POLITECNICO DI TORINO

Corso di Laurea Magistrale in
Ingegneria Informatica (Computer Engineering)

Tesi di Laurea Magistrale

NIDS-IX: Anomaly Detection per un Internet Exchange

Studio e progettazione di un Network Intrusion Detection System
per infrastrutture di rete di utenze connesse ad un Internet Exchange

Relatore
prof. Fulvio Risso

Laureando
Lorenzo Moro
matricola: 235425

Supervisore aziendale
Consorzio TOP-IX
dott. Leonardo Camiciotti

Anno accademico 2017 – 2018

Lorenzo Moro
NIDS-IX: Anomaly Detection per un Internet Exchange

A chi c'è sempre stato.

Ringraziamenti

La gratitudine non dovrebbe mai essere qualcosa di dovuto, quanto piuttosto la via di mezzo tra un'innata volontà e la necessità di riconoscere chi ha avuto un ruolo nei nostri successi. La tesi presentata in queste pagine è contemporaneamente il risultato del lavoro di uno studente e la somma dei contributi che molte altre persone hanno lasciato intersecando in modi diversi la mia vita: a tutte loro va la mia sincera riconoscenza, poiché il successo è già semplicemente essere arrivati a questo punto.

La famiglia innanzitutto, non solo per l'immancabile affetto e supporto materiale che mi hanno sostenuto in questi cinque anni di università, ma soprattutto per aver sempre incentivato lo sviluppo delle passioni e degli interessi che hanno scandito la mia crescita personale e culturale: il loro contributo a questa tesi non è certo da ricercarsi nei contenuti, bensì nell'avermi dato gli strumenti per affrontarla con la curiosità, la forza di volontà e la sicurezza necessari.

Non possono poi mancare gli amici, quelli di una vita, quelli che nonostante le strade diverse mi hanno sempre dimostrato affetto e un genuino interesse nel mio percorso: a loro va un sincero ringraziamento per avermi supportato nelle scelte e nell'aver sempre avuto una bella parola per me, anche nei momenti più ardui. Un ringraziamento speciale va al Mu Nu Chapter of IEEE-Eta Kappa Nu, un formidabile gruppo di studenti e studentesse accomunati dalla volontà di eccellere, l'obiettivo più nobile in quest'epoca di mediocrità: a partire dal gruppo fondatore, passando dai miei compagni di squadra durante la nostra gestione, fino ad arrivare ai più recenti iscritti nell'Associazione, sono grato per avermi dato la possibilità di riscoprirmi una persona diversa da quella che ero diventato.

Quindi ringrazio il Prof. Fulvio Risso, sia per avermi ascoltato, consigliato e aiutato nella definizione di questo lavoro di tesi che per aver sempre messo il suo ruolo didattico al primo posto: la capacità di trasmettere il sapere e il modo accattivante di farlo hanno trasformato una curiosità verso il mondo delle reti in un vivo interesse.

Infine un ringraziamento va ai ragazzi e le ragazze del Consorzio TOP-IX, un team giovane ed energico che con inaspettata serenità e spensieratezza quotidiana, mi ha aiutato nella costruzione del lavoro descritto in questa tesi: rappresentano l'anima di una realtà lavorativa di rara coesione e affiatamento, a cui auguro la crescita e il successo che merita.

Lorenzo

Sommario

Il lavoro di tesi consiste nello studio e nella progettazione di un Network Intrusion Detection System basato sull'Anomaly Detection. Il sistema è destinato alle reti terminali di soggetti operanti a livello geografico e tra loro interconnessi mediante un'infrastruttura fisica e distribuita, l'Internet Exchange, in cui il sistema opera: tali soggetti sono enti e società erogatrici di servizi locali (istruzione, trasporto pubblico, servizio idrico, distribuzione di elettricità e gas ecc.), alcuni dei quali rientrano nell'ambito della Direttiva UE 2016/1148 in quanto fornitori di servizi essenziali. Finalità del sistema è la possibilità di attivare un perimetro di sicurezza a protezione dell'Internet Exchange e dei soggetti interconnessi, garantendo la continuità dei servizi anche in caso di eventi potenzialmente dannosi.

Molti dei soggetti a cui il sistema è destinato, infatti, provengono dal mondo dell'Operational Technology, ossia quell'insieme di hardware e software in grado di controllare dispositivi fisici, processi ed eventi all'interno di un'impresa produttiva. I recenti sviluppi nella convergenza tra l'Information Technology e l'Operational Technology consentono il miglioramento e l'efficientamento dei processi aziendali, abilitando il modello di business dell'Industria 4.0 e della smart factory. Tuttavia la convergenza tra i due mondi apre il secondo ai rischi legati alla cybersecurity propri del primo, per cui occorrono competenze, risorse e sistemi adeguati per prevenire, rilevare e rispondere alle minacce: il sistema progettato per questa tesi si pone come una prima soluzione a queste problematiche.

Il lavoro è stato commissionato e sviluppato presso il Consorzio TOP-IX, fornitore del servizio di Internet Exchange per il Nord-Ovest d'Italia e quindi topologicamente centrale nella rete di soggetti afferenti su cui il sistema opera.

L'architettura del sistema si basa sul monitoraggio distribuito, ossia l'uso di un meccanismo di tipo client-server con cui l'acquisizione dei dati avviene direttamente sulla rete terminale mentre la loro interpretazione per il rilevamento di eventuali anomalie è eseguita in remoto, nell'infrastruttura dell'Internet Exchange: il processo decisionale è predisposto per l'analisi di insieme dei dati provenienti da più reti terminali, consentendo l'individuazione di anomalie generalizzate e solo apparentemente slegate.

Nell'implementazione del sistema si è data priorità a strumenti già in uso presso il Consorzio TOP-IX con particolare riguardo per l'open-source, in modo tale da lasciare spazio a futuri miglioramenti e ottimizzazioni. La componente implementativa del lavoro si è concentrata sullo sviluppo del framework per il rilevamento delle intrusioni, validato quindi attraverso un concreto caso di studio.

Indice

Elenco delle figure

Capitolo 1

Introduzione

L'uso sempre più intensivo delle tecnologie IT e delle reti nella gestione delle attività aziendali, così come l'adozione sempre più diffusa del cloud computing nell'elaborazione dei dati di processo, fanno sì che la sicurezza delle infrastrutture informatiche sia diventata una priorità nella progettazione e nella gestione dei sistemi aziendali.

In particolare, se è vero che fino a pochi anni fa la cybersecurity era una caratteristica propria del dominio IT, fondamentale per quelle aziende del terziario con un business fortemente orientato alla fornitura di servizi, oggi si assiste ad un'integrazione sempre più marcata delle infrastrutture IT nel mondo dell'*Operational Technology* (OT), dove alla gestione di sensori e attuatori tramite tradizionali sistemi di controllo industriale si stanno affiancando sistemi software di acquisizione ed elaborazione dei dati, connessi da reti tipicamente basate su IP.

Questo modello è abilitante per una gestione puntuale delle risorse aziendali, consentendo un miglior rilevamento e abbattimento delle inefficienze così come una reazione più rapida e precisa alle problematiche che possono verificarsi durante i processi. L'aumento della velocità delle reti, l'accesso sempre più capillare a Internet e l'introduzione di modelli di business basati sul paradigma *as-a-Serivce* (aaS) stanno portando verso la delocalizzazione dell'elaborazione e allo spostamento verso datacenter remoti di una crescente mole di dati aziendali.

Per questo motivo, i rischi legati ad una conoscenza superficiale delle tecnologie IT, soprattutto per quanto riguarda le problematiche legate alla cybersecurity, non possono essere sottovalutati. Dai malware in grado di compromettere sistemi ad altissimo rischio, come il celebre Stuxnet che colpì alcuni impianti di trattamento di materiale nucleare in Iran, ai più recenti ransomware in grado di paralizzare infrastrutture critiche come il sistema sanitario nazionale britannico, abbiamo diversi esempi che testimoniano l'impatto che minacce e vulnerabilità possono produrre sui sistemi aziendali. I danni ai processi si trasformano in notevoli perdite economiche, a cui si aggiungono i necessari costi di ripristino: questo spiega perché l'azione preventiva e la corretta progettazione siano la scelta migliore per un uso consapevole e produttivo di queste tecnologie.

La centralità di tali tematiche è stata evidenziata anche dalla pubblicazione di un serie di normative appositamente studiate per rispondere all'esigenza di standard di riferimento, necessari per favorire l'integrazione dei sistemi e una risposta coordinata a minacce ed eventi accidentali. La Direttiva UE 2016/1148 (NIS), descritta approfonditamente in seguito, rappresenta il primo tentativo di armonizzare le regole sulla sicurezza informatica a livello comunitario, individuando tutti quei soggetti particolarmente sensibili per i quali è necessaria una particolare attenzione alla cybersecurity dei sistemi aziendali.

Il particolare background tecnologico e informatico italiano, per molti aspetti arretrato rispetto al quadro di riferimento internazionale, è terreno fertile per lo sviluppo di sistemi innovativi e fortemente ottimizzati per rispondere adeguatamente alle esigenze di sicurezza nel rispetto delle normative. Ad oggi solo pochi player, tipicamente i più importanti e strutturati, hanno autonomamente adottato protocolli di rilevazione e reazione ad anomalie dei sistemi informatici di natura accidentale o criminale; invece i soggetti più piccoli, dotati quindi di infrastrutture più semplici ma non per questo meno critiche, per dotarsi di misure di sicurezza adeguate fanno spesso ricorso a risorse esterne specializzate.

In questo contesto il ruolo di quei soggetti che, come gli Internet Exchange, assumono naturalmente una posizione centrale nelle reti di interconnessione territoriali, può risultare particolarmente adeguato per fornire soluzioni di sicurezza remote efficaci, economiche e fortemente ottimizzate.

1.1 L'idea

Il Consorzio TOP-IX, presso cui è stato svolto il lavoro di tesi qui presentato, fornisce il servizio di Internet Exchange (IX) per il Nord-Ovest d'Italia, le cui modalità e infrastrutture sono descritte approfonditamente in seguito. Fin da subito l'attività del Consorzio si è estesa al di là del tradizionale servizio di IX, fornendo un'offerta a sostegno di progetti di imprenditorialità, innovazione tecnologica e cultura digitale. Parallelamente è rilevante l'eterogeneità dei consorziati, tra cui figurano sia grandi player nazionali che piccole realtà territoriali afferenti a diversi settori economici (dai gruppi bancari alle istituzioni scolastiche, dai carrier agli ISP).

Nel contesto d'impresa descritto, tenuto conto delle premesse presentate nell'introduzione, il Consorzio ha rilevato la potenzialità nell'arricchire la propria proposta con un servizio di cybersecurity esplorativo, accessorio a quello di IX, pensato per i già e futuri consorziati: l'idea prevede la possibilità di attivare un perimetro di sicurezza "data-driven" a protezione dei consorziati e dell'IX stesso, le cui policy siano definite dinamicamente sulla base di dati acquisiti monitorando le reti connesse. La variabilità del perimetro segue il livello di allarme dell'infrastruttura considerata nel suo insieme, ovvero sia della rete dell'IX che delle reti terminali dei soggetti afferenti. Alcuni di questi sono annoverati tra i cosiddetti Operatori di Servizi Essenziali (OSE) e Fornitori di Servizi Digitali (FSD) ai sensi della Direttiva NIS, così come definita nell'introduzione, e pertanto sono tenuti a rispettare diversi requisiti di cybersecurity: il servizio ideato può quindi rappresentare una prima soluzione in tal senso. Il valore aggiunto rappresentato dal considerare le infrastrutture degli OSE e degli FSD come un unico sistema, caratterizzato da uno specifico stato di "salute", è un'ipotesi di ricerca nella progettazione del servizio.

Infatti, gli studi compiuti all'Imperial College London sul progetto denominato Digital City Exchange, descritto approfonditamente in seguito, hanno mostrato il valore che l'analisi congiunta dei dati prodotti da soggetti eterogenei operanti a livello territoriale può produrre, soprattutto se contestualizzati, normalizzati e messi a disposizione su una piattaforma comune: la correlazione e la successiva interpretazione di questi dati abilita processi innovativi rivolti a migliorare l'efficienza, la disponibilità e la ricchezza dell'offerta. Questi studi rappresentano un interessante contributo per la progettazione del servizio in quanto, essendo topologicamente centrale la posizione di rete del Consorzio rispetto ai consorziati, ben si presta a trarre beneficio dalla correlazione dei dati prodotti da questi ultimi al fine di proteggerli e proteggere la rete del Consorzio.

Nella stesura del progetto emerge la necessità di confrontarsi con l'impatto che un servizio di questo tipo potrebbe avere dal punto di vista non solo tecnologico, ma anche legale: quando si parla di acquisizione ed elaborazione dei dati, infatti, è sempre più necessario considerare la legittimità delle azioni nel rispetto delle leggi sulla privacy e, nel contesto aziendale considerato, sulla tutela del lavoro. Nel capitolo conclusivo sono presenti alcune considerazioni in merito suffragate da opinioni di esperti del dominio.

1.2 Il progetto

Lo studio del dominio e l'analisi dello stato dell'arte hanno permesso di tradurre l'idea di servizio, proposta dal Consorzio TOP-IX, nel progetto di un sistema che lo implementi. Questa fase è avvenuta con una serie di contestualizzazioni volte a concretizzare l'idea, modellandone la portata pratica entro i limiti imposti da un lavoro di tesi, traendo ispirazione dagli studi già condotti in merito, e provando a mappare i requisiti espressi nelle funzionalità del sistema.

La prima contestualizzazione è stata fatta sui soggetti destinatari del sistema: i consorziati potenzialmente più interessati, infatti, sono quelli meno strutturati e con una bassa "autonomia informatica" (intesa come l'indipendenza da soggetti terzi nella progettazione, installazione, configurazione e uso delle tecnologie informatiche), tra cui le istituzioni scolastiche, gli enti territoriali, le aziende municipalizzate, le aziende di trasporto pubblico, gli istituti sanitari ecc. Di questi gli OSE e gli FSD sono certamente quelli che più possono trarne beneficio, ma la portata si è rivelata adeguata anche per altri tipi di soggetti.

La componente tecnologica della ricerca si è focalizzata sugli strumenti noti come *Intrusion Detection System* (IDS), dal cui funzionamento il sistema progettato trae origine: nello specifico si parla di *Network Intrusion Detection System* (NIDS), presentati approfonditamente in un capitolo dedicato, basati su *Anomaly Detection* (AD). Il perimetro di sicurezza ipotizzato, quindi, è stato contestualizzato nell'insieme di policy che regolano l'attività del NIDS, dove la componente "data-driven" sfrutta l'acquisizione di informazioni mediante alcuni strumenti di monitoraggio già in uso presso il Consorzio: questa scelta è stata fatta per favorire un'alta curva di apprendimento e una veloce messa in produzione del sistema. La componente decisionale inoltre sfrutta strumenti open-source, così da lasciare spazio a futuri miglioramenti e ottimizzazioni.

Per rispettare il vincolo di minimo impatto del sistema si è valutato adeguatamente il trade-off tra efficacia e invasività: a soluzioni più complete ma di impatto maggiore, come l'analisi dei pacchetti in tempo reale, si sono preferite scelte più semplici ma non per questo meno efficaci, come il monitoraggio di parametri prestazionali dei sistemi terminali. La particolare architettura del sistema, infatti, consente di delegare il processo decisionale ad un'elaborazione remota in grado di scalare adeguatamente a seconda della dimensione della rete in analisi, senza richiedere notevoli prestazioni per gli apparati locali: questo permette l'uso di algoritmi di Machine Learning per aumentare l'intelligenza del sistema, come specificato nel capitolo dedicato all'architettura.

L'efficacia del sistema è dimostrabile su un ampio spettro di minacce conosciute, ma ai fini del lavoro di tesi qui presentato ci si è concentrati su un caso di studio specifico, con focus su un tipo di minaccia molto comune e di non facile rimedio: l'attacco Denial of Service basato su TCP SYN Flood. I risultati si sono rivelati compatibili con quelli attesi, costituendo un valido presupposto per la continuazione delle ricerche e la successiva messa in produzione.

1.3 Struttura della tesi

La tesi è organizzata come segue. Il Capitolo 2 descrive i presupporti al lavoro sviluppato in questa tesi, ovvero il contesto di soggetti, tecnologie e normative analizzate. Il Capitolo 3 presenta la tecnologia studiata e adottata per il lavoro di tesi. Il Capitolo 4 riporta lo stato dell'arte, descrivendo i lavori correlati. Il Capitolo 5 descrive le scelte progettuali e l'architettura del sistema sviluppato. Il Capitolo 6 presenta l'implementazione del sistema, con particolare riferimento agli strumenti utilizzati. Il Capitolo 7 riporta il caso di studio analizzato e la conseguente validazione sperimentale del sistema. Infine il Capitolo 8 conclude il lavoro di tesi con riferimento ai possibili sviluppi futuri.

Capitolo 2

Background

Il lavoro descritto in questa tesi riguarda aree di studio eterogenee e talvolta lontane, e per questo motivo è opportuno presentare il contesto di tecnologie, soggetti e normative che sono stati considerati. L'eterogeneità è rappresentata dall'ampio spettro di soggetti coinvolti, afferenti al mondo dell'industria, delle telecomunicazioni, dell'informatica e della giurisprudenza. La lontananza è spesso solo apparente, in quanto domini così distinti si sono rivelati fortemente accoppiati nella necessità di far fronte comune ai problemi descritti nell'introduzione.

Questo capitolo inizia introducendo il mondo dell'Operational Technology, l'ambito entro cui operano i principali destinatari del sistema sviluppato, per poi affrontare l'evoluzione di quest'ultimo nel processo di convergenza verso l'Information Technology e i problemi che questo comporta. Un excursus sulle normative del settore, sui soggetti da queste individuati e sui rischi a cui sono sottoposti completa il contesto in cui si inserisce questo lavoro di tesi. Il capitolo si chiude con un approfondimento sugli Internet Exchange e nello specifico sul Consorzio TOP-IX, in cui e per cui il sistema è stato sviluppato.

2.1 L'Operational Technology

I sistemi automatici di controllo industriale permettono a numerosi settori di operare con adeguati livelli di affidabilità e sicurezza: dalla chimica alla manifattura, dall'energia all'acqua, dai trasporti al trattamento dei rifiuti, i processi industriali sono da molto tempo gestiti e resi efficienti tramite l'uso della cosiddetta Operational Technology (OT), definita da Gartner come *"l'hardware e il software in grado di rilevare o produrre un cambiamento attraverso il monitoraggio diretto e/o il controllo di dispositivi fisici, processi ed eventi all'interno di un'impresa"* [1].

A seconda delle funzionalità disponibili e della sofisticatezza dell'azione di controllo, il termine *Industrial Control System* (ICS) può fare riferimento a diverse tipologie di sistemi e differenti tecnologie, tra le quali quelle più comuni e diffuse sono le seguenti:

- **SCADA** (Supervisory Control And Data Acquisition) In un sistema SCADA il supervisore invia a PLC e microcontrollori le "istruzioni di processo" sotto forma di comandi, ma sono poi questi ultimi a controllare il processo verificando e regolandone i parametri, anche in caso di indisponibilità del sistema SCADA (spesso remoto, accessibile tramite WAN)
- **DCS** (Distributed Control System) I sistemi DCS hanno una forte interazione tra microcontrollori locali e sistema di supervisione, solitamente realizzato mediante sistemi operativi deterministici che garantiscono tempi di risposta certi, connessi nella stessa LAN.
- **EMS** (Energy Management System) Un EMS è un insieme di strumenti informatici usati dagli operatori delle reti elettriche per monitorare, controllare e ottimizzare le prestazioni dei sistemi di generazione e trasmissione dell'energia. Spesso è accoppiato a sistemi SCADA.

Più in generale, tutti gli ICS sono riconducibili ad un'architettura basata su tre componenti principali: il centro di controllo, i controller di campo e le sonde/attuatori. Il primo esegue le funzioni di comando e controllo per tramite di *Human-Machine Interface* (HMI), software di gestione dei processi e Historian DB; il centro di controllo comunica tramite opportuni protocolli con i controller di campo: tali dispositivi implementano la logica di controllo e possono essere costituiti da *Programmable Logic Controller* (PLC), *Remote Terminal Unit* (RTU) e *Programmable Automation Controller* (PAC); infine i controller di campo comunicano mediante protocolli di campo con sonde e attuatori, i quali rappresentano l'I/O del processo: le sonde sono sostanzialmente misuratori e sensori, mentre tra gli attuatori possono essere valvole, motori, relè ecc.

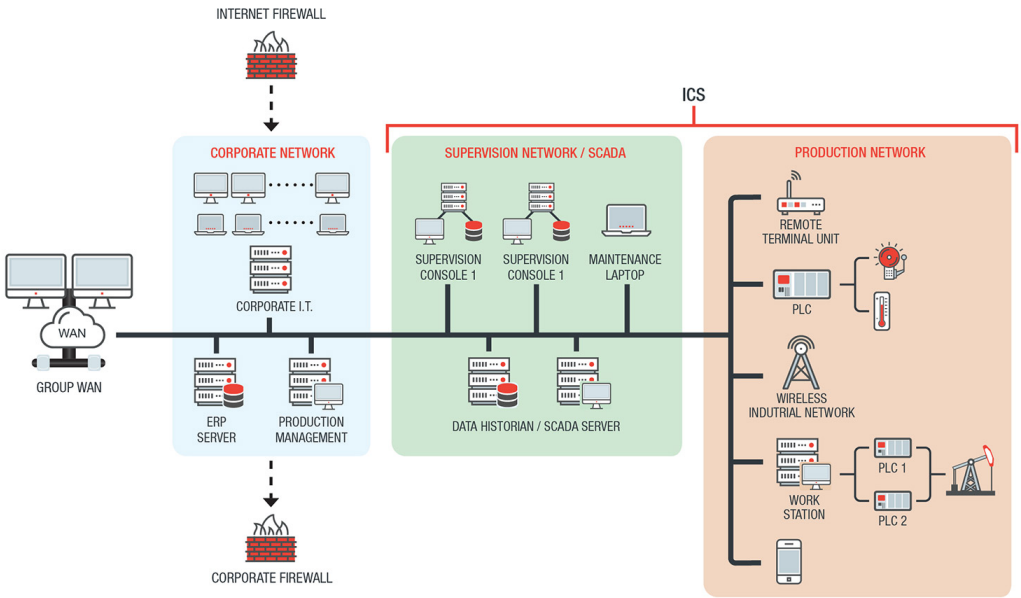

Figura 2.1. Scenario applicativo di una rete ICS/SCADA

2.2 Convergenza IT/OT

Storicamente, le strutture aziendali di Information Technology e di Operational Technology potevano funzionare indipendentemente: l'OT si preoccupava della regolare operatività degli impianti, mentre l'IT delle business application per gli uffici. I punti di incontro tra i due mondi sono sempre stati occasionali e poco significativi, spesso nati solo in occasione di problemi come incidenti di sicurezza, guasti di sistema, inattività non pianificate ecc.

Negli ultimi anni si sta diffondendo la consapevolezza che per abilitare i nuovi modelli di business legati all'Industria 4.0 e alla *smart factory* sia necessaria una convergenza tra il mondo IT e quello OT, supportata dalla diffusione dei sistemi cosiddetti cyberfisici, ossia sistemi informatici in grado di interagire in modo continuo con i sistemi fisici in cui operano: i dispositivi che rientrano in questa definizione sono dotati individualmente di capacità computazionale, comunicazione e capacità di controllo, caratteristiche proprie sia del mondo informatico che di quello del controllo industriale.

Tra i possibili scenari, individuati da Cisco [2], che meglio rappresentano i benefici della convergenza IT/OT nelle industrie di nuova generazione vi sono:

- La capacità di fare scelte di processo in tempo reale tramite il **fog computing**.

  L'enorme mole di dati raccolta tramite i sistemi cyberfisici e l'Industrial Internet of Things (IIoT) rappresenta un valore solo nel momento in cui viene usata per prendere decisioni: il cloud computing da solo non è sufficiente, in quanto le decisioni time-sensitive devono essere prese il più vicino possibile ai dispositivi che producono e lavorano sui dati, minimizzando latenza e potenziali problemi; al contempo i tradizionali sistemi SCADA non permettono la stessa flessibilità nella condivisione dei dati a livello di business application nel mondo dello smart manifacturing.

  Il paradigma del fog computing rappresenta una valida soluzione, in quanto adeguato sia per rispondere alle esigenze IT che OT: i dati time-sensitive possono essere analizzati nel nodo fog più vicino al dispositivo generatore (es. a bordo macchina), quelli che possono attendere secondi o minuti possono essere passati ad un nodo intermedio (es. a livello di impianto produttivo) e quelli non prioritari vengono mandati al cloud per data mining e storage. In ogni caso, il potenziale decisionale del dato non viene mai perduto.

- L'eliminazione dei downtime non pianificati tramite la **manutenzione predittiva**.

  Oggigiorno i piani di manutenzione preventiva sono lo standard per prevenire i downtime imprevisti negli impianti produttivi. Ma se la manutenzione preventiva è sicuramente un'alternativa migliore all'attesa di un guasto, certamente non è perfetta: è un insieme di procedure molto costose, sia in termini di tempo che di denaro, che non sempre sono efficaci in risposta a eventi straordinari. Basandosi sulla prassi, e non sui dati effettivi, è inevitabile che si verifichino comunque una piccola percentuale di downtime occasionali e sprechi di risorse.

  La manutenzione predittiva, al contrario, consente di acquisire e analizzare in tempo reale i dati di produzione in modo da rilevare potenziali situazioni rischiose e

programmare la manutenzione durante i downtime pianificati. Passare dalla manutenzione preventiva a quella predittiva significa aumentarne notevolmente l'efficienza, e la transizione è agevolata dalla convergenza IT/OT: il ruolo dell'OT è raccogliere i dati utili dai PLC e dai sensori a bordo macchina, mentre l'IT analizza e interpreta tali dati per determinare situazioni rischiose, così da condurre azioni mirate e correttamente dimensionate.

- L'uso delle **reti wireless** negli impianti di produzione.

  Gli ambienti industriali variano molto, dalle complessità nell'organizzare spazi ed edifici alle difficili condizioni ambientali, come polvere, umidità elevata, temperatura, vibrazioni, campi magnetici ecc. Per tali motivi, i direttori di stabilimento sono sempre stati scettici sull'idea che le tecnologie wireless possano essere adeguate per il numero di dispositivi, la larghezza di banda, la latenza e la sicurezza richiesta da applicazioni mission-critical: questo ha tipicamente portato all'installazione di chilometri di cavi che collegano ogni angolo degli stabilimenti produttivi.

  Negli ultimi anni, tuttavia, le ricerche sulle tecnologie wireless hanno prodotto dispositivi con maggior resilienza, affidabilità e prestazioni senza compromettere la rapidità di installazione. Il wireless è una delle tecnologie abilitanti per una smart factory: consente maggior flessibilità, predispone ad un monitoraggio remoto più completo, aumenta l'efficienza della catena di montaggio e abilita nuove iniziative per il controllo qualità di prodotti e forniture. Non ultimo, in ambito industriale il wireless consente di risparmiare fino a 10 volte rispetto al cablato.

2.3 Cybersecurity nell'OT

A causa delle differenti finalità dei sistemi ICS rispetto ai sistemi IT, l'approccio alla sicurezza deve essere adattato. Per esempio, analizzando il profilo di rischio, in ambito IT prevale la gestione dei dati, la confidenzialità e l'integrità, mentre in ambito OT sono fondamentali la sicurezza fisica e la disponibilità dei processi. Anche i cambiamenti hanno tempi e modalità diverse: se un sistema informatico ha una vita utile di 3-5 anni, viene automaticamente aggiornato e il programma di sostituzione è regolare, un ICS deve prevedere una vita utile ben più lunga (spesso oltre i 15 anni), gli aggiornamenti richiedono approvazione preventiva e prove approfondite, e la sostituzione (spesso non programmata) richiede tempi lunghi. [3]

Tali differenze evidenziano una serie di criticità proprie dell'ambito ICS. Come detto, infatti, se tradizionalmente la rete SCADA era "isolata" e quindi "sicura", oggi questo isolamento sta svanendo a più livelli (tecnologico, organizzativo e di interconnessione), creando seri problemi di sicurezza:

- **Policy di sicurezza IT su SCADA insufficienti**: spesso le policy pensate per le infrastrutture informatiche non coprono completamente le best practices relative alla cybersecurity del mondo OT, soprattutto lavorando con diversi profili di rischio.

- **Protocolli non protetti**: molti dei protocolli usati dai sistemi SCADA non sono stati testati specificamente sulla sicurezza, e spesso mancano di procedure di autenticazione o sono basasti su standard ASN.1 (linguaggio di descrizione dell'intefaccia), come il *Distributed Network Protocol* (DNP) e l'*Inter-Control Center Communications Protocol* (ICCP).

- **Uso di OS standard per molte componenti**: essendo il mondo OT poco reattivo all'aggiornamento dei sistemi di produzione (dove si rileva un patching limitato o assente), l'uso di OS standard rappresenta un grave problema di sicurezza; talvolta vulnerabilità identificate anni prima sono ancora presenti nelle reti SCADA, dove spesso le utenze di default non sono disattivate.

- **Limitata presenza di controlli/sistemi di sicurezza sugli host**: è frequente la presenza di servizi di sistema non utilizzati e tenuti comunque attivi (es. Server HTTP), una gestione carente delle utenze (utenze condivise, password deboli, di default o "eterne") e una limitata o assente gestione dei permessi nel filesystem.

- **Vulnerabilità ad attacchi "tradizionali"**: DoS e DDoS sui sistemi di accesso per il telecontrollo, worm e virus sulle reti di processo, phishing, DNS poisoning/redirection ecc. sono tra le minacce più comuni sui sistemi industriali.

- **Insufficiente segmentazione delle reti**: l'insufficiente segmentazione delle reti non isola adeguatamente gli ambienti SCADA, con comunicazioni non criptate (perché ritenute affidabili) e perimetri di sicurezza scarsamente identificati.

Le criticità evidenziate individuano una serie di vettori di attacco e di relativi macro rischi ad essi collegati, come mostrato nella figura 2.2.

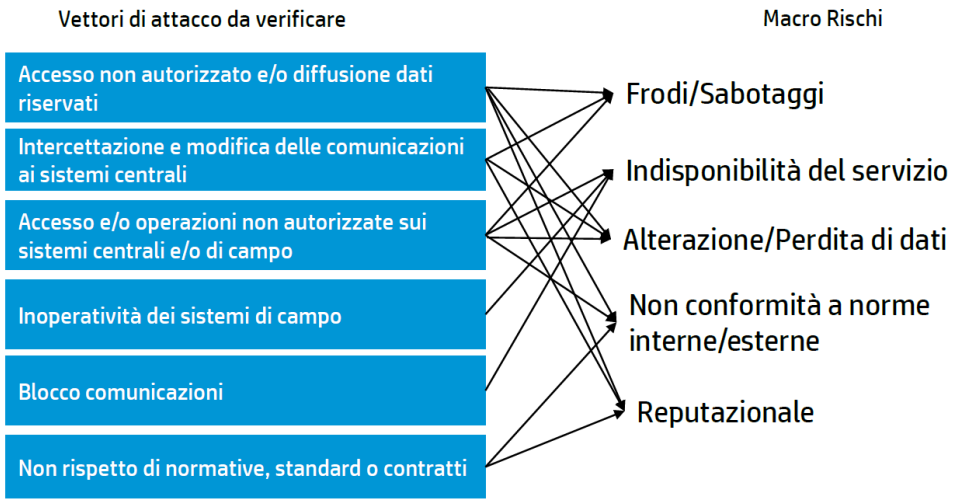

Figura 2.2. Vettori di attacco e relativi macro rischi

2.4 Normative

Come anticipato nell'introduzione, la centralità della cybersecurity nel processo di convergenza tra l'Information Technology e l'Operational Technology ha dato il via alla pubblicazione di un serie di normative appositamente studiate per rispondere all'esigenza di standard di riferimento, necessari per favorire l'integrazione dei sistemi e una risposta coordinata a minacce ed eventi accidentali. Nel processo normativo gli USA hanno anticipato notevolmente i tempi rispetto all'Unione Europea, ma si sta ora convergendo verso un quadro più definito, completo e condiviso anche a livello internazionale. Le norme del settore hanno dato un importante contributo nella fase di analisi del dominio, e sono state tenute in considerazione per verificarne la rispondenza del sistema progettato.

2.4.1 La Direttiva UE 2016/1148 (NIS)

Il 6 luglio 2016 il Parlamento Europeo ha adottato la Direttiva UE 2016/1148 "recante misure per un livello comune elevato di sicurezza delle reti e dei sistemi informativi nell'Unione", detta comunemente Direttiva NIS. [4] Essa rappresenta il primo insieme di regole sulla sicurezza informatica univoco a livello dell'Unione Europea. I punti chiave della Direttiva NIS sono:

- obbligare tutti gli Stati membri ad adottare una strategia nazionale in materia di sicurezza della rete e dei sistemi informativi;
- istituire un gruppo di cooperazione al fine di sostenere e agevolare la cooperazione strategica e lo scambio di informazioni tra Stati membri e di sviluppare la fiducia tra di essi;
- creare una rete di gruppi di intervento per la sicurezza informatica in caso di incidente, detta rete CSIRT (Computer Security Incident Response Team), per contribuire allo sviluppo della fiducia tra Stati membri e promuovere una cooperazione operativa rapida ed efficace;
- stabilire obblighi di sicurezza e di notifica per gli operatori di servizi essenziali e per i fornitori di servizi digitali;
- obbligare gli Stati membri a designare autorità nazionali competenti, punti di contatto unici e CSIRT con compiti connessi alla sicurezza della rete e dei sistemi informativi.

In merito al primo punto, tra gli aspetti che una strategia nazionale dovrebbe includere vengono citati gli obiettivi strategici, le priorità nazionali, la governance, l'individuazione di misure proattive, di risposta e di recovery; sensibilizzazione, formazione ed istruzione; incentivazione della cooperazione tra settore pubblico e settore privato; lista degli attori coinvolti nella attuazione della strategia. Ogni Stato dovrà infatti designare uno o più CSIRT responsabili del monitoraggio degli incidenti a livello nazionale, fornendo allarmi tempestivi, avvisi ed annunci con lo scopo di diffondere informazioni su rischi e incidenti.

La cooperazione tra i vari enti dei singoli Stati membri è un punto fondamentale della direttiva NIS. Proprio per questo è stato stabilito un gruppo di cooperazione che faciliti i rapporti tra gli Stati membri e che aumenti la fiducia. Questo gruppo di cooperazione sarà composto da rappresentati degli Stati membri, dalla Commissione e dall'ENISA (European

Union for Network and Information Security Agency). Le quattro aree di lavoro del gruppo saranno: pianificazione, guida, segnalazione e condivisione.

Il quarto punto chiave della Direttiva riguarda gli Operatori di Servizi Essenziali (OSE) e i Fornitori di Servizi Digitali (FSD) nazionali. Gli OSE sono aziende pubbliche o private che hanno un ruolo importante per la società e l'economia, quelli che comunemente vengono chiamate "infrastrutture critiche". La direttiva NIS obbligherà queste entità a dotarsi di misure di sicurezza appropriate e a notificare all'autorità nazionale competente gravi incidenti di sicurezza secondo parametri di numero di utenti coinvolti, durata dell'incidente e diffusione geografica. Le misure di sicurezza richieste comprendono: prevenzione dei rischi; garanzia della sicurezza dei sistemi, delle reti e delle informazioni; capacità di gestire gli incidenti. Queste entità verranno identificate direttamente da ogni Stato membro, all'interno dei seguenti ambiti: energia, trasporti, banche e società finanziarie, salute, acqua ed infrastrutture digitali. I criteri che determinano quali enti saranno inclusi in questa lista sono: l'essenzialità del servizio offerto per il mantenimento di attività critiche in ambito economico e sociale; la dipendenza del servizio da sistemi informatici; il rischio che l'incidente di sicurezza possa avere effetti gravi e significativi sulla fornitura di un servizio essenziale.

Anche gli FSD sono tenuti, secondo la Direttiva NIS, ad attuare misure di sicurezza appropriate e a notificare incidenti rilevanti. Oltre alle misure già previste per gli operatori di servizi essenziali, le misure di sicurezza relative ai fornitori di servizi digitali prevedono alcuni fattori specifici, come ad esempio la sicurezza dei sistemi e degli impianti, la gestione della continuità operativa, il monitoraggio e la conformità a norme internazionali. Per quanto riguarda i parametri per la valutazione di un incidente rilevante che va segnalato, vi sono oltre a quelli già elencati prima, anche l'entità dell'interruzione del servizio e l'impatto sulle attività economiche e sociali. Tra gli FSE la Direttiva NIS cita i mercati on-line, i servizi di cloud e i motori di ricerca.

2.4.2 Gli Standard NERC-CIPv5

La quinta versione degli standard NERC (North American Reliability Corporation) CIP (Critical Infrastructure Protection) sono progettati per mettere in sicurezza gli asset fondamentali del sistema elettrico di produzione e distribuzione (BES) nel Nord America. Il BES comprende gli impianti di generazione così come stazioni, linee e torri di trasmissione. Tra questi, gli asset critici sono gli impianti di generazione, le stazioni di trasmissione e i centri di controllo. Invece i cosiddetti "cyber asset" sono i sistemi di controllo di tipo SCADA, i sistemi di gestione energetica e i sistemi di controllo distribuito degli impianti, anch'essi sottoposti agli standard NERC-CIP.

La conformità alla norma è garantita dal rispetto di 8 standard, per totale di 42 requisiti, rispondenti alle funzionalità di identificazione, protezione, rilevamento, risposta e ripristino:

- **Cyber asset critici**: definizione di metodi per l'identificazione, la verifica e l'approvazione annuale degli asset critici.
- **Gestione della sicurezza**: definizione delle policy di sicurezza e delle eccezioni, controllo degli accessi, change control e gestione delle configurazioni.

- **Sicurezza del personale**: training, verifica dei rischi legati al personale, accesso.
- **Perimetro di sicurezza elettronico**: gestione delle identità e degli accessi digitali, monitoraggio degli accessi, vulnerability assessment, gestione della documentazione.
- **Sicurezza fisica e ambientale**: definizione del piano della sicurezza fisica, controllo degli accessi fisici, protezione dei sistemi di controllo accessi, monitoraggio degli accessi fisici.
- **Integrità dei sistemi e dei dati**: definizione delle procedure di test, identificazione delle porte e dei servizi, patching, prevenzione di virus e malware, monitoraggio della sicurezza, terminazione sicura degli asset.
- **Risposta agli incidenti**: definizione del piano di risposta agli incidenti di cybersecurity e relativa documentazione.
- **Pianificazione delle contingenze**: definizione del piano di recupero, pianificazione delle esercitazioni, programmazione delle procedure di backup e ripristino, test dei backup.

2.5 I servizi essenziali

In questa sezione si presenta l'analisi effettuata sugli OSE in relazione alla cybersecurity. Da quanto analizzato emerge una sostanziale percezione del problema, che tuttavia trova risposte concrete solo nei grandi player dei vari settori. Nel caso specifico italiano, la situazione è caratterizzata da una profonda eterogeneità di infrastrutture, il che rende ardua l'attuazione del principio "security-by-design".

Il settore più interessato dall'analisi è quello energetico, percepito come di importanza strategica su tutti gli altri ambiti; il settore idrico condivide alcuni tipi di minacce con quello energetico, sebbene soggetto a minor rischio a causa di una minore superficie di attacco; i settori trasporti, banche e finanza hanno già provveduto da tempo ad attuare politiche di gestione della cybersecurity, in quanto già interessanti da direttive del settore, e per tale motivo non sono oggetto di ricerca; il settore della sanità si è dimostrato particolarmente critico dal punto di vista della resilienza, rappresentando uno degli ambiti più a rischio dell'analisi; infine le infrastrutture digitali sono naturalmente più interessante dal problema della cybersecurity, soprattutto per lo scostamento dal mondo OT (tipico degli altri settori) a favore di una più spiccata essenza IT, ma per questo motivo sono già dotati di strumenti per la protezione delle proprie reti.

2.5.1 Energia

Il settore energetico è percepito come quello più strategico dal punto di vista della disponibilità, e per questo motivo è quello su cui sono state condotte più analisi e ricerche. La fonte più recente è l'"Energy Cybersecurity Report 2018" prodotto dall'Energy & Strategy Group della School of Management del Politecnico di Milano [5]. La filiera energetica sta diventando sempre più soggetta a rischi di natura informatica in quanto sta aumentando la superficie di attacco: fonti rinnovabili, generazione distribuita e smart grid richiedono

una crescente interconnessione e integrazione delle reti ICT centrali, degli impianti e degli apparati, esponendo le infrastrutture (soprattutto dei soggetti più piccoli e giovani) a minacce simili a quelle dirette ai sistemi informativi delle reti aziendali.

I target più frequenti sono i sistemi ICS e nello specifico SCADA: negli ultimi dieci anni sono stati individuati cinque malware appositamente progettati per compromettere questo tipo di sistemi, a partire da Stuxnet [6] fino al più recente TRITON/TRISIS [7]. L'analisi di queste minacce denota una profonda conoscenza del dominio da parte dei cracker, pattern di attacco simili ed effetti volti ad ottenere informazioni [8], compromettere i sistemi fisici [6], creare un disservizio [9][10] o deprimere i sistemi di sicurezza fisica (safety) [7].

Lo studio ha rilevato che il tema è molto sentito, ma appena la metà degli operatori svolge attività di risk analisys in modo sistematico, e solo il 23% del campione intervistato dichiara di aver investito nella cybersecurity OT [5]. Tra coloro i quali sono sia produttori che consumatori di energia elettrica, quindi potenzialmente più esposti a causa della natura collaterale del loro ruolo di operatori energetici, solo il 6% ritiene che l'operatività degli impianti possa essere compromessa da un attacco, mentre il 35% ritiene che gli strumenti di sicurezza inseriti dai fornitori siano sufficienti a garantirne la copertura.

2.5.2 Servizio Idrico Integrato

Il settore idrico, comprendente sia la fornitura di acqua corrente che la gestione delle acque reflue, è anch'esso un ambito fortemente a rischio di attacchi informatici. Come per il settore energetico, la maggior parte dei sistemi di monitoraggio e controllo degli impianti sono ICS di tipo SCADA, quindi condividono lo stesso tipo di minacce già presentato. Più nello specifico, sono stati già da tempo individuati i seguenti rischi propri del settore [11]:

- dosaggio chimico eccessivo o insufficiente durante il trattamento dell'acqua;
- spegnimento o controllo non autorizzato delle stazioni di pompaggio;
- riduzione della pressione agli idranti antincendio;
- riversamento di liquami non trattati nelle vie d'acqua.

Non sono ancora stati rilevati malware specificatamente progettati per attaccare i sistemi idrici, ma si segnalano diversi casi di intrusione nelle infrastrutture, come l'episodio avvenuto in Australia nel 2001 in cui un ex dipendente di una software house insoddisfatto si è introdotto nel sistema di controllo di un impianto di depurazione, riversando nei fiumi vicini un milione di litri di acque reflue non trattate e causando un importante danno ambientale [12].

2.5.3 Sanità

Il settore della sanità è diventato uno dei maggiori obiettivi di attacchi informatici nel 2015, come evidenziato da IBM nel rapporto "Security trends in the healthcare industry" [13]. Il motivo principale è il grande valore che i dati sanitari hanno sul mercato nero, in quanto contengono molte informazioni personali utilizzabili per creare documenti falsi o acquistare attrezzatura medica da rivendere; inoltre il furto di questo tipo di dati è particolarmente

difficile da scoprire, a differenza per esempio del numero di una carta di credito, facilmente bloccabile.

Per questi motivi la principale finalità degli attacchi in ambito sanitario è il data breach. Ad aggravare il contesto è l'enorme costo che l'industria sanitaria deve sostenere per ogni record sottratto, stimato di $355 nel 2016, più del doppio rispetto al costo medio in ambito industriale. Un altro tipo di minaccia che ha trovato terreno fertile nel settore è il ransomware, ovvero un malware in grado di cifrare i dati del dispositivo vittima dell'attacco e chiedere all'utente un riscatto per la loro restituzione: questo attacco si è rivelato particolarmente efficace con cliniche e ospedali in quanto più facilmente disposti a pagare il riscatto a fronte di un costo di inoperatività stimato intorno a $8.000 al minuto per incidente [14].

Per quanto concerne la sanità italiana, il quadro è ancora più complicato: pur non essendo stati ancora registrati attacchi diretti al Sistema Sanitario Nazionale, la situazione degli ospedali italiani ad oggi non consente di attuare efficaci politiche di difesa. Infatti, il proliferare di dispositivi eterogenei, l'uso di sistemi operativi obsoleti, e le reti che sono cresciute negli anni senza un'efficace strategia di insieme richiederebbero la riprogettazione completa delle infrastrutture, nel principio del security by design, cosa che non è compatibile con il limitato budget che le istituzioni sanitarie hanno a disposizione per le risorse IT.

2.6 I servizi digitali

I mercati online, i motori di ricerca online e i servizi di cloud computing rientrano tra i cosiddetti Fornitori di Servizi Digitali (FSD) ai sensi della Direttiva NIS. Molte imprese nell'Unione Europea e nel mondo fanno affidamento su questi soggetti per la fornitura dei loro servizi. Alcuni servizi digitali rappresentano una risorsa fondamentale per i loro utenti, tra cui figurano anche gli OSE, in quanto essi spesso non hanno alternative a disposizione. La sicurezza, la continuità e l'affidabilità di questo tipo di servizi, quindi, sono di primaria importanza per il corretto funzionamento di molti processi aziendali.

L'ENISA (European Union Agency for Network and Information Security) è il centro di esperti di rete e cybersecurity dell'Unione Europea: lavora per sviluppare regolamenti e raccomandazioni sulle buone pratiche nella sicurezza informatica, assistendo gli stati membri dell'UE nell'implementazione delle leggi comunitarie per aumentare il livello resilienza delle infrastrutture critiche. Per tale motivo, l'ENISA ha individuato una serie di misure di sicurezza per i Fornitori di Servizi Digitali, raggruppate in alcuni macro-obiettivi, di cui si riportano i più rilevanti [15]:

- Policy per la sicurezza dei dati
- Ruoli di sicurezza
- Controlli in background
- Sicurezza fisica e ambientale
- Controllo accessi alla rete e ai sistemi
- Integrità dei componenti della rete e dei sistemi

- Rilevazione e risposta agli incidenti di sicurezza
- Segnalazione degli incidenti di sicurezza
- Business continuity
- Disaster ricovery
- Monitoraggio e registrazione
- Conformità
- Sicurezza delle interfacce
- Sicurezza del software
- Interoperabilità e portabilità

Per ciascun obiettivo le rispettive misure di sicurezza sono classificate entro tre livelli di sofisticatezza: base, standard industriale e stato dell'arte, a seconda della dimensione e dell'importanza del provider. Si può notare come questi obiettivi abbiano molteplici intersezioni con gli standard americani della raccolta NERC-CIP v5, a riprova che il sistema sviluppato possa trovare applicazione anche per soggetti diversi dagli OSE: chiaramente gli FSD, trattando nativamente business legati all'IT, sono statisticamente più pronti a rispettare i requisiti di cybersecurity e a proteggere adeguatamente i propri sistemi.

2.7 Gli Internet Exchange Point

Internet è una rete di reti, ciascuna controllata da un'entità distinta. Queste entità sono genericamente chiamate *Internet Service Provider* (ISP), e le reti che controllano sono raggruppate in *Autonomous System* (AS). Perché un AS abbia connettività verso Internet deve necessariamente essere connesso a un altro AS che già disponga di connettività verso Internet; quando tali AS appartengono a ISP diversi, la connettività è regolata da accordi economici e prende il nome di *transito*. Tutti gli ISP devono acquistare transito da altri ISP tranne una ristretta cerchia di operatori globali classificati come Tier-1, i quali hanno accesso a Internet semplicemente per il fatto di essere reciprocamente connessi. In questo modello gerarchico, tutto il traffico scambiato tra AS di ISP "piccoli" (detti Tier-2 o Tier-3) dovrebbe transitare attraverso gli AS di ISP "grandi" a fronte di un corrispettivo oneroso.

Per ovviare a questo problema, così da ridurre i costi e rendere più efficiente lo scambio di traffico, alcuni operatori di dimensioni simili spesso decidono di connettere i propri AS direttamente: questa pratica è conosciuta come *peering* e solitamente non comporta un corrispettivo oneroso. Tuttavia il collegamento tra AS di ISP distinti è un'operazione costosa, e attuarla su scala globale tra tutti gli ISP non sarebbe economicamente sostenibile, scalabile e gestibile. Gli *Internet Exchange Point* (IXP) forniscono una soluzione parziale a questo problema.

Un IXP, conosciuto anche come Network Access Point, è una struttura tecnica che consente a diversi operatori di rete lo scambio reciproco di traffico Internet attraverso un'infrastruttura fisica condivisa. L'AS di un ISP connesso a un IXP ha la possibilità di scambiare traffico con gli AS di qualsiasi altro ISP connesso allo stesso IXP sfruttando un singolo

collegamento fisico, superando così il problema di scalabilità dei collegamenti individuali. Questa pratica è chiamata *peering pubblico* all'opposto del cosiddetto *peering privato* tra gli AS di due ISP direttamente connessi, così come descritto in precedenza [16]. Una panoramica del modello gerarchico delineato è visibile nella figura 2.3.

Gli IXP sono una componente chiave dell'ecosistema Internet e rappresentano un'opportunità fondamentale per migliorare l'affidabilità e la qualità delle connessioni nelle comunità locali. Sono infatti solitamente diffusi sui vari territori nazionali per consentire alle reti locali di scambiarsi dati eliminando la necessità di appoggiarsi a operatori esteri. Infatti gli IXP scambiano traffico in modo simile a quanto fanno gli aeroporti nazionali e regionali con i passeggeri: così come le linee aeree si scambiano i viaggiatori domestici in punti strategici sul territorio nazionale, piuttosto che in aeroporti internazionali esteri, così gli IXP instradano il traffico locale e regionale localmente piuttosto che attraverso reti internazionali. Mano a mano che le nazioni, le metropoli e le città si dotano di propri IXP, sempre più traffico viene scambiato e instradato localmente, riducendo così i costi e i ritardi della rete, aumentando le velocità di accesso ai contenuti e incoraggiando la crescita e la diffusione dei fornitori di servizi digitali locali. [17]

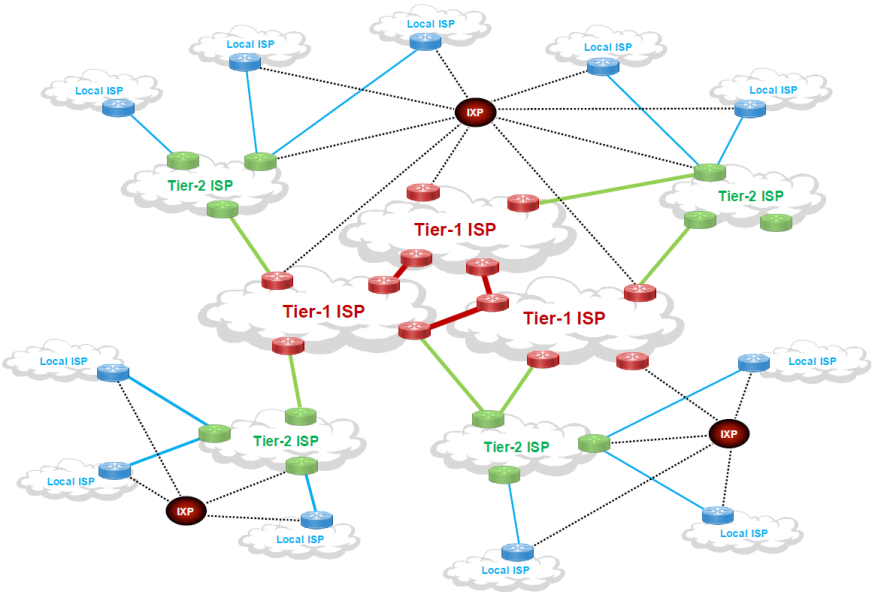

Figura 2.3. Il modello gerarchico delle entità su Internet

2.7.1 Il Consorzio TOP-IX

TOP-IX (TOrino Piemonte Internet eXchange) è un consorzio senza fini di lucro nato nel 2002 con lo scopo di creare e gestire un IXP per lo scambio del traffico Internet nell'area del Nord Ovest d'Italia. Oltre a costruire e gestire l'infrastruttura di rete per fornire i servizi tipici di un IXP, promuove e supporta, attraverso il Development Program, progetti di innovazione tecnologica e di business basati sull'utilizzo di Internet a banda larga.

I servizi di interconnessione offerti dal Consorzio TOP-IX ai suoi consorziati consentono l'implementazione di accordi di peering pubblico o privato per l'interscambio di traffico Internet tramite l'utilizzo dei protocolli BGP4 (IPv4) e BGP4+ (IPv6): i primi sono accordi gratuiti che prevedono l'utilizzo di VLAN (dette di peering pubblico) propagate a tutti gli afferenti e sulle quali non è ammesso il transito; i secondi sono accordi privati, implementati su VLAN dedicate e propagate esclusivamente ai soggetti interessati all'accordo stesso.

A differenza della concezione classica di IXP che solitamente risulta costituito da un'unica location ove sono dislocati gli apparati di accesso, l'infrastruttura di interscambio offerta dal Consorzio ai propri membri è geograficamente distribuita sul territorio: essa è costituita da diversi nodi interconnessi in fibra ottica e classificati come *core, backbone* ed *edge* in base ai relativi livelli di robustezza, affidabilità e prestazioni. La capillare presenza territoriale di nodi backbone ed edge consente ai consorziati di poter accedere all'infrastruttura semplicemente collegando la propria rete al nodo più vicino.

Da quanto descritto si evince che le caratteristiche di un Internet Exchange in generale, e la specifica topologia dell'infrastruttura del Consorzio TOP-IX in particolare, sono perfettamente adeguate per l'attivazione di un sistema distribuito che coinvolga le reti terminali degli afferenti e i nodi centrali della rete di backbone.

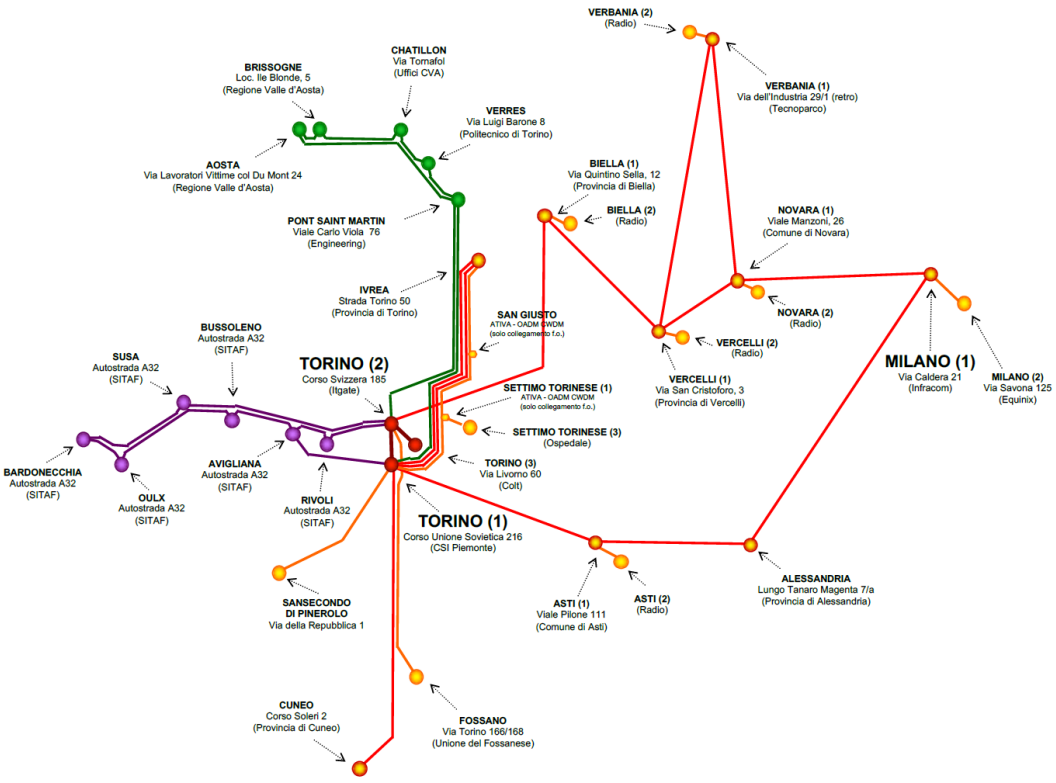

Figura 2.4. L'infrastruttura del Consorzio TOP-IX

Capitolo 3

Intrusion Detection Systems

Con l'aumento continuo nella banda disponibile e delle minacce di sicurezza, lo studio degli IDS sta ricevendo molta attenzione da parte della comunità scientifica e del mondo informatico in generale. Prima di tutto è necessario fare una chiara distinzione tra intrusione, rilevamento dell'intrusione, IDS e IPS. L'intrusione è il tentativo di compromettere la cosiddetta CIA, ovvero la terna delle caratteristiche di sicurezza del dato (Confidenzialità, Integrità e Disponibilità), oppure di bypassare i meccanismi di sicurezza di un computer o di una rete; il rilevamento dell'intrusione è il processo di monitoraggio degli eventi in un sistema informatico o in una rete, analizzandoli alla ricerca di tracce di un'intrusione; l'Intrusion Detection System è il software o l'hardware in grado di automatizzare il processo di rilevamento dell'intrusione; infine l'*Intrusion Prevention System* (IPS) è il sistema dotato delle caratteristiche di un IDS e in grado di fermare possibili intrusioni.

3.1 Tassonomia degli IDS

Per capire nel dettaglio il funzionamento degli IDS è necessario fare una classificazione dei sistemi in base alle metodologie di analisi, agli approcci di rilevamento e alle tecnologie: il prossimo capitolo naviga con metodo all'interno della tassonomia dei sistemi, di cui si fornisce un'utile panoramica nella figura 3.1, approfondendo gli aspetti più rilevanti per ogni tecnica. Ognuna, infatti, evidenzia le sua superiorità e limitazioni rispetto alle altre, e metodi diversi rispondo a diverse esigenze e contesti di utilizzo. Il lavoro sviluppato in questa tesi, pur senza inserirsi rigidamente all'interno di una specifica categoria, ha beneficiato da questa analisi per comprendere e definire i binari su cui orientare l'architettura del sistema.

3.1.1 Metodologie di analisi

Le metodologie di analisi delle intrusioni sono classificate in tre grandi categorie: *signature-based detection*, *anomaly-based detection* e *stateful protocol inspection*.

Signaure-based

Una firma (signature) è un pattern o una stringa che corrisponde a un attacco o a una minaccia conosciuti. È il processo per comparare i pattern con gli eventi catturati al fine di rilevare, in caso di compatibilità, possibili intrusioni. La tecnica è conosciuta anche come *knowledge-based detection* in quanto sfrutta la conoscenza accumulata da attacchi passati e vulnerabilità dei sistemi. I vantaggi di questa tecnica sono la semplicità e l'efficacia con attacchi conosciuti, così come l'analisi contestuale dettagliata; al contrario, non è efficace nel rilevamento di attacchi sconosciuti, di evasione o variazioni di attacchi noti, non sfrutta la conoscenza dei protocolli e richiede un costoso e complicato aggiornamento costante del database delle firme.

Anomaly-based

Un'anomalia (anomaly) è una deviazione da un comportamento noto, ovvero da un profilo che rappresenta il comportamento normale o atteso derivato dal monitoraggio delle attività regolari, delle connessioni di rete, degli host e degli utenti in un determinato periodo di tempo. I profili possono essere statici o dinamici e possono essere generati per diversi attributi, come i tentativi di accesso falliti, l'utilizzo del processore, il conteggio di mail inviate ecc. Quindi il processo confronta i profili normali con gli eventi osservati per riconoscere attacchi significativi. La tecnica è conosciuta anche come *behavior-based detection*, basandosi appunto sull'analisi comportamentale. È particolarmente adeguata per rilevare vulnerabilità nuove o impreviste, non dipende strettamente dal sistema operativo dei dispositivi e facilita il rilevamento di abuso dei privilegi; d'altro canto, è soggetta al rischio di costruire profili "deboli" a causa delle continue variazioni degli eventi osservati e non è efficace in fase di ricostruzione dei profili.

Stateful protocol analisys

"Stateful" indica che l'IDS può accedere e tenere traccia degli stati del protocollo (es. abbinando richieste e risposte). Sebbene la tecnica appaia simile all'anomaly-based, sono diverse nella sostanza: la seconda fa uso di profili precaricati specifici della rete o dell'host, mentre la prima dipende dal profilo generico sviluppato dal vendor di un apparato per uno specifico protocollo. Generalmente i modelli dei protocolli di rete sfruttati si basano sugli standard protocollari originali sviluppati dai vari enti (es. IETF, IEEE): per questo motivo si parla anche di *specification-based detection*. La tecnica, basandosi sugli stati dei protocolli, riconosce le sequenze impreviste di comandi; tuttavia è esosa in termini di consumo delle risorse, non è efficace contro attacchi che simulano un comportamento benigno del protocollo e potrebbe essere incompatibile con sistemi operativi o applicazioni specifici.

3.1.2 Approcci di rilevamento

Tradizionalmente gli studi sul rilevamento delle intrusioni partono da due macroscopici punti di vista, il rilevamento delle anomalie e quello degli abusi, pur senza evidenziare grosse differenze nei risultati. Liao et al. [18] hanno invece individuato cinque sottoclassi

che ben coprono i possibili approcci al rilevamento delle intrusioni: *statistics-based*, *pattern-based*, *rule-based*, *state-based* e *heuristic-based*; a questi si aggiunge il relativamente nuovo approccio *information-theoretical based* discusso da Eimann [19].

Statistics-based

Questo tipo di approccio si basa su soglie predefinite, media e deviazione standard, e probabilità di identificare le intrusioni. La letteratura presenta studi sulla distanza euclidea (perfetti per una risposta in tempo reale), sui metodi bayesiani (ottimali nella definizione del modello statistico) e sulla teoria dei giochi (lasciano poco controllo in quanto è forte la componente di auto-apprendimento). La metodologia più comune in questo caso è quella basata sulle anomalie: i parametri monitorati spaziano dai profili utente, all'utilizzo delle risorse (disco e memoria), il traffico di rete, gli eventi di sistema e gli errori.

Pattern-based

Questo approccio si focalizza sugli attacchi conosciuti mediante corrispondenza di stringhe. La metodologia più comune è quella basata sulle firme: vengono monitorati i record contenuti nei log, nei file di configurazione, nei file di sistema, nei file utente ecc. alla ricerca di firme di attacchi conosciuti. Gli studi si concentrano sul pattern matching (semplice ma poco flessibile), le reti di Petri (che producono una descrizione grafica intuitiva), il monitoraggio della battitura (sfruttando i pattern di digitazione dell'utente) e il controllo del file system (verificando l'integrità dei file).

Rule-based

Con questo approccio il modello e il profilo delle intrusioni conosciute sono costruiti mediante regole del tipo if-then o if-then-else. La ricerca si concentra sul data mining (in grado di generare automaticamente i modelli), la generazione manuale di modelli/profili (varie tecniche disponibili) e le support vector machine (basso tasso di falsi positivi e alta accuratezza). L'approccio è compatibile sia con il rilevamento basato su firma che sulle anomalie: prevede l'individuazione di pattern di regole per i profili utente e le policy, l'analisi dei dati e la generazione di una banca dati per la scoperta delle regole di associazione.

State-based

In questo caso vengono sfruttati le macchine a stati finiti originate dal comportamento della rete per identificare gli attacchi. L'approccio prevede tecniche di analisi delle transizioni tra stati (garantiscono una buona flessibilità), l'identificazione delle intenzioni dell'utente (riconoscono pattern per le interazioni ad alto livello), il modello del processo markoviano (probabilistico e basato sull'autoapprendimento) e l'analisi protocollare (basso tasso di falsi positivi ma poco efficace). Queste tecniche rientrano nel rilevamento basato prevalentemente sulle anomalie e, nell'ultimo caso, sull'analisi protocollare: le fonti sono i diagrammi di transizione tra stati per attacchi conosciuti, le sequenze di system call o comandi, i file di log e il modello di utilizzo normale di un protocollo.

Heuristic-based

Questo approccio si ispira ai concetti biologici e all'intelligenza artificiale. Sfrutta le reti neurali (basate sull'autoapprendimento e tolleranti ai guasti), la logica fuzzy (configurabile, scalabile e flessibile), gli algoritmi genetici (apprendimento euristico ed evolutivo), i sistemi immuni (distribuiti e dotati di alti livelli di sicurezza) e l'intelligenza di sciame (ispirata al mondo animale). Risponde prevalentemente ai requisiti del rilevamento di anomalie e osserva le sequenze di comandi, predice gli eventi, monitora il traffico di rete (TCP/UDP/ICMP), confronta pattern binari e scansiona i file di log.

Information-theoretical based

Questo è un approccio relativamente nuovo che ha acquisito popolarità negli ultimi anni. La sua strategia di rilevamento si basa sulla stima della quantità di informazione trasportata in un flusso di rete: questo approccio acquisisce campioni dal flusso (sia sotto forma di flow record per TCP che come semplici sequenze di byte per protocolli arbitrari) e ne stima il contenuto informativo in due modalità:

- calcolando il numero di passi elementari richiesti per costruire la stringa campione (misura della complessità);
- calcolando la frequenza con cui i nuovi pattern compaiono nella stringa campione (misura dell'entropia).

La tecnica di misura della complessità assume che i campioni contengano una quantità di informazione indicativamente costante durante la normale operatività. La tecnica di misura dell'entropia, invece, assume che la frequenza con cui i nuovi pattern compaiono nei campioni sia costante. L'approccio information-theoretical per il rilevamento di eventi di rete interpreta le variazioni improvvise nella complessità o nell'entropia come indicatori di eventi di rete. Sia gli attacchi di rete che i guasti agli apparati possono provocare queste variazioni. Il range di eventi rilevabili è quindi, almeno in principio, lo stesso rispetto all'approccio statistico. Tuttavia potrebbe diventare anche più ampio, in quanto la scelta di parametri osservabili può coprire tutte le firme dei pacchetti, abilitando inedite correlazioni tra parametri in grado di influenzare il risultato.

3.1.3 Tecnologie

Al giorno d'oggi esistono molti tipi di tecnologie di IDS. Categorizziamo le tecnologie in quattro classi in base al luogo in cui i sistemi sono installati per ispezionare le attività sospette e al tipo di eventi che sono in grado di riconoscere: *Host-based IDS* (HIDS), *Network-based IDS* (NIDS), *Wireless-based IDS* (WIDS), *Network Behavior Analisys* (NBA) e *Mixed IDS* (MIDS), una tecnologia ibrida tra le precedenti.

I componenti di un IDS includono agenti e sensori, dove i primi vengono usati tipicamente nella prima tecnologia e i secondi nelle altre. Sia i sensori che gli agenti possono inviare dati a un Server di Gestione e un Server di Database, dove il primo è un dispositivo centralizzato per il processamento degli eventi catturati, mentre il secondo è solo un archivio per lo storage delle informazioni relative agli eventi. Inoltre troviamo due principali tipi di architetture di rete: una rete dedicata per la gestione del software di sicurezza per celare

la presenza dell'IDS agli intrusi, separata dalla rete principale e per questo motivo costosa in termini di installazione e gestione; la rete principale, condivisa e senza protezione, su cui transitano sia il traffico normale che quello "di sicurezza" funzionale all'IDS, la cui sicurezza può essere aumentata con la configurazione di una VLAN dedicata.

Un difetto comune a tutte le tecnologie di IDS è l'impossibilità di fornire un rilevamento accurato al 100%. I falsi positivi e i falsi negativi sono due indicatori del grado di accuratezza: i primi si verificano quando l'IDS identifica erroneamente attività benigne come maligne, mentre i secondi si verificano quando l'IDS fallisce nell'identificare le attività maligne. Molti amministratori di sicurezza privilegiano tenere alto il tasso di falsi positivi a discapito di quello di falsi negativi per mantenere alto il livello di sicurezza anche a costo di un calo di performance nei servizi. Statisticamente si nota che la maggior parte di errori si verificano con falsi negativi a causa della non conformità agli standard di molti protocolli usati a livello applicativo.

Un'ultima interessante tassonomia degli IDS è quella basata sull'architettura del sistema di elaborazione: troviamo architetture centralizzate, che raccolgono e analizzano i dati da un singolo sistema monitorato; architetture distribuite, che raccolgono dati da diversi sistemi monitorati in modo da rilevare attacchi singoli, distribuiti e cooperativi; e architetture ibride che combinano i due precedenti casi.

Host-based IDS

Un NIDS monitora e raccoglie le caratteristiche degli host che contengono informazioni sensibili, server che erogano servizi pubblici, e attività sospette. Si basa su agent di tipo software installati sul singolo host e per questo motivo è l'unica tecnologia di IDS in grado di analizzare le attività di comunicazione con cifratura end-to-end. Tuttavia la mancanza di informazioni sul contesto rende più ardua una rilevazione accurata, la condivisione di risorse computazionali con l'host può causare ritardi nella generazione degli allarmi e dei report, e la coabitazione nello stesso sistema operativo può creare conflitti con i sistemi di sicurezza preesistenti. Le informazioni più rilevanti catturate sono il traffico di rete, le system call e l'attività del file system.

Network-based IDS

Un NIDS monitora il traffico in certi specifici segmenti di rete attraverso sensori e, come conseguenza, analizza le attività delle applicazioni e dei protocolli per riconoscere incidenti sospetti. Si basa su sensori attivi o passivi installati nei vari segmenti di rete e sugli host in grado di analizzare i protocolli applicativi con la massima visibilità sul contesto. Gli svantaggi sono l'incapacità di monitorare i protocolli wireless, l'alto tasso di falsi positivi e falsi negativi, l'incapacità di rilevare attacchi a partire dal traffico cifrato e la mancanza di supporto all'analisi estensiva in caso di picchi di carico. Le informazioni raccolte riguardano essenzialmente gli host, i sistemi operativi, le applicazioni e il traffico di rete.

Wireless-based IDS

Un WIDS e simile a un NIDS, ma cattura il traffico wireless nelle reti ad-hoc, nelle reti di sensori e nelle reti mash. È costituito da vari sensori passivi installati sui nodi della

WLAN e sui client wireless rendendo il sistema particolarmente accurato grazie al suo focus selettivo sull'attività dei protocolli wireless. Raccoglie dati dalla WLAN, dai vari dispositivi connessi (access point, controller ecc.) e dai client terminali; non è però adatto a monitorare i livelli applicativo, trasporto e rete, non è resistenze ai tentativi di evasione, i sensori sono suscettibili ad attacchi fisici di interferenza, e non può compensare all'uso di protocolli wireless non sicuri.

Network Behavior Analisys

Un sistema NBA ispezione il traffico di rete per riconoscere attacchi a partire da flussi di dati inaspettati. Si basa su sensori prevalentemente passivi installati nella rete, è dotato di una capacità di rilevamento avanzata nelle scansioni di perlustrazione, ed è in grado di ricostruire le dinamiche delle infezioni da malware e degli attacchi DDoS. La principale limitazione è il ritardo nel rilevamento degli attacchi causato dall'elaborazione dei flussi di dati sotto forma di blocchi e non in tempo reale. Le principali fonti di informazioni sono gli hosti, i sistemi operativi e i servizi (IP, TCP, UDP ecc.).

3.2 Il protocollo SNMP

Il Simple Management Network Protocol (SNMP) è un protocollo di livello applicativo usato per gestire e monitorare gli apparati di rete e le loro funzionalità. SNMP fornisce un linguaggio comune ai dispositivi di rete per trasferire informazioni di gestione in ambienti single-vendor o multi-vendor su LAN o WAN. Essendo molto diffuso, SNMP è supportato da un ampio spettro di dispositivi, dagli apparati di rete tradizionali come swith, router e access point ai terminali come PC e stampanti ai più recenti dispositivi rientranti nel campo dell'IoT. Oltre all'hardware, SNMP può essere sfruttato per monitorare servizi come DHCP, HTTP, FTP ecc.

Vi sono tre componenti principali in una rete gestita mediante SNMP:

- **SNMP agent**: è il programma in esecuzione sull'hardware o sul servizio monitorato che raccoglie dati relativi a diverse metriche, come la larghezza di banda o l'utilizzo del disco. Quando è interrogato dall'SNMP manager, l'agent risponde con i dati raccolti localmente. Un agent può anche lavorare proattivamente notificando al manager il verificarsi di errori (trap mode). La maggior parte degli apparati è nativamente dotata di un agent SNMP pre-installato, il quale richiede solo una basilare configurazione.
- **SNMP manager**: conosciuto anche come Network Manager System (NMS), è una piattaforma software che agisce come console centralizzata verso cui gli agent inviano i dati. Richiede ai vari agent di inviare i dati raccolti a intervalli regolari. Ciò che un NMS è in grado di fare con i dati ricevuti dipende dalle funzionalità del sistema.
- **Management Information Base** (MIB): è un database, sotto forma di file testuale, contenente l'elenco di tutti i componenti disponibili su un determinato dispositivo che possono essere interrogati e controllati usando SNMP. Questo database deve essere caricato sul NMS così che possa identificare e monitorare lo stato dei vari oggetti. A ciascun elemento della MIB è assegnato uno specifico Object Identifier (OID).

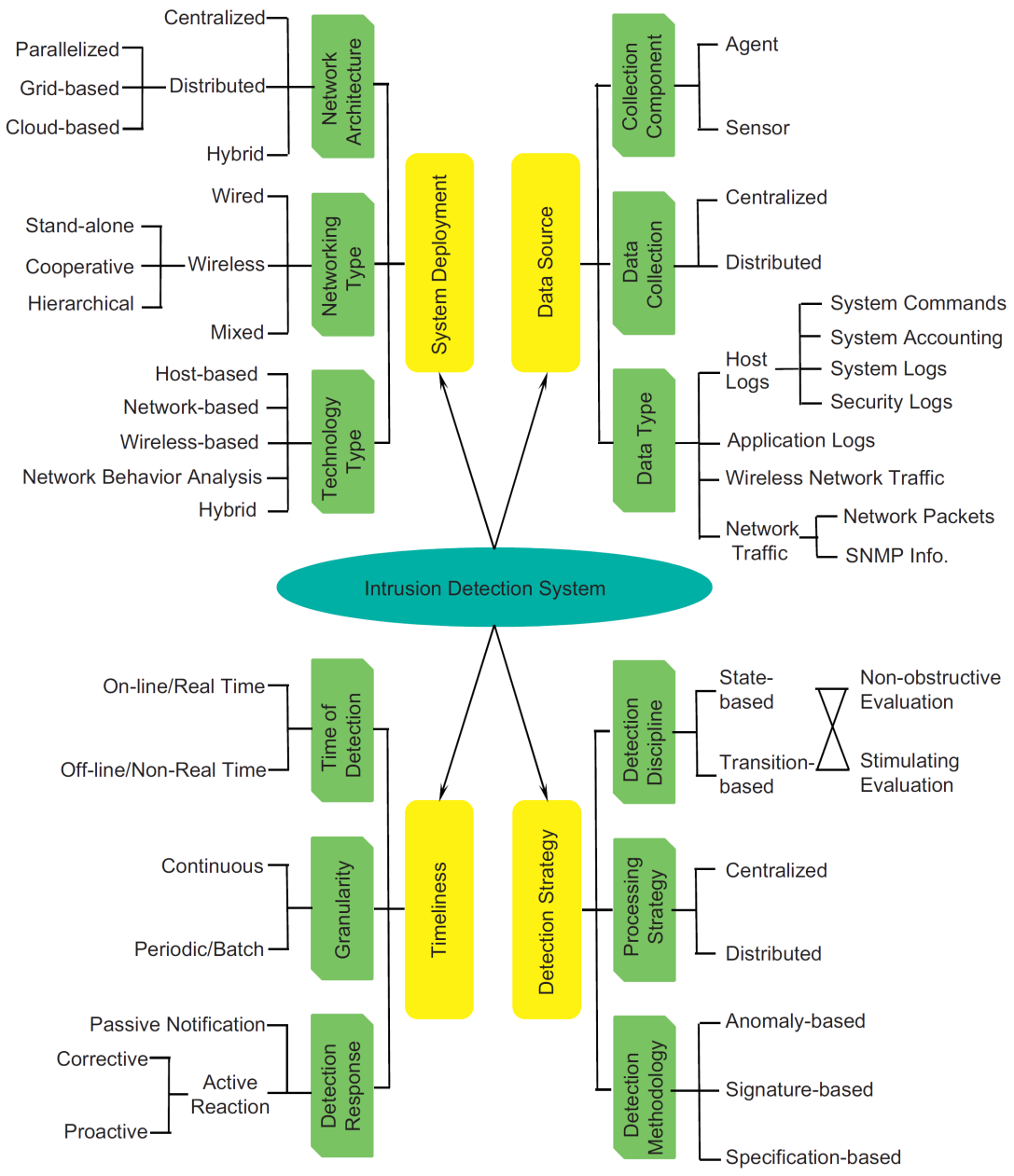

Figura 3.1. Una panoramica sulla tassonomia degli IDS

Il protocollo SNMP svolge molte funzioni, e si basa su un misto di comunicazioni di tipo push e pull tra gli apparati di rete e il sistema di gestione. Può inviare e ricevere comandi, come il reset di una password o la modifica di un parametro di configurazione. Può restituire informazioni sull'utilizzo corrente di banda, CPU, memoria ecc. e alcuni SNMP

manager possono inviare mail e notifiche agli amministratori in caso alcune soglie vengano superate. Nella maggior parte dei casi SNMP funziona secondo un modello sincrono, con le comunicazioni avviate dal manager e le risposte provenienti dagli agent. Classifichiamo le PDU del protocollo in base al tipo di messaggio trasportato:

- **GET**: generata dal manager e inviata a un agent per ottenere il valore di una variabile identificata da uno specifico OID all'interno di una MIB;
- **RESPONSE**: inviata dall'agent al manager in risposta ad una GET, contenente il valore della variabile richiesta;
- **GETNEXT**: inviata dal manager all'agent per richiedere il valore della variabile associata all'OID successivo all'interno della gerarchia della MIB;
- **GETBULK**: inviata dal manager all'agent per ottenere i valori di una serie di variabili con un'unica richiesta, equivalente all'esecuzione di diversi GETNEXT;
- **SET**: inviata dal manager all'agent per impostare un parametro o eseguire un comando;
- **TRAP**: notifica asincrona inviata dall'agent al manager per segnalare un evento significativo, come un errore o un guasto.

Le versioni 1 e 2c del protocollo non prevedono un meccanismo di autenticazione forte, ma ne forniscono uno elementare basato sulle community: sono stringhe testuali, inviate a corredo di ogni comando, che definiscono l'appartenenza di agent e manager allo stesso gruppo di dispositivi. Esistono community di tipo *monitor*, per accesso in sola lettura, *control*, per accesso in lettura e scrittura e *trap*, per i messaggi del relativo tipo. Ogni agent può appartenere a più gruppi, e quindi accettare più community. La più recente implementazione di SNMP, la versione 3, include vari miglioramenti di sicurezza che consentono l'autenticazione e la cifratura dei messaggi e dei pacchetti.

3.2.1 SNMP per l'Anomaly Detection

La sua ampia diffusione, la compatibilità e la grande varietà di informazioni in grado di raccogliere, fanno di SNMP un protocollo particolarmente adeguato al ruolo di sorgente di dati in un NIDS basato su anomaly detection. Negli ultimi dieci anni molti ricercatori hanno compiuto diversi studi sul rilevamento di anomalie e attacchi su reti informatiche. La maggior parte delle soluzioni individuate fin'ora si basano sull'analisi delle caratteristiche del traffico a basso livello (il numero di pacchetti, gli indirizzi IP, le porte, il flusso ecc.), cosa che quasi sempre richiede la cattura dei pacchetti. Parallelamente sono state proposte alcune (poche) soluzioni basate su SNMP, con approcci orientati sia all'analisi statistica che al machine learning.

Tra i lavori più rilevanti si segnalano quelli di: Cabrera et al. [20], sul rilevamento preventivo di attacchi DDoS mediante estrazione casuale e analisi statistica di variabili MIB come precursori; Yu et al. [21], per un NIDS che sfrutta il machine learning basato su una Support Vector Machine (SVM) operante sui dati prodotti tramite SNMP; Cerroni at al. [22], i quali hanno presentato un sistema decentralizzato basato sul peer-to-peer per la raccolta di dati statistici da SNMP e il successivo data mining (clustering basato su K-means); Priya

et al. [23], che si sono concentrati sul rilevamento di Distributed Reflection DoS (DNS attack e TCP SYN reflection flooding attack) mediante classificazione con algoritmo C4.5 e successiva correlazione del rango.

Si osserva che la maggioranza dei lavori citati si limitano a casi specifici e a un numero limitato di attacchi. Inoltre, alcuni fra questi studi si concentrano sul rilevamento di traffico anomalo in quanto distinto da quello normale, senza l'interesse a determinare il tipo di attacco. Questo dimostra che ci sono ancora ampi margini per lo sfruttamento dei dati derivati da SNMP nella gestione della sicurezza su reti informatiche. Ghazi et al. [24] hanno recentemente proposto una soluzione innovativa per l'anomaly detection in grado di classificare diversi tipi di attacchi (TCP-SYN e UDP flood, ICMP Echo, HTTP flood, Slowpost, Slowloris, Brute Force ecc.), anch'essa basata sul machine learning.

L'idea sviluppata in questa tesi, invece, si pone ad un livello più alto e sfrutta SNMP come primo approccio di indagine per il rilevamento di intrusioni. Alcune fra le variabili MIB disponibili, infatti, si sono rivelate degli ottimi indicatori di possibili anomalie, e il valore delle informazioni cui sono associate può essere sfruttato direttamente, anche prima degli eventuali step successivi di data mining e machine learning.

Capitolo 4

Stato dell'arte

In questo capitolo vengono presentati alcuni lavori e studi che sono stati compiuti sull'ambito di ricerca affrontato in questo lavoro di tesi, insieme con alcuni prodotti commerciali già disponibili sul mercato con cui il sistema progettato va a confrontarsi. Altri lavori pregressi vengono presentati nei capitoli successivi, in quanto si ritiene più coerente ai fini della comprensione della ricerca svolta.

4.1 Digital City Exchange

Digital City Exchange (DCE) è un programma di ricerca nell'ambito della digital economy con l'obiettivo di promuovere nuove opportunità tecnologiche e di business, connettendo servizi e utility all'interno delle città. [25] Il programma è sviluppato all'Imperial College London e ha ricevuto un finanziamento quinquennale per £5.9 milioni dal Research Councils UK's Digital Economy Programme.

4.1.1 Il contesto

Il programma di ricerca si inserisce in un contesto di innovazione dei sistemi urbani: per la prima volta si assiste alla necessità di mettere a fattor comune i vari servizi che definiscono il sistema urbano, molto spesso evolutosi negli anni senza un vero e proprio piano strategico. L'assenza di connessioni reciproche o di adeguate configurazioni è il risultato di sviluppi indipendenti e talvolta concorrenziali, sia sul piano economico che tecnologico e legale. La conseguenza più evidente, soprattutto per il cittadino, è l'incapacità del sistema di reagire a situazioni anomale come i picchi associati alla vita cittadina (trasporti pubblici, energia ecc.)

Questa situazione può essere vista come un'opportunità per l'attivazione di una serie di tecnologie innovative e potenziali, sia digitali che fisiche: reti dinamiche reattive, rilevazione pervasiva dei dati, modellazione su larga scala, nuove tecniche di analisi e di ottimizzazione, tecnologie web, IoT ecc. Tutto questo richiede la nascita di nuovi modelli di business studiati per incoraggiare l'apertura e la disponibilità dei dati a fronte del valore aggiunto che la gestione coordinata di questi dati può produrre.

4.1.2 Ambiti di ricerca

La ricerca sul DCE è trasversale e tocca gli ambiti di ingegneria, analisi dei sistemi, innovazione e uso dei modelli di business, analisi della sicurezza dei dati e della privacy con l'obiettivo di creare e analizzare modelli per la fornitura di servizi integrati. Nello specifico il programma prevede sei attività elementari:

a. Creazione di uno standard per i metadati e di un modello di sintesi intelligente.

b. Casi di studio di adozione dei servizi ed esperimenti sul comportamento degli utenti nei confronti dell'innovazione sistemica di trasporti, energia e altri settori.

c. Implementazione pilota dell'architettura di analisi, modellazione e servizio.

d. Sviluppo di strumenti per la gestione dell'innovazione e la creazione di modelli di business.

e. Sviluppo di un meccanismo di prototipazione virtuale per i nuovi servizi digitali e per la loro attivazione basato sui dati raccolti nelle altre attività.

f. Misurazione delle performance dei servizi e dell'impatto su utenti, aziende e sistemi urbani.

Gli obiettivi della ricerca possono essere riassunti in tre ambiti principali: la gestione dei picchi (garanzie di sicurezza, robustezza e resilienza), lo sfruttamento di data set eterogenei (integrazione dei flussi di dati al di là dei loro confini tradizionali) e l'innovazione dei servizi (in chiave globale, di sistema urbano dotato di risorse e servizi).

4.1.3 Concinnity

Tra i contesti applicativi basati sul programma DCE c'è la piattaforma Concinnity. [26] Concinnity si basa su lavori preesistenti nella gestione di dati derivati da sensori e nella progettazione di sistemi di gestione del flusso di lavoro per la composizione di modelli. La piattaforma si basa su cinque punti chiave: collaborazione, eterogeneità dei dati, integrazione a risoluzioni e scale diverse, gestione dell'incertezza e dell'affidabilità dei dati, modellazione e processo decisionale.

Lo schema architetturale della piattaforma prevede una gerarchia a tre livelli: un livello di raccolta e immagazzinamento dei dati provenienti da diverse sorgenti e in diversi formati, caratterizzato da un design schema-less e da un massiccio uso dei metadati (basato sul progetto WikiSensing [27]); un livello di esecuzione del flusso di lavoro che supporti il processamento dei dati e l'integrazione dei modelli a scale diverse (basato sul motore HierSynth [28]); un livello di sviluppo e pubblicazione di applicazioni che supporti la composizione dei dati e dei modelli per generare nuovi flussi di dati e attivare nuovi servizi (un esempio è mostrato nella figura 4.1).

Uno scenario di applicazione della piattaforma Concinnity riguarda lo studio dell'impatto dei veicoli elettrici sulla rete elettrica cittadina. [29] In questo esempio il flusso di lavoro si basa su tre modelli: un modello di attività degli agenti (che genera percorsi di una popolazione sintetica di utenti su veicoli elettrici), un modello che traduce tale attività in richieste di energia contestualizzate nel tempo e nello spazio, e un modello di ottimizzazione

dei profili di carico per ogni stazione (che simula la rete di distribuzione e ne definisce la configurazione ideale per rispondere alle richieste di energia).

I risultati hanno dimostrato diversi vantaggi: la piattaforma consente un'esecuzione veloce della catena di modelli, facilita la configurazione di nuovi scenari di analisi e permette l'inserimento di altri modelli all'interno del flusso di lavoro per migliorarlo o estenderlo. Inoltre la piattaforma rende possibile la realizzazione di nuovi servizi basati sul flusso di lavoro, tra cui strumenti a supporto del processo decisionale, dashboard o app mobile.

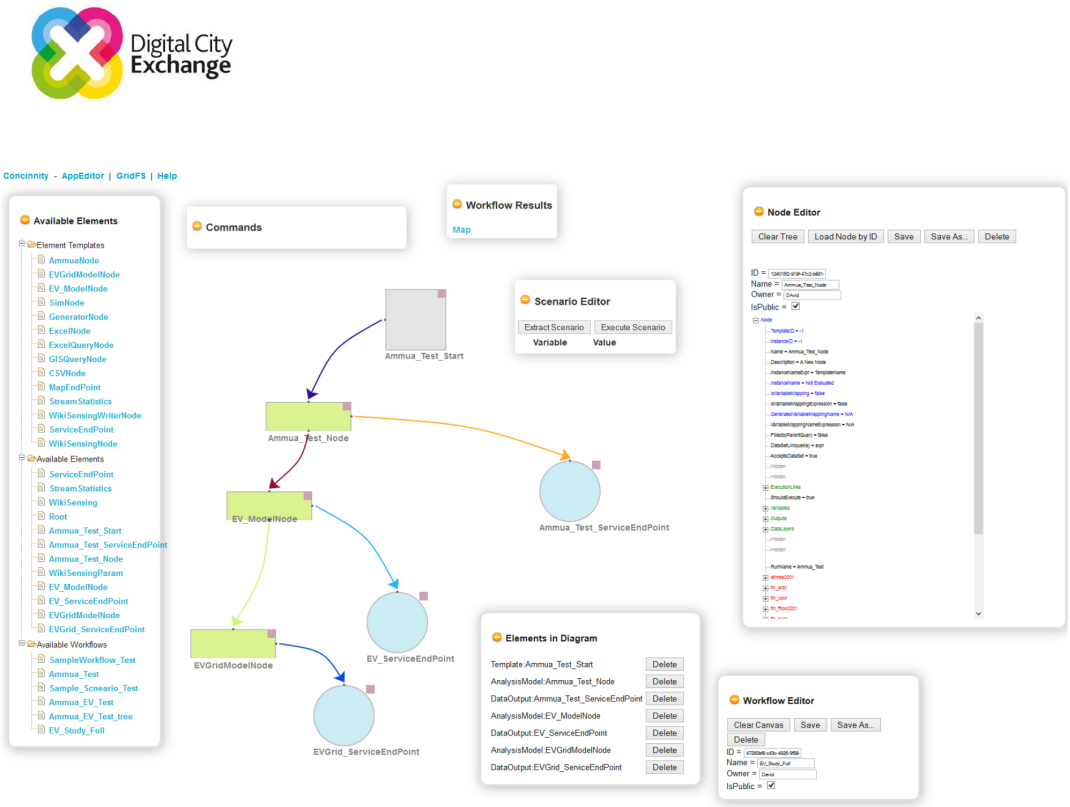

Figura 4.1. Schermata dell'AppEditor della piattaforma Concinnity

4.1.4 Problematiche aperte

Gli studi compiuti all'Imperial College London sul programma Digital City Exchange hanno mostrato il valore che l'analisi congiunta dei dati prodotti da soggetti eterogenei operanti a livello territoriale può produrre, soprattutto se contestualizzati, normalizzati e messi a disposizione su una piattaforma comune. Se è vero che la piattaforma DCE è stata pensata per un ambito più spiccatamente "big data", in cui la correlazione e la successiva interpretazione dei dati abilita processi innovativi rivolti a migliorare l'efficienza, la disponibilità e la ricchezza dell'offerta, è comunque chiaro che l'architettura sia valida per la gestione di vari tipi di dati, tra cui quelli acquisiti mediante monitoraggio di sicurezza.

Dal punto di vista del lavoro di tesi descritto in queste pagine, l'architettura della piattaforma Concinnity si è rilevata potenzialmente adeguata per estendere le funzionalità del sistema abilitando un modello decisionale basato non più solo sui dati prodotti dal singolo soggetto per cui viene rilevata l'anomalia, ma anche su quelli prodotti da altri soggetti connessi alla stessa infrastruttura. Le anomalie di rete, infatti, spesso sono sintomo di problemi più vasti il cui veicolo di diffusione può essere individuato nelle infrastrutture di interconnessione, tra cui gli Internet Exchange: osservare le anomalie da un punto di vista più esteso consente di individuare altri problemi apparentemente sconnessi ma potenzialmente rilevanti per identificarne la comune sorgente.

Lo sviluppo del programma DCE, tuttavia, è rimasto fermo a livello architetturale e non ha ancora trovato una concreta messa in produzione. Concinnity rappresenta l'unica implementazione annunciata, ma il relativo codice sorgente non è attualmente pubblico e disponibile. Una scelta orientata all'open-source sicuramente consentirebbe una ripresa del progetto con la probabile individuazione di stakeholders interessati alla messa in servizio nei loro domini.

4.2 Prodotti commerciali

Nel lavoro di studio dello stato dell'arte si è ritenuto utile osservare alcuni dei prodotti già disponibili sul mercato che offrono meccanismi di Anomaly-Based Network Intrusion Detection: nessuno fra i prodotti individuati è stato appositamente progettato per l'infrastruttura condivisa di un Internet Exchange e tutti condividono la necessità progettuale di avere un impatto notevole sui sistemi monitorati.

aramis

aramis è un NIDS basato su Network Behavior Anomaly Detection (NBAD) sviluppato da aizoOn Group. aramis fa uso di una tecnologia di machine learining appositamente progettata, di tecniche di data mining e di capacità di thread intelligence per consentire alle aziende di identificare minacce sconosciute all'interno dei loro perimetri promuovendo l'analisi umana.

aramis si basa su un workflow di raccolta, arricchimento, correlazione e visualizzazione dei dati. Una serie di sonde di rete passive sono posizionate strategicamente in vari nodi della rete sulla base del throughput dei flussi monitorati e della quantità di informazioni trasferite. Ogni sensore raccoglie informazioni dal segmento di rete in cui è installato, le analizza in tempo reale e invia i risultati ad un server locale: i sensori possono essere configurati come "honey pot", ossia esche per attirare e rilevare più agilmente le minacce.

L'arricchimento dei dati avviene tramite la cyber intelligence: i dati ricevuti dalle sonde vengono arricchiti sul server locale con informazioni proveniente dall'OSINT e dalla Thread Intelligence, sfruttando altresì dati propri del dominio del cliente. L'arricchimento è completato dalla correlazione: aramis esegue costantemente due tipi di analisi sui dati acquisiti, ovvero la modulazione continua dei parametri basata sulla variazione dinamica del rischio misurato e l'analisi del comportamento di ogni singolo nodo attraverso il motore di AI.

La visualizzazione consente di identificare e reagire prontamente. Le informazioni elaborate sono rappresentate su una dashboard tramite un approccio di "visualizzazione cognitiva" che consente di evidenziare le minime deviazioni dai pattern ripetitivi; questi grafici forniscono agli analisti uno strumento potente per l'identificazione e l'indagine degli allarmi.

Allot DDoS Secure

Si tratta di uno strumento di protezione da attacchi DDoS e di contenimento delle minacce pensato per le reti di Service Provider. La protezione in tempo reale da DDoS è realizzata tramite la tecnologia NBAD in grado di identificare gli attacchi volumetrici a partire dalle anomalie che essi causano nel comportamento, solitamente tempo-invariante, delle statistiche sul rateo di pacchetti L3 e L4. Allot parallelamente ispeziona ogni pacchetto sulla rete per identificare firme note delle minacce conosciute, creando dinamicamente regole di mitigazione per filtrare il traffico malevolo senza compromettere la disponibilità dei servizi.

Tra i tipi di evento rilevati da Allot DDoS Secure vi sono l'alto rateo di pacchetti, la dimensione anomala dei pacchetti (molto piccoli o molto grossi), fan-in e fan-out, stati TCP invalidi (combinazioni inconsuete di SYN, ACK, FIN, RST), traffico ICMP, scansione degli indirizzi e delle porte, richieste massive a SMTP e DNS ecc. La reazione si basa sul filtro del traffico usando firme dei pacchetti dinamiche, la notifica degli utenti medianti redirect o email, la limitazione del throughput disponibile per il singolo utente e altre contromisure.

Arbor DDoS Attack Protection

Arbor offre una serie di protezioni in-cloud e on-premise per contrastare gli attacchi DDoS basandosi sulla thread intelligence.

Per le reti molto estese e gli attacchi DDoS più sofisticati il sistema consente di analizzare dati provenienti da Netflow, BGP e SNMP, così da ottenere una visibilità pervasiva della rete e rilevare rapidamente attacchi DDoS. Rilevato l'attacco, il sistema può instradare il traffico verso il servizio di Thread Management System installato in un centro di riciclaggio del traffico condiviso oppure verso un dispositivo specifico (es. router Cisco ASR9K) per la mitigazione fino a 400Gbps di ogni tipo di attacco DDoS. Per le reti più piccole Arbor APS è in grado di rilevare e mitigare attacchi DDoS fino a 40Gbps sia verso che a partire dalla rete del cliente.

Capitolo 5

Architettura del sistema

In questo capitolo viene descritta l'architettura del sistema progettato nel lavoro di tesi. Come anticipato nell'introduzione, per la sua definizione sono stati adottati alcuni strumenti già in uso presso il Consorzio TOP-IX, opportunamente configurati e modificati per abilitare le funzionalità richieste. Nei capitoli successivi viene presentata un'implementazione concreta dell'architettura e il suo funzionamento in un caso di studio.

L'idea presentata dal Consorzio TOP-IX si è tradotta nel progetto di un sistema di monitoraggio distribuito basato su un'architettura three-tier di tipo client-server, in grado di rilevare anomalie sulle reti terminali dei soggetti connessi all'Internet Exchange, produrre degli allarmi, arricchirli con quante più informazioni possibili sul contesto ed eventualmente fornire una prima basilare risposta.

5.1 Lo schema

L'architettura del sistema (figura 5.1) si articola attraverso tre moduli, ciascuno deputato ad una specifica funzione: il modulo di *acquisizione dei dati*, il modulo di *rilevamento delle anomalie* e il modulo di *elaborazione dei dati*. Il flusso delle informazioni non è lineare, ma coinvolge alternativamente i vari moduli a seconda della fase in cui il sistema si trova: nei prossimi paragrafi vengono presentati dettagliatamente, con particolare riferimento alle reciproche interazioni e al flusso dei dati attraverso di essi.

5.1.1 Acquisizione

Il modulo deputato all'acquisizione dei dati è installato all'interno della rete locale oggetto di monitoraggio: per affinità con lo strumento utilizzato nell'implementazione, d'ora in poi ci si riferirà a questo modulo come **proxy**. Concettualmente si tratta di un dispositivo connesso alla LAN come un qualsiasi altro host: non necessita di particolari procedure di deployment, ma può semplicemente ottenere la configurazione di rete tramite DHCP. Gli unici vincoli che richiede tale dispositivo sono la possibilità di raggiungere Internet e la visibilità sull'intero segmento di rete in cui si trova, ossia non devono esserci router tra di esso e i dispositivi monitorati.

Il proxy svolge prevalentemente attività di acquisizione dei dati dai singoli host e apparati presenti sulla rete locale, anche se il suo ciclo di vita si articola attraverso varie fasi, così come descritto nella sezione successiva dedicata al funzionamento. Ha il compito di "rappresentare" il modulo di rilevamento delle anomalie all'interno della LAN, agendo prevalentemente come *reverse proxy* (da qui il nome) tra l'esterno della rete e gli host monitorati: a differenza di un reverse proxy web, in cui i server web locali sono mascherati ai client su Internet per motivi di sicurezza, bilanciamento del carico, o performance, in questo caso sono i singoli host monitorati ad essere mascherati al resto del sistema (che si estende oltre la rete locale).

Dal punto di vista del sistema, il proxy agisce come una backdoor benigna all'interno della rete locale, consentendo agli altri moduli di superare le difese del firewall (ove presente) e avere così accesso a dispositivi e dati locali. Naturalmente non è possibile esimersi dal fare le dovute considerazioni sulla sicurezza di questa soluzione, e alcune riflessioni sono presentate nel capitolo dedicato all'implementazione del sistema.

5.1.2 Rilevamento

Il modulo che si occupa del rilevamento delle anomalie può essere installato in qualsiasi nodo di Internet, preferibilmente in una posizione quanto più vicina possibile alle reti terminali oggetto di monitoraggio: come per il proxy, anche in questo caso ci si riferirà a tale modulo come **server**. Concettualmente si tratta di un software installato su un server raggiungibile mediante indirizzo pubblico, cui si connettono i vari proxy installati nelle reti terminali. Il punto privilegiato per l'installazione del server è all'interno della rete dell'Internet Exchange, in modo tale che le reti interconnesse ne abbiamo visibilità diretta, potendo così evitare di esporlo su Internet.

Il server assume due ruoli a seconda della fase in cui si trova il sistema. Per la maggior parte del tempo agisce come controllore del sistema: coordina i vari proxy installati nelle reti locali, monitora i singoli host terminali, definisce le metriche con cui rilevare possibili anomalie, confronta l'attività degli host con soglie, genera allarmi in caso di problemi e comunica con il modulo di elaborazione dei dati.

Il secondo ruolo entra in gioco in caso di anomalia, cioè quando il server inizia la comunicazione con il modulo di elaborazione dei dati, presentato dettagliatamente in seguito: in questo caso il server affianca al suo ruolo di controllore quello di relay, ossia consente al modulo di elaborazione di interagire con tutte le altre componenti del sistema sfruttando le connessioni già aperte e controllate dal server stesso. Questo comportamento ha termine quando l'anomalia viene gestita, cosa che può tradursi in un allarme o in una basilare reazione alla minaccia.

5.1.3 Elaborazione

Il modulo deputato all'elaborazione dei dati risiede nella stessa posizione del server, in quanto ha necessità di comunicare solo con quest'ultimo. Per analogia con i casi precedenti, ci si riferirà a questo modulo come **backend**. Anch'esso è un componente software, e le sue funzioni sono quelle di apprendere il comportamento della rete e di intervenire nella gestione delle anomalie: se il rilevamento preliminare è affidato al server, è al backend che

compete la contestualizzazione del problema, l'indagine avanzata, e l'eventuale reazione correttiva; queste operazioni diventano particolarmente efficaci nel momento in cui sono chiari il comportamento della rete, i parametri che lo definiscono e i valori tipici delle informazioni prodotte dagli host.

Come anticipato, l'attività del backend può concludersi in due modalità differenti: attraverso l'emissione dell'allarme arricchito, che rientrerà solo in seguito a correzione manuale o quantomeno esterna al sistema, oppure attraverso l'attuazione di azioni correttive automatiche. Quest'ultimo caso è subordinato alla disponibilità da parte del soggetto gestore della rete terminale al consentire interventi dispositivi esterni, possibilità che esula dalle tipiche prerogative di un Intrusion Detection System e sconfina nel campo degli Intrusion Prevention System.

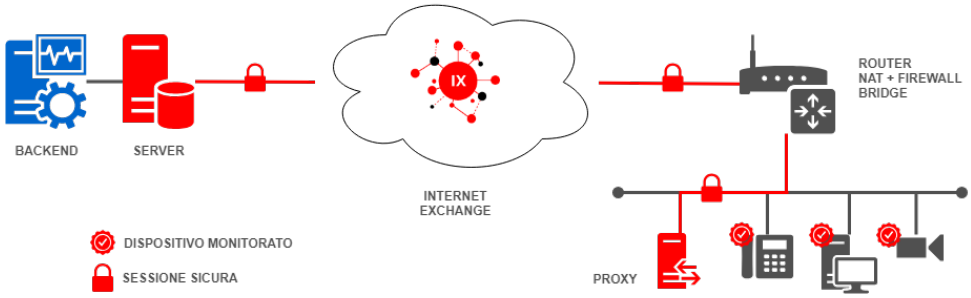

Figura 5.1. Lo schema architetturale del sistema

5.2 Il ciclo di vita

Il ciclo di vita del sistema si sviluppa attraverso otto fasi:

1. **Installazione**: è la fase iniziale, in cui il sistema viene attivato.

   Ogni nuova attivazione prevede l'installazione di un proxy all'interno delle rete locale che si desidera monitorare: il proxy ha una configurazione di base fissa, indipendentemente dalla topologia e dai parametri della rete in cui viene installato, ed è automaticamente in grado di stabilire una connessione crittografata con il server (sempre attivo e unico per tutte le installazioni). Nella maggior parte dei casi non è necessario intervenire sul firewall, in quanto la connessione sfrutta la porta TCP/80 tipicamente aperta per il traffico web.

   Potenzialmente, almeno nella versione del sistema più basilare e in un contesto di rete molto semplice, l'installazione del proxy può essere eseguita in modalità plug-and-play anche da personale non qualificato del soggetto gestore della rete: nel capitolo dedicato all'implementazione viene presentato un possibile caso di questo tipo.

2. **Configurazione**: è l'unica fase in cui l'operatore deve intervenire sui dispositivi.

   Una volta installato il proxy, questo deve essere impostato per la scoperta di tutti gli apparati sul segmento di rete: la configurazione consiste nella scelta di un set di

parametri da valutare per la rilevazione e l'identificazione degli host, e viene attuata da remoto, tramite il server, una volta stabilita la connessione.

Inoltre è in questa fase che eventualmente si interviene sui singoli host oggetto di monitoraggio: tale operazione consiste nell'attivazione di determinati servizi o l'installazione di applicazioni in grado di facilitare l'acquisizione di informazioni, in modo più o meno "invasivo", a seconda del livello di servizio concordato con il soggetto responsabile della rete.

3. **Scoperta**: è la fase in cui avviene la mappatura della rete.

   Dopo essere stato opportunamente configurato, il proxy inizia la scansione della rete locale alla ricerca di host: ogni nuovo host rilevato viene annunciato al server, il quale lo identifica in base al valore dei parametri precedentemente definiti e lo inserisce all'interno del sistema. L'accuratezza di questa fase è strettamente dipendente dal tipo di configurazione degli host.

   Durante il ciclo di vita la fase di scoperta viene eseguita periodicamente, in modo da registrare gli eventuali nuovi host non ancora presenti al momento dell'installazione. Tutte le informazioni acquisite sulla topologia della rete, il tipo di host e la loro configurazione sono memorizzati sia sul server che sul proxy: in questo modo, in caso di riavvio del server o del proxy, la configurazione diventa immediatamente disponibile.

4. **Interrogazione**: in questa fase avviene l'interrogazione degli host.

   L'interrogazione è attuata dal proxy per conto del server: consiste nell'interagire periodicamente con host e apparati con lo scopo di acquisire informazioni sul loro stato. L'interazione ha una frequenza variabile a seconda del tipo di informazione che si vuole acquisire, per cui risulta più alta per le informazioni a breve o brevissima scadenza e più bassa per quelle più stabili: questa differenziazione è utile per evitare di sovraccaricare il sistema e minimizzare le interazioni, che per quanto brevi hanno inevitabilmente un minino impatto sulle performance dell'host.

   Una volta acquisite le informazioni, queste vengono memorizzate localmente sul proxy (per consentire la ritrasmissione in caso di problemi) e trasmesse al server per gli step successivi: il server memorizza i dati e li inserisce all'interno del sistema, rendendoli accessibili al backend per le eventuali indagini successive.

5. **Apprendimento**: è la fase in cui il sistema apprende il comportamento della rete.

   Lo studio del comportamento della rete è utile per le successive fasi di rilevamento, investigazione e azione. L'apprendimento è attuato dal backend, il quale interroga il server mano a mano che quest'ultimo acquisisce informazioni sugli host durante la fase di interrogazione: l'apprendimento consiste, da una parte, nell'acquisizione delle informazioni topologiche sulla rete e, dall'altro, nella memorizzazione storica di tali valori e il calcolo dinamico di media e deviazione standard contestualizzate nel tempo, così da tracciare un profilo tipico del comportamento dei dispositivi di rete. La fase di apprendimento può essere supportata dal server, eseguendo una prima elaborazione delle informazioni e restituendo al backend dati statistici già disponibili.

6. **Rilevamento**: è la fase in cui avviene il rilevamento dell'anomalia.

   La fase di rilevamento è attuata dal server. I dati acquisiti durante l'interrogazione vengono costantemente confrontati con alcune soglie e, in caso di un superamento rilevante e duraturo, il server genera un allarme per una possibile anomalia: questo viene trasmesso al backend per la successiva fase di indagine del problema, in modo da determinare se si tratti di un falso allarme o di un problema vero e proprio.

   Le soglie possono essere definite staticamente sulla base del tipo di host a cui si applicano, oppure calcolate e aggiornate dinamicamente in base al comportamento della rete studiato durante la precedente fase di apprendimento. Quest'ultimo caso è il più interessante ed efficace, in quanto consente di ridurre notevolmente il numero di falsi allarmi e conseguentemente di scaricare il sistema da eccessive elaborazioni, entrando nella successiva fase di investigazione solo se strettamente necessario.

7. **Investigazione**: questa fase prevede l'indagine relativa all'anomalia.

   Dopo il ricevimento di un allarme da parte del server, il backend inizia la vera e propria gestione dell'anomalia articolata nelle due fasi di investigazione e azione.

   L'investigazione può essere rappresentata concettualmente come la visita in profondità all'interno dell'albero di possibili sviluppi del percorso investigativo, dove ad ogni nodo occorre una consultazione approfondita di uno degli elementi del sistema al fine di prendere una decisione. Ogni consultazione prende forma di un comando contestualizzato, impartito dal backend a un host o a un apparato per mezzo del server e del proxy, finalizzato ad acquisire informazioni aggiuntive sullo stato del dispositivo in questione. L'esito del comando viene elaborato dal backend, il quale compie una scelta che lo porta verso un percorso o un altro all'interno dell'albero. La fase termina quando il backend raggiunge una foglia dell'albero, che corrisponde ad un'azione.

8. **Azione**: è la fase in cui si conclude la gestione dell'anomalia.

   Il sistema può gestire l'anomalia mediante due tipi di azione: l'inoltro dell'allarme e la reazione all'anomalia. La prima è l'operazione con cui il backend comunica ai suoi destinatari tutte le informazioni che è riuscito ad individuare in relazione all'anomalia rilevata inizialmente dal server: se la fase di indagine ha avuto successo, l'allarme in uscita consente una correzione (manuale) al problema efficace e mirata, in quanto le cause sono state circoscritte ed è chiara la procedura che porta alla soluzione.

   La reazione all'anomalia, come anticipato, è un'operazione opzionale e subordinata al livello di servizio sottoscritto con il soggetto responsabile della rete monitorata. Essa si sviluppa con una modalità simile all'investigazione, in cui il backend invia comandi contestualizzati agli host e agli apparati con l'obiettivo di ripristinare la situazione di normalità e far rientrare l'allarme. Concettualmente si tratta anche in questo caso di una visita in profondità di un albero, dove la decisione presa ad ogni nodo corrisponde all'esito più o meno positivo del relativo comando precedentemente impartito. La fase termina quando in un nodo si ottiene un esito positivo al relativo comando, e il problema risulta risolto.

Capitolo 6

Implementazione del sistema

In questo capitolo vengono descritti gli strumenti adottati per lo sviluppo del sistema e le scelte implementative compiute. La traduzione dell'architettura delineata nel precedente capitolo in una concreta messa in opera rappresenta la fase più densa del lavoro di tesi, in quanto ha richiesto lo studio approfondito degli strumenti, la loro riconfigurazione per adattarsi alle esigenze contingenti e la valutazione di scelte e compromessi per rispettare i requisiti architetturali.

Come anticipato nell'introduzione, si è scelto di implementare il sistema progettato sfruttando, ove possibile, strumenti già disponibili e conosciuti all'interno del Consorzio TOP-IX, con un occhio di riguardo all'open-source. Per lo specifico tipo di lavoro di tesi, fortemente orientato alla progettazione di un intero sistema più che allo sviluppo di un singolo modulo software, infatti, si è ritenuto utile partire da prodotti già esistenti e dal funzionamento consolidato, così da potersi concentrare sull'architettura senza dover eccessivamente scendere e confrontarsi con i dettagli di basso livello.

Gli strumenti selezionati sono due: **Zabbix** e **Apache NiFi**. Entrambi software open-source, il primo è pensato per il monitoraggio di reti e sistemi informatici, mentre il secondo è progettato per automatizzare il flusso di dati tra sistemi software. Nelle prossime sezioni ne vengono descritti i concetti fondanti, i componenti strutturali, i ruoli che assumono nell'implementazione del sistema, le configurazioni e le reciproche interazioni.

6.1 Zabbix

Zabbix [30] è un software di monitoraggio distribuito per infrastrutture IT. È stato creato dallo sviluppatore russo Alexei Vladishev nel 1998 come strumento proprietario all'interno di un istituto bancario e nel 2001 è stato reso pubblico sotto i termini della GNU General Public License. Attualmente è sviluppato e mantenuto dalla società Zabbix SIA.

Zabbix è in grado di monitorare un ampio spettro di risorse all'interno di una rete, valutandone in particolare disponibilità, integrità e performance. Usa un meccanismo di notifica flessibile che consente agli utenti di configurare allarmi in relazione a qualsiasi tipo di

evento, consentendo una rapida reazione ai problemi riscontrati. Tutte le informazioni acquisite durante il monitoraggio sono memorizzate in un database relazionale che costituisce un'utile storico per svariati usi interni ed esterni.

6.1.1 L'architettura

Zabbix è costituito da tre componenti software principali: *server*, *proxy* e *agent*.

Server

Il server è il componente centrale di Zabbix: interroga gli host per acquisire i dati, riceve le notifiche asincrone prodotte da proxy e agent, calcola i trigger e invia le notifiche agli utenti. È il repository centrale in cui sono memorizzate tutte le configurazioni, le statistiche e i dati operazionali, ed è l'entità di Zabbix che si occupa di avvertire gli amministratori quando si verifica un problema in uno qualsiasi dei sistemi monitorati.

Il funzionamento del server è ripartito in tre moduli distinti secondo la classica architettura three-tier delle web application: il back-end (scritto in C, è il motore del server), il front-end (scritto in PHP, fornisce l'interfaccia web per l'accesso remoto a Zabbix da qualsiasi piattaforma) e il database (relazionale, basato alternativamente su MySQL, PostgreSQL, SQLite o DB2).

Proxy

Il proxy può raccogliere i dati su performance e disponibilità per conto del server: è un componente opzionale dell'installazione di Zabbix, ma è fondamentale per realizzare un monitoraggio distribuito, diluendo e riducendo il carico del server su sistemi di grandi dimensioni. Un proxy può essere utilizzato per monitorare siti remoti o siti con comunicazioni inaffidabili, per scaricare il server quando deve monitorare migliaia di dispositivi e per semplificare la manutenzione per sistemi distribuiti. Il proxy necessita di una singola connessione TCP al server, richiedendo pertanto la configurazione al più di una sola regola sul firewall.

Tutti i dati raccolti dal proxy vengono memorizzati localmente prima di essere trasmessi al server (quindi anche il proxy si basa su un database relazionale locale): in questo modo non viene perdura alcuna informazione in caso di temporanei problemi di connettività. Naturalmente il proxy è solo un raccoglitore di dati, quindi non calcola trigger, non processa eventi e non invia allarmi, funzioni che sono esclusivamente competenza del server.

Agent

Gli agent sono installati a bordo degli host per monitorare attivamente le risorse locali e le applicazioni, riportando i dati raccolti al server (eventualmente per mezzo del proxy) per un processamento ulteriore: in caso di problema (come un hard disk pieno o il processo di un servizio terminato inaspettatamente) il server può così notificare un allarme agli amministratori. Gli agent sono particolarmente efficienti in quanto usano le system call native dei sistemi operativi ospiti per raccogliere dati statistici: in questo modo il loro impatto sul sistema monitorato è minimo.

Gli agent possono effettuare controlli passivi e attivi: con i primi rispondono a richieste di dati da parte del server o del proxy, come il carico di una CPU, restituendo i relativi valori; i secondi richiedono un processo più complesso, in quanto l'agent deve prima recuperare dal server una lista di parametri da monitorare autonomamente, e in seguito inviare periodicamente i nuovi valori al server.

6.1.2 Gli elementi

L'oggetto del monitoraggio effettuato da Zabbix è l'*host*, un qualsiasi dispositivo (fisico o virtuale) raggiungibile tramite indirizzo IP o DNS sulla rete locale monitorata. Un host è sempre associato almeno a un'interfaccia tramite cui il server può recuperate informazioni, dalla più semplice verifica di disponibilità (ping) alle più complesse aggregazioni di dati statistici sull'uso delle risorse. Gli host possono essere inseriti manualmente all'interno del software, conoscendo l'indirizzo delle loro interfacce, oppure possono essere individuati automaticamente mediante regole di *discovery* opportunamente configurate.

Alla base della raccolta di dati effettuata da Zabbix ci sono gli *item*: un item definisce una specifica metrica oppure semplicemente ciò che è interessante ottenere da un host. Un item può essere di vari tipi, a seconda della sorgente dell'informazione cui è associato: un agent Zabbix (attivo o passivo), un controllo semplice (stato di una porta TCP/UDP, ping ecc.), un demone SNMP, l'aggregazione di altri item, un comando (shell locale, SSH, Telnet), un protocollo (HTTP, FTP, SMTP ecc.), una query SQL e così via. Ogni item è associato all'interfaccia di un host tramite cui recupera il valore della metrica, aggiorna il dato con una certa frequenza, lo mantiene per un certo lasso di tempo, e fa parte di un'application.

Le *application* sono usate per raggruppare gli item in insiemi logici: per esempio l'application MySQL può comprendere tutti gli item correlati al server MySQL, ovvero la disponibilità del servizio, lo spazio su disco, il carico del processore, le transazioni al secondo, il numero di query lente ecc.

Gli item incapsulano semplicemente dati, ma la loro valutazione automatica è affidata ai *trigger*: un trigger è un'espressione logica che definisce una soglia di accettabilità del valore di un dato; se tale livello è superato dal valore di un dato in arrivo, il trigger scatta e transita nello stato di "Problem", segnalando che è avvenuto qualcosa che potrebbe richiedere attenzione (associato ad un livello di gravità); quando il livello ritorna accettabile, il trigger transita nuovamente nello stato di "Ok". L'espressione di un trigger può essere anche molto complessa e tenere conto dei valori associati a vari item: è costruita dalla combinazione di funzioni e operazioni relative a valori, tempo e altri fattori (somma, media, assenza, delta, conteggio, data ecc.).

Il cambiamento dello stato di un trigger è la più frequente e importante fonte di *event*: l'event contiene i dettagli relativi a cosa è successo, in quale contesto è successo e qual è il nuovo stato. Seguendo i due possibili stati di un trigger, anche gli event possono essere di tipo "Problem" e "Ok": il primo tipo è scatenato da un trigger scattato mentre si trovava nello stato di "Ok", e il secondo tipo può essere scatenato da un trigger rientrato, da un altro evento o da un intervento manuale di un utente.

Gli event possono essere seguiti da operazioni di allarme ed eventualmente di ripristino e di conferma: le operazioni veicolano sempre un messaggio, che può essere comunicato tramite un comando remoto oppure tramite un *media type*. I media type rappresentano le interfacce di Zabbix verso gli utenti, e possono essere messaggi email, SMS, script oppure fare uso di sistemi di notifica come Jabber e Ez Texting.

Gli *script* (la compatibilità del linguaggio è dipendente dal sistema operativo su cui Zabbix è installato) sono precedentemente memorizzati in un percorso predefinito sul server e vengono eseguiti quando i relativi media types sono invocati da un evento. Zabbix supporta anche la creazione diretta di *script globali* all'interno del server, immediatamente accessibili a tutte le funzionalità del software, sia dal server che dai proxy e dagli agent.

Application, item e trigger possono essere definiti in modo generico all'interno di *template* associabili manualmente o automaticamente agli host: l'associazione automatica può avviene in conseguenza al matching di una specifica regola di discovery, in cui la verifica del valore di alcuni parametri è indicatore sia della presenza dell'host che della compatibilità con uno o più template. L'uso di template semplifica notevolmente la configurazione del software, consentendo la generazione di item e trigger in modo automatico e standard.

Zabbix, al di là della sua natura open-source, mette a disposizione degli sviluppatori una ricca serie di *API* basate su web service (JSON-RPC) tramite le quali è possibile operare su qualsiasi configurazione del server sfruttando semplici HTTP POST: questa caratteristica abilita l'automazione del processo di monitoraggio della rete, e si rivela particolarmente adatta all'integrazione con Apache NiFi, lo strumento presentato di seguito. A seconda del contesto, infine, Zabbix esporta delle *macro* con cui è possibile accedere a una serie di valori e parametri utili per estenderne le funzionalità.

Figura 6.1. Una tipica schermata della dashboard di Zabbix

6.2 Apache NiFi

Apache NiFi [31] è un progetto software dell'Apache Software Foundation ideato per automatizzare il flusso di dati tra sistemi software. Trae origine e nome dal progetto *Niagara Files* sviluppato dalla National Security Agency, reso open-source nel 2014; lo sviluppo del software è attualmente gestito da Hortonworks.

I punti di forza di NiFi sono la sua semplicità e la facilità nella creazione di flussi di dati in grado di veicolare enormi quantità di eventi tra sistemi garantendo una precisa semantica di consegna. NiFi consente un controllo in tempo reale sugli eventi, facilitando il movimento di dati da qualsiasi sorgente a qualsiasi destinazione. È una piattaforma agnostica dal punto di vista della sorgente dei dati, supportando diverse fonti e formati: dai flussi TCP/UDP ai RDBMS, dalle REST API ai file di log, consentendo al contempo il parsing, l'arricchimento e la trasformazione in tempo reale dei dati. Queste caratteristiche rendono NiFi particolarmente adeguato alle soluzioni di cybersecurity, dove la frequenza dei dati, i formati differenti, l'arricchimento e le trasformazioni sono fondamentali.

Il modello di design di NiFi fornisce vari benefici allo sviluppo di applicazioni basate sulla piattaforma: si presta bene alla creazione visuale e alla gestione di grafi di processi; è nativamente asincrona, consentendo un throughput molto alto e un buffering naturale anche in caso di fluttuazione nella frequenza dei dati; fornisce un modello ad alto parallelismo senza che lo sviluppatore debba preoccuparsi di gestire i tipici problemi legati alla concorrenza; promuove lo sviluppo di componenti coesi e poco accoppiati in modo che possano essere riusati in nuovi contesti; inoltre definisce punti di ingresso e uscita dal sistema univoci e facilmente tracciabili.

6.2.1 I concetti

Il design di NiFi si basa sul paradigma di sviluppo detto *Flow-based programming* (FBP): tale approccio definisce le applicazioni come una rete di processi "black-box" in grado di scambiare dati attraverso specifiche connessioni definite esternamente ai processi; questi ultimi possono quindi essere riusati più volte, così da realizzare applicazioni diverse senza doverli modificare internamente. Di seguito vengono presentati i concetti chiave alla base di NiFi, mostrando le corrispondenze con gli omologhi concetti del FBP:

- **FlowFile** Un FlowFile rappresenta un oggetto "in movimento" attraverso il sistema; per ogni FlowFile, NiFi tiene traccia di una mappa di attributi chiave/valore e del loro contenuto di zero o più byte. Nel paradigma FBP il FlowFile corrisponde all'*Information Packet*.

- **FlowFile Processor** Il processor è l'elemento che di fatto elabora i dati, eseguendo combinazioni, trasformazioni o mediazioni tra flussi. I processor hanno accesso agli attributi di uno specifico FlowFile e al flusso del suo contenuto: possono operare su uno o più FlowFile nella stessa unità di elaborazione e fare sia commit che rollback delle operazioni compiute. Il concetto di processor corrisponde a quello di *Back Box* nel paradigma FBP.

Il processor è il componente che più si presta ad un'implementazione personalizzata, per quanto la piattaforma ne metta già a disposizione più di 200: tra questi le tipologie principali riguardano la trasformazione dei dati (compressione/decompressione, rimpiazzo del contenuto ecc.), l'accesso ai database (query SQL), l'estrazione di attributi (valutazione di espressioni JSON, estrazione di testo ecc.), la data ingestion (acquisizione di dati), la data egress (distribuzione dei dati), lo splitting e l'aggregazione.

- **Connection** Le connection rappresentano i collegamenti tra processor. Agiscono come code, consentendo a processor diversi di interagire con diverse frequenze. Le code possono avere priorità dinamica e limiti superiori di riempimento. Nel FBP corrispondono ai *Bounded Buffer*.

- **Flow Controller** Il Flow Controller mantiene la conoscenza su come i processi sono connessi, e gestisce i thread e l'allocazione delle risorse usate da ogni processo; agisce come broker per facilitare lo scambio di FlowFile tra processi. Nel paradigma FBP corrisponde allo *Scheduler*.

- **Process Group** Un Process Group è uno specifico insieme di processi e relative connessioni, il quale può ricevere dati in input attraverso determinate porte e restituirli in output tramite altre: in questo modo i Process Group consentono la creazione di nuovi componenti semplicemente ricomponendone altri preesistenti. Il concetto corrisponde alla *Subnet* nel FBP.

6.2.2 L'architettura

NiFi è scritto in Java, quindi per l'esecuzione fa uso della Java Virtual Machine su un sistema operativo ospite. Di seguito vengono descritti i componenti principati del software, così come mostrato nella figura 6.2:

- **Web Server** Gestito internamente al software, rende disponibile l'interfaccia di controllo e visualizzazione di NiFi via HTTP, nonché le REST API. Una tipica interfaccia di NiFi è mostrata nella figura 6.6.

- **Flow Controller** È il "cervello" delle operazioni: gestisce i processi, lancia i thread per l'esecuzione delle extension e programma l'accesso alle risorse.

- **Extension** Le extension consentono di estendere le funzionalità di NiFi abilitando nuovi tipi di processi e di elaborazioni; anch'esse eseguono sulla JVM.

- **FlowFile Repository** È il componente dove NiFi tiene traccia dello stato di ciò che è conosciuto su un certo FlowFile attualmente presente nel flusso di elaborazione. L'implementazione di questo repository è modulare, e quella predefinita si basa su un log di tipo write-ahead memorizzato in una specifica partizione del disco.

- **Content Repository** È il componente che immagazzina il contenuto in byte dei vari FlowFile. Anche in questo caso l'implementazione è modulare: l'approccio predefinito è basato su un semplice meccanismo di memorizzazione di blocchi di dati su disco.

- **Provenance Repository** È il componente dove si tiene traccia dell'origine degli eventi che interessano il flusso di elaborazione. L'implementazione predefinita fa uso di uno o più volumi su dischi fisici, e all'interno di un volume gli eventi sono indicizzati e facilmente ricercabili.

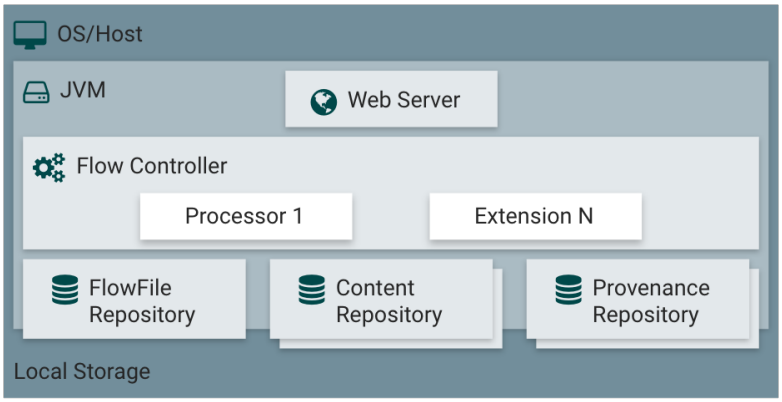

Figura 6.2. Lo schema architetturale di Apache NiFi

6.3 I ruoli

Zabbix e Apache NiFi si sono rivelati software adeguati a rispondere alle esigenze architetturali del sistema sviluppato. Delle numerose funzionalità fornite dai prodotti ci si è concentrati su quelle adatte per l'implementazione dei moduli di acquisizione, rilevamento ed elaborazione, così come descritti nel capitolo dedicato all'architettura.

Considerando il ciclo di vita del sistema, Zabbix è protagonista delle fase di installazione, configurazione, scoperta, interrogazione e rilevamento, mentre Apache Nifi entra in gioco nelle fasi di apprendimento, investigazione e azione.

6.3.1 Gli host

Tra i requisiti del sistema progettato c'è il basso impatto sulla rete terminale oggetto di monitoraggio. Tenuto conto di ciò, gli interventi sugli host devono essere limitati all'attivazione dei servizi minimi per l'acquisizione di informazioni, in base del tipo di host monitorato. Esistono vari livelli di intervento, a impatto crescente, da cui si ricavano dati via via più completi e utili.

Controlli semplici

Sono i controlli che non richiedono interventi sugli host, in quanto immediatamente disponibili sui dispositivi. Tra di essi, gli item più significativi per il sistema sono:

`icmpping`	È usato per verificare la disponibilità di un host (`icmppingloss` e `icmppingsec` indicano rispettivamente la percentuale di pacchetti persi e il tempo di risposta medio).
`net.tcp.service`	È usato per verificare la disponibilità di un servizio TCP su un host (`net.tcp.service.perf` indica il tempo richiesto dalla connessione).
`net.udp.service`	È usato per verificare la disponibilità di un servizio UDP su un host (`net.udp.service.perf` indica il tempo prima della risposta).

In caso l'host sia una macchina o un cluster su cui è installato VMWare, Zabbix mette a disposizione tutta una serie di item per la verifica immediata dello stato del sistema (`vmware.*`), dell'hypervisor (`vmware.hv.*`) e delle virtual machine (`vmware.vm.*`).

SNMP

Il protocollo SNMP è nativamente disponibile e abilitato su buona parte degli apparati di rete nei settori SoHo e professionale (router, switch, firewall, stampanti, NAS ecc.): per questo tipo di host, quindi, è necessaria al più la configurazione delle community (SNMPv1/v2c) o delle credenziali (SNMPv3). Su Windows occorre abilitarlo come funzionalità aggiuntiva (tramite interfaccia grafica o a linea di comando) e configurarlo attraverso il registro di sistema, mentre sulle distribuzioni Linux che non lo prevedono nativamente occorre installarlo e modificare opportunamente il file di configurazione.

Come anticipato nel capitolo dedicato al protocollo, SNMP prevede la possibilità di impostare i valori di alcune variabili MIB tramite il comando SET. Questa funzionalità è prevalentemente sfruttata da implementazioni del protocollo di terze parti, ovvero con specifiche MIB gestite dai produttori degli apparati di rete (es. Cisco, Brocade, Juniper ecc.): nell'implementazione del sistema i comandi SET sono utilizzati dal backend per il controllo dei dispositivi compatibili.

SSH

Uno degli strumenti di cui può fare uso il backend per condurre un'indagine approfondita sugli host ed eventualmente predisporre una strategia di reazione è l'invio di comandi remoti mediante SSH. Per abilitare questo comportamento occorre predisporre gli host installando il server SSH in ascolto sulla porta TCP/22: questa soluzione, tuttavia, comporta problemi nell'invio delle credenziali per l'accesso alla console remota e richiede che venga predisposto un utente specifico, dotato di massimi privilegi, sugli host.

Agent Zabbix

L'agent Zabbix è lo strumento più completo per l'acquisizione di informazioni dagli host. Il processo `zabbix_agentd`, così come gli omologhi per il proxy e il server, viene installato come servizio sugli host che richiedono tale livello di monitoraggio: supporta le piattaforme Linux, IBM AIX, FreeBSD, NetBSD, OpenBSD, HP-UX, Max OS X, Solaris e Windows.

Come per il proxy, la configurazione avviene tramite file testuale, di cui si riportano i parametri più rilevanti:

`Server=0.0.0.0/0`	Indica che l'agent accetta connessioni da qualsiasi indirizzo IPv4, compreso quello (ignoto) del proxy.
`Hostname=[...]`	Imposta il nome dell'agent.
`EnableRemoteCommands=1`	Abilita l'esecuzione sull'agent di comandi remoti inviati dal proxy.
`AllowRoot=1`	Consente all'agent di acquisire i privilegi di root, necessari per l'esecuzione di alcuni comandi remoti.
`TLSConnect=psk`	Definisce il modo in cui l'agent si connette al proxy, impostandolo per l'uso di TLS basato su una chiave condivisa.
`TLSPSKIdentity=[...]`	Definisce una stringa univoca per l'identificazione della chiave condivisa tra proxy e agent.

Insieme alla funzionalità SET di SNMP e al server SSH, l'agent Zabbix è il terzo strumento che viene utilizzato dal backend per inviare comandi remoti agli host, abilitando così l'indagine approfondita e l'eventuale meccanismo di reazione.

6.3.2 Il proxy

Il processo `zabbix_proxy` viene installato su un sistema operativo basato su Linux come un servizio, in modo tale che in caso di reboot del sistema si avvii automaticamente: la relativa configurazione è gestita tramite un file testuale, di cui si riportano i parametri più rilevanti:

`ProxyMode=0`	Definisce la modalità di funzionamento del proxy, impostandolo come attivo: significa che è il proxy a connettersi al server, a richiedere la configurazione e ad inviargli i dati.
`Server=[...]`	Imposta l'indirizzo IP del server.
`ServerPort=10051`	Imposta la porta TCP su cui il server è in ascolto.
`Hostname=[...]`	Imposta il nome del proxy, che deve necessariamente corrispondere a quello impostato sul server.
`EnableRemoteCommands=1`	Abilita l'esecuzione sul proxy di comandi remoti inviati dal server.
`ConfigFrequency=60`	Imposta la frequenza con cui il proxy richiede una configurazione aggiornata al server, in secondi.

`DataSenderFrequency=1`	Imposta la frequenza con cui il proxy invia i dati raccolti al server, in secondi. Si noti che non indica la frequenza di interrogazione (relativa al singolo item e impostata dal server).
`AllowRoot=1`	Consente al proxy di acquisire i privilegi di root, necessari per l'esecuzione di alcuni comandi remoti.
`TLSConnect=psk`	Definisce il modo in cui il proxy si connette al server, impostandolo per l'uso di TLS basato su una chiave condivisa.
`TLSPSKIdentity=[...]`	Definisce una stringa univoca per l'identificazione della chiave condivisa tra server e proxy.
`TLSPSKFile=[...]`	Indica il percorso locale del file contenente la chiave condivisa.

Il valore del parametro `ServerPort` può variare da `1024` a `32767`, con `10051` come predefinito. Per minimizzare la probabilità di dover intervenire con regole opportune sui firewall perimetrali (non rispettando il requisito di basso impatto del sistema) si è scelto di configurare una specifica regola di NAT su *iptables* per modificare la porta di uscita per i pacchetti diretti al server dalla porta `10051` alla porta `80` (tipicamente aperta per il traffico web), mediante il seguente comando:

```
iptables -t nat -A OUTPUT -p tcp -d A.B.C.D --dport 10051
-j DNAT --to-destination A.B.C.D:80
```

Il proxy, in quanto elemento fondamentale per il sistema, deve essere esso stesso oggetto di monitoraggio dal server. Per tale motivo, sullo stesso sistema operativo viene installato anche il processo `zabbix_agentd`, la cui configurazione è stata dettagliata in precedenza.

6.3.3 Il server

Anche il processo `zabbix_server` viene installato su un sistema operativo basato su Linux come servizio, così da garantire l'avvio automatico in caso di reboot. La configurazione del server avviene tramite interfaccia web: il front-end si appoggia ad Apache HTTP Server, configurato per ascoltare sulla porta TCP `8080`, in quanto la porta `80` è già utilizzata per la connessione tra server e proxy. A tal proposito, per fare in modo che i pacchetti in arrivo dal proxy raggiungano correttamente il socket aperto dal server sulla porta `10051` (non può essere configurato per ascoltare su una porta inferiore alla `1024`), occorre impostare una regola di NAT su *iptables* simmetrica a quella impostata sul proxy:

```
iptables -t nat -A PREROUTING -p tcp --dport 80
-j REDIRECT --to-port 10051
```

Tale comando imposta *netfilter* per redirigere tutti i pacchetti provenienti dalla porta TCP `80` verso la porta `10051`, senza modificare altri parametri.

La configurazione del server avviene in due fasi: la prima viene effettuata una tantum, all'installazione, mentre l'altra viene ripetuta ad ogni attivazione di un nuovo sistema. Durante la prima fase si procede con la configurazione dei template, dei relativi event, degli utenti, dei media type e degli script globali.

- **Template**

  Zabbix fornisce una serie di template predefiniti pronti all'uso, sufficienti per un monitoraggio basilare ma non adeguati a tutte le esigenze del sistema sviluppato. Alla luce di questo, vengono creati diversi template personalizzati specificamente adatti per monitorare quei parametri che possono essere indicatori di problemi di sicurezza: tali template, quando associati agli host, rappresentano il primo livello di indagine del sistema, ossia il rilevamento di anomalie operato dal server.

  Di seguito vengono mostrati alcuni dei template più significativi tra quelli predefiniti, mentre nel capitolo in cui si presenta il caso di studio viene descritto dettagliatamente un template personalizzato.

  – `Template App X Service`

    È il prototipo di template per il monitoraggio di un servizio `X` di livello applicativo. Tra i servizi già disponibili ci sono HTTP, HTTPS, FTP, IMAP, LDAP, NNTP, NTP, POP, SMTP, SSH e Telnet, ma qualsiasi altro servizio accessibile tramite una porta TCP o UDP può essere monitorato.

    Il prototipo di template prevede una singola application contenente un singolo item di tipo "simple check", chiave `net.tcp/udp.service[X]`, frequenza di aggiornamento di 1 minuto e cronologia di 1 settimana, abbinato ad un singolo trigger di gravità media attivato da tre tentativi di connessione al servizio falliti.

  – `Template OS Linux`

    È il template per il monitoraggio di un host con sistema operativo basato su Linux, dotato di agent Zabbix (da cui si recuperano tutti i dati). Prevede varie application relative al monitoraggio di CPU (context switch al secondo, tempo di iowait, tempo di idle, interrupt al secondo, carico del processore ecc.), filesystem (spazio libero, inodes liberi ecc.), memoria (disponibile, di swap ecc.), interfacce di rete (traffico entrante, traffico uscente ecc.) e processi (numero attivi ecc.).

    Il template `Template OS Linux SNMPv2` raccoglie un sottoinsieme delle informazioni indicate, ma esclusivamente tramite SNMPv2, senza la necessità di installare sugli host l'agent Zabbix. Infine i template `Template OS Windows` e `Template OS Windows SNMPv2` sono le controparti per ambienti Microsoft.

  – `Template DB MySQL`

    È il template per il monitoraggio di un databse MySQL in esecuzione su un host dotato di agent Zabbix. Prevede una sola application contenente vari item per monitorare le frequenze delle operazioni di begin, select, commit, rollback, delete, insert, delle query, dei byte ricevuti ecc. Un solo trigger di gravità media segnala la non disponibilità del servizio dopo un ping fallito.

- **Event**

  Gli event configurati in questa fase sono scatenati dai trigger associati agli item presenti nei relativi template. Il comportamento è simile per tutti gli event, e prevede la generazione di un allarme con invio del relativo messaggo al backend per tramite del media type associato, così come specificato al punto successivo. Ulteriori event possono comportare notifiche dirette agli amministratori per eventi non associati ad anomalie o possibili problemi di sicurezza.

- **Media type**

  Viene creato un media type di tipo script, denominato "HTTP POST", per inoltrare al backend i messaggi associati agli allarmi generati in seguito alla rilevazione di anomalie. Il semplice script cui è associato il media type (`post.sh`, di seguito), riceve come argomenti l'indirizzo su cui è in ascolto il backend e la macro `{ALERT.MESSAGE}` per recuperare il contenuto del messaggio.

  ```
  #!/bin/bash
  to = $1
  data = $2
  curl -X POST -d "$2" $1
  ```

  Il media type HTTP POST viene associato all'utente "Monitor", attivato a corredo e deputato all'interazione con il backend: Monitor fa parte del gruppo "Zabbix administrators" ed è plenipotenziario sull'intero sistema.

- **Script globali**

  Vengono definiti tre script globali "segnaposto", ossia pronti per essere individuati ed aggiornati opportunamente dal backend: in accordo con la loro destinazione sono denominati `Script on server`, `Script on proxy` e `Script on agent`. La loro presenza è fondamentale per gestire le operazioni di indagine da parte del backend, in quanto sono univoci (e non occorre tenere traccia dei loro id in fase di aggiornamento) e facilmente aggiornabili a seconda del contesto d'uso.

Durante la seconda fase, invece, si procede con la configurazione (lato server) del proxy, delle regole di discovery e dei relativi event.

- **Proxy**

  Il proxy viene definito lato server con lo stesso nome inserito lato proxy nel campo `Hostname`, è selezionato come attivo e vengono impostate la chiave e l'identità PSK così come precedentemente definite nel file di configurazione sul proxy.

- **Discovery**

  Viene creata una regola di discovery associata al proxy che la deve applicare, ovvero quello appena definito. Nell'implementazione più generica, dove l'amministratore non conosce l'indirizzo della rete di destinazione, è possibile impostare il range di indirizzi IP con `10.0.0.0/8, 172.16.0/12, 192.168.0.0/16` per scansionare l'intero spazio di indirizzamento privato: questo è da evitarsi, in quanto il protocollo interno

prevede l'invio da parte del proxy di 3 ARP Request per ogni indirizzo, rendendo l'intera procedura insostenibilmente lenta; è invece preferibile una minima conoscenza della rete così da poter restringere il range il più possibile, in modo da velocizzare le operazioni di discovery.

L'intervallo di update, ovvero la frequenza di aggiornamento, dipende dalla dinamicità della rete di destinazione: nel contesto di Operational Technology, per cui il sistema è stato pensato, non è necessaria una frequenza alta (si può impostare a 1w, una settimana) in quando si opera con reti piuttosto stabili dove gli host (controllori e attuatori) cambiano raramente; in un contesto più dinamico, per esempio la rete di un istituto scolastico, potrebbe essere necessario un aggiornamento più frequente (es. 1g).

La regola di discovery richiede l'impostazione di uno o più controlli, ovvero i parametri preliminari che vengono verificati per determinare la presenza e il tipo di dispositivo (al di là del suo indirizzo IP, che rappresenta comunque il criterio di univocità dell'host). Si indicano i controlli più significativi impostati:

`ICMP ping`	È un controllo semplice per verificare la presenza o meno dell'host
`SNMPv2 agent "1.3.6.1.2.1.1.1"`	Verifica che il demone SNMPv2 sia attivo sull'host controllando che risponda all'OID indicato (`sysDescr`)
`Zabbix agent "system.uname"`	Verifica che l'agent Zabbix sia attivo sull'host controllando che risponda all'item indicato
`HTTP`	Controlla che sull'host sia attivo un server HTTP in ascolto sulla porta 80
`FTP`	Controlla che sull'host sia attivo un server FTP in ascolto sulla porta 21
`SSH`	Controlla che sull'host sia attivo un server SSH in ascolto sulla porta 22

È possibile configurare regole di discovery aggiuntive, oppure estendere quella già configurata, per l'identificazione degli apparati di rete di specifici produttori: questo è possibile verificando la risposta positiva ad un controllo di tipo *SNMPv2 agent* sull'OID appartenente alla MIB proprietaria del produttore.

- **Event**

  Quando viene individuato un nuovo host che soddisfa la regola di discovery, si scatena un evento opportuno per la gestione del suo inserimento nel sistema. Viene configurato un evento per ogni tipologia di host identificato, in relazione ai controlli verificati nella regola di discovery:

`ICMP ping`	Aggiunta dell'host
`SNMPv2 agent "1.3.6.1.2.1.1.1"`	Aggiunta dell'host e associazione al template `Template OS Linux SNMPv2` oppure al template `Template OS Windows SNMPv2` a seconda del valore ricevuto
`Zabbix agent "system.uname"`	Aggiunta dell'host e associazione al template `Template OS Linux` o `Template OS Windows` a seconda del valore ricevuto
`HTTP`	Aggiunta dell'host e associazione al template `Template App HTTP Service`
`FTP`	Aggiunta dell'host e associazione al template `Template App FTP Service`
`SSH`	Aggiunta dell'host e associazione al template `Template App SSH Service`

In accordo con quanto indicato nel paragrafo precedente, in caso siano configurate regole di discovery aggiuntive mirate all'individuazione di specifici apparati di rete, vengono conseguentemente configurati altrettanti eventi per la gestione dell'inserimento e l'associazione ai relativi template. In caso di soddisfacimento di più condizioni, l'operazione di aggiunta dell'host è idempotente.

6.3.4 Il backend

Apache NiFi viene installato sulla macchina deputata all'elaborazione, basata su Linux, come servizio. La configurazione del software avviene tramite file: la maggior parte dei controlli disponibili riguarda proprietà del core, dei repository, di sicurezza, del cluster (in caso di tale tipo di installazione) e del web server, tra cui il parametro `nifi.web.http.port=8081` modificato per accedere all'interfaccia web di gestione (l'`8080` è già utilizzata da Apache HTTP Server per il front-end di Zabbix).

Il funzionamento di Apache NiFi si sviluppa attraverso una serie di processi, ciascuno costituito da catene di processor e connection e deputato a svolgere una determinata funzione. Di concerto con quanto fatto per Zabbix, di seguito viene presentato il framework di processi comuni a tutte le implementazioni del backend, con particolare riferimento ai sotto-processi che ne realizzano le funzioni elementari; nel capitolo in cui si presenta il caso di studio ne verrà descritto dettagliatamente il relativo processo decisionale.

Nella figura 6.3 è mostrata la panoramica del backend, con i quattro macro-blocchi fondamentali: più propriamente si tratta di "process-group", ossia di raggruppamenti di sotto-processi via via più semplici. Si noti che questi blocchi ricalcano il funzionamento del backend così come presentato nel capitolo dedicato all'architettura, tramite le fasi di ricezione dell'allarme, gestione dell'anomalia (indagine ed eventuale reazione) e inoltro dell'allarme.

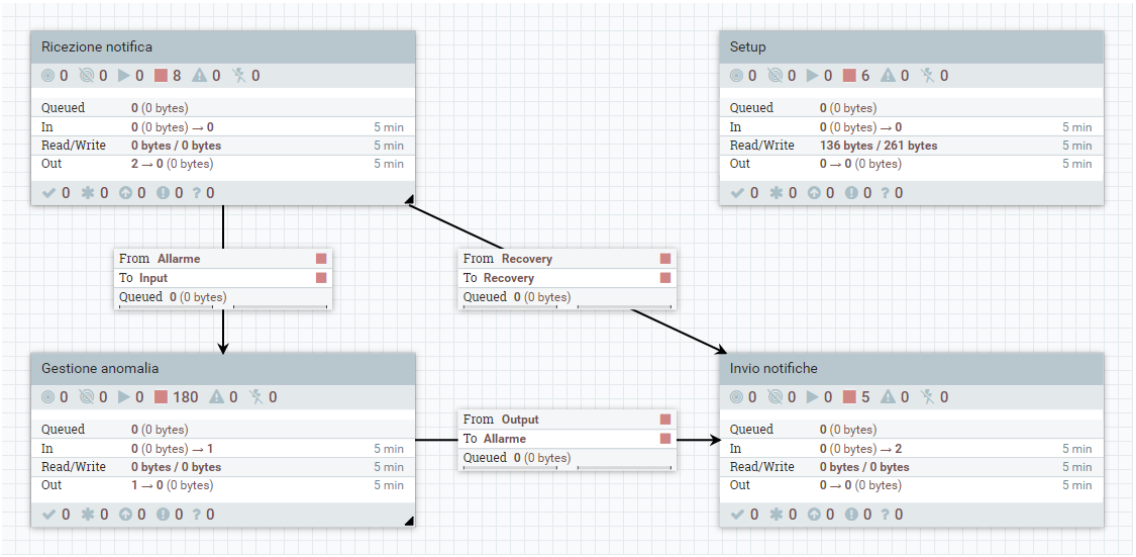

Figura 6.3. Panoramica del backend

Autenticazione del backend

Il processo di autenticazione (`user.login`) fa parte del gruppo "Setup", nel quale si svolgono le operazioni di auto-configurazione iniziale di NiFi. È un processo la cui esecuzione è indipendente dal resto del sistema e periodica (la frequenza dipende dalla scadenza di una sessione). Nella figura 6.4 è mostrata la rappresentazione grafica del processo:

1. **GenerateFlowFile**: è il processor, comune a molti processi, in cui viene inizialmente generato il FlowFile e associati gli attributi comuni; il ciclo di vita di quest'ultimo è spesso molto complesso, e termina solo alla fine dell'intera gestione dell'evento. In questo caso viene associato l'attributo `CEI:auth`, utilizzato successivamente nel processor `PutDistributedMapCache`.

2. **FetchFile**: è il processor con cui viene caricato un file testuale come contenuto del FlowFile; in questo caso viene caricato il JSON che costituirà il body della HTTP POST verso Zabbix tramite cui avverrà l'autenticazione.

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "Monitor",
        "password": "<password>"
    },
    "auth": null,
    "id": 1
}
```

3. **InvokeHTTP**: è il processor deputato all'invio della richiesta HTTP POST all'indirizzo su cui è in ascolto il web server che fornisce le API di Zabbix; il Content-Type è di tipo `application/json-rpc`, il timeout è impostato a 5 secondi e il body è il JSON recuperato tramite `FetchFile`.

4. **EvaluateJsonPath**: i due blocchi costituiti da questo tipo di processor servono per estrarre il token di autorizzazione dal JSON ottenuto come risposta (che sostituisce il contenuto del FlowFile precedente, ovvero la richiesta) e caricarli rispettivamente come attributo e contenuto del FlowFile in uscita.

5. **PutDistributedMapCache**: è il blocco che carica il token di autorizzazione in una "Distributed Map Cache" visibile a tutto il sistema di elaborazione (con chiave `auth`, recuperata come valore dell'attributo `CEI` impostato durante la generazione del FlowFile); questo rende il token una sorta di variabile globale, che potrà essere utilizzata per l'autorizzazione delle richieste successive.

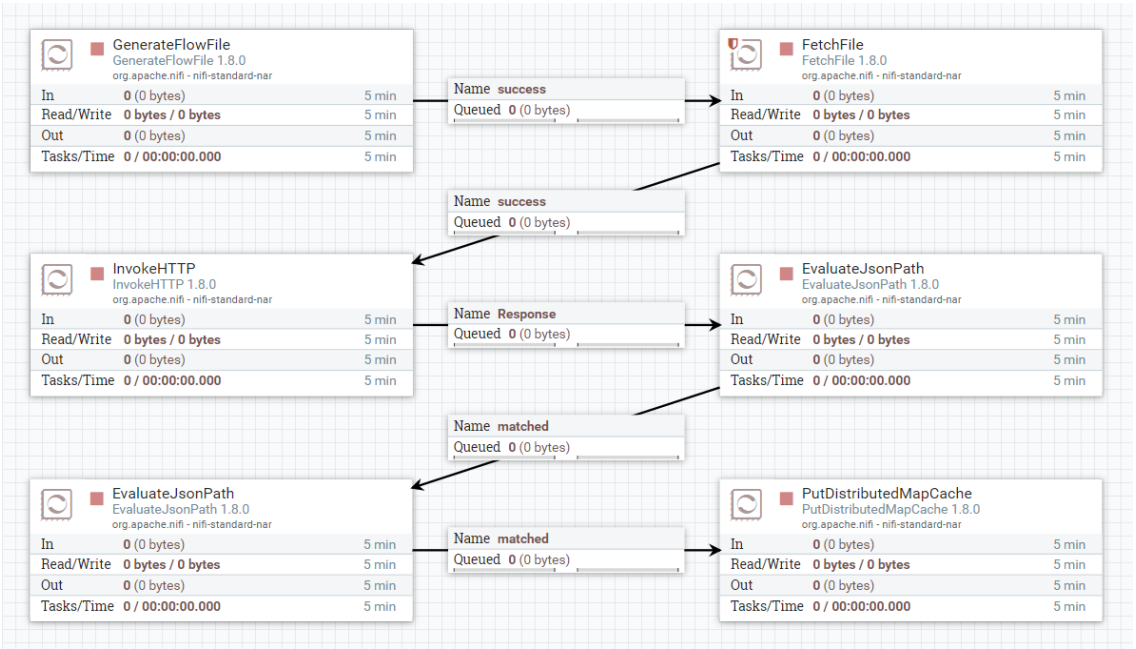

Figura 6.4. Panoramica del processo di autenticazione del backend

Autenticazione JSON

L'autenticazione JSON è il processo tramite cui un JSON ricevuto in input viene preparato per l'invio al web server di Zabbix inserendo nel campo `auth` il token di autenticazione ottenuto nella fase descritta in precedenza. È usato come processo elementare all'interno di tutti i processi che prevedono comunicazioni con il server, come i due che seguono nella presentazione. La panoramica è mostrata nella figura 6.5, e di seguito si dettagliano i processor più significativi:

1. **UpdateAttribute**: è il processor, usato due volte, tramite cui rispettivamente si inserisce e si rimuove l'attributo `auth` nel FlowFile tramite cui il processor successivo di tipo `FetchDistributedMapCache` può recuperare il token di autorizzazione dalla "Distributed Map Cache" condivisa.

2. **UpdateRecord**: è il processor che si occupa di aggiornare il contenuto del FlowFile di tipo JSON ricevuto in input; il JSON viene parsificato tramite un service di tipo "JsonTreeReader", il record `auth` viene aggiornato con il token già disponibile come attributo, e infine il JSON viene ricostruito tramite un service di tipo "JsonRecordSetWriter"; lo schema del JSON utilizzato da tali service è contenuto in uno Schema Registry basato su Apache Avro.

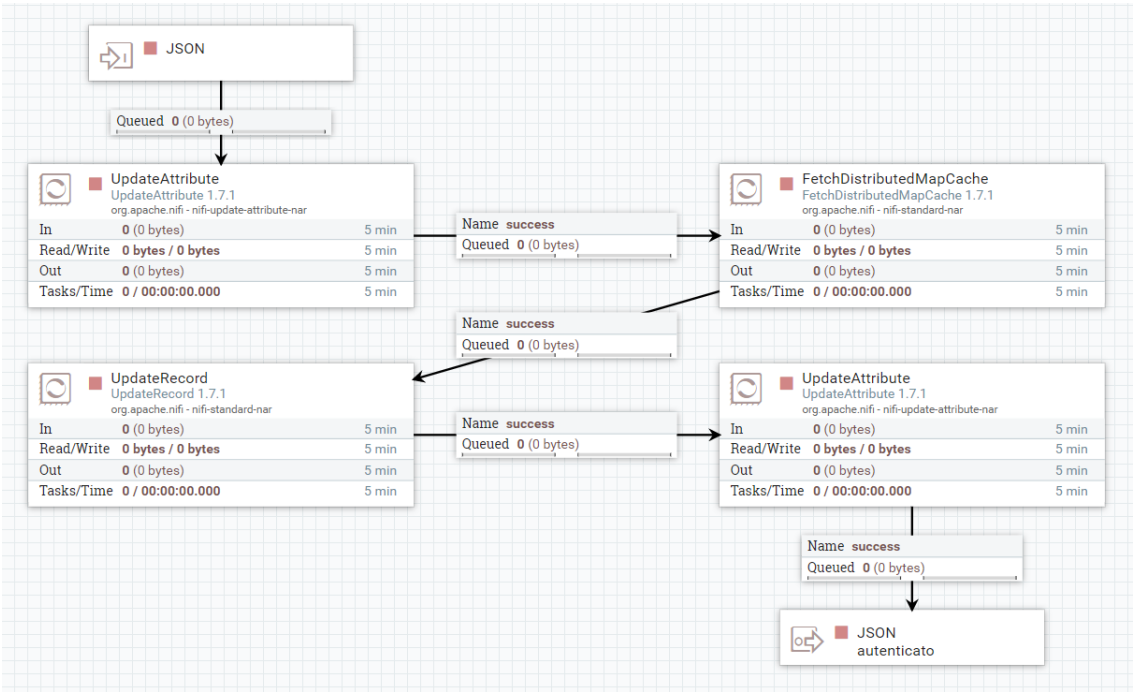

Figura 6.5. Panoramica del processo di autenticazione del JSON

Aggiornamento di uno script

Il processo di aggiornamento di uno script viene lanciato in preparazione dell'esecuzione di comandi remoti inviati dal backend al server, al proxy o direttamente all'host. Questo blocco, insieme al successivo, trova impiego durante l'invio dei comandi remoti al resto del sistema. Il processo accetta in input un FlowFile con gli attributi `scriptid` e `command` impostati opportunamente, e procede all'aggiornamento del relativo script memorizzato sul server Zabbix. La figura 6.6 mostra la panoramica del processo, i cui blocchi sono simili per configurazione e funzionamento a quelli presentati per i processi precedenti.

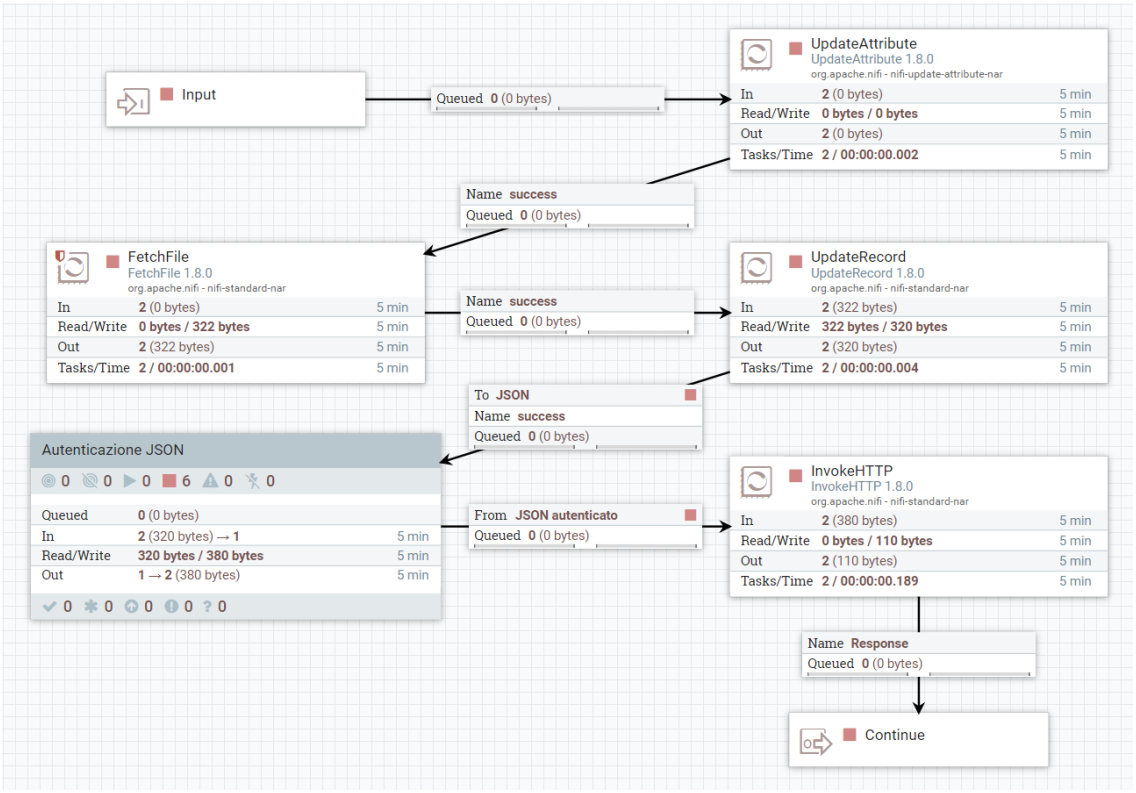

Figura 6.6. Panoramica del processo di aggiornamento di uno script

Esecuzione di uno script

Il processo è del tutto simile a quello di aggiornamento di uno script, cui solitamente segue nel flusso di eventi: entrambi sono usati in fase di gestione dell'anomalia. Le uniche differenze sono i parametri in ingresso, che in questo caso sono `script.id` e `host.id`, inseriti nel JSON dal processor `UpdateRecord`.

Invio di un comando remoto

Il processo di invio di un comando remoto è utilizzato a più riprese all'interno del gruppo "Gestione anomalia" per veicolare tutte le comunicazioni da e per il resto del sistema. Si tratta di un processo composito, ossia costituito da diversi blocchi e sotto-processi elementari, di cui si riporta una panoramica nella figura 6.7.

La caratteristica saliente del processo è il processor `RouteOnAttribute`, l'equivalente dell'istruzione `switch` in molti linguaggi di programmazione: in questo contesto è usato per discriminare il destinatario del comando remoto, ovvero il server, il proxy o l'agent su cui verrà eseguito; seguono i blocchi per l'aggiornamento e l'esecuzione dello script associato al comando, così come presentati precedentemente; termina la catena i processor per l'estrazione e l'interpretazione dell'output del comando dalla risposta ricevuta dal server.

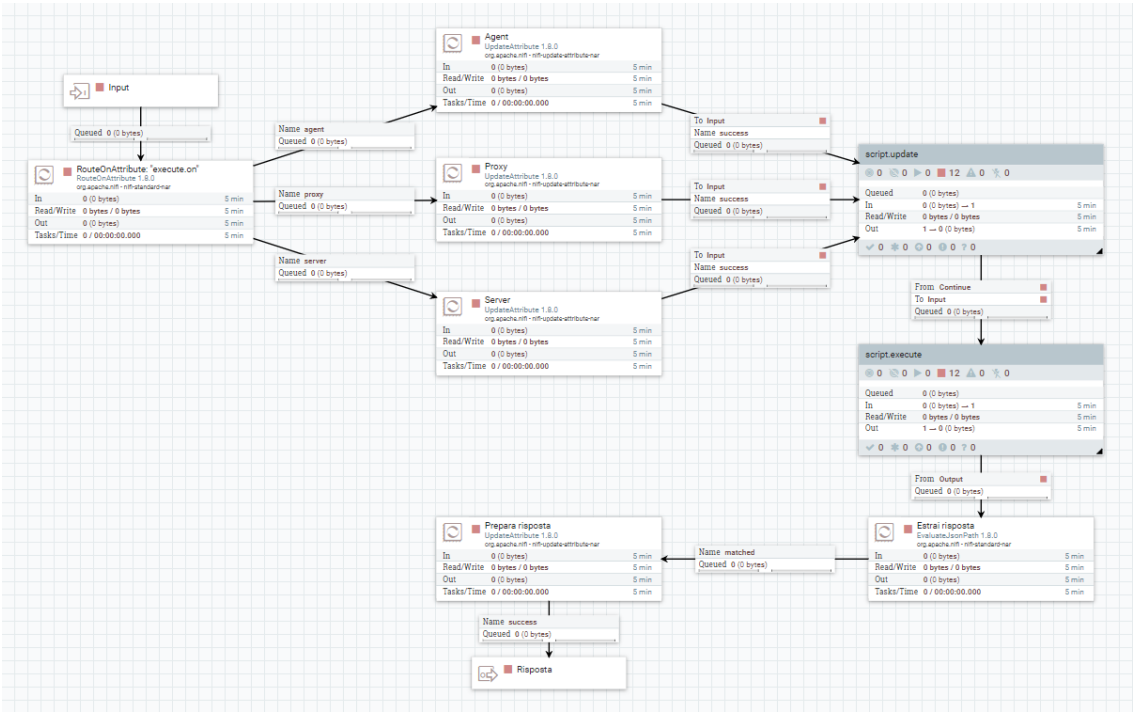

Figura 6.7. Panoramica del processo di invio di un comando remoto

Ricezione di una notifica

Il processo di ricezione di una notifica interviene ogni qualvolta il server Zabbix, dopo aver rilevato una possibile anomalia oppure la sua risoluzione, genera una notifica e la invia al backend. È fondamentalmente costituito da un web service, in ascolto su una porta nota al server, e una catena di processor per estrarre dal message body della HTTP POST ricevuta i parametri utili per condurre l'indagine successiva o inoltrare la notifica di risoluzione.

La figura 6.8 mostra la panoramica del processo. Il web service è costituito dalla coppia di processor `HandleHttpRequest` e `HandleHttpResponse`; il messaggio inviato dal server Zabbix è formattato in JSON, così da essere facilmente interpretabile, ed i record più significativi contenuti al suo interno vengono mappati sui seguenti attributi del FlowFile:

`action.*` Attributi relativi all'azione che invia l'allarme: ID e nome

`event.*` Attributi relativi all'allarme vero e proprio: ID, nome, data, ora, severità e stato (notifica o problema)

`host.*` Attributi relativi all'host su cui è stata rilevata l'anomalia: ID, IP e nome

`item.*`	Attributi relativi all'item associato al valore anomalo: ID, nome, chiave e valore
`proxy.name`	Nome del proxy relativo alla rete monitorata
`trigger.*`	Attributi relativi al trigger scattato: ID, nome e nome del template che lo contiene

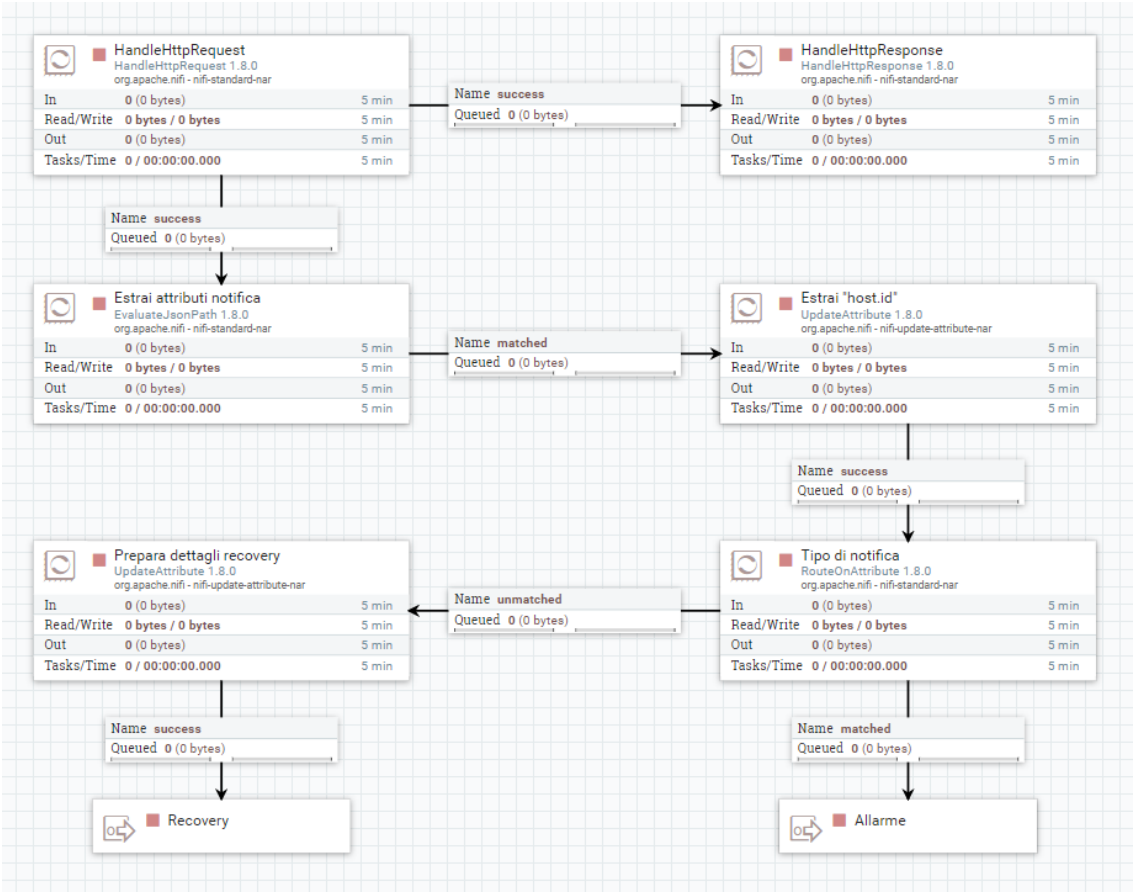

Figura 6.8. Panoramica del processo di ricezione di una notifica

Invio di una notifica

Quando il processo di indagine è concluso, l'allarme arricchito viene inoltrato ai destinatari opportuni (tipicamente gli amministratori di rete o dei sistemi) a mezzo notifica push. Il sistema adottato per la gestione delle notifiche push è Prowl: si tratta di un prodotto commerciale specificamente pensato per il sistema operativo Apple iOS, ma sono presenti numerose implementazioni per altri dispositivi; la scelta è giustificata dall'uso che già se ne fa all'interno del Consorzio TOP-IX dove il sistema è stato sviluppato.

Il processo è simmetrico a quello di ricezione, e prevede l'assemblaggio del nuovo messaggio a partire dagli attributi associati al FlowFile in ingresso. Di seguito vengono presentati i processor che realizzano la funzione:

1. **UpdateAttribute**: è il processor che, manipolando gli attributi del FlowFile in ingresso, assembla un nuovo attributo corrispondente al contenuto del messaggio testuale notificato (che verrà associato al parametro `description` nel processor successivo). Il messaggio è costruito in modo standard, con una dicitura comune a tutti i tipi di notifica, dove l'elemento variabile contiene i dettagli associati all'indagine o alla risoluzione: questi ultimi sono assemblati al di fuori del processo corrente, in quanto strettamente dipendenti dal tipo di problema riscontrato.
2. **PostHTTP**: è il processor deputato all'invio del messaggio al server Prowl per la gestione delle notifiche push; l'invio avviene con una richiesta di tipo HTTP POST, dove l'url contiene tutte le informazioni necessarie (`apikey` per l'identificazione, `priority` per selezionare l'importanza della notifica, `application` come mittente, `event` come oggetto e `description` come contenuto del messaggio).

Capitolo 7

Validazione

In questo capitolo viene presentato il caso di studio selezionato per verificare il funzionamento del sistema, ovvero gli attacchi di tipo **Denial of Service (DoS)** flood-based. Per completare il quadro vengono descritti l'ambiente in cui è stata condotta la prova, la configurazione del sistema e i risultati sperimentali ottenuti. Lo studio si è concentrato in particolare sull'attacco **TCP SYN Flood**, affrontandolo sia lato vittima che lato attaccante: se infatti è comprensibile il pericolo rappresentato da questo tipo di minaccia per il suo obiettivo, è meno evidente (ma sempre più diffuso) il caso in cui anche l'attaccante è una vittima all'interno dello scenario di attacco.

7.1 Il caso di studio

Gli attacchi di tipo *Denial of Service* (DoS) sono celebri al giorno d'oggi per essere tra le minacce più comuni ai servizi Internet: dal VoIP ai server DNS, dal gaming online alle piattaforme di e-commerce, sono molti i provider che hanno subito almeno una volta un attacco DoS. Questo tipo di minaccia rende il servizio inaccessibile agli utenti genuini: nel caso più semplice la finalità ultima è proprio il disservizio, mentre sempre più spesso è precursore di minacce più gravi e mirate.

Gli attacchi DoS sono classificati in due gruppi: basati su vulnerabilità e di tipo flood. Nel primo gruppo rientrano gli attacchi con cui, sfruttando una debolezza del sistema vittima o di un protocollo, viene generato un disservizio (es. Ping of Death, LAND, Targa3, Neptune). Il secondo gruppo, invece, comprende quegli attacchi in cui l'attaccante satura le risorse dell'host vittima, inondandolo di traffico malevolo e sovraccaricandone l'hardware (carico della CPU, allocazione di memoria, buffer delle schede di rete ecc.): ad oggi questi attacchi sono i più diffusi, in quanto se le vulnerabilità possono essere eliminate, modificare il meccanismo di funzionamento dei protocolli è ben più arduo.

Quando l'attaccante è più di uno si parla di DDoS (Distributed Denial of Service), dal funzionamento identico ma realizzato utilizzando numerose macchine attaccanti che insieme costituiscono una *botnet*. Una botnet è una rete controllata da un host (detto botmaster) e composta da dispositivi infettati da malware specializzato, detti bot o zombie.

Spesso i dispositivi connessi ad Internet al cui interno sussistono vulnerabilità di sicurezza (es. telecamere di videosorveglianza, home gateway di fascia bassa o dispositivi IoT economici) sono buoni candidati a diventare parte della botnet. Nello scenario di convergenza IT/OT presentato nel Capitolo 2, le botnet rappresentano un enorme rischio per le aziende che stanno affrontando questa transizione tecnologica [32]: per tale motivo è importante adottare misure di protezione non solo per difendersi da attacchi DoS, ma anche per evitare di esserne le inconsapevoli sorgenti.

Tra gli attacchi di tipo flood più comuni vi sono il *TCP SYN Flood*, l'*UDP Flood* e l'*ICMP Flood*, basati rispettivamente sui principi di funzionamento dei protocolli TCP, UPD e ICMP. Il primo, più sofisticato, sfrutta in modo malevolo il meccanismo del *three-way handshake* proprio del protocollo TCP, con cui vengono stabilite le nuove sessioni:

1. il client richiede l'apertura di una nuova sessione inviando un messaggio SYN al server, richiedendo così di sincronizzare il *Sequence Number* (SN), ovvero la posizione del segmento TCP all'interno del flusso completo;

2. il server, nel caso abbia sufficienti risorse, alloca spazio in memoria per la nuova sessione e risponde positivamente al client inviando un messaggio SYN-ACK con il SN del client incrementato di 1 (e il suo SN);

3. il client conferma inviando un messaggio ACK con entrambi gli SN aumentati di uno, così che il server possa deallocare la memoria precedentemente riservata e stabilire definitivamente la nuova sessione.

Il meccanismo descritto è rappresentato graficamente nella figura 7.1.

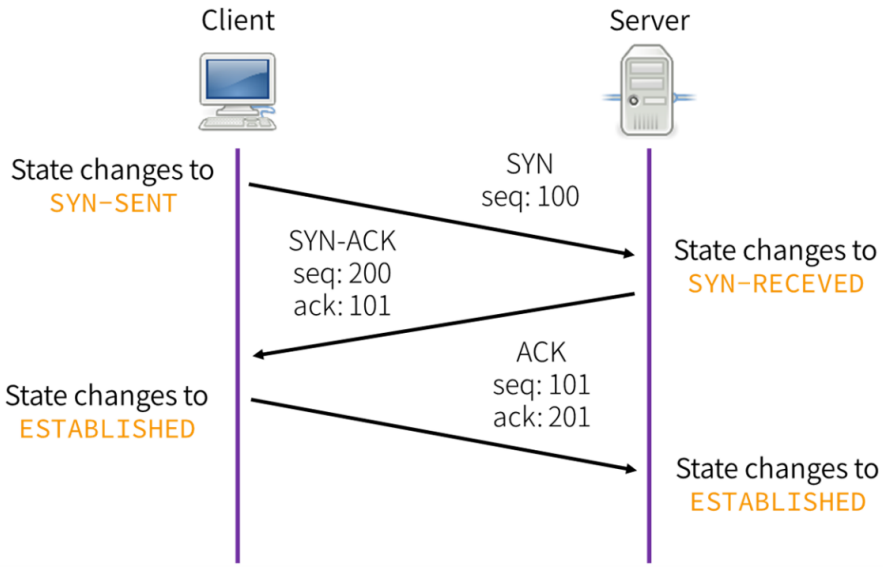

Figura 7.1. Rappresentazione del TCP Three-way handshake

Nell'attacco DoS di tipo TCP SYN Flood, l'attaccante invia ad altissima frequenza una serie di messaggi SYN, ignorando le risposte SYN-ACK del server e non inviando mai i messaggi ACK: questo fa sì che sul server si stabilisca un gran numero di sessioni semi-aperte che porteranno rapidamente alla saturazione della memoria disponibile, impedendo l'apertura di sessioni anche da parte di client genuini. Una variante dell'attacco prevede che l'attaccante invii i messaggi SYN utilizzando lo *spoofing*, ovvero camuffando il proprio indirizzo IP (mittente) con uno diverso: in questo modo le risposte SYN-ACK del server raggiungeranno un destinatario diverso dal mittente che, non avendo mai richiesto l'apertura della sessione, ignorerà i messaggi e non concluderà l'handshake.

Durante il three-way handshake il client e il server transitano attraverso vari stati, così come definiti dal protocollo e mostrati nella figura 7.2. L'osservazione delle transizioni tra stati in una sessione è un buon indicatore della genuinità della stessa: nell'attacco TCP SYN Flood le sessioni semi-aperte non verranno mai completate transitando nello stato di `ESTABLISHED`, ma rimarranno (lato server) nello stato di `SYN_RCVD` fino alla scadenza del timeout, quando transiteranno direttamente verso lo stato di `CLOSED`.

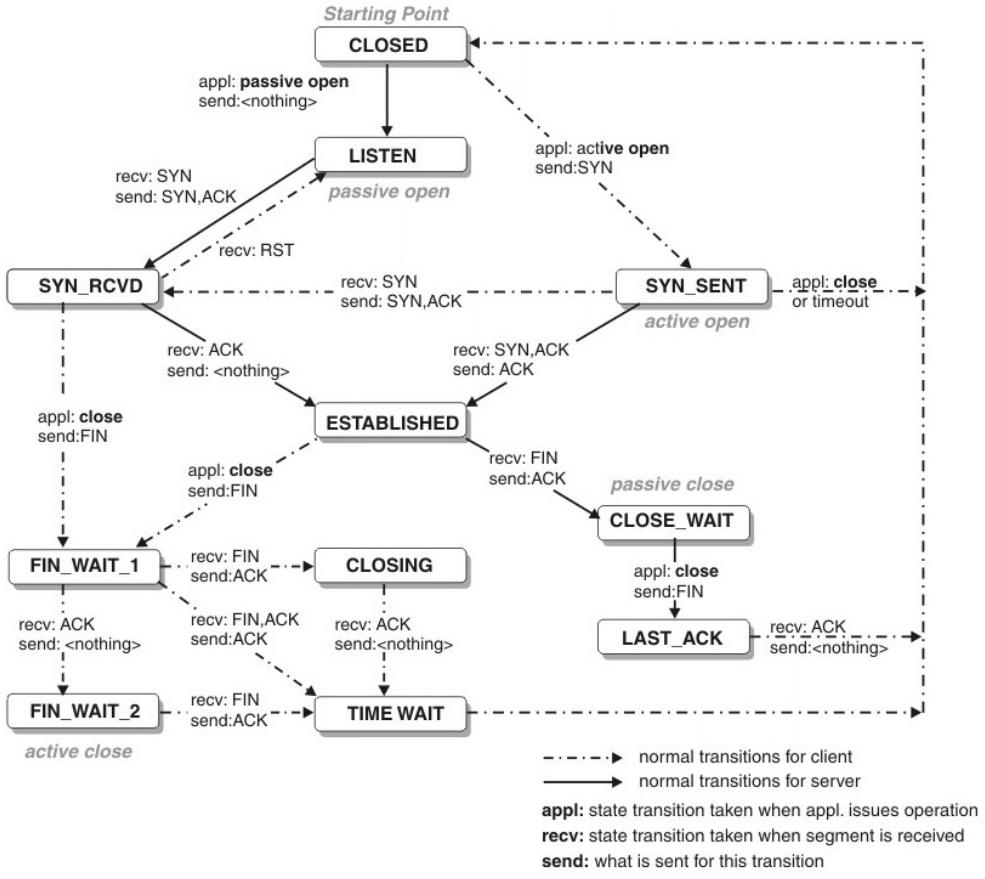

Figura 7.2. Diagramma degli stati del protocollo TCP

Gli attacchi UDP Flood e ICMP Flood, invece, sono meno sofisticati e si basano sull'invio massiccio rispettivamente di datagram UDP e pacchetti "ICMP Echo Request" (ping); l'attacco punta a saturare la banda disponibile della vittima, sia in ingresso che in uscita: quest'ultimo caso si verifica quando l'UDP Flood è diretto ad una porta chiusa e l'host risponde inviando pacchetti "ICMP Destination Unreachable", oppure durante un ICMP Flood quando l'host risponde con pacchetti "ICMP Echo Reply".

7.2 L'ambiente di prova

L'ambiente di prova è costituito da una combinazione di dispositivi fisici e virtuali: la rete monitorata è costituita da alcuni host virtuali, ma i dispositivi e gli apparati di rete sono fisici. Lo schema del sistema è mostrato nella figura 7.3.

Per il proxy si è deciso di utilizzare un **Raspberry Pi 2 Model B** Rev 1.1 dotato di CPU ARM Cortex-A7 @ 900MHz e 1GB di RAM su sistema Raspbian 9, soluzione economica e adeguata per la fase prototipale del sistema: in un contesto di produzione, con il monitoraggio di decine di item per centinaia o migliaia di host, potrebbe essere necessario un hardware dotato di prestazioni superiori. Sperimentalmente si è osservato che le risorse computazionali richieste crescono maggiormente con l'aumento della frequenza di interrogazione, piuttosto che con il numero di item monitorati.

Il server e il backend sono ospitati sulla stessa macchina fisica, un server **HP ProLiant DL360 G5** dotato di due CPU Intel Xeon E3553 @ 2.00GHz e 32GB di RAM su sistema Ubuntu Server 16.04: la scelta è dettata da esigenze di disponibilità hardware in fase prototipale ma, considerata la natura fortemente "espansiva" di Apache NiFi, in un contesto di produzione potrebbe essere sensato installare il backend su un cluster dedicato (così da favorire resilienza e scalabilità).

La rete monitorata è costituita da **quattro host**, di cui uno fisico (Windows 10) e tre virtuali (Ubuntu 18.04) ospitati tramite VirtualBox sul sistema fisico: il virtualizzatore è configurato in modalità bridge, in modo tale che le VM abbiano accesso diretto alla LAN principale. Sui vari host sono abilitati servizi diversi per poter sperimentare il sistema in vari contesti: su Windows e su un Linux è abilitato solo il protocollo SNMPv2, sul secondo Linux è attivo solo l'agent Zabbix e sul terzo sia l'agent Zabbix che SNMPv2. Inoltre su tutti i sistemi Linux è installato il server SSH e su uno di essi anche il server HTTP.

La rete è gestita da un apparato **Mikrotik hAP ac lite** (RouterBOARD 952Ui-5ac2nD) con RouterOS v6.40: il dispositivo integra le funzionalità di router, firewall, NAT e bridge, è dotato di interfaccia wireless e fornisce numerosi servizi di rete; nel contesto del sistema è usato in modalità "station", ossia si comporta come un client Wi-Fi sull'interfaccia WAN.

7.3 La configurazione

La configurazione del sistema è in larga parte coincidente con quella presentata nel capitolo dedicato all'implementazione, soprattutto per i blocchi standard. Quello che differisce sono i template di Zabbix per il rilevamento dei tipi di attacco e la catena di processi NiFi per condurre la fase di indagine.

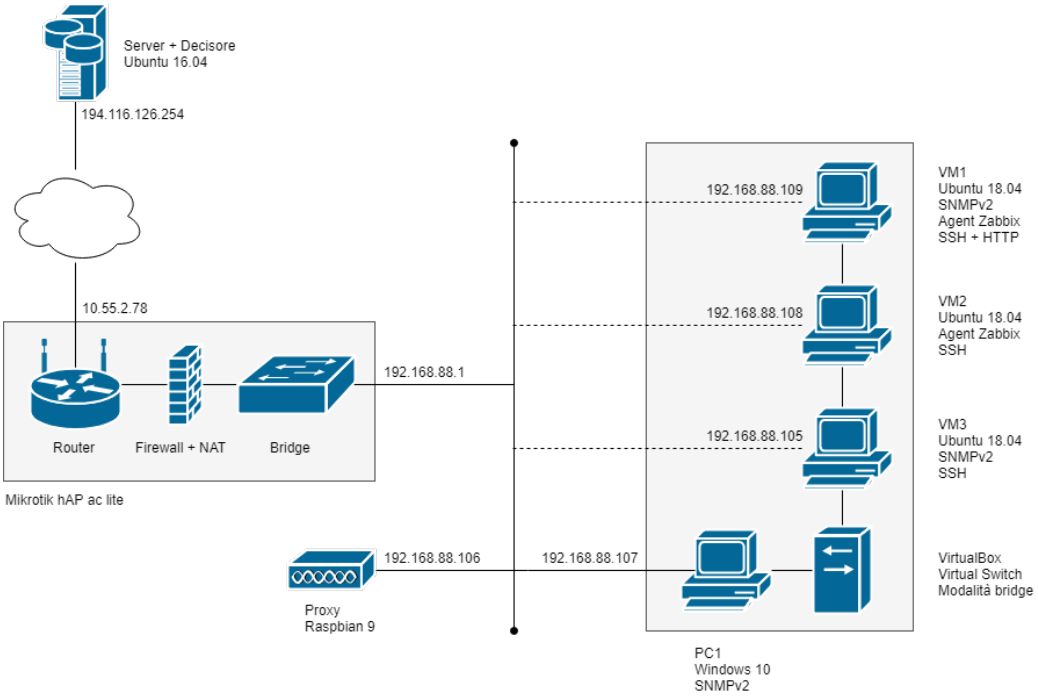

Figura 7.3. Schema dell'ambiente di prova

7.3.1 I template

Per il rilevamento di attacchi di tipo DoS è stato sviluppato il template **SNMPv2 DoS**, basato appunto su SNMPv2. È costituito da una sola application (*Security*) contenente diversi item afferenti alle MIB ICMP, IP, TCP e UDP:

`icmpInDestUnreachs` Il numero di messaggi "ICMP Destination Unreachable" ricevuti. La variabile può essere sfruttata nel rilevamento di attacchi UDP Flood lato attaccante, in quanto è il tipo di messaggi inviato in risposta ad un datagram UDP diretto ad una porta chiusa.

`icmpInMsgs` Il numero totale di messaggi ICMP ricevuti. L'aumento improvviso di questo valore può essere indicatore di un attacco ICMP Flood in corso, sia lato attaccante che lato vittima.

`icmpOutDestUnreachs` Il numero di messaggi "ICMP Destination Unreachable" inviati. La variabile può essere sfruttata nel rilevamento di attacchi UDP Flood lato vittima, in quanto è il tipo di messaggi inviato in risposta ad un datagram UDP diretto ad una porta chiusa.

`ipInDelivers`	Il numero totale di datagram consegnati con successo ai protocolli applicativi basati su IP. Correlato con `ipInReceivers` può essere un indicatore di attività sospetta.
`ipInReceives`	Il numero totale di datagram IP ricevuti, compresi quelli contenenti errori. Correlato con `ipInDelivers` o `ipOutRequests` può essere un indicatore di attività sospetta.
`ipOutDiscards`	Il numero di datagram IP mai inviati a causa di problemi del protocollo. La crescita di questo valore può essere indicatore di un attacco ICMP Flood in corso, lato attaccante.
`ipOutRequests`	Il numero totale di datagram IP di cui è stata richiesto l'invio dai protocolli applicativi basati su IP. Correlato con `ipInReceivers` può essere un indicatore di attività sospetta.
`tcpAttemptFails`	Il numero di volte che le connessioni TCP sono passate direttamente allo stato CLOSED o LISTEN a partire dallo stato SYN-RCVD. Questa variabile è un ottimo indicatore, lato vittima, di un attacco TCP SYN Flood in corso.
`tcpOutRsts`	Il numero di segmenti TCP inviati contenenti il flag RST. Questi messaggi vengono inviati per terminare immediatamente una sessione TCP in caso di problemi, per esempio quando si riceve un messaggio SYN o SYN-ACK per un socket non in ascolto: è un ottimo indicatore di un attacco TCP SYN Flood lato attaccante.
`udpInErrors`	Il numero di datagram UDP ricevuti ma non consegnati per ragioni diverse dall'assenza di un'applicazione in ascolto. La crescita rapida di questo valore è un ottimo indicatore di un attacco UDP Flood in corso.

A tali item è stato associato un trigger progettato per scattare in caso di un aumento improvviso dei relativi valori. Essendo le variabili di cui sopra essenzialmente contatori progressivi, per evitare falsi allarmi in caso di riavvio degli host (e consequente reset dei contatori) nella progettazione della formula interna al trigger è stata aggiunto un controllo di validità. Di seguito si mostra nel dettaglio la formula del trigger associato all'item `tcpAttemptFail` per il rilevamento di un attacco TCP SYN Flood lato vittima:

```
{ tcpAttemptFails.delta(#10) } > 1000 AND
{ tcpAttemptFails.last(#10) } <= { tcpAttemptFails.last() }
```

in cui la funzione `delta(#10)` restituisce la massima differenza tra le ultime dieci letture, `last(#10)` restituisce il valore della decima ultima lettura e `last()` il valore dell'ultima lettura: la prima parte dell'espressione definisce la soglia di problematicità, mentre la seconda le condizioni di validità (il valore non deve essere stato resettato). L'impianto mostrato in questa formula è lo stesso adottato per gli altri trigger.

Oltre agli item descritti, se ne sono aggiunti un paio di tipo "derivato": questo tipo di item non è associato direttamente a una variabile SNMP o all'agent Zabbix, ma è calcolato a partire dai valori di altri item. Nella fattispecie sono stati costruiti gli item `deliverRatio` e `responseRatio`: il primo misura il rapporto tra il numero di pacchetti IP consegnati ai protocolli applicativi e il numero totale di pacchetti ricevuti; il secondo, invece, misura il rapporto tra il numero di pacchetti IP di cui è stato richiesto l'invio dai protocolli applicativi e il numero totale di pacchetti ricevuti. Di seguito si mostrano le relative formule, dove il `+1` al denominatore previene la divisione per zero:

```
last("ipInDelivers") / (last("ipInReceives")+1)
last("ipOutRequests") / (last("ipInReceives")+1)
```

Questi due item sono utili al backend per capire se l'allarme generato da un trigger rappresenta effettivamente una situazione di minaccia oppure no: infatti durante un attacco di tipo flood questi valori cambiano nettamente rispetto alla norma. Il `deliverRatio` tipicamente è prossimo all'unità (attività normale, con la maggior parte dei pacchetti IP consegnati ai protocolli applicativi), quindi valori inferiori a **0.8** devono destare attenzione (capacità del buffer saturata, pacchetti scartati).

Discorso diverso per il `responseRatio`, anch'esso normalmente prossimo all'unità: durante un attacco, nell'host vittima il valore tende ad abbassarsi (i pacchetti ricevuti non vengono consegnati ai protocolli applicativi e non generano risposte), mentre nell'host attaccante tende ad alzarsi (i pacchetti inviati non ricevono risposta).

7.3.2 I processi

La gestione dell'anomalia inizia con l'identificazione della relativa tipologia, così come mostrato nella figura 7.4: in base al tipo di notifica ricevuta (parametro `action.name` interpretato dal processor `RouteOnAttribute`) il sistema discrimina il process-group adeguato per la gestione del problema e, come anticipato, in questo caso di studio ci si è concentrati sul TCP SYN Flood, sia lato vittima che lato attaccante.

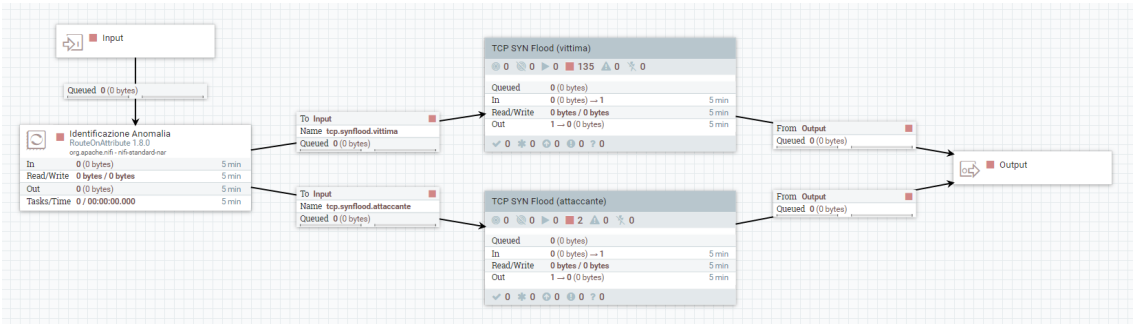

Figura 7.4. L'identificazione dell'anomalia

Gestione lato vittima

1. **Verifica validità allarme**

   La verifica della validità dell'allarme consente di ridurre la probabilità che non si tratti di un falso allarme. Il processo consiste nell'acquisizione dal server della variazione dei valori `deliverRatio` e `responseRatio` ed il confronto con le relative soglie: in caso di attacco, infatti, ci si aspetta che almeno uno dei due scenda più del **20%**, per i motivi descritti in precedenza.

   I blocchi più rilevanti che costituiscono il processo sono: il process-group `item.get`, per l'acquisizione dei due valori dal server (caricamento dello script, parametrizzazione, autenticazione ed esecuzione), il processor `ElavuateJsonPath` per l'estrazione dei due valori dal risultato dello script e `RouteOnAttribute` per il loro confronto con le relative soglie.

2. **Identificazione servizio**

   L'identificazione del servizio sotto accatto coincide di fatto con l'identificazione della porta su cui è in ascolto. Il processo prevede l'esecuzione di un comando remoto sull'host vittima tramite l'agent Zabbix, sfruttando quindi il relativo modulo descritto nel capitolo precedente. La panoramica del processo è visibile nella figura 7.5. Seguono il comando remoto e la relativa descrizione:

   ```
   netstat -tna | grep SYN_RECV | awk -F "[ :]+" '{print $5}' |
   awk '{++a[$0]}END{for(i in a)if(a[i]>max){max=a[i];k=i}print k}'
   ```

   Il comando è costituito a sua volta da una pipe di quattro comandi: il primo è `netstat`, usato per visualizzare lo stato delle connessioni instaurate sul dispositivo (`-tna` seleziona i socket attivi e non attivi TCP, mostrando le informazioni numeriche); il secondo, `grep`, filtra solo le sessioni nello stato `SYN_RECV`, ovvero quelle semi-aperte (il cui gran numero è sintomo di un TCP SYN Flood in corso); il terzo, `awk`, seleziona la colonna corrispondente alla porta di destinazione, mentre il quarto determina quella più frequente (molto probabilmente quella sotto attacco).

3. **Identificazione attaccante**

   Anche l'identificazione dell'attaccante è ottenuta tramite un comando remoto eseguito sull'host sotto attacco, molto simile al precedente:

   ```
   netstat -tna | grep SYN_RECV | awk -F "[ :]+" '{print $6}' |
   awk '{++a[$0]} END {for(i in a) if(a[i]*100/NR>60) print i}'
   ```

   In questo caso, tra tutte le sessioni TCP nello stato `SYN_RECV`, viene selezionato l'indirizzo IP del mittente che compare con una frequenza superiore al 60%: questa assunzione è verificata sperimentalmente, in quanto la riuscita di un attacco TCP SYN Flood richiede che l'attaccante "monopolizzi" il buffer delle sessioni semi-aperte della vittima. Pertanto, appurato che l'attacco è in corso, l'assenza di un indirizzo mittente dominante deve necessariamente significare che l'attaccante sta ricorrendo

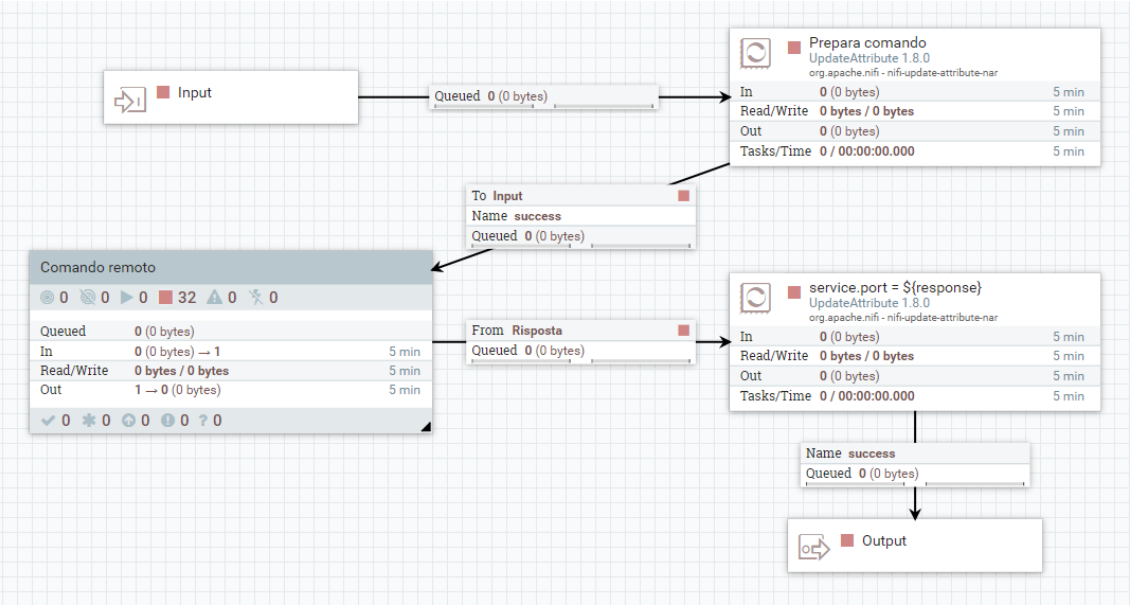

Figura 7.5. L'identificazione del servizio sotto attacco

alla tecnica dello spoofing casuale, forgiando pacchetti con indirizzi mittente random per ridurre le probabilità di essere rintracciato.

I rimanenti processor all'interno del blocco servono appunto per discriminare la natura dell'attacco (singolo o multiplo) così che possano eventualmente essere messe in pratica azioni correttive.

4. **Reazione**

Per il caso di studio adottato si è deciso di predisporre anche una primitiva reazione automatica alla minaccia rilevata, così da mostrare le potenzialità del sistema. La possibilità di attuare un'azione correttiva dipende dalla riuscita identificazione di un singolo attaccante, mentre non è prevista (almeno in questo contesto di prova) in caso di attacco multiplo. Come per gli step precedenti anche la reazione si basa sull'esecuzione di un comando remoto sull'host vittima:

```
ip route add blackhole ${sender.ip}
```

Il comando inserisce una rotta speciale, detta di *blackholing*, all'interno della routing table dell'host: questa rotta imposta il sistema per instradare tutti i pacchetti diretti all'indirizzo IP dell'attaccante verso un blackhole, ovvero una destinazione inesistente. Questo comportamento ha come effetto l'impossibilità di attivare nuove sessioni TCP tra i due host, in quanto non potranno scambiarsi i messaggi necessari per il three-way handshake: di conseguenza l'attacco TCP SYN Flood cessa di causare effetti dannosi sull'host vittima.

Si è deciso di intervenire a livello di routing, e non di firewall (con iptables) in quanto il routing agisce abbastanza presto nello stack protocollare (secondo solo alla catena di PREROUTING) e prevede un accesso più efficiente alle relative regole (hash map invece che linked list), migliorando il tempo di reazione e riducendo l'impatto sulle performance. Questa considerazione perde di validità nel momento in cui la tabella di routing cresce notevolmente di dimensione, rendendo più appetibile intervenire sul firewall (se non esageratamente carico di regole).

5. **Preparazione allarme**

   Il blocco deputato alla preparazione dell'allarme è essenzialmente costituito da una rete di processor di tipo `UpdateAttribute` con il compito di assemblare la parte variabile del messaggio di allarme: questa verrà poi consegnata al blocco deputato all'invio delle notifiche, dove verrà inserita all'interno del messaggio completo ed inviata al sistema di notifica.

Gestione lato attaccante

Di seguito vengono dettagliati solo i processi che differiscono sostanzialmente da quelli presentati nella gestione lato vittima. In questo caso il processo di reazione non è previsto.

1. **Verifica validità allarme**

   Come per il caso precedente, il processo consiste nell'acquisizione dal server della variazione dei valori `deliverRatio` e `responseRatio` ed il confronto con le relative soglie: in caso di attacco, infatti, il primo dovrebbe scendere almeno del **20%** e/o il secondo salire della stessa percentuale. I blocchi che costituiscono il processo sono gli stessi descritti nella gestione lato vittima.

2. **Identificazione vittima e servizio**

   Il processi di identificazione della vittima e del servizio sotto attacco (figura 7.6) condividono la catena di processor fino all'ultimo step, usato per ottenere i due valori. A differenza del caso precedente l'analisi lato attacante non può essere eseguita intervenendo direttamente sull'host, in quanto non è dotato di agent Zabbix per l'esecuzione di comando remoti: l'identificazione viene quindi eseguita sul proxy in modo più sofisticato, intercettando il traffico malevolo e analizzandone le caratteristiche.

   Il processo prevede l'invio e l'esecuzione in sequenza di una serie di comandi remoti. Il primo è `arpspoof`, un tool contenuto nel pacchetto **dsniff** usato per fare *arp poisoning* su uno specifico target: questa tecnica consiste nell'inviare pacchetti di tipo ARP Reply ad un host contenenti informazioni mendaci, così da indurlo ad inviare i pacchetti destinati ad un certo indirizzo IP verso un indirizzo MAC diverso da quello legittimo, tipicamente quello dell'host che esegue il poisoning; quest'ultimo, per non destare sospetti, può inoltrare le trame ricevute al vero destinatario, realizzando così un attacco di tipo *man-in-the-middle*. In questo contesto il comando è usato in modo legittimo per consentire al proxy di accedere al traffico prodotto dall'host attaccante, così da identificare vittima e servizio sotto attacco.

Il secondo comando è `tcpdump`, tradizionale tool dei sistemi Linux basato sulla libreria **libpcap** usato per intercettare i pacchetti in transito su un'interfaccia di rete del sistema su cui è lanciato.

La sintassi dei due comandi è la seguente:

```
timeout 1m tcpdump -n -c 100 src host ${host.ip}
and not dst host ${proxy.ip} > out &

timeout 10s arpspoof ${default.gateway} -t ${host.ip} &
```

Per il primo comando, `-n` richiede l'output completamente numerico (senza risolvere i nomi a dominio), `-c 100` ferma il processo dopo la cattura di 100 pacchetti, la parte relativa agli host richiede la cattura dei soli pacchetti prodotti dall'attaccante e non destinati al proxy stesso (evitando per esempio quelli relativi al protocollo SNMP) e `> out &` richiede un output su file e l'esecuzione in background.

Per il secondo comando, invece, `${default.gateway}` è l'indirizzo IP del router e `-t ${host.ip}` è l'indirizzo dell'attaccante nonché "vittima" dell'arp poisoning: entrambi i valori sono acquisiti durante la fase di apprendimento della rete, ossia durante il processo di Setup; i processor all'inizio del blocco sono necessari per recuperare i due valori dalla `DistributedMapCache` condivisa con tutti i processi del decisione. La falsificazione dell'indirizzo IP del router è necessaria per re-indirizzare il traffico in uscita dalla LAN, in quanto si suppone che la vittima sia un servizio esterno.

Il prefisso `timeout` serve per fermare automaticamente i processi rispettivamente dopo 1 minuto e 10 secondi dall'avvio: il primo è utile per fermare `tcpdump` in caso non sia stato possibile acquisire 100 pacchetti, mentre il secondo lancia `arpspoof` per il tempo necessario per l'acquisizione degli stessi. Il tool ripristina la ARP Table corretta nell'host attaccante prima di terminare.

Sul proxy è stato precedentemente abilitato il forwarding, ossia la capacità di inoltrare verso il default gateway i pacchetti diretti ad un indirizzo IP non assegnato a una delle sue interfacce: in questo modo il traffico flood raggiunge correttamente la destinazione e si riduce la probabilità che il malware generatore riconosca la sofisticazione. Il forwarding viene abilitato scrivendo `1` nel contenuto del file `/proc/sys/net/ipv4/ip_forward`.

Viene infine inviato un ultimo comando al proxy per l'interpretazione dei risultati:

```
cat out | grep [S] | awk -F "[ :]" '{print $7}' |
awk '{++a[$0]}END{for(i in a)if(a[i]>max){max=a[i];k=i}print k}'
```

dove, similmente ai casi precedenti, viene estratto dall'output di `tcpdump` l'indirizzo IP e la porta del destinatario più frequente sui 100 pacchetti acquisiti con il flag SYN impostato (ovvero la firma dell'attacco TCP SYN Flood): la stringa così acquisita viene interpretata dal processor `UpdateAttribute` per estrarre l'indirizzo IP della vittima e la porta corrispondente al servizio sotto attacco.

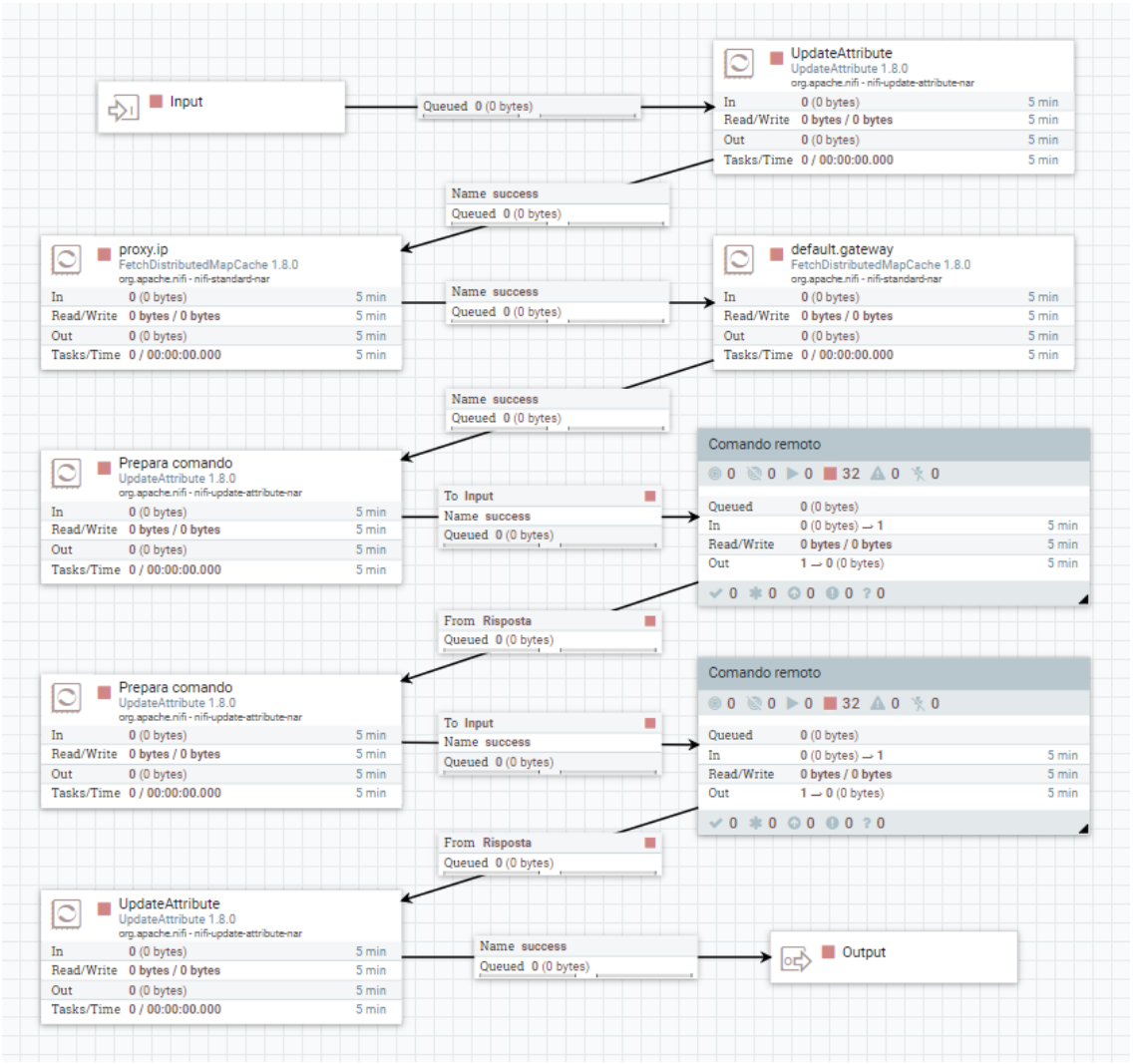

Figura 7.6. L'identificazione della vittima e del servizio sotto attacco

7.4 L'esecuzione e i risultati

Analisi lato vittima

All'interno dello schema di rete mostrato precedentemente, la prima verifica è stata condotta selezionando l'host VM2 (192.168.88.108) come attaccante e l'host VM1 (192.168.88.109) come vittima. L'host VM2 è dotato solo di agent Zabbix, mentre VM1 anche di demone SNMP, tramite cui viene identificato l'attacco.

La simulazione di attacco è stata eseguita mediante il programma **hping3**, un assemblatore/analizzatore di pacchetti TCP/IP orientato alla linea di comando. L'interfaccia testuale è ispirata alla sintassi del comando unix *ping*, ma mentre quest'ultimo invia solo pacchetti

ICMP Echo Request, hping3 supporta la generazione e l'invio di pacchetti ICMP, TCP, UDP e RAW-IP (senza protocolli di livello trasporto).

In questo contesto il comando è stato usato con la seguente sintassi:

```
hping3 -i u1 -S -p 80 192.168.88.109
```

dove `-S` imposta l'invio di pacchetti TCP con il flag SYN settato, `-p 80` definisce la porta e `192.168.88.109` l'indirizzo IP di destinazione e `-i u1` richiede l'invio di un pacchetto al microsecondo. Questa combinazione di parametri descrive esattamente un tipico attacco di tipo TCP SYN Flood condotto verso un servizio in ascolto (HTTP sulla porta 80).

I risultati ottenuti (valori medi su 10 prove) sono in linea con quelli attesi: il problema viene rilevato in 2 secondi e risolto in 34 secondi; l'allarme viene notificato dopo 29 secondi dal rilevamento e la notifica di risoluzione dopo 45 secondi dal rilevamento.

I grafici seguenti mostrano il profilo del primo attacco così come osservato dal server: nella figura 7.7 si può notare il calo improvviso del parametro `responseRatio` sulla vittima, uno dei due discriminanti usati dal backend per validare l'allarme; nella figura 7.8, invece, si nota l'incremento repentino del parametro `tcpAttemptFails` dal quale viene attivato l'allarme. In seguito alla risoluzione del problema la situazione torna alla normalità, con entrambi i parametri stabili sul valore raggiunto.

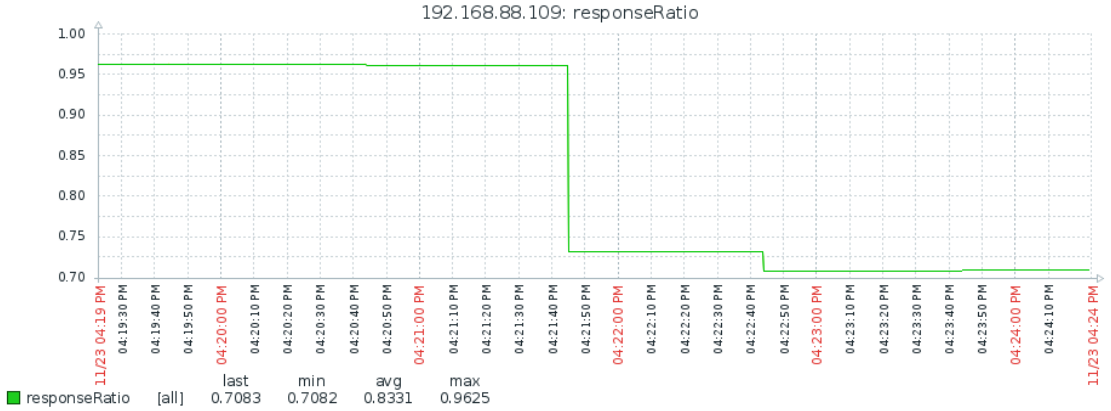

Figura 7.7. La variazione del parametro responseRatio sulla vittima

Analisi lato attaccante

Per questa seconda verifica, l'attacco è stato condotto tra gli host VM3 (192.168.88.105, l'attaccante) e il sito web *google.it* (172.217.23.99, la vittima). L'host attaccante è monitorato solo tramite demone SNMP. Anche in questo caso è stato utilizzato il tool *hping3* nel seguente modo:

```
hping3 -i u1 -S -p 80 google.it
```

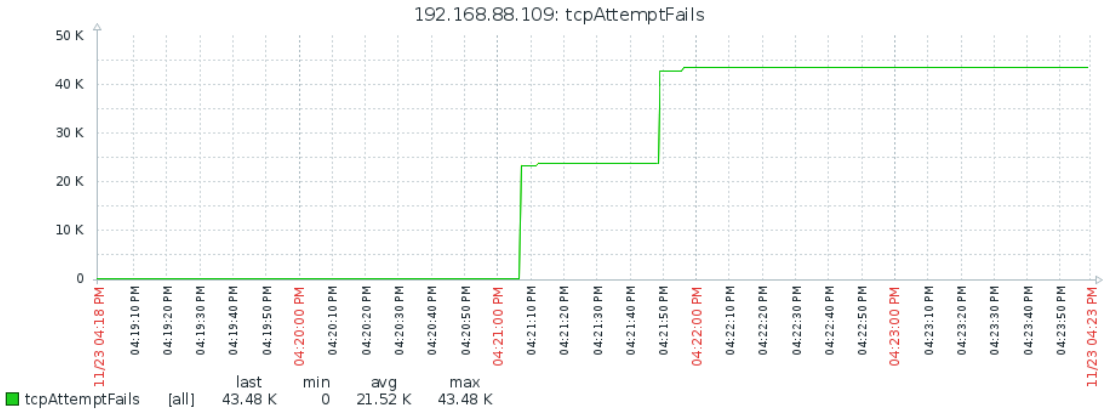

Figura 7.8. La variazione del parametro tcpAttemptFails sulla vittima

dove la sintassi è identica al caso precedente; in questo caso l'host sotto attacco è al di fuori della rete locale, trattandosi infatti di una delle cache del motore di ricerca Google.

Anche in questo caso i risultati sono in linea con quanto atteso (valori medi su 10 prove), con il problema rilevato in 2 secondi e l'allarme notificato dopo 44 secondi dal rilevamento. L'aumento del ritardo tra il rilevamento dell'anomalia e la notifica dell'allarme, rispetto al caso precedente, è dovuto ai tempi necessari per la ridirezione e il campionamento del traffico prodotto dall'host attaccante, controllato dai timeout mostrati in precedenza.

La figura 7.9 mostra l'incremento del `responseRatio` sull'attaccante (molti pacchetti inviati a fronte di poche risposte ricevute), mentre la figura 7.10 mostra l'incremento del parametro `tcpOutRsts`, utilizzato per discriminare il tipo di minaccia. Il questo caso, non essendo prevista alcuna reazione automatizzata, la stabilizzazione dei valori deriva da un intervento manuale (terminazione del processo di attacco).

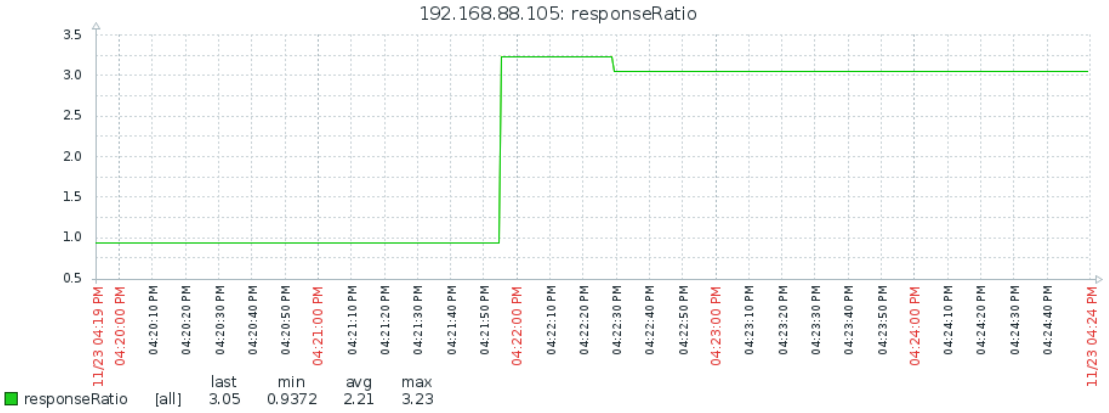

Figura 7.9. La variazione del parametro responseRatio sull'attaccante

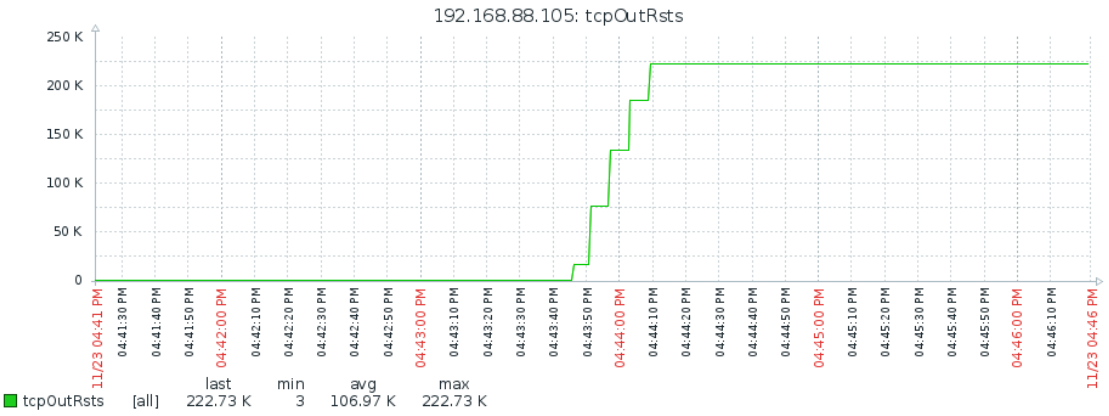

Figura 7.10. La variazione del parametro tcpOutRsts sull'attaccante

Capitolo 8

Conclusioni

Il lavoro descritto in queste pagine rappresenta una soluzione adeguata, semplice e scalabile al problema del rilevamento di intrusioni e minacce di sicurezza all'interno delle reti terminali dei soggetti cui è rivolto, siano essi afferenti al mondo dell'Operational Technology, operatori di servizi essenziali o semplici utenze connesse all'infrastruttura condivisa dell'Internet Exchange. La posizione strategicamente centrale di quest'ultimo rende particolarmente efficace l'architettura distribuita del sistema, così come la sua natura modulare consente l'attivazione di processi custom basati sulla topologia delle reti monitorate nell'ottica della fornitura di un innovativo servizio di *Cybersecurity as-a-Service*.

Le scelte implementative compiute sono state dettate dall'esigenza di produrre un sistema di facile utilizzo, flessibile e adatto a una rapida riconfigurazione. I blocchi che costituiscono i processi decisionali, infatti, consentono la ricostruzione delle funzioni di acquisizione e analisi senza la necessità di intervenire e ricompilare il codice, mettendo a disposizione gli strumenti per realizzare pattern anche complessi a fronte di una semplicità tipica dei linguaggi interpretati.

Come anticipato nell'introduzione, la progettazione di un tale tipo di sistema ha richiesto un'analisi che coinvolgesse anche ambiti collaterali, come l'impatto nei processi aziendali e il trattamento dei dati personali. In relazione a quest'ultimo, si è rilevato che il monitoraggio prestazionale di dispositivi con cui il personale di un'azienda entra in contatto (a titolo di esempio le postazioni utente e i terminali portatili) può essere considerato alla stregua del monitoraggio delle prestazioni lavorative del personale stesso: l'esperienza di alcune realtà con cui ci si è confrontati durante l'analisi preliminare ha evidenziato episodi di contestazione dopo i quali è stato necessario ricorrere ad opportuni accordi sindacali.

Per questi motivi e per la necessità di minimizzare l'impatto del sistema all'interno dei processi aziendali, si è optato per non condurre un'analisi diretta del traffico di rete ma di basarsi su dati aggregati e statistici. Infatti l'analisi dei pacchetti, come mostrato nel caso di studio, viene condotta solo nell'ambito dell'indagine (e non della rilevazione) e attraverso un campionamento indicativo della minaccia ma non del contenuto di informazione veicolato dagli stessi pacchetti.

L'approccio adottato, per quanto data-driven nella sua componente di rilevamento e indagine, fa quindi affidamento a una minima quantità di informazioni acquisite dalla rete. Tuttavia l'architettura del sistema non preclude un suo uso più estensivo ed integrato all'interno di contesti meno critici dal punto di vista della privacy: gli strumenti selezionati per il processo decisionale, infatti, sono nativamente compatibili con l'analisi dei cosiddetti "big-data", abilitando l'estrazione di informazioni statistiche ancora più accurate e aprendo la porta a numerosi sviluppi futuri del sistema.

8.1 Sviluppi futuri

La modularità e la scelta di usare strumenti open source lascia ampio margine di miglioramento e di arricchimento delle funzionalità del sistema. Inoltre il framework progettato può essere facilmente esteso per supportare un ampio numero di processi di anomaly detection, ricombinando in modalità innovative e contestualizzate i moduli fondamentali presentati in queste pagine: l'analisi basata sui dati statistici, per esempio, rappresenta un ottimo supporto al rilevamento di minacce inedite e sofisticate.

Il sistema, inoltre, si presta molto bene all'integrazione con altri software e servizi. In particolare il processo decisionale, così come anticipato precedentemente, è predisposto per l'analisi congiunta di dati provenienti da fonti diverse: per questo lavoro ci si è concentrati sul monitoraggio di reti terminali, ma gli stessi strumenti utilizzati per i processi di acquisizione e rilevamento trovano ampio utilizzo anche sulla dorsale dell'Internet Exchange, dove i margini di intervento sono sicuramente più ampi (si pensi alla network automation a supporto della configurazione degli apparati).

Un possibile sviluppo del sistema può essere l'integrazione con strumenti di analisi dei flussi come Netflow, JFlow, NetStream o sFlow: il backend può infatti correlare le informazioni provenienti da tali strumenti lato backbone con i dati acquisiti da Zabbix sulle reti terminali per creare dei modelli di attacco e diffusione delle minacce, avendo ancora più informazioni a disposizione per intervenire con azioni correttive.

Il livello di sofisticatezza con cui il sistema può rilevare e reagire alle minacce dipende dalla ricchezza dei processi decisionali e dalle risorse disponibili, ma la semplicità degli strumenti, la diffusa applicabilità e le possibilità di sviluppo rendono il sistema un'interessante risposta alle esigenze di cybersecurity che sempre più spesso trovano spazio all'interno delle strategie aziendali.

Bibliografia

[1] Gartner, https://www.gartner.com/it-glossary/operational-technology-ot/

[2] *IT/OT Convergence - Moving Digital Manufacturing Forward*, Cisco Systems, 2018

[3] *Aspetti di sicurezza si sistemi SCADA e Smart Metering*, Hewlett-Packard Development Company, 2015

[4] *Direttiva (UE) 2016/1148 del Parlamento Europeo e del Consiglio del 6 luglio 2016 recante misure per un livello comune elevato di sicurezza delle reti e dei sistemi informativi nell'Unione*, Gazzetta ufficiale dell'Unione Europea, 19 luglio 2016.

[5] Energy & Strategy Group, *Energy Cybersecurity Report 2018*, Politecnico di Milano - Dipartimento di Ingegneria Gestionale, Milano, 2018.

[6] N. Falliere, L. O Murchu, E. Chien, *W32.Stuxnet Dossier*, Symantec, 2011

[7] B. Johnson, D. Caban, M. Krotofil, D. Scali, N. Brubaker, C. Glyer, *Attackers Deploy New ICS Attack Framework "TRITON" and Cause Operational Disruption to Critical Infrastructure*, 14 Dicembre 2017, https://www.fireeye.com/blog/threat-research/2017/12/attackers-deploy-new-ics-attack-framework-triton.html

[8] *Dragonfly Security Response*, Symantec, 2014.

[9] *BlackEnergy Security Report*, ThreatSTOP, 2016.

[10] A. Cherepanov, *WIN32/INDUSTROYER - A new threat for industrial control systems*, ESET, 2017.

[11] *Roadmap to Secure Control Systems in the Water Sector*, Water Sector Coordinating Council Cyber Security Working Group, 2008.

[12] D. Dickinson, *Protecting Water Industry Control and SCADA Systems from Cyber Attacks*, Phoenix Contact, 2010.

[13] M. Alvarez, *Security trends in the healthcare industry*, IBM Security, Somers, NY, 2017.

[14] *Top Cyber Security Risks In Healthcare*, INFOSEC Institute, 2016, https://resources.infosecinstitute.com/category/healthcare-information-security/healthcare-cyber-threat-landscape/top-cyber-security-risks-in-healthcare.

[15] *Technical Guidelines for the implementation of minimum security measures for Digital Service Providers*, ENISA, Dicembre 2016

[16] *Best Current Operational Practices*, Euro-IX, `https://www.euro-ix.net/en/forixps/set-ixp/ixp-bcops/`.

[17] *Policy Brief - Internet Exchange Points*, Internet Society, 30 October 2015.

[18] H. Liao, C. R. Lin, Y. Lin, K. Tung, *Intrusion detection system: A comprehensive review*, Journal of Network and Computer Applications, January 2013

[19] R. E. A. Eimann, *Network Event Detection with Entropy Measures*, The University of Auckland, 2008

[20] J. B. Cabrera, L. Lewis, X. Quin, W. Lee, R. K. Mehra, *Proactive intrusion detection and distributed denial of service attacksa case study in security management*, Journal of Network and Systems Management, 2002

[21] J. Yu, H. Lee, M. S. Kim, D. Park, *Traffic flooding attack detection with SNMP MIB using SVM*, Computer Communications, 2008

[22] W. Cerroni, G. Moro, R. Pasolini, M. Ramilli, *Decentralized detection of network attacks through P2P data clustering of SNMP data*, Computers & Security, 2015

[23] P. M. Priya, V. Akilandeswari, S. M. Shalinie, V. Lavanya, M. S. Priya, *The Protocol Independent Detection and Classification (PIDC) system for DRDoS attack*, In Recent Trends in Information Technology (ICRTIT), IEEE, 2014

[24] A. Ghazi, E. A. Mouhammd Al-kassassbeh, *Exploiting SNMP-MIB Data to Detect Network Anomalies using Machine Learning Techniques*, Intelligent Systems Conference, 2018

[25] D. Gann, E. Autio, Y. Guo, A. Leiponen, R. Ozaki, J. Polak, N. Shah, E. Yeatman, *Digital City Exchange: A New Digital Economy Programme*, Digital Engagement, 15 Nov 2011.

[26] D. Birch, O. Tsinalis, K. H. van Dam, C. Lee, D. Silvia, C. Wu, M. Ghanem, Y. Gue, *Concinnity: A Digital City Exchange Platform*, Digital Economy, 2013.

[27] D. Silva, M. Ghanem, Y. Guo, *Wikisensing: An online collaborative approach for sensor data management*, Sensors, 2012.

[28] D. Birch, P. H. Kelly, A. J. Field, *Computationally unifying urban masterplanning*, ACM Computing Fronties, May 2013.

[29] K. H. van Dam, *Modelling electric vehicle demand in London using the DCE platform*, Systems-NET Webinar series, 9 April 2014.

[30] *https://www.zabbix.com/*

[31] *Apache NiFi* `https://nifi.apache.org/index.html`

[32] N. Tuptuk, S. Hailes, *Security of smart manufacturing systems*, Journal of Manufacturing Systems, 2018

[33] IETF, *RFC 4022 - Management Information Base for the Transmission Control Protocol (TCP)* `https://tools.ietf.org/html/rfc4022`