**University of Verona**

DEPARTMENT OF COMPUTER SCIENCE

**Master Degree in Computer Science and Engineering**

*Specialization in*
Security and Software Engineering

# MODELING CYBER-THREATS

*ADOPTING BAYES' PRINCIPLES IN THE ATTACK GRAPHS THEORY*

**AUTHOR**

MATTIA ZAGO
VR 367531

**SUPERVISOR**

PROF. ISABELLA MASTROENI

**EXAMINER**

PROF. ALESSANDRO FARINELLI

**CO-SUPERVISOR**

FULL PROF. GREGORIO MARTINEZ
*University of Murcia*

**Academic Year**
2014/2015

**Keywords**
Bayesian Networks, Attack Graphs, Multigraph Simulator

# Abstract

This master thesis will analyze how the Bayesian Theory can be applied to the Intrusion Prevention and Response Strategy research area. I am going to present a brief summary on the graphical security modelling technique, with the objective of describing a common point between the existing formalism and aiming to implement a Security Model Simulator that allows the expert to both run and compare different solutions and approaches to the same problem (or architecture).

The focus of this work is on the simulator, presented in chapter 3, in particular on the technical details of the implementation and on the innate difficulties related to the lack of standard basic structure.

# Abstract

Lo scopo di questa tesi di laurea magistrale è l'analisi e l'applicazione della teoria Bayesiana nell'ambito della sicurezza delle reti. Nel documento verrà mostrato come il rischio legato alle minacce informatiche possa essere studiato ed approfondito attraverso strumenti grafici quali i grafi d'attacco o le reti Bayesiane.

In un primo momento verrà approfondito lo stato dell'arte con l'obiettivo di descrivere uno standard comune rispetto ai molteplici formalismi esistenti in letteratura, mentre nella seconda parte verrà presentato un software creato *ad-hoc* per simulare e comparare differenti modelli.

Il fulcro del documento è il simulatore, descritto nel capitolo 3, la sua implementazione e le difficoltà intrinseche dovute alla mancanza di uno standard condiviso.

# Indice dei Contenuti

# Appendici



# Bibliografia

# Capitolo 1

## 1 INTRODUZIONE

L'analisi e la protezione di un sistema o di una rete può essere ridotto a due macro problemi: il primo è quello relativo alla gestione dell'incertezza (ovvero la mancanza di dati certi e/o completi), mentre il secondo riguarda la scelta delle contromisure da applicare (ovvero le misure di sicurezza atte a diminuire il rischio complessivo del sistema). Il problema dell'incertezza si delinea attraverso supposizioni ed informazioni ipotetiche, vaghe e spesso parziali impedendo così una gestione precisa ed accurata del rischio. La questione in sé non risiede nel come vengono raccolti i dati, bensì nella natura intrinseca della sicurezza informatica: non è possibile conoscere a priori chi sta attaccando, per quali motivi, con quali obiettivi o con quali abilità tecniche. È inoltre difficile stabilire con precisione se un sistema non sia ancora stato violato o se sia stato attaccato in passato.

L'incertezza può essere radicata a livello architetturale (durante la progettazione), a livello di analisi in tempo reale (situazione di normalità) oppure a livello di analisi forense (post-attacco). A livello architetturale essa è rappresentata dalla mancanza di conoscenza sulle le vulnerabilità del sistema (si pensi ad esempio ad un bug nel software) e del loro possibile sfruttamento (esiste un exploit, è possibile utilizzarlo?). Tale incertezza è *statica* poiché riflette le caratteristiche intrinseche del sistema che si sta progettando. In contrapposizione esiste l'incertezza *dinamica* che riflette la natura temporale ed ambientale della sicurezza informatica, ovvero il dubbio e l'impossibilità di determinare in tempo reale lo stato del sistema (esiste un malintenzionato interessato a violare il sistema? Quali scelte ha compiuto? Quali sono le sue abilità tecniche? Quante barriere ha già superato?). L'analisi forense post-attacco rientra nella categoria dell'incertezza dinamica, cercando di fornire risposte a domande relative alla storia dell'attacco subito attraverso l'analisi dei dati estratti dagli *IDS*[1].

Storicamente il problema fu trattato ampiamente con l'ausilio di strumenti logici, dove le relazioni causa-effetto vennero analizzate con l'assunto che se una pre-condizione si fosse rivelata vera, allora la post-condizione sarebbe risultata vera anch'essa. Questo approccio fu applicato per anni, sia nell'analisi delle vulnerabilità [1, 2, 3, 4] sia nella correlazione delle informazioni [1, 5, 6, 7, 8, 9], tuttavia, se pur importante, non fu in grado di rappresentare l'ignoto che collega le osservazioni effettuate con le cause effettive.

[1] Gli *Intrusion Detection Systems* sono dispositivi software o hardware che identificano operazioni non autorizzate sui sistemi informatici analizzando i log generati dal sistema e dalle sue componenti.

L'esempio più lampante riguarda le vulnerabilità "*Zero-day*"[2], ovvero quelle che non possono essere previste ma che devono essere considerate nello stabilire il livello di rischio di un sistema.

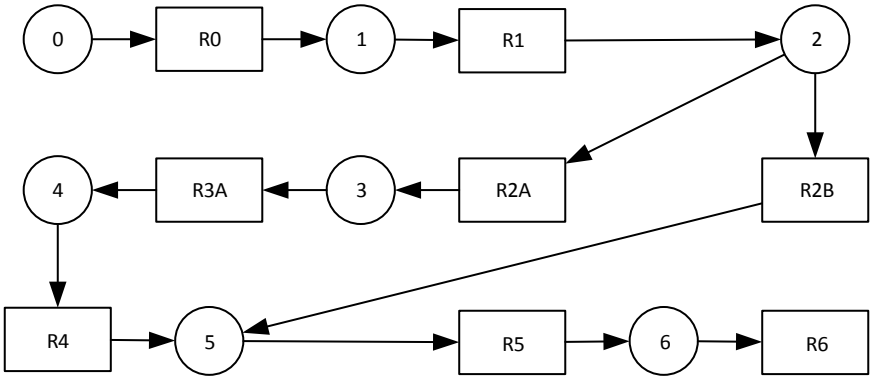



*Figura 1: Esempio di modello logico*

L'alternativa ai formalismi logici è costituita dai modelli statistici [10, 11], ovvero quei modelli che cercano di derivare un pattern di attacco comune per ogni sistema. Molti eventi relativi alla sicurezza informatica hanno una natura statistica (ad esempio il traffico di rete, dove un picco può significare sia interesse virale sia un *DOS*[3] in corso) e lo studio di queste relazioni può evidenziare variazioni nello stato "normale" di un sistema, anche se ogni attaccante rappresenta un individuo unico il cui comportamento segue strategie personali e difficilmente prevedibili.

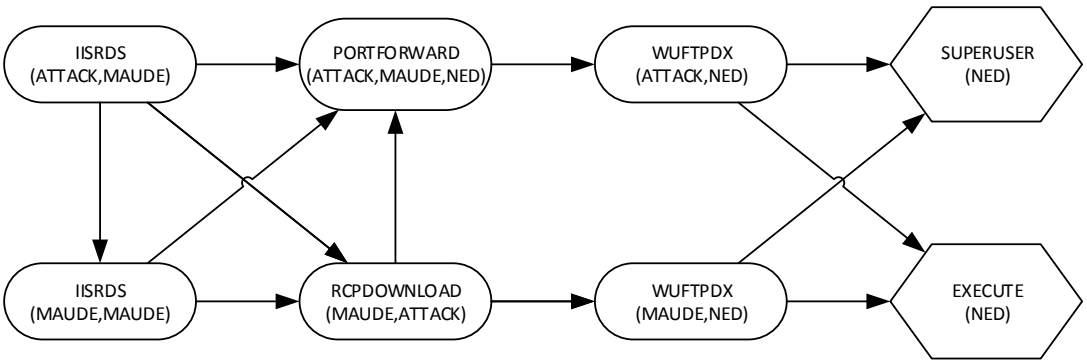



*Figura 2: Esempio di grafo d'attacco*

L'evoluzione naturale di questi due approcci, logico e statistico, risulta quindi essere una combinazione di entrambi, si ha quindi un modello che comprende dipendenze logiche e deterministiche affiancate da strumenti probabilistici in grado di gestire le informazioni parziali

[2] Una vulnerabilità *Zero-Day* è una falla sconosciuta in sistema (sia software che hardware) che viene sfruttata da una cerchia ristretta di malintenzionati. Una vulnerabilità di questo tipo è resa nota prima della disponibilità di una patch correttiva.

[3] Un *DOS* (*Denial Of Service*) indica un malfunzionamento di un sistema dovuto all'esaurimento delle risorse causato da un attacco deliberato oppure da un'azione accidentale.

o ignote. I primi modelli guidati da questo ideale sono rappresentati dai grafi d'attacco [12, 13, 14, 15, 16], ovvero strutture grafiche che combinano la semantica formale degli algoritmi sui grafi con tecniche intuitive ed *user-friendly* proprie dei modelli visivi. Ogni grafo d'attacco in letteratura ha in comune con gli altri la struttura e le funzionalità di base, pur mantenendo sostanziali (e spesso inconciliabili) differenze nelle funzionalità avanzate e nel metodo di approccio al problema studiato: ad esempio Amman *et al.* [12] utilizzarono gli exploit come base, mentre Jajodia *et al.* [14] si avvalsero di una combinazione di stati e vulnerabilità.

L'approccio alla sicurezza informatica richiede dinamicità e strategia poiché occorre preventivare i possibili attacchi e reagire anticipandone le mosse. Il modello di riferimento è lo SCADA[4], ovvero il ciclo di vita di un sistema; tuttavia questo documento non tratterà ogni fase del modello bensì si concentrerà sull'analisi preventiva, cioè lo stabilire il rischio del sistema e le possibili contromisure applicabili. Questa tesi, nello specifico, analizzerà uno degli approcci più in voga al momento, le *Reti Bayesiane* applicate alla sicurezza informatica, evoluzione spontanea dei grafi d'attacco.

Principalmente il modello presenta tre vantaggi:

1. Permette di evidenziare le relazioni causali tra variabili di un dato dominio, in altre parole ogni conseguente attacco atomico sarà condizionalmente dipendente dal successo di ogni suo predecessore.
2. Consente di rappresentare in modo compatto ed esaustivo le informazioni circa i percorsi di raggiungibilità, risultando così un modello flessibile e scalabile.
3. Fornisce un formalismo pensato per trattare informazioni parziali ed incerte; in questo modo può incorporare ipotesi fornite dall'amministratore ed al tempo stesso formulare conclusioni coerenti ed aggiornate.

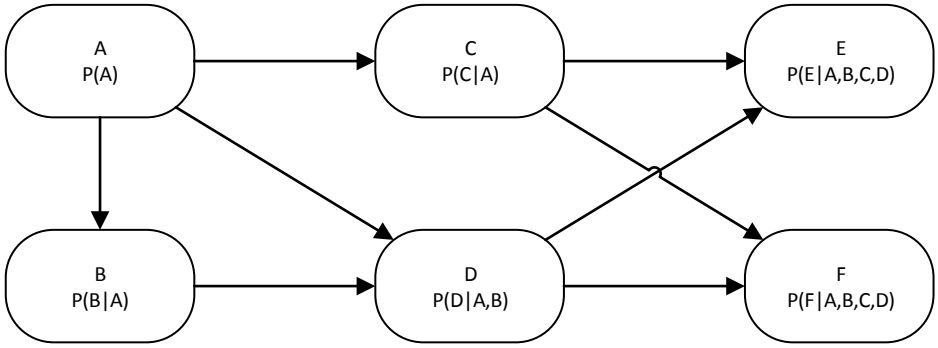



*Figura 3: Esempio di rete Bayesiana*

Nello specifico, dopo una breve introduzione teorica sullo stato dell'arte verrà presentato un prototipo funzionante e realizzato in gran parte dall'autore di questo documento. Il simulatore "*Multigraph*", presentato [17] durante la prima conferenza JNIC[5] a Leon in Spagna, ha l'obiettivo di bypassare i problemi legati alla mancanza di uno standard condiviso e di consentire ai ricercatori di eseguire e comparare diversi modelli matematici basati su grafi d'attacco e reti Bayesiane.

[4] Lo *SCADA* è un modello per la sicurezza che definisce le fasi di maturità di un sistema. Ulteriori informazioni sono disponibili sul sito di Red Tiger Security [72].

[5] Jornadas Nacionales de investigación en Ciberseguridad, Leon (Spagna) 14-16 settembre 2015. http://jornadasciberseguridad.riasc.unileon.es/

# Capitolo 2

## 2 RASSEGNA CRITICA

A differenza dei primi modelli di grafi d'attacco, i modelli probabilistici sono in grado di combinare efficacemente l'analisi delle vulnerabilità con il rischio di sfruttamento delle stesse. L'interesse verso questi modelli iniziò a diffondersi nei primi anni del XXI secolo, quando autori come Jajodia, Noel, [3, 8], Dantu [18] e Qin [19] decorarono i vertici dei grafi d'attacco con distribuzioni di probabilità.

Le reti Bayesiane furono applicate alla sicurezza informatica a partire dal 2005, sembra in maniera indipendente, da Dantu e Kolan [20] (attraverso l'utilizzo delle probabilità congiunta ed a posteriori), Liu e Man [21] (attraverso l'inferenza statistica) ed An *et al.* [22] (utilizzando la probabilità congiunta come strumento di analisi dei log).

Dantu [18] e Liu [21] realizzarono un modello in cui ad ogni nodo è associata una probabilità che ne descrive la Likelihood (cfr. app. 2.3.5) d'attacco che, combinata sotto forma di probabilità congiunta con le perdite attese, fornisce una misura del rischio complessivo del sistema. Solo nel 2009 [23] verranno risolti i problemi legati all'intervento di personale esperto nell'assegnazione delle probabilità a priori e condizionali.

Nel 2006, An, Jutla e Cercone compresero che la dinamicità temporale di un attacco è un fattore determinante nella gestione del rischio, tuttavia non furono in grado di modellare una struttura completa e stabile. La modellazione temporale verrà risolta solamente quattro anni dopo dal gruppo di Jajodia [24] attraverso le "Reti Bayesiane Dinamiche" in grado di modellare la sequenzialità degli attacchi utilizzando le metriche temporali del CVSS (cfr. app. 1).

Nel 2008 Frigault e Wang [25] utilizzarono le probabilità condizionali per modellare le interazioni tra le diverse vulnerabilità, mentre Althebyan e Panda [26] utilizzarono la probabilità congiunta per combinare i "*Dependencies Graph*" [6] con i "*Knowledge Graph*" [7], realizzando una rete Bayesiana in grado di modellare il rischio di un *insider*[8].

[6] Un "*Dependencies Graph*" (*grafo delle dipendenze*) è un grafo gerarchico che mostra tutte le dipendenze tra le risorse di un sistema. [80]

[7] Un "*Knowledge Graph*" (*grafo della conoscenza*) è un grafo che rappresenta diversi pezzi di informazione disponibili ad un insider. [80]

[8] Un *insider* è un individuo che ha accesso ad informazioni riservate interne all'azienda. L'esempio classico di insider è quello di un dipendente malcontento che sottrae e rivende o pubblica online i dati privati della compagnia.

Nel 2009 il gruppo di Jajodia ed il gruppo di Sommestad svilupparono due modelli indipendenti per l'analisi e l'ottimizzazione delle contromisure (cfr. 2.1.8 e 2.1.9), mentre Poolsappasit *et al.* [23] svilupparono il modello chiamato BAG, o "Grafo Bayesiano d'Attacco", utilizzato ancora oggi come strumento di analisi statica delle reti.

A partire dal 2010 l'interesse verso le reti Bayesiane diminuì, lasciando irrisolti problemi come l'ottimizzazione multi-obiettivo delle contromisure (affrontato senza successo da Noel *et al.* nel 2003 [3] e formalizzato nel 2007 da Dewri *et al.* [27] con il nome di "Dilemma dell'Amministratore") o la crescita esponenziale legata alla dimensione del sistema da analizzare (introdotto da Ou *et al.* [15] e parzialmente risolto attraverso il vincolo di monotonicità presentato da Amman *et al.* [12]).

## 2.1 ARTICOLI PRINCIPALI

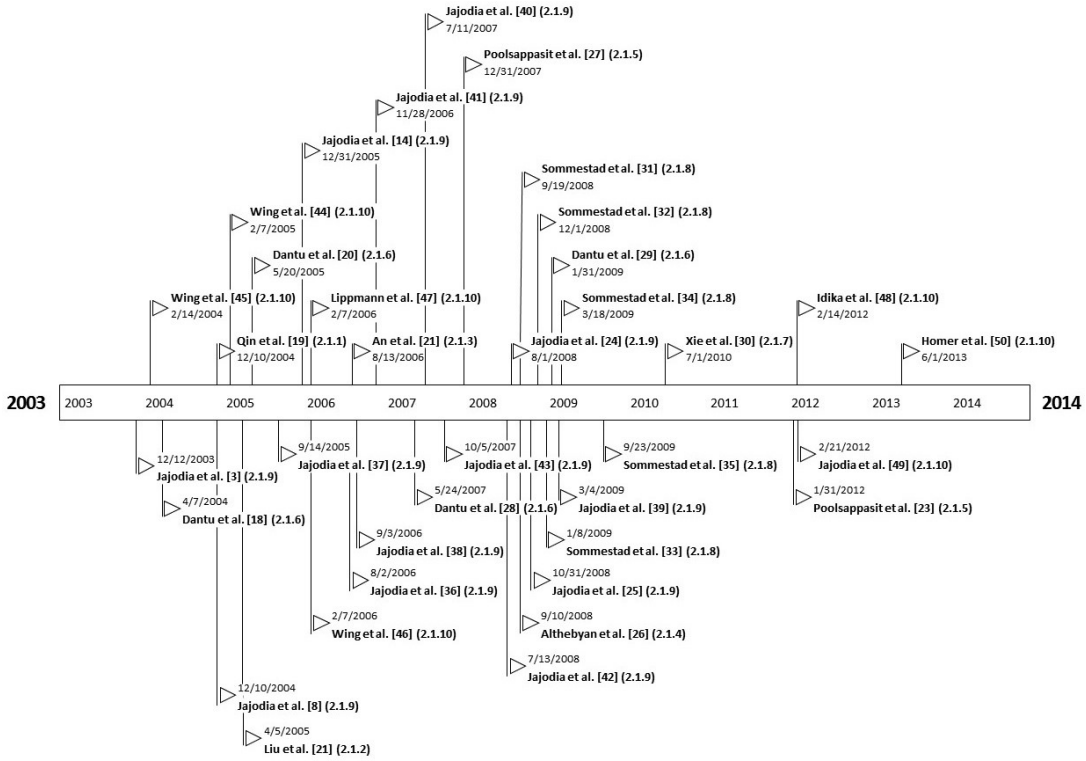



*Figura 4: Timeline dei principali articoli*

### 2.1.1 Attack plan recognition and prediction using causal networks (2004) [19]

Qin e Lee proposero per primi di integrare i grafi d'attacco con l'inferenza probabilistica al fine di "*valutare la probabilità di successo di un attacco e predire un potenziale attacco imminente*". Il loro lavoro si concentrò sulla correlazione tra diversi scenari d'attacco e, utilizzando le informazioni generate dagli IDS, riuscirono a stabilire la consequenzialità di attacchi apparentemente scollegati (in termini di tempo, risorse sfruttate o sistemi colpiti).

Il loro algoritmo fornì uno strumento utile per evidenziare le relazioni indirette (ovvero due attacchi che condividono un obiettivo ma seguono diversi percorsi) attraverso l'applicazione dell'inferenza probabilistica a due fasi: bottom-up (probabilità a posteriori dei padri rispetto ai figli) e top-down (probabilità a posteriori dei figli rispetto ai padri).

### 2.1.2 Network vulnerability assessment using Bayesian networks (2005) [21]

Liu e Man realizzarono un'implementazione funzionante in C++ di una rete Bayesiana basata su matrici di adiacenza. L'analisi quantitativa integrava nove gruppi di vulnerabilità (facenti capo allo standard CVE[9]) con le tecniche di inferenza Bayesiana (ottimizzata grazie all'ausilio dell'algoritmo di "Bucket Elimination"[10]).

[9] Il CVE, "*Common Vulnerabilities and Exposures*", è un database che indicizza le informazioni relative alle vulnerabilità rese pubbliche. [65]

[10] Cfr. app. 2.3.6

Nel loro modello ogni nodo rappresenta una singola violazione, ogni arco rappresenta una o più vulnerabilità sfruttabili ed ogni percorso risulta essere una serie di exploit tali da indicare un potenziale attacco. Ogni vulnerabilità è atomica, ed è costruita sulla base di un template composto da pre-condizioni, post-condizioni ed indicatori di rischio.

### 2.1.3 Privacy intrusion detection using dynamic Bayesian networks. (2006) [22]

An *et al.* evidenziarono che il flusso delle informazioni all'interno di una rete aziendale non può essere modellato da una rete Bayesiana semplice (come un HMM[11]) a causa della sua incapacità di gestire le relazioni temporali tra i domini delle vulnerabilità.

Analizzando gli aspetti legati alla privacy, implementarono un modello basato sulle reti Bayesiane dinamiche in grado di determinare la natura delle attività presenti o passate di un operatore (potrebbero indicare, ad esempio, un interesse verso informazioni riservate).

### 2.1.4 A knowledge-based Bayesian model for analyzing a system after an insider attack (2008) [26]

Althebyan e Panda in questo lavoro del 2008 analizzarono il problema legato agli insider attraverso il modello da loro definito "*Knowledge Bayesian Attack Graph*" (KBAG).

Un KBAG, come ogni altro grafo d'attacco, evidenzia i possibili percorsi che potrebbero consentire ad un malintenzionato di raggiungere l'obiettivo. La differenza sostanziale risiede nel fatto che questo modello rappresenta ogni possibile dato come un oggetto (ad esempio una password, un documento cartaceo, ecc.) rappresentabile nel grafo topologico delle informazioni aziendali. Ogni nodo raggiungibile, se sfruttato, andrà quindi a sommarsi alla conoscenza complessiva dell'insider, permettendogli quindi di sbloccare nuove risorse o accedere a sistemi prima a lui preclusi. Il modello parte dall'assunto che un attaccante non cercherà di raggiungere l'obiettivo per la via più breve, bensì vorrà ottenere quanta più conoscenza possibile dell'intero sistema prima di arrivare al traguardo.

Dal punto di vista probabilistico, ogni nodo del KBAG è associato con le funzioni di probabilità definite per le reti Bayesiane, rendendo di fatto il modello una rete Bayesiana decorata con tutte le proprietà sopradescritte. Le informazioni disponibili all'insider vengono quindi etichettate come evidenze ed utilizzate come base per l'inferenza statistica.

### 2.1.5 Articoli del gruppo di Poolsappasit dal 2007 al 2009 [23, 27]

Poolsappasit, Dewri e Ray realizzarono un modello basato sulle reti Bayesiane in grado di quantificare le risorse ed i cambiamenti di stato del sistema attraverso l'analisi dinamica e l'ottimizzazione delle contromisure applicabili. Si deve a loro la formulazione del problema chiamato "*Dilemma dell'Amministratore*" [27], ovvero la problematica del selezionare un sottoinsieme di contromisure da implementare minimizzando sia il costo complessivo (che deve rispettare un certo budget) sia il rischio associato alle vulnerabilità residue.

Il loro principale risultato fu la creazione di un modello denominato "Bayesian Attack Graph" (BAG), una rete Bayesiana in grado di minimizzare la quantità di informazione inserita come input. In un BAG i nodi, rappresentati come variabili Bayesiane, codificano gli stati dei sistemi facenti parte della rete, mentre gli archi rappresentano uno o più exploit necessari al raggiungimento di uno stato successivo. Ogni arco è associato ad un vettore CVSS, rendendo di fatto il collegamento tra due variabili una relazione condizionale la cui probabilità è oggettiva e definita da uno standard internazionale.

[11] Gli "Hidden Markov Model" sono modelli statistici che descrivono la probabilità condizionale di un residuo data una serie di residui precedenti.

A differenza delle reti Bayesiane classiche e dei modelli contemporanei basati su tale approccio, un BAG non necessità dell'intervento di un esperto esterno per la definizione delle probabilità condizionali proprie di ogni relazione. Chiaramente le probabilità a priori associate ai nodi esterni e la topologia stessa della rete devono essere forniti come input. Nello specifico, le CPT[12] vengono calcolate sulla base della tipologia di relazione padre/figlio: dato un nodo $(n)$, i suoi padri $(\mathrm{pa}(n))$ possono essere connessi al nodo stesso in forma disgiuntiva (OR) oppure in forma congiuntiva (AND). Una connessione disgiuntiva implica che per violare il nodo $n$ è necessario violare almeno uno dei nodi padre, viceversa una connessione congiuntiva richiede di violare tutti i nodi padre di $n$ prima di poter violare il nodo stesso.

Le probabilità condizionali sono pertanto ricavate secondo le formule sotto elencate.

Congiunzione:

$$P(n|\,\mathrm{pa}(n)) = \begin{cases} 0 & \exists p \in \mathrm{pa}(n)\,|P(p) = 0 \\ \prod_{\forall p \exists e | e=(p,n)} P(e) & altrimenti \end{cases}$$

Disgiunzione:

$$P(n|\,\mathrm{pa}(n)) = \begin{cases} 0 & \forall p \in \mathrm{pa}(n)\,|P(p) = 0 \\ 1 - \left( \prod_{\forall p \exists e | e=(p,n)} [1 - P(e)] \right) & altrimenti \end{cases}$$

La probabilità associata ad ogni arco $(e)$ non è calcolata secondo le equazioni indicate in appendice (cfr. app. 1.1), bensì secondo la formula:

$$P(e) = 2 \cdot AV \cdot AC \cdot AU$$

Dove *AV*, *AC*, *AU*, sono gli indici CVSS "Access Vector", "Access Complexity" ed "Authentication".

Il modello consente inoltre di calcolare uno degli indici più utilizzati nella gestione del rischio, il ROI (*Return on Investment*), moltiplicando le probabilità associate ad ogni variabile con le perdite previste in caso di sfruttamento delle stesse. La mitigazione del rischio avviene attivando una serie di contromisure non categorizzate a priori ed ottimizzando la funzione obiettivo definita come:

$$obj = \alpha\,\mathrm{LG}(\mathbb{C}) - \beta\,\mathrm{SCC}(\mathbb{C})$$

Con:

- $\alpha$ e $\beta$ parametri fissati che codificano le preferenze dell'utente, sono tali per cui $\alpha + \beta = 1$ e $0 \leq \alpha, \beta \leq 1$;
- $\mathbb{C}$ insieme delle contromisure attivate nella rete;
- $\mathrm{LG}(\mathbb{C})$ definito come la somma delle perdite (o guadagni) delle singole variabili, nello specifico se $\mathrm{LG}(\mathbb{C}) \geq 0$ si tratterà di un guadagno complessivo, altrimenti di una perdita;
- $\mathrm{SCC}(\mathbb{C})$ definito come la somma dei costi delle contromisure.

L'ottimizzazione di questa funzione è stata eseguita seguendo un classico approccio genetico.

[12] "Conditional Probability Table" cfr. app. 2.3.2.3

Il modello presentato in [23] è stato scelto come primo modello da implementare nel simulatore descritto nel capitolo 3.

### 2.1.6 Articoli di Dantu *et al.* dal 2004 al 2009 [18, 20, 28, 29]

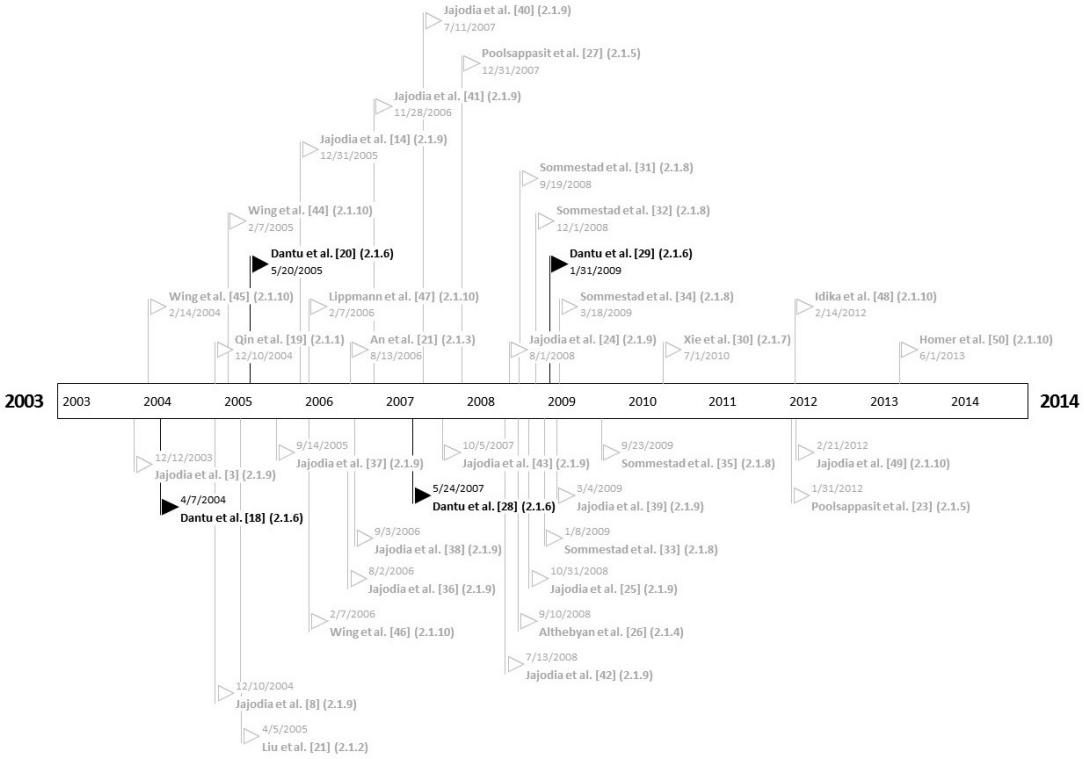



*Figura 5: Timeline articoli di Dantu et al.*

In una serie composta di quattro articoli, Dantu *et al.* proposero di integrare i grafi d'attacco con le caratteristiche espresse dal comportamento dell'attaccante (ad esempio livello tecnico, tenacità, capacità finanziare, ecc.) per delineare al meglio gli obbiettivi di valore, le sequenze di attacchi ed il profilo stesso del malintenzionato. Definirono un insieme di metriche che associarono ai nodi ed agli archi di una rete Bayesiana, per poi integrarli attraverso la probabilità *congiunta* (cfr. app. 2.3.4) e quella a *posteriori* (cfr. app. 2.3.6) [20]. L'inferenza statistica fu quindi applicata alla rete sia per predire il passo successivo che per stabilire la causa di un attacco date le prove di un attacco avvenuto.

Gli autori classificarono l'attaccante secondo tre classi distinte: opportunista, hacker ed esploratore [28]. Ogni classe è contraddistinta da un livello di risorse sfruttate per l'attacco, sia in termini di costo finanziario e tempo che in termini di motivazione e capacità tecniche (ad esempio uno "*script kiddie*" avrà a disposizione molte meno risorse rispetto ad un attaccante motivato e pagato da una società rivale).

L'obiettivo finale fu quello di evidenziare il percorso ottimale per l'attaccante, e di stabilire di conseguenza le misure di protezione necessarie [29].

### 2.1.7 Using Bayesian networks for cyber security analysis (2010) [30]

Xie *et al.* studiarono i sistemi di protezione in real-time, evidenziando come la mancanza di una comune e condivisa metodologia impedisca di far interagire tra di loro diversi modelli. Gli autori fecero notare come le probabilità condizionali non siano in grado di separare gli attacchi sulla base del tipo di incertezza, implementando un modello parziale in grado di distinguere tra incertezza relativa ai log IDS, al successo o insuccesso di un attacco ed alle scelte ignote dell'attaccante.

### 2.1.8 Articoli di Sommestad *et al.* dal 2008 al 2009 [31, 32, 33, 34, 35]

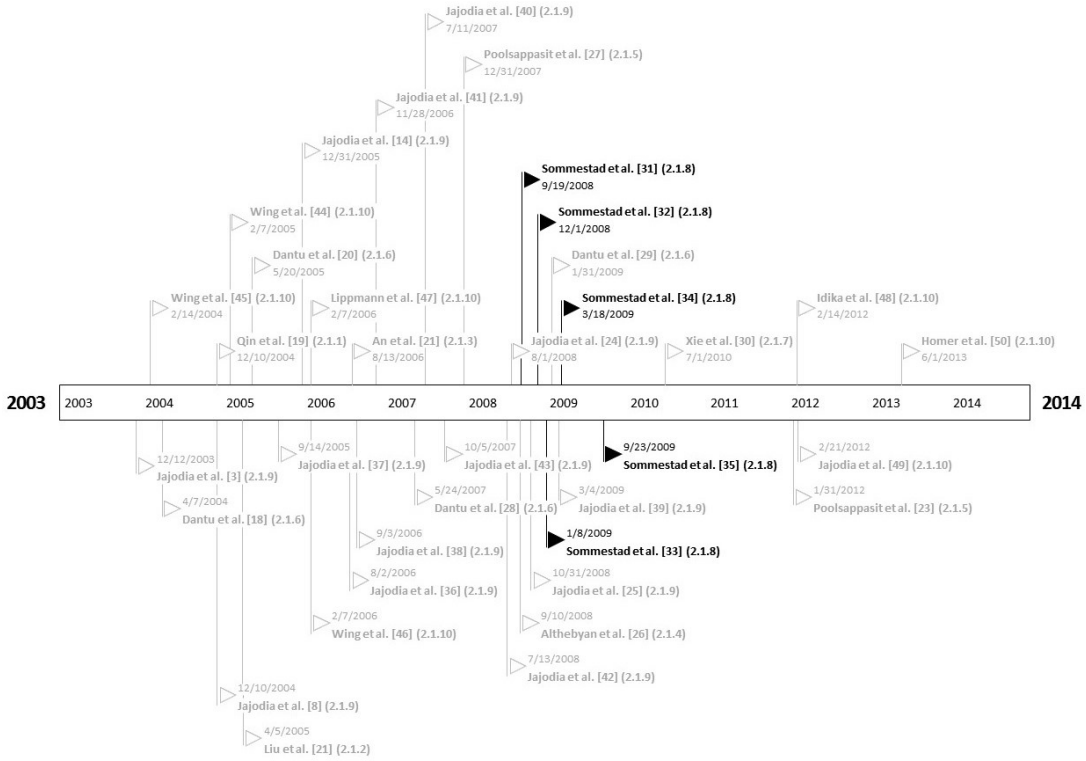



*Figura 6: Timeline articoli di Sommestad et al.*

Finanziati dal Royal Institute di Stoccolma, nel corso del 2008, Sommestad *et al.* realizzarono un framework basato sulle reti Bayesiane in grado di gestire i meccanismi di difesa direttamente durante la fase di analisi strutturale (nello specifico l'idea fu quella di evidenziare effettivamente i sistemi esistenti e non solamente quelli utilizzabili [33]).

Il loro modello, chiamato "Extended Influence Diagram" [31] è un'estensione delle reti Bayesiane in cui ad ogni nodo ed ad ogni arco è associato una tipologia: gli archi, ad esempio, possono essere distinti in *archi di definizione* (aggiunti dall'amministratore, definiscono la gerarchia padre-figlio del grafo) ed *archi causali* (aggiunti dalle interazioni derivate, definiscono le connessioni reali tra i sistemi). Allo stesso modo i nodi sono classificati in *nodi di decisione* (che rappresentano lo stato del sistema) e *nodi di servizio* (che rappresentano nodi associati alle diverse combinazioni di stati [31]). Le contromisure, invece, rappresentano un sottoinsieme delle

82 metriche[13] utilizzate per valutare il rendimento ed il rischio complessivo della rete. Questi indicatori sono sia generici che specifici e, ad esempio, possono essere utilizzati sia come metriche a posteriori del rateo di successo di un attacco, sia come descrittori dello stato delle contromisure [32]. La versione finale del loro modello è stata pubblicata nei primi mesi del 2009 [33] con il nome di "*Bayesian Defense Graph*".

In sintesi, un Grafo Bayesiano di Difesa è una rete Bayesiana che affianca le funzionalità di azione-reazione al grafo d'attacco consentendo all'amministratore di gestire le relazioni causali tra variabili e parametri ignoti. Il modello presentato in [33] è stato scelto come possibile candidato per il simulatore descritto nel capitolo 3.

### 2.1.9 Articoli del gruppo Jajodia, Wang, Noel, Singhal, Frigault, O' Berry dal 2003 al 2010 [3, 8, 14, 25, 24, 36, 37, 38, 39, 40, 41]

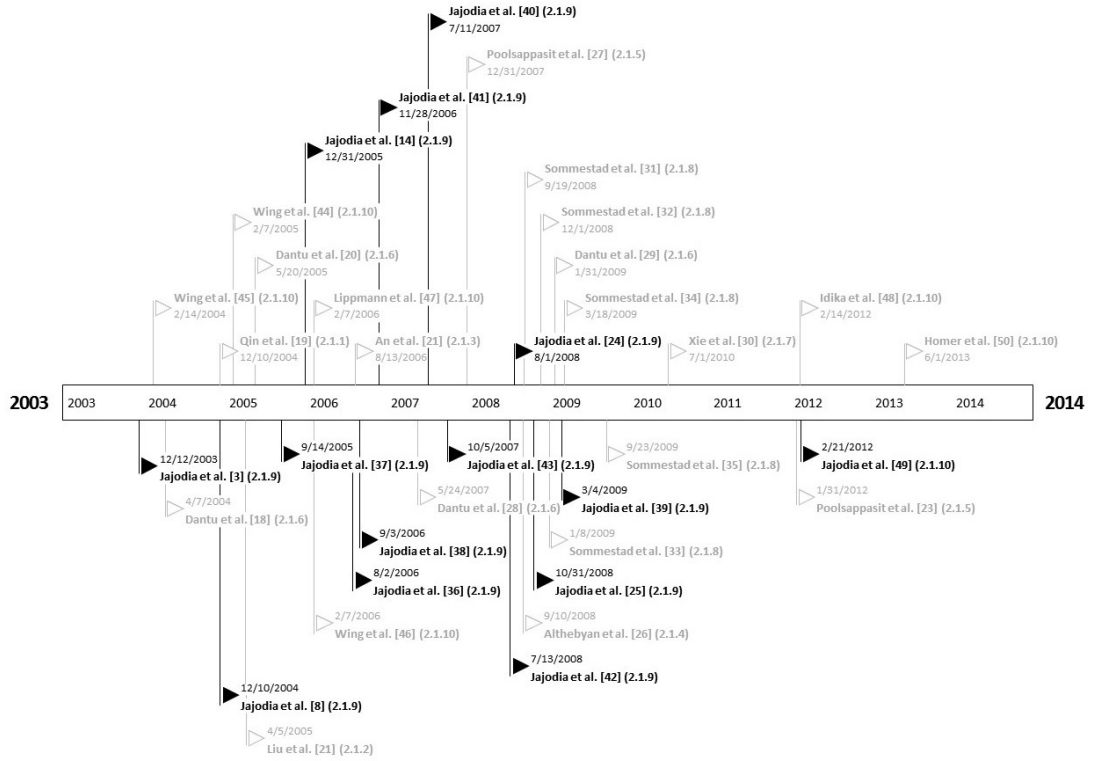



*Figura 7: Timeline articoli di Jajodia et al.*

La filosofia portante di ogni articolo riguarda la correlazione delle vulnerabilità attraverso modelli sempre più evoluti. Gli autori esplorarono queste relazioni attraverso una serie di articoli, presentando un framework generale [40], alcune metriche [41, 42] ed un modello Bayesiano [24, 25].

Il loro primo approccio [3] al problema dell'ottimizzazione delle contromisure venne pubblicato nel 2003; basato sui grafi d'attacco, il TVA (dall'inglese "Topological Vulnerability Analysis") prevede di analizzare le condizioni iniziali anziché le vulnerabilità stesse e,

[13] La classificazione delle contromisure avviene secondo lo standard JQRR (Joint Quarterly Readiness Review), uno standard di valutazione trimestrale di stampo militare che consiste nell'identificazione delle carenze e del rischio associato alle risorse impiegate, con lo scopo di ridurre il rischio complessivo.

postulando la monotonicità del modello, evitarono di analizzare l'insieme dei cammini sul grafo concentrandosi esclusivamente sull'insieme di contromisure applicabili con l'obiettivo di minimizzarne il costo complessivo. A tal scopo trovarono il modo di ridurre il numero di contromisure totali applicando solamente quelle necessarie alla protezione di particolari sottografi.

La ricerca di queste aree del grafo avviene attraverso l'analisi delle precondizioni iniziali, ovvero quelle precondizioni il cui stato è indipendente dal grafo stesso. Il modello, infatti, prevede una codifica booleana delle contromisure durante la fase di ingegnerizzazione, nello specifico il successo di un exploit è definito come una combinazione booleana di contromisure (attivate o disattivate) assumendo una mappatura tale per cui se le precondizioni di un attacco risultano soddisfatte allora anche le sue post-condizioni lo devono essere. I sottografi definiti dalle precondizioni iniziali sono esattamente quelli da analizzare per calcolare l'insieme minimo di contromisure, inoltre essi sono gli unici elementi del grafo sotto stretto controllo da parte dell'amministratore, in quanto un attaccante esperto potrebbe bypassare le misure di sicurezza, rendendo fallace la modellazione negli stati più avanzati del modello.

Nel 2005 [14] realizzarono l'omonimo tool (TVA), basato su strumenti automatici di penetration testing. Il software fornisce uno spunto interessante circa la modellazione topologica delle vulnerabilità utilizzando una funzione di distanza tra le vulnerabilità stesse, nello specifico si basarono su un precedente lavoro sulla correlazione di eventi ed avvisi ottenuti dagli IDS [8].

Qualche anno più tardi, Wang et al. [43] definirono una metrica per stabilire la resistenza del sistema agli attacchi. Basato sul modello matematico descritto in [14] e via via migliorato in [8, 36, 37, 38, 41] consiste in un grafo d'attacco in grado di rappresentare la conoscenza a priori delle vulnerabilità e le loro dipendenze. I nodi sono divisi in due categorie, exploit e condizioni, dove gli exploit rappresentano un'azione (o un insieme di azioni) compiute per sfruttare una vulnerabilità mentre le condizioni codificano uno stato specifico il cui il sistema, o parte di esso si trova. Ogni arco nel grafo può congiungere solamente condizioni con exploit e viceversa (ma non exploit con exploit e condizioni con condizioni). Un exploit, ad esempio, richiede l'esistenza di almeno due condizioni, l'esistenza della vulnerabilità (precondizione) e la condizione che indica la violazione del sistema (post condizione). Gli archi assumono il significato di prerequisito e di implicazione a seconda dell'origine: un arco da un exploit ad una condizione è definito come implicazione (ovvero lo sfruttamento dell'exploit implica lo stato violato), mentre viceversa, un arco da una condizione ad un exploit è definito come prerequisito (per violare l'exploit è necessario che il sistema si trovi in uno stato che soddisfa la condizione). Il concetto di resistenza del sistema dipende strettamente dal numero di vulnerabilità e dalla complessità delle operazioni necessarie per violarle.

Per meglio analizzare le dipendenze tra le vulnerabilità Frigault *et al.* convertirono i modelli precedenti di grafo d'attacco in una rete Bayesiana [25] e, nello stesso anno, realizzarono la necessità di codificare la temporalità all'interno delle reti Bayesiane [24]. Questa esigenza nacque dal problema dell'evoluzione delle vulnerabilità nel tempo, ovvero nel aumento del rischio collegato alla diffusione dei dettagli tecnici relativi agli exploit così come i cambiamenti nel grado di severità. La struttura dei nodi e degli archi non cambia sostanzialmente rispetto ai precedenti modelli, eccezion fatta per le neo introdotte probabilità condizionali associate alle vulnerabilità (definite sulla base delle metriche base e temporali del CVSS).

Infine nel 2009 [39] Noel *et al.* presentarono l'ultima versione del loro software iniziale, il TVA, ora in grado di analizzare il sistema (o la rete) attraverso l'integrazione con altri tool commerciali e con le banche dati di exploit e vulnerabilità. Il TVA è in grado sia di effettuare analisi offensive (penetration testing) sia difensive (ottimizzazione delle contromisure),

lavorando con "algoritmi polinomiali di ordine quadratico" [39], mantenendo comunque problemi irrisolti circa la visualizzazione di grafi di grandi dimensioni.

### 2.1.10 Articoli relativi alle metriche utilizzate [44, 45, 46, 47, 48, 49, 50]

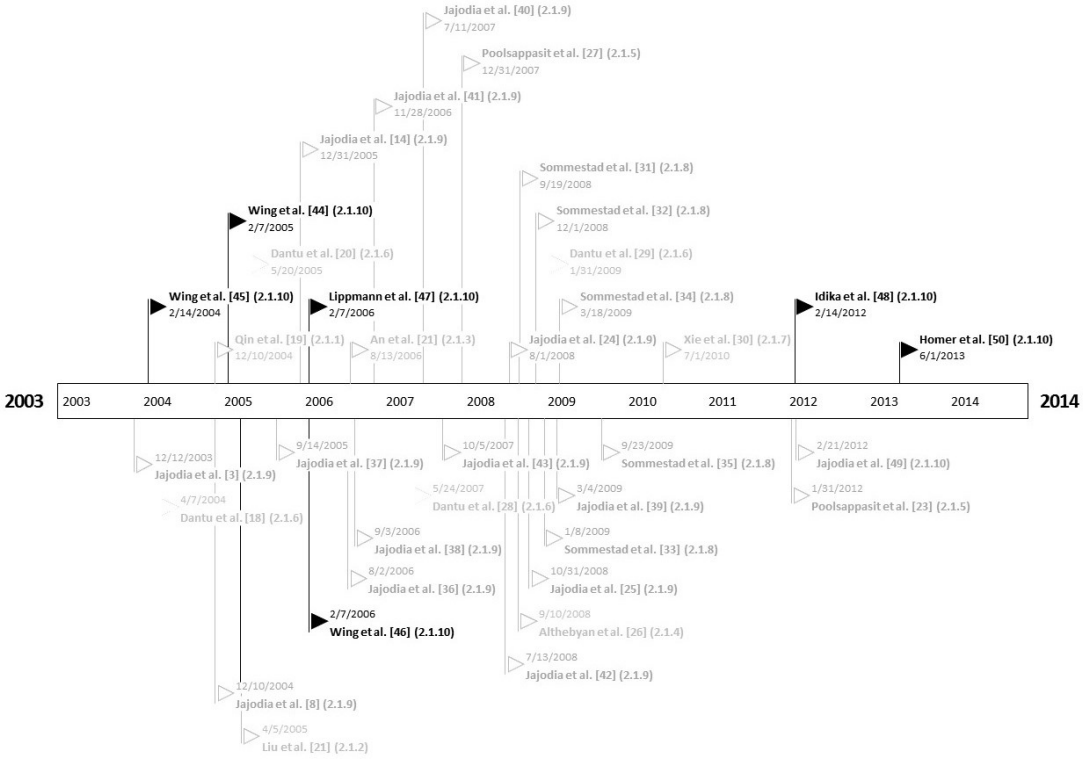



*Figura 8: Timeline articoli relativi alle metriche*

Le relazioni tra le vulnerabilità sono trattate in modo assolutamente arbitrario da ogni autore in letteratura, i cui risultati pertanto non possono essere agevolmente confrontati con altri modelli. L'approccio alle metriche varia nel corso degli ultimi dieci anni, spaziando dalle superfici di attacco di Wing et al. [44, 45, 46], alla gestione del rischio di Pamula et al. [36]. Altri sforzi in questa direzione furono compiuti da Lippmann et al. (NCP[14] [47]) che fornirono un indice per una valutazione immediata dei miglioramenti (o peggioramenti) del sistema al variare di vulnerabilità e contromisure.

Idika e Bhargava [48] si proposero di migliorare tre metriche largamente utilizzate: "*Shortest-Path*" (una metrica grezza che risponde alle domande come "qual è lo sforzo minimo richiesto per violare il sistema?"), "*Number-of-Paths*" (il numero di possibili percorsi d'attacco) e "*Mean-of-Paths-Length*" (la metrica che stima lo sforzo medio richiesto per ogni percorso). Ciascuna metrica presa in esame dagli autori presenta dei problemi di fondo, la prima ignora completamente la quantità dei cammini esistenti, la seconda non prende in considerazione le difficoltà degli attacchi, la terza non considera il sistema complessivo. La metrica proposta dagli autori fu chiamata "Path Length" e corredata da funzioni statistiche come la normalizzazione, la deviazione standard, la moda e la mediana. Su esse basarono una serie di algoritmi volti a

[14] Metrica NCP (*Network Compromise Percentage*) indica la percentuale di risorse compromesse.

confrontare due grafi d'attacco (ad esempio venne formalizzato il concetto di dominanza, fornendo un algoritmo per la sua analisi).

Diametralmente opposte alle metriche di Idika e Bhargava si trova lo standard di riferimento per le vulnerabilità (CVSS). Gli approcci basati su tale standard, tuttavia, non sono in grado di integrare efficacemente i valori da esso ottenuti. I primi ad affrontare l'integrazione come problema a sé stante furono Cheng et al. [49] ed Homer et al. [50], i quali interpretarono ed aggregarono i punteggi dei vettori al fine di migliorarne l'usabilità e l'interoperabilità con i modelli esistenti.

Nello specifico, Cheng et al. [49] notarono come le relazioni tra i punteggi siano visibili solamente con il più alto livello di dettaglio, situazione che preclude appunto l'utilizzo corretto da parte dei modelli di gestione del rischio. L'approccio proposto prevede di modificare i punteggi effettivi sulla base delle relazioni esistenti nel grafo (ad esempio, se un nodo presenta una vulnerabilità di tipo *AdjacentNetwork*[15] e l'attaccante ha già accesso alla rete, il valore verrà aggiornato a *Network*, aumentando di fatto il rischio relativo al nodo). Essi definiscono come "effettivo" questo valore aggiornato. L'aggregazione di questi valori tuttavia è basata su formule matematiche in cui vengono utilizzati dei "*magic numbers*", cioè dei coefficienti numerici (0.6395 oppure 0.2794) i cui valori non sono giustificati in alcun modo, vanificando quindi la validità dell'insieme di equazioni utilizzate.

Homer et al. [50] aggregarono le metriche delle singole vulnerabilità al fine di determinare l'esposizione del sistema (approccio analogo alla superficie d'attacco sopra citata), evidenziando come un'elevata esposizione non implica necessariamente un rischio maggiore rispetto ad un sistema meno esposto. Essi notarono e formalizzarono il problema dell'input impreciso, ovvero la definizione di proprietà basate su dati stimati. Nell'aggregare le informazioni Homer et al. analizzarono il concetto di "*d-separazione*"[16], utilizzandolo per migliorare la definizione classica di probabilità condizionale, fornendo due algoritmi per il calcolo e la valutazione del risultato finale.

[15] Cfr. app. 1.1.1: CVSS Punteggio Base *AccessVector*.

[16] Esempio di "d-separazione" con $d = 3$: due nodi nel grafo sono 3-separati se la loro distanza sul grafo è pari a 3 nodi.

# Capitolo 3

## 3 IL SIMULATORE

Il software realizzato concretizza gli studi eseguiti in questo settore disciplinare dall'autore in collaborazione con il team del prof. Gregorio Martinez dell'Università di Murcia (Spagna).

Lo scopo del progetto riguarda lo studio e l'implementazione di un framework in grado di eseguire differenti modelli matematici per grafi d'attacco al fine di compararne l'efficacia e testarne l'affidabilità. I possibili utilizzi spaziano dal confronto tra diverse teorie ed algoritmi all'aiuto nell'implementazione di nuove varianti o nuovi approcci al problema dell'ottimizzazione delle contromisure. A tal proposito il framework fornisce un'interfaccia unica per ogni modello, consentendo al simulatore stesso di interagire con il codice utente indipendentemente dalla sua implementazione.

Le principali sfide affrontate riguardano le differenze intrinseche tra i modelli da implementare i quali, pur essendo classificati nella stessa categoria (grafi d'attacco o reti Bayesiane), presentano differenze teorico/pratiche tali da impedirne un confronto diretto. Il "come" comparare due modelli rappresenta un problema "difficile" in quanto non esistono due modelli che risolvano esattamente la stessa cosa in due modi differenti: ogni modello, infatti, presenta strutture peculiari e particolarità volte a risolvere un particolare aspetto del problema generale. Il confronto in termini di risultato è pertanto utilizzabile esclusivamente in un contesto ridotto rispetto alla generalità delle soluzioni presenti in letteratura.

Un confronto tra due modelli è teoricamente possibile nonostante queste difficoltà, tuttavia all'atto pratico un paragone teorico (come un'analisi di complessità in termini di spazio/tempo) risulta pressoché inutile. L'esempio può essere trovato analizzando l'articolo di Poolsappasit *et al.* [23], i quali proposero una breve analisi sulla complessità generale del loro modello senza, tuttavia, dettagliarne gli algoritmi impiegati e le analisi delle formule ricorsive impiegate (ad esempio non viene analizzata la questione relativa all'esponenzialità dell'inferenza Bayesiana).

Il simulatore ha poi evidenziato come lo sviluppo concreto di strutture teoriche (i grafi ACT di Trivedi *et al.* così come le reti BAG di Poolsappasit *et al.*) non sia possibile senza compiere drastiche scelte implementative. Nel caso relativo alla codifica del modello BAG gli autori glissano sulla parte relativa all'algoritmo genetico utilizzato e sulla struttura di base dell'insieme di contromisure. Quest'ultima, in particolare, risulta essere particolarmente dubbia: per definizione l'insieme delle contromisure è un vettore ordinato, tuttavia le operazioni compiute su di esso sono caratteristiche degli insiemi, o più in generale delle collezioni non ordinate di

dati. Analogamente in [23] viene presentato un esempio derivato da un contesto reale, senza tuttavia fornirne gli elementi di input e rendendo pertanto impossibile la validazione del risultato ottenuto dal software implementato.

Il confronto tra modelli basato su questi presupposti è pertanto difficile e, soprattutto, arbitrario e soggettivo. Il simulatore tuttavia potrebbe aiutare i ricercatori a risolvere questi problemi attraverso l'implementazione di un modello funzionante a cui far riferimento nei futuri lavori di ricerca.

Multigraph è realizzato completamente in Java[17], senza vincoli sulle tecnologie utilizzate nell'implementazione dei modelli di decisione. È stato testato positivamente su tre differenti distribuzioni di Unix (Debian 7, Ubuntu 14 LTS ed Arch Linux) e due differenti versioni di Windows (7 ed 8.1).

[17] Compatibile con versioni 8+.

## 3.1 ARCHITETTURA

Il simulatore si sviluppa in un contesto multi-thread, con sottosistemi distinti ed indipendenti facenti capo alla classe centrale (*MainClass*) alla quale sono affidate le funzionalità di integrazione dei dati. I blocchi principali sono tre: modulo interfaccia (cfr. 3.1.1), modulo decisionale (cfr. 3.1.2) e modulo di gestione eventi (cfr. 3.1.3), qui rappresentati graficamente in Figura 9. Ciascun sottosistema può essere sostituito con una versione personalizzata e più adeguata al modello da implementare il cui unico limite è definito dal rispettare l'interfaccia di accesso pubblica.

Il flusso di controllo relativo alle funzionalità principali è descritto nella sezione 3.2.

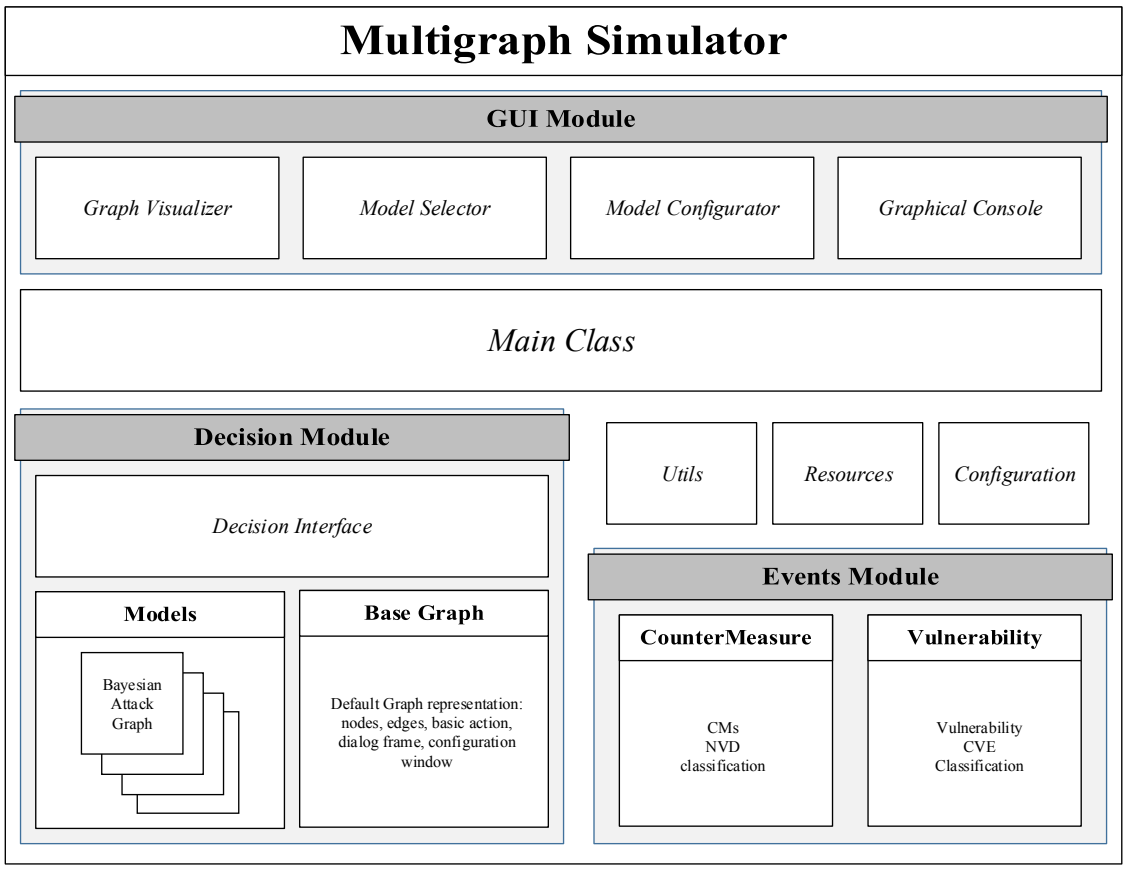



*Figura 9: Architettura Simulatore*

### 3.1.1 Interfaccia Grafica (GUI Module)

#### 3.1.1.1 Renderizzatore

Basato sulla libreria grafica JGraphX [51], la finestra di visualizzazione grafica consente di rappresentare il grafo oggetto di studio durante ogni fase dell'esecuzione di un modello (oltre che consentire l'interazione utente con il grafo stesso). In Figura 10 è possibile vederne un esempio realizzato sul modello base presentato da Poolsappasit *et al.* [23], in cui attraverso una scala colorata viene rappresentato il livello di rischio dei singoli nodi. La versione attuale del simulatore consente l'interazione con il grafo in modo limitato, l'utente può esclusivamente interagire con il layout e la disposizione degli elementi visualizzati.

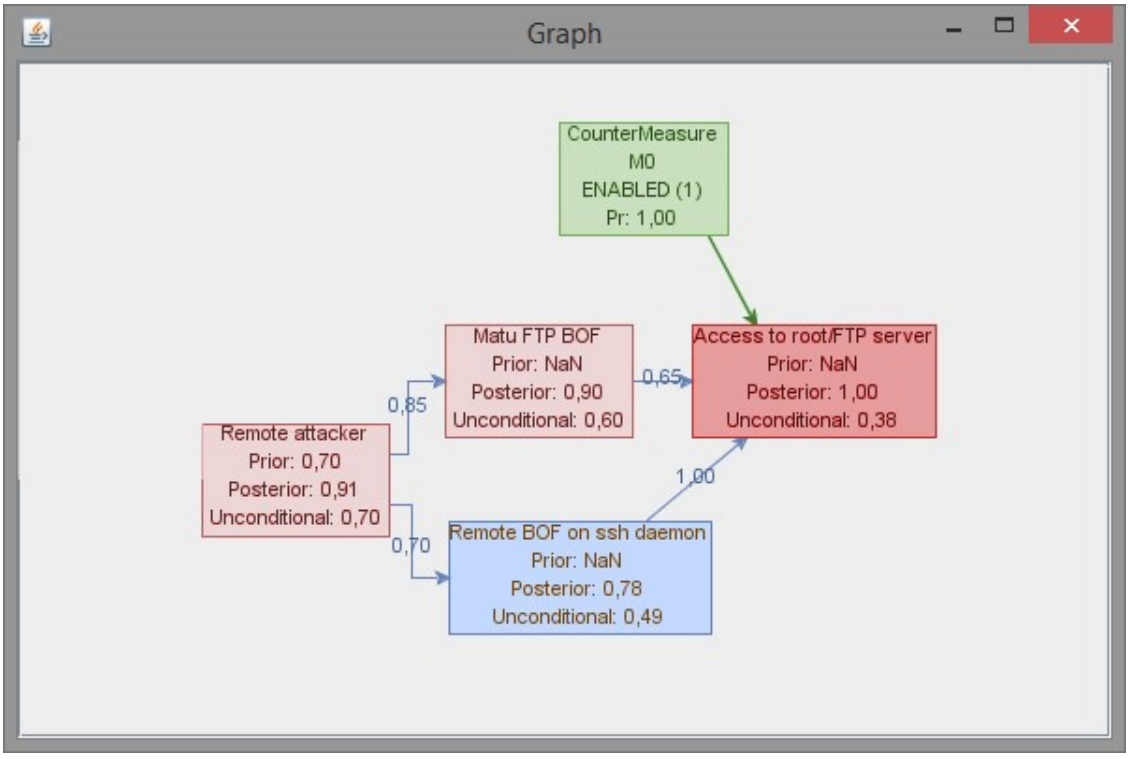



*Figura 10: Esempio di grafo renderizzato dal simulatore*

#### 3.1.1.2 Finestra Principale (Model Selector)

La finestra principale, mostrata in Figura 11, riflette le funzionalità essenziali della classe di controllo (MainClass). Da questa finestra è possibile attivare (o disattivare) ogni singolo elemento del simulatore, compresi i thread decisionali e di gestione eventi. È altresì possibile inizializzare il simulatore con un modello precaricato, bypassando di fatto l'inserimento dei dati.

Nella zona di sinistra sono presenti la console di azioni rapide (come attivare/disattivare i vari thread), il selettore per il modello da eseguire e la console di log. Nella zona di destra sono invece disponibili grafici di riepilogo personalizzabili dal modello che si intende eseguire (in Figura 11 è visibile l'utilizzo di memoria rispetto al tempo di esecuzione, il pannello tempo fornisce una misurazione rispetto ai tempi di elaborazione, mentre il terzo pannello fornisce un grafico per l'evoluzione della dimensione del grafo).

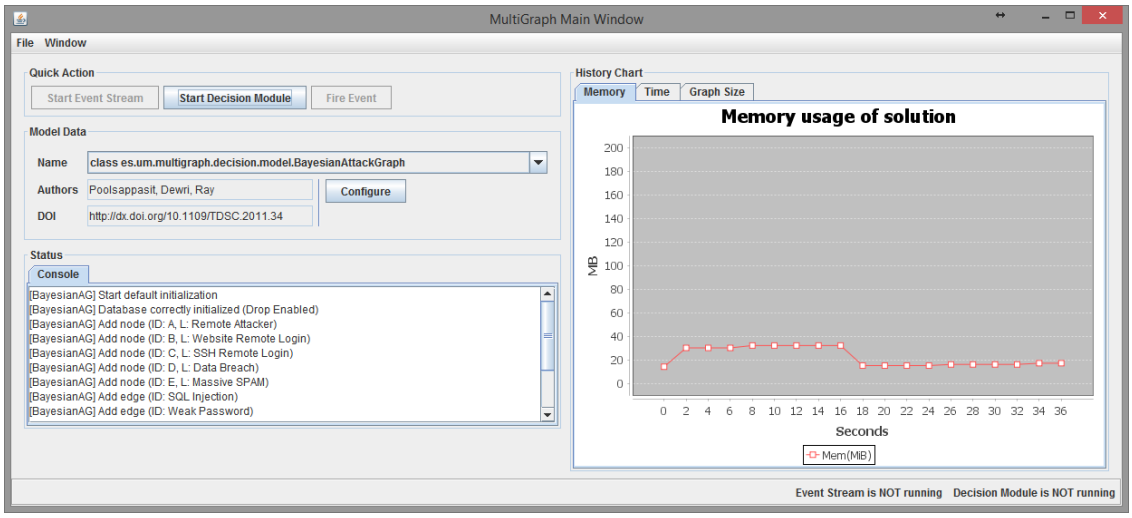



*Figura 11: Finestra principale del simulatore*

#### 3.1.1.3 Configuratore

Il configuratore è una delle finestre principali e richiede una personalizzazione profonda a seconda di ogni modello. A tal proposito si noti la differenza tra le finestre mostrate in Figura 12 ed in Figura 13: la prima consiste in un'interfaccia grafica spartana, che utilizza le funzionalità di "*reflection*" Java per manipolare nodi ed archi di qualsiasi modello, la seconda invece è un'interfaccia personalizzata per il modello di Poolsappasit et al. [23] che consente interazioni con il modello molto più avanzate e dettagliate.

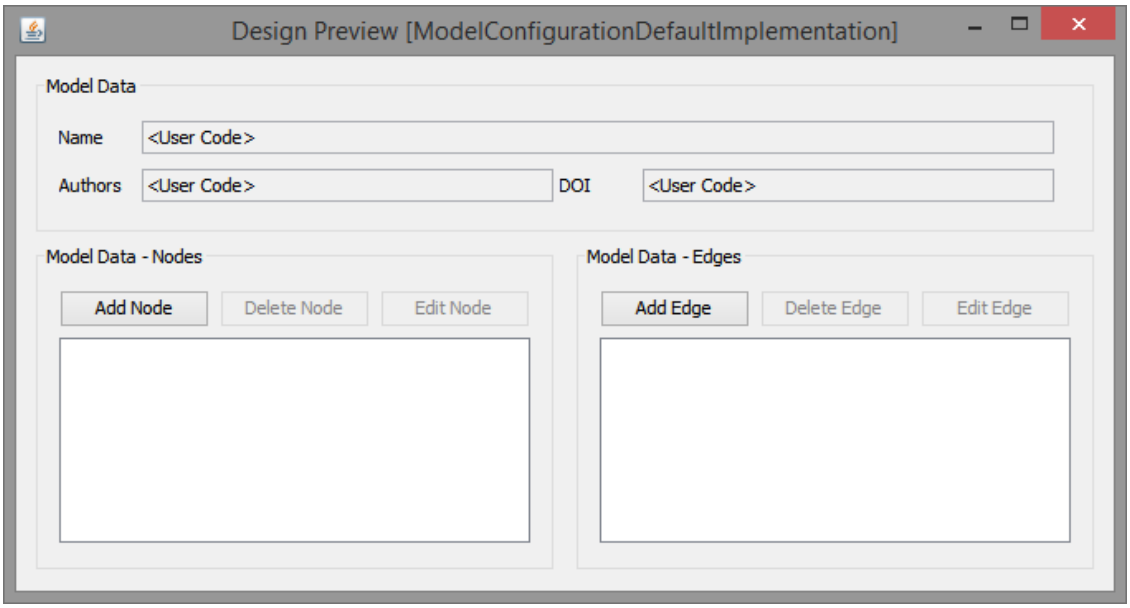



*Figura 12: Configuratore di default per i moduli decisionali del simulatore*

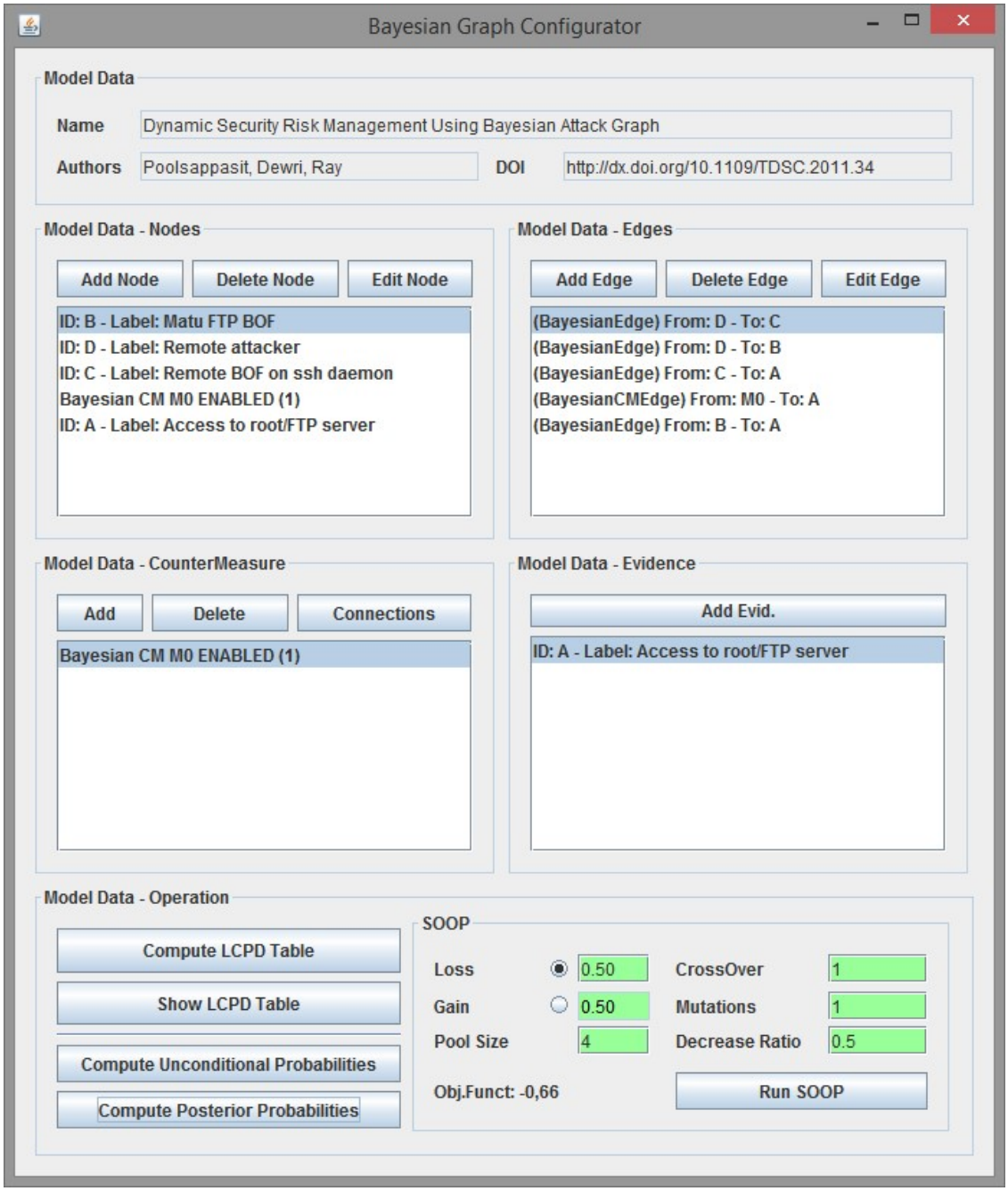



*Figura 13: Configuratore per il modello presentato in [23]*

### 3.1.2 Modulo di Decisione

L'implementazione di ogni modello deve rispettare i vincoli posti dalla classe interfaccia `DecisionInterface` al fine di consentire una interazione continuativa con il simulatore stesso.

#### 3.1.2.1 Interfaccia pubblica (DecisionInterface)

L'interfaccia generica per i modelli da implementare richiede la parametrizzazione dei nodi e degli archi (`N extends Node, E extends Edge`). Per il caso specifico dell'implementazione del BAG [23] sono "`BayesianNode`" per i nodi e "`BayesianEdge`" per gli archi.

L'interfaccia estende `EventListener` per consentire la gestione degli eventi dalla classe principale e `Runnable` per consentire l'esecuzione multi-thread del simulatore stesso. L'inizializzazione del costruttore è lasciata libera, tuttavia è richiesta l'implementazione di un metodo `init(MainClass main)` che richiede come argomento il puntatore alla classe principale.

Il modello deve implementare metodi pubblici per le informazioni di base (autore, titolo ecc.), e per il controllo della struttura di base del grafo (aggiunta, rimozione, modifica di nodi ed archi).

#### 3.1.2.2 Struttura default

La struttura di base di ogni modello utilizzabile con il simulatore deve estendere le classi `Node` per i vertici del grafo ed `Edge` per gli archi. Un nodo è definito da un id (`getID`), un'etichetta (`getLabel`), l'insieme dei padri (`getIn`) e l'insieme dei figli (`getOut`). Sono inoltre presenti i metodi che forniscono le finestre grafiche per l'interazione con l'utente (`getJDialog`). Analogamente un arco è definito da un id (`getID`) e dai suoi due estremi (`getTo` e `getFrom`).

Sia nodi che archi non sono di default ordinabili, lasciando al modello la possibilità di implementare un comparatore.

### 3.1.3 Modulo di Gestione Eventi

Il modulo si occupa della gestione di ogni tipo di evento con il quale il simulatore dovrà interagire. Dato un generico evento (ad esempio una nuova vulnerabilità, un avviso da un sensore, ecc.) questo modulo deve essere in grado di decodificarne il contenuto per rielaborarlo in funzione di un utilizzo futuro da parte del modulo centrale. L'idea è quella di consentire al simulatore di interagire con eventi differenti, provenienti da fonti eterogenee e la cui tipologia può variare nel tempo (si pensi ad esempio a contromisure la cui efficacia cala nel tempo). A tal proposito il simulatore considera il gestore eventi come una scatola nera, al quale fornisce dettagli e specifiche per gli eventi che sono riconosciuti dal sistema, e dal quale riceve eventi non predicibili.

Nel prototipo presentato in [17] questo sottosistema è limitato alla codifica delle vulnerabilità e delle contromisure di esempio implementando rispettivamente le interfacce `Vulnerability` e `Solution`. La classe interfaccia `Vulnerability` consta di due metodi principali: `getRisk` (che restituisce il rischio associato alla vulnerabilità) e `getAffectedElements` (che restituisce la codifica dei sistemi che possono essere affetti da tale vulnerabilità). Analogamente, la classe interfaccia `Solution` consta di due gruppi di funzionalità: i metodi per il rischio (`getLowList`, `getMediumList`, `getHighList`, `getCost`, `getPriority`) ed i metodi per la parametrizzazione delle contromisure applicate (`getParams`, `getListModel`, `getControl`).

## 3.2 FLUSSO DATI E FLUSSO DI CONTROLLO

Nell'esempio in Figura 14 sono codificate il 90% delle funzionalità previste del simulatore. Consta di quattro attori principali (gestione eventi, modulo centrale, modulo decisionale, simulatore di rete) intenti ad analizzare un nuovo evento (come una vulnerabilità) ed eventualmente reagire con una soluzione ottimale.

L'inizializzazione del simulatore avviene attivando ogni modulo, tuttavia l'esecuzione principale inizia esclusivamente dopo la ricezione di un evento da parte del gestore eventi. Il nucleo centrale del simulatore decide a quale modulo decisionale assegnare l'evento (potenzialmente a tutti), ciascuno di essi dovrà interpretarlo ed elaborare una strategia di risposta.

Se la strategia di risposta include cambiamenti questi verranno inviati al simulatore di rete per l'applicazione e lo studio della loro efficacia. In caso di malfunzionamenti o comportamenti inaspettati il modulo di decisione dovrà analizzare questi nuovi dati alla ricerca di una possibile correlazione e proporre una soluzione alternativa all'evento iniziale.

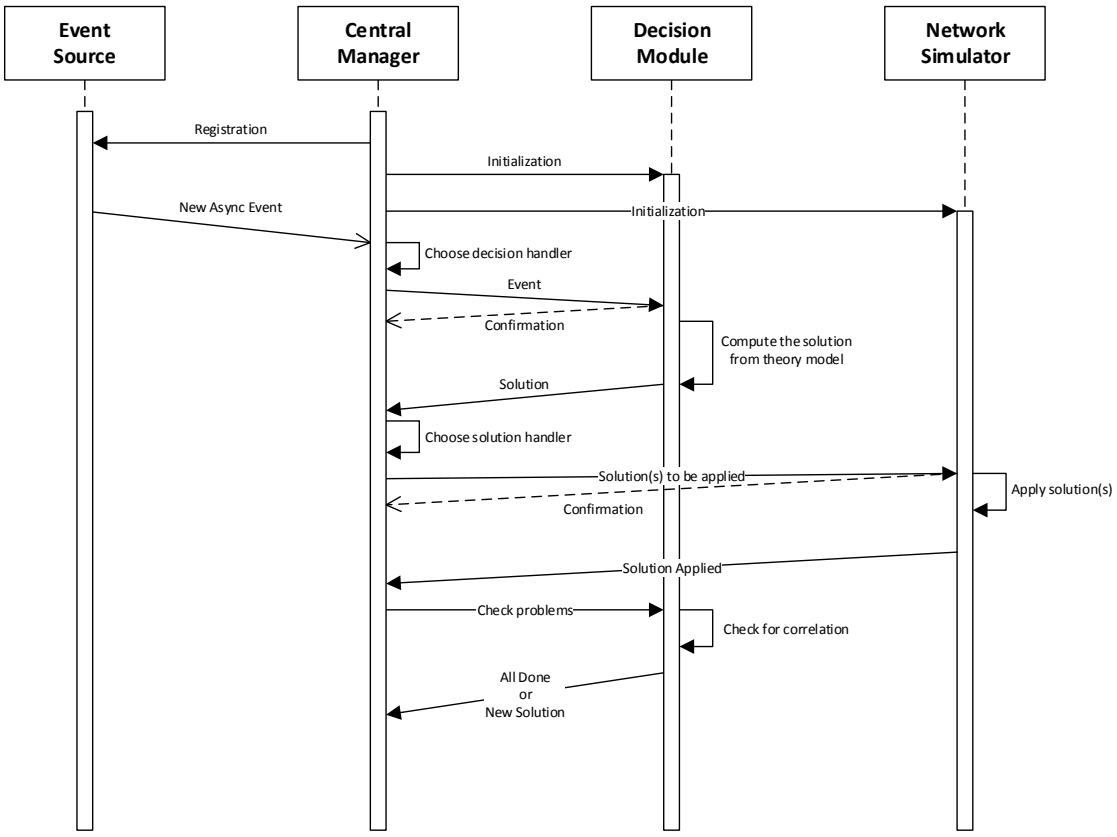



*Figura 14: Flusso di controllo previsto per il simulatore completo*

Attualmente, la versione presentata in [17], è limitata alle funzionalità mostrate in nero nella rappresentazione disponibile in Figura 15. Il simulatore è, infatti, in grado di gestire uno o più modelli, tuttavia non è ancora pronto per l'utilizzo in un contesto reale con eventi ottenuti da IDS od aggregatori e l'applicazione reale delle contromisure scelte.

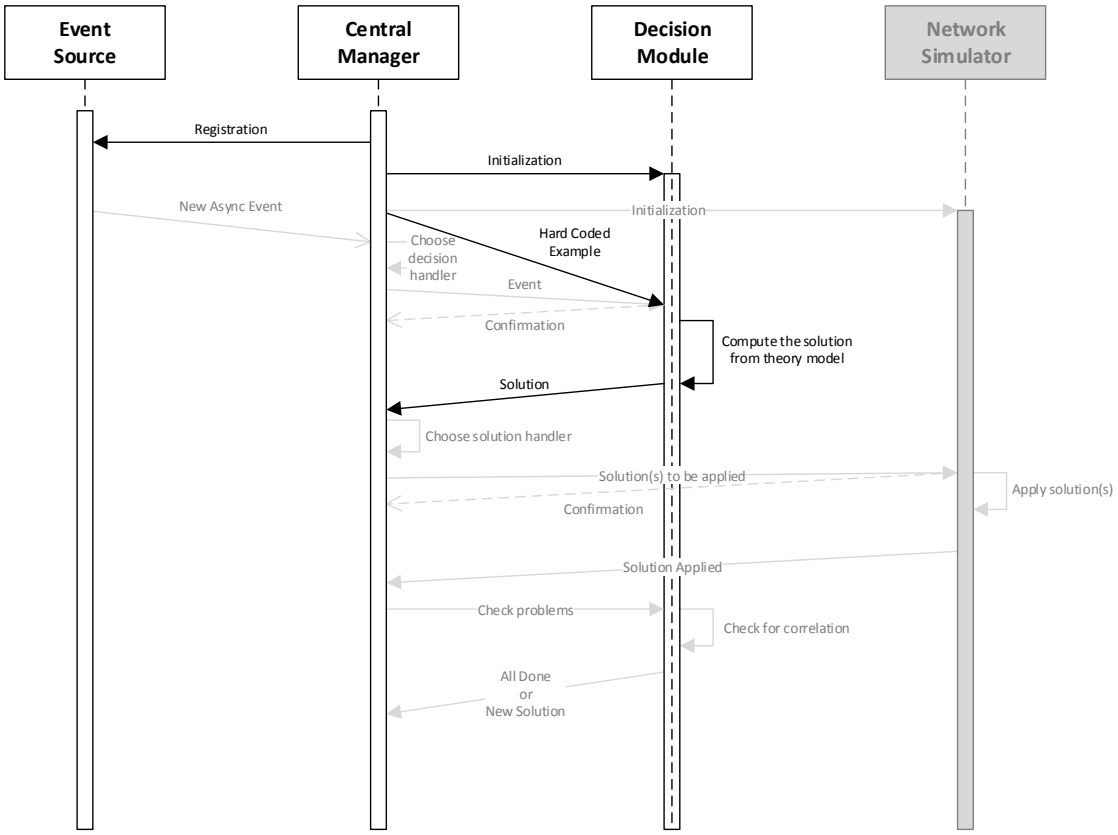


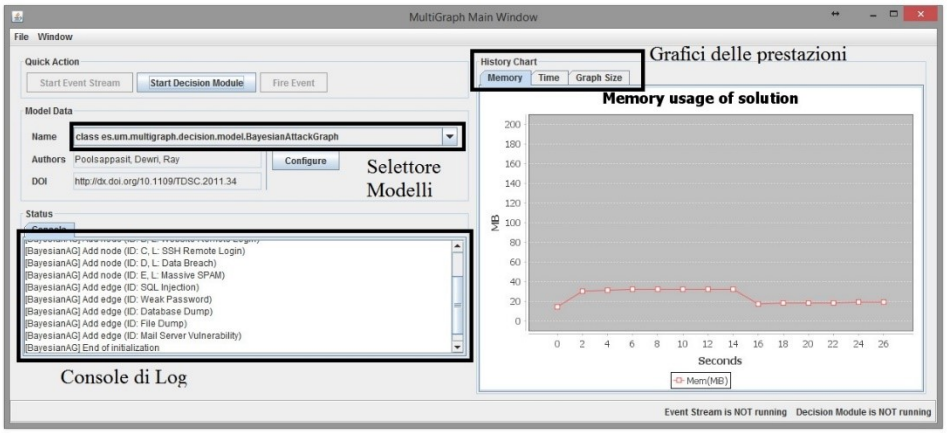

*Figura 15: Flusso di controllo implementato nell'attuale versione*

## 3.3 SVILUPPI FUTURI

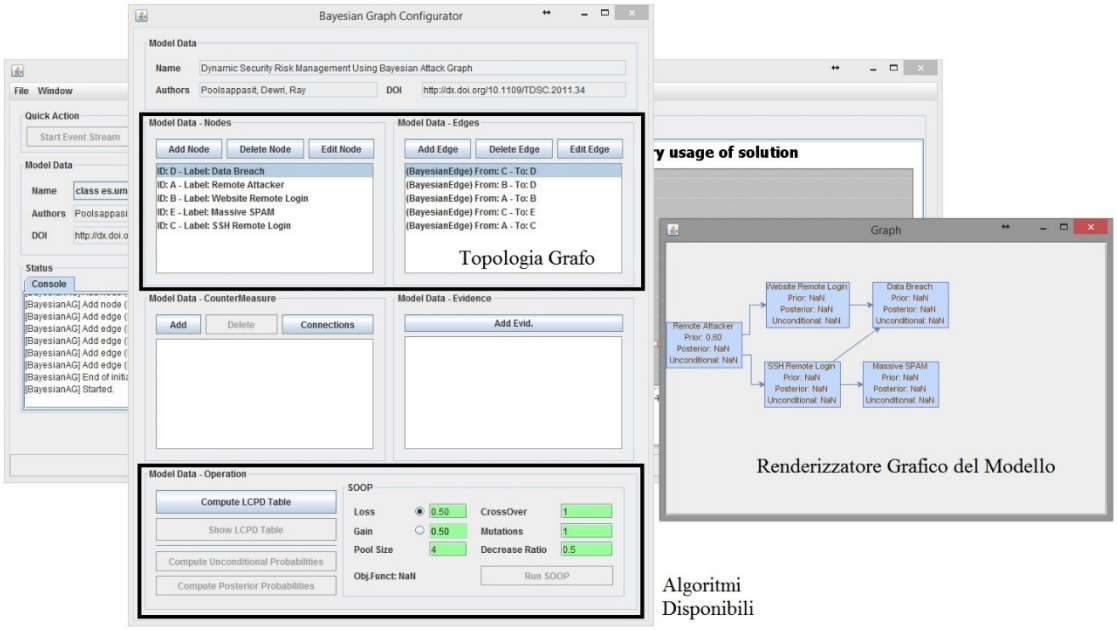
Il simulatore qui presentato nel dettaglio è ancora in fase prototipale, pertanto ha a disposizione solamente limitate funzioni di interazione con l'utente. Ad esempio per quanto riguarda il visualizzatore verranno implementati metodi che consentiranno di visualizzare informazioni multi-livello, come ad esempio il rischio complessivo, i dettagli interni di ogni nodo, la navigazione tra nodi correlati e la modifica istantanea di proprietà e valori.

## 3.4 ESECUZIONE SIMULATA

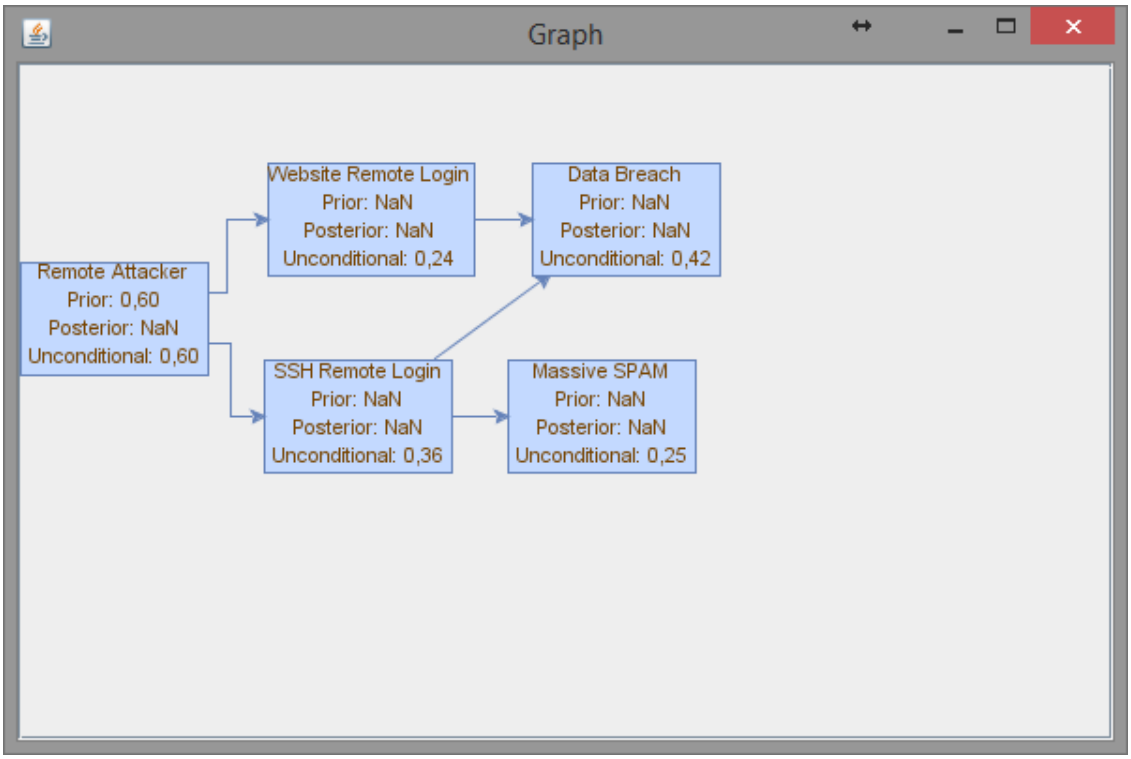
L'esecuzione simulata di seguito rappresentata riporta il comportamento del simulatore durante l'esecuzione del modello [23].



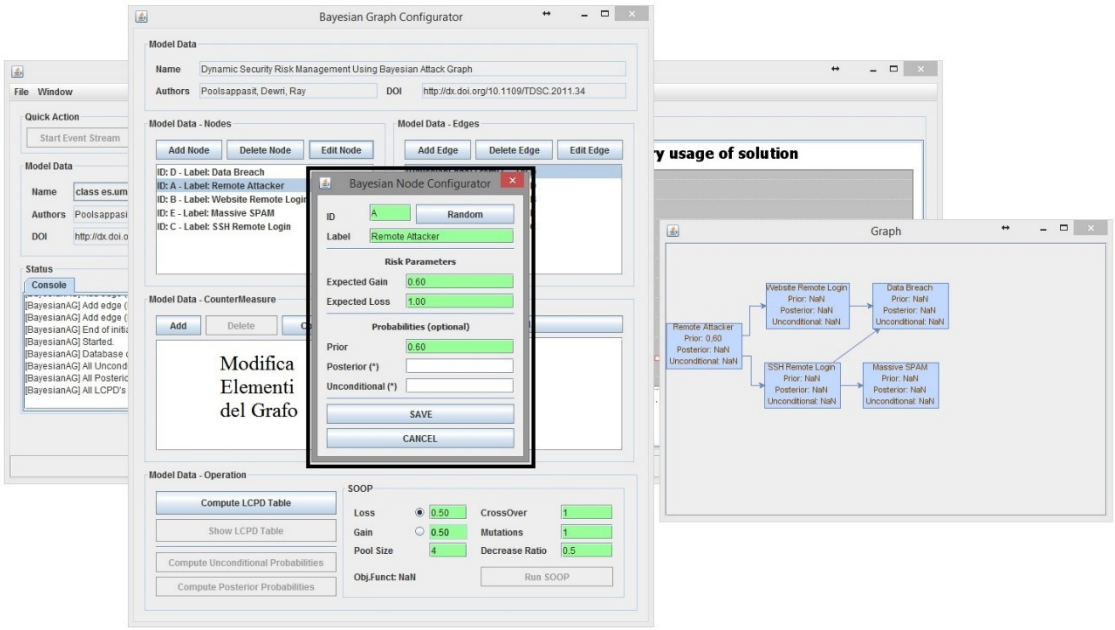

*Simulatore - Fase 1: Finestra principale*



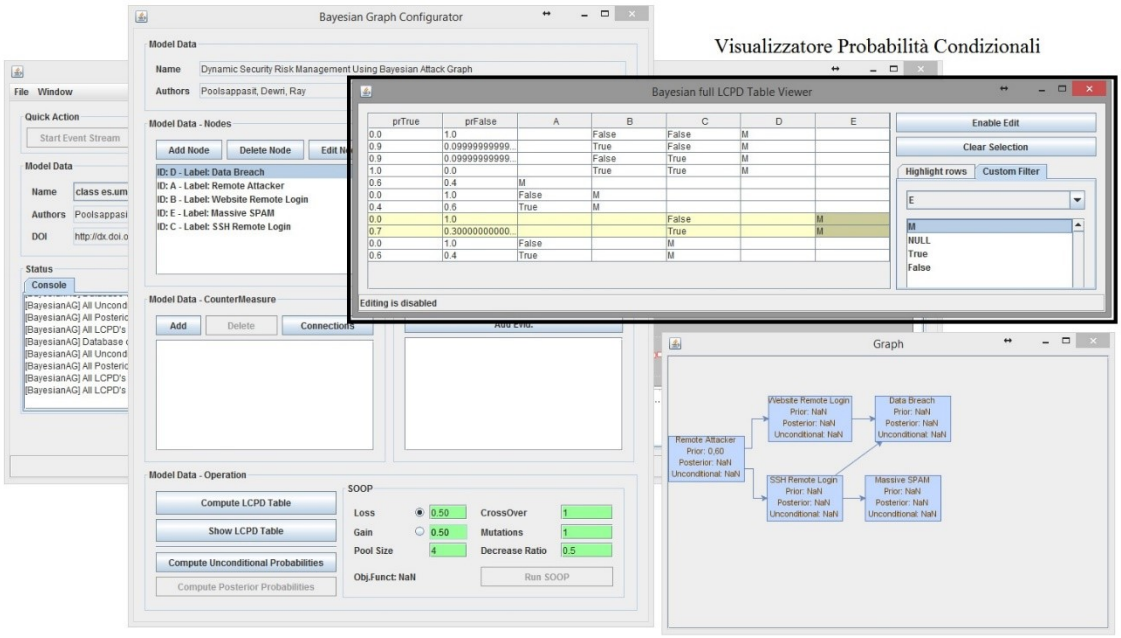

*Simulatore - Fase 2: Avvio del modello*



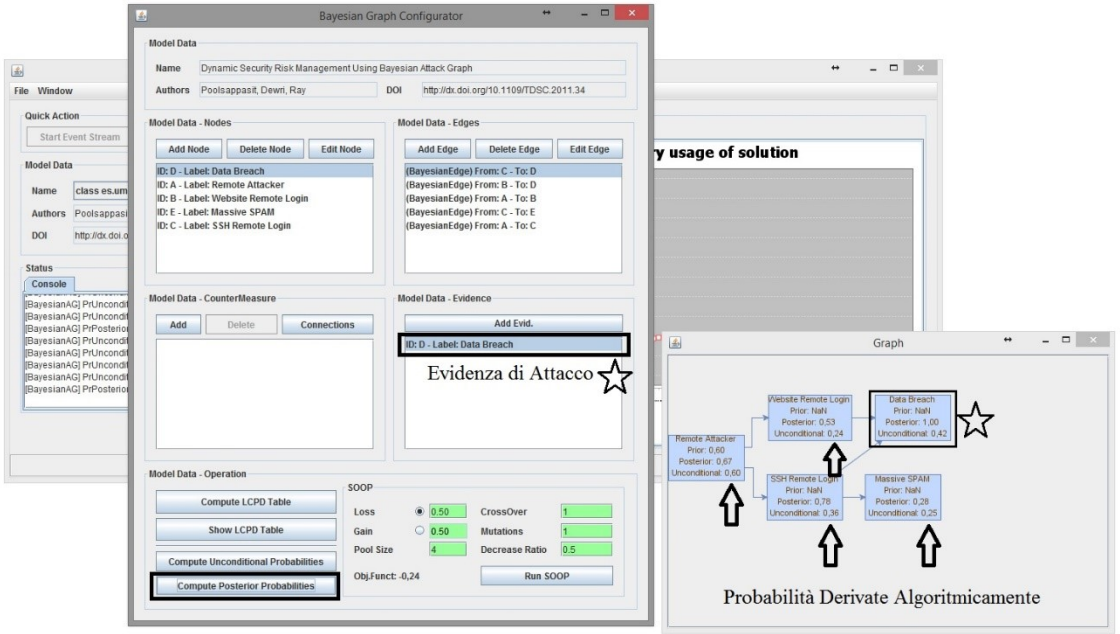

*Simulatore - Fase 3: Dettaglio del grafo*



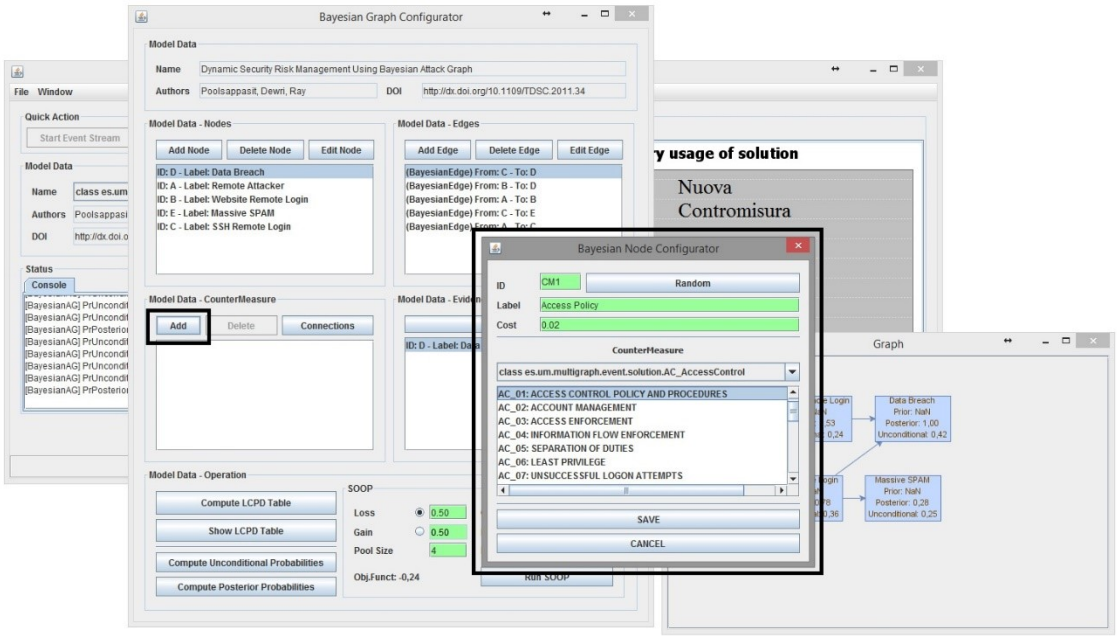

*Simulatore - Fase 4: Modifica di un nodo*



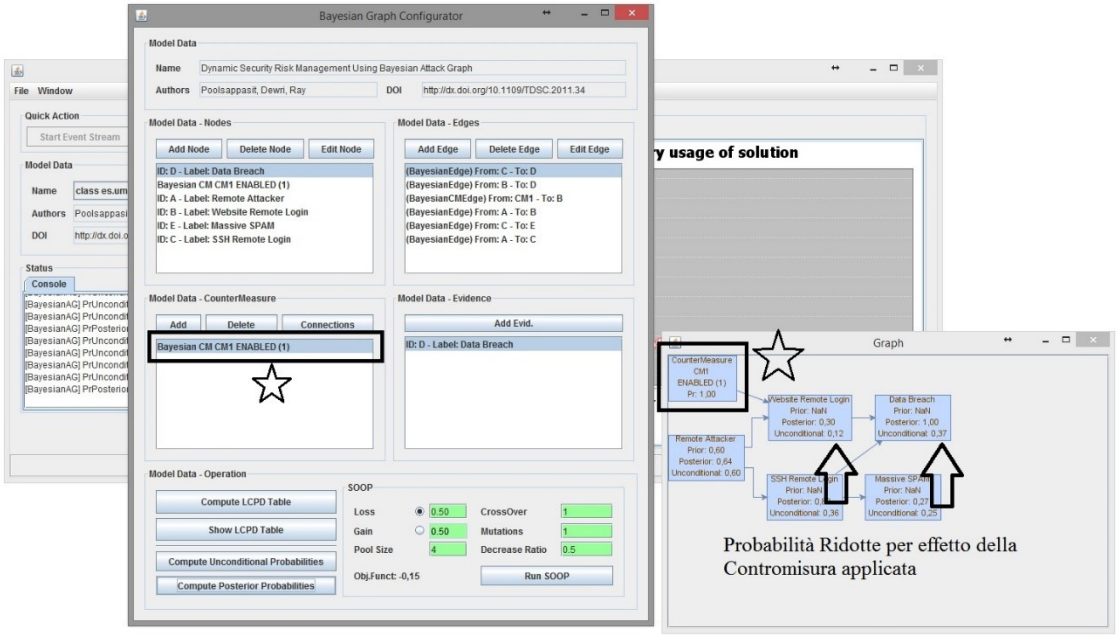

*Simulatore - Fase 5: Visualizzazione della tabella delle probabilità condizionali*




*Simulatore - Fase 6: Aggiunta di una evidenza e calcolo delle probabilità a posteriori*




*Simulatore - Fase 7: Aggiunta di una contromisura*




*Simulatore - Fase 8: Aggiornamento delle probabilità in funzione della nuova contromisura*

# Capitolo 4

## 4 CONCLUSIONI

Gli approcci alla sicurezza informatica studiati in questo documento possono essere associati a due filoni di ricerca distinti: il primo è quello dell'unificazione e standardizzazione (come i lavori dei gruppi di Jajodia e Sommestad che proposero nuovi modelli volti ad amalgamare i precedenti risultati), mentre il secondo è quello della specificazione (ovvero lo studio di un problema specifico, come i lavori di An *et al.* ed Althebyan *et al.*). I primi cercarono di proporre soluzioni generali corredate di analisi e descrizioni formali o di studi teoretici dei problemi, ottenendo sia risultati di successo [23, 52, 48] sia "*case-study*" isolati. I secondi invece svilupparono tecniche di risoluzione specifiche, ottenendo risultati brillanti nei rispettivi domini di applicazione [53], ma di difficile trasposizione nei modelli più generali [22, 26].

La ricerca per analizzare metodicamente i sistemi e le reti tenendo conto di informazioni parziali o ignote è ancora agli albori, tuttavia i risultati ottenuti dall'applicazione delle reti Bayesiane, e più in generale dai grafi d'attacco sono promettenti ed in rapida espansione: si veda ad esempio la generazione di modelli basata su tool automatici (TVA [39], MulVAL [4], GeNIe & SMILE [54]) sempre più efficienti e tuttavia non ancora completamente autonomi. L'amministratore necessita infatti di un compromesso tra le misure di sicurezza applicabili (ovvero un insieme in grado di bloccare ogni possibile attacco), il danno causato dalle vulnerabilità non coperte (o residui) ed il budget previsto per la sicurezza (Dilemma dell'Amministratore [27]).

In questo contesto nasce l'idea di uno strumento in grado di comparare diversi modelli ed aiutare i ricercatori nella ricerca di nuovi approcci. Il simulatore sviluppato ha difatti evidenziato come in questo ambito sia assente perfino una strategia comune ai vari modelli, dimostrando come un approccio comune sia necessario sia a livello strutturale sia a livello algoritmico.

Nel dettaglio il progetto Multigraph ha gettato le basi per la prototipazione dei modelli facenti capo alla categoria generica dei grafi d'attacco, pur mostrando i punti deboli dei lavori esistenti in letteratura. Si pensi ad esempio al modello di Poolsappasit *et al.* [23] che include ogni aspetto formale circa il modello, glissando però sui dettagli implementativi o sullo svolgimento concreto delle formule: eccezion fatta per analogie e differenze teoriche, su quali basi può essere confrontato con altri modelli?

Il simulatore è in grado di indicare con estrema precisione quali sono gli aspetti inadeguati dei modelli che verranno implementati, evidenziando come l'eterogeneità di pensiero non sia solamente a livello di idea ed approccio al problema ma anche a livello strutturale ed algoritmico.

I risultati ottenuti sono pertanto promettenti e giustificano ulteriori sforzi di ricerca in questa direzione, nello specifico sono molte le domande che ancora attendono una risposta: come dovrebbe essere definita la probabilità di un sistema di essere compromesso? Dobbiamo rappresentare il sistema (o la rete) in termini di sistemi, di servizi o di entrambi? Cosa cambia in un contesto virtualizzato? Ed in uno distribuito?

In sintesi la ricerca deve continuare cercando di porre l'attenzione sullo sviluppo di un modello strutturale comune, con input/output standardizzati e comparabili, applicabile verticalmente dall'analisi dei log all'ottimizzazione delle contromisure.

# Appendici

## 1 COMMON VULNERABILITY SCORE SYSTEM (V2.0)

Il sistema CVSS [55] è un sistema ideato dal NIAC[18] nel 2004 [56] con l'obiettivo di "*proporre un sistema di punteggi per le vulnerabilità al fine di promuovere una miglior conoscenza delle vulnerabilità stesse ed il loro impatto*" [56]. Le metriche furono ampiamente utilizzate [24, 23, 42, 40, 43, 57] e, nel 2011 la seconda versione è stata formalmente adottata come standard internazionale con il codice ITU-TX.1521[19].

Il CVSS si basa su tre scale principali: *base* (che rappresenta le caratteristiche fondamentali di ogni vulnerabilità), *temporale* (che rappresenta le caratteristiche che cambiano nel tempo) ed *ambientale* (che rappresenta le caratteristiche rilevanti esclusivamente per uno specifico ambiente). Ciascuna metrica è basata su di un punteggio compreso fra 0.0 e 10.0.

Il punteggio associato ad ogni scala principale è calcolato sulla base di molteplici indicatori:

- **Punteggio Base**
  - *Access Vector* (AV): riflette come una vulnerabilità viene sfruttata; i valori ammessi sono Local (L), Adjacent Network (A), e Network (N).
  - *Access Complexity* (AC): riflette la complessità del codice utilizzato per l'exploit; i valori ammessi sono High (H), Medium (M) e Low (L).
  - *Authentication* (AU): riflette il numero di autenticazioni necessarie per poter sfruttare la vulnerabilità; i valori ammessi sono Multiple (M), Single (S) e None (N).
  - *Confidentiality Impact* (C): riflette l'impatto sulla confidenzialità; i valori ammessi sono None (N), Partial (P) e Complete (C).
  - *Integrity Impact* (I): riflette l'impatto sull'integrità; i valori ammessi sono None (N), Partial (P) e Complete (C).
  - *Availability Impact* (A): riflette l'impatto sulla disponibilità; i valori ammessi sono None (N), Partial (P) e Complete (C).
- **Punteggio Temporale:**
  - *Exploitability* (E): riflette lo stato attuale circa la diffusione del codice dell'exploit; i valori ammessi sono Unproven (U), Proof-Of-Concept (POC), Functional (F), High (H) e Not-Defined (ND).

[18] National Infrastructure Advisory Council (NIAC) http://www.dhs.gov/national-infrastructure-advisory-council
[19] https://www.itu.int/rec/T-REC-X.1521-201104-I/en.

  - *Remediation Level* (RL): riflette la disponibilità di patch od aggiornamenti; i valori ammessi sono Official-Fix (OF), Temporary-Fix (TF), Workaround (W), Unavailable (U) e Not-Defined (ND).
  - *Report Confidence* (RC): riflette la fiducia nell'esistenza effettiva della vulnerabilità; i valori ammessi sono Unconfirmed (UC), Uncorroborated (UR), Confirmed (C) e Not-Defined (ND).
- **Punteggio Ambientale**
  - *Collateral Damage Potential* (CDP): riflette il danno fisico potenziale; i valori ammessi sono None (N), Low (L), Low-Medium (LM), Medium-High (MH), High (H) e Not-Defined (ND).
  - *Target Distribution* (TD): riflette la diffusione di sistemi vulnerabili; i valori ammessi sono None (N), Low (L), Medium (M), High (H) e Not-Defined (ND).
  - *Confidentiality Requirement* (CR): riflette l'importanza delle risorse in termini di confidenzialità; i valori ammessi sono Low (L), Medium (M), High (H) e Not-Defined (ND).
  - *Integrity Requirement* (IR): riflette l'importanza delle risorse in termini di integrità; i valori ammessi sono Low (L), Medium (M), High (H) e Not-Defined (ND).
  - *Availability Requirement* (AR): riflette l'importanza delle risorse in termini di disponibilità; i valori ammessi sono Low (L), Medium (M), High (H) e Not-Defined (ND).

Ogni gruppo è codificato in un vettore, nel quale ogni punteggio è separato da *slash* ("/") e costituito dall'abbreviazione relativa più il valore assegnato:

- **Vettore di Base**: $\langle \text{AV}: \{\text{L}, \text{A}, \text{N}\}/\text{AC}: \{\text{H}, \text{M}, \text{L}\}/\text{AU}: \{\text{M}, \text{S}, \text{N}\}/\text{C}: \{\text{N}, \text{P}, \text{C}\}/\text{I}: \{\text{N}, \text{P}, \text{C}\}/\text{A}: \{\text{N}, \text{P}, \text{C}\}\rangle$
- **Vettore Temporale**: $\langle \text{E}: \{\text{U}, \text{POC}, \text{F}, \text{H}, \text{ND}\}/\text{RL}: \{\text{OF}, \text{TF}, \text{W}, \text{U}, \text{ND}\}/\text{RC}: \{\text{UR}, \text{RC}, \text{C}, \text{ND}\}\rangle$
- **Vettore Ambientale**: $\langle \text{CDP}: \{\text{N}, \text{L}, \text{LM}, \text{MH}, \text{H}, \text{ND}\}/\text{TD}: \{\text{N}, \text{L}, \text{M}, \text{H}, \text{ND}\}/\text{CR}: \{\text{L}, \text{M}, \text{H}, \text{ND}\}/\text{IR}: \{\text{L}, \text{M}, \text{H}, \text{ND}\}/\text{AR}: \{\text{L}, \text{M}, \text{H}, \text{ND}\}\rangle$

## 1.1 EQUAZIONI

### 1.1.1 Punteggio Base

$$\text{score}_{base}\left(\overline{\text{V}}, f(\text{I})\right) = \left\lceil \left(\frac{6 \cdot \text{I}}{10} + \frac{4 \cdot \text{E}}{10} - 1.5\right) \cdot f(\text{I}) \cdot 10 \right\rceil \cdot \frac{1}{10}$$

Con:

- $\text{I} = 10.41 \cdot \left(1 - (1 - \text{I}_\text{C}) \cdot (1 - \text{I}_\text{I}) \cdot (1 - \text{I}_\text{A})\right)$
- $\text{E} = 20 \cdot \text{AV} \cdot \text{AC} \cdot \text{AU}$
- $f(\text{I}) = \begin{cases} 0 & \text{se I} = 0 \\ 1.176 & \text{altrimenti} \end{cases}$
- $\text{AV} = \begin{cases} 0.35 & \text{se AV} = N \\ 0.61 & \text{se AV} = A \\ 0.71 & \text{se AV} = L \end{cases}$

- $\mathrm{AC} = \begin{cases} 0.35 & \text{se AC} = H \\ 0.61 & \text{se AC} = M \\ 0.71 & \text{se AC} = L \end{cases}$
- $\mathrm{AU} = \begin{cases} 0.45 & \text{se AU} = M \\ 0.56 & \text{se AU} = S \\ 0.704 & \text{se AU} = N \end{cases}$
- $\mathrm{I_C} = \begin{cases} 0.0 & \text{se } \mathrm{I_C} = N \\ 0.275 & \text{se } \mathrm{I_C} = P \\ 0.660 & \text{se } \mathrm{I_C} = C \end{cases}$
- $\mathrm{I_I} = \begin{cases} 0.0 & \text{se } \mathrm{I_I} = N \\ 0.275 & \text{se } \mathrm{I_I} = P \\ 0.660 & \text{se } \mathrm{I_I} = C \end{cases}$
- $\mathrm{I_A} = \begin{cases} 0.0 & \text{se } \mathrm{I_A} = N \\ 0.275 & \text{se } \mathrm{I_A} = P \\ 0.660 & \text{se } \mathrm{I_A} = C \end{cases}$

### 1.1.2 Punteggio Temporale

$$\mathrm{score}_{temporal}\left(\overline{\mathrm{V}}, f(\mathrm{I})\right) = \left\lceil \mathrm{score}_{base}\left(\overline{\mathrm{V}}, f(\mathrm{I})\right) \cdot \mathrm{E} \cdot \mathrm{RL} \cdot \mathrm{RC} \cdot 10 \right\rceil \cdot \frac{1}{10}$$

Con

- $f(\mathrm{I})$ del punteggio base
- $\mathrm{E} = \begin{cases} 0.85 & \text{se E} = U \\ 0.90 & \text{se E} = POC \\ 0.95 & \text{se E} = F \\ 1.00 & \text{se E} = H \\ 1.00 & \text{se E} = ND \end{cases}$
- $\mathrm{RL} = \begin{cases} 0.87 & \text{se RL} = OF \\ 0.90 & \text{se RL} = TF \\ 0.95 & \text{se RL} = W \\ 1.00 & \text{se RL} = U \\ 1.00 & \text{se RL} = ND \end{cases}$
- $\mathrm{RC} = \begin{cases} 0.90 & \text{se RC} = UC \\ 0.95 & \text{se RC} = UR \\ 1.00 & \text{se RC} = C \\ 1.00 & \text{se RC} = ND \end{cases}$

### 1.1.3 Punteggio Ambientale

$$\mathrm{score}_{temporal}(\overline{\mathrm{V}}) = \left\lceil \left(\mathrm{adj}_{temporal}(\overline{\mathrm{V}}) + \mathrm{CDP} \cdot \left(10 - \mathrm{adj}_{temporal}(\overline{\mathrm{V}})\right)\right) \cdot \mathrm{TD} \cdot 10 \right\rceil \cdot \frac{1}{10}$$

Con:

- $\mathrm{adj}_{temporal}(\overline{\mathrm{V}}) = \mathrm{score}_{temporal}\left(\overline{\mathrm{V}}, f'(\mathrm{I})\right)$

- $f'(\mathrm{I}) = \min\left(10 \quad 10.41 \cdot \left(1 - (1 - \mathrm{I_C R_C}) \cdot (1 - \mathrm{I_I R_I}) \cdot (1 - \mathrm{I_A R_A})\right)\right)$
- $\mathrm{CDP} = \begin{cases} 0.0 & \text{se CDP} = N \\ 0.1 & \text{se CDP} = L \\ 0.3 & \text{se CDP} = LM \\ 0.4 & \text{se CDP} = MH \\ 0.5 & \text{se CDP} = H \\ 0.0 & \text{se CDP} = ND \end{cases}$
- $\mathrm{TD} = \begin{cases} 0.00 & \text{se TD} = N \\ 0.25 & \text{se TD} = L \\ 0.75 & \text{se TD} = M \\ 1.00 & \text{se TD} = H \\ 1.00 & \text{se TD} = ND \end{cases}$
- $\mathrm{R_C} = \begin{cases} 0.50 & \text{se } \mathrm{R_C} = L \\ 1.00 & \text{se } \mathrm{R_C} = M \\ 1.51 & \text{se } \mathrm{R_C} = H \\ 1.00 & \text{se } \mathrm{R_C} = ND \end{cases}$
- $\mathrm{R_I} = \begin{cases} 0.50 & \text{se } \mathrm{R_I} = L \\ 1.00 & \text{se } \mathrm{R_I} = M \\ 1.51 & \text{se } \mathrm{R_I} = H \\ 1.00 & \text{se } \mathrm{R_I} = ND \end{cases}$
- $\mathrm{R_A} = \begin{cases} 0.50 & \text{se } \mathrm{R_A} = L \\ 1.00 & \text{se } \mathrm{R_A} = M \\ 1.51 & \text{se } \mathrm{R_A} = H \\ 1.00 & \text{se } \mathrm{R_A} = ND \end{cases}$

## 1.2 IMPLEMENTAZIONE

Dal momento che la maggior parte dei modelli trattati in questo studio utilizza le metriche CVSS per classificare le vulnerabilità, è stata creata, dall'autore di questa tesi, una libreria open source che ne svolge le funzionalità essenziali. Tale libreria è scritta in Java 8 ed è disponibile gratuitamente online all'indirizzo: https://github.com/jiraky90/CVSS.

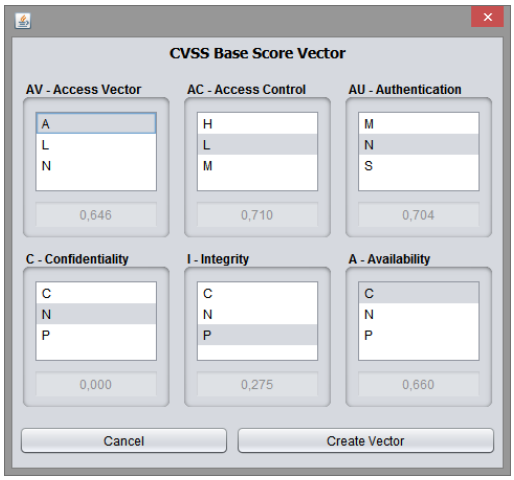



*Figura 16: Schermata di esempio per l'input del vettore base CVSS*

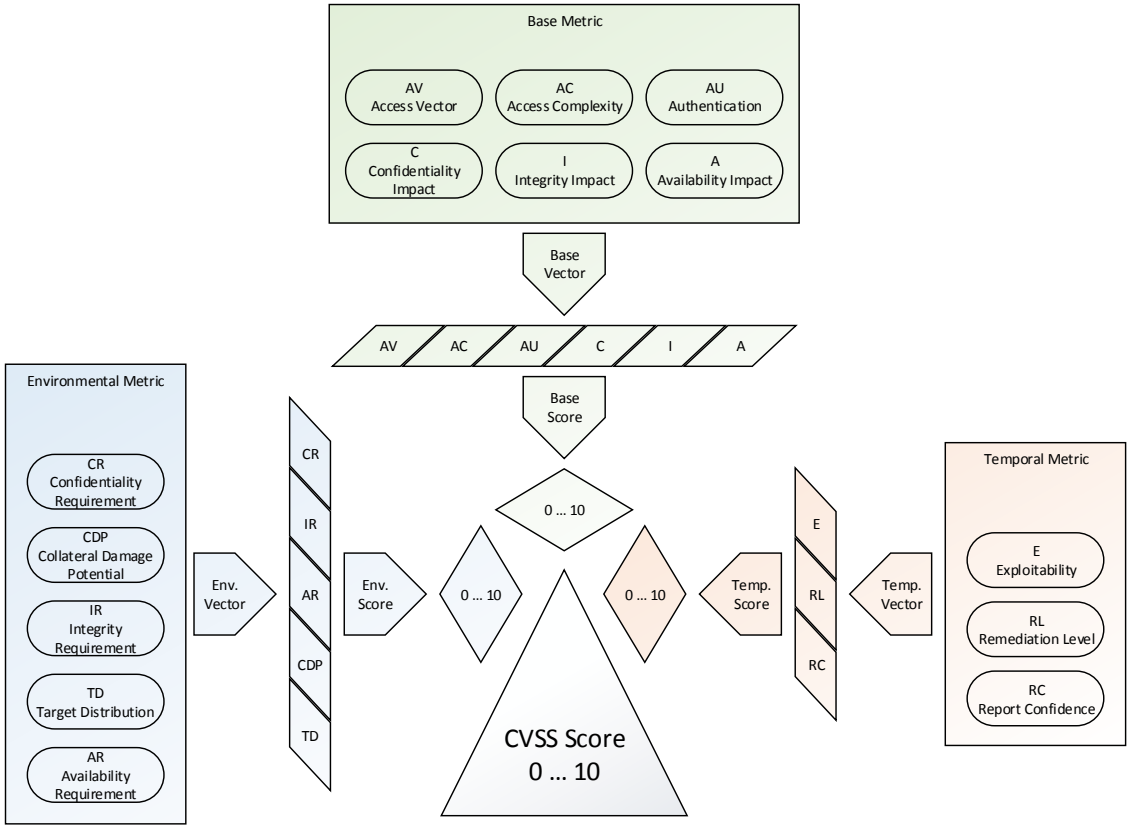



*Figura 17: Sistema di punteggio del CVSS*

# 2 RETI BAYESIANE

Le reti Bayesiane (BN, da Bayesian Network) sono un modello grafico per modellare sistemi ignoti o parzialmente ignoti. Sono rappresentate attraverso Grafi Diretti Aciclici (DAG), dove ogni nodo rappresenta una variabile (nel senso Bayesiano del termine, ovvero una quantità osservata, un'ipotesi oppure un parametro dal valore ignoto) ed ogni arco rappresenta una dipendenza condizionale tra le corrispondenti variabili. In ogni rete Bayesiana sono definite una o più radici e, partendo da esse, ogni nodo può essere via via raffinato fino ad ottenere il livello di dettaglio desiderato. Questo processo di raffinamento consiste nel suddividere il nodo padre in due o più nodi figli (sia utilizzando una congiunzione che una disgiunzione) e, ricorsivamente, consente di ripartire un obiettivo complesso in obiettivi più semplici. Il processo di raffinamento pone la condizione di mutua-esclusività e di esaustività sui domini di ogni variabile (ad esempio valori booleani, enumeratori come: *basso*, *medio*, *alto*, ecc.).

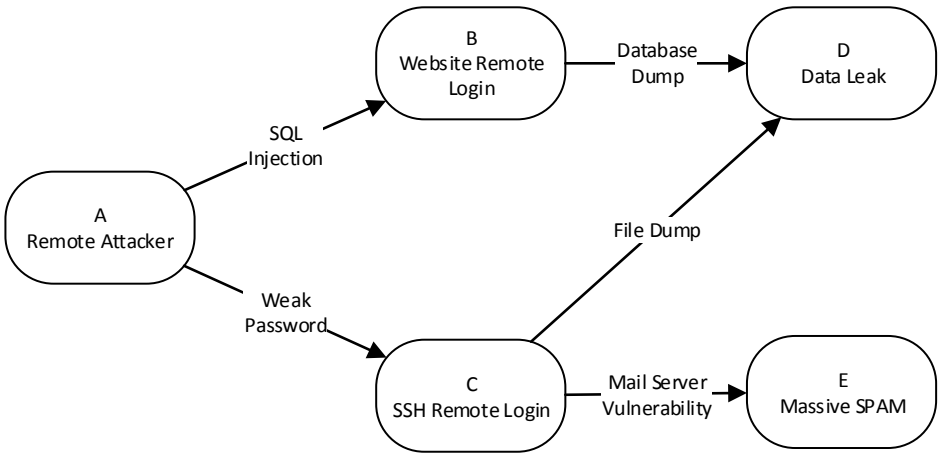



*Figura 18: Esempio di rete Bayesiana*

La Figura 18 espone un esempio di rete Bayesiana che rappresenta un possibile attacco. Nell'esempio sono stati definiti due obiettivi principali, **D** che rappresenta una fuga di notizie ed **E** che rappresenta l'invio di email di SPAM dal server. L'amministratore di rete che ha modellato questa struttura ha previsto che la perdita di dati possa avvenire solamente attraverso la compromissione del sito web (**B**) oppure del server dati (**C**). Quest'ultimo, se compromesso, potrebbe causare l'invio non autorizzato di email (**E**). In questo esempio l'unica probabilità esplicitamente indicata è quella relativa agli archi, che rappresentano la possibilità di sfruttare la corrispondente vulnerabilità.

Le reti Bayesiane, per lo meno nella versione originale, non sono in grado di modellare gli aspetti temporali che caratterizzano un attacco informatico. Per risolvere questo problema, negli anni '90 Dagum, Galper e Horvitz [58] combinarono cloni di una BN come se fossero istantanee temporali del modello. I modelli più semplici di DBN (dall'inglese "*Dynamic Bayesian Network*") sono chiamati "*Hidden Markov Models*" (HMM) [59], e vengono applicati negli IDS fin dal 2002 [60] per le loro abilità di modellazione di serie temporali di eventi. Gli HMM tuttavia non sono in grado di gestire correttamente le relazioni tra gli oggetti rappresentati, ma soprattutto l'utilizzo dell'inferenza statistica è difficile e poco performante [61].

In generale valgono due assiomi:

- Dato un nodo $x$, ogni padre di $x$ influenza indipendentemente dagli altri padri lo stato di $x$.
- Un attaccante non attaccherà due volte lo stesso nodo, ovvero non ritornerà mai su di un nodo già compromesso.

Il secondo assioma fornisce una regola analoga alla monotonicità descritta da Ammann *et al.* nel 2002 [12]. È ragionevole poiché l'attaccare un sistema già compromesso non fornisce alcuna informazione extra.

## 2.1 NOTAZIONE

Facendo riferimento all'esempio in Figura 18, ogni nodo ricopre un ruolo ben definito rispetto alla sua posizione relativa ed alle sue dipendenze:

- Il nodo **A** è definito come *antenato* dei nodi **B**, **C**, **D** ed **E**;
- Il nodo **A** è il *padre* dei nodi **B** e **C**;
- Il nodo **A** è la *radice* (nodo *esterno*) dell'albero;
- I nodi **B** e **C** sono nodi *figli* di **A** e nodi *padri* del nodo **D**;
- I nodi **B** e **C** sono nodi *interni* (nodi *intermedi*);
- I nodi **D** ed **E** sono *discendenti* dei nodi **A**, **B** e **C**;
- I nodi **D** ed **E** sono nodi *foglia* (nodi *terminali*).

## 2.2 ESEMPIO

Facendo riferimento all'esempio in Figura 18, dopo averne definito la struttura, l'amministratore di rete stabilirà le probabilità a *priori* per ogni nodo esterno e per ogni relazione:

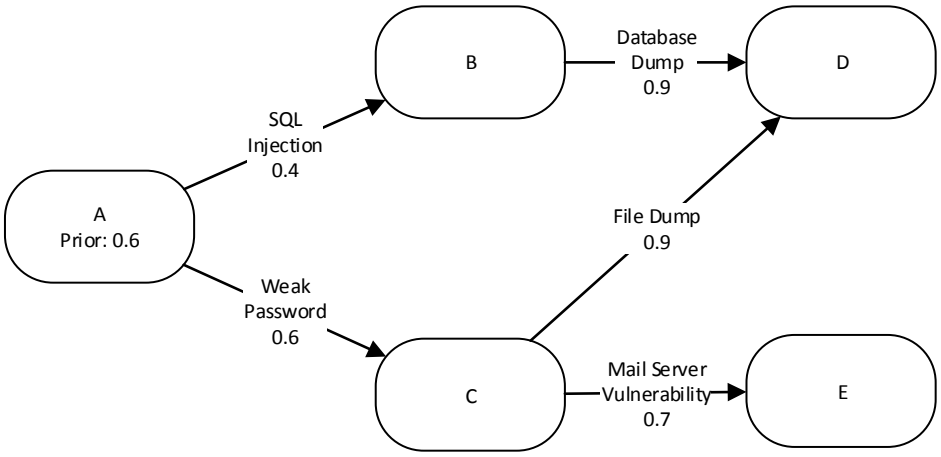



*Figura 19: Esempio di rete Bayesiana con probabilità a priori*

Una volta che la topologia e le probabilità a priori sono stabilite è possibile definire le probabilità condizionali (sotto forma di tabelle condizionali). Poi, supponendo di aver trovato le prove di un login sul server non autorizzato, è possibile aggiornare il grafo impostando per il nodo **C** la probabilità a posteriori uguale ad 1.0.

Il grafo aggiornato risulterà essere quello indicato in Figura 20, dove è possibile notare che la aumentata probabilità di compromissione del nodo **A** (da 0.6 a 1.0) è dovuta alle nuove informazioni sul nodo **C**.

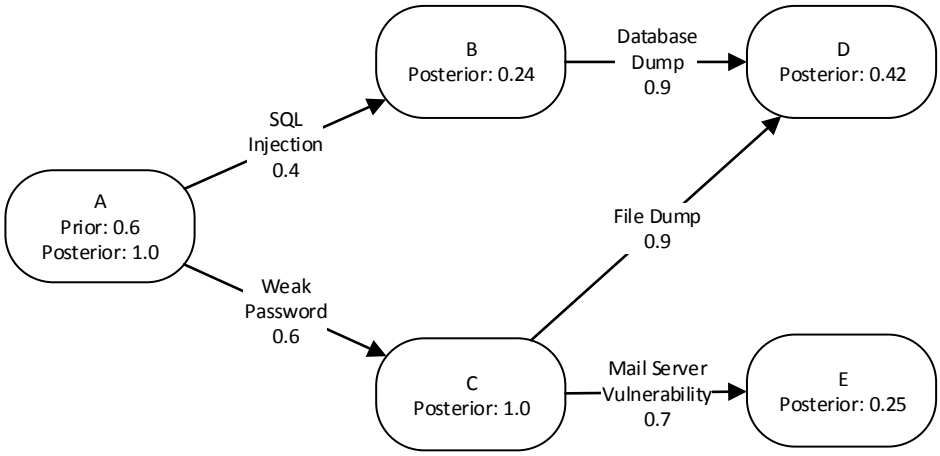



*Figura 20: Esempio di rete Bayesiana con probabilità a posteriori*

## 2.3 PROBABILITÀ

Ogni nodo è decorato con una o più probabilità, tipicamente si parla di *probabilità a Priori*, *probabilità a Posteriori* e *Likelihood*. Attraverso la loro struttura ben definita possono essere aggiornate in tempo reale ogni volta che la rete stessa viene modificata (ad esempio tramite l'inserimento o la cancellazione di vertici o relazioni, così come l'aggiunta di nuove evidenze o la disponibilità di nuove informazioni). Nella sopracitate Figura 19 e Figura 20 sono indicate alcune probabilità basilari delle reti Bayesiane.

Nelle reti Bayesiane la maggior parte delle probabilità fa riferimento a variabili piuttosto che ad eventi. Ogni variabile ha infatti associato un dominio di stati, che dovrebbe essere finito, esaustivo e mutualmente esclusivo. Questa condizione è sufficientemente forte da garantire che ogni variabile sia in uno ed un solo stato, anche se potendo essere lo stato incognito, la sua probabilità deve essere vincolata a quella di tutti gli altri stati. Più formalmente, dato un dominio $\mathfrak{X}$ di stati per la variabile $X$, ciascuno stato $x_1, \dots, x_n \in \mathfrak{X}$ avrà associata una probabilità $p_i$, e, dal momento che si tratta di stati mutualmente esclusivi ed esaustivi, la probabilità complessiva deve convergere ad 1.0:

$$\sum_i p_i = 1.0$$

### 2.3.1 Notazione

Questa tesi è fedele alla nomenclatura classica dove le lettere in maiuscolo $(A, B, X)$ rappresentano variabili e le lettere in minuscolo $(a, b, x)$ rappresentano il valore dello stato della corrispondente variabile (es. $x$ rappresenta lo stato $X = x$ se $X$ ha un dominio enumerativo, True se $A$ ha un dominio booleano).

Le lettere maiuscole doppie $(\mathbb{A}, \mathbb{B}, \mathbb{X})$ identificano insiemi di variabili (es. $A_1, \dots, A_n \in \mathbb{A}$).

Le lettere maiuscole gotiche $(\mathfrak{A}, \mathfrak{B}, \mathfrak{X})$ identificano i domini delle rispettive variabili (es. il dominio di $X$ è $\mathfrak{X}$).

Inoltre le lettere maiuscole in corsivo indicano formule di probabilità, ad esempio $\mathcal{P}(A)$ indica la probabilità a priori della variabile A.

### 2.3.2 Fondamenti di Probabilità

La definizione classica (Laplace, 1812) indica la probabilità dell'occorrenza di un evento A come il rapporto tra il numero di risultati favorevoli ed il numero di risultati possibili. Essa tuttavia si applica solamente a fenomeni con risultati equiprobabili e finiti, pertanto non è rilevante nel contesto di questo documento. La probabilità in questa tesi è quella definita assiomaticamente da Kolmogorov nel 1933, di seguito riportata.

Dato un qualsiasi esperimento casuale, i cui risultati possibili costituiscono gli elementi di un insieme non vuoto Ω (detto spazio campionario), la probabilità viene intesa come una funzione che associa ad ogni sottoinsieme di Ω un numero reale non negativo tale che la somma di tutte le parti (non sovrapposte) sia 1.0. La probabilità è pertanto un numero $P(E)$ tale che:

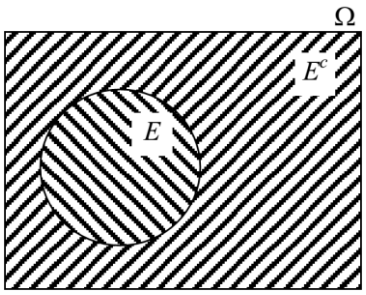



*Figura 21: Teorema degli eventi complementari*

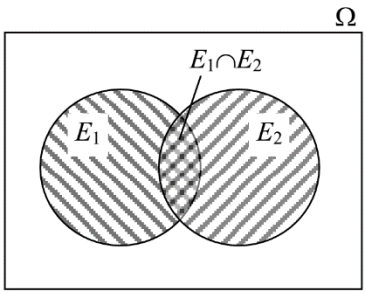



*Figura 22: Teorema dell'evento totale*

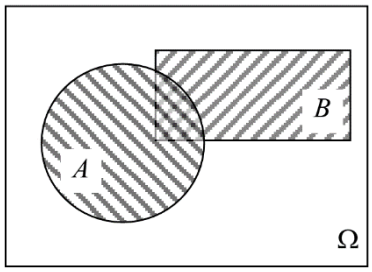



*Figura 23: Teorema della probabilità condizionata e composta*

1. $P(E) \geq 0.0$
2. $P(\Omega) = 1$
3. Dati due eventi mutualmente esclusivi $A$ e $B$, $P(A \cup B) = P(A) + P(B)$

Di seguito un elenco dei principali teoremi di probabilità classica. Alcuni di essi verranno ripresi in modo dettagliato nei paragrafi seguenti contestualizzandoli nell'ambito delle reti Bayesiane.

#### 2.3.2.1 Teoremi Fondamentali

##### 2.3.2.1.1 Teorema dell'evento complementare

Si definisce evento complementare

$$E^c = \Omega \backslash E$$

l'evento che comprende tutti i punti campionari di Ω non compresi in $E$ (Figura 21). Pertanto sussistono le seguenti proprietà:

- $E \cap E^c = \emptyset$
- $E \cup E^c = \Omega$
- $P(E^c) = 1 - P(E)$

##### 2.3.2.1.2 Teorema dell'evento totale

Il teorema dell'evento totale consente di calcolare la probabilità dell'unione di due sottoinsiemi di eventi, ovvero la probabilità che si verifichi un evento preso da uno qualsiasi dei sottoinsiemi. Dati due eventi $E_1, E_2 \subseteq \Omega$ (Figura 22):

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2)$$

Da cui segue che:

$$P(E_1 \cup E_2) \leq P(E_1) + P(E_2)$$

#### 2.3.2.1.3 Teorema della probabilità condizionata e composta

Si dice probabilità condizionata (o condizionale) di $A$ dato $B$, la probabilità che ha l'evento $A$ di verificarsi dopo aver saputo che l'evento $B$ si è verificato. La probabilità del verificarsi di $A$ si può interpretare come la probabilità di occorrenza di $A$ nello spazio campionario ridotto determinato da $B$ (Figura 23):

$$P(A|B) = P(A \cap B)/P(B)$$

### 2.3.2.2 Teorema delle probabilità totali (o marginalizzazione)

Nella teoria probabilistica questo teorema afferma che la probabilità complessiva di un evento può essere dipendente da molteplici eventi distinti. In altre parole risponde alle domande del tipo: "*dato un evento A e data una distribuzione di probabilità dell'evento B, qual è la probabilità complessiva di A?*".

Formalmente, sia **B** una variabile casuale (il cui dominio, definito come $\mathfrak{B} = \{b_1, \dots, b_n\}$, è finito, esaustivo e mutualmente esclusivo), per ogni possibile variabile **A** nello stesso spazio di probabilità, è definita la seguente regola:

$$\mathcal{M}_B(A) = \sum_{b_i \in \mathfrak{B}} P(A \cap b_i) = \sum_{b_i \in \mathfrak{B}} P(A|b_i) \cdot P(b_i)$$

### 2.3.2.3 Probabilità Condizionale

Il concetto più semplice della teoria Bayesiana è quello di *probabilità condizionale*, ovvero della probabilità che si verifichi un evento consequenzialmente ad un altro. In altre parole questa probabilità risponde alle domande del tipo "qual è la probabilità di **H**, sapendo che si è verificato l'evento **E**?". Essa è definita come $P(H|E) = p$, dove gli eventi **H** ed **E** sono chiamati rispettivamente *ipotesi* ed *evidenza* e $p$ è la probabilità che si verifichi l'ipotesi successivamente all'evento **E**. Formalmente:

$$P(H|E) = \frac{P(H \cap E)}{P(E)}$$

Dove $P(H \cap E)$ è la probabilità che si verifichino allo stesso tempo entrambi gli eventi, mentre $P(E)$ è la probabilità che si verifichi l'evento **E**. Quest'ultima è definita come la *probabilità a priori* (cfr. app. 2.3.3) di **E** nel caso in cui questo sia *esterno*, come la *probabilità marginale* (cfr. app. 2.3.2.2) altrimenti.

Condizionando **E** ad un altro evento F, la formula può essere riscritta in:

$$P(H|E \cap F) = \frac{P(H \cap E|F)}{P(E|F)}$$

La probabilità condizionale è applicabile in generale a tutte le coppie di eventi, tuttavia nel caso di *eventi indipendenti*[20] è uguale alla probabilità della sola ipotesi. Lo stesso ragionamento si applica a gruppi di eventi, ad esempio data una terna di eventi **H**, **E** ed **F**, se gli eventi **H** ed **E** sono *condizionalmente indipendenti* da $\mathbb{F}$ (ovvero l'evento **E** non cambia la probabilità dell'ipotesi dopo aver osservato $\mathbb{F}$) potremmo escludere E dalla formula:

$$P(H|E \cap \mathbb{F}) = P(H|\mathbb{F})$$

[20] In effetti l'indipendenza tra due eventi è un caso speciale dell'indipendenza condizionale, dove (come nel nostro caso) $\mathbb{F}$ è l'insieme vuoto.

La probabilità condizionale può essere applicata anche a variabili ed ad insiemi di variabili oltre che agli eventi. In questo specifico caso non si parla più di un numero che rappresenta questa probabilità, bensì di una *distribuzione di probabilità*.

Questa distribuzione è generalmente memorizzata come una tabella di probabilità condizionali, dall'inglese *Conditional Probability Table* (o *CPT*).

#### 2.3.2.4 Teorema di Bayes

Il teorema di Bayes descrive la probabilità di un evento basandosi sugli eventi precedenti che possono averlo influenzato.

Formalmente, dati due eventi **A** e **B**, la probabilità di "***A** dato **B***" è definita come:

$$P(A|B) = \frac{P(B|A) \cdot \mathcal{P}(A)}{P(B)}$$

Dove:

- $\mathcal{P}(A)$ è la *probabilità a priori* di **A**;
- $P(A|B)$ è la *probabilità a posteriori* di **A** dato **B**;
- $P(B|A)$ è la *Likelihood* di **B** dato **A**.

Nel caso di eventi multipli, ad esempio **B** e **C**:

$$P(A|B,C) = \frac{P(B|A,C) \cdot P(A|C)}{P(B|C)}$$

#### 2.3.2.5 Chain Rule

La Chain Rule è una regola di derivazione, che consente di calcolare la probabilità composta di un insieme di variabili utilizzando solamente le probabilità condizionali.

Dato un insieme di variabili $A_1, \dots, A_n$:

$$P\left(\bigcup_{i=1}^{n} A_i\right) = \prod_{i=1}^{n} P\left(A_i \middle| \bigcap_{j=1}^{i-1} A_j\right)$$

Con quatto variabili, la regola produce la seguente probabilità:

$$P(A,B,C,D) = P(D|C,B,A) \cdot P(C|B,A) \cdot P(B|A) \cdot P(A)$$

### 2.3.3 Probabilità a Priori

La probabilità a priori esprime l'incertezza di un evento (o di una variabile) prima che ulteriori dati vengano presi in considerazione. Generalmente si hanno due categorie di probabilità a priori: le distribuzioni informative e le distribuzioni non informative.

Una distribuzione a priori informativa esprime un'informazione circa una variabile, ad esempio un valore atteso o una previsione. Una distribuzione a priori non informativa, invece, esprime incertezza (il principio di indifferenza [21] ne è l'esempio più semplice) tuttavia generalmente queste probabilità richiedono l'intervento esterno di un esperto.

[21] Il *principio di indifferenza* stabilisce che se esistono $n$ eventi mutualmente esclusivi che rappresentano esaustivamente ogni possibile evento, e se questi eventi sono indistinguibili (eccezion fatta per il nome) allora ad ognuno di essi sarà assegnata la probabilità $\frac{1}{n}$.

### 2.3.4 Probabilità Congiunta

La probabilità congiunta (JPT, dall'inglese *Joint Probability Function*), è una distribuzione multivariata di $k$ variabili casuali che descrive la probabilità che ogni evento si verifichi allo stesso tempo. Informalmente, risponde a domande del tipo "*qual è la probabilità che* $\mathrm{X} = \mathrm{True}$ *sia vero, dati* $\mathrm{Y} = \mathrm{True}$ *e* $\mathrm{Z} = \mathrm{False}$*?*".

Data una variabile $V$, ed il suo stato previsto $v$, dato un insieme di variabili $\mathbb{X} = X_1, \dots, X_n$ e dato l'insieme dei loro stati previsti $\mathbb{x} = x_1, \dots, x_n$, la probabilità congiunta si esprime come:

$$\mathrm{JPT}(v, x_1, \dots, x_n) = P(V = v, X_1 = x_1, \dots, X_n = x_n)$$

Nel caso discreto, è possibile espandere questa probabilità utilizzando la regola della catena (cfr. app. 2.3.2.5):

$$P(v) \cdot P(x_1|v) \cdot P(x_2|x_1, v) \cdot \dots \cdot P(x_n|x_1, \dots, x_{n-1}, v) = \mathcal{P}(v) \cdot \prod_{i=1}^{n} P\left(x_i \middle| \bigcap_{j=1}^{i-1} x_j\right)$$

Riformulando in termini di *Likelihood*:

$$\mathrm{JPT}(v, x_1, \dots, x_n) = \mathcal{P}(V = v) \cdot \mathcal{L}(v|X_1) \cdot \dots \cdot \mathcal{L}(v, X_1, \dots, X_{n-1}|X_n)$$

Dato un nodo $n$, dal momento che ogni altro nodo non antenato di $n$ gli è condizionalmente indipendente, la probabilità congiunta può essere espressa in termini ricorsivi:

$$\mathrm{JPT}(n) = P(n|\,\mathrm{pa}(n))$$

Dove $\mathrm{pa}(n)$ è l'insieme dei nodi *padri* di $n$.

### 2.3.5 Likelihood

La Likelihood è una funzione utilizzata nei processi di inferenza statistica, come ad esempio il calcolo della probabilità a posteriori. Una probabilità è utilizzata per descrivere un evento dato un parametro fissato, mentre con Likelihood si intende descrivere la funzione di un parametro dato l'evento. Ad esempio, data una moneta ed una serie di lanci, qual è la probabilità di ottenere testa 5 volte? Data una serie di 10 lanci, i cui risultati contano 5 volte testa e 5 volte croce, possiamo definire la moneta *verosimilmente* equa (Likelihood).

La differenza tra probabilità e Likelihood è quindi ben precisa in campo statistico, tuttavia una distribuzione di probabilità potrebbe essere interpretata in entrambi i modi: dato il parametro fissato qual è il valore? Inoltre, dato il valore fissato, qual è il parametro?

Formalmente è una funzione:

$$\mathcal{L}(b|A): b \mapsto P(A|B = b)$$

Questi valori sono generalmente definiti dall'amministratore del sistema, o da colui che definisce il modello, e vengono specificati in forma di tabella *CPT* (cfr. app. 2.3.2.3).

### 2.3.6 Probabilità a Posteriori

La probabilità a posteriori è la probabilità condizionale di una variabile data un'informazione che ne modifichi il comportamento. Essa è direttamente proporzionale alla *probabilità a priori* per l'*impatto*, ovvero la probabilità di un evento dati i suoi parametri, formalmente, date due variabili chiamate *ipotesi* (**H**) ed *evidenza* (**E**):

$$P(H|E) = \mathcal{P}(H) \cdot \mathcal{I}(E, H) = \frac{\mathcal{P}(H) \cdot P(E|H)}{\mathcal{M}_H(E)} = \frac{\mathrm{JPT}(H, E)}{\mathcal{M}_H(E)}$$

Nel caso di evidenze multiple ($\forall e \in \mathbb{E}$), essa è definita come:

$$P(H|\mathbb{E}) = \mathcal{P}(H) \cdot \prod_{e \in \mathbb{E}} \mathcal{I}(e,H) = \frac{\mathcal{P}(H) \cdot \prod_{e \in \mathbb{E}} P(e|H)}{\prod_{e \in \mathbb{E}} \mathcal{M}_H(e)}$$

L'impatto è quindi definito come l'indice $\mathcal{I}$:

$$\mathcal{I}(E,H) = \frac{P(E|H)}{\mathcal{M}_H(E)}$$

Segue che se l'indice di impatto è pari a 1.0 allora l'evidenza è indipendente dal modello esaminato.

Il calcolo della probabilità a posteriori, definito anche come Inferenza Bayesiana è NP-Hard nel caso esatto. Tuttavia esistono molteplici algoritmi in grado di raggiungere un buon compromesso tra accuratezza, generalità e complessità. L'esempio principe di tali algoritmi è il "Bucket Elimination" (BE), proposto da Dechter nel 1998 [62]. Il BE ha complessità esponenziale (sia in tempo che in spazio) rispetto al peso del grafo indotto minimo, Le reti Bayesiane, tuttavia, possono essere frammentate in sotto reti riducendo di molto l'esponente durante la valutazione della probabilità a priori. Questo raggruppamento può essere basato sia su similarità tra i sistemi sia sulle policy di sicurezza applicate, oppure essere un processo completamente casuale.

# 3 OTTIMIZZAZIONE

La ricerca e l'ottimizzazione di contromisure è un problema che consiste nella ricerca di punti stazionari[22], ovvero i punti di minimo[23] e di massimo[24] (locali o assoluti) ed i punti di sella[25].

Lo scopo di un algoritmo di ottimizzazione è quello di trovare (per qualche istanza $x$) una soluzione ottimale $y$ tale che:

$$m(x, y) = g\{m(x, y')|y' \in f(x)\}$$

Nel caso specifico il problema consiste nel trovare un insieme di contromisure da associare all'insieme di vulnerabilità dato in input con l'obiettivo di minimizzare i residui[26] e minimizzare il costo complessivo delle misure di sicurezza applicate (problema NP-Hard [63]).

La selezione di contromisure è un esempio di ottimizzazione multi-obiettivo in quanto non esiste una soluzione ottima comune a tutti gli interessi in atto (ad esempio massima efficienza delle contromisure vs minor costo). Il processo decisionale richiede quindi l'intervento umano per vagliare le proposte che ottimizzano i singoli obiettivi. Possiamo definire il problema come una situazione in cui la riallocazione delle risorse che migliora la condizione di una parte del sistema (l'attivazione di una contromisura aumenta la sicurezza e diminuisce il rischio) peggiora necessariamente la condizione di un'altra parte del sistema (l'attivazione di una contromisura aumenta il costo complessivo, peggiorando l'incidenza sul budget). L'insieme delle soluzioni ottime viene quindi definito "*Fronte di Pareto"* e consiste nell'insieme dei *punti non dominati*[27].

Nell'ambito della sicurezza informatica il fronte di Pareto risolve il problema chiamato "*Dilemma dell'Amministratore*" (cfr. 2.1.5).

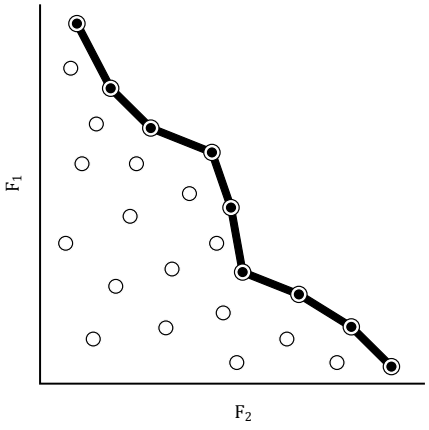



*Figura 24: Esempio di un Fronte di Pareto*

[22] Punto il cui piano tangente è orizzontale.
[23] Punto stazionario con la particolarità di avere nel suo intorno solo punti ad una quota superiore.
[24] Punto stazionario con la particolarità di avere nel suo intorno solo punti ad una quota inferiore.
[25] Punto stazionario non rappresentante né un minimo né un massimo.
[26] Un residuo è una vulnerabilità non coperta da una contromisura.
[27] Un punto non dominante è un punto per il quale non esiste nessun punto che sia migliore contemporaneamente per tutti gli obiettivi considerati nella funzione di ottimizzazione.

In letteratura esistono differenti proposte atte a risolvere il problema che prevedono combinazioni di operazioni aritmetiche [64] (è stato dimostrato [64] che se gli obbiettivi possono essere combinati tra di loro, questo approccio è il più efficiente nonché il più semplice realizzabile), combinazioni pesate di funzioni diverse [65] e marcatura di obiettivi specifici. Gli algoritmi evolutivi si prestano ad essere un ottimo framework di valutazione per ciascuno di questi approcci [66].

Il metodo classico consiste nell'assegnare un vettore di pesi ai vari obiettivi (codificando in esso le preferenze per la funzione di ottimizzazione) riducendo così la complessità a quella di un problema di ottimizzazione singola. Tale meccanismo però non consente una visione globale complessiva del modello e non consente di effettuare decisioni drastiche *dopo* aver compiuto il processo di ottimizzazione.

## 3.1 ALGORITMI EVOLUTIVI

Una delle tecniche più utilizzate attualmente prevede di servirsi degli algoritmi evolutivi (EMO[28]) poiché trattano contemporaneamente molteplici soluzioni (popolazione) convergendo globalmente alle soluzioni sul fronte di Pareto. Gli EMO però non forniscono alcuna garanzia circa l'effettiva terminazione in tempi "ragionevoli" poiché la computazione risulta essere molto elaborata ed estremamente legata all'inizializzazione dei parametri.

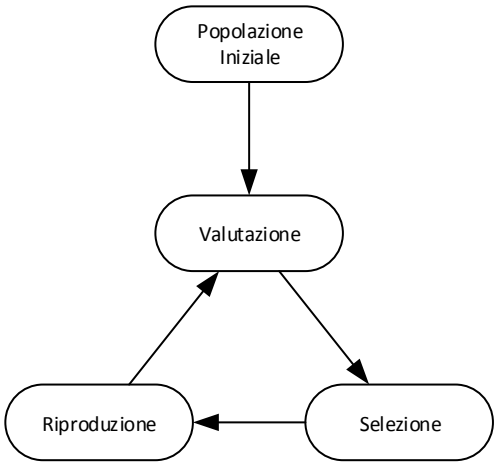



*Figura 25: Schema generico di un algoritmo evolutivo*

In letteratura sono molteplici gli algoritmi utilizzati seguendo questa filosofia [64, 67], in particolare il "*Non-dominated sorting genetic algorithm*" (o NSGA-II) [68] che deve la sua fama all'efficienza in termini di convergenza e di diversità di soluzioni ottenute, nonché all'utilizzo estensivo in applicazioni su problemi concreti.

### 3.1.1 Algoritmi Genetici

L'approccio genetico rientra nella categoria degli algoritmi evolutivi e, pertanto risolve problemi di ottimizzazione e di ricerca riproducendo il processo evolutivo naturale. Partendo da una *popolazione* di *individui* che rappresenta le possibili soluzioni del problema un algoritmo genetico produce nuove *generazioni* che mirano ad ottenere individui migliori rispetto alla generazione precedente. Un individuo è migliore di un altro se la sua funzione di *fitness* è

[28] "*Evolutionary algorithms for Multi-Objective Optimization*".

migliore, ovvero se l'adattamento (dall'inglese "*fit*") è migliore rispetto all'ambiente (la funzione di fitness da ottimizzare può anche rappresentare la funzione obiettivo o una sua trasformata). Generalmente un algoritmo genetico non trova una soluzione ottima globale tuttavia è in grado di trovare molteplici soluzioni buone in tempi ragionevoli.

Lo schema generale di un algoritmo evolutivo è il seguente:

```
1. Inizializza la popolazione S = S_0
2. F = fit(S)
3. While (F < σ) do
4.  Estrai un sottoinsieme S' ⊆ S
5.  Espandi S'
6. Done
7. Output S ⇒ fit(S) ≥ σ
```

Gli algoritmi genetici sono una classe di algoritmi evolutivi dove la funzione di espansione è sostituita dalle funzioni di mutazione (ovvero la modifica casuale dei geni di uno o più individui) e di ricombinazione (ovvero lo scambio di geni tra due individui).

La ricombinazione ("*crossover*"), inoltre, consente di mantenere una soluzione "buona" ed al tempo stesso esplorare nuove possibili soluzioni relativamente vicine a quella originale.

Dati due individui $X$ ed $Y$, possiamo rappresentare ogni genoma come una stringa $x_1, \dots, x_n$, e, dato un numero casuale $0 < k \leq n$, definiamo i due individui ricombinati come:

$$X' = x_1, \dots, x_{k-1}, y_k, \dots, y_n$$

$$Y' = y_1, \dots, y_{k-1}, x_k, \dots, x_n$$

Graficamente:

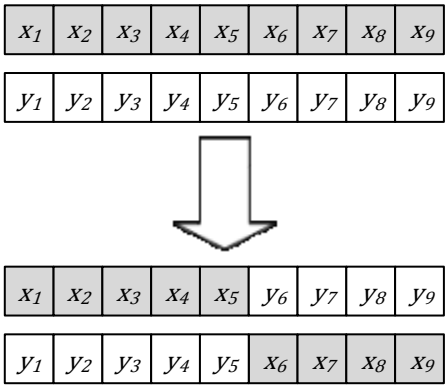



*Figura 26: Esempio di ricombinazione*

La scelta di una o più funzioni di fitness è un passo cruciale nel design di un algoritmo genetico.

Gli algoritmi genetici sono efficienti ed altamente parallelizzabili, un esempio è fornito da Ma e Sun in [69], qui rappresentato in Figura 27.

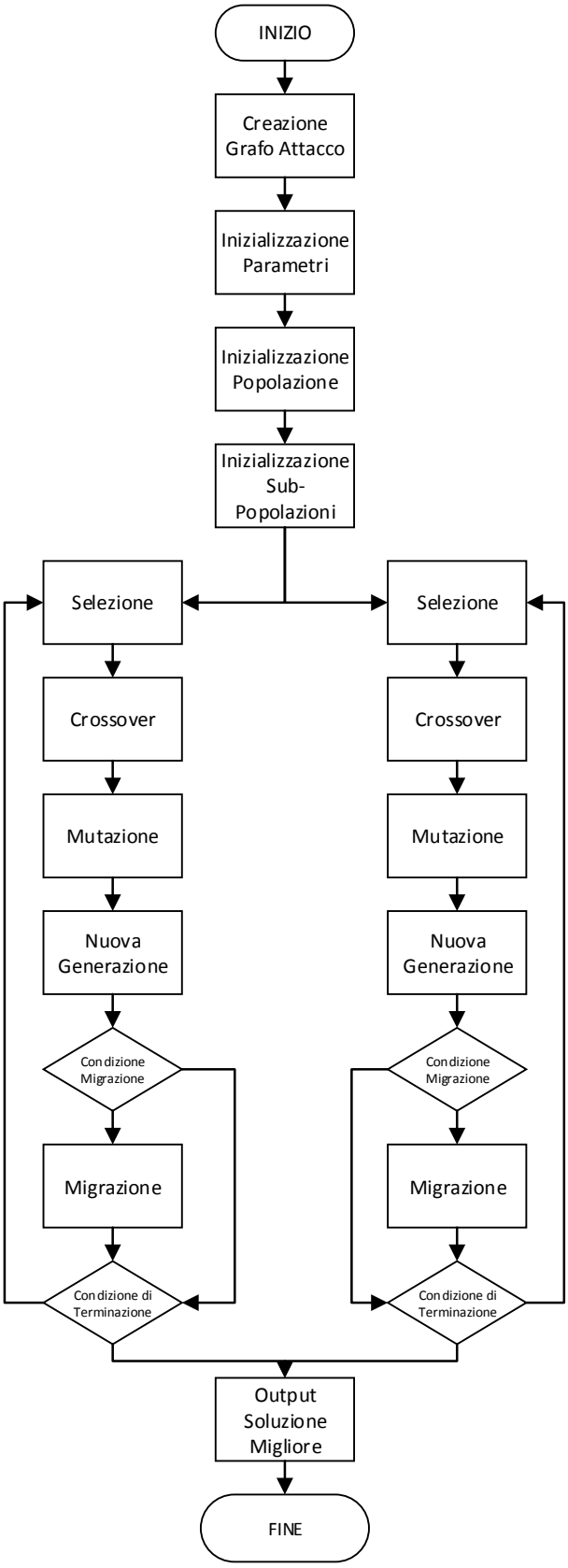



*Figura 27: Esempio di algoritmo genetico parallelo*

### 3.1.2 Ant Colony Optimization

Dorigo et al. [70] svilupparono questa particolare tecnica di ottimizzazione basata sullo studio delle colonie di formiche. L'algoritmo risolve problemi di ricerca di un percorso minimo su grafi generici basandosi sul concetto di modifica ambientale come mezzo di comunicazione. Nel modello i nodi rappresentano gli stati mentre gli archi rappresentano le possibili transizioni. Ogni "formica" è identificata dallo stato iniziale, da uno stato finale e da una memoria per tenere traccia delle decisioni prese ad ogni nodo. Ad ogni nodo viene deciso l'arco uscente da seguire secondo una funzione probabilistica condizionata dal numero di formiche che già hanno intrapreso quel percorso e dalla compatibilità con la destinazione della formica.

L'approccio Ant Colony Optimization è utilizzato in alcuni modelli basati sulle metriche di cammino minimo [71, 72].

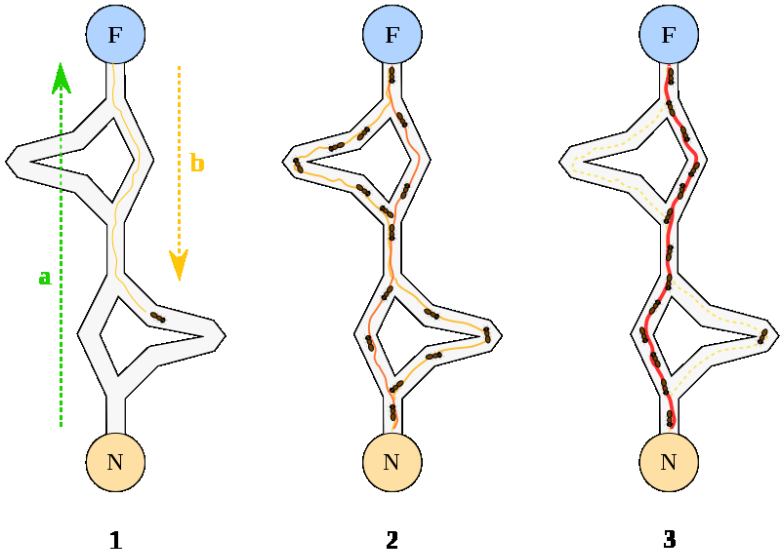



*Figura 28: Esempio di ricerca del cammino più breve da parte di una colonia di formiche*

In Figura 28 è mostrato un esempio di ottimizzazione a cura di Johann Dréo[29]. La prima fase consiste nella ricerca di una fonte di cibo "**F**" utilizzando un percorso casuale "**a**" rilasciando una traccia di ferormoni[30] lungo il percorso. Nella seconda fase le formiche seguono uno qualsiasi dei quattro possibili percorsi, tuttavia il rilascio di ferormoni da ognuna di esse renderà il percorso ottimale più appetibile alle altre formiche. Infine, nella terza fase, le formiche seguono esclusivamente il percorso ottimale, poiché i ferormoni rilasciati sugli altri percorsi sono evaporati.

[29] "*Shortest path find by an ant colony*" © 2006, Johann Dréo @ Wikimedia Commons

[30] Un ferormone è un marcatore chimico percepito da molti animali in natura, è utilizzato dalle formiche per tracciare il percorso seguito.

### 3.1.3 Particle Swarm Optimization

A differenza dei classici algoritmi evolutivi il PSO non si basa sulla selezione; tipicamente infatti, tutti i membri della popolazione sopravvivono fino alla fine del processo di ottimizzazione [73]. Sviluppato nei primi anni ’90 da Kennedy ed Eberhart [74], questa tecnica deriva dall’analisi dei meccanismi di interazione all’interno di un gruppo quando gli individui sono uniti da un obiettivo comune. Il comportamento dello sciame (swarm) è complesso poiché ogni individuo è libero di muoversi come meglio crede, autolimitandosi in base a criteri di separazione (repulsione a corto raggio tra due individui), allineamento (l’individuo si dirige nella stessa direzione dello sciame) e coesione (attrazione a lungo raggio tra l’individuo e lo sciame). L’algoritmo bilancia perfettamente le fasi di esplorazione e di sfruttamento proprie degli algoritmi evolutivi tuttavia essendo una tecnica di ottimizzazione prettamente numerica non ha riscontrato successo tra i ricercatori nell’ambito della sicurezza informatica.

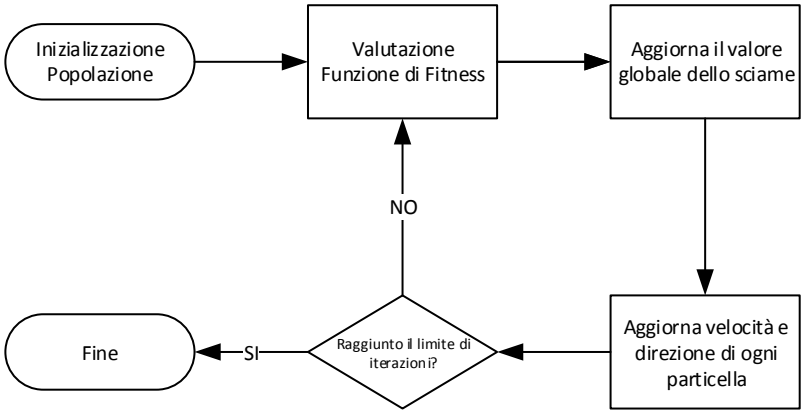



*Figura 29: Esempio di algoritmo di ottimizzazione Particle Swarm*

# Bibliografia

## 1 FONTI

[1] S. Cheung, U. Lindqvist e M. W. Fong, «Modeling multistep cyber attacks for scenario recognition,» in *Proceedings of DARPA Information Survivability Conference and Exposition, 2003*, 2003.

[2] J. Dawkins e J. Hale, «A systematic approach to multi-stage network attack analysis,» in *Second IEEE International Information Assurance Workshop*, 2004.

[3] S. Noel, S. Jajodia, B. O'Berry e M. Jacobs, «Efficient minimum-cost network hardening via exploit dependency graphs,» in *19th Annual Computer Security Applications Conference, 2003.*, 2003.

[4] X. Ou, S. Govindavajhala e A. W. Appel, «MulVAL: A Logic-based Network Security Analyzer,» in *Proceedings of the 14th conference on USENIX Security Symposium*, 2005.

[5] F. Cuppens e A. Miege, «Alert correlation in a cooperative intrusion detection framework,» in *IEEE Symposium on Security and Privacy, 2002.* , 2002.

[6] B. Morin, L. Mé, H. Debar e M. Ducassé, «M2D2: A formal data model for IDS alert correlation,» in *Recent Advances in Intrusion Detection*, Springer Berlin Heidelberg, 2002, pp. 115-137.

[7] P. Ning, Y. Cui, D. S. Reeves e D. Xu, «Techniques and tools for analyzing intrusion alerts.,» *ACM Transactions on Information and System Security (TISSEC),* vol. 7, n. 2, pp. 274-318, 2004.

[8] S. Noel, E. Robertson e S. Jajodia, «Correlating intrusion events and building attack scenarios through attack graph distances.,» in *20th Annual Computer Security Applications Conference, 2004.*, 2004.

[9] F. Valeur, G. Vigna, C. Kruegel e R. A. Kemmerer, «Comprehensive approach to intrusion detection alert correlation.,» *IEEE Transactions on Dependable and Secure Computing,* vol. 1, n. 3, pp. 146 - 169, 2004.

[10] S. Axelsson, «A preliminary attempt to apply detection and estimation theory to intrusion detection,» 2000.

[11] P. Helman e G. Liepins, «Statistical foundations of audit trail analysis for the detection of computer misuse,» *IEEE Transactions on Software Engineering,* vol. 19, n. 9, pp. 886 - 901 , 1993.

[12] P. Ammann, D. Wijesekera e S. Kaushik, «Scalable, graph-based network vulnerability analysis.,» in *Proceedings of the 9th ACM Conference on Computer and Communications Security.*, 2002.

[13] K. Ingols, R. Lippmann e K. Piwowarski, «Practical attack graph generation,» in *22nd Annual Computer Security Applications Conference, 2006. ACSAC '06.*, 2006.

[14] S. Jajodia, S. Noel e B. O'Berry, «Topological analysis of network attack vulnerability.,» in *Managing Cyber Threats: Issues, Approaches, and Challenges.*, Springer, 2005, pp. 247-266.

[15] X. Ou, W. F. Boyer e M. A. McQueen, «A scalable approach to attack graph generation,» in *Proceedings of the 13th ACM conference on Computer and communications security* , 2006.

[16] O. Sheyner, J. Haines, S. Jha, R. Lippmann e J. M. Wing, «Automated generation and analysis of attack graphs,» in *IEEE Symposium on Security and Privacy, 2002. Proceedings. 2002*, 2002.

[17] M. Zago, M. Gil Perez, G. Martinez Pérez e J. J. Andreu Blazquez, «Multigraph Project: First steps towards the definition of a multiple attack graph model simulator,» in *JNIC - Jornadas Nacionales de investigación en Ciberseguridad*, Leon (ES), 2015.

[18] R. Dantu, K. Loper e P. Kolan, «Risk management using behavior based attack graphs.,» in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, 2004.

[19] X. Qin e W. Lee, «Attack plan recognition and prediction using causal networks.,» in *Computer Security Applications Conference, 2004. 20th Annual.*, 2004.

[20] R. Dantu e P. Kolan, «Risk management using behavior based bayesian networks.,» in *Proceeding of IEEE International Conference on Intelligence and Security Informatics, ISI 2005.*, 2005.

[21] Y. Liu and H. Man, "Network vulnerability assessment using Bayesian networks.," *Proceedings of SPIE Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2005.,* vol. Vol. 5812, pp. 61-71, 2005.

[22] X. An, D. Jutla e N. Cercone, «Privacy intrusion detection using dynamic Bayesian networks.,» in *ACM International Conference Proceeding Series.*, 2006.

[23] N. Poolsappasit, R. Dewri e I. Ray, «Dynamic security risk management using bayesian attack graphs,» *IEEE Transactions on Dependable and Secure Computing,* vol. 9, n. 1, pp. 61-74, 2012.

[24] M. Frigault, A. Singhal, S. Jajodia e L. Wang, «Measuring network security using dynamic Bayesian network,» in *Proceedings of the 4th ACM Workshop on Quality of Protection*, 2008.

[25] M. Frigault e L. Wang, «Measuring Network Security Using Bayesian Network-Based Attack Graphs,» in *The 32nd Annual IEEE International Conference on Computer Software and Applications (COMPSAC 08)*, 2008.

[26] Q. Althebyan e B. Panda, «A knowledge-based Bayesian model for analyzing a system after an insider attack.,» in *Proceedings of The IFIP TC 11 23rd International Information Security Conference*, 2008.

[27] R. Dewri, N. Poolsappasit, I. Ray e D. Whitley, «Optimal security hardening using multi-objective optimization on attack tree models of networks,» in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007.

[28] R. Dantu, P. Kolan, R. Akl e K. Loper, «Classification of attributes and behavior in risk management using bayesian networks,» in *Intelligence and Security Informatics*, 2007.

[29] R. Dantu, P. Kolan e J. Cangussu, «Network risk management using attacker profiling,» *Security and Communication Networks,* vol. 2, n. 1, pp. 83-96, January/Febraury 2009.

[30] P. Xie, J. H. LI, X. Ou, P. Liu e R. Levy, «Using Bayesian networks for cyber security analysis,» in *International Conference on Dependable Systems and Networks (DSN), 2010 IEEE/IFIP*, 2010.

[31] T. Sommestad, M. Ekstedt e P. Johnson, «Combining defense graphs and enterprise architecture models for security analysis.,» in *12th International IEEE Enterprise Distributed Object Computing Conference, 2008. EDOC '08.*, 2008.

[32] U. Franke, T. Sommestad, M. Ekstedt e P. Johnson, «Defense graphs and enterprise architecture for information assurance analysis,» Royal Inst. of Tech. Stockholm (Sweden), 2008.

[33] T. Sommestad, M. Ekstedt e P. Johnson, «Cyber security risks assessment with bayesian defense graphs and architectural models,» in *International Conference on System Sciences, 2009. HICSS '09.*, 2009.

[34] M. Ekstedt e T. Sommestad, «Enterprise architecture models for cyber security analysis,» in *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, 2009.

[35] T. Sommestad, M. Ekstedt e L. Nordström, «Modeling security of power communication systems using defense graphs and influence diagrams,» *IEEE Transactions on Power Delivery,* vol. 24, n. 4, pp. 1801-1808, 2009.

[36] J. Pamula, S. Jajodia, P. Ammann e V. Swarup, «A weakest-adversary security metric for network configuration security analysis,» in *Proceedings of the 2nd ACM workshop on Quality of protection* , 2006.

[37] L. Wang, A. Liu e S. Jajodia, «An Efficient and Unified Approach to Correlating, Hypothesizing, and Predicting Intrusion Alerts,» in *10th European Symposium on Research in Computer Security*, 2005.

[38] L. Wang, S. Noel e S. Jajodia, «Minimum-cost network hardening using attack graphs,» *Computer Communications,* vol. 29, n. 18, p. 3812–3824, 2006.

[39] S. Noel, M. Elder, S. Jajodia, P. Kalapa, S. O'Hare e K. Prole, «Advances in topological vulnerability analysis,» in *Proceedings of the 2009 Cybersecurity Applications & Tec hnology Conference for Homeland Security. CATCH '09*, 2009.

[40] L. Wang, A. Singhal e S. Jajodia, «Measuring network security using attack graphs,» in *Proceedings of the 3rd ACM workshop on Quality of protection (QoP'07)*, 2007.

[41] L. Wang, G. M. University, C. Yao, A. Singhal e S. Jajodia, «Interactive Analysis of Attack Graphs Using Relational Queries,» in *20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2006.

[42] L. Wang, T. Islam, A. Singhal e S. Jajodia, «An attack graph-based probabilistic security metric,» in *Proceeedings of the 22nd Annual IFIP WG 11.3 Working Conf erence on Data and Applications Security (DAS'2008)*, 2008.

[43] L. Wang, A. Singhal e S. Jajodia, «Measuring the Overall Security of Network Configurations Using Attack Graphs,» in *Data and Applications Security XXI*, Springer Berlin Heidelberg, 2007, pp. 98-112.

[44] M. Howard, J. Pincus e J. M. Wing, Measuring relative attack surfaces, Springer US, 2005.

[45] P. Manadhata e J. M. Wing, «Measuring a System's Attack Surface,» CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 2004.

[46] P. Manadhata, J. Wing, M. Flynn e M. McQueen, «Measuring the attack surfaces of two FTP daemons,» in *Proceedings of the 2nd ACM workshop on Quality of protection*, 2006.

[47] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz e Cunningham, «Validating and restoring defense in depth using attack graphs,» in *IEEE Military Communications Conference, 2006. MILCOM 2006.*, 2006.

[48] N. Idika e B. Bhargava, «Extending attack graph-based security metrics and aggregating their application,» *IEEE Transactions on Dependable and Secure Computing,* vol. 9, n. 1, pp. 75 - 85, 01 2012.

[49] P. Cheng, L. Wang, S. Jajodia e A. Singhal, «Aggregating CVSS Base Scores for Semantics-Rich Network Security Metrics,» in *IEEE 31st Symposium on Reliable Distributed Systems (SRDS), 2012* , 2012.

[50] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan e A. Singhal, «Aggregating vulnerability metrics in enterprise networks using attack graphs,» *Journal of Computer Security,* vol. 21, n. 4, pp. 561-597 , 2013.

[51] alderg e davidjgraph, «JGraphX Library,» [Online]. Available: https://github.com/jgraph/jgraphx. [Consultato il giorno 01 12 2014].

[52] S. Noel, S. Jajodia, L. Wang e A. Singhal, «Measuring Security Risk of Networks Using Attack Graphs,» *International Journal of Next-Generation Computing,,* vol. Vol. 1, n. 1, pp. 135-147, 2010.

[53] N. Feng e J. Xie, «A Bayesian networks-based security risk analysis model for information systems integrating the observedcases with expert experience,» *Scientific Research and Essays,* vol. 7, n. 10, pp. 1103-1112, 2012.

[54] «GeNIe & SMILE,» Decision Systems Laboratory, 2013. [Online]. Available: https://dslpitt.org/genie/. [Consultato il giorno 2014 01 01].

[55] M. Schiffman, G. Eschelbeck, D. Ahmad, A. Wright e S. Romanosky, «CVSS: A Common Vulnerability Scoring System,» National Infrastructure Advisory Council (NIAC), 2004.

[56] J. T. Chambers e J. W. Thompson, «Common Vulnerability Scoring System,» 12 10 2004. [Online]. Available: https://www.first.org/cvss/cvss-dhs-12-02-04.pdf. [Consultato il giorno 2014 01 01].

[57] R. E. Sawilla e X. Ou, «Identifying Critical Attack Assets in Dependency Attack Graphs,» in *Computer Security - ESORICS 2008*, Springer Berlin Heidelberg, 2008, pp. 18-34.

[58] P. Dagum, A. Galper e E. Horvitz, «Dynamic Network Models for Forecasting,» in *Proceedings of the eighth international conference on uncertainty in artificial intelligence*, 1992.

[59] S. R. Eddy, «Hidden Markov models,» *Current Opinion in Structural Biology,* vol. 6, n. 3, p. 361–365, 1996.

[60] H.-J. Park e S.-B. Cho, «Privilege flows modeling for effective intrusion detection based on HMM,» *Proceedings CDWS2 in PRICAI,* 2002.

[61] P. Smyth, D. Heckerman e M. I. Jordan, «Probabilistic independence networks for hidden Markov probability models,» *Neural computation,* vol. 9, n. 2, 1997.

[62] R. Dechter, «Bucket elimination: A unifying framework for probabilistic inference,» in *Learning in Graphical Models*, Springer Netherlands, 1998, pp. 75-104.

[63] C. E. Leiserson, T. Asano, T. H. Cormen, C. Stein e R. Rivest, Introduction to algorithms second edition., MIT Press, 2001.

[64] C. A. Coello, «An updated survey of GA-based multiobjective optimization techniques,» *ACM Computing Surveys (CSUR),* vol. 32, n. 2, pp. 109-143, 2000.

[65] W. Jakob, M. Gorges-Schleuter e C. Blume, «Application of Genetic Algorithms to Task Planning and Learning.,» in *Parallel Problem Solving from Nature 2 (PPSN-II)*, 1992.

[66] M. Gupta, J. Rees, A. Chaturvedi e J. Chi, «Matching information security vulnerabilities to organizational security profiles: a genetic algorithm approach,» *Decision Support Systems,* vol. 41, pp. 592-603, 2006.

[67] K. Deb, Multi-objective optimization using evolutionary algorithms, John Wiley & Sons, 2001.

[68] K. Deb, A. Pratap, S. Agarwal e T. Meyarivan, «A fast and elitist multiobjective genetic algorithm: NSGA-II,» *IEEE Transactions on Evolutionary Computation,* vol. 6, n. 2, pp. 182 - 197, 2002.

[69] J.-c. Ma e J.-y. Sun, «Optimal Network Hardening Model Based on Parallel Genetic Algorithm,» in *International Conference on Industrial Control and Electronics Engineering (ICICEE), 2012*, 2012.

[70] M. Dorigo, V. Maniezzo e A. Colorni, «Ant system: optimization by a colony of cooperating agents,» *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 26, n. 1, pp. 36-39, 1996.

[71] H.-H. Gao, H.-h. Yang e X.-Y. Wang, «Ant colony optimization based network intrusion feature selection and detection,» in *Proceedings of 2005 International Conference on Machine Learning and Cybernetics, 2005.*, 2005.

[72] N. Feng, H. J. Wang e M. Li, «A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis,» *Business Intelligence in Risk Management,* vol. 256, pp. 57-73, 2014.

[73] J. Kennedy, «Particle Swarm Optimization,» in *Encyclopedia of Machine Learning*, Springer US, 2010, pp. 760-766.

[74] R. C. Eberhart e J. Kennedy, «A new optimizer using particle swarm theory,» in *Proceedings of the sixth international symposium on micro machine and human science*, 1995.

[75] M. Scutari, «Learning Bayesian networks with the bnlearn R package,» *Journal of Statistical Software,* vol. 35, n. 3, pp. 1-22, 2010.

[76] R. E. Neapolitan, Learning bayesian networks, Prentice Hall, 2004.

[77] R. P. Lippmann e K. W. Ingols, «An annotated review of past papers on attack graphs,» 2005.

[78] J. M. Keynes, A Treatise on Probability, Macmillan, 1921.

[79] W. L. Fithen, S. V. Hernan, P. F. O'Rourke e D. A. Shinberg, «Formal modeling of vulnerability,» *Bell Labs Technical Journal,* vol. 8, n. 4, pp. 173 - 186, 2004.

[80] Q. Althebyan e B. Panda, «A Knowledge-Base Model for Insider Threat Prediction,» in *Information Assurance and Security Workshop, 2007. IAW'07. IEEE SMC*, 2007.

[81] «CVE - Common Vulnerabilities and Exposures,» [Online]. Available: https://cve.mitre.org/. [Consultato il giorno 01 01 2015].

[82] Red Tiger Security, LLC (USA), «SCADA Security Maturity Model,» [Online]. Available: http://redtigersecurity.com/services/scadaics-security-consulting/scada-security-maturity-model/. [Consultato il giorno 2014 01 01].

[83] C. Phillips e L. P. Swiler, «A graph-based system for network-vulnerability analysis,» in *Proceedings of the 1998 workshop on New security paradigms. ACM*, 1998.

# 2 INDICE DELLE FIGURE



## 2.1 ESEMPIO DI ESECUZIONE DEL SIMULATORE

# 3 INDICE ANALITICO

***F***



***G***



***I***



***J***



***K***



***L***



***M***



***N***



***P***

***R***



***T***



***Z***