsVector ed il contatore delle occorrenze interno dell'item viene incrementato di volta in volta. Al contrario, l'item non viene considerato e si passa al successivo. Tale item quindi non verrà più preso in considerazione neppure in futuro, siccome un item infrequente non potrà mai far parte di un itemset frequente. Ciò è dovuto al fatto che il valore nel contatore rappresenta la somma delle occorrenze di diversi item: se tale somma non raggiunge il supporto, neppure ciascuna sua singola componente (l'occorrenza di ciascun singolo item) riesce ovviamente a farlo.

Siccome, come detto nel capitolo 3 a riguardo della proprietà dei dati contenuti in questi log, gran parte degli item compaiono solo poche volte, l'attuazione di questa procedura fa si che molti valori presenti nei contatori non riescano a raggiungere il supporto minimo, quindi molti item vengano giustamente ignorati, ottenendo un forte risparmio in termini di occupazione della memoria.

Al termine della seconda scansione, ogni elemento del wordsVector con valore del contatore non inferiore alla soglia minima del supporto, contiene nella propria lista gli item che rientrano in quella posizione in base all'applicazione della funzione hash, e ciascun item è caratterizzato anche dal proprio numero di occorrenze.

A questo punto, viene effettuata una scansione del wordsVector, andando ad analizzare tutti gli item presenti nelle liste non vuote delle varie celle. Se il valore del contatore interno dell'item è inferiore al supporto minimo, viene eliminato dalla lista, altrimenti viene mantenuto, per poi passare al'item successivo. Al termine di questo passo, le varie liste all'interno del wordsVector contengono solamente gli item veramente frequenti, i quali vengono inseriti ad uno ad uno in una struttura vocabulary di tipo vettore per avere una visione più precisa ed ordinata dei vari item. Essi vengono prelevati dal wordsVector (in base all'ordine in cui vengono incontrati) ed inseriti progressivamente in questa nuova struttura. Ad ogni item viene aggiornato il campo relativo alla posizione di inserimento in questa stessa struttura, informazione utile per i passi successivi.

Se l'insieme degli gli item frequenti appena calcolato è vuoto, l'algoritmo termina, altrimenti si passa alla creazione della vera e propria struttura che viene sfruttata per il mining.

4.2.2 Creazione delle strutture per il mining

Lo scopo è quello di costruire le cosiddette struttura trie degli intemset, ovvero strutture dati compatte che memorizzano i vari itemset in modo utile per effettuare ricerche efficienti di tali elementi. Per comodità, per definire una struttura di questa tipologia viene utilizzato semplicemente il termine albero, anche se consiste in una struttura più complessa.

L'algoritmo Apriori, come già descritto nella sezione 4.1, non è molto adatto per processare data set che contengono itemset frequenti la cui dimensione è elevata. La soluzione adottata è costituita da una tecnica

in grado di ridurre la dimensione degli alberi, in modo che continui a rappresentare tutti gli itemset frequenti ma la spazio di ricerca sia minore.

L'idea di partenza è che nelle transazioni esistono molte forti correlazioni tra gli itemset frequenti; di conseguenza, utilizzando Apriori, l'albero creato presenta diversi rami con informazioni che si ripetono. Lo scopo della tecnica è quello di concentrarsi solo su quei rami che contengono dati unici, che non si verificano in altri settori dell'albero.

Innanzitutto è opportuno ricordare che la cover di un itemset X = $\{x_1,x_2,\ldots,x_n\}$ rappresenta l' insieme di identificatori di transazioni che contengono al loro interno lo stesso itemset X.

Se si considera F = $\{f_1,f_2,\ldots,f_m\}$ l'insieme degli item frequenti, si può definire l'insieme di dipendenza (dependency set [6]) di un item f_i, come l'insieme di tutti gli item f_j, differenti da f_i, la cui copertura contiene la copertura dell'item f_i, ovvero:

$$ins.di\ dipend.(f_i) = \{f_j \mid f_j \neq f_i, cover(\{f_i\}) \subseteq cover(\{f_j\})\}$$

In pratica, quando in una transazione è presente l'item f_i, si è certi che nella stessa transazione sono presenti anche tutti gli item f_j contenuti nel suo insieme di dipendenza, ovvero quando incontro f_i c'è la certezza che avrò anche i vari f_j. In definitiva, contiene gli item cooccorrenti (definiti anche cooccorrenze o dipendenze) di un determinato item. La dimensione m dell'insieme di dipendenza per l'item f_i indica che l'item stesso f_i presenta m dipendenze.

Il prefisso di dipendenza (dependency prefix) di un item f_i, viene invece definito come l'insieme di tutti gli item f_j appartenenti all'insieme di dipendenza di f_i, che si trovano in posizione inferiore in base ad un determinato ordinamento:

$$pref.di\ dipend.(f_i) = \{f_j\ \epsilon\ ins.di\ dipend.(f_i), f_j < f_i)\}$$

In pratica, quando in una transazione è presente l'item f_i, nella stessa transazione sono presenti anche tutti gli item (dipendenze) e questi si trovano in una posizione antecedente ad i. L'ordinamento può basarsi su diversi parametri; quello più generale è la posizione assunta all'interno della transazione (ad esempio in una transazione T = {a,b,d,f} si a che a<b<c<d). Altri tipi di ordinamento possono essere attuati in base al valore del supporto, alla dimensione degli insiemi di dipendenza (come impiegato nella Tesi in seguito), ecc..

Se si considera un itemset $\{f_1,f_2,\ldots,f_m\}$ invece che un singolo item f_i, il suo prefisso di indipendenza viene definito come l'unione dei prefissi di dipendenza dei singoli item che compongono l'itemset, ovvero:

$$pref.di\ dipend.(\{f_1,f_{2,\ldots,}f_m\}) = \bigcup_{j=1}^{k} pref.di\ dipend.(f_j)$$

Da quest'ultima definizione scaturiscono alcune importanti proprietà:

- se un itemset X contiene il suo prefisso di dipendenza, allora anche lo stesso itemset dal quale viene eliminato l'elemento alla posizione più grande conterrà il prefisso di dipendenza di questo itemset appena creato, ovvero:

$$se\ pref.di\ dipend.(\{f_1,f_{2,\ldots,}f_m\}) \subseteq \{f_1,f_{2,\ldots,}f_m\}$$
$$allora\ pref.di\ dipend.(\{f_1,f_{2,\ldots,}f_{m-1}\}) \subseteq \{f_1,f_{2,\ldots,}f_{m-1}\}$$

- se un itemset X contiene il suo prefisso di dipendenza, allora il supporto dell'itemset coincide con il supporto calcolato sull'itemset dal quale viene eliminato il prefisso di dipendenza dello stesso itemset, cioè:

$$se\ pref.di\ dipend.(X) \subseteq X \quad allora \quad supp\big(X \setminus pref.di\ dipend.(X)\big) = supp(X)$$

  Questo è derivato sfruttando la regola della transitività.

La tecnica quindi consiste principalmente nell'evitare di creare un nodo nell'albero per un certo itemset se questo non contiene il suo prefisso di dipendenza, perché l'informazione corrispondente risulta inutile.

4.2.3 Creazione degli insiemi di dipendenza

A questo punto risulta necessario, per ciascuno degli itemset frequenti, calcolare il corrispondente insieme di dipendenza e la sua dimensione.

Innanzitutto viene effettuata una nuova scansione e vengono utilizzate diverse strutture dati di tipo vettore (è utile ricordare che si è scelto di mantenere gli indici di vocabulary per identificare i vari item).

Per ogni transazione rilevata, viene creato un vettore (localIndexDepSet) che mantiene gli indici legati alla posizione in vocabulary degli item frequenti riscontrati in quella stessa transazione (mantiene le informazioni solo per quella transazione, quindi le dipendenze locali). Non si tratta di un insieme di dipendenza per un determinato item, in quanto contiene gli indici di tutti gli item frequenti della transazione (a livello locale), ma durante l'esecuzione, andando ad analizzare di volta in volta ciascun item che lo compone, permette successivamente di ottenere (come si vedrà poi) i veri e propri insiemi di dipendenza.

Durante l'esecuzione, il vettore localIndexDepSet interagisce con una matrice globalIndexDepSet, che opera secondo questo principio: alla riga i viene mantenuto il vettore dell'insieme minimo degli indici (posizione j in vocabulary) degli item frequenti che fino a quel momento sono stati rilevati quando è presente anche l'item frequente in analisi (alla posizione i in vocabulary), ovvero le dipendenze a livello globale.

Inoltre viene sfruttato un vettore indexPresence settando ad 1 le celle in corrispondenza degli indici (posizioni in vocabulary) degli item frequenti rilevati.

Come struttura di supporto, viene utilizzato un vettore (depSetIndexCounters) che opera in maniera simile alla matrice globalIndexDepSet, in quanto alla posizione i viene mantenuto il numero minimo di item frequenti che fino a quel momento (aggiornato all'avanzare della scansione delle transazioni) sono stati rilevati quando è presente anche l'item frequente che si sta analizzando (alla posizione i in vocabulary); si tratta quindi del numero di dipendenze a livello globale.

All'inizio dell'analisi per un certo item (alla posizione i in vocabulary), alla riga i della matrice globalIndexDepSet è presente lo stesso vettore localIndexDepSet , mentre in depSetIndexCounters[i] si trova il numero totale di item frequenti rilevati nel localIndexDepSet.

Durante l'analisi viene verificato che tutti gli elementi a livello globale (ovvero quelli contenuti in globalIndexDepSet[i]) siano presenti anche a livello locale (in localIndexDepSet) attraverso la verifica della loro esistenza o meno tramite indexPresence. Se non tutti gli elementi a livello globale (fino a quel

momento) sono presenti anche a livello locale, viene aggiornato sia il numero minimo di item frequenti (DepSetIndexCounters[i]) che gli elementi in globalIndexDepSet[i], mantenendo solo quelli che continuano ad essere presenti anche a livello locale.

Al termine dell'esecuzione, in globalIndexDepSet[i] è presente l'insieme di dipendenza minimo (costituito dagli indici degli item) e in depSetIndexCounters[i] il valore finale della dimensione minima degli insiemi di dipendenza, entrambi riguardanti l'item frequente situato alla posizione i di vocabulary.

4.2.4 Creazione dei prefissi di dipendenza

Il principio si basa sul riordinamento degli item frequenti in base all'ordine crescente del grado di dipendenza, ovvero rispetto alla posizione in una "classifica" stilata in base alla dimensione crescente degli insiemi di dipendenza; ciò consiste in un importante perfezionamento riguardo l'efficienza della tecniche di riduzione applicate. Un'opzione alternativa poteva essere quella di ordinarli in base al supporto, ma in questo modo, se in una transazione un item si verifica precedentemente rispetto ad un altro, anche la dimensione dell'insieme di dipendenza del primo item sarà minore della dimensione legata all'insieme di dipendenza del secondo item. Scegliendo la prima possibilità, viene incrementata l'efficienza della riduzione dell'albero, in quanto la probabilità che il prefisso di dipendenza contenga tutti gli elementi dell'insieme di dipendenza aumenta.

Come primo passo, sfruttando le dimensioni degli insiemi di dipendenza di ogni item frequente (DepSetIndexCounters), si procede con la creazione di una struttura di tipo vettore (OrderDepSetSize) formata da due elementi:

- indice dell'item in vocabulary
- dimensione dell'insieme di dipendenza dell'item

Facendo così si ottiene la corrispondenza tra ciascun item e la dimensione del corrispondente insieme di dipendenza. Gli elementi del vettore, inoltre, vengono ordinati in modalità crescente in base alla dimensione dell'insieme di dipendenza, in modo da ottenere un ordinamento $x_1<x_2<\ldots<x_n$. Questa può essere vista come una sorte di classifica, dall'ultimo classificato fino al primo, e quindi ad ogni posizione corrisponde un piazzamento, che viene sfruttato per creare la struttura successiva.

Infatti, OrderDepSetSize viene utilizzato per creare il vettore OrderDepSetRank: alla posizione i corrisponde l'indice dell'item in vocabulary, ed il suo contenuto rappresenta la posizione (valutata in ordine crescente) che ciascun item assume in base alla dimensione del suo dependency set riscontrata in OrderDepSetSize. Ovvero se l'item ad indice i in vocabulary si trova all'indice j in OrderDepSetSize, alla posizione i di OrderDepSetRank è contenuto il numero j+1, indicante che l'item in posizione i in vocabulary si classifica come il j+1-esimo elemento tra i più frequenti (tenendo presente però che questa sorte di "classifica" viene stilata in ordine crescente). In questo modo, viene semplificato il lavoro per ottenere l'obiettivo prefissato, ovvero il calcolo dei prefissi di dipendenza per ciascun item.

A questo punto viene effettuata la scansione del vettore OrderDepSetRank, partendo dall'elemento che si è classificato con la dimensione più piccola tra gli insiemi di dipendenza. Anche qui vengono utilizzate diverse strutture dati di tipo vettore; a differenza di quanto adottato per l'identificazione degli item nella costruzione degli insiemi di dipendenza, in questa occasione si è scelto di utilizzare la posizione in "classifica" in OrderDepSetRank.

Per ogni item classificato in posizione i in OrderDepSetRank, viene creato un vettore (localIndexDepPrefix) che mantiene le posizioni in "classifica" degli item frequenti tra quelli presenti nel suo globalIndexDepSet[i] e situati ad una posizione inferiore rispetto ad i (sfruttando OrderDepSetRank). Le informazioni riguardano solo quell'item, quindi a livello locale.

Ciascun item di globalIndexDepSet[i], rappresentato dell'indice j, deve quindi essere analizzato e solo quelli che sono caratterizzati da una posizione inferiore rispetto all'item i (sfruttando la "classifica" di OrderDepSetRank) possono essere inseriti in localIndexDepPrefix. Per tutti quelli che soddisfano tale condizione, si ricorre (come nella creazione degli insiemi di dipendenza) all'utilizzo di una matrice globalDepPrefix: alla riga i viene mantenuto il vettore appena creato localIndexDepPrefix, e quindi la matrice conserva le informazioni globali riguardanti i vari prefissi di dipendenza.

Inoltre, deve essere creato un vettore globalDepPrefixSize, il cui indice i corrisponde alla posizione in "classifica" in OrderDepSetRank (e non agli indici di vocabulary), e al suo interno viene mantenuto il numero (incrementato di volta in volta, quindi a livello globale) degli item rilevati nel vettore localIndexDepPrefix.

Al termine dell'esecuzione, in globalDepPrefix[i] è presente il prefisso di dipendenza (costituito dalle posizioni in "classifica" degli item) e in globalDepPrefixSize[i] il valore della dimensione dello tesso prefisso di dipendenza, entrambi relativi all'item frequente classificato alla posizione i di OrderDepSetRank. Gli indici di globalDepPrefix e di globalDepPrefixSize consistono nelle posizioni rilevate in OrderDepSetRank; di conseguenza tali strutture sono (per costruzione) ordinate in base a tale posizione e questa caratteristica è fondamentale per la creazione successiva degli alberi.

Infine, la struttura vocabulary subisce una trasformazione in quanto alla posizione i non corrisponde più l'indice stabilito inizialmente ma la posizione in classifica. Alla posizione i è presente il valore che era presente all'indice j (ricavato in base a OrderDepSetRank) del vocabulary iniziale. Viene adottata questa scelta per poter lavorare più velocemente con vocabulary sotto l'ottica della posizione in classifica, in quanto d'ora in avanti le informazioni importanti sono rappresentate principalmente dai prefissi di dipendenza. Infatti, le strutture che verranno ancora utilizzate saranno vocabulary, globalDepPrefix e globalDepPrefixSize.

4.2.5 Creazione dell'albero delle transazioni

Questo passo consiste nell' effettuare una scansione del log, ed inserire le informazioni fondamentali di ciascuna transazione rilevata in una struttura ad albero, definita transTrie. Le informazioni importanti sono costituite dagli item frequenti e dalla loro posizione in "classifica" rispetto alla dimensione dei propri prefissi di dipendenza. In questo modo si ottiene una struttura semplice che non contiene item non frequenti, quindi informazioni non rilevanti, e che può essere sfruttata agevolmente nei passi successivi dell'algoritmo.

Il procedimento consiste quindi nel ricavare da ciascuna transazione l'itemset $F = \{f_1,f_2,\ldots, f_m\}$ formato esclusivamente dagli item frequenti contenuti nella stessa transazione; da questo insieme di item viene ricavato l'insieme $P = \{p_1,p_2,\ldots,p_m\}$ delle corrispondenti posizioni in "classifica" dei vari item che lo compongono. Tale insieme viene raccolto in un vettore ed ordinato per essere inserito in modo corretto nel transTree, ovvero le posizioni devono essere collocate in un determinato cammino $p_1,p_2,\ldots,p_m$ dell'albero

secondo determinate regole, fino a formare il nodo N(p_1,p_2,…,p_m) caratterizzato dal contatore con valore m, dove m è il numero di transazioni che contengono l'itemset F.

Sostanzialmente, ogni elemento del vettore ordinato dell'itemset { p_1,p_2,…,p_m } viene inserito secondo un certo ordine all'interno del transTree e nella corretta posizione seguendo il principio che nodi dello stesso livello devono essere ordinati in senso crescente da sinistra a destra riguardo valore della posizione in classifica. Trovandomi in un nodo N del transTree caratterizzato da n figli, considerando il figlio i-esimo e dovendo inserire il nodo p, si ha che:

- se i > n oppure (classif. di p) < (classif di figlio i) -> viene creato un nuovo nodo nella posizione i e viene considerato come punto di partenza il figlio i
- se (classif. di p) > (classif di figlio n-1) -> viene creato un nuovo nodo nella posizione n-1 e si considera come punto di partenza il figlio n-1
- viene effettuata una ricerca binaria tra la posiz i e n-1, dimensione totale pari a (n-1 -i +1) e posizione mediana pari a ((n-1 –i +1)/2)
  Se (classif. di p) = (classif di figlio k) -> il nodo è già presente quindi non ne viene creato uno nuovo; altrimenti viene creato un nuovo nodo quando (classif di p) < (classifi figlio k) o dimensione totale è pari a 1
  Viene considerato come punto di partenza il figlio k

Infine, per ogni cammino creato, il valore del contatore del nodo N viene incrementato di un'unità se già presente, altrimenti viene settato ad uno in caso di nuovo inserimento.

Albero delle transazioni ridotto

Nel procedimento di creazione del tranzTree, ad ogni transazione rilevata vengono creati gli itemset formati dagli item frequenti della transazione per essere aggiunti (se non presenti) nel transTree, indipendentemente dal numero di volte che questi itemset compaiono nelle varie transazioni, ovvero senza tener conto della dimensione della loro copertura.

Come ulteriore miglioria è possibile scremare il numero di itemset da inserire nel transTree, aggiungendo solamente quelli che compaiono in almeno sufficiente di transazioni. L'idea è quella di trovare l'insieme di intemset formati da item frequenti che compaiono almeno m volte nelle varie transazioni , ovvero la cui dimensione della copertura è uguale o maggiore ad m.

L'obiettivo consiste nel cercare di inserire nel tranTree solamente le informazioni riguardanti le transazioni che si verificano più spesso, in modo tale che la struttura transTree riesca a risiedere nella memoria principale e siano presenti i dati delle transazioni che vengono maggiormente utilizzati durante il mining. L'albero transTree, infatti, deve venire frequentemente utilizzato nei passi successivi; il fatto che possa essere scandito in modo più immediato (in quanto presente in modo persistente nella memoria principale piuttosto che su disco) permette di analizzare le informazioni più significative tramite una migliore rappresentazione, velocizzando l'esecuzione e facendo in modo che l'algoritmo non dipenda dalla quantità di memoria disponibile, ma solo dalla mole di dati immagazzinati nel transTree, che può venire regolata tramite il valore m.

Gli itemset che non riescono ad essere presenti in un numero sufficiente di transazioni non vengono ignorati: le transazioni che li contengono vengono inserite in un file che viene analizzato in modo simile ma

tenendo presente che quegli itemset sono contraddistinti da un supporto diverso, come vedremo successivamente.

Per scoprire quali itemset risultino presenti in almeno m transazioni, viene utilizzato un procedimento simile a quello già descritto nel caso del rilevamento degli item frequenti. Infatti, invece di andare a contare quante volte ogni singolo itemset formato da item frequenti è presente nelle transazioni (metodo che impiega un grande consumo di memoria) vengono presi in considerazione solo gli itemset che potrebbero essere contenuti in almeno m transazioni ed ignorati i rimanenti. Anche qui viene utilizzata una struttura vettore transVector con un numero configurabile di celle, ma in questa circostanza il tipo degli elementi del vettore è l'intero. Non è necessario utilizzare una struttura composta formata da più campi in quanto non viene memorizzato l'insieme degli itemset con item frequenti accomunati dallo stesso hash, ma solamente il contatore delle occorrenze. Questo perché gli itemset vengono calcolati di volta in volta durante la scansione delle transazioni; non assume particolare importanza se ciascun contatore è associato a più valori, in quanto si vuole ottenere solo una stima, un numero verosimile di transazioni in cui gli itemset sono presenti. Lo scopo è solamente quello di diminuire ulteriormente il numero di itemset da analizzare, escludendo quelli effettivamente infrequenti, mentre il calcolo di quelli possibilmente frequenti, molto più laborioso, viene effettuato durante i passi successivi.

Anche qui è presente la fase di applicazione della funzione hash, ma solamente per andare ad incrementare l'i-esimo contatore (dove i è il valore risultato dell'hash); viene eseguita all'interno della fase riguardante la costruzione delle strutture di mining, ed in particolare dopo aver costruito il vettore localIndexDepSet. Infatti, durante quella scansione, per ogni transazione rilevata sono già stati presi in considerazione gli item frequenti presenti in quella stessa transazione. Ciascuno di questi è caratterizzato dalla stringa del proprio codice identificativo; tali stringhe vengono concatenate opportunamente tra di loro e separate da spazi vuoti, per formare una stringa unica che viene utilizzata come chiave che identifica quell'itemset formato dagli item frequenti di quella transazione (nel caso della rilevazione degli item frequenti la chiave era rappresentata dal codice identificativo del singolo item). La stringa finale viene sottoposta alla funzione hash che produce il valore i, e alla posizione i del vettore transVector viene incrementato il relativo contatore.

La fase di verifica se l'itemset è presente in almeno m transazioni (per vedere se essere o meno inserito nel transTree) viene effettuata invece all'interno della fase di creazione del transTree, dopo aver ricavato dalla transazione in analisi l'itemset $F = \{f_1,f_2,\ldots,f_m\}$ formato esclusivamente dagli item frequenti contenuti nella stessa transazione. Viene anche qui formata la stringa unica data dalla concatenazione delle varie stringhe degli item per identificare quell'itemset. A questa viene applicata la funzione hash, viene controllato se il contatore alla posizione del valore generato dall'hash è inferiore al numero m; se il controllo è negativo l'itemset può essere inserito nel transTree seguendo le regole prima descritte, altrimenti viene inserito nel file di supporto (outTrans).

La scelta di costruire il transTree senza adottare questa miglioria, ovvero inserendo tutti gli item indipendentemente dal loro numero di occorrenze, equivale ad utilizzare la metodologia appena descritta impostando però a zero il valore di m.

4.2.6 Creazione dell'albero dei pattern

L'algoritmo procede rilevando gli itemset frequenti in maniera stratificata, costruendo la struttura ad albero per la creazione dei pattern (pattTree) livello per livello; durante tale procedura, come vedremo, vengono utilizzate delle tecniche per ridurre il numero di nodi nell'albero in modo da diminuire il consumo di memoria ed il tempo di esecuzione.

Inizialmente viene creato il nodo radice e successivamente il primo livello di nodi. Al nodo radice vengono attaccati solamente gli item frequenti i cui prefissi di dipendenza sono vuoti, in quanto sono quelli che presentano un supporto maggiore; per metter in pratica ciò viene scansionato il vettore ordinato globalDepPrefixSize. Se tutti i prefissi di dipendenza risultano vuoti, nel nodo radice viene indicato che può essere applicato successivamente il metodo Apriori (viene attivato il suo flag relativo ad Apriori); se invece ne esiste almeno uno non vuoto, è da ritenersi concettualmente incompleto (viene attivato il corrispondente flag).

Da questo punto in poi, l'algoritmo continua con la costruzione dell'albero strato per strato (quello attuale viene identificato con k), partendo da un livello k pari a 1. Il primo passo è la generazione dei nodi candidati, lavorando esclusivamente con patTree e controllando tutti i nodi a profondità k.

Per prima cosa viene controllato se il genitore del nodo N è di tipo incompleto: se affermativo, non viene attaccato nessun nodo candidato aggiuntivo a quel nodo. Successivamente viene verificato se è il caso di applicare Apriori, testando se il genitore del nodo N sotto analisi è caratterizzato dal flag Apriori. Se l'esito ha successo i nodi candidati figli di N vengono generati secondo la modalità di Apriori, ovvero impostando a zero i loro contatori e attivando il flag Apriori del nodo N. A questo punto, i nodi a pari livello di N (i suoi fratelli) vengono sfruttati in quanto contengono i nodi per generare i figli candidati di N. Se il padre di N possiede m figli, l'i-esimo figlio N sfrutta come nodi candidati i suoi m-i fratelli alla sua destra.

Questo metodo rappresenta un'ulteriore miglioria che porta alla diminuzione del numero di candidati figli generati per il nodo N. Infatti, se non fosse adottato questo procedimento, il numero di nodi candidati figli aumenta quando gli item frequenti sono caratterizzati da insiemi di dipendenza con pochissimi elementi, ovvero quando gli item frequenti possiedono pochissime dipendenze.

Al contrario, quando gli item frequenti hanno insiemi di dipendenza non vuoti, si può passare alla generazione dei candidati applicando le ottimizzazioni per la riduzione della dimensione dell'albero. Ciò porta alla riduzione della dei nodi e dei rami che contengono informazioni già presenti nella struttura, quindi inutili, e al miglioramento, di conseguenza, dell'efficienza dell'algoritmo. Comunque, nel caso si presentino le condizioni per le quali è inevitabile l'applicazione di Apriori, l'algoritmo si trasforma all'occorrenza in tale tipologia.

Per mettere in pratica le ottimizzazioni, si sfrutta contemporaneamente sia la struttura pattTree che quella transTree, partendo ad ogni livello e in entrambi i casi dalla radice dei rispettivi alberi.

Innanzitutto viene scandito in ampiezza il transTree precedentemente creato, prendendo in considerazione gli itemset presenti nei vari cammini. Per maggior precisione, bisogna dire che questi item sono rappresentati dalle loro posizioni in classifica, e quindi a ciascun itemset $\{x_1,x_2,\ldots,x_m\}$ corrisponde l'insieme delle posizioni $\{p_1,p_2,\ldots,p_m\}$ (m = dimensione dell'insieme di dipendenza).

Gli elementi di ciascun itemset vengono così confrontati con quelli presenti all'interno del patternTree per verificarne la presenza o l'eventuale inserimento (viene utilizzato un vettore positions che memorizza le posizioni in "classifica" dei vari itemset ed il vettore path per identificare se gli stessi item sono stati analizzati durante il cammino attraversato). I nodi del livello successivo vengono sostanzialmente creati elaborando ciascun elemento di ciascun itemset e attraversando ricorsivamente il pattTree, partendo sempre dal nodo radice. Se si indica con k il livello al quale si è giunti nella la creazione dei nodi definitivi, e con l quello del livello raggiunto durante la scansione, i passi della procedura per la creazione dei nodi candidati a livello k+1, applicando le migliorie, sono i seguenti:

- se è in analisi un nodo a livello inferiore rispetto a quello relativo alla creazione dei candidati (l<k), vengono effettuati subito due controlli preliminari per verificare queste condizioni:
    - la posizione p_m è minore rispetto a quella del primo figlio del nodo in analisi, oppure la p_1 è maggiore di quella dell'ultimo figlio del nodo in analisi
    - la dimensione dell'itemset m è inferiore rispetto al numero di nodi rimasti nel cammino fino al livello k (m < s dove s = k + 1 – l)

  In entrambi i casi, se il controllo produce esito positivo, la procedura termina e l'attraversamento si conclude in quanto l'itemset non può essere utilizzato perché i suoi elementi non potranno mai essere inseriti: nel primo caso a causa del valore delle loro posizione, mentre nel secondo caso perché l'itemset fornisce un numero non sufficiente rispetto a quello richiesto nel cammino dal nodo in analisi in poi.

  Altrimenti, viene effettuata una ricerca (sfruttando quella già adottata per la costruzione del transTree, oppure un controllo manuale se la dimensione del cammino è inferiore rispetto ad una certa lunghezza) partendo dal nodo corrente, con lo scopo di scoprire se esiste un vertice x_1. In caso affermativo la sua esistenza nel cammino viene registrata settando il corrispondente indice nel vettore path E si procede con la ricerca della rimanente parte dell'itemset iniziale ($\{x_2,\ldots,x_m\}$) partendo da questo nodo; altrimenti l'attraversamento si conclude.

- se il nodo durante l'attraversamento si trova al medesimo livello di quello in esame (d = k), allora il cammino che porta al nodo corrente è rappresentato dall'itemset $\{y_1,y_2,\ldots,y_k\}$ il quale contiene il prefisso di dipendenza di x_1.
  Di conseguenza, si adotta lo stesso procedimento del caso passo precedente (evitando però i controlli iniziali), effettuando una ricerca partendo dal nodo corrente fino ad un nodo candidato (per scoprire se esiste un vertice con x_1). Se l'esito è positivo, significa che il nodo candidato è già stato precedentemente ritrovato e quindi il suo contatore per supporto viene incrementato con il valore presente nel corrispondente nodo del transTree in modo da tener conto della somma dei supporti dei vari cammini giunti fino a lui; in caso contrario, questo rappresenta un nuovo itemset da inserire, in quanto contenente nuove informazioni (molto probabilmente che deve essere creato un nuovo nodo candidato).
  Prima però è necessario verificare che il nodo non sia caratterizzato dal flag Aprori attivato, e che ogni suo elemento del prefisso di dipendenza sia presente nel cammino. Tutto ciò viene svolto sfruttando le strutture globalDepPrefix, globalDepPrefixSize, positions e path. Se questo genera esito negativo, il nodo viene considerato un nodo incompleto e quindi etichettato come tale, altrimenti viene costruito nel patTree il nuovo nodo candidato $N(y_1,y_2,\ldots,y_k)$, ed il contatore per il suo supporto viene impostato al valore presente nel nodo del transTree;

- a questo punto viene elaborata ricorsivamente la parte rimanente dell'itemset iniziale ($\{x_2,\ldots, x_m\}$), ricominciando quindi dal primo passo di questa procedura

Utilizzo del file delle transazioni poco frequenti

Se presente, viene scandito il file outTrans precedentemente creato, andando a prendere in considerazione le transazioni presenti in tale file. A differenza del caso precedente, questa volta gli elementi da analizzare sono delle semplici transazioni (non gli itemset presenti nel transTree), che possono essere trattate come degli itemset non frequenti, e che quindi sono stati scartati dall'essere inseriti nel transTree. Di conseguenza, alcuni item che li compongono possono non far parte degli itemset frequenti, quindi non hanno un prefisso di dipendenza; perciò non sono contraddistinti da una loro posizione in "classifica" e quindi essere identificati in base a tal parametro. La soluzione è quella di rappresentarli trasformando la loro stringa in intero ed utilizzare un vettore positions (come nel caso precedente) che li memorizza in base a questi valori. In questo caso, non è necessario il vettore path per vedere quali item sono stati analizzati durante il cammino attraversato, in quanto non si attraversa nessuna struttura (come fatto invece per il transTree).

Anche qui vengono creati i nodi per lo stesso livello k del caso precedente e attraversando ricorsivamente il pattTree (partendo sempre dal nodo radice e attuando la stessa procedura). Qui però, quando avviene la verifica se il nodo candidato è già stato precedentemente incontrato oppure deve essere creato un nuovo nodo, il suo valore del contatore per il supporto viene aggiornato non più con il valore presente nel corrispondente nodo del transTree (in quanto non presente) ma di una quantità unitaria.

Pruning dei nodi candidati

Dopo creato i nodi candidati a livello k+1 del patTree, viene effettuato il pruning, ovvero vengono rimossi i quei nodi (sempre a livello k+1) caratterizzati da valori dei loro corrispondenti contatori inferiore alla sogli a minima del supporto impostata. Durante la scansione del patTree, i nodi per i quali è stato già verificato che non devono subire l'eliminazione vengono etichettati come nodi chiusi, in modo da identificarli da distinguerli precisamente nei passi successivi del mining. Se tutti i candidati risulta che debbano essere rimossi, allora l'algoritmo termina; altrimenti viene incremento il livello in esame (k = k + 1) e l'algoritmo procede con l'analisi di tale livello seguendo i medesimi passi precedenti della fase principale riguardante la creazione dell'albero dei pattern.

4.2.7 Derivazione dei pattern

Grazie alla prima proprietà che caratterizza i prefissi di copertura (affrontata all'inizio del paragrafo 4.2.2), se un itemset frequente contiene il suo prefisso di copertura, allora il nodo corrispondente si trova obbligatoriamente nell'albero finale pattTree. Infatti, una volta conclusa l'esecuzione dell'algoritmo, ciascun nodo nell'albero (ad eccezione di quello radice) costituisce un itemset frequente (che definiamo generale) contenente il suo prefissi di copertura. A sua volta, questo itemset frequente rappresenta un corrispondente insieme di itemset frequenti che presentano lo stesso supporto dell'itemeset generale, grazie alla seconda proprietà dei prefissi di dipendenza (descritta sempre nella parte iniziale del paragrafo 4.2.2). Gli itemset che possono quindi essere ricavati da questo singolo itemset generale sono identificati dall'insieme:

$$\{X \setminus Y \mid Y \subseteq pref.di\ dipend.(X)\}$$

Ciascun itemset può essere ricavato solamente da un determinato nodo, il quale è composto dall'unione di quell'itemset con il prefisso di dipendenza dello stesso itemset. Per ricavare quindi gli itemset frequenti (definiti pattern), in questa fase viene attraversato (tramite una ricerca in ampiezza) il patTree per creare l'insieme dei pattern frequenti di tipo closed realmente presenti.

Prima però è necessario ricordare che un itemset $X = \{x_1, x_2, \ldots, x_m\}$ viene definito closed quando non è presente nessun proprio superinsieme che presenta il suo stesso supporto. Durante la scansione, viene analizzato ciascun cammino e, per ognuno di essi, ciascun item che lo compone (rappresentato dalla sua posizione in classifica). Anche qui viene utilizzato il vettore path per conoscere quali item sono stati effettivamente già analizzati durante l'attraversamento del cammino ed il vettore positions per sapere quali sono le posizioni in "classifica" di questi stessi item fin qui attraversati.

Se durante il percorso un nodo presenta lo stesso supporto del nodo genitore significa che l'itemset che caratterizza quel cammino potrebbe non essere un pattern di tipo closed. Perciò, viene mantenuta una variabile per identificare se la condizione di chiusura fino a quel momento dell'attraversamento è verificata o meno per quell'itemset. Ovviamente, se sotto analisi è il nodo radice, la chiusura è da ritenersi per il momento non soddisfatta.

Per costruire l'insieme dei pattern frequenti di tipo closed viene utilizzata una struttura a lista di insiemi di pattern, ordinata in senso crescente in base al valore del supporto. Ogni elemento di questa lista consiste infatti in un insieme (anch'esso di tipo lista) di pattern accomunati da un determinato supporto. Tale struttura creata secondo questo principio permette la facile e veloce ricerca (tramite un'apposita funzione) della posizione del nuovo pattern da inserire (se non già presente) sfruttando il valore del supporto.

Innanzitutto, per ogni itemset in analisi viene rilevata la posizione possibile per l'eventuale inserimento nella lista generale dei pattern, ed in particolare nella lista interna relativa allo stesso supporto che caratterizza l'itemset nel patTree.

Successivamente, è necessario verificare che l'itemset non sia di tipo closed, e quindi è necessario controllare che non esista un altro pattern (già presente all'interno della lista interna relativa allo stesso supporto) che contenga tutti gli item dell'itemset (oltre ad altri eventuali item), altrimenti deve rimpiazzarlo.

Come detto prima, l'itemset è caratterizzato dal vettore path che permetter di sapere quali item sono stati effettivamente già analizzati fino a quella posizione nell'attraversamento. Perciò, viene fatta una scansione di tutti i pattern presenti con quello stesso supporto, e per ciascuno di essi viene effettuato un confronto tra gli item contenuti in esso e quelli esterni presenti nell'itemset (sfuttando il vettore path):

- se il numero di item dell'itemset è minore o uguale rispetto al numero di item presenti nel pattern esterno, e se quest'ultimo contiene tutti gli item dell'itemset, significa che il superset è già presente in lista, perciò non è di tipo closed e non deve essere inserito nella lista con quel supporto; la scansione dei pattern esterni si conclude
- se il numero di item dell'itemset è maggiore rispetto al numero di item presenti nel pattern esterno, ma almeno un item di quest'ultimo non è contenuto anche nel pattern esterno, allora l'itemset non è il suo superset e quindi la scansione continua con il pattern successivo
- se il numero di item dell'itemset è maggiore rispetto al numero di item presenti nel pattern esterno, e l'itemset contiene tutti gli item del pattern esterno, allora l'itemset rappresenta il

superset del pattern esterno. Quest'ultimo quindi viene sostituito dall'itemset che diventa di tipo closed, e la scansione continua con il pattern successivo

4.2.8 Gestione dei risultati

In questa fase viene scandita la lista generale dell'insieme globale dei pattern, visitando le varie liste interne supporto per supporto. Viene rappresentato ciascun pattern di ciascuna lista, indicando le stringhe che identificano i vari item che lo compongono. E' importante ricordare che ogni transazione è formata da un numero fissato di item presenti nelle rispettive posizioni della stessa transazione in base al tipo di item, (l'attributo specifico che esso rappresenta); quindi vengono riprodotte a video le varie stringhe che rappresentano gli item nella posizione corretta. Per risalire all'identificativo della posizione e alla stringa, siccome nella lista sono presenti solamente le posizioni, viene utilizzata la struttura vocabulary, i cui elementi sono appunto indicizzati in base alla posizione, e contengono le informazioni utili allo scopo.

Per effettuare un'analisi dei risultati ottenuti è opportuno che i pattern contengano lo stesso numero di elementi . I pattern presenti nella lista però rappresentano gli itemset frequenti che si verificano durante le transazioni; il numero di item frequenti potrebbe essere inferiore al numero di campi di una transazione perché alcuni campi possono non rientrare nei pattern frequenti. Questi campi non presenti nel pattern vengono rappresentati nella notifica a video dal simbolo asterisco (come accennato nella sezione), indicante che il valore di quell'attributo per quel pattern è variabile, ovvero può assumere diversi valori in quanto non frequente. Per sapere dove e quanti asterischi inserire, anche qui vengono analizzati gli identificativi di posizione che non sono presenti tra gli item nel pattern frequente da riprodurre.

Oltre a stampare a video l'elenco degli itemset frequenti (rappresentati dai rispettivi pattern) ottenuti con un determinato supporto, tale risultato viene memorizzato in un file. Se il file di log iniziale è definito "nomeSettimana", allora viene creata una cartella "nomeSettimana_Itemset_Results" nella directory dove è presente il log, e i risultati del mining degli itemset frequenti vengono inseriti nel file "nomeSettimana_Itemset_Mining".

5 Mining mediante clustering

5.1 Necessità di una nuova tecnica e formulazione iniziale del problema

Gli algoritmi di clustering operano suddividendo un insieme di elementi in gruppi definiti cluster; gli elementi all'interno di ciascun cluster sono molto simili tra loro e molto dissimili dagli elementi presenti negli altri cluster. Quelli che per loro caratteristiche, non rientrano in nessuno dei cluster presenti rappresentano un gruppo a se stante e vengono definiti outlier.

Gli elementi che possono essere classificati in cluster possono consistere nelle righe presenti all'interno di un log, e quindi possono un possono essere benissimo sottoposte a questa tecnica di mining.

Ciascun log, infatti è costituito da righe che possono essere trattate come dei punti; a loro volta questi punti sono costituiti da m elementi, le parole, le quali rappresentano i valori che assumono m determinati attributi. Il log quindi può essere visto come un insieme di punti (le righe) ognuno dei quali è costituito da m valori di attributi (le parole).

Ciascuna linea può rientrare in un determinato pattern, e quindi tutte le linee che vi appartengono presentano una certa similarità tra loro; allo stesso modo presentano molte diversità rispetto alle linee che non appartengono a quel pattern ma ad altri. Tali pattern possono essere considerati quindi dei cluster le cui righe presentano e sono accomunate da determinate caratteristiche che contraddistinguono il cluster. Le caratteristiche consistono nei valori corrispondenti ai vari m attributi, i quali formano la spazio dei dati ad m dimensioni.

La tecnica del clustering applicata ai file di log può identificare molti pattern riguardanti righe che indicano la normale attività di un sistema e che quindi possono venire incluse nel profilo di sistema, senza dover applicare ulteriori azioni quali ad esempio scoprire le regole di associazioni nascoste. Oppure gli outlier possono contenere righe non particolarmente frequenti che notificano un comportamento inusuale di un sistema che necessitano di un controllo più approfondito.

5.2 Problemi dei dati nei file di log

Esistono però diversi problemi che devono essere presi in considerazione quando si progetta un algoritmo di clustering, soprattutto quando i metodi tradizionali sono pensati per essere applicati a dati di tipo numerico e caratterizzati da spazi dei dati con poche dimensioni.

I dati contenuti nei file di log invece presentano caratteristiche diverse; infatti le righe trattate come dei punti sono costituite da m elementi, le parole, le quali rappresentano i valori che assumono m determinati attributi. Questi attributi non sono di tipo numerico,bensì di tipo categorico, in quanto ad ogni posizione della riga corrisponde un preciso tipo di attributo. Ad esempio, in un log, la riga "124 192.168.0.1 80" può essere rappresentata dal punto ('124','192.168.0.1','80') dove 124,192.168.0.1 e 80 sono i valori dei rispettivi attributi categorici Id, indirizzo IP e porta.

Se si tiene presente che il clustering è definito come dato un insieme di punti con n attributi nello spazio dei dati R^n, creare una partizione dei punti per formare dei cluster in modo che i punti all'interno di ogni cluster siano vicini, ovvero simili, ad ogni altro punto sempre di quel cluster.

Per determinare quanto vicini, quindi simili siano due punti a e b tra di loro, viene impiegata una funzione di distanza d(a,b). La misurazione della distanza tra punti caratterizzati da attributi di tipo categorico non è banale, a differenza di quanto avviene per attributi numerici. Infatti, non possono venire utilizzate le misurazioni più comuni, ma è necessario verificare con attenzione quale funzione di distanza adottare (ad esempio il coefficiente di Jaccard) che si presti meglio caso per caso.

Inoltre il numero di attributi in un log, che definisce la dimensione dello spazio su cui risiedono i vari punti e sui cui opera l'algoritmo di clustering, spesso è elevato (ad esempio quasi sicuramente maggiore di 4) in quanto ciascuna è costituita da molti elementi che denotano la presenza di molti corrispondenti attributi.

All'aumentare del numero di attributi, quindi delle dimensioni dello spazio dei punti, può verificarsi che, in alcune dimensioni, alcuni punti risultino talmente distanti dagli altri, da rendere molto impegnativo, se non impossibile, il rilevamento dei cluster. Inoltre, può accadere che alcuni punti non risultino vicini tra loro per l'originale spazio dei dati ad alte dimensioni, sebbene risultino invece realmente vicini in alcuni sottospazi dello spazio originale, ovvero in alcune dimensioni di quest'ultimo.

Inoltre molti algoritmi sono progettati per insieme di dati di tipo generico come i market basket, che non adottano particolari assunzioni riguardo la natura dei dati. Questo aspetto assume un ruolo chiave quando si sceglie di progettare un algoritmo di clustering per essere applicato ai file di log; i dati presenti in questi file (come già detto nel capitolo 3) presentano delle particolari caratteristiche differenti dagli altri tipi di insiemi di dati. Queste peculiarità li contraddistinguono dalle altre tipologie di dati in circolazione e quindi devono essere assolutamente tenute in considerazione.

5.3 Descrizione dettagliata dell'algoritmo

L'algoritmo di clustering presente nella Tesi cerca di risolvere tutti i vari aspetti problematici messi in risalto nella sezione precedente.

Si basa sulle idee presentate nel paragrafo 5.1, ovvero tratta le righe del log come un insieme di punti ognuna delle quali è costituita da m parole, che rappresentano gli m valori dei relativi attributi categorici. Lo spazio generale dei dati su cui opera contiene quindi punti con attributi categorici ed è caratterizzato da m dimensioni.

Dalla definizione finale nel capitolo 3, ogni allarme A deve essere considerato perciò una tupla costituita da una serie di elementi (in questo caso definiti punti) ognuno formato dalla coppia valore attributo-numero posizione:

$A = \{(A_{id},1),(A_{srcIP},2),\ldots\}$

Ogni cluster finale rappresenta l'insieme dei punti che soddisfano le cui caratteristiche soddisfano quel pattern, ed il numero di punti viene definita la densità del cluster; una volta creato viene identificato e descritto di conseguenza da un pattern formato da parole costanti e variabili, e corrisponde alla descrizione generale che raccoglie i punti con quelle caratteristiche (i valori degli attributi costanti) e che occorrono con

una sufficiente frequenza (analogamente ai pattern relativo agli itemeset frequenti nell'algoritmo del capitolo precedente). I punti che non appartengono a nessun cluster rilevato vengono chiamati outlier.

L'obiettivo su cui si basa l'algoritmo è quello di partizionare le linee del file di log in cluster in modo che ogni cluster possieda un supporto uguale o superiore rispetto alla soglia minima di densità definita: se la soglia è D, riporta i cluster che contengono almeno D punti.

L'algoritmo è fondato su un approccio di clustering basato sulla densità. Una regione S è un sottospazio dello spazio generale dei dati, che contiene dei punti per i quali il gli attributi $a_1,a_2,\ldots, a_k$ (sottoinsieme degli attributi totali, ovvero $1\leq k\leq m$) presentano gli stessi valori $v_1,v_2,\ldots, v_k$, ovvero se x è un elemento che appartiene ad S, allora l'attributo a_1 assume il valore v_1, a_2 il valore v_2, ecc.. Le coppie attributo-valore che caratterizzano la regione formano l'insieme $\{(a_1,v_1), (a_2,v_2),\ldots,(a_k,v_k)\}$ definito l'insieme degli attributi fissi della regione S. Se k = n, la regione viene definita la n-regione.

Lo scopo dell'algoritmo è quello di trovare tutte le regioni che contengono un numero pari o maggiore al valore della soglia minima impostata. Tali regioni vengono definite dense: se viene impostato un valore D come valore della soglia minima di densità, lo scopo è quello di cercare le regioni che contengono almeno D punti. Le regioni dense rappresentano i cluster presenti nei sottospazi dello spazio dei dati generale.

In definitiva, è basato sulla ricerca di regioni dense associate alla combinazione di valori degli attributi frequenti; perciò, per ricercare i cluster, il concetto base è che ogni valore costante assunto dagli attributi in un cluster deve essere frequente, ovvero occorrere D o più volte nel file di log. Di conseguenza, diventa inutile adottare una nozione di distanza; questa scelta è supportata anche dal fatto che tale misura diventa un'informazione pressoché inutile quando si ha a che fare con spazi dei dati caratterizzati da molte dimensioni come già indicato.

L'algoritmo è stato progettato per fare il minor numero possibile di scansione del log, e consiste principalmente in queste fasi:

- Generazione delle 1-regioni dense
- Generazione dei cluster candidati
- Selezione dei cluster effettivi
- Perfezionamento dei cluster

Ora si procede con la descrizione accurata di ogni singola fase.

5.3.1 Generazione delle 1-regioni dense

Innanzitutto viene effettuata una scansione del log con lo scopo di identificare tutte le regioni dense di tipo 1-regione. Se il valore della soglia minima è impostato a D, ciascuna regione da creare deve quindi contenere almeno D elementi accomunati dal fatto che un particolare attributo a assume il valore v. In questo caso l'insieme $\{(a_1,v_1), (a_2,v_2),\ldots,(a_k,v_k)\}$ degli attributi fissi di ciascuna regione si riduce all'insieme $\{(a,v)\}$, costituto da una sola coppia che identifica l'unico attributo fisso per quella regione, in quanto k=1.

Questa fase consiste quindi nella ricerca degli elementi il cui attributo presenta lo stesso valore (parole identificate dalla stessa stringa e presenti nella stessa posizione) e che compaiono almeno D volte all'interno dei punti (righe) del log. Di conseguenza, il procedimento equivale esattamente alla ricerca delle parole frequenti all'interno del log: le regione dense di tipo 1-regione sono paragonabili agli itemset

frequenti di dimensione unitaria; quindi questa fase è analoga a quella già presente nel mining dei pattern frequenti realizzato nel capitolo 4. Per tal motivo l'algoritmo effettua le stesse procedure messe in pratica in quella circostanza fino al passo relativo alla costruzione delle strutture di mining (non compreso).

Al termine, si ottiene un word vector che viene sfruttato dai passi successivi dell'algoritmo e che contiene tutti gli item frequenti; ciascuno di questi rappresenta esattamente una regione densa di tipo 1-regione. Anche qui, il wordsVector, per effettuare veloci ricerche future dei propri elementi, è caratterizzato dal fatto che ogni cella è contraddistinta da elementi (in questo caso identificanti regioni dense 1-regione) accomunati dallo stesso valore se li viene applicato la funzione hash al codice della relativa parola (valore dell'unico attributo fisso della relativa regione). Inoltre, viene utilizzata anche la struttura vocabulary creata in questa fase e contenente gli elementi presenti nelle varie liste all'interno di wordsVector ma mantenuti in maniera separata tra loro.

A questo punto, d'ora in avanti, affinché la descrizione dell'algoritmo di clustering sia giustamente focalizzata nel concepire il log come un caso che si presta a tale procedimento, ogni elemento viene trattato e definito in termini adatti al clustering:

- una parola presente nella posizione i-esima della riga viene definita il valore dell'attributo i-esimo, che per semplicità viene definito a sua volta value
- una linea (formato da m parole) viene identificata come un punto (caratterizzato dagli m valori che assumono gli m attributi)
- l'insieme delle parole frequenti presenti in una riga viene definito i value fissi che caratterizzano quel punto

Di norma, gli algoritmi tradizionali di clustering prevedono che ogni punto risulti appartenente esclusivamente ad un unico cluster. Tuttavia esistono diversi casi in cui un punto può risultare benissimo presente in diversi cluster, e questo con lo scopo di ottenere dei risultati finali maggiormente comprensibili. Nell'algoritmo di clustering della Tesi è possibile scegliere tra le due alternative; la differenza sostanziale nel procedimento si incontra principalmente nella fase riguardante la selezione dei cluster effettivi da quelli candidati generati nella fase precedente (anche se già nella fase di generazione dei candidati devono essere effettuate delle azioni per poter poi applicare i procedimenti della fase successiva).

L'idea consiste nel fatto che con la prima opzione (definita "unico cluster), siccome ogni punto non può appartenere a più cluster, tra i cluster finali (nella fase di selezione degli stessi) non possono essere presenti cluster generici che contengono cluster più specifici, altrimenti il punto appartenente a quello più specifico viene inserito anche in quello più generico. Con la seconda opzione (definita "più cluster), invece, può venir generata tutta le gerarchia di cluster senza alcun problema.

5.3.2 Generazione dei cluster candidati

A questo punto viene effettuata la seconda scansione del log, con lo scopo di generare tutti i cluster candidati. Di ogni punto rilevato viene ricercato e definito l'insieme degli attributi fissi che lo caratterizzano; se questo insieme non è vuoto (esiste almeno un attributo fisso) il punto è da considerarsi appartenente ad una o più regioni di tipo 1-regione, e di conseguenza deve essere creato (se non già esistente) un candidato caratterizzato da questi attributi fissi.

Il punto caratterizzato dall'insieme di attributi fissi $\{(a_1,v_1), (a_2,v_2),\ldots,(a_k,v_k)\}$ appartiene quindi a k regioni dense di tipo 1-regione, e quindi deve essere creato (se non esiste) un cluster candidato (o regione) caratterizzato dall'insieme di attributi fissi $\{(a_1,v_1), (a_2,v_2),\ldots,(a_k,v_k)\}$.

Come detto precedentemente, già in questa fase devono venire attuate delle azioni ulteriori se viene scelta la seconda possibilità, affinché vengano opportunamente applicati i procedimenti della fase successiva. Infatti, se si sceglie la prima opzione, in questa fase viene memorizzata per lo più una tipologia di cluster candidati (che per costruzione, come vedremo, sono quelli più specifici), altrimenti vengono considerati anche tutti i loro corrispondenti cluster canditati più generici. Da ricordare che ciascun cluster candidato è caratterizzati dal suo insieme di value fissi.

Per mettere in pratica tutto ciò, innanzitutto deve essere creato in entrambi i casi il vettore clustersVector; se viene scelta la prima soluzione, la scopo di questa struttura consiste solamente nel memorizzare tutti i vari value fissi in modo che la loro ricerca futura risulti veloce. Se invece si opta per la seconda soluzione, allora viene sfruttato anche per applicare lo stesso principio su cui si basa il transVector costruito nel mining di itemset frequenti nella sezione 4.2.5. Se in quel caso lo scopo era quello di controllare, in base al contatore, quali itemset potevano essere presenti in almeno m transazioni ed ignorare i rimanenti, in questa circostanza l'obiettivo è quello di verificare quali insiemi di value fissi sono presenti in almeno m punti. Questo sempre con lo scopo di diminuire ulteriormente il numero di value fissi da analizzare, escludendo effettivamente quelli che non formano sicuramente regioni dense, mentre il calcolo di quelli effettivamente densi, più complesso, viene effettuato nei passi successivi.

Il clustersVector consiste in una struttura vettore con un numero m configurabile di celle, ciascuna contraddistinta da due campi:

- una lista di value fissi accomunati dallo stesso hash
- il contatore della densità associato a quella lista (inizializzato al valore zero)

Se viene scelta la seconda opzione, vengono sfruttati entrambi i campi e si procede con un passo aggiuntivo rispetto al normale funzionamento dell'algoritmo. Viene effettuata una scansione del log e viene analizzato ogni punto; il valore di ciascun attributo è contraddistinto da una stringa che rappresenta il proprio codice identificativo. Ad ogni stringa viene applicata la funzione hash, viene generato il valore i, viene effettuata una ricerca alla posizione i del wordsVector per vedere se è presente quel value, ed in caso affermativo significa che il value è un value fisso.

Le stringhe dei vari value fissi rilevati vengono poi concatenate opportunamente tra di loro e separate da spazi vuoti, per formare una stringa unica che viene utilizzata come chiave che identifica quell'insieme di value fissi ritrovati in quel punto. La stringa generale viene sottoposta anch'essa alla funzione hash, produce il valore i, e, alla posizione i del vettore clustersVector, viene incrementato il contatore della densità.

Quando si incontrano diversi insiemi di value fissi, ciascuno può essere costituto da un numero diverso di value. Se alcuni value fissi si presentano con maggiore frequenza rispetto ad altri, gli insiemi di value fissi che contengono esclusivamente questi value fissi risultano meno frequenti di quelli che contengono anche altri value fissi. Quando in un insieme vengono incontrati anche altri value fissi oltre a quelli più frequenti, i value fissi più frequenti vengono considerati (per costruzione delle stringhe) come appartenenti solamente agli insiemi che contengono anche altri value fissi; perciò, il contatore viene incrementato solamente per questi insiemi e non per quelli che contengono esclusivamente quelli più frequenti.

Gli insiemi di value fissi che contengono esclusivamente i value fissi più frequenti consistono sostanzialmente in insiemi più generici, mentre quelli che contengono anche altri value fissi sono quelli legati a loro ma più specifici. In questo modo, al termine dell'aggiornamento dei contatori nelle varie posizioni, è probabile che le celle del clustersVector, contenenti insiemi di value fissi più generici, siano contraddistinte da valori dei contatori della densità inferiori rispetto a quelle presenti nelle celle contenenti i corrispettivi insiemi di value fissi più specifici; ciò è corretto perché la seconda opzione consiste proprio nel cercare di mantenere solamente quelli più specifici in modo che un punto poi appartenga ad un unico cluster.

Da questo punto in poi si procede con le azioni relative alla prima opzione.

Se si è giunti fin qui avendo scelto la prima opzione, allora viene sfruttato solamente il primo campo della struttura clustersVector, saltando il passo precedentemente descritto relativo alla seconda opzione.

In entrambi le scelte, viene effettuata una scansione del log, e per ogni punto viene applicata la funzione hash alla stringa generale data dalla concatenazione delle stringhe dei vari value fissi ritrovati in quel punto, ottenendo il valore i.

Se prima si era optato per la seconda opzione, ora nel contatore alla posizione i del vettore clustersVector è contenuto il valore della densità totale dei vari insiemi di values fissi per quel valore di hash. Perciò viene effettuato il controllo se tale valore è uguale o maggiore alla soglia minima impostata; in caso negativo, l'insieme di value fissi in analisi viene ignorato.

Per le motivazioni date precedentemente, è probabile che le celle contenenti insiemi di value fissi più generici siano contraddistinte da valori bassi (inferiori rispetto a quelli presenti nelle celle contenenti i corrispettivi insiemi di value fissi più specifici) e che quindi gli insiemi di value fissi considerati appartenenti a queste celle vengano correttamente ignorati in quanto consistenti in candidati non validi. Se invece la condizione è soddisfatta, oppure la seconda opzione non è stata precedentemente scelta, l'insieme dei value fissi in analisi è da ritenersi un candidato valido a formare una regione densa (un cluster candidato), e quindi viene inserito nella lista dell'insieme di value fissi alla posizione i del clustersVector. Se l'insieme è già presente, il suo relativo contatore singolo viene incrementato di un'unità, altrimenti viene settato ad 1.

Al termine della creazione del clustersVector, i cluster candidati all'interno delle varie liste in clustersVector che presentano un valore del contatore relativo alla densità globale della lista a cui appartengono non inferiore alla soglia minima devono essere inseriti in una struttura apposita (in quanto consistono in potenziali regioni dense), mentre gli altri vengono ignorati. La struttura (totalSlices) viene pensata tenendo conto della possibilità di optare eventualmente per la seconda scelta (viene usata comunque anche nel caso dell'altra opzione).

Infatti, nella fase successiva, se viene scelta la seconda opzione, l'azione principale consiste nel comparare ogni candidato potenziale con ogni altro candidato in modo da trovare i suoi sottocluster più specifici. Siccome tale comparazione risulta un procedimento caratterizzato da una complessità quadratica e l'insieme totale di candidati potenziali presenti in wordsVector può essere elevato, il completamento di tale operazione può richiedere molto tempo.

Per ridurre la complessità temporale, la soluzione è quella di suddividere l'insieme dei candidati potenziali in porzioni, ed il raggruppamento avviene in base alla dimensione dei vari insiemi. Per mettere in pratica ciò, ogni cluster candidato valido presente nelle varie liste di clustersVector, viene inserito nella struttura

ordinata di tipo vettore già accennata prima (totalSlices), rappresentante le varie porzioni in base alle dimensioni dei candidati in esse contenute. Infatti, le singole celle contengono la lista di tutti gli insiemi di value fissi accomunati dal fatto che ogni insieme presenta lo stesso numero di elementi. In particolare, alla posizione i di totalSlices sono contenuti tutti gli insiemi di value fissi ciascuno di dimensione i+1, e quindi la struttura risulta ordinata in senso crescente rispetto alla dimensione degli insiemi dei value fissi candidati. Ogni insieme di value fissi non viene creato nuovamente in questa struttura ma viene definito tramite un riferimento all'elemento corrispondente presente nella struttura clustersVector; il valore del supporto viene comunque mantenuto anche in questa struttura, attingendolo da quello presente sempre in clustersVector.

Per rendere ancora più veloce la ricerca dei sottocluster più specifici, ogni porzione inoltre è costituita da un vettore descrittivo bitVector, che indica quali value fissi sono presenti all'interno della totalità degli insiemi di value fissi in quella stessa porzione. E' caratterizzato da una dimensione pari al numero di value che formano una regione di tipo 1-regione, ovvero quelli presenti in wordsVector e vocabulary precedentemente costruiti. Ad ogni posizione corrisponde l'indice che contraddistingue quel value in vocabulary ed il contenuto indica se quel value è presente o meno in almeno un insieme di value fissi di quelli presenti in quella porzione. In questo modo, ogni volta che viene aggiunto alla cella i un cluster candidato all'insieme di cluster di dimensione i+1, viene aggiornata la presenza dei vari value fissi di quel candidato nell'insieme dei candidati fin'ora aggiunti in quella cella. Tali informazioni vengono sfruttate nella fase della selezione dei cluster effettivi per rendere più veloce la ricerca dei cluster.

La struttura totalSlices, in definitiva, viene sfruttata nella fase successiva principalmente per la generazione veloce dei cluster più specifici da quelli più generici, entrambi comunque creati in quella fase. Inoltre, siccome è una struttura ordinata, viene utilizzata per creare un'altra struttura, totalCLuster, che consiste in una semplice lista, creata sulla base di totalSlices, di tutti i candidati potenziali, ovvero gli insiemi di value fissi, ordinati sempre secondo il medesimo principio. La combinazione dell'utilizzo delle due strutture totalCluster e totalSlices, unita alle informazioni contenute in clustersVector, permette di ricercare efficacemente i veri cluster nella fase successiva.

5.3.3 Selezione dei cluster effettivi

In questa fase viene analizzato l'insieme dei cluster candidati, ovvero gli insiemi dei value fissi (sfruttando totalCluster) che rappresentano le regioni candidate generate nella fase precedente, in modo da estrarre i cluster veri e propri.

Se è stata scelta la prima opzione, i cluster effettivi vengono generati andando semplicemente ad analizzare l'insieme dei cluster candidati in totalCuster ed eliminando quelli il cui valore del contatore singolo della propria densità è inferiore rispetto alla soglia stabilita; gli elementi che rimangono dopo tale eliminazione rappresentano i cluster finali, mentre vengono definitivamente eliminati quei candidati rimasti dopo la fase precedente che sono generici (rispetto a quelli specifici effettivamente presenti), perché dotati, per le motivazioni esposte prima, di densità minore.

Se invece è stata scelta la seconda opzione, prima del procedimento appena descritto, è necessario generare per ogni cluster totalCuster gli eventuali cluster corrispondenti più generici sfruttando totalSlices.

L'idea è quella di ispezionare i cluster candidati in totalCLuster e comparare ogni candidato con ogni altro candidato in modo da trovare i suoi sottocluster più specifici. In pratica, vengono considerate le relazioni

tra candidati, e per questo viene applicata un'intersezione tra i vari insiemi delle varie porzioni. Questo procedimento richiede la presenza di tutti i possibili candidati, ed ecco perché, se si sceglie la seconda opzione, non viene effettuato il controllo della densità nella fase precedente. Se questo avvenisse, quelli più generici in quel momento verrebbero eliminati (perché con densità bassa) ed in questo passaggio non verrebbero generati i risultati corretti.

Per mettere in pratica l'idea, innanzitutto è necessario che ogni cluster sia caratterizzato da un elemento che rappresenti tutti i suoi cluster più generici (tramite una lista di padri) ed un altro che rappresenti invece quelli più specifici (tramite una lista di figli). Si crea quindi una struttura ad albero che rappresenta le gerarchia dei cluster in ordine crescente dall'altro verso al basso secondo il loro grado di specificità, sfruttando anche i riferimenti agli elementi presenti in clustersVector.

A questo punto viene scandito l'insieme dei cluster candidati nella struttura ordinata totalCluster, e per ciascun cluster C in esame di dimensione i (cluster da considerarsi generico) vengono scandite le porzioni i cui cluster hanno dimensione maggiore ad i, ovvero le liste contenute in totalSlices nelle celle con indice pari o superiore ad i (cluster possibilmente più specifici di C). Se esiste almeno un cluster di dimensione j > i (la lista della cella in esame con indice j-1 non è vuota), allora viene controllato che tutti gli i value fissi di C siano presenti per lo meno all'interno della totalità degli insiemi di value fissi di dimensione j (quelli della cella con indice j-1). La procedura viene effettuata confrontando gli indici nel vocabulary dei value fissi di C con il vettore descrittivo bitVector presente nella cella j-1 che indica la loro presenza nella lista di quella cella. Viene così evitata un'inutile ricerca in tutti i cluster di quella lista, quando invece si è già a conoscenza che almeno un value non è presente e quindi nessun cluster contenuto nella lista della cella j-1 può essere un cluster più specifico rispetto a C.

Se invece viene rilevata la presenza di tutti i value fissi di C, allora possono essere presenti dei cluster più specifici riguardanti il cluster C; per verificarlo si passa alla scansione della lista dei cluster di dimensione j, analizzandoli uno ad uno e testando che tutti gli i value fissi di C siano contenuti nel cluster in esame C2 di dimensione j. Se l'esito è positivo significa che C2 è un cluster più specifico rispetto al cluster C. C2 quindi viene inserito tra i figli di C mentre C viene inserito tra i padri di C2. Inoltre, la conseguenza è che tutti i punti che appartengono al sottocluster C2, appartengono anche al cluster più generico C (un punto può appartenere simultaneamente a più cluster); dunque il contatore della densità di C viene incrementato del valore del contatore della densità di C2, in quanto, essendo C più generico, assorbe anche la densità dei corrispondenti cluster più specifici.

Al termine della scansione di tutti i cluster di totalCLuster con l'aggiornamento delle rispettive densità, si procede come fatto quando viene scelta la prima opzione, ovvero viene effettuata una nuova scansione di questa struttura in modo da eliminare i cluster candidati con densità inferiore alla soglia stabilita e quelli rimanenti sono considerati i cluster finali. A differenza della prima opzione, in questo caso sono presenti anche tutti i cluster più generici, perché al termine della fase di generazione dei candidati sono stati mantenuti tutti i cluster candidati (anche quelli generici); inoltre in questa fase è stata riconosciuto il supporto aggiuntivo dei loro più specifici facendoli rientrare nella condizione che li identifica come cluster finali (seppure il loro valore iniziale di densità nella generazione dei candidati era inferiore alla soglia).

5.3.4 Perfezionamento dei cluster

In questa fase si cerca di perfezionare i vari cluster, in particolare attraverso due migliorie: la generazione degli outlier e il rendere più precisi i pattern descrittivi che identificano i vari cluster appena trovati.

Per quanto riguarda la prima parte, vengono ricercati tutti i punti che non appartengono a nessuno dei cluster finali generati nella fase precedente. Essi rappresentano dei punti isolati che non rientrano in quelli finali: possono consistere in possibili anomalie che necessitano di un controllo più approfondito e perciò vengono inseriti in un file apposito.

Viene effettuata una scansione del log e viene analizzato ogni punto; viene generata la stringa generale che rappresenta l'insieme dei value fissi di quel punto al quale viene applicata la funzione hash che genera il valore i in modo da controllare se quell'insieme di value fissi (al quale appartiene il punto in esame) rappresenta un cluster finale.

Quindi viene effettuata la ricerca all'interno del clustersVector nella lista di cluster finali in posizione i. Se il cluster non è presente significa che il punto in analisi non fa parte di nessun cluster finale ma rappresenta un outlier; per tal motivo la stringa formata dai value di quel punto viene inserita nel file relativo agli outlier. Se invece la ricerca ha prodotto esito positivo, allora il punto appartiene a uno o più cluster finali. Ogni punto di questo genere può essere sfruttato per applicare la seconda miglioria.

La seconda miglioria consiste nel rendere più precisi i pattern che descrivono i cluster, andando a trattare, tra le stringhe testuali che compongono ciascun pattern, quelle relative esclusivamente agli elementi variabili (mantenendo invece invariate quelle che identificano i value fissi). L'idea è quella di ricercare nel log delle possibili parti che accomunano gli elementi rappresentati dal simbolo '*', sia in termini di prefisso che di suffisso, in modo da perfezionare ulteriormente i pattern dei cluster.

Quando, per un certo punto in esame, si giunge al termine della prima miglioria ed il risultato è che quel punto appartiene a uno o più cluster finali, si prendono in considerazione tutti i cluster che lo contengono. Se questi cluster vengono analizzati in questa fase per la prima volta, (è stato trovato per la prima volta un punto e questo è contenuto in essi) allora i value variabili (o, per maggiore precisione, le stringhe che li identificano) di questi cluster assumono il valore dei rispettivi value contenuti nel punto in esame. Quando invece vengono successivamente riscontrati altri punti che appartengono a tali cluster, viene effettuato un confronto tra le stringhe dei value variabili presenti nei cluster fino a quel momento e quelle dei rispettivi value del punto in analisi. Il risultato del confronto tra ogni coppa di queste stringhe consiste in una nuova stringa che sostituisce quella associata al value variabile fino a quel momento. Tale stringa è una versione uguale o ridotta di quest'ultima, in quanto mantiene solamente le parti comuni del prefisso e del suffisso che la stringa del value variabile presentava fino a quel punto rispetto a quella del value corrispondente del punto in esame.

In questo modo, al termine dell'analisi di tutti i punti, i pattern dei vari cluster presentano, dove possibile, dei value variabili più precisi: non sono più definiti semplicemente dal simbolo generale '*', ma indicano anche le parti comuni che si possono riscontrare nei vari value purtroppo non densi.

5.3.5 Gestione dei risultati

Analogamente alla circostanza relativa agli itemset frequenti, anche qui viene riprodotti a video l'elenco dei cluster finali per un certo valore di densità indicando i rispettivi pattern. Tale risultato viene inoltre inserito in un file: se il file di log iniziale è nominato "nomeSettimana", allora viene generata una cartella "nomeSettimana_Cluster_Results" nella directory dove è presente il log, e i risultati del clustering vengono memorizzati nel file "nomeSettimana_Cluster_Mining".

6 Tecniche per il rilevamento dei falsi allarmi

Per le motivazioni descritte nel paragrafo 1.2, i falsi allarmi e quelli irrilevanti prodotti da un IDPS non possono essere evitati nella maggior parte degli ambienti e delle situazioni. Per distinguerli dagli allarmi veramente importanti è necessaria l'applicazione di tecniche per l'analisi dei log.

Molti approcci consistono nell'analizzare i log precedenti, quindi del recente passato, per estrarre delle informazioni le quali vengono utilizzate svariate motivazioni: prendere delle decisioni riguardo l'eliminazione di regole contraddistinte da un alto tasso di fasi allarmi, creare di filtri e regole di correlazione da applicare ai log futuri, ecc.. Questi metodi però sono da considerarsi semi-automatici in quanto richiedono che le informazioni derivate dai log passati vengano analizzate ed interpretate da persone esperte effettuare le azioni prima elencate; è un procedimento dispendioso che richiede tempo, un alto grado di preparazione e che e deve avvenire regolarmente a causa del continuo mutamento dell'ambiente, del comportamento del sistema, delle minacce, delle regole, ecc..

In questa Tesi di Laurea vengono proposti due approcci ("Itemset" e "Cluster") che utilizzano le due tecniche di data mining descritte nei capitoli precedenti, ovvero gli algoritmi per il mining degli itemset frequenti ed il clustering, entrambi dei metodi di tipo unsupervised learning. Tali metodologie vengono applicate ai log passati in modo da estrarre le informazioni importanti, le quali vengono poi utilizzate come training data; infatti vengono processate in modo automatico, tramite l'applicazione diversi procedimenti, per costruire i relativi classificatori di allarmi che vengono impiegati per i log recenti e futuri. I classificatori creati vengono così utilizzati per distinguere in modo automatico i falsi allarmi e quelli di poca importanza da quelli effettivamente interessanti nei nuovi log sulla base delle informazioni ricavate dai log precedenti. I vari allarmi quindi vengono classificati applicando un'etichetta in base al risultato ottenuto dall'applicazione della metodologia. Gli allarmi che vengono classificati come falsi allarmi o allarmi di poca importanza vengono etichettati come allarmi di tipo "Routine" mentre quelli che possono essere dei potenziali veri allarmi da considerare con maggior attenzione di tipo "Interessante".

In questa sezione, nonostante si utilizzino ancora gli algoritmi di data mining, non viene più utilizzato il termine transazione (nel caso del mininq degli itemset frequenti) o punto (nel caso del clustering) quando si vuole indicare una particolare riga presente nel log. Bensì viene utilizzato in entrambi i casi il termine allarme in quanto lo scopo di questa sezione è proprio incentrato su questa tipologia di elementi. Allo stesso modo, non viene più utilizzata la parola item o value quando si considera la parola di una riga, ma si parla del valore che un determinato attributo assume in quell'allarme.

Se L è l'insieme degli allarmi rilevati dall'IDPS e presenti nei log passati analizzati attraverso gli algoritmi di data mining, mentre A è un allarme attuale o futuro rilevato dall'IDPS per essere poi classificato, allora la metodologia che sfrutta il corrispondente algoritmo di data mining può essere rappresentata come una funzione: applicata a ciascun allarme A, produce in output il valore 0 (allarme di tipo "Routine") o il valore 1 (allarme di tipo "Interessante"), sulla base delle informazioni ricavate applicando il mining corrispondente (con ulteriori metodologie e concetti) all'insieme L.

6.1 Prima metodologia di classificazione: “Itemset"

La prima tecnica sfrutta il metodo basato sul mining degli itemset frequenti per creare il classificatore corrispondente; il log degli allarmi passati deve essere trattato come un database di transazioni (come descritto nel paragrafo 4.1) che viene indicato con D(L). Per indicare invece l’insieme finale degli itemset frequenti ottenuto dall’algoritmo di mining corrispondente viene utilizzato il termine F(D(L),s) dove s è la soglia minima impostata del supporto.

6.1.1 Prima versione

La classificazione più semplice di ciascun allarme nel log da classificare si basa sull’idea che un allarme è “Interessante” solo quando si verifica con poca frequenza, invece è “Routine” quando fa parte dell’enorme mole di allarmi che si ripetono più spesso, quelli ridondanti che occorrono frequentemente, e quindi teoricamente ignorabili.

E’ opportuno dare una definizione specifica relativa ad un pattern che riesce a raccogliere in se la caratteristiche di un allarme. In particolare, considerando:

- un allarme A = (i,X);
- un pattern P caratterizzato dalla stesso formato dell’itemset X, ovvero una tupla P = {(A_{id},1),(A_{srcIP},2),…} dove A_α può assumere anche il simbolo asterisco ‘*’.

si può affermare che A soddisfa la condizione di match con P se $P \subseteq A$, ovvero se, per ogni attributo α, $P_\alpha = A_\alpha$ oppure P_α = ‘*’.

Di conseguenza questo si traduce col dire che, una volta creato F(D(L),s), un allarme A è di tipo “Routine” quando rientra nei pattern frequenti di F(D(L),s), cioè se esiste almeno un itemset frequente $f \in F(D(L),s)$ (rappresentato dal pattern P) per cui A soddisfa la condizione di match con lo stesso P; altrimenti, se questo controllo produce esito negativo per tutti gli f, significa che A è un allarme poco frequente e quindi può rappresentare una vera minaccia da tenere in maggior considerazione.

6.1.2 Seconda versione

Il metodo appena descritto si basa però su un’idea troppo semplicistica, in quanto possono verificarsi delle particolari condizioni che fuorviano la corretta classificazione. Tali condizioni possono presentarsi ad esempio quando si verifica un attacco di tipo intensivo oppure quando si ha una eccessiva generalizzazione dei pattern prodotti dal mining precedente; queste portano in entrambi i casi a trattare allarmi effettivamente interessanti invece come allarmi di tipo “Routine”.

Se durante la rilevazione di ogni allarme A è presente un attacco di tipo intensivo, questa tipologia di attacco può generare molti allarmi che a loro volta portano alla creazione (nell’algoritmo di mining degli itemset frequenti) di itemset frequenti con quelle caratteristiche, nonostante l’attacco si verifica per sua natura solamente per un breve lasso di tempo. Un allarme futuro che presenta le caratteristiche dell’attacco, può facilmente verificare la condizione di match, perché è presente l’itemset tra quelli

frequenti e quindi può essere erroneamente etichettato come “Routine” quando invece è un allarme altamente “Interessante”.

Inoltre, alcuni allarmi presenti in F(D(L),s) possono essere troppo generici, soprattutto quando l’attributo che presenta un valore variabile è il campo Id; questo implica che tutti i pattern che presentano gli stessi valori fissi ma con un qualsiasi valore relativo all’attributo Id, vengano trattati alla stessa maniera perché rientrano sotto questo pattern generico (ovvero sono i suoi sottopattern); ciò si rivela insensato.

Perciò è necessario che vengano generati possibilmente dei pattern caratterizzati da elementi più specifici, e per far questo è necessario apportare delle modifiche alla versione iniziale semplice del classificatore, trattando il log (rappresentato dall’insieme di allarmi L) dal quale si ottengono le informazioni per il training come una sequenza di finestre temporali e quindi partizionandoli in rispettive porzioni. Tale concetto viene ora spiegato con più precisione.

In generale, gli algoritmi assumono che ogni voce di un log abbia due attributi: il tempo di occorrenza dell’evento e il tipo. Come detto precedentemente (paragrafo 3.2.2) vengono analizzati gli eventi in base alla linea; per quanto riguarda il primo parametro invece, il log stavolta non viene trattato come un insieme unico viene suddiviso in s porzioni e in base a due possibilità:

- suddivisione in s porzioni non sovrapposte in base a certi criteri. Un pattern frequente si ottiene quando una certa combinazione di tipi di evento è presente al’interno di almeno s porzioni.
- il log viene considerato come un insieme di finestre sovrapposte, ciascuna di esse contenente eventi che accadono in un lasso di tempo di t secondi. Allo stesso modo, si è in presenza di un pattern frequente se la combinazione di tipi di evento è presente al’interno di almeno s finestre.

In questa occasione viene utilizzata la prima opzione, ovvero il log L viene considerato seconda una serie di finestre temporali distinte $t_s…t_e$ e viene diviso in m porzioni corrispondenti a finestre temporale ciascuna caratterizzata dalla stessa dimensione d, che è stata impostata ad un valore temporale pari ad un’ora.

Dato il tempo t e l’intervallo di tempo d, la porzione $L_{t,d}$ contiene tutti gli allarmi pervenuti dall’ IDPS tra il tempo t e quello t+d(escluso), ovvero:

$L_{t,d} = \{A \mid A \in L, t \leq A_{time} < t+d\}$

Ogni porzione P_i rappresenta il corrispondente insieme degli allarmi $L_{ts+i*d),\ (ts+i*d)+d}$ ovvero l’insieme degli allarmi per la finestra temporale che inizia all’istante ts+i*d e termina all’istante (ts+i*d) + d.

$L_{ts+i*d),\ (ts+i*d)+d} = \{A \mid A \in L, t \leq A_{time} < t+d\}$

Questo insieme costituisce il database delle transazione $D_i = D(L_{ts+i*d),\ (ts+i*d)+d})$.

A questo punto, ogni database D_i viene trattato singolarmente e sottoposto all’algoritmo di mining degli itemset frequenti. Viene creato il corrispondente insieme di itemset frequenti $F_i = F(D_i,s)$.

Viene creata una struttura totalFrequentItemset data dall’unione di tutti gli insiemi degli itemset frequenti F_i, in modo da eliminare tutti i pattern frequenti che eventualmente si ripetono tra le varie porzioni. In ogni caso per ogni pattern rimasto nella struttura, viene mantenuto il contatore delle occorrenze nelle porzioni che indica quante porzioni contengono quel pattern.

Tale informazione viene utilizzata nel passo successivo ed è utile per vedere qual è il grado di rilevanza di ogni pattern; in questo passo, infatti, vengono scanditi tutti i pattern frequenti presenti in totalFrequentItemset e viene costruita la struttura relevantFrequentItemset. Questa consiste nella struttura totalFrequentItemset dalla quale però vengono eliminati i pattern frequenti che presentano un valore del contatore inferiore alla soglia minima impostata k. Il valore k rappresenta il grado di rilevanza che deve raggiungere un pattern per ritenersi importante all'interno del log considerato nella sua globalità. In questo modo vengono mantenuti solamente quelli che risultano presenti in almeno k porzioni. Inoltre, vengono scartati anche quei pattern che presentano un valore variabile (simbolo asterisco) per l'attributo nella prima posizione, ovvero l'Id.

In questo modo, impostando dei valori adeguatamente grandi per i diversi parametri (il lasso di tempo che definisce la dimensione di ogni porzione, il numero di porzioni, il grado di rilevanza) nella nuova struttura relevantFrequentItemset viene mantenuto l'insieme di tutti quei pattern frequenti e rilevanti che possono corrispondere facilmente ad allarmi di tipo "Routine".

Una volta creato relevantFrequentItemset, un allarme A è di tipo "Routine" se viene soddisfatta la condizione di match per almeno uno dei pattern frequenti presenti nello stesso relevantFrequentItemset; altrimenti, se questo controllo fallisce per tutti gli f, implica che A è un allarme potrebbe essere di tipo "Interessante".

In questo modo sono state apportate delle migliorie all'algoritmo precedente. Questo metodo infatti permette la ricostruzione automatica del classificatore ogni volta che si desidera (ad esempio quotidianamente ad una precisa ora) che produce risultati corretti in quanto mantenendosi aggiornato si adatta alle eventuali modifiche dell'ambiente; ma soprattutto non è sensibile al rumore a breve termine, ovvero quando si presentano gli stessi allarmi in un breve periodi di tempo non tende ad identificarli come pattern relativi ad allarmi di "Routine"; un intemset frequente rientra in quelli della struttura finale solamente se si verifica più volte in più finestre di tempo, e quindi solo se soddisfa il grado di rilevanza impostata. Inoltre rimuove i pattern che non hanno un valore precisato per quanto riguarda l'Id; non considerandoli per i passi futuri, in quanto è importante che almeno l'ID, che rappresenta in modo unico l'allarme assume un valore ben preciso altrimenti tutte le altre informazioni non avrebbero senso quindi diventerebbero inutili.

6.1.3 Terza versione

Il metodo appena descritto, seppure ottimizzato, soffre ancora della generalizzazione elevata dei pattern, in questo caso quando le parti variabili riguardano gli attributi diversi dal campo Id. Infatti i pattern più specifici dove all'attributo variabile è associato invece un valore ben preciso, rientrino nella stessa categoria del pattern generico. Infatti, le parti variabili dei pattern (indicate dal simbolo asterisco nelle rispettive posizioni del pattern) trattano tutte le possibili alternative che si presentano per quella stessa posizione; tra queste possibilità vi possono essere degli elementi che in base alla loro tipologia di attributo (la loro posizione) devono essere trattati caso per caso perché possono nascondere delle vere insidie che nella versione generale non vengono notate.

Perciò si presenta ancora il problema della generalizzazione affrontata precedentemente, anche se in quantità minore perché riguarda attributi differenti.

La soluzione è quella di generare dei particolari insiemi (definiti attraverso la creazione di una nuova struttura), creati secondo determinati principi, che vengono utilizzati successivamente come supporto per la classificazione. Ogni insieme contiene tutte le distinte combinazioni finali (opportunamente selezionate rispetto a determinati parametri) che si verificano per un particolare valore dell'attributo ID (valore A_{ID} = id) e relative ad una data serie di attributi. Ogni insieme di questo tipo può essere visto come l'insieme frequente finale per un certo valore di ID, viene denominato frequent endpoint di id e definito H_{id}. Esso rappresenta in sostanza un filtro per gli allarmi potenzialmente di tipo "Routine" per quel valore di id; quindi viene impiegato successivamente per la classificazione di tale tipologia di allarmi.

Innanzitutto viene analizzato l'insieme di itemset frequenti e rilevanti per ricercare tutti i possibili valori distinti che l'attributo ID assume in questi itemset. Per ciascun valore i assunto da questo attributo viene effettuata una ricerca preliminare per controllare se il suo corrispondente H_{id} è già presente (ovvero se tal valore è già stato incontrato durante la scansione); se l'esito è positivo significa che il suo insieme è già stato creato a e quindi si passa al valore id successivo nella scansione, altrimenti deve essere creata tale insieme procedendo come segue.

Viene scandita l'ultima porzione del log suddiviso, cioè quella più recente e vengono considerati solamente gli allarmi contraddistinti dal valore dell'attributo ID pari a id. Se non esiste nemmeno un allarme con tale caratteristica, viene immediatamente creato l'H_{id} corrispondente, che rappresenta in questo caso un insieme definitivamente vuoto, etichettato come tale, e si passa all'analisi dell'id successivo nella scansione. Altrimenti, di questo insieme di allarmi (di dimensione d) vengono ricercate tutte le possibili distinte combinazioni che si verificano relative ad una particolare serie di attributi definita come series = $\{A_1, A_2, \ldots, A_m\}$.

La serie di attributi non è unica, ma ne esistono diverse, ciascuna pensata per raggruppare dei precisi attributi i cui corrispondenti valori si ripetono spesso simultaneamente nei vari allarmi e quindi possono rappresentare dei falsi allarmi o degli allarmi che sono di normale amministrazione. Le serie di attributi vengono applicate singolarmente e di volta in volta nell'ordine in cui vengono elencate, ma viene applicata una nuova serie solamente quando quella precedente ha fallito nella creazione dell' H_{id} relativo, ovvero se ha prodotto un H_{id} vuoto (ad eccezione della prima che ovviamente viene sempre applicata in ogni caso):

- (IP sorgente, Ip destinatario, Porta destinatario, Tipologia di protocollo)
- (Ip destinatario, Porta destinatario, Tipologia di protocollo)
- (Ip sorgente, Porta destinatario, Tipologia di protocollo)
- (Porta destinatario, Tipologia di protocollo)

Come si può osservare, se fallisce l'applicazione della combinazione precedente, quella successiva è più generica in quanto se ha fallito la ricerca utilizzando una combinazione, si tenta di applicare una meno restrittiva in modo le possibilità di ottenere risultati (anche più generici) aumentino.

Nella struttura utilizzata per creare un insieme H_{id} viene mantenuta una variabile di tipo intero per controllare a che serie di attributi corrisponde la generazione di quell'insieme ed inoltre un vettore che rappresenta le posizioni degli attributi di quella stessa serie; entrambi sono informazioni fondamentali per la fase futura di classificazione degli allarmi. Se è presente il valore -1 significa che non è presente nessun allarme nella porzione analizzata contraddistinto dallo stesso id, e di conseguenza l'insieme risulta vuoto. Se è presente un valore n = {1,2,3,4} significa che fino a quel momento è stata analizzata la combinazione n-esima. Infine, il valore 5 indica che sono state verificate tutte le combinazioni possibili ma l'insieme risulta sempre vuoto.

Una caratteristica che accomuna le varie combinazioni è che viene dato particolare attenzione ai campi relativi alla destinazione (in particolare la porta) e al protocollo perché se sono presenti elementi che si ripetono di questo tipo è più probabile che si riferiscano ad attacchi non effettivi; invece vengono presi in minor considerazione gli attributi relativi alla sorgente (addirittura trascurando totalmente la porta sorgente) in quanto gli attacchi veri e propri sono caratterizzati proprio da un ampia possibilità di valori riguardanti la sorgente e quindi difficilmente si riferiscono ad attacchi di tipo "Routine" (quelli che invece rientrano in H_{id}), bensì potenzialmente interessanti.

Per procedere sfruttando queste serie di attributi, innanzitutto viene creata una struttura tabellare che raccoglie tutti gli allarmi contraddistinti dal valore id ma mantenendo solo le informazioni necessarie in quel momento; infatti ad ogni riga della tabella corrisponde un vettore rappresentante l'allarme con valore id ma le cui celle contengono solamente i valori che l'allarme assume per la serie di attributi series in esame.

Una volta creata tale struttura, lo scopo è quello di ricercare quindi le distinte combinazioni degli elementi presenti per poi determinare le occorrenze di ogni combinazione. Questo viene fatto sfruttando l'algoritmo di mining degli itemset frequenti e applicandolo (in modalità locale) a tale struttura. Viene impostata come soglia minima di supporto il valore zero (la richiesta di tempo e memoria non è eccessiva perché gli elementi da analizzare sono solo una piccola parte di quelli totali) in modo da ricavare tutti i possibili pattern con i relativi supporti ed inserirli nella solita struttura globalFrequentItemset utilizzata per raccogliere gli itemset frequenti. Tra questi è necessario eliminare quelli contenenti parti variabili, in quanto la loro versione utile (quella con tutti gli elementi specifici) è già presente (con supporto minore) nei rimanenti pattern trovati.

Al termine del mining locale, la struttura globalFrequentItemset contiene tutte le possibili distinte combinazioni per il valore id con i relativi supporti e relative ad una particolare serie di attributi series. Per eliminare gli elementi che possono fuorviare l'analisi e quindi produrre risultati non corretti, vengono eliminate le combinazioni poco frequenti. Per prima cosa vengono scartate quelle particolarmente infrequenti (che consistono nel cosiddetto rumore) e cioè contraddistinte da un supporto molto basso pari a uno o due. Dopo tale eliminazione, il numero di combinazioni rimanenti è pari a r. Poi, tra quelle rimaste, vengono eliminate quelle che occorrono con un valore molto minore rispetto alla media. Se viene considerata la totalità degli allarmi in quella porzione con ID = id, pari a d e un fattore percentuale $l < 1$ (definito lFactor, con valori compresi tra 0.1 e 0.25), la media è data da d/r*l.

Se dopo queste due tipologie di eliminazione il numero di combinazioni rimanenti è inferiore rispetto ad un valore impostato m (definito mFactor e compreso tra 10 e 50) e si tratta di una percentuale superiore al p% (definito pFactor e compreso tra 0 e 100) del valore d, allora l'insieme delle di queste combinazioni finali rappresenta un insieme frequente finale per il valore id, ovvero l' H_{id}. Se invece queste condizioni non vengono soddisfatte l'H_{id} è da considerarsi un insieme vuoto.

Al termine dell'applicazione di ogni serie di attributi si controlla quindi che l'H_{id} risulti vuoto o meno; se risulta vuoto e l'etichetta che si riferisce al numero di transazione presenta:

- il valore n<4, si riparte applicando la successiva serie n+1 di attributi in lista; da qui in poi si procede seguendo gli stessi passi appena descritti, ma considerando la nuova serie di attributi. Perciò si inizia creando la corrispondente nuova struttura tabellare fino ad arrivare ad ottenere l'insieme finale H_{id} (che può essere nuovamente vuoto) e via così.

- Altrimenti indica che sono state verificate tutte le combinazioni possibili ma senza esito positivo perché l'insieme risulta sempre vuoto; di conseguenza l'insieme per quel id è definitivamente vuoto e si passa all'analisi dell'id successivo nella scansione.

Una volta costruiti gli H_{id} per ogni valore distinto che assume l'attributo ID nell'insieme degli itemset frequenti e rilevanti, avviene la vera e propria classificazione. Per ogni allarme A con valore A_{ID} = id, viene scandito lo stesso insieme degli itemset frequenti e rilevanti e per ciascuno di questi pattern p viene innanzitutto verificato se a verifica la condizione di match con p. Se l'esito è positivo l'allarme non viene subito considerato di tipo "Routine" e quindi etichettato come tale (come avveniva invece nella versione precedente più semplice del classificatore); bensì, si passa controlli ulteriori e più specifici sfruttando l' H_{id} corrispondente creato nella fase precedente appena descritta, per verificare ulteriormente e con maggior precisione che sia "Interessante", senza escluderlo a priori.

Il primo controllo consiste nel verificare che l' H_{id} relativo sia vuoto, e questo accade quando l'etichetta presenta i valori -1 oppure 5; il fatto che l'H_{id} risulti vuoto ma si sia verificato comunque il match con il pattern P indica che l'allarme A non fa parte di combinazioni particolari frequenti di attributi ma comunque risulti in pattern frequenti e quindi è da considerarsi "Routine".

Se l'H_{id} non risulta vuoto, il secondo controllo va ad analizzare con quale serie di attributi si è ottenuto un valido H_{id}, quindi contenente almeno un elemento. Per far questo viene considerato il numero di combinazione all'interno dell'H_{id} al quale corrisponde il vettore delle posizioni che gli attributi hanno all'interno dell'allarme. Viene scandito il vettore e per ogni posizione relativa all'attributo della serie viene verificato che all'interno del pattern in esame P il valore dell'attributo in quella posizione sia un valore fisso e non variabile.

- Se vale per tutte le posizioni della serie allora significa che il pattern p è sufficientemente specifico per rientrare in quella serie; siccome l'allarme ha verificato precedentemente il match con tale pattern, di conseguenza l'allarme è da considerarsi di tipo "Routine" e si passa all'allarme successivo
- Se invece alcuni attributi della serie nel pattern presentano valori variabili, quindi risulta troppo generico, è necessario analizzare l'allarme A e creare un vettore contenente i valori degli attributi assunti da a nelle posizioni relative agli attributi della serie. Viene ricercata quindi la presenza di tale vettore tra le combinazioni all'interno di H_{id} e se presente significa che è lo stesso allarme A rientrare tra gli elementi frequenti di H_{id} e quindi è da considerasi di tipo "Routine". A questo punto si passa all'analisi si dell'allarme successivo

Se tutti i controlli hanno dato esito negativo significa che per i vari pattern analizzati fino a quel momento l'allarme è potenzialmente di tipo "Interessante", ma non lo si può affermare con certezza finché non vengono analizzati tutti i pattern presenti nell'insieme degli itemset frequenti e rilevanti. Perciò i passa al pattern p successivo nella scansione.

Se al termine della scansione non si è mai verificato un esito positivo dei controlli per nessun pattern p preso in considerazione, allora l'allarme A è da considerarsi definitivamente un pattern "Interessante" e si passa all'allarme successivo.

6.2 Seconda metodologia di classificazione: “Cluster"

La seconda metodologia, nella sua versione finale, sfrutta sia il metodo basato sul mining degli itemset frequenti che quello del clustering; per maggior precisione l’algoritmo di clustering viene impiegato sfruttando le informazioni generati dal primo algoritmo, ovvero gli itemset frequenti e rilevanti.

Un cluster può essere rappresentato dal relativo pattern (similmente agli itemset), e quindi un allarme può soddisfare o meno la condizione di match con un determinato pattern corrispondente ad un cluster. La novità però (può essere considerata una sorta di proprietà derivata implicitamente), è presentata dal fatto che se gli allarmi che soddisfano la condizione di match con un itemset frequente rappresentato dal corrispondente pattern P appartengono ad uno o più cluster C_1,C_2,… definiti dai pattern {P_{c1},P_{c2},…, P_{cm}} allora P è contenuto nei vari P_{c1},P_{c2},…, P_{cm} in quanto gli elementi di P consistono in parole che fanno parte di ogni allarme. Di conseguenza, ogni allarme che soddisfa il match con P_{c1},P_{c2},…, P_{cm} soddisfa anche il match con P e quindi P_{c1},P_{c2},…,P_{cm} sono dei sottopattern di P (perché più specifici), ciò ad un perfezionamento nella bontà dei risultati relativi alla classificazione degli allarmi.

6.2.1 Prima versione

Nella versione basilare, la classificazione più semplice di un allarme (similmente a quanto descritto nella precedente metodologia basata esclusivamente sul mining degli itemset frequenti) si basa sul principio che un allarme è “Interessante” solo quando è presente in poche circostanze all’interno del log, altrimenti è di tipo “Routine” quando è caratterizzato da un grande densità e quindi può essere trascurato.

Il log degli allarmi passati viene considerato quindi come un insieme di punti (come descritto nel paragrafo 5.1) denominato P(L). L’insieme finale dei cluster prodotto dall’algoritmo di clustering viene definito invece C(P(L),d) dove d riguarda la soglia minima impostata della densità.

Perciò il classificatore più semplice si basa sull’idea basilare che un allarme A è di tipo “Routine” quando rientra nei cluster finali di C(P(L),d), cioè se in questi è presente almeno un cluster per cui è soddisfatta la condizione di match; se questo invece non accade per nessuno dei cluster finali, allora l’allarme è caratterizzato da bassa densità, e può essere facilmente una potenziale pericolo.

6.2.2 Seconda versione

Tale idea si rivela purtroppo troppo riduttiva perché se applicata può inesorabilmente produrre risultati errati sotto svariate condizioni (traffico intensivo, pattern generalizzati, ecc.) e quindi deve essere assolutamente vietato. Dunque è opportuno introdurre procedimenti più sofisticati.

Innanzitutto, come nel caso della prima metodologia, viene applicato l’algoritmo di mining degli itemset frequenti. Il log L viene considerato seconda una serie di finestre temporali distinte t_s…t_e e viene diviso in m porzioni corrispondenti a finestre temporale ciascuna caratterizzata dalla stessa dimensione dim (per distinguerla dalla densità d), che è stata impostata ad un valore temporale pari ad un’ora.

Ogni porzione P_i rappresenta il corrispondente insieme degli allarmi

$L_{ts+i*dim),\,(ts+i*dim)+dim} = \{A \mid A \in L, t \leq A_{time} < t+dim\}$

Questo insieme costituisce il database delle transazione $D_i = D(L_{ts+i*dim), (ts+i*dim)+dim})$.

Ora, ad ogni database Di viene applicato l'algoritmo di mining degli itemset frequenti dal quale ne deriva il relativo insieme di itemset frequenti $F_i = F(D_i,s)$. Ogni itemset frequente mantiene il numero di quante porzioni lo contengono, definito il grado di rilevanza di quell' itemset frequente. Successivamente viene creata la struttura relevantFrequentItemset che consiste nella versione ridotta di totalFrequentItemset in quanto contiene solamente i pattern frequenti rilevanti (il loro grado di rilevanza è uguale o maggiore rispetto a quello minimo impostato di valore k) e contraddistinti da un pattern dove all'attributo ID è associato un valore fisso id.

Al termine, relevantFrequentItemset contiene tutti gli itemset frequenti e rilevanti rappresentati da pattern che identificano allarmi di tipo "Routine".

Fino a questo punto, questa seconda metodologia risulta pressoché identica rispetto alla prima; ora però viene sfruttato l'algoritmo di clustering descritto nel capitolo 5, introducendo anche ulteriori procedimenti.

Le informazioni fondamentali che si ottengono dalla fase precedente sono rappresentate dall'insieme di itemset frequenti rilevanti memorizzati nella struttura relevantFrequentItemset.

L'idea centrale è quella di andare a scandire relevantFrequentItemset, e per ogni itemset frequente rilevante f definito dal pattern P, scoprire quali sono i cluster in cui possono essere raggruppati tutti gli allarmi in L che soddisfano la condizione di match con il pattern P. Ogni pattern P, infatti, esiste perché sono presenti degli allarmi particolarmente frequenti che hanno indotto la sua creazione e che vengono rappresentati dallo stesso pattern; tale insieme di allarmi viene definito L(P). Quindi, per ogni pattern P viene scandito L e vengono considerati tutti gli allarmi che vengono rappresentati da tale pattern; vengono memorizzarti in una struttura a matrice rows che rappresenta l'insieme L(P), e a questa viene applicato l'algoritmo di clustering definito da C(L(P),d) che restituisce la lista dei cluster $C_p = \{C_1,C_2,\ldots\}$ (definito dalla struttura relevantFrequentItemset) e che viene assegnata, sottoforma di lista, al rispettivo pattern P. Successivamente si procede con il cluster p successivo nella scansione di relevantFrequentItemset.

Al termine della scansione, si ottiene l'insieme totale di cluster C, costituito a sua volta dagli insiemi dei cluster corrispondenti ad ogni singolo pattern P di relevantFrequentItemset. Il fatto di creare l'insieme di cluster C_p per ogni pattern p porta al vantaggio di ridurre il problema di aver a che fare con pattern troppo generici per la classificazione.

La classificazione finale vera e propria di un allarme in questo caso si basa sul principio che un allarme è di tipo "Routine" se esiste un f all'interno degli itemset frequenti e rilevanti nella struttura relevantFrequentItemset per cui l'allarme A soddisfa la condizione di match con il pattern P che rappresenta f, quindi una condizione più specifica rispetto all'esistenza di almeno un pattern solamente frequente. Per essere considerato "Routine", l'allarme A deve inoltre soddisfare la condizione di match anche con il pattern relativo ad un cluster appartenente alla lista di cluster C_p corrispondenti a P, e quindi deve esistere anche un cluster che soddisfi tali caratteristiche.

Quindi, per prima cosa si passa con la ricerca tramite scansione di relevantFrequentItemset di almeno un pattern P relativo ad f che soddisfi il match con l'allarme e l'aggiornamento della situazione della ricerca viene indicato sfruttando una variabile booleana matchTest; se l'esito è negativo allora l'allarme viene subito etichettato come "Interessante", altrimenti potrebbe essere potenzialmente di tipo "Routine". Perciò si passa alla ricerca all'interno della lista dei cluster associata al pattern P di almeno un cluster per il

quale p soddisfi la condizione di match, sempre sfruttando la variabile match Test. Se questo controllo da esisto negativo allora l'allarme è definitivamente di tipo "Interessante", altrimenti si tratta di un allarme di tipo "Routine".

In conclusione, se almeno una delle due condizioni non viene soddisfatta, allora l'allarme è assolutamente di tipo interessate, altrimenti se soddisfa entrambe è di tipo "Routine".

6.2.3 Seconda versione

Seppure rimane sempre valida l'ipotesi (descritta nel capitolo 3 riguardo le proprietà dei dati contenuti nei log) che solo poche tipologie di allarmi generano molte voci nei log e quindi creano pochi itemeset frequenti in relevantFrequentItemset, questi possono comunque diventare assai numerosi quando viene associato un piccolo valore alla soglia minima del supporto. Una piccola parte dei pattern associati a questi itemset sono generici mentre molti sono quelli più specifici.

In questo caso affiora la pecca che può caratterizzare questo classificatore, e che consta nella possibile presenza di itemset ridondanti e quindi trascurabili se viene considerata la proprietà descritta precedentemente. Se l'allarme che soddisfa la condizione di match con un itemset frequente (rappresentato dal corrispondente pattern P) appartiene ad uno o più cluster C_1, C_2,… definiti dai pattern $\{P_{c1}, P_{c2},\ldots,P_{cm}\}$ allora P è contenuto nei vari P_{c1}, P_{c2},…,$P_{cm.}$ Quindi l'allarme che soddisfa il match con P_{c1}, P_{c2},…,P_{cm} soddisfa anche il match con P e P_{c1}, P_{c2},…,P_{cm} sono dei pattern più specifici di P.

Perciò ogni pattern specifico S contiene il pattern P_c relativo ad un qualche cluster C ottenuto da un certo pattern generico G, ovvero s è un superset per il pattern P_c.

Per costruzione G deve essere contenuto nel pattern P_c; quindi se s è contenuto in P_c, e P_c è contenuto in G, quando un allarme soddisfa il match con il pattern S, allora deve per forza soddisfare il match anche con G e con P_c (l'allarme, per come è definita la vera e propria parte del classificatore, risulta quindi di tipo "Routine"). Di conseguenza, il pattern specifico S possiede delle informazioni già presenti e quindi deve essere trattato con minor importanza rispetto ai rispettivi pattern più generici, ovviamente in base al medesimo valore dell'attributo ID.

A questo punto nella scansione dei vari pattern in relevantFrequentItemset è opportuno che vengano presi in considerazione prima i pattern generici (che contengono quindi meno elementi) rispetto a quelli più specifici, sempre a parità di valore dell'attributo ID. In questo modo, durante l'analisi dei vari pattern è molto più probabile che, attraverso i controlli che vedremo poi, molti pattern specifici vengano rilevati come ridondanti. In tal caso vengono ignorati e quindi non viene applicato l'algoritmo di clustering all'insieme di allarmi che rappresentano, con un conseguente risparmio di tempo e memoria; inoltre vengono etichettati come ridondanti all'interno della struttura relevantFrequentItemset in quanto tali non vengono presi in considerazione durante la fase finale di classificazione vera e propria.

Per far in modo che vengano analizzati prima i pattern generici rispetto a quelli specifici durante la scansione, è dunque necessario applicare un ordinamento della lista relevantFrequentItemset secondo determinati aspetti prima che inizi la scansione. Siccome la lista non è una struttura che si presta molto bene all'ordinamento degli elementi in essa contenuti, gli stessi vengono inseriti in una struttura vettore relevantFrequentItemsetArray che rappresenta esattamente il contenuto di relevantFrequentItemset e che

viene utilizzata d'ora in poi come sostituiva della lista. Tale vettore viene quindi sottoposto ad un ordinamento (di tipo bubble sort) in base a questi principi:

- i pattern vengono ordinati in ordine crescente in base alla stringa alfanumerica che rappresenta il valore che assume l'attributo ID
- se è presente lo stesso valore di ID, vengono ordinati in ordine crescente rispetto al grado di specificità in modo da avere come pattern iniziali quelli più generici. Questo si traduce nel ordinarli in senso crescente rispetto al numero di valori fissi presenti nel pattern

In questo modo si crea un vettore che rappresenta l'insieme degli itemset frequenti rilevanti relevantFrequentItemsetArray definiti dai relativi pattern $\{P_1,P_2,..,P_n\}$ e caratterizzato da un ordinamento completo dove $P_1 < P_2 < P_n$ dove n è la dimensione dell'insieme relevantFrequentItemset.

Ora, per sfruttare l'ordinamento in modo da rilevare i pattern specifici ridondanti, per ogni pattern p preso in considerazione durante la scansione di relevantFrequentItemset (prima della fase di scansione di L per creare la struttura che definisce L(P) e applicare il clustering) viene introdotto un particolare controllo. Vengono scanditi tutti i pattern P_i inferiori (in base all'ordinamento stabilito) a P e per ciascuno viene analizzata a sua volta la lista di cluster a lui associata prodotta dal clustering precedente applicato a P_i. Se tra la totalità di questi cluster esiste almeno un cluster contenuto nel pattern P in esame, allora significa che le informazioni di P sono già presenti perché si tratta di un pattern specifico ed è già presente quello generico. In tal caso viene ignorato perché ridondante e viene inserito in una lista apposita che contiene queste tipologia di pattern. Inoltre la stringa globale che lo rappresenta viene posta al valore nullo (metodo per etichettarlo come ridondante e quindi per poterlo escludere durante la fase finale di classificazione) e si passa al pattern P successivo nella scansione. Altrimenti il pattern contiene informazioni utili non ritrovate in precedenza e quindi si procede con lo scandire L per creare la struttura che rappresenta L(P), applicare il clustering e associare i cluster trovati (sottoforma di lista) al pattern P.

La classificazione finale di un allarme segue lo stesso principio di quella affronta nel classificatore precedente; la differenza è che in questo caso, quando viene effettuata la scansione all'interno di relevantFrequentItemsetArray per vedere se viene soddisfatta la prima condizione, per ogni pattern viene innanzitutto verificato che non sia un pattern ridondante, andando a controllare che la stringa globale che lo rappresenta risulti vuota; se lo è significa che è ridondante e quindi viene prontamente ignorato.

6.2.4 Terza versione

Nonostante i miglioramenti apportati, anche questo classificatore necessita di ulteriore sviluppo. Infatti, bisogna tenere ben presente che quando si applica l'algoritmo di clustering ai vari insiemi di allarmi che vengono rappresentati dai diversi pattern presenti in relevantFrequentItemsetArray, le dimensioni di questi insiemi e le caratteristiche dei pattern rispettivi possono variare anche sensibilmente in base al tipo di pattern stesso in esame di volta in volta. Quindi è impossibile applicare un unico valore di soglia minima di densità che sia perfettamente adatto a tutte le occasioni, altrimenti può produrre dei risultati diversificati.

Quindi l'idea è quella di applicare, ad ogni insieme di allarmi L(P) rappresentati dal pattern P, più volte l'algoritmo di clustering e ciascuna applicazione è contraddistinta da un diverso valore di soglia minima di densità. Per ogni insieme L(P) viene quindi applicata una serie di soglie, partendo dal valore maggiore della serie; i risultati ottenuti vengono ritenuti migliori, rispetto a quelli raggiunti fino a quel momento dai vari clustering, se le caratteristiche di questo nuovo insieme l'insieme appena ottenuto soddisfano determinate

condizioni in base a diversi parametri. Se l'esito è positivo, allora questo nuovo insieme sostituisce quello fino a quel momento ritenuto il migliore (partendo da un insieme vuoto) e anche la soglia di densità migliore è ritenuta quella in esame (partendo da un valore iniziale pari a -1). Al termine delle varie applicazioni di clustering, si ottiene perciò il miglior insieme di cluster per l'insieme di allarmi L(P), al quale corrisponde il migliore valore della soglia minima di densità. Se però, per ciascuna densità, non viene rilevato nessun insieme di cluster soddisfacente le varie condizioni, al termine dei vari clustering per le varie densità, il cluster migliore non esiste perché risulta un insieme vuoto (con densità pari a -1); in questa occasione come cluster migliore viene ritenuto lo stesso pattern P (rappresentante l'insieme di allarmi L(P) su cui vengono applicati gli stessi clustering). Questo significa che il cluster migliore che rappresenta il pattern non porta ad un minor numero di condizioni soddisfatte di match degli allarmi con il pattern P durante la classificazione finale; infatti, ogni allarme che soddisfa il match con P soddisfa obbligatoriamente anche la condizione di match con i suoi cluster, perché si tratta del medesimo unico elemento P.

Per valutare la bontà dei risultati ottenuti, ovvero l'insieme di cluster C(L(P),d) (relativi all'insieme di allarmi L(P) per una determinata soglia di densità d) generati al termine dell'applicazione di ogni clustering C(L(P),d), vengono valutati diversi parametri. Se ciascuno i valori di questi parametri soddisfano tutte le rispettive condizioni allora C(L(P),d) è ritenuto il migliore fino a quel momento.

Innanzitutto vengono considerate le dimensioni dei cluster C ottenuti e viene considerata quella minima definita D. Il valore D indica che tutti i cluster ottenuti contengono almeno D punti al loro interno; quindi più grande è questo valore e più grande è il numero di value fissi nei pattern corrispondenti ai vari cluster. Questo significa che i pattern sono più specifici, e quindi tale situazione è altamente preferibile, perché devono essere più specifici possibile. Quindi, la condizione che deve essere soddisfatta consiste nel fatto che tale valore D deve essere uguale o maggiore rispetto a quello fin'ora raggiunto dall'applicazione dei vari clustering, partendo da un valore iniziale pari a 0.

Il secondo parametro (definito maxCuster) importante consiste nella dimensione dell'insieme di cluster creati, ovvero nel numero degli stessi cluster generati. Tale valore è opportuno che sia piuttosto limitato, cioè che si abbia a che fare con un numero moderato di cluster. Quindi, per rientrare nella condizione voluta, tale valore deve essere uguale o inferiore rispetto ad una soglia massima impostata di cluster; un valore massimo ritenuto buono è compreso tra i 20 ed i 30 cluster, con preferenza meno restrittiva verso quest'ultimo numero.

Infine, un parametro rilevante è rappresentato dal numero di outlier generati dal clustering (definito pctOutlier). E' importante che il numero di cluster copra la maggior parte della totalità degli allarmi, mentre l'insieme rimanente di punti (gli outlier) rappresenti solo una minima parte di tutti gli allarmi. Il numero di outlier rilevati quindi deve rappresentare una quantità che rientra al di sotto di una percentuale massima agli allarmi totali.Un buon valore del numero che identifica la percentuale massima è compreso tra 1 e 5 (quindi tra l'1% ed il 5%), con predilezione meno limitativa verso quest'ultimo valore.

In questo modo viene creato un metodo che si può considerare auto-adattivo in quanto gestisce situazioni facilmente contraddistinte da peculiarità molto differenti tra loro (in termini della dimensione di ciascun insieme di allarmi e delle caratteristiche del pattern relativo). Infatti, applicando diverse volte il clustering con diverse soglie, viene ricercato il migliore valore di soglia minima di densità che produce il conseguente miglior insieme di cluster per ogni insieme di allarmi rappresentato dal corrispondente pattern P.

L'applicazione di questo perfezionamento avviene per ogni pattern p preso in considerazione durante la scansione di relevantFrequentItemsetArray, nel momento successivo alla fase di scansione di L per creare la

struttura che definisce L(P). In questo caso, infatti, invece di applicare per ogni insieme L(p) un solo clustering caratterizzato da una soglia minima di densità, viene richiamata una funzione che applica al suo interno i diversi clustering (caratterizzati dalle diverse soglie minime di densità) all'insieme L(P) e ritorna il migliore insieme di cluster finale. Al suo interno viene scandito il vettore ricevuto in ingresso e contenente la serie di soglie minime di densità; per ciascuna di esse viene effettuato un clustering sull'insieme L(P) ricevuto anch'esso in ingresso, viene controllato il soddisfacimento delle diverse condizioni (precedentemente descritte) utilizzando anche i diversi parametri, e vengono aggiornati la densità e l'insieme di cluster migliori ottenuti fino a quel momento.

Al termine dei clustering per le varie soglie di densità, viene ritornato l'insieme di cluster finale ritenuto migliore per l'insieme di allarmi L(P) e viene associato all'itemset frequente caratterizzato dal pattern P presente in relevantFrequentItemsetArray sottoforma di lista di cluster; se invece nessun insieme di cluster creato per ogni soglia non soddisfa le condizioni, viene ritornato il valore nullo, che viene associato anch'esso alla lista di cluster del pattern p, in modo indicare che suo cluster di appartenenza è costituito dallo stesso pattern P.

Il principio della classificazione finale rispecchia quello della versione precedente; la differenza è che in questo caso, quando viene effettuata la scansione all'interno della lista del pattern P (per la verifica della seconda condizione), se questa lista risulta vuota viene controllato se A soddisfa il match solamente con il pattern P (è l'unico cluster che rappresenta lo stesso P).

6.3 Gestione dei risultati

Per entrambi le tipologie, i risultati della classificazione riguardano diversi aspetti e vengono inseriti in differenti file. L'idea principale si basa comunque sul fatto di dare maggiore importanza agli allarmi di tipo "Routine", in quanto il loro rilevamento è il cuore della classificazione.

Se il file di log sottoposto alla classificazione è definito "nomeSettimana" e la metodologia "nomeMetodologia" ("Itemset" per il primo tipo di classificazione, "Cluster" per il secondo) viene creata una cartella "nomeSettimana_nomeMetodologia_Results) nella directory dove è presente il log, contenente:

- un file "nomeSettimana_nomeMetodologia_Class" analogo al file di log "nomeSettimana" dove però al termine di ogni allarme è presente l'etichetta prodotta dalla classificazione per quell'allarme ("Interessante" oppure "Routine")
- un file "nomeSettimana_nomeMetodologia_Class_Summary.txt" contenente:
    - un riepilogo dei risultati totali ottenuti per la classificazione di quel log (numero totale di allarmi, numero di allarmi di tipo "Interessante", di tipo "Routine", percentuale di tipo "Routine")
    - un elenco di tutte le regole che hanno prodotto quelli allarmi con i corrispondenti risultati (secondo lo stesso principio dei risultati totali)
- una cartella "nomeSettimana_nomeMetodologia_Rules" contenente cartelle ciascuna definita dal codice identificativo "idRegola" della rispettiva regola. Ogni cartella contiene a sua volta due file:
    - un file "idRegola_Class.txt" contenente gli allarmi relativi a quella regola con la rispettiva etichetta. A questi però viene aggiunto il numero di riga indicante la posizione nel file di log

originale; inoltre, se l'allarme è di tipo "Routine" viene riportato il motivo per cui è stato ritenuto tale, in modo da indicare come l'algoritmo ha funzionato e quali pattern viene lo rappresentano per le sue caratteristiche

Nel caso della metodologia "Itemset" viene indicato il pattern relativo all'itemset generico che lo identifica e con cui ha soddisfatto il match durante l'algoritmo. Inoltre, se presente, viene notificato anche il pattern corrispondente all'itemset più specifico che è riuscito a soddisfare il match con un elemento degli insiemi Hid; può essere di due tipi:

- (P) quando è il pattern che soddisfa il match con un elemento di Hid
- (A) quando è l'allarme che soddisfa il match con un elemento di Hid

Se viene utilizzata la metodologia "Cluster", viene indicato anche in questo caso il pattern relativo all'itemset generico che ha soddisfatto il match; in questa occasione però viene notificato sempre e comunque anche il pattern più specifico corrispondente al cluster che rappresenta quell'allarme. Infatti, se l'allarme è di tipo "Routine" significa che ha soddisfatto, senza ombra di dubbio, il match sia con un itemset (pattern generico) che con il rispettivo cluster (pattern più specifico)

- un file "idRegola_Final.txt" che riassume tutti i pattern (ciascuno costituito da quello generico ed eventualmente da quello specifico) incontrati nel file descritto precedente "idRegola_Class.txt" con il rispettivo valore di occorrenza. Vengono ottenuti effettuando sostanzialmente il mining di quelli presenti in quel file.

7 Analisi dei risultati

In questo capitolo vengono riportati alcuni dei molteplici risultati ottenuti con l'applicazione dei metodi fin qui descritti. Innanzitutto vengono descritti quali file di log vengono utilizzati per i vari test ed i parametri da impartire per sfruttare gli applicativi, per poi passare alla vera e propria analisi dei dati.

L'analisi iniziale si riferisce all'applicazione delle due tecniche di data mining, dove viene effettuato un breve confronto.

Successivamente vengono affrontati i risultati più complessi raggiunti nella classificazione dei falsi allarmi sfruttando le due metodologie "Itemset" e "Cluster". Antecedente a questo studio, viene descritto come si è raggiunta una configurazione adeguata delle due metodologie in modo da produrre i migliori risultati possibili; inoltre, vengono utilizzati dei casi di studio per l'ulteriori verifica della bontà dei dati ottenuti. Anche in questa circostanza viene condotta una significativa comparazione tra le due tecniche.

7.1 File di log utilizzati e comandi

I log analizzati sono stati gentilmente forniti dal Tecnico Amministrativo responsabile dell'infrastruttura di rete del Dipartimento di Scienze Ambientali, Informatica e Statistica con sede nel Polo Scientifico di via Torino, Mestre(VE), Italia.

I log sono stati ottenuti impiegando l'utilizzo dell'applicativo Snort (versione 2.9.1.2) sul server presente in Dipartimento dedicato esclusivamente alla funzione di firewall dello stesso Dipartimento. Snort (più precisamente il suo sensore), in questo modo, riesce a monitorare l'intero traffico entrante ed uscente dalla sede e che riguarda quindi tutti i diversi server presenti.

Le regole impiegate sono quelle ufficiali sviluppate dal gruppo di ricerca VRT (Sourcefire Vulnerability Research Team) aggiornate a dicembre 2011. Consistono in regole (quelle attivate sono circa 200) verificate e testate rigorosamente, ma sono adatte per una prima configurazione generale di qualsiasi sistema; devono essere quindi opportunamente regolate in base alle diverse caratteristiche della situazione dove vengono impiegate.

Riguardo la collocazione di questo NIDPS (facendo riferimento al paragrafo 2.4.1), in questo caso Snort viene collegato all'interno della macchina dedicata rappresentante dal firewall (in base alla prima problematica); inoltre, (all'interno del firewall) si colloca concettualmente a monte, in quanto è a diretto contatto con il traffico che arriva dall'esterno prima che giunga al firewall che protegge l'intera rete di Dipartimento (in riferimento alla seconda problematica di collocazione di un NIDPS).

I log sfruttati per gli scopi della Tesi consistono essenzialmente in due file; tali file sono relativi al rilevamento degli allarmi per un arco di tempo di due settimane (che possono essere definite rispettivamente settimana 1 e settimana 2) tra il giorno 17/01/2012 ed il giorno 31/01/2012. Ogni log, relativo ad una settimana, presenta circa 100.000 allarmi.

Per quanto riguarda il test delle tecniche di data mining, è stato utilizzato il log relativo alla settimana 1; in relazione alla verifica delle metodologie per la classificazione dei falsi allarmi, è stato impiegato quello

corrispondente alla settimana 1 per la fase di training, mentre quello della settimana 2 per la fase di test. Ovviamente, questi log vengono sottoposti alla fase di preprocessamento prima di essere sfruttati dagli applicativi (come descritto nel paragrafo 3.2.1).

Infine, questi sono i diversi parametri che possono essere impartiti agli applicativi realizzati (valgono per entrambi gli applicativi, ad eccezione di quelli esplicitamente dichiarati solo per il caso singolo):

- "-m": indica che viene applicata solamente la tecnica di data mining
- "-c": indica che viene applicata la metodologia di classificazione
- "-s" seguito dal valore del supporto (in termini di percentuale, ma senza il simbolo "%")
- "-i" seguito dal percorso del file di log utilizzato nella fase di training durante la classificazione
- "-i" seguito dal percorso del file di log utilizzato nella fase di test durante la classificazione
- "-n" seguito dal numero di slice
- "-w" seguito dalla dimensione del wordsVector
- "-o": indica che viene applicata la modalità "più cluster", altrimenti, di default, è attiva quella di tipo "unico cluster" (solo nel caso del secondo applicativo)
- "-p": indica che viene applicata la fase di perfezionamento dei cluster (solo nel caso del secondo applicativo)

7.2 Risultati legati alle tecniche di data mining

L'applicazione delle tecniche di data mining produce dei dati esclusivamente oggettivi, ovvero non possono essere soggetti a possibilità di interpretazione (a differenza di quelli successivi riguardo il rilevamento dei falsi allarmi). Perciò viene effettuato un confronto tra il mining degli itemset frequenti ed il clustering utilizzando gli stessi valori sia per la soglia minima del supporto che per quella di densità.

Supporto/Densità	N.itemset	N.cluster
2%	84	69
5%	39	34
10%	8	7

Tabella 7.1 Confronto tra le due tecniche di data mining variando supporto/densità

Ora, viene riportato lo screenshot relativo al contenuto dei file (la parte finale del file, dove sono presenti i pattern più specifici) dove vengono memorizzati i risultati dell'applicazione delle due tecniche. I file sono quindi rispettivamente (come indicato nei paragrafi 5.3.5 e 6.3) "nomeSettimana_Itemset_Mining"e "nomeSettimana_Cluster_Mining". Viene utilizzato, a titolo di esempio, il medesimo valore percentuale relativo ad una soglia pari al 2%. Per quanto riguarda il clustering, viene utilizzata l'opzione "più cluster" in quanto si avvicina maggiormente alla modalità di creazione dei pattern relativi agli itemset frequenti, e quindi si ottiene un confronto più significativo.

```
- Itemset 81 -

[1:100000160:2] 157.138.20.12 * * * UDP

Supporto: 2248
--


- Itemset 82 -

[1:100000160:2] 157.138.20.12 * * 53 *

Supporto: 2205
--


- Itemset 83 -

[1:100000160:2] 157.138.20.12 * * 53 UDP

Supporto: 2203
--


- Itemset 84 -

* * * 157.138.20.183 3389 TCP

Supporto: 2160
--


-- Numero di itemset: 84 --
```

Figura 7.1 Parte finale dei risultati (con supporto del 2%) contenuti in "nomeSettimana_Itemset_Mining"

```
- Cluster 66 -

[1:1852:3] * * * 80 TCP

Densita': 2456
--


- Cluster 67 -

[1:100000135:1] 137.138.41.57 * 157.138.20.12 143 TCP

Densita': 2317
--


- Cluster 68 -

[1:100000160:2] 157.138.20.12 * * * UDP

Densita': 2271
--


- Cluster 69 -

[1:100000160:2] 157.138.20.12 * * 53 UDP

Densita': 2205
--


-- Numero di cluster finali: 69 --
```

Figura 7.2 Parte finale dei risultati (con supporto del 2%) contenuti in "nomeSettimana_Cluster_Mining"

Inoltre, per quanto riguarda il clustering, vengono riportati i risultati ottenuti, sotto le stesse condizioni, ma applicando l'opzione "unico cluster", che caratterizza (come principio di funzionamento) la maggior parte degli algoritmi di clustering.

Densità	N. cluster
2%	19
5%	9
10%	5

Tabella 7.2 Risultati del clustering al variare del supporto e in modalità "unico cluster"

7.3 Risultati legati al rilevamento dei falsi allarmi

7.3.1 Configurazione adeguata delle metodologie

Il rilevamento dei falsi allarmi e di quelli irrilevanti (in particolare per il sistema NIDPS in utilizzo quale Snort) si rileva un aspetto molto difficile da valutare perché fortemente influenzato da una grande varietà di fattori. Di conseguenza, i risultati che si possono ottenere non possono subire una valutazione esclusivamente oggettiva e quindi perfettamente esatta; bensì devono essere considerati come un'approssimazione di quelli reali e devono essere trattati come tali.

Ciascuna metodologie implementata ("Itemset" e "Cluster") è contraddistinta da diversi parametri, che, modificati, portano a risultati finali anche molto differenti tra loro. Perciò, è necessario innanzitutto tarare i due metodi (impostando i diversi parametri) in modo che i risultati ottenuti siano identici (o per lo meno simili in modo soddisfacente) a classificazioni già effettuate e sicuramente esatte.

Per tale motivo vengono sfruttati i dataset DARPA dell'anno 1999. Essi consistono in particolari dataset contenenti traffico di rete (sia benevolo che malevolo, in formato tcpdump) generato opportunamente dal cosiddetto Cyber Systems and Technology Group (nel laboratorio di ricerca e sviluppo di MIT Licoln finanziato dal governo federale degli Stati Uniti d'America), con lo scopo di "addestrare" e testare le prestazioni dei sistemi IDPS.

Vengono utilizzate la settimana 2 e 3 per la fase di training, mentre la 4 e la 5 per la fase di test. Per queste ultime due è presente una lista che indica quali minacce reali sono effettivamente contenute nelle due corrispondenti settimane.

Questa è una tabella riassuntiva che mostra, per ogni classe di attacco, qual è il numero di attacchi reali effettuati e l'insieme specifico delle minacce che rientrano in quella classe. Come si può vedere, ogni minaccia (in totale 190) viene definita da una vera e propria stringa, a differenza di Snort che li identifica tramite un codice identificativo.

Classe di Attacco	Tipi specifici di attaco
DoS (63)	arppoison, apache2, back, crashiis, dosnuke, land, smurf, neptune, teardrop, tcpreset, processtable, mailbomb, pod, syslogd, selfping, warezclient, udpstorm
Probe (37)	illegal-sniffer, ipsweep, lsdomain, ntinfoscan, msscan, ntinfoscan, queso, satan
R2L (53)	dict, framespoof, ftpwrite, guest, httptunnel, named, netcat, netbus, phf, sendmail, imap, ncftp, sshtrojan, snmpget, xlock, xsnoop, ppmacro
U2R (37)	casesen, eject, fdformat, ffbconfig, nukepw, ntfsdos, loadmodule, sechole, ps, secret, xterm, yaga, sqlattack, perl

Tabella 7.3 Classi di attacco (e relative specifiche minacce) presenti nel dataset DARPA del 1999

I file in formato tcmdump relativi alla due settimane di training vengono poi unificati in unico file (stessa cosa per quelli relativi alla fase di test). I due file ottenuti vengono quindi analizzati da Snort, che produce i relativi log di allarmi.

Questi dataset sono stati ottenuti molti anni fa, ma per l'obiettivo della Tesi ciò non si rivela un grande problema in quanto, una volta ottenuti gli allarmi rilevati da Snort, l'analisi si basa esclusivamente su di loro. Analizzando la lista della effettive minacce con gli allarmi che Snort produce (attraverso un confronto tra i timestamp e la similarità della tipologia di attacco), si nota l'unico aspetto negativo: Snort riesce sia a produrre allarmi per minacce non presenti (falsi allarmi) che ad ignorare alcune effettivamente contenute (falsi negativi). Questo perché le regole che utilizza Snort sono aggiornate e quindi molte minacce possono essere diventate ormai innocue (oppure le loro caratteristiche sono mutate), mentre altre sono evidentemente nate successivamente. In questo modo però si sfruttano dati e classificazione sicuri e testati, altrimenti molto complessa da ottenere.

Alla fine, Snort rileva solamente 145 minacce rispetto alle totali 190, e le rimanenti 196.065 consistono in falsi allarmi, per un totale di 196.210 allarmi complessivamente.

A questo punto vengono utilizzate le metodologie "Itemset" e "Cluster": ciascuna analizza come file di tranining il log prodotto da Snort relativo alle settimana 2 e 3, mentre file di test il log generato da Snort e corrispondente alle settimane 4 e 5.

Se si identifica con:

- VP (veri positivi): numero di allarmi ciascuno correttamente etichettato come "Interessante"
- FP (falsi positivi): numero di allarmi ciascuno erroneamente etichettato come "Interessante"
- FN (falsi negativi): numero di allarmi ciascuno erroneamente etichettato come "Routine"
- VN (veri negativi): numero di allarmi ciascuno correttamente etichettato come "Routine"

Si possono calcolare i valori delle metriche quali la precisione e la recall. Il valore della precisione si ottiene tramite la formula:

$$Precisione = \frac{VP}{VP + FP}$$

Mentre il valore della recall tramite:

$$Recall = \frac{VP}{VP + FN}$$

Vengono effettuate molteplici test variando i diversi parametri, e i migliori risultati ottenuti per le due metodologie vengono riportati in tabella:

Metodologia	VP	FP	FN	VN	Precisione	Recall
"Itemset"	131	13	14	196.053	0,907	0,903
"Cluster"	136	12	9	196.052	0,918	0,937

Tabella 7.4 Verifica della bontà dei risultati ottenuti

Per quanto riguarda la metodologia “Itemset”, questi risultati si sono ottenuti associando ai diversi parametri questi valori:

- supporto: 3%
- numero di porzioni: 168 (in quanto l’obiettivo è ottenere porzioni temporali di un’ora ed il log di training riguarda una settimana)
- grado di rilevanza k: 168/2 = 84
- lFactor: 0.25
- mFactor: 50
- pFactor: 10

Mentre in riferimento alla metodologia “Cluster”, i valori adottati sono:

- supporto: 3%
- densità: 80%, 40%, 10%
- numero di porzioni: 168 (per il medesimo motivo)
- grado di rilevanza: 84
- maxCluster: 30
- pctOutlier: 5

7.3.2 Sfruttamento dei casi di studio

Per avere comunque una controprova che verifichi la bontà dei risultati ottenuti, sono stati presi in considerazione i dati contenuti in alcune delle pubblicazioni scientifiche ([8], [7]) utilizzate anche per la fruizione delle informazioni su come attuare le varie tecniche implementate.

Le condizioni in cui l’autore delle pubblicazioni effettua i propri test le seguenti:

- la versione di Snort è differente
- opera su ben tre sensori di una grande istituzione finanziarie
- ciascun sensore è caratterizzato dall’utilizzo di circa 16.000 regole (la maggior parte delle quali facilmente differenti da quelle utilizzate nel caso della Tesi)
- ciascun log genera circa 75.000 allarmi per settimana

Analizzando questi dati e paragonandoli con quelli della Tesi, risulta evidente che le caratteristiche sono molto differenti rispetto a quelle presenti nella Tesi. I risultati ottenuti dall'autore riguardo la percentuale di falsi allarmi vengono tuttavia presi in considerazione in quanto, come si vedrà poi, sono realmente molto simili a quelli della Tesi.

Per testare ulteriormente la validità dei risultati, viene utilizzata una concisa informazione che gli esperti del gruppo VRT ([10]) associano ad una regola per indicare se tale regola è noto produca o meno falsi positivi.

Inoltre, vengono sfruttati dei dati gentilmente forniti dal gruppo che opera nel progetto snortattack ([11]). I dati consistono nell'associare ad ogni regola una percentuale (non specifica ma rientrante in un certo valore) di quanti falsi allarmi quella regola genera in base a dati statistici e all'esperienza di questo gruppo di esperti.

7.3.3 Regole rilevate

In questa sezione, innanzitutto vengono riportate le regole che producono gli allarmi rilevati nel file di log sottoposto alla fase di test. Tali regole, possono essere derivate dopo la classificazione, tramite il file riassuntivo "nomeSettimana_nomeMetodologia_Class_Summary.txt" oppure tramite le cartelle contenute nella directory "nomeSettimana_nomeMetodologia_Rules".

Le regole incontrate vengono raggruppate in base alla classe di attacco alle quali appartengono e, per ciascuna di esse, viene indicato:

- il codice identificativo Id
- il nome del file delle regole dove essa è contenuta (al quale va aggiunta l'estensione ".rules")
- la descrizione, ovvero il messaggio che indica in cosa consiste quell'allarme rilevato

Id	File	Descrizione
466	icmp	ICMP L3retriever Ping
469	deleted	ICMP PING NMAP
621	deleted	SCAN FIN
624	deleted	SCAN SYS FIN
882	web-cgi	WEB-CGI calendar access
1112	deleted	WEB-MISC http directory traversal attempt
1113	deleted	WEB-MISC http directory traversal att inf
1201	attack-responses	INDICATOR-COMPROMISE 403 Forbidden
1213	web-misc	WEB-MISC backup access INF LEAK
1301	web-php	WEB-PHP admin.php access
1411	snmp	SNMP public access udp
1417	snmp	SNMP request udp
1418	snmp	SNMP request tcp
1421	snmp	SNMP AgentX/tcp request
1616	dns	DNS named version attempt

Tabella 7.5 Regole relative alla classe di attacco attempted-recon (Attempted Information Leak)

Id	File	Descrizione
937	web-frontage	WEB-FRONTPAGE _vti_rpc access
962	web-frontage	WEB-FRONTPAGE shtml.exe access access
972	deleted	WEB-IIS %2E-asp access
990	web-frontage	WEB-FRONTPAGE _vti_inf.html access
1042	web-iis	WEB-IIS view source via translate header
1078	sql	SQL counter.exe access
1149	web-cgi	WEB-CGI count.cgi access
1288	web-frontage	WEB-FRONTPAGE /_vti_bin/ access .
1560	web-misc	WEB-MISC /doc/ access
1852	web-misc	WEB-MISC robots.txt access
1872	web-misc	WEB-MISC Oracle Dynamic Monitoring Services dms access
2077	web-php	WEB-PHP Mambo upload.php access
2281	web-php	WEB-PHP Setup.php access
2393	web-php	WEB-PHP /_admin access

Tabella 7.6 Regole relative alla classe di attacco web-application-activity (Access to a potentially vulnerable web application)

Id	File	Descrizione
240	ddos	DDOS shaft agent to handler
2418	misc	Microsoft Windows Terminal Server no encryption session initiation attempt
2500	deleted	POP3 SSLv3 invalid data version attempt
2501	deleted	POP3 SSLv3 invalid data version attempt
2502	deleted	POP3 SSLv3 invalid data version attempt
2505	deleted	WEB-MISC SSLv3 invalid data version attempt
2522	web-misc	WEB-MISC SSLv3 invalid Client_Hello attempt
2534	deleted	POP3 SSLv3 invalid Client_Hello attempt
2537	pop3	POP3 SSLv3 invalid Client_Hello attempt
2540	deleted	SMTP SSLv3 invalid Client_Hello attempt
2544	smpt	SMTP SSLv3 invalid Client_Hello attempt
100000158	ddos	COMMUNITY SIP INVITE message flooding
100000160	ddos	COMMUNITY SIP TCP/IP message flooding directed to SIP proxy

Tabella 7.7 Regole relative alla classe di attacco attempted-dos (Attempted Denial of Service)

Id	File	Descrizione
2183	smtp	SMTP Content-Transfer-Encoding overflow attempt
2515	web-misc	WEB-MISC PCT Client_Hello overflow attempt
2516	deleted	POP3 PCT Client_Hello overflow attempt
2517	imap	IMAP PCT Client_Hello overflow attempt
2518	pop3	POP3 PCT Client_Hello overflow attempt
2519	deleted	SMTP Client_Hello overflow attempt
2528	smtp	SMTP PCT Client_Hello overflow attempt
2657	web-misc	WEB-MISC SSLv2 Client_Hello with pad Challenge Length overflow attempt
2666	pop3	POP3 PASS format string attempt
100000135	imap	COMMUNITY IMAP GNU Mailutils request tag format string vulnerability
100000928	deleted	COMMUNITY EXPLOIT LANDesk Management Suite Alerting Service buffer overflow

Tabella 7.8 Regole relative alla classe di attacco attempted-admin (Attempted Administrator Privilege Gain)

Id	File	Descrizione
1365	deleted	WEB-ATTACKS rm command attempt
100000122	web-misc	COMMUNITY WEB-MISC mod_jrun overflow attempt
100000187	web-misc	COMMUNITY WEB-PHP XSS attempt
100000519	web-misc	COMMUNITY WEB-PHP VUBB index.php SQL injection attempt
100000745	web-php	COMMUNITY WEB-PHP Diesel Joke Site category.php SQL injection attempt

Tabella 7.9 Regole relative alla classe di attacco web-application-attack (Web Application Attack)

Id	File	Descrizione
2003	sql	SQL Worm propagation attempt
3070	imap	IMAP fetch overflow attempt Misc Attack
100000689	smptp	COMMUNITY SMTP Mytob MAIL FROM Attempt

Tabella 7.10 Regole relative alla classe di attacco misc-attack (Misc-Attack)

Id	File	Descrizione
485	deleted	ICMP Destination Unreachable Communication Administratively Prohibited
486	deleted	ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited
2436	file-identify	WEB-CLIENT Microsoft wmf metafile access

Tabella 7.11 Regole relative alla classe di attacco misc-activity (Misc-activity)

Id	File	Descrizione
254	dns	DNS SPOOF query response with TTL of 1 min. and no authority Potentially Bad Traffic
477	deleted	ICMP Source Quench Potentially Bad Traffic

Tabella 7.12 Regole relative alla classe di attacco bad-unknown (Potentially Bad Traffic)

Id	File	Descrizione
537	deleted	NETBIOS SMB IPC$ share access
1448	misc	Microsoft Windows Terminal server request attempt

Tabella 7.13 Regole relative alla classe di attacco protocol-command-decode (Generic Protocol Command Decode)

Id	File	Descrizione
2329	Sql	SQL probe response overflow attempt
3192	web-client	WEB-CLIENT Microsoft Windows Media Player directory traversal via Content-Disposition attempt

Tabella 7.14 Regole relative alla classe di attacco attempted-user (Attempted User Privilege Gain)

Id	File	Descrizione
2570	web-misc	WEB-MISC Invalid HTTP Version String

Tabella 7.15 Regole relative alla classe di attacco non-standard-protocol (Detection of a non-standard protocol or event)

Id	File	Descrizione
2182	deleted	BACKDOOR typot trojan traffic

Tabella 7.16 Regole relative alla classe di attacco trojan-activity (A Network Trojan was detected)

Id	File	Descrizione
1861	web-misc	WEB-MISC Linksys router default username and password login attempt

Tabella 7.17 Regole relative alla classe di attacco default-login-attempt (Attempt to login by a default username and password)

7.3.4 Risultati ottenuti dalla classificazione

I risultati finali ottenuti applicando le due metodologia di classificazione (associando ai diversi parametri i valori accuratamente testati nel paragrafo 7.3.1) vengono paragonati a quelli presenti nelle pubblicazioni scientifiche (indicate nel paragrafo 7.3.2) tramite questa tabella:

Metodologia	Totali	Int.	Rout	%Rout
Pubblicaz. scientif.	13.010.354	314.135	12.696.219	97,34%
"Itemset"	121.678	11.342	110.336	90,67%
"Routine"	121.678	8.725	112.953	92,82%

Tabella 7.18 Confronto tra i risultati della classificazione rilevati dalle pubblicaz.scientif. e quelli relativi alle due metodologie

Inoltre viene mostrato anche uno screenshot del contenuto dei file (la parte iniziale) indicante i risultati finali globali dell'applicazione delle due metodologie di classificazione ("nomeSettimana_Itemset_Class_Summary.txt" e "nomeSettimana_Cluster_Class_Summary.txt")

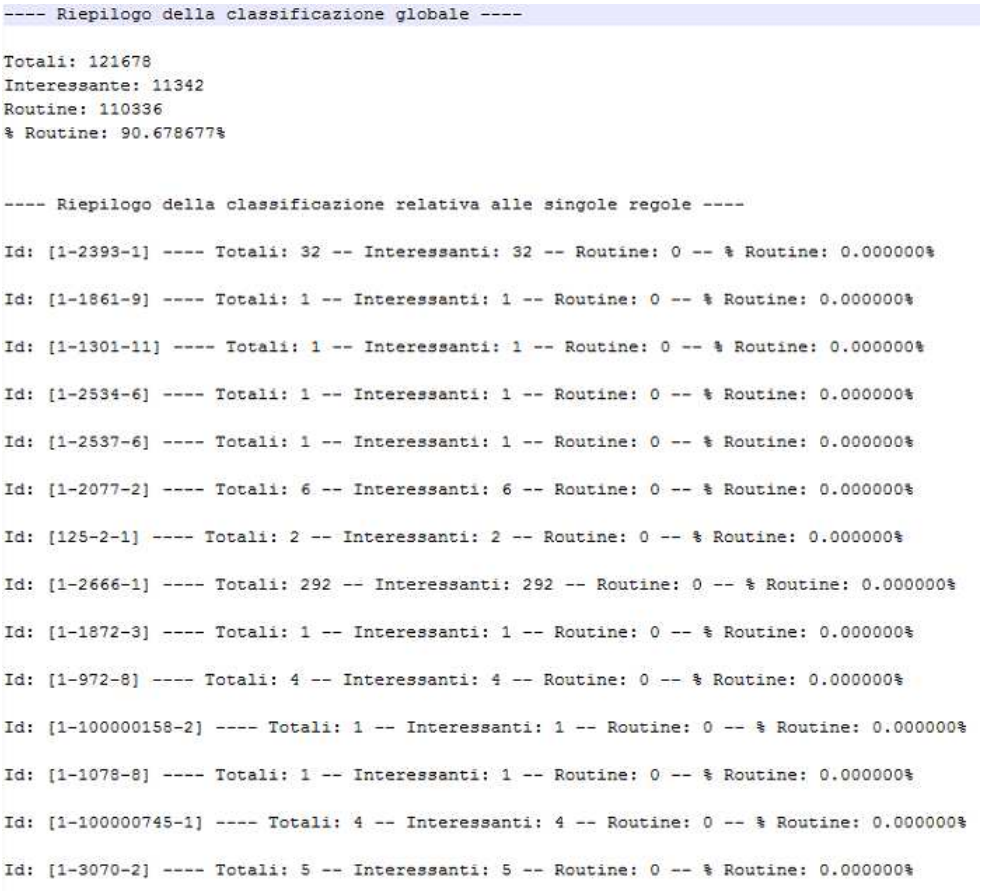

Figura 7.3 Screenshot dei risultati finali riassuntivi ottenuti tramite la metodologia"Itemset"

```
---- Riepilogo della classificazione globale ----

Totali: 121678
Interesting: 8725
Routine: 112953
% Routine: 92.829435%


---- Riepilogo della classificazione relativa alle singole regole ----

Id: [1-2393-1] ---- Totali: 32 -- Interesting: 32 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-1861-9] ---- Totali: 1 -- Interesting: 1 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-1301-11] ---- Totali: 1 -- Interesting: 1 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-2534-6] ---- Totali: 1 -- Interesting: 1 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-2537-6] ---- Totali: 1 -- Interesting: 1 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-2077-2] ---- Totali: 6 -- Interesting: 6 -- Routine: 0 -- % Routine: 0.000000%

Id: [125-2-1] ---- Totali: 2 -- Interesting: 2 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-2666-1] ---- Totali: 292 -- Interesting: 292 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-1872-3] ---- Totali: 1 -- Interesting: 1 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-972-8] ---- Totali: 4 -- Interesting: 4 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-100000158-2] ---- Totali: 1 -- Interesting: 1 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-1078-8] ---- Totali: 1 -- Interesting: 1 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-100000745-1] ---- Totali: 4 -- Interesting: 4 -- Routine: 0 -- % Routine: 0.000000%

Id: [1-3070-2] ---- Totali: 5 -- Interesting: 5 -- Routine: 0 -- % Routine: 0.000000%
```

Figura 7.4 Screenshot dei risultati finali riassuntivi ottenuti tramite la metodologia"Cluster"

Come si può osservare, nonostante le caratteristiche differenti dei log, le percentuali sono molto simili in entrambe le metodologie di classificazione e si avvicinano molto a quelli dei casi di studio presi in considerazione.

A questo punto, ciascuna regola può essere associata concettualmente all'insieme di allarmi che questa ha generato nel file di log di test. Quindi, per ogni regola viene riportato:

- il codice identificativo id
- il numero di allarmi generati (indicato con il termine "Totali")
- il numero di allarmi ciascuno etichettato come "Interessante" dopo la classificazione tramite prima la metodologia "Itemset" (indicato con il termine "Int. 1")
- il numero di allarmi ciascuno etichettato come "Routine" dopo la classificazione tramite la prima metodologia "Itemset" (indicato con il termine "Rout. 1")
- la percentuale di allarmi ciascuno etichettato come "Routine" dopo la classificazione tramite la prima metodologia "Itemset" (indicata con il termine "% Rout. 1")
- il numero di allarmi ciascuno etichettato come "Interessante" dopo la classificazione tramite la seconda metodologia "Cluster" (indicato con il termine "Int. 2")
- il numero di allarmi ciascuno etichettato come "Routine" dopo la classificazione tramite la seconda metodologia "Cluster" (indicato con il termine "Rout. 2")
- la percentuale di allarmi ciascuno etichettato come "Routine" dopo la classificazione tramite la seconda metodologia "Cluster" (indicata con il termine "% Rout. 2")

- l'indicazione (da parte degli esperti del gruppo VRT) se è noto che la regola genera falsi allarmi (indicata con il simbolo "VRT"); l'informazione non è presente viene utilizzato il termine "np"
- una percentuale indicativa (fornita dai componenti del gruppo di snortattack) di quanti falsi allarmi la regola in generale produce (indicata con il termine "% sa")

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
466	200	200	0	0%	200	0	0%	np	< 30%
469	4	4	0	0%	4	0	0%	np	< 30%
621	6	6	0	0%	6	0	0%	np	< 30%
624	52	52	0	0%	52	0	0%	np	< 30%
882	4.687	4.286	401	8,57%	0	4.687	100%	si	> 70%
1112	1	1	0	0%	1	0	0%	np	tra 30% e 70%
1113	34	34	0	0%	34	0	0%	no	< 30%
1201	374	0	374	100%	0	374	100%	no	tra 30% e 70%
1213	80	80	0	0%	80	0	0%	no	< 30%
1301	1	1	0	0%	1	0	0%	np	< 30%
1411	5.315	0	5315	100%	981	4.334	81,54%	np	tra 30% e 70%
1417	5.315	0	5315	100%	981	4.334	81,54%	np	tra 30% e 70%
1418	4	4	0	0%	4	0	0%	np	< 30%
1421	6	6	0	0%	6	0	0%	np	< 30%
1616	3	3	0	0%	3	0	0%	np	< 30%

Tabella 7.19 Classificazione relativa alla classe di attacco attempted-recon

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
937	4	4	0	0%	4	0	0%	np	< 30%
962	4	4	0	0%	0	6.962	100%	np	> 70%
972	4	4	0	0%	4	0	0%	no	< 30%
990	4	4	0	0%	4	0	0%	np	< 30%
1042	54	54	0	0%	54	0	0%	np	< 30%
1078	1	1	0	0%	1	0	0%	np	< 30%
1149	11	11	0	0%	11	0	0%	np	< 30%
1288	4	4	0	0%	4	0	0%	np	tra 30% e 70%
1560	42	0	42	100%	0	42	100%	np	tra 30% e 70%
1852	2.573	2.090	483	18,77%	0	2.573	100%	np	> 70%
1872	1	1	0	0%	1	0	0%	np	< 30%
2077	6	6	0	0%	6	0	0%	np	< 30%
2281	2.957	0	2.957	100%	0	2.957	100%	np	tra 30% e 70%
2393	32	32	0	0%	32	0	0%	np	< 30%

Tabella 7.20 Classificazione relativa alla classe di attacco web-application-activity

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
240	1	1	0	0%	1	0	0%	no	tra 30% e 70%
2418	5.422	0	5.422	100%	0	5422	100%	np	> 70 %
2500	12	12	0	0%	12	0	0%	no	< 30%
2501	258	258	0	0%	258	0	0%	np	< 30%
2502	12	12	0	0%	12	0	0%	np	< 30%
2505	21	21	0	0%	21	0	0%	no	< 30%
2522	9	9	0	0%	9	0	0%	np	< 30%
2534	1	1	0	0%	1	0	0%	np	tra 30% e 70%
2537	1	1	0	0%	1	0	0%	np	< 30%
2540	12	12	0	0%	12	0	0%	no	< 30%
2544	12	12	0	0%	12	0	0%	np	< 30%
100000158	1	1	0	0%	1	0	0%	np	< 30%
100000160	48.624	0	48.624	100%	29	48.595	99,94%	np	> 70%

Tabella 7.21 Classificazione relativa alla classe di attacco attempted-dos

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
2183	1	1	0	0%	1	0	0%	np	< 30%
2515	48	48	0	0%	48	0	0%	np	< 30%
2516	43	43	0	0%	43	0	0%	np	< 30%
2517	22	22	0	0%	22	0	0%	no	< 30%
2518	48	48	0	0%	48	0	0%	no	< 30%
2519	3	3	0	0%	3	0	0%	no	tra 30% e 70%
2528	79	79	0	0%	79	0	0%	np	< 30%
2657	7	7	0	0%	7	0	0%	np	< 30%
2666	292	292	0	0%	292	0	0%	np	< 30%
100000135	18.421	0	18.421	100%	0	18.421	100%	np	> 70%
100000928	3	3	0	0%	3	0	0%	np	< 30%

Tabella 7.22 Classificazione relativa alla classe di attacco attempted-admin

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
1365	9	9	0	0%	9	0	0%	np	< 30%
100000122	129	129	0	0%	129	0	0%	np	< 30%
100000187	1	1	0	0%	1	0	0%	np	tra 30% e 70%
100000519	2	2	0	0%	2	0	0%	np	< 30%
100000745	4	4	0	0%	4	0	0%	np	< 30%

Tabella 7.23 Classificazione relativa alla classe di attacco web-application-attack

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
2003	6.962	0	6.962	100%	0	6.962	100%	np	tra 30% e 70%
3070	5	5	0	0%	5	0	0%	np	< 30%
100000689	6	6	0	0%	6	0	0%	np	< 30%

Tabella 7.24 Classificazione relativa alla classe di attacco misc-attack

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
485	3.431	0	3.431	100%	2.785	646	18,82%	np	tra 30% e 70%
486	4.687	4.285	401	8,55%	0	1.027	100%	np	> 70%
2436	136	136	0	0%	136	0	0%	no	< 30%

Tabella 7.25 Classificazione relativa alla classe di attacco misc-activity

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
254	11	11	0	0%	12	0	0%	np	< 30%
477	7	7	0	0%	7	0	0%	np	< 30%

Tabella 7.26 Classificazione relativa alla classe di attacco bad-unlnown

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
537	19	19	0	0%	19	0	0%	no	< 30%
1448	5.326	0	5.326	100%	0	5.326	100%	no	tra 30% e 70%

Tabella 7.27 Classificazione relativa alla classe di attacco protocol-command-decode

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
2329	427	427	0	0%	427	0	0%	np	< 30%
3192	1	1	0	0%	1	0	0%	np	< 30%

Tabella 7.28 Classificazione relativa alla classe di attacco attempted-user

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
2570	1.699	923	776	45,67%	923	776	45,67%	np	tra 30% e 70%

Tabella 7.29 Classificazione relativa alla classe di attacco non-standard-protocol

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
2182	11	11	0	0%	11	0	0%	np	< 30%

Tabella 7.30 Classificazione relativa alla classe di attacco trojan-activity

Id	Totali	Int. 1	Rout. 1	% Rout. 1	Int. 2	Rout. 2	% Rout. 2	VRT	% sa
1861	1	1	0	0%	1	0	0%	np	< 30%

Tabella 7.31 Classificazione relativa alla classe di attacco default-login-attempt

Come si può osservare, i risultati ottenuti mostrano che le due metodologie operano nella quasi totalità dei casi in maniera opportuna, etichettando correttamente i vari allarmi; inoltre producono gli stessi simili risultati per entrambi le classificazioni, a dimostrazione che il principio di funzionamento e la sua messa in pratica è avvenuta con successo.

Osservando i dati si può evidenziare ad esempio l'enorme quantità di falsi allarmi (ed etichettati correttamente come "Routine") generati dalle regole con id 100000160 e 100000160. Controllando in Internet, si evince che la prima regola è famosa nel produrre molti allarmi che però sono di minima importanza in quanto si riferiscono ad un exploit (relativo all'applicativo GNUMail) "riparato" ormai da ben sette anni. Anche la seconda regola può venire benissimo ignorata, in quanto molti proxy SIP e dispositivi NAT possono inviare un grande volume di traffico assolutamente legittimo.

8 Conclusioni e considerazioni finali

La Tesi di Laurea ha evidenziato, in particolare, come i sistemi IDPS consistano in dispositivi molto potenti che svolgono funzioni che altri strumenti non riescono ad attuare. Si rivelano quindi degli elementi da tenere in forte considerazione quando si progetta la sicurezza informatica di un singolo sistema o di una rete. Tuttavia, il loro sofisticato funzionamento porta ad una loro conseguente difficile gestione, che li rende inevitabilmente una tecnologia da trattare con estrema cautela.

L'utilizzo degli applicativi implementati ha dimostrato che la gestione di tali strumenti da parte del responsabile della sicurezza (ma anche di un qualsiasi normale utente) può essere notevolmente facilitata.

Le due tecniche di data mining applicate ai log rappresentano già di per se un notevole supporto, perché sono in grado di evidenziare gli aspetti comuni che caratterizzano un'enorme quantità di allarmi in modo immediato e preciso.

Le due metodologie di classificazione apportano una sensibile riduzione degli allarmi, eliminando tutte quelle informazioni inutili che possono facilmente intensificare e fuorviare l'operato del responsabile.

Si è cercato soprattutto di concentrare gli sforzi su quest'ultimo aspetto, in quanto è un problema che spesso porta le aziende a non affidarsi a queste tecnologie (soprattutto nel caso, come visto, della categoria dei sistemi IPS). Sebbene sia un problema che risulta inevitabile per la natura propria di questi strumenti, gli applicativi hanno cercato di porre un rimedio, se non altro una "toppa" a questo aspetto negativo.

Il rilevamento dei falsi allarmi e di quelli irrilevanti si è rilevato più problematico del previsto (in particolare per il sistema NIDPS in utilizzo quale Snort) in quanto di difficile trattamento e valutazione; i fattori influenzanti sono svariati:

- la modalità di collocazione dello stesso sistema
- l'ambiente dove viene impiegato
- il tipo di traffico
- i servizi da tenere sotto controllo
- la sua configurazione e le regole che utilizza
- il grado di specificità voluto per il rilevamento

L'avere a disposizione solamente dei log, non facilita l'obiettivo finale, in quanto ogni log, ogni regola, persino ogni singolo allarme, devono essere interpretati in base a tutti gli elementi sopra citati.

I risultati ottenuti dagli applicativi non devono essere considerati come perfettamente esatti, ma, come dimostrato, si avvicinano molto alla situazione reale. In questo modo, il responsabile della sicurezza, effettuando delle opportune considerazioni (in base alle conoscenze dell'elemento monitorato ed alla propria esperienza) può decidere quali regole devono essere tenute in maggiore considerazione e quali invece possono venire benissimo disattivate nella configurazione del sistema IDPS. Ciò porta ad un miglioramento consistente e continuo nella messa a punto e configurazione del sistema stesso.

Un piccolo rammarico è dovuto al fatto che i log veramente fruibili per gli scopi della Tesi purtroppo si sono rivelati solamente quelli indicati nella sezione 7.1. Questo è stato causato molto probabilmente da una disattivazione inaspettata (per un motivo sconosciuto) del funzionamento di Snort sul server; a questo si è

aggiunto il mancato controllo, da parte dell'autore della Tesi, che la registrazione dei log continuasse con la regolarità (in quanto convinto che tutto funzionasse senza intoppi).

L'idea iniziale era quella di lavorare con una notevole quantità di allarmi, in modo da ottenere riscontri maggiori; tuttavia, i log in possesso fortunatamente si sono rilevati più che sufficienti in quanto contenenti una gran mole di dati utili al raggiungimento degli obiettivi iniziali preposti.

Sviluppi futuri

Gli applicativi implementati possono venire facilmente impiegati anche per log differenti da quelli prodotti dal software in analisi Snort. Applicazioni di sistema, dispositivi quali router, firewall, switch, stampanti, ecc. sono in grado di produrre i log dei loro eventi memorizzandoli sul sistema e sfruttando il protocollo syslog oppure tramite formati personalizzati. Gli applicativi possono essere applicati a tali log senza alcun problema, in quanto le proprietà dei dati contenuti nella quasi totalità di log sono le medesime.

Nella Tesi sono stati presi in considerazione solo alcuni attributi degli allarmi (id, indirizzo ip e porta sorgente, indirizzo ip e porta destinatario, protocollo). Nulla vieta di prendere in considerazione anche i rimanenti attributi che caratterizzano gli allarmi, in modo da ottenere dei risultati più specifici.

Gli applicativi possono venire sfruttati in modo automatico. Ad esempio, alla mezzanotte di ogni giorno può venire rigenerato il classificatore utilizzando come log per la fase di training quelli relativi ad un lasso di tempo pari alla due settimane precedenti. Come log per la fase di test viene di conseguenza impiegato il log relativo al giorno in esame.

Nelle tecniche di data mining può essere introdotto il concetto del tempo. Sebbene spesso i log non si prestino favorevolmente a tale soluzione (a causa della tipologia di dati in essi contenuti) possono essere apportate delle opportune modifiche agli applicativi, affinché riescano a rilevare associazioni temporali tra gli allarmi. Tale metodo si rivela abbastanza complicato, ma il risultato finale può essere utile per identificare concatenazione di allarmi che nascondono minacce più complesse.

9 Bibliografia

[1] B. Caswell, J. Beale e A. Baker, Snort IDS and IPS Toolkit, Syngress, 2007.

[2] C. Gerg e K. J. Cox, Managing Security with Snort and IDS Tools, O'Reilly Media, 2004.

[3] J. Koziol, Intrusion Detection with Snort, Sams Publishing, 2003.

[4] M. Rash, A. D. Orebaugh, G. Clark, B. Pinkard e J. Babbin, Intrusion Prevention and Active Response: Deploying Network and Host IPS, Syngress, 2005.

[5] R. Vaarandi, «Mining event logs with SLCT and LogHound,» in *Proceedings of the 2008 IEEE/IFIP Network Operations and Management Symposium*, 2008, pp. 1071-1074.

[6] R. Vaarandi, «A Breadth-First Algorithm for Mining Frequent Patterns from Event Logs,» in *Proceedings of INTELLCOMM*, 2004, pp. 293-308.

[7] R. Vaarandi, «A Data Clustering Algorithm for Mining Patterns From Event Logs,» in *IP Operations and Management*, 2003, pp. 119-126.

[8] R. Vaarandi, «Real-time Classification of IDS Alerts with Data Mining Techniques,» in *Proceedings of the 2009 IEEE MILCOM Conference*, 2009, pp. 1-7.

[9] R. Vaarandi e K. Podiņš, «Network IDS alert classification with frequent itemset mining and data clustering,» in *Proceeding of Network and Service Management (CNSM)*, 2010, pp. 451-456.

[10] «Sourcefire Vulnerability Research Team (VRT),» [Online]. Available: http://www.snort.org/vrt. [Consultato il giorno 29 09 2012].

[11] «Snortattack - Comunicazione, informazione, conoscenza,» 19 11 2005. [Online]. [Consultato il giorno 20 09 2012].

[12] T. Raven, Snort e Debian - Ecco come ti tiro su un ids open source e ti rilevo gli attacchi alla tua rete, 2008.