UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

DIPARTIMENTO DI SCIENZE TEORICHE E APPLICATE

CORSO DI STUDIO TRIENNALE IN
INFORMATICA

Analisi dei modelli di minaccia per sistemi ibridi IoT in ambito smart home

Relatore:
Prof.ssa Sabrina Sicari

Correlatore:
Dr. Fulvio Valenza

Tesi di Laurea di:
Gabriele Sassi
Matricola 745892

Anno accademico:
2022/2023

"You could stop the rest of your IT and pull all your resources into security for a year, and still not be".

Owen O'Connor.

Indice

Elenco delle figure

Capitolo 1

Introduzione

L'avanzamento delle nuove tecnologie ha portato ad un'enorme diffusione dei dispositivi *smart*, che sono diventati sempre più accessibili e convenienti per l'utilizzo quotidiano, integrandosi rapidamente nei diversi livelli di società. *Internet of Things*, punto di congiunzione tra il mondo fisico e quello digitale, si è imposto come paradigma chiave alla base dei sistemi *smart*. Tuttavia, l'aumento di questi dispositivi connessi in rete ha portato anche ad un aumento delle minacce alla sicurezza e della vulnerabilità dei dati personali, diventando argomenti di primaria importanza, poiché la tecnologia continua ad avanzare, insieme alla quantità di dispositivi IoT connessi in rete.

A causa della loro complessità, della natura dei componenti e dell'ampia estensione, le falle di sicurezza all'interno di questi ecosistemi possono manifestarsi su diversi livelli e avere un impatto e delle conseguenze molto differenti. Le proprietà di sicurezza di questi sistemi dipendono dalle caratteristiche *hardware* e *software* dei dispositivi e dalle vulnerabilità presenti nei *firmware* proprietari. Ad esempio, la mancata implementazione di funzioni d'autenticazione robuste o la mancanza di aggiornamenti *software* possono rendere i dispositivi facili prede di intrusioni esterne. Inoltre, la mancanza di uniformità degli standard di comunicazione tra i componenti di raccolta dati e le unità di elaborazione possono rendere difficile la gestione della sicurezza delle informazioni raccolte. Anche il fattore umano rappresenta un altro importante aspetto critico per garantire la sicurezza dei dispositivi IoT: gli utenti che non hanno una conoscenza adeguata della sicurezza informatica possono commettere errori come l'utilizzo di *password* deboli o la mancata protezione dei propri dispositivi da eventuali attacchi. Pertanto, diventa fondamentale assicurare integrità e confidenzialità dei dati trasmessi tra le varie componenti, autenticare ed autorizzare solo ed esclusivamente le entità coivolte nell'iterazione, proteggere in maniera continuativa il sistema da eventuali intrusioni, prendere in considerazione le minacce alla sicurezza derivanti dal loro utilizzo e "sfruttarle" per rispondere tempestivamente ad un attacco in maniera definitiva o anche solo contenitiva.

Il *threat modeling* rappresenta una possibile soluzione a queste problematiche. Esso è una metodologia utile ad identificare le minacce informatiche che possono colpire un sistema o un'applicazione. Attraverso diverse fasi l'obiettivo è quello di valutare il rischio ed identificare soluzioni per mitigare un'azione malevola e per migliorare la sicurezza del sistema, dopo aver definito il dominio applicativo e la natura delle componenti del sistema oggetto di analisi.

Lo scopo di questo lavoro di tesi è condurre una *threat analysis* derivante da un *threat model*, applicando tale approccio ad uno scenario IoT in ambito smart home. In particolare, si cercherà, attraverso *TAMELESS*, un *tool* di valutazione delle vulnerabilità per sitemi ibridi, di identificare le minacce alla sicurezza associate all'uso di dispositivi IoT in una casa domotica, analizzare i rischi derivanti dalle minacce identificate e definire delle possibili contromisure per mitigare i rischi e proteggere i dati personali.

La tesi è suddivisa nel modo seguente:

- **Capitolo 2:** presenta gli aspetti più rilevanti riguardanti lo stato dell'arte delle tecnologie coinvolte nell' IoT, argomentando come queste dipendono fortemente dalla sicurezza fornita;
- **Capitolo 3:** descrive le tecnologie studiate e l'architettura del sistema, approfondendo i concetti chiave e gli strumenti utilizzati ai fini dell'analisi di sicurezza quali *threat model*, *threat analysis* e TAMELESS;
- **Capitolo 4:** presenta lo scenario proposto, la modellazione e l'analisi del lavoro svolto, motivando la sua implementazione;
- **Capitolo 5:** descrive i risultati ottenuti dal processo di valutazione delle minacce in ambito smart home;
- **Capitolo 6:** discute le conclusioni, le considerazioni sul lavoro svolto e sui possibili sviluppi futuri.

Capitolo 2

Stato dell'arte

Nell'IoT, gli oggetti fisici e virtuali interagiscono e comunicano scambiandosi informazioni sull'ambiente e sul contesto in cui si trovano. L'eterogeneità delle tecnologie e l'enorme quantità di dati gestiti portano a nuove sfide nel campo della sicurezza, richiedendo che vengano progettati e standardizzati servizi in grado di offrire agli utenti finali un'interfaccia con cui interagire in modo semplice e sicuro, consentendo di raccogliere e sfruttare i dati richiesti evitando qualsiasi rischio per la loro sicurezza e *privacy*. Un punto cruciale è quindi la definzione di metodi in grado di valutare il livello di sicurezza e *privacy* dei dati eterogenei in ingresso, dei dispositivi connessi e delle interazioni che le differenti tecnologie realizzano con l'utente al fine di scegliere i dati adatti e soddisfare i requisiti dei servizi: tali esigenze sono affrontate in questo lavoro [1]. Tuttavia, l'architettura originale presentata non definisce meccanismi di supporto per il controllo degli accessi e per la gestione delle comunicazioni, quindi si è reso necessario ridimensionare il sistema superando tali limitazioni, adottando delle politiche specifiche per l'IoT, in grado di gestire le interazioni tra le entità coinvolte nell'ambito di politiche ben definite. La soluzione proposta in questo articolo [2], è in grado di garantire qualità, sicurezza e *privacy* dei dati anche in presenza di tentativi di violazione.
Come riportato inizialmente, i dati simboleggiano un'entità molto sensibile all'interno degli scenari IoT, ma questi non sono gli unici a rappresentare un potenziale punto d'accesso per l'attaccante.

L'abbondanza di sorgenti eterogenee e la diversificazione di *firmware*, *hardware* e standard di comunicazione degli oggetti interconnessi aumentano notevolmente il rischio di violazioni. Un passo importante consiste nell'adottare strategie di valutazione del rischio con l'obiettivo di scrutare la robustezza e l'affidabilità delle componenti dell'ambiente IoT, adottando approcci che considerano in modo bilanciato sia gli aspetti statici che dinamici, in modo da prevenire attacchi malevoli. In diverse ricerce sono state affrontate questioni riguardanti la valutazione del rischio in contesti applicativi vari.
Ad esempio, gli autori in [3], nel campo dei servizi *cloud*, identificano tre modelli di rischio focalizzati su un architettura di casa intelligente. Questi modelli includono il rischio legale, valutando la protezione della *privacy* dei dati degli utenti nell'archiviazione *cloud*, il rischio di guasto degli elettrodomestici legato al rispetto del protocollo della smart home e il rischio di sicurezza delle risorse, sorvegliando possibili minacce esterne ai dispositivi.

Cosi facendo è possibile rivelare problemi potenziali che potrebbero influenzare la sicurezza o il corretto funzionamento del sistema, fornendo ai proprietari la possibilità di intervenire per ripristinare la situazione in modo da preservare le proprietà di sicurezza e gli utenti stessi. Rimanendo in tema, un'altra ricerca interessante è stata presentata in[4], dove viene esaminata l'analisi dei rischi all'interno di un sistema di automazione domestica intelligente.
Il lavoro coinvolge dispositivi mobili e IoT, permettendo il controllo da remoto delle funzioni della casa. E' rilevante notare che monitorare le azioni degli utenti all'interno di tale ambiente può comportare la raccolta e l'abuso di dati personali, introducendo vulnerabilità e ampliando gli spazi d'attacco.
L'approccio adottato per la valutazione si basa sul metodo ben noto *ISRA* (*Information Security Risk Analysis*)[5]: questo metodo suddivide il sistema in cinque componenti diverse valutando il rischio di ciascuna, considerando la loro capacità nel preservare i principi di sicurezza fondamentali, quali riservatezza, integrità e disponibilità.

In generale, nella letteratura è stato investito uno sforzo notevole nello sviluppo di modelli e analisi delle minacce. Le sfide di sicurezza uniche derivano dalle prospettive offerte da questi sistemi intelligenti avanzati; infatti riconsocere le possibili minacce in tali sistemi e garantire una protezione adeguata richiede un livello di complessità superiore rispetto ai sistemi informatici tradizionali. Ciò è dovuto a varie ragioni, in particolare dall'integrazione di elementi umani e fisici nella progettazione. I dispositivi intelligenti interagiscono in diverse modalità con gli esseri umani, rendendo i sistemi IoT vulnerabili ad attacchi che sfruttano debolezze fisiche, umane e informatiche, i quali possono essere combinati dinamicamente e in modi complessi. Violare fisicamente l'accesso ad un sistema di sorveglianza, per esempio, potrebbe consentire ad un aggressore di compromettere l'intera rete ad esso collegata. Negli ultimi tempi, l'analisi delle minacce per i sistemi *cyber-fisici* e intelligenti è diventata un argomento di grande interesse nel campo della sicurezza informatica. Alcuni risultati promettenti sono evidenziati in [6], [7], [8], [9], dove gli autori presentano nuovi modelli di minaccia. Tuttavia le attuali prospettive e approcci di analisi non affrontano in maniera integrata gli aspetti informatici, fisici e umani del sistema, concentrandosi principalmente su uno dei tre elementi tra gli aspetti sopra indicati. Oltre agli studi citati in precedenza, per esempio, gli autori in [10] e [11] propongono diverse metodologie per la modellazione delle minacce per gli attacchi informatici; o ancora in [12] e [6], dove vengono considerati solamente due tra i diversi aspetti di natura differente. Esiste però una mancanza di analisi che considerino in modo completo tutte e tre le dimensioni e come potrebbero interagire tra loro, in modo da analizzare vulnerabilità e possibili contromisure. Tsigkanos et al. [7], [8] hanno delineato un approccio consapevole della sicurezza adattiva per i sistemi *cyber-fisici*, ponendo l'attenzione sull'interazione tra spazi *cyber* e fisici all'interno dell'ambiente operativo.

Sebbene condividano alcune idee con il lavoro di questa tesi, il loro obiettivo differisce: mirano ad individuare potenziali violazioni dei requisiti di sicurezza, attraverso un' analisi speculativa delle minacce, invece che concentrarsi sull'identificazione degli elementi non affidabili del sistema preso in considerazione. Lemaire et al. [6] hanno presentato un'applicazione che effettua in modo formale l'analisi delle minacce nei Sistemi di Controllo Industriale (ICS). Utilizzando un sistema basato sulla conoscenza, l'applicazione estrae vulnerabilità sia a livello di componente che di sistema, in modo che gli architetti della sicurezza possono successivamente adattare il sistema stesso per mitigare o eliminarle. Akella et al. [13] hanno sviluppato un metodo formale per rilevare attacchi nelle comunicazioni riservate. L'approccio proposto identifica minacce analizzando le interazioni tra sistemi fisici e componenti informatiche. Attraverso l'osservazione del comportamento di queste ultime è possibile dedurre informazioni sensibili riguardanti un componente fisico. Rocchetto e Tippenhauer [14] hanno esaminato vari modelli e profili di aggressori proposti in letteratura, concentrando la loro attenzione sui sistemi *cyber-fisici* e i sistemi di controllo industriale. Essi hanno definito una tassonomia e individuato diversi profili di aggressori (come utente base, criminale informatico, hacktivista, stato-nazione) in base a misure degli attributi stabiliti nella tassonomia. Il loro *framework* sembra non tenere conto dei tipi di attacchi ibridi considerati in questa tesi, ossia quelli che combinano fasi di attacco fisico e informatico come parte delle stesso attacco. Essi considerano due dimensioni di attacco rilevanti: la dimensione Scopo-Fisico e Scopo-Virtuale, che indica se l'obiettivo dell'attaccante è una componente fisica o virtuale, e la distanza dell'attaccante rispetto al bersaglio.

Per ovviare a tali limitazioni, l'articolo di riferimento [15] propone di sviluppare un nuovo approccio di modellazione delle minacce su sistemi definiti "ibridi", che includono aspetti fisici, umani e digitali, esplorando le loro interconnessioni e identificando i punti deboli. In questo studio si estende, modifica e arricchisce il modello chiamato FATHoM sviluppato dagli autori in [11] per l'analisi delle minacce nei sistemi virtualizzati (*cloud*). Gli autori in [15] si ispirano alla descrizione formale fornita da FATHoM per quanto riguarda le relazioni tra i componenti del sistema e le proprietà di sicurezza. Tuttavia, essi vanno oltre, introducendo nuovi elementi, definendo nuove regole di derivazione e creando una rappresentazione del grafo d'attacco. L'obiettivo di queste aggiunte è quello di modellare e rilevare in modo efficace le minacce ibride all'interno dei sistemi intelligenti, tenendo conto della loro tripla natura.

Per una comprensione più approfondita di questi sistemi e del loro significato in contesti reali, il documento analizzato introduce alcuni casi emblematici: L'attacco *"jackpot"*, ad esempio, sfrutta le debolezze nei componenti fisici e creando una piccola apertura vicino al tastierino del codice PIN, l'aggressore è in grado di collegare un cavo ad un *laptop* e ordinare al *bancomat* di erogare l'intera quantità di denaro disponibile. Altri attacchi sfruttano le fragilità umane, un esempio è la corruzione dei dipendenti, introducendoli a cedere informazioni sensibili o fornire dettagli rilevanti per condurre attacchi di *phishing* contro la clientela.

Concludendo, stabilire uno standard risulta essere un compito estremamemente complesso. Ciò è dovuto al fatto che la modellazione e l'analisi dei rischi correlati è strettamente vincolata dai contesti applicativi, i quali spesso presentano esigenze di sicurezza e *privacy* che variano. In relazione a ciò, dall'analisi approfondita dello stato attuale delle ricerche emerge che finora mancano metodi standard concordati o ampiamente conosciuti dalla comunità scientifica, specialmente in contesti ampi e applicazioni di vasta portata.

Capitolo 3

Tecnologie coinvolte e sicurezza

In questo capitolo verranno esposti gli aspetti chiave relativi all'IoT, le tecnologie, i linguaggi e gli strumenti utilizzati ai fini del lavoro di tesi proposto. Verranno infine analizzati i principali standard di comunicazione sfruttati dai dispositivi per adattarsi alle esigenze delle diverse applicazioni (si veda Figura 3.1), con particolare attenzione ai limiti e alle fragilità che caratterizzano la sicurezza di questi protocolli.

3.1 Internet of Things (IoT)

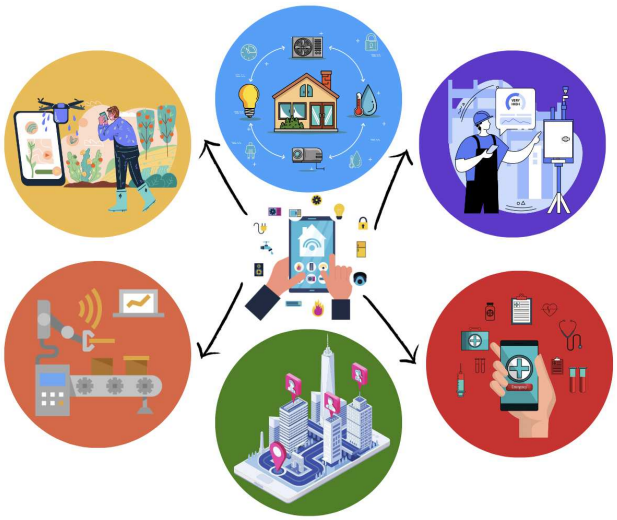

Figura 3.1: Principali domini applicativi dell'IoT.

La nascita dell'IoT risale agli anni '80 con l'esigenza di incrementare la cosidetta *pervasiveness*[1] dell'informatica, rendendo l'informazione e l'interconnessione digitale parte integrante della vita umana, anche se il termine viene utilizzato per la prima volta nel 1999 da Kevin Ashton[16].
Attualmente non si è ancora riusciti a dare una definizione concreta di IoT, ma generalmente si tratta di una rete in rapida crescita di dispositivi connessi a Internet e in grado di comunicare tra loro.

[1]Questo termine si riferisce alla diffusione capillare delle tecnologie informatiche in tutti gli aspetti della vita quotidiana.

Può essere quindi delineato sotto due particolari aspetti: dal punto di vista tecnologico, il termine *"Things"* è usato per denotare vari oggetti fisici di uso quotidiano che incorporano l'elettronica per renderli intelligenti e adatti a far parte di un'infrastruttura di rete globale. Da un punto di vista logico, un sistema IoT può essere caratterizzato come un insieme di dispositivi intelligenti che interagiscono su base collaborativa per raggiungere un obiettivo comune, acquisendo dati e agendo sull'ambiente in cui si trovano[2]. Questi dispositivi possono riguardare scenari di diverso dominio applicativo come *smart home*, *smart city*, *e-health*, *smart agriculture*, *smart transportation* e *smart learning* e vanno dagli elettrodomestici intelligenti ai macchinari industriali passando per dispositivi di monitoraggio della salute. L'IoT ha il potenziale per rivoluzionare il modo in cui interagiamo con la tecnologia e il mondo che ci circonda, creando nuove opportunità di efficienza, praticità e innovazione, con la capacità di raccogliere grandi quantità di dati da un'ampia gamma di dispositivi al fine di ottimizzare i processi, monitorare la vita umana e facilitare le operazioni quotidiane.

3.1.1 Architettura

Al fine di creare una rete sicura ed affidabile che rispetti le caratteristiche chiave dell'IoT, nel corso degli anni, diversi gruppi d'interesse e ricercatori cercano di definire un'architettura universale per l'IoT[17].
Inizialmente, a livello architetturale il sistema IoT era strutturato su tre livelli base, come rappresentato in Figura 3.2:

- ***Application*;**
- ***Network***;
- ***Perception***.

Nello specifico, il livello più basso dell'architettura è il *perception layer*, ovvero il livello fisico che include tutti i dispositivi come sensori, attuatori e sistemi di raccolta dati progettati per acquisire ed elaborare le informazioni interagendo con l'ambiente circostante; di conseguenza è fondamentale identificare ogni oggetto che comunica con un dispositivo in modo univoco[18].
Il *network layer*, livello intermedio, è considerato il cervello del dispositivo. Gestisce la connettività dei *device* intelligenti e la comunicazione tra di essi, garantendo che i dati raccolti siano trasmessi in modo affidabile e sicuro.
Alcune delle funzionalità tipiche includono: instradamento dei dati tra dispositivi IoT, gestione dell'indirizzamento e del traffico di rete.
Infine, l'*application layer* è il livello più alto dell'architettura, responsabile delle applicazioni specifiche che utilizzano i dati raccolti dai dispositivi IoT. Esso fornisce un'interfaccia tra l'utente o l'applicazione e la rete IoT, consentendo ad essi di accedere alle informazioni registrate e di utilizzarle per scopi specifici.

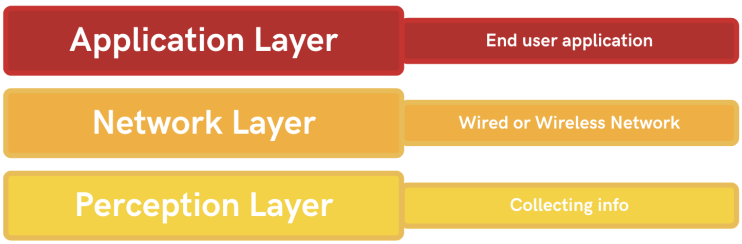

Figura 3.2: L'architettura a tre livelli dell'IoT.

A causa del miglioramento dell'IoT, con sistemi sempre più articolati, con un numero crescente di dispositivi connessi e una grande quantità di dati generati dagli stessi si è reso necessario procedere all'aggiunta di nuovi livelli all'architettura di rete dell'*Internet of Things* come mostrato in Figura 3.3, permettendo di migliorare interoperabilità, scalabilità, sicurezza e *privacy* e per consentire l'implementazione di ulteriori funzionalità[19].
I nuovi livelli includono:

- ***Business;***
- ***Application;***
- ***Processing;***
- ***Transport;***
- ***Perception***.

In particolare, *perception* e *application layer* svolgono le medesime funzioni precedentemente descritte. Il *trasport layer* si occupa di garantire la trasmissione delle informazioni tramite l'ausilio di tecnologie come LAN, 3G e 4G interfacciandosi con il livello *perception* e il livello successivo.
Questo livello è chiamato *processing* o *middleware*, può essere implementato in diversi modi a seconda delle specifiche esigenze dell'applicazione ed ha come obiettivo principale quello di assicurare che i dati raccolti dai dispositivi vengano elaborati in modo efficace e utile per l'applicazione IoT in questione. Le informazioni inviate dal *transport layer* vengono memorizzate in *cloud* o su specifici *database* collegati. L'ultimo, il *business layer*, rappresenta un livello innovativo fondamentale che si concentra sulla gestione dei processi di *business* dell'applicazione IoT, creando visualizzazioni o rappresentazioni grafiche per comprendere in modo rapido e molto semplice i dati. Ciò si traduce in una migliore strategia da comprendere in base alla quantità di dati accurati ricevuti dall'*application layer* e dal processo di analisi dei dati. Sulla base dei risultati dell'analisi, aiuterà i responsabili funzionali e prendere decisioni più accurate e flessibili sulle strategie aziendali e piani futuri[20].

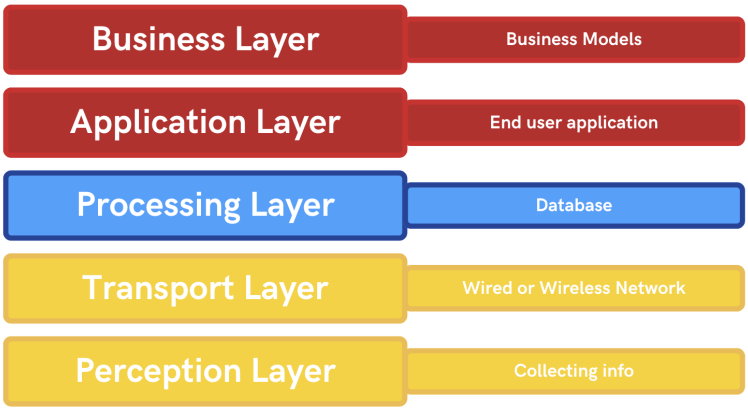

Figura 3.3: L'architettura a cinque livelli dell'IoT.

3.1.2 Wireless Sensor Network

Wireless Sensor Network (WSN) è una delle tecnologie abilitanti all'IoT ed è ampiamente utilizzata in questo lavoro di tesi. Sono reti estremamente dense, composte da un elevato numero di nodi cooperanti, che prendono il nome di "*sensor nodes*". Questi nodi lavorano attraverso comunicazioni *wireless multi-hop*[2], in cui ciascun nodo non solo rileva e trasmette i propri dati (***sensing***), ma anche quelli provenienti da altri nodi (***forwarding***). Tutte le informazioni raccolte vengono aggregate da un nodo denominato *"sink"*.
Questo elemento, come si può osservare in Figura 3.4, ha il compito di archiviare i dati internamente per poi instradarli attraverso una rete "convenzionale", come Internet, satelliti, ecc.

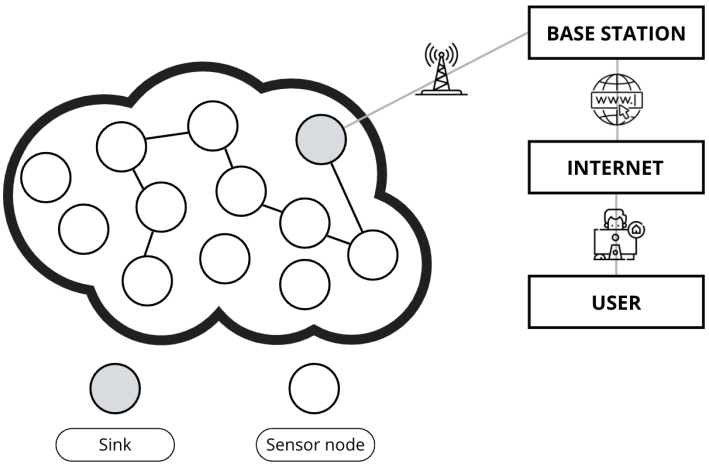

Figura 3.4: Esempio di *Wireless Sensor Network*.

[2]contrariamente ad una connessione diretta tra mittente e destinatario, i dati vengono trasmessi attraverso nodi intermedi.

Attualmente è in corso un'accelerazione nell'implementazione delle reti di sensori *wireless*, preannunciando un futuro in cui il mondo potrebbe essere interconnesso da queste reti accessibili via Internet entro un arco temporale di 10-15 anni[21]. Questo sviluppo potrebbe essere interpretato come un'affermazione dell'Internet come struttura fisica tangibile. L'emergente tecnologia porta con sè un entusiasmante potenziale, con applicazioni che spaziano in molteplici ambiti, come l'ambiente, il settore medico, le applicazioni militari, il trasporto, l'intrattenimento, la gestione delle emergenze e la sicurezza nazionale. Per permettere tutto ciò è fondamentale valutare il livello *hardware*: un sensore deve avere peso, costi e dimensioni limitate, ma allo stesso tempo integrare tutte le funzionalità necessarie per l'applicazione, essere in grado di operare autonomamente e adattarsi all'ambiente in cui viene installato.
Nello specifico ogni nodo è composto da *sensing unit*, che misura le grandezze fisiche trasformando le misurazioni in valori digitali, *processing unit* formata da CPU e memoria di piccole dimensioni, adatta per eseguire operazioni *"on board"*[3], *modulo ricetrasmittente e antenna* e, in alcuni casi, *GPS e mobilizer* che permette di localizzare e spostare fisicamente il sensore.
Il componente fondamentale è rappresentato da *power unit*, una batteria che fornisce l'alimentazione necessaria agli altri componenti. Un *sensor node* ha solitamente risorse energetiche e potenza limitate e una batteria che spesso non è nè ricaricaribile nè sostituibile, quindi il suo *life-time* è breve. Al fine di massimizzare la durata del sensore è cruciale massimizzare l'efficienza energetica, in modo che la stessa quantità di energia prolunghi il funzionamento del nodo. Si rende necessario lo sviluppo di protocolli orientati al risparmio energetico (*"Power Aware"*), poichè la trasmissione delle informazioni rappresenta la fase maggiormente energivora. Pertanto, la soluzione consiste nel far sì che ogni nodo impieghi la sua energia per aggreggare i dati propri e quelli ricevuti, permettendo la successiva trasmissione di un unico dato.
Osservando le caratteristiche sopra elencate è opportuno considerare, nella fase di progettazione, ulteriori fattori che possono influenzare lo sviluppo di una rete di sensori, come:

- **Affidabilità:** capacità di mantenere le funzionalità senza alcuna interruzione dovuta ad eventuali problemi, come danni fisici e/o ambientali.
- **Scalabilità:** il comportamento della rete non deve cambiare drasticamente all'aggiunta/rimozione di nodi.
- **Costi di produzione:** il costo di ogni singolo nodo deve essere il più basso possibile ($< \$1$), ma i costi attuali sono ancora troppo alti.

[3]termine utilizzato per indicare che le operazioni avvengono o sono integrate direttamente all'interno del dispositivo, anzichè essere gestite o eseguite esternamente.

- **Mezzo trasmissivo:** generalmente la comunicazione avviene mediante onde elettromagnetiche a radiofrequenza, applicazioni particolari possono usare altri mezzi. Il mezzo trasmissivo determina banda, ritardi, raggio di copertura e bit d'errore.
- **Topologia:** cambia dinamicamente e con molta frequenza in seguito alle caratteristiche del sensore.

In aggiunta, per tutto quello che concerne la trasmissione delle informazioni nelle reti di sensori è fondamentale evidenziare che i protocolli/algoritmi di *routing* tradizionali non sono adeguati all'uso per via delle limitate proprietà ampiamente discusse. Sono stati allora definiti numerosi protocolli appositi per le WSN che idealmente dovrebbero selezionare sempre i percorsi più corti per ogni dato trasmesso, evitare che lo stesso dato venga inutilmente inviato più volte a uno stesso nodo, minimizzare il consumo di energia e sfruttare una conoscenza globale della topologia della rete.
Essi si suddividono in tre categorie:

- ***Data-centric*:** si focalizzano sui dati scambiati tra i nodi piuttosto che sull'identità dei nodi stessi.
- **Gerarchici:** schemi di comunicazione che sfruttano la suddivisione delle reti di sensori in diversi livelli al fine di ottimizzare la trasmissione.
- ***Location-based*:** si basano sulla posizione fisica dei *sensor nodes*.

Per WSN, spesso implementate in modalità *"ad hoc"*, il processo iniziale di routing implica l'individuazione dei nodi. Durante questa fase, vengono trasmessi sequenze di messaggi o pacchetti simultaneamente, creando tabelle che includono informazioni basilari riguardo i vicini, come ad esempio ID e posizione. Pertanto, è essenziale che i nodi abbiano una comprensione della loro posizione geografica prima di poter stabilire connessioni. Oltre a queste informazioni, le tabelle di routing spesso incorporano dati riguardanti energia residua dei nodi, i ritardi attraverso ciascun nodo e una valutazione della qualità della connettività. Definite le tabelle, la maggior parte degli algoritmi di routing guidano i messaggi da un punto di partenza a una destinazione basata su coordinate geografiche, anziché sull'identificativo. Un esempio di algoritmo che si attiene a questa logica è *"GF"(Geographic Forwarding)* dove il nodo trasmittente può calcolare quale vicino costituisce il passo successivo verso la meta sfruttando principi di distanza, e inoltrare il messaggio al nodo specifico. Un modello di routing altrettanto rilevante per le WSN è *"Direct Diffusion"*, il protocollo unisce aspetti di routing, interrogazioni e aggregazione di dati. In questa logica, viene diffusa una *query* che esprime interesse per i dati da nodi remoti. Un nodo con i dati pertinenti risponde fornendo un insieme di attributi-valori. Questa coppia viene indirizzata verso il nodo richiedente tramite gradienti definiti e aggiornati durante il processo di diffusione di *query* e risposte.

Oltre alle basi di routing appena presentate, ci sono molti altri problemi chiave da considerare nello sviluppo e soprattutto nella scelta del giusto protocollo. Un aspetto rilevante riguarda l'affidabilità, poiché i messaggi viaggiano su più salti è importante avere un'elevata affidabilità su ogni collegamento, tale livello può essere monitorato attraverso metriche, come ad esempio il rapporto di consegna. Per risparmiare energia, molte WSN posizionano i nodi in stati di sospensione. Un esempio è il protocollo *"SPIN-2"* che introduce una soglia sotto il quale il nodo passa a uno stato *"Dormant"*, riducendo la partecipazione per conservare l'energia propria e degli altri nodi.
Per alcune applicazioni, i messaggi devono arrivare a destinazione entro una scadenza, protocolli come *"Speed"* e *"RAP"* vengono sviluppati per dare priorità alla trasmissione di pacchetti, considerando la velocità una metrica fondamentale che unisce la scadenza e la distanza che un messaggio deve percorrere. La sicurezza rappresenta un concetto critico e ampiamente discusso: se esistono degli avversari, possono sfruttare un'ampia varietà di attacchi nell'algoritmo. Sfortunatamente, quasi tutti gli algoritmi di routing per WSN hanno ignorato la sicurezza e sono vulnerabili ad attacchi specifici (*black hole*, *replay*, *wormhole*, *DoS*). Considerando questa problematica, protocolli come *"SPIN"* hanno iniziato ad affrontare soluzioni di instradamento sicuro.
Infine, aspetti come mobilità e congestione devono essere analizzati quando si parla di routing nelle WSN. L'instradamento è complicato se l'origine o la destinazione del messaggio sono in movimento. Per sistemi più robusti e impegnativi che elaborano audio e video creando più traffico trasversale, risulta importante evitare di trasmettere una quantità di dati superiore alla capacità di gestione della rete.

3.1.3 Protocolli utilizzati

L'IoT coinvolge una vasta gamma di dispositivi, sensori e sistemi che operano in diversi contesti e scenari; i protocolli di comunicazione risultano quindi essenziali per garantire una comunicazione efficace ed efficiente.
La scelta del corretto standard dipende dalle caratteristiche e specifiche richieste dai dispositivi, come ad esempio, velocità di trasmissione, qualità del servizio, impatto energetico e sicurezza, quest'ultima di significativa importanza poichè presente sin dalle fasi di progettazione dello *stack* protocollare e non aggiunta come elemento successivo. Considerando la molteplicità di protocolli sviluppati, di seguito verranno analizzati nello specifico alcuni fondamentali:

- ***ZigBee;***
- ***Message Queue Telemetry Transport*** (**MQTT;**)
- ***Constrained Application Protocol*** (**CoAP**).

ZigBee

ZigBee è uno standard di comunicazione *wireless* a bassa potenza e basso consumo energetico, sviluppato per soddisfare esigenze di connettività per reti lente e *short range*, (come sensori ed attuatori) e progettato per essere facile da configurare, sicuro ed in grado di gestire un gran numero di dispositivi su una singola rete, come descritto in [22]. La rete ZigBee include tre diversi tipi di dispositivi:

- ***Coordinator*:** è il nodo centrale della rete, ha il compito di gestire la configurazione della rete e la sicurezza.
- ***Router*:** dispositivi intermediari che facilitano la comunicazione tra nodi, con la funzionalità di instradare le informazioni, e su particolari richieste, fungere da ripetitori.
- ***End devices*:** rappresentano i nodi finali della rete, acquisiscono e trasmettono le informazioni comunicando solo con i nodi identificati come genitori.

Il protocollo sviluppa tre differenti topologie di rete, come mostrato in Figura 3.5, in modo da permettere la cooperazione tra i numerosi *device* connessi, in base alle caratteristiche richieste. *"Stella"*, ovvero la topologia più semplice, adatta per reti con un numero limitato di nodi dove tutti i dispositivi si connettono ad un nodo centrale, chiamato *coordinator*, attraverso cui possono comunicare tra di loro. *"Cluster"*, secondo cui i dispositivi sono organizzati in gruppi gestiti da un *coordinator* che funge da nodo centrale per i nodi associati a quel *cluster*. E' una topologia adatta per reti che richiedono una certa organizzazione come, ad esempio, i sensori divisi in gruppi in base alla loro posizione o funzione, ciò consente una facile espansione della rete permettendo l'aggiunta di nuovi dispositivi senza interrompere la comunicazione. Infine la topologia *"mesh"*, più avanzata e scalabile, permette la comunicazione tra qualsiasi nodo all'interno della rete, anche se non direttamente collegati, questo è possibile grazie alla funzionalità di *routing* incorporata. Le sue caratteristiche implicano una maggiore sicurezza poichè i dati vengono trasmessi attraverso più percorsi, rendendo più difficile per un attaccante intercettare o manipolare dati. In questi tre casi i dispositivi oltre a comunicare tra loro possono operare come inoltri di un segnale, in modo da estenderne la portata e raggiungere la destinazione desiderata.

MQTT

MQTT è un protocollo di comunicazione di livello applicativo, ideale per scenari in cui viene richiesto un basso impatto energetico e computazionale sui singoli nodi. E' uno standard asincrono, basato su TCP, in modo da offrire forti garanzie e utilizza un modello di *publish/subscribe* per la comunicazione tra dispositivi. Un nodo server, chiamato ***broker*** o ***dispatcher***, si occupa di mettere in comunicazione i nodi client, chiamati ***publishers*** e ***subscribers***.

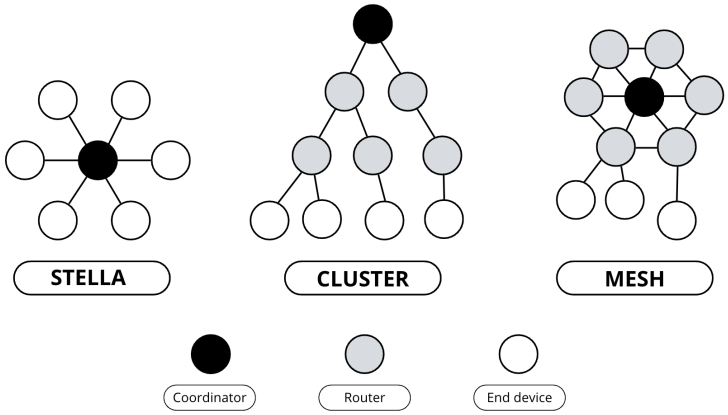

Figura 3.5: Topologia e tipi di nodi possibili in una rete ZigBee.

Il suo funzionamento avviene mediante operazioni e messaggi scambiati tra i nodi, come illustrato in Figura 3.6: I nodi publisher pubblicano notizie inerenti ad un certo *topic* sul nodo broker, inviando la notifica dell'argomento pubblicato. I nodi subscribers, iscritti ad un determinato topic, ricevono le notifiche dal broker, reperiscono la notizia e la leggono.

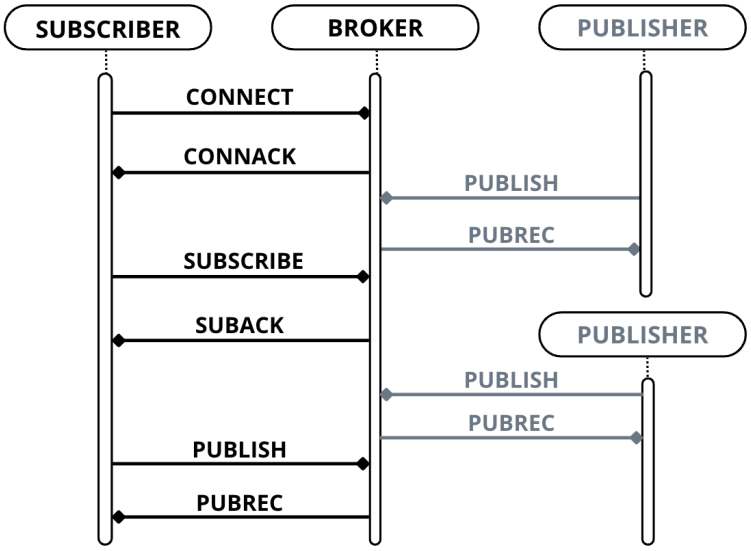

Figura 3.6: Esempio di flusso comunicativo in MQTT.

I topic sono argomenti rappresentati da stringhe e organizzati in uno o più livelli, separati da "/", come un semplice *file system*. I nodi client si iscrivono a determinati argomenti, pubblicati dai pusblisher. Il nodo broker utilizza i topic di un messaggio per decidere quali client devono riceverlo e vengono rimossi quando non sono più sottoscritti. Quando un client si iscrive ad un topic, può farlo in maniera puntuale, specificando il nome esatto del topic, oppure utilizzare ***wildcards*** per iscriversi a più topic contemporaneamente.

Esistono due tipi differenti di wildcards:

- ***Single-Level*(+):** utilizzato per sottoscrivere un solo livello di topic. Considerando l'esempio in Figura 3.7; se viene effettuata l' iscrizione a questo topic, si riceveranno messaggi da argomenti che contengono qualunque scritta al posto del simbolo +.

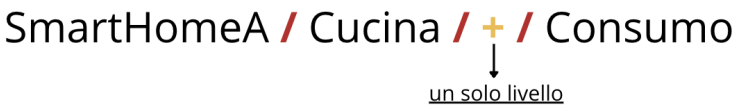

Figura 3.7: Esempio di *topic* singolo livello.

- ***Multi-Level*(#):** utilizzato per sottoscrivere uno o più livelli di topic, il simbolo viene posizionato come ultimo carattere e preceduto da "/". Osservando la Figura 3.8, la sottoscrizione a questo topic comporta la ricezione di tutti i messaggi di un argomento che inizia con il modello prima del carattere #.

Figura 3.8: Esempio di *topic* multi livello.

Un ulteriore caratteristica fondamentale di MQTT è l'inclusione di una meccanismo di ***Quality of Service*** (**QoS**), che consente ai client di specificare il livello di affidabilità richieste per la trasmissione dei messaggi, coinvolgendo entrambi le fasi di consegna dello stesso; da publisher a broker e da broker a subscriber, con la possibilità di utilizzare due diverse configurazioni.
Sono disponibili tre livelli:

- **0: "*at most once*":** il client invia un messaggio al broker senza attendere alcuna conferma di ricezione ed un eventuale rinvio. Cosi facendo si adotta un approcio *"best effort"* e non si ha nessuna garanzia di consegna, riscontrando possibili ritardi dei messaggi e consegna di duplicati.
- **1: "*at least one*":** il mittente mantiene in memoria il messaggio finchè non riceve l'ACK di avvenuta ricezione. Se non lo si riceve entro un tempo limite viene reinviato. Questo livello garantisce l'arrivo dei messaggi ma con duplicati, come ad esempio in reti congestionate quando il messaggio di conferma non viene ricevuto immediatamente.
- **2: "*exactly one*":** il processo prevede un "doppio rimbalzo" tra le parti comunicanti, al fine di confermare al mittente l'effettiva presa in carico del messaggio. Questa modalità comporta una minore velocità ma risulta più affidabile; il messaggio arriva esattamente una volta.

Riassumendo, la scelta del livello di qualità del servizio dipende da diversi fattori come affidabilità della rete ed importanza e frequenza dei messaggi inviati.

CoAP

CoAP è un protocollo di livello applicativo che esegue gli stessi compiti di *http* ma è adattato ai dispositivi *contraint*, ovvero limitati in termini di risorse, energia e operazioni. L'obiettivo principale di CoAP è offrire *web-based services* per i nodi vincolanti in esecuzione su reti *6LowPAN*[4].
I servizi vengono gestiti dai *server*, associati ad un indirizzo IP, tradotti tramite DNS ricorsivamente e iterativamente, identificati mediante URL e accessibili in modo asincrono, ovvero le informazioni possono arrivare al richiedente anche se memorizzate in tempi precedenti.
Il modello di transazione utilizzato (descritto in Figura 3.9) si basa su un sistema ***Client-Server*** mettendo a disposizione solo messaggi ***"request"***, dove il client chiede al server l'invio di un servizio, e ***"response"*** tramite cui il server risponde alla richiesta del client. Il protocollo utilizza una trasmissione definita *"stop and wait"*[5] e nello scambio di informazioni una tecnica di *"piggybacked"*, utile per ridurre i messaggi di controllo, cercando di inviare nello stesso pacchetto più informazioni, quali i dati effettivi, contenuti nel *Payload* e tante informazioni di controllo, incluse nell'*Header*, trasmesse contemporaneamente per realizzare in modo sicuro lo scambio.
Il protocollo di comunicazione utilizzato per l'invio dei messaggi è **UDP**, il che permette di ridurre il consumo energetico. Talvolta, client e server possono scambiarsi dei ***token*** come parte della comunicazione. Questi sono utilizzati per garantire la sicurezza e l'autenticazione dei messaggi trasmessi; rappresentano un campo opzionale nell'*Header* di pacchetto e sono associati ad una specifica richiesta.

---

[4]architettura di rete progettata per consentire la comunicazione di dispositivi a basso consumo attraverso operazioni di compressione dei pacchetti IPv6.

[5]garantisce che il mittente invii i dati in modo controllato e che il destinatario confermi l'avvenuta ricezione prima che nuovi dati siano trasmessi.

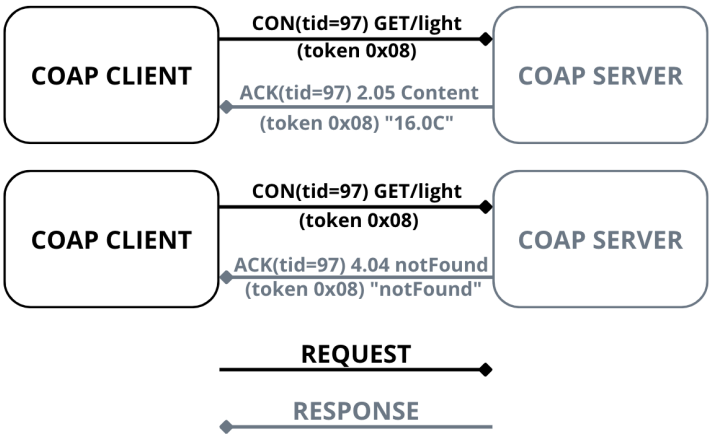

Figura 3.9: Simulazione del modello di transazione di CoAP.

Infine, risulta importante puntualizzare che siccome CoAP fa uso del protocollo UDP, è possibile implementare DTLS (*Datagram Transport Layer Security*) applicato ai datagrammi per la sicurezza dei dati: utilizza le medesime strategie crittografiche di TLS, ma è stato progettato per gestire le caratteristiche peculiari della trasmissione dei datagrammi come affidabilità dei pacchetti e potenziale perdita o ritrasmissione degli stessi. Inoltre fornisce una crittografia *end-to-end* dei dati trasmessi attraverso la rete, con la possibilità di autenticare client e server attraverso certificati digitali oltre a gestire le sessioni crittografiche. Questo evita il dispendio computazionale associato alla creazione di una nuova sessione crittografica per ogni singolo datagramma trasmesso. Tuttavia, l'inclusione di DTLS comporta un incremento delle dimensioni dei pacchetti, con conseguente aumento delle esigenze computazionali del dispositivo; requisiti che potrebbero non essere adempiuti da tutti i nodi.
I protocolli esaminati in questa ricerca presentano diversi vantaggi che si "sposano" perfettamente con le esigenze dell'IoT. Tuttavia risulta interessante approfondire i benefici specifici che possono derivare dall'adozione di CoAP e MQTT, considerati una valida alternativa rispetto al tradizionale protocollo http. Nella Tabella 3.10, si può osservare come CoAP risulti particolarmente efficiente in vari aspetti, come dimostrato in [23].

PARAMETRI	HTTP	COAP
Byte per trasmissione	1451	154
Consumi energetici (mW)	1333	141
Tempo di vita (giorni)	0.744	84

Figura 3.10: Confronto tra CoAP e HTTP.

Inoltre, la Tabella 3.11 mette in luce come MQTT consenta un notevole risparmio in termini di consumo, pur mantenendo un tasso di trasmissione e ricezione dati più elevato sia in connessioni 3G che Wi-Fi.

PARAMETRI		3G		WIFI	
		HTTPS	MQTT	HTTPS	MQTT
RICEZIONE	messages / hour	1,708	160,278	3,628	263,314
	% battery / msg	0.01709	0.00010	0.00095	0.00002
	msgs (note losses)	240 / 1024	1024 / 1024	524 / 1024	1024 / 1024
INVIO	msg / hour	1,926	21,685	5,229	23,184
	% battery / msg	0.00975	0.00082	0.00104	0.00016

Figura 3.11: Confronto tra MQTT e HTTP.

3.1.4 Limiti e problemi di sicurezza

Gartner[6] ipotizza che i *device smart* saranno quasi 6 miliardi e mezzo già il prossimo anno, mentre un altro analista, IHS Markit[7], guarda oltre e stima che saranno ben 75 miliardi nel 2025. La complessità nell'iterazione e nel controllo della pluralità di dispositivi implica una difficile gestione della sicurezza all'interno delle reti IoT. Come è possibile osservare dalle sezioni di testo precedenti, questa tecnologia si trova ad affrontare delle limitazioni di notevole rilevanza, sopratutto quando si considera la delicata gestione della *privacy* e della *confidenzialità* ed *integrità* dei dati. Queste limitazioni sono in gran parte intrinseche alla necessità primaria di ottimizzare e mantenere estremamente bassi i consumi energetici dei singoli dispositivi. Un ulteriore aspetto critico da considerare è la natura *wireless* delle tecnologie impiegate nell'IoT: poichè molti dispositivi comunicano tramite queste reti, si aprono varie possibilità per potenziali violazioni e attacchi informatici, siccome questi strumenti comunicativi sono essenzialmente più vulnerabili rispetto alle comunicazioni cablate; in quanto i segnali trasmessi possono essere intercettati più facilmente da attori malevoli. Si è reso quindi necessario trovare il giusto equilibro tra il livello di protezione garantito e le prestazioni raggiunte.

[6] `https://it.linkedin.com/company/gartner`

[7] `https://it.linkedin.com/company/ihsmarkit?trk=public_profile_topcard-current-company`

Per capire accuratamente i concetti è opportuno porre una significativa attenzione ad alcune proprietà di sicurezza riguardo i protocolli precedentemente discussi: MQTT, ad esempio, integra una semplice autenticazione nativa, basata sulla coppia *"username"* e *"password"*; risultando estremamente semplice per un attaccante entrare in possesso di tali informazioni per utilizzarle successivamente con scopi malevoli. Un'alternativa proposta è l'autenticazione con *One Time Password* (OTP) combinata con *blockchain* di Ethereum, analizzata in [24], oppure un approccio che si basa su crittografia ellittica e funzioni *hash* in modo da garantire autenticazione mediante un processo che include numeri casuali e chiavi segrete, come proposto in [25], riducendo l'uso di risorse grazie a pacchetti più piccoli e meno *handshake*. Nel protocollo MQTT, la riservatezza è uno dei punti più interessanti poichè nativamente non è previsto alcun meccanismo di cifratura dei dati, anche quando è in atto un processo di autenticazione. Questa vulnerabilità apre la strada a possibili attacchi passivi, come *"packet sniffing"*, attraverso cui un attaccante potrebbe intercettare e leggere informazioni scambiate tra client e broker. Poichè MQTT opera con TCP, è possibile mitigare questa problematica di sicurezza adottando il protocollo TLS. Questa soluzione però è praticabile fintanto che i dispositivi IoT coinvolti non presentano limiti eccessivi in termini di risorse. Contrariamente è necessario adottare tecniche differenti.
CoAP, sfrutta il protocollo DTLS per garantire integrità, autenticazione e riservatezza, ma non è dotato di un sistema integrato per la gestione delle autorizzazioni. Per affrontare questa carenza, è necessario sviluppare dei metodi *ad hoc*: gli autori in [23], ad esempio, propongono l'adozione di una soluzione denominata CoAP-ECC, che si basa sulla generazione di coppie di chiavi pubblica-privata per gestire autorizzazioni all'interno della rete, fondamentale per garantire che solo i dispositivi registrati siano in grado di interagire con successo attraverso il sistema CoAP. Se un'entità malintenzionata tentasse di comunicare, i suoi messaggi verrebbero scartati in quanto il sistema non riconoscerebbe il dispositivo tra quelli registrati.
ZigBee possiede invece di per sè strumenti per garantire le proprietà di sicurezza fondamentali. Più specificatamente, ciascun dispositivo conserva tre varietà di chiavi per gestire la crittografia. Inoltre il protocollo incorpora nativamente meccanismi per certificare la sicurezza delle comunicazioni *end-to-end*. Attraverso l'algoritmo AES in modalità *"Cryptographic Block Chipers"* cifra le informazioni, e parallelamente, garantisce autenticazione durante la trasmissione. Riguardo l'integrità dei dati, utilizza *"Message Integrity Code"* che rileva eventuali modifiche non autorizzate dei dati o errori causati da interferenze. Tuttavia, gli autori in [26] sottolineano che i terminali ZigBee non incorporano meccanismi che preservino la sicurezza da possibili manipolazioni fisiche ma risultano particolarmente sicuri in riferimento agli attacchi alle comunicazioni come, ad esempio, attacchi *"a replay"*.

Riassumendo, osservando la Tabella 3.12, l'IoT nonostante l'eterogeneità funzionale dei *device* e le immense potenzialità nel trasformare numerosi aspetti della vita quotidiana, si trova di fronte a sfide significative in termini di gestione della *privacy* e sicurezza, con l'obiettivo di impedire all'attaccante di prendere il controllo dei dispositivi, oltre che ad evitare la violazione di informazioni riservate o impedire l'accesso ad intrusi.
Il bilanciamento tra l'ottimizzazione dei consumi energetici e l'implementazione di solide misure di protezione rappresenta quindi un nodo cruciale da sciogliere per garantire un futuro "intelligentemente" sicuro ed affidabile.

Figura 3.12: Proprietà di sicurezza garantite dai protocolli IoT.

3.2 Analisi sulla sicurezza

Con il continuo evolversi di tecnologie, protocolli, connessioni e considerando una maggiore dipendenza dai dispositivi digitali e da Internet, sia in ambito privato che negli affari, basti solo considerare, ad esempio, la crescente integrazione della digitalizzazione e automazione nei processi produttivi; le minacce informatiche sono aumentate notevolmente. I risultati di un rapporto di *McAfee Enterprise*[8] e *FireEye*[9] indicano che durante la pandemia, l'81% delle organizzazioni globali ha subito un aumento delle minacce informatiche, con il 79% che ha sperimentato lunghi tempi di inattività a causa di un attacco informatico, con una conseguente perdita di dati, indisponibilità delle applicazioni e guasti alle apparecchiature utilizzate. La sicurezza rappresenta quindi un elemento indispensabile che deve essere considerato e gestito in maniera attiva, sin dalle prime fasi della progettazione di un prodotto e per tutto il suo ciclo di vita, adottando sia un approccio *proattivo*, con la previsione di eventuali minacce durante la progettazione, che *reattivo* permettendo di affrontare problemi inizialmente non previsti.

[8] https://www.mcafee.com/it-it/consumer-corporate/about.html?news_id=d9df8b95-3ba6-4906-9b93-713aef860127&PIFld=-1&rfhs=1&ctst=1

[9] https://fireeye.dev/docs/about/fireeye/

L'analisi di sicurezza può considerarsi come un processo di valutazione sistematica dei rischi di sicurezza associati ad un sistema informatico, con l'obiettivo di identificare eventuali vulnerabilità e stimare l'impatto di determinate o inattese circostanze negative, proponendo allo stesso tempo soluzioni per mitigare i rischi e criteri per valutare la bontà di queste protezioni.
Questo concetto è di fondamentale importanza per la progettazione e la gestione dei sistemi informatici sicuri. In un mondo in cui le minacce informatiche sono sempre più diffuse e sofisticate, è necessario adottare un approccio olistico alla sicurezza[10], che vada oltre la sola tecnologia, prendendo in considerazione tutti gli aspetti del sistema. L'analisi di sicurezza si basa su una serie di tecniche e metodologie di valutazione dei rischi, che consentono di identificare le minacce potenziali e le vulnerabilità del sistema, di valutare l'impatto dei rischi, l'analisi della vulnerabilità, la gestione degli accessi e delle identità, e la crittografia, che consentono di garantire la *privacy* e la sicurezza dei dati.
Prima di introdurre nello specifico alcune metodologie di analisi della sicurezza utilizzate in questo lavoro di tesi è importante chiarire degli aspetti fondamentali: una "minaccia" è l'obiettivo dell'attaccante, definita come una qualsiasi situazione potenzialmente in grado di danneggiare un sistema[27], provocando la distruzione delle risorse da esso possedute, la divulgazione o l'accesso non autorizzato ad informazioni riservate o una discontinuità nella fornitura di uno o più servizi. Una minaccia può essere frutto della natura di una certa entità, come ad esempio una falla di sicurezza nel *software* o nell'*hardware* di un sistema, causato da difetti o vulnerabilità presenti nel sistema o nel prodotto; ma può anche scaturire dall'abilità di un attaccante di sfruttare una o più debolezze del sistema stesso.
L'identificazione delle minacce è uno degli *step* fondamentali dell'approccio strutturato alla valutazione della sicurezza, ma spesso è necessario interrogarsi anche sulle relazioni dirette o indirette che intercorrono tra le entità coinvolte, al fine di individuare catene o percorsi d'attacco che non emergerebbero altrimenti. Pertanto, una chiara comprensione della natura e delle iterazioni tra entità,vulnerabilità e minacce interne ed esterne al sistema è fondamentale per la gestione del rischio e per la progettazione di sistemi sicuri. Questo processo, descritto in Figura 3.13, è noto come *"risk assessment"* o "analisi dei rischi".

[10] richiede una visione d'insieme e una strategia di protezione globale, che include, ad esempio la gestione dei processi, delle politiche e delle persone.

Figura 3.13: Schema del processo di *"risk assessment"*.

3.2.1 Threat Model

Un *threat model* è un'astrazione, frutto di un processo detto di *"threat modeling"* che studia i possibili modi in cui un attaccante può danneggiare gli *assets* di un sistema, evidenziando le potenziali fonti di minaccia, le cause e le ripercussioni che avrebbero sullo scenario, valutandone i rischi e soppesandone le contromisure, permettendo di comprendere soprattutto:

- **Risorse** che il prodotto sta cercando di proteggere;
- **Minacce** indrodotte dalle risorse;
- **Vulnerabilità**, ovvero tutti i punti deboli che vengono sfruttati.

Il *threat model* implica la comprensione della complessità del sistema e l'identificazione di tutte le possibili minacce, indipendentemente dal fatto che possano essere sfruttate o meno. Esso categorizza le informazioni disseminate nel contesto osservato attraverso una sintassi predefinita. E' caratterizzato da un linguaggio in grado di unire, grazie anche all'espressività di regole su cui si basa, al livello di dettaglio e di profondità richiesta da un'analisi di questo tipo, l'immediatezza e le capacità di sintesi dei formalismi che introduce.

Obiettivo

Lo scopo principale del *threat model* è quello di fornire una visione completa dello stato attuale delle minacce informatiche di un sistema preso in analisi, valutando il potenziale danno in modo da fornire successivamente una contromisura efficace.

L'identificazione delle minacce aiuta a sviluppare requisiti di sicurezza realistici e significativi. Questo è particolarmente importante, perchè se i requisiti di sicurezza sono difettosi, la definizione di sicurezza per quel sistema è difettosa, e quindi il sistema non può essere sicuro. La corretta identificazione delle minacce e la successiva selezione appropriata delle contromisure riduce la capacità dell'attaccante di abusare del sistema.

Sviluppo

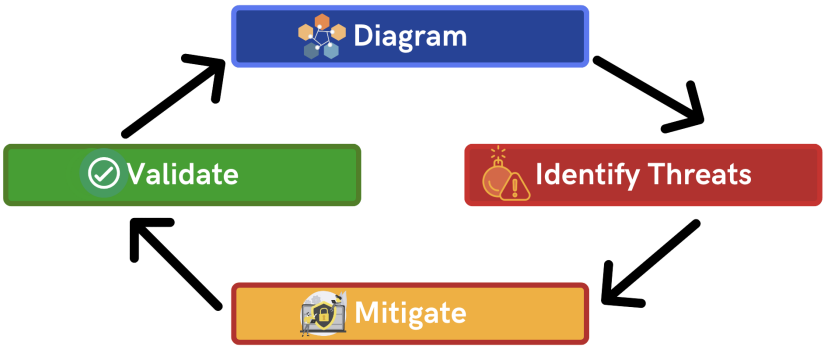

Figura 3.14: Fasi ad alto livello del processo di *threat modeling*.

Un modello di minaccia non può essere creato semplicemente facendo un *"brainstorming"* sulle possibili intenzioni di un attaccante. Pertanto, è importante essere sistematici durante tutto il processo, per garantire che il maggior numero di minacce e vulnerabilità venga scoperto dagli sviluppatori, non dagli aggressori[28]. Questo può includere diagrammi, schemi e descrizioni dettagliate delle minacce e dei loro potenziali impatti.
Un threat model quindi può essere espresso in una sequenza di operazioni solitamente riconducibili a quattro fasi ben distinte, rappresentate in Figura 3.14 ed elencate di seguito:

- **Caratterizzazione del sistema:** il primo passo consiste nel comprendere a fondo il sistema in questione ed il suo funzionamento (senza questo, la modellazione potrebbere essere "viziata"[11]), individuando le componenti e le relazioni piu rilevanti rappresentandole utilizzando una sintassi o dei diagrammi che ne riassumano le proprietà principali. Ciò di cui si ha bisogno è un modello di sistema che riveli le caratteristiche essenziali dello scenario esaminato. Ovviamente, maggiore è l'attenzione al dettaglio in questa fase più numerosi saranno gli spunti per quelle successive.

[11] trascurando importanti dettagli o vulnerabilità rilevanti potrebbero mettere a rischio la sicurezza del sistema.

- **Identificazione:** è importante che l'analista impari a pensare come l'attaccante: a partire dal resoconto ottenuto, l'identificazione delle risorse tangibili, come processi e dati, o concetti astratti come la coerenza dei dati e l'individuazione dei punti d'accesso (*socket* aperti, file di configurazione, ecc.) è un passaggio fondamentale. Si inizia cosi a riflettere, analizzando nodi e collegamenti, su quelle azioni specifiche che potrebbero recare pericolo agli *assets*[12] del sistema. Risorse e minacce sono strettamente collegate, una minaccia non può esistere senza una risorsa bersaglio. Le minacce possono essere identificate esaminando ciascuna di queste entità critiche e creando ipotesi di minaccia che violano la riservatezza, l'integrità o la disponibilità dell'entità. L'output rappresenterà un profilo delle minacce per il sistema, che in genere sono classificate in sei classi in base al loro effetto[27].
- **Contromisure:** per ogni possibile minaccia individuata vengono proposte delle contromisure in grado di contrastare le criticità che li alimentano. Avere una visione globale del comportamento, della stuttura del sistema e conoscere le proprietà di sicurezza che caratterizzano l'infrastruttura è essenziale per una buona riuscita del processo di valutazione delle soluzioni a disposizione.
- **Validazione:** la protezione dei sistemi riguarda i compromessi; spesso trovare un equilibro ideale è una vera sfida. E' quindi importante condurre un'approfondita sessione di gestione del rischio per analizzare l'adeguatezza delle soluzioni in base ai requisiti di sicurezza comuni al sistema. Questo ultimo *step* è volto ad esaminare la ricchezza e la precisione nella rappresentazione generata, la coerenza nell'identificazione delle minacce e l'efficacia delle contromisure; applicando il modello ad uno o più scenari noti o predeterminati. Le mancanze individuate durante questo passaggio offrono spunti per eventuali modifiche o raffinamenti dei processi precedenti.

3.2.2 Threat Analysis

"Threat model" e *"threat analysis"* sono due concetti cooperanti, entrambi legati alla sicurezza informatica, ma con significati leggermente diversi.
A differenza del *modelling*, l'analisi delle minacce è un processo più ampio e generale che costituisce un ciclo continuo, il quale si concentra sull'identificazione e l'analisi dei rischi specifici associati ad un sistema o un'applicazione, includendo:

[12]si riferisce a qualsiasi componente del sistema che svolge una funzionalità essenziale, come dati, persone, processi, programmi ed entità fisiche.

- **Identificazione e valutazione vulnerabilità:** rappresenta un passo cruciale nella gestione della sicurezza del sistema. E' un processo di individuazione e rilevamento delle potenziali debolezze o "falle" all'interno di un sistema informatico. Se sfruttati in modo opportuno da un attaccante potrebbero compromettere la sicurezza, l'integrità e la disponibilità del sistema stesso.
- **Identificazione degli obiettivi:** si individuano gli obiettivi sensibili o critici del sistema che potrebbero essere presi di mira dalle minacce. Possono includere dati sensibili, informazioni finanziarie, hardware e altro ancora.
- **Identificazione delle possibili minacce:** insieme di tecniche volte ad individuare, catalogare e valutare le diverse potenziali fonti di pericolo che potrebbero rappresentare un punto critico per la sicurezza del sistema, fornendo una base solida per lo sviluppo di strategie di protezione ed eventuali contromisure da adottare.
- **Identificazione dei possibili impatti:**: include una serie di operazioni che mirano a valutare e comprendere le conseguenze negative che potrebbero derivare da un attacco analizzato. Queste informazioni consentono di pianificare ed implementare contromisure adeguate per mitigare i rischi associati.

Inoltre, per condurre un'analisi completa delle minacce, è importante integrare nel processo una *"matrice del rischio 5x5"* (un esempio viene riportato in Figura 3.15) per determinare il livello di rischio in base alla *probabilità* e all' *impatto* di un attacco in un particolare scenario. Ciò è altrettanto utile poichè fornisce uno strumento prezioso per rappresentare in modo strutturato e visivo il livello d'influenza e d'effetto delle minacce. In questo modo l'utente può considerare un ulteriore elemento per valutare, priorizzare e gestire efficacemente i rischi di sicurezza e le azioni in relazione alla gravità dell'attacco.

Figura 3.15: Esempio di matrice del rischio.

Benefici

In generale, l'obiettivo principale della *threat analysis* è quello di fornire un insieme di informazioni specifiche tali da essere consultate e valutate, in modo da "imparare" dai risultati e prendere decisioni utili a proteggere *assets* digitali, dati sensibili e operazioni. Grazie a ciò, l'analisi di sicurezza consente alle organizzazioni di adottare misure preventive per mitigare i rischi prima che si verifichino incidenti di sicurezza, in modo da essere in grado di rispondere più rapidamente ed efficacemente ad eventuali danni, e identificare le aree in cui queste possono essere migliorate o rafforzate per affrontare ulteriori minacce emergenti. Inoltre garantisce che un sistema o un'applicazione sia conforme ai requisiti normativi e alle leggi sulla *privacy*, promuovendo parallelamente una maggiore cultura e consapevolezza alla sicurezza tra i soggetti direttamente o indirettamente coinvolti. In conclusione, l'analisi di sicurezza è un processo in miglioramento continuo e per garantire i benefici sopra citati è fondamentale integrare in modo evolutivo,costante e continuativo l'intero processo di analisi supportato da tutti i suoi strumenti.

3.3 Linguaggi e strumenti utilizzati

3.3.1 SWI-Prolog e Prolog

SWI-Prolog[13] è un ambiente di sviluppo per il linguaggio di programmazione *Prolog*, *open source* e multi-piattaforma utilizzato in supporto ai fini del lavoro proposto in questa tesi. Include un *editor* di testo, un *debugger* e strumenti per la creazione di interfacce utente grafiche.
Prolog è un linguaggio di programmazione logico che si basa sulla logica del primo ordine[14]. In Prolog, i programmi vengono scritti come insiemi di fatti e regole e il calcolo procede attraverso l'unificazione di questi elementi:

- **Fatti:** affermazioni che specificano relazioni tra oggetti e predicati, descrivendo informazioni vere o false rispetto al dominio rappresentato dal programma.
- **Regole:** definiscono come dedurre nuove informazioni dai fatti esistenti e da altre regole.

Una delle caratteristiche distintive di Prolog è che il calcolo procede attraverso la ricerca di soluzioni per predicati, ovvero il linguaggio è in grado di risolvere problemi attraverso la generazione di risposte a domande che sono espresse sotto forma di predicati.

[13] `https://www.swi-prolog.org`

[14] Una logica del primo ordine è una teoria formale in cui è possibile esprimere enunciati e dedurre le loro conseguenze logiche in modo del tutto formale e meccanico.

In Prolog, un predicato è un costrutto fondamentale che rappresenta una relazione tra gli oggetti nel dominio del problema e può essere pensato come una funzione che restituisce un valore *booleano (true o false)* in base alla corrispondenza tra i suoi argomenti e i fatti/regole del programma. Tramite questo linguaggio, un predicato è definito con **clausole di Horn**, ovvero espressioni logiche che seguono una forma particolare[15]. Nello specifico, una clasuola di Horn è costituita da:

- **Testa:** rappresentata da atomo, che simboleggia un'entità indivisibile e senza struttura.
- **Corpo:** è una congiunzione di atomi o clausole di Horn.

Ad esempio, considerando il seguente predicato **"padre(X,Y)"** che esprime la relazione **"X è il padre di Y"**, può essere definito in Prolog come una clausola di Horn con la testa "padre(X,Y)" e il corpo "vuoto".

```
padre(X,Y) :- .
```

Stabiliti uno o più predicati, per renderli utili bisogna specificare le regole e i fatti che definiscono la relazione padre-figlio, ad esempio:

```
padre(mario, luigi).
padre(mario, antonio).
padre(luigi, alice).
```

Sono stati definiti tre fatti che affermano che Mario è il padre di Luigi, Antonio e Alice. Si può quindi utilizzare il predicato **"padre(X,Y)"** per eseguire delle *query* sulle relazioni tra i padri e i figli. Si può procedere, ad esempio, chiedendo a SWI-Prolog se Mario è il padre di Luigi, utilizzando la seguente sintassi:

```
 ?- consult("prova.pl"). %caricamento del file interrogato
 ?- padre(mario, luigi).
true.
```

Il programma restituisce *"true"* perchè precedentemente è stato definito il fatto che Mario è il padre di Luigi.
Si può anche domandare chi è il padre di Alice:

```
?- padre(X, alice).
X = luigi.
```

In questo caso, il programma restuisce "Luigi" come valore di X, siccome viene modellato il fatto che Luigi è padre di Alice.

[15]Un esempio può essere: $(P \vee X) \wedge \neg R$.

3.3.2 TAMELESS

Considerando la complessità dei sistemi odierni, la continua espansione di minacce, la moltitudine di componenti connessi, le differenti relazioni tra essi e il gran numero di possibili percorsi e *pattern* d'attacco; si è reso praticamente infattibile condurre un' analisi di sicurezza completa in modo manuale.
"Threat Attack ModeEL Smart System"(TAMELESS)[15] è un *tool* automatico che definisce un *threat model* e una *threat analysis* destinata a sistemi ibridi come edifici intelligenti, sistemi di controllo industriale e parchi eolici. Lo strumento è utile per supportare l'analisi di sicurezza ispirandosi alla logica del primo ordine, nello specifico, riceve in *input*:

- **specifiche del sistema da analizzare**: rappresentate da componenti e minacce.
- **relazioni**: tra i componenti stessi e tra componenti e minacce.
- **assunzioni di sicurezza**: proprietà di sicurezza iniziali note riguardo le componenti del sistema.

Una volta forniti gli elementi sopra descritti, l'utente può eseguire la *threat analysis* attraverso *query*. In questo modo il *tool* provvederà ad analizzarli automaticamente e visualizzare graficamente le possibili minacce permettendo di avere una visione completa e comprensiva dello stato di sicurezza del sistema, inclusi tutti i possibili rischi di natura umana, *cyber* e fisica, come si può osservare in Figura 3.16. Per derivare gli *output* vengono introdotte un *set* di regole che consentono allo strumento di lavorare su un qualunque sistema descritto, tenendo conto di differenti fattori. Nei paragrafi successivi verranno illustrate, nello specifico, la due parti processate da TAMELESS.

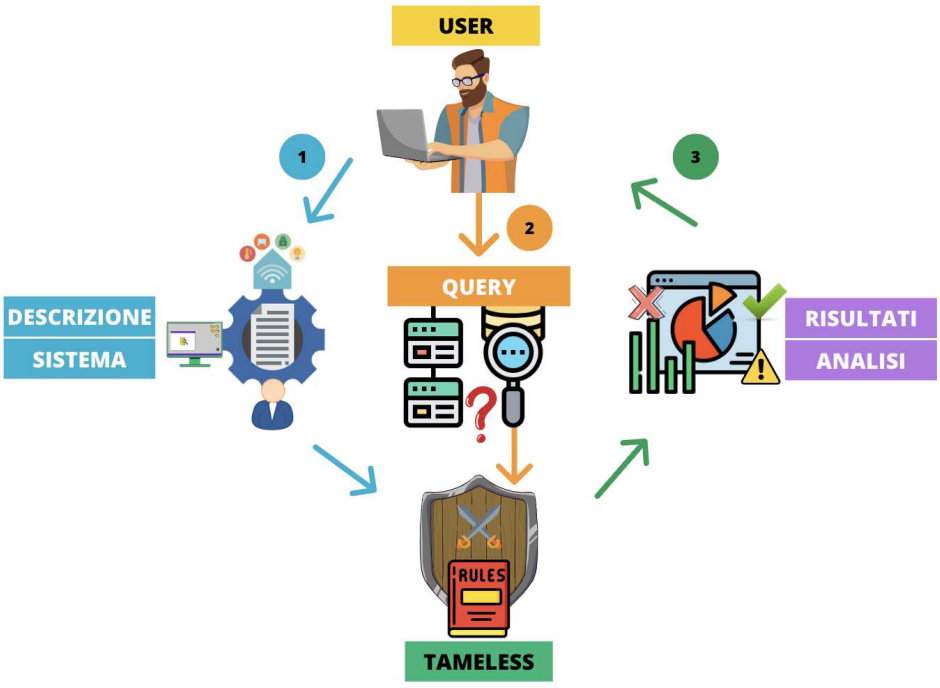

Figura 3.16: Architettura e *workflow* TAMELESS.

Hybrid threat model

Il *threat model* di TAMELESS fornisce una metodologia utile per astrarre le componenti del sistema considerato e per interpretare le possibili connessioni fornite secondo le relazioni stabilite.
Questo modello comprende:

- **Entità**: rappresentano il sistema o le componenti di un sistema, possono essere di natura umana,fisica o informatica. L'insieme delle entità sono denotate con **E**.
- **Threat**: sono una o più sequenze di azioni che cambiano direttamente o indirettamente le proprietà e che comportano una possibile mutazione dello stato di sicurezza delle entità. L'insieme delle minacce vengono etichettate con **T**.

I legami tra entità, o tra *threat* ed entità, vengono rappresentati sotto forma di relazioni e proprietà di sicurezza predefinite.

Esempi di *relazioni tra entità* sono:

- ***Contain*** *Contain(A,B)*: l'entità A contiene un'altra entità B. Ad esempio il sistema centrale contiene i dati relativi alle rilevazioni dei sensori.
- ***Control*** *Control(A,B)*: significa che A controlla B. Si può pensare ad un utente che controlla la propria *password* d'accesso al sistema.
- ***Connect*** *Connect(A,B,C)*: indica che l'entità B connette le due entità A,C. In altre parole, A può raggiungere B attraverso C. E' importante sottolineare il fatto che si tratta di una relazione unidirezionale.
  Ad esempio si può considerare la porta che collega la cucina alla *hall* di una smart home.
- ***Depend*** *Depend(A,B)*: L'entità A dipende da B, ovvero è funzionante solo se lo è B. Ad esempio un *database* dipende dal suo *server*.
- ***Check*** *Check(A,B)*: A controlla B che funzioni normalmente e quindi rileva malfunzionamenti. A differenza di *Monitor*, *Check* si riferisce alla funzionalità dei sistemi.
- ***Replicate*** *Replicate(A,B)*: A è una replica di B. La disponibilità di una replica rende possibile riparare un'entità, come ad esempio il *backup* relativo ad un *server*.

Relazioni tra entità e minacce permettono di rappresentare quale entità è vulnerabile ad una particolare minaccia, identificare quali altre entità sono vulnerabili o quali possono proteggersi a vicenda, ad esempio:

- ***Protect*** *Protect(A,B,T)*: indica che A protegge B da una possibile minaccia T. Un *firewall*, ad esempio, protegge il sistema da un traffico non autorizzato. E' necessario sottolineare il fatto che se un'entità è vulnerabile e non protetta, può essere compromessa.
- ***Monitor*** *Monitor(A,B,T)*: A monitora B da una minaccia T. Questo significa che gli attacchi possono essere rilevati ma non prevenuti. A differenza di *Check*, *Monitor* si riferisce alla sicurezza dei sistemi. E' opportuno esplicitare che se un'entità monitora un'altra entità questo non significa che viene anche protetta.
- ***Spread*** *Spread(A,T)*: descrive che A può propagare la minaccia T. Si consideri come esempio un attacco di *phishing* via *e-mail* usato per diffondere un *malware*.
- ***Potentially Vulnerable*** *PotentiallyVul(A,T)*: A è vulnerabile alla minaccia T. Tornando all'esempio precedente, un utente può essere vulnerabile ad un attacco di *phishing*.

Dopo aver definito componenti e relazioni, è necessario introdurre l'insieme di *proprietà di sicurezza*. Si distinguono in: *proprietà di base*, come:

- ***Compromised*** *Comp(A,T)*: indica che l'entità A è stata compromessa dalla minaccia T.
- ***Malfunctioned*** *Malfun(A)*: l'entità A è malfunzionante, ovvero una o più funzionalità non sono attualmente disponibili o le *performance* risultanti non sono quelle attese. Ad esempio, un *server* che dovrebbe essere attivo risulta *offline*.
- ***Vulnerable*** *Vul(A,T)*: l'entità A ha una vulnerabilità conosciuta che potrebbe essere sfruttata dalla minaccia T.

proprietà ausiliare, che descrivono lo stato di funzionamento dell'entità a cui si riferiscono, ad esempio:

- ***Detected*** *Det(A,T)*: descrive che è stato rilevato che l'entità A è stata compromessa da una minaccia T.
- ***Restored*** *Rest(A)*: indica che il controllo su A è stato ripristinato, solitamente dopo che questa viene compromessa, ma può risultare ancora malfunzionante.
- ***Fixed*** *Fix(A)*: rivela che la funzionalità di A è riparata, di solito dopo un malfunzionamento. Ad esempio, la serratura di una porta viene riparata o l'*anitivirus* viene aggiornato. La funzionalità può essere riparata ma l'entità può risultare ancora compromessa.

Dopo aver specificato proprietà e relazioni è necessario introdurre l'insieme delle *proprietà di alto livello*, elencate in Figura 3.17. Il loro compito è di rappresentare lo stato complessivo di un'entità inclusi i suoi componenti, dipendenze e connessioni. Cosi facendo l'utente è in grado di comprendere meglio lo stato di sicurezza del sistema:

- ***Valid*** *Val(A)*: un'entità A risulta valida quando non è stata compromessa e non è malfunzionante.
- ***Defended*** *Def(A,T)*: A è difesa nei confronti di una minaccia T, quando esiste un'entità B che protegge A da T e B è valida. Contrariamente A non è difesa quando non esiste una misura di protezione oppure esiste e non è valida.
- ***Safe*** *Safe(A,T)*: indica che A è sicura rispetto ad una minaccia T quando A non è vulnerabile o può essere difesa dall'attacco T. Quando non si verificano queste condizioni, A risulta essere non sicura alla minaccia T.
- ***Monitored*** *Mon(A,T)*: A è monitorata per la minaccia T, quando esiste un'entità B che monitora A rispetto T. Se non esiste o non è valida, A non è monitorata.
- ***Checked*** *Che(A)*: quando un'entità B controlla le funzionalità di A e B risulta valida, allora A è verificata. Altrimenti non lo è se non esiste B oppure non è valida.
- ***Replicated*** *Rep(A)*: A è replicato, quando esiste almeno un'entità B che replica A e B è valida. L'entità A non viene replicata quando non esiste un tale B che replichi A, oppure esiste ma non è valido.

$$
\begin{array}{ll}
\text{Val}(A) := \neg\text{Comp}(A) \wedge \neg\text{Malfun}(A) & \neg\text{Val}(A) := \text{Comp}(A) \vee \text{Malfun}(A) \\
\text{Def}(A,T) := \exists B.\ \text{Protect}(B,A,T) \wedge \text{Val}(B) & \neg\text{Def}(A,T) := \nexists B.\ \text{Protect}(B,A,T) \vee (\forall B.\text{Protect}(B,A,T) \wedge \neg\text{Val}(B)) \\
\text{Safe}(A,T) := \neg\text{Vul}(A,T) \vee \text{Def}(A,T) & \neg\text{Safe}(A,T) := \text{Vul}(A,T) \wedge \neg\text{Def}(A,T) \\
\text{Mon}(A,T) := \exists B.\ \text{Monitor}(B,A,T) \wedge \text{Val}(B) & \neg\text{Mon}(A,T) := \nexists B.\ \text{Monitor}(B,A,T) \vee (\forall B\ \text{Monitor}(B,A,T) \wedge \neg\text{Val}(B)) \\
\text{Che}(A) := \exists B.\ \text{Check}(B,A) \wedge \text{Val}(B) & \neg\text{Che}(A) := \nexists B.\ \text{Check}(B,A) \vee (\forall B\ \text{Check}(B,A) \wedge \neg\text{Val}(B)) \\
\text{Rep}(A) := \exists B.\ \text{Replicate}(B,A) \wedge \text{Val}(B) & \neg\text{Rep}(A) := \nexists B.\ \text{Replicate}(B,A) \vee (\forall B.\ \text{Replicate}(B,A) \wedge \neg\text{Val}(B))
\end{array}
$$

Figura 3.17: Proprietà di alto livello.

In conclusione, è importante sottolineare che le categorie *proprietà di base e proprietà di alto livello* possono essere interpretate in maniera diversa a seconda del contesto analizzato:

- **Assunte**: se utilizzate per effettuare delle ipotesi, si indicano con α.
- **Derivate**: se risultanti dalla procedura di derivazione, rappresentate con κ.

La Figura 3.18 riassume un esempio di *threat model* implementato in Prolog.

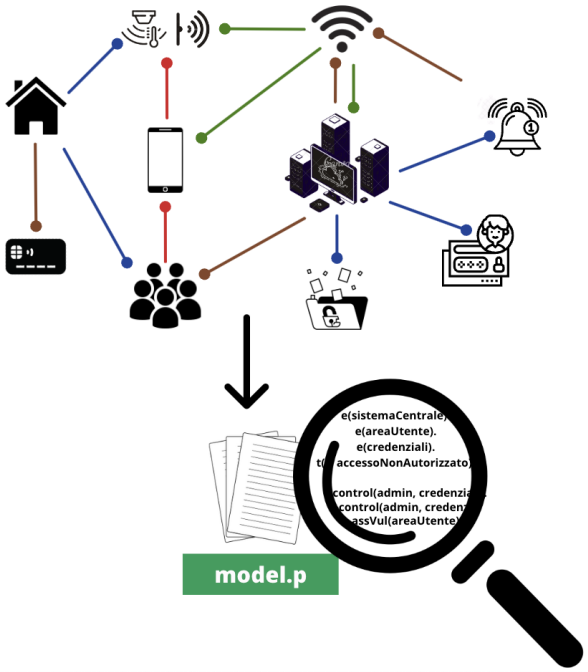

Figura 3.18: Esempio di *threat model* implementato in Prolog.

Applicazione della threat analysis

La *"threat analysis"* implementata da TAMELESS è responsabile del vero e proprio processo di analisi delle minacce. Dopo aver fornito al *tool* la descrizione del sistema secondo la sintassi e semantica richiesta, viene elaborato l'*output* applicando alcune *regole di derivazione* in base alle proprietà e relazioni note. In questo modo si stabiliscono nuove conoscenze riguardo lo stato di sicurezza del sistema preso in esame, derivando quindi quali entità possono diventare vulnerabili, compromesse e/o malfunzionanti. Per comprendere intuitivamente il lavoro proposto nel Capitolo 4, le regole vengono etichettate numericamente e si possono distinguere in:

Regole di derivazione base (Figura 3.19), attraverso le quali si può concludere che se una proprietà si assume vera allora può essere naturalmente derivata vera. Hanno l'obiettivo di individuare i potenziali rischi legati alla struttura del sistema:

$$\text{(1)}\ \alpha\text{Comp}(A,T) \rightarrow \kappa\text{Comp}(A,T)$$
$$\text{(2)}\ \alpha\text{Vul}(A,T) \rightarrow \kappa\text{Vul}(A,T)$$
$$\text{(3)}\ \alpha\text{Malfun}(A) \rightarrow \kappa\text{Malfun}(A)$$

Figura 3.19: Regole di derivazione base.

Altre regole di derivazione mettono in relazione le entità del sistema e le proprietà descritte precedentemente, esaminando lo scenario mediante una logica precisa e con l'obiettivo di analizzare i rischi e valutare l'efficacia delle eventuali contromisure esistenti. Tali regole possono essere derivate per compromissione, malfunzionamento e vulnerabilità. Tra queste, ad esempio, si può osservare in Figura 3.20 il caso in cui un'entità A è controllata dall'entità B compromessa:

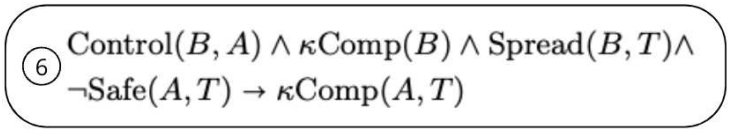

Figura 3.20: Regola di derivazione 6.

In alternativa è presente anche il caso in cui la medesima entità compromessa sia legata da una relazione di connessione con l'entità vittima dell'attacco, come mostrato in Figura 3.21:

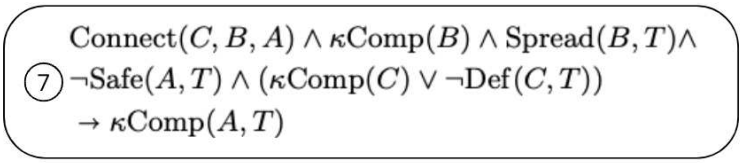

Figura 3.21: Regola di derivazione 7.

O ancora quando questa sia contenuta (o contenga) un'entità compromessa, come indicato in Figura 3.22:

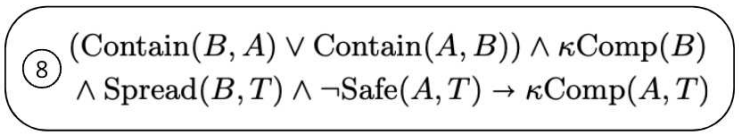

Figura 3.22: Regola di derivazione 8.

Di seguito, in Figura 3.23, si riassumono ulteriori regole di derivazione implementate da TAMELESS, che includono anche i casi di rilevamento, ripristino e riparazione:

⑨ $\kappa\text{Comp}(A, T) \rightarrow \kappa\text{Malfun}(A)$
⑩ $\text{Depend}(A, B) \ \wedge \ \kappa\text{Malfun}(B) \rightarrow \kappa\text{Malfun}(A)$
⑪ $\kappa\text{Malfun}(A) \wedge \text{Che}(A) \rightarrow \kappa\text{Fix}(A)$
⑫ $\kappa\text{Det}(A, T) \wedge \text{Rep}(A) \rightarrow \kappa\text{Rest}(A)$
⑬ $\kappa\text{Comp}(A, T) \wedge \text{Mon}(A, T) \rightarrow \kappa\text{Det}(A, T)$
⑭ $\kappa\text{Malfun}(A) \wedge \text{PotentiallyVul}(A, T) \rightarrow \kappa\text{Vul}(A, T)$

Figura 3.23: Altre regole di derivazione.

Capitolo 4

Soluzione proposta

Il lavoro di tesi si concentra sulla valutazione delle minacce utilizzando TAMELESS e sfruttando le relative tecnologie impiegate, quali *threat modeling* e *threat analysis*. Per poter comprendere a fondo le operazioni svolte e le tecniche impiegate è opportuno avere una panoramica generale dello scenario applicativo su cui l'analisi è incentrata.

4.1 Scenario applicativo

Il sistema è caratterizzato da una smart home divisa in due appartamenti (A e B) entrambi dotati di meccanismi di monitoraggio dei consumi e delle presenze all'interno delle singole abitazioni, come raffigurato in 4.1 e 4.2.

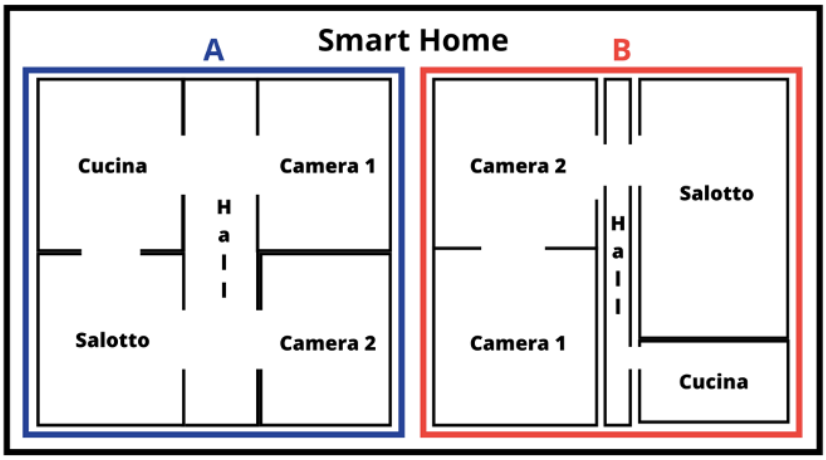

Figura 4.1: Struttura dello scenario applicativo.

Infrastruttura

Vengono utilizzati dei sensori installati su contatori elettrici o sui singoli elettrodomestici (microonde, frigorifero, lavatrice, luci stanze ecc.), in modo da monitorare i consumi energetici dell'abitazione. Per quanto riguarda il monitoraggio delle presenze si utilizzano dei sensori di movimento installati in punti strategici di ciascuna stanza (soggiorno, cucina, camere da letto). Infine, è presente un sensore che rileva l'apertura della porta d'ingresso.

I sensori analizzano, raccolgono ed elaborano le rilevazioni[1] e le inviano ad un sistema centrale di raccolta dati dove è possibile visualizzare tutte le informazioni circa lo stato della propria smart home (si osservi Figura 4.3).

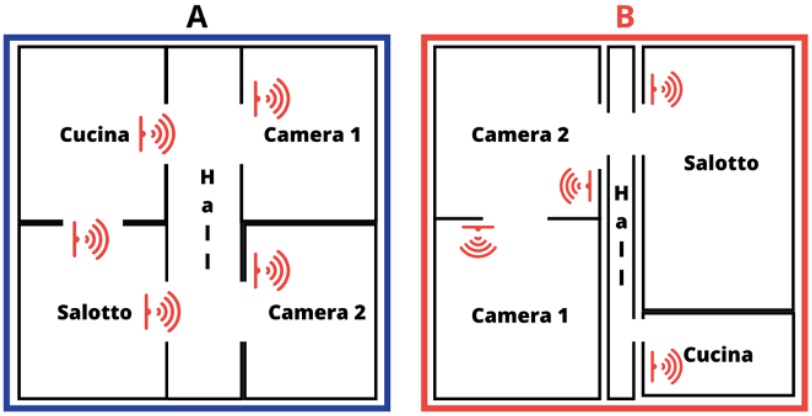

Figura 4.2: Schema dei sensori di movimento installati.

Requisiti

Per permettere al sistema di interagire e lavorare correttamente è opportuno utilizzare dei sensori programmati in modo da rilevare i dati correttamente ed una connessione Internet stabile. Questo permette il monitoraggio efficiente e la successiva visualizzazione delle relative informazioni attraverso il sistema centrale, riservate solamente agli utenti autorizzati, in possesso di credenziali d'accesso valide e suddivisi in:

- ***Admin***: accesso completo e privilegi speciali per controllare, configurare e gestire il sistema. Possono apportare modifiche alle impostazioni, installare o disinstallare *software*, gestire *account* utenti e avere autorità su molte altre funzionalità del sistema. Hanno la responsabilità di mantenere il sistema in funzione e garantire la sicurezza delle informazioni.
- ***User***: accesso limitato alle funzionalità e alle risorse, utilizzano le applicazioni e i servizi forniti dal sistema, ma hanno restrizioni sugli accessi e sui permessi per modificare o configurare le impostazioni. La responsabilità principale è utilizzare correttamente il sistema e rispettare le regole stabilite, gestendo solamente le proprie risorse e dati personali.
- ***Guest***: possiedono gli stessi requisiti e possono svolgere le medesime azioni degli utenti. La differenza è che essi possiedono una *password* che permette l'accesso ad un *account* temporaneo, il cui tempo di attività viene stabilito dall'amministratore di sistema.

[1]Dataset "ad hoc" utilizzato: https://uninsubria365-my.sharepoint.com

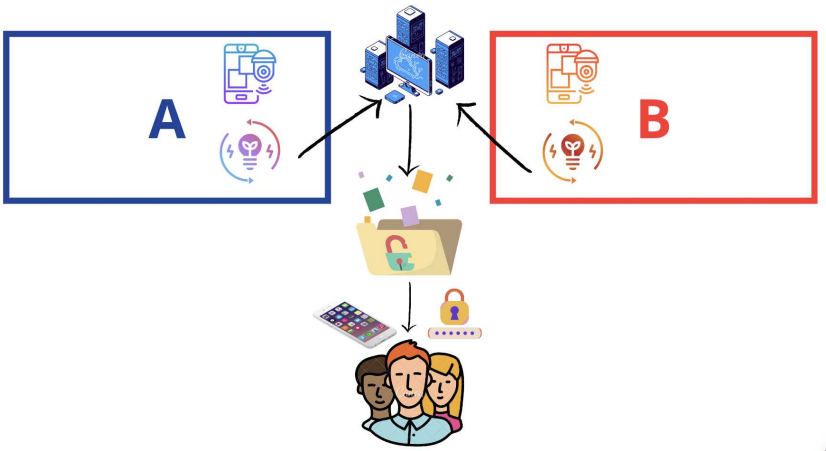

Figura 4.3: Panoramica ad alto livello sul funzionamento del sistema.

4.2 Obiettivo

Considerato lo scenario sopra descritto, è importante analizzare dove e come un attaccante può raggirare il sistema sfruttando le proprie abilità e le diverse caratteristiche delle entità coinvolte. Utilizzando TAMELESS vengono modellati gli elementi che compongono il sistema e definite le proprietà di ciascuno. Interrogando il *tool* è possibile ottenere un'analisi completa riguardo il comportamento della singola entità o dell'intero sistema in relazione alle minacce. L'obiettivo consiste nell'avere una panoramica completa circa lo stato di sicurezza dello scenario coinvolto, per poter, in una fase successiva, mitigare e prevenire attacchi da parte di figure non autorizzate.

4.3 Implementazione

Ai fini del corretto apprendimento dell'implementazione svolta è opportuno considerare la suddivisione del lavoro in due fasi parallele. Nella prima viene eseguito il *"threat modeling"* del sistema, in modo da avere una panoramica sulle entità coinvolte, le vulnerabilità assunte e le relazioni che intercorrono tra i diversi componenti e le possibili minacce note. Questi sono modellati attraverso proprietà di base e proprietà di alto livello (si consulti sezione 3.3.2). In questo modo vengono descritti una serie di scenari che potrebbero essere utilizzati da un attaccante per violare la sicurezza del sistema, aiutando a comprendere le potenziali minacce. Nella fase successiva, viene esaminato nel dettaglio il sistema: eseguendo *query*, viene fornito l'*output* derivante dalla *"threat analysis"*, il quale specifica lo stato di sicurezza determinato attraverso apposite regole di derivazione implementate da TAMELESS (si veda sezione 3.3.2). Viene focalizzata quindi l'attenzione sull'identificazione e sull'analisi dei rischi specifici associati al sistema, riconoscendo i componenti vulnerabili, compromessi e malfunzionanti mediante una descrizione di come lo stato dell'entità si è evoluto in relazione alla precedente modellazione fornita in *input*.

Phishing e accesso ai sistemi

Modellando il contesto descritto si hanno delle smart home in cui vari sensori sono connessi e controllati attraverso un sistema centrale. Il sistema permette agli amministratori di avere un pannello di controllo dedicato al funzionamento generale dei componenti e al coordinamento delle attività. Le informazioni vengono poi visualizzate dall'utente o su pagina *web* o tramite apposita applicazione, nello spazio riservato e in base ai privilegi a loro assegnati. Utenti e amministratori possono accedere ai contenuti solo se in possesso di credenziali valide in modo da evitare possibili intrusioni da parte di malintenzionati.

```
e(user).
e(admin).
e(sistemaCentrale).
e(areaUtente).
e(credenzialiAdmin).
e(credenzialiUtente).
e(twoFA).

/*gli utenti controllano le proprie credenziali,
queste sono vulnerabili ad un possibile furto.*/
control(admin, credenzialiAdmin).
spread(admin, furtoCredenziali).
assVul(credenzialiAdmin, furtoCredenziali).

control(user, credenzialiUtente).
spread(user, furtoCredenziali).
assVul(credenzialiUtente, furtoCredenziali).

/*le credenziali e altre misure di protezione difendono
il sistema da un potenziale accesso non autorizzato.*/
protect(credenzialiAdmin, sistemaCentrale, accessoNonAutorizzato)
protect(twoFA, sistemaCentrale, accessoNonAutorizzato).
contain(sistemaCentrale, credenzialiAdmin).
assVul(sistemaCentrale, accessoNonAutorizzato).

protect(credenzialiUtente, areaUtente, accessoNonAutorizzato).
contain(areaUtente, credenzialiUtente).
spread(credenzialiUtente, accessoNonAutorizzato).
assVul(areaUtente, accessoNonAutorizzato).
```

Attraverso il sistema centrale l'amministratore può gestire gli utenti connessi e i dispositivi. L'area riservata è utilizzata dall'utente per controllare i sensori, visualizzare le informazioni raccolte, (ad esempio il consumo energetico degli elettrodomestici) e monitorare eventuali accessi e/o presenze all'interno della propria abitazione. Queste attività sono coordinate grazie alla rete domestica che permette la comunicazione tra sistema centrale, dispositivi IoT e interfaccia utente. Il punto di accesso alla rete è distinto in base all'abitazione e la connessione è permessa solo se si conosce la *password*, gestita e distribuita dall'amministratore dei due appartamenti intelligenti:

```
%....
e(accessPointA).
e(accessPointB).
e(rete).
e(passwordWifi).

/*per accedere alla rete è necessario conoscere la password,
in entrambe le smart home.*/
protect(passwordWifi, accessPointA, accessoNonAutorizzato).
control(passwordWifi, accessPointA).
spread(passwordWifi, accessoNonAutorizzato).
assVul(accessPointA, accessoNonAutorizzato).
depend(rete, accessPointA).

protect(passwordWifi, accessPointB, accessoNonAutorizzato).
control(passwordWifi, accessPointB).
assVul(accessPointB, accessoNonAutorizzato).
depend(rete, accessPointB).
%....
```

Da queste proprietà è già possibile sfruttare alcune regole di derivazione per ottenere informazioni circa lo stato iniziale di sicurezza del sistema. Come si può notare esistono delle vulnerabilità in alcune entità, ma non le condizioni, per un attaccante, di trarne vantaggio. Infatti le misure di protezione messe in atto risultano effettivamente valide, ad esempio:

$\neg Safe(credenzialiUtente) = assVul(credenzialiUtente, furtoCredenziali)$
$\wedge \neg Def(credenzialiUtente, furtoCredenziali).$

$Def(areaUtente, accessoNonAutorizzato) = protect(credenzialiUtente,$
$areaUtente, accessoNonAutorizzato) \wedge Valid(credenzialiUtente).$

$\neg Safe(passwordWifi) = assVul(passwordWifi, furtoCredenziali)$
$\wedge \neg Def(passwordWifi, furtoCredenziali).$

$Def(accessPointA, accessoNonAutorizzato) = protect(passwordWifi, accessPointA,$
$accessoNonAutorizzato) \wedge Valid(passwordWifi).$

Aggiungendo però un ulteriore modifica alla precedente modellazione, la situazione cambia, creando alcuni possibili percorsi d'attacco per un'entità malintenzionata:

```
%....
t(phishing).
assComp(user, phishing).
assComp(admin, phishing).
%....
```

Assumendo che le due entità siano vittima di *phishing*, ad esempio ricevendo una *e-mail* fasulla, implica che le credenziali d'accesso associate non siano più valide e quindi non sicure per proteggere il sistema centrale, il quale può essere compromesso da un accesso non autorizzato. E' importante però distinguere i due casi: in entrambi le situazioni l'attaccante riesce ad entrare in possesso della *password*, in questo modo può accedere facilmente all'interfaccia utente, siccome l'unica misura di protezione attuata è stata raggirata.
Interrogando il sistema l'analisi restituisce:

```
?- canbeComp(areaUtente,T).
%....
T=accessoNonAutorizzato
```

L'*output* di questo predicato si deriva dalla regola 6 implementata da TAMELESS:

$Control(user, credenzialiUtente) \wedge Comp(user) \wedge Spread(user, accessoNonAutorizzato) \wedge \neg Safe(credenzialiUtente, accessoNonAutorizzato) \rightarrow Comp(credenzialiUtente, furtoCredenziali)$.

L'attacco portato a termine non risulta effettivamente efficiente, il malintenzionato ha utilizzato le proprie risorse senza un fine certo. Da descrizione precedente può risultare che l'utente vittima di *phishing* faccia parte della classe *"guest"*, e la *password* appena sottratta abbia superato il termine di validità. Se un attaccante prova a introdursi nei sistemi nel momento in cui le credenziali non sono più valide, lo scenario può essere cosi riassunto:

```
%....
e(timeValidity).
e(guestAccount).

/*viene modellata l'ipotesi di un utente
con credenziali temporanee.*/
control(credenzialiUtente, guestAccount).
assVul(guestAccount,accessoNonAutorizzato).
protect(timeValidity, guestAccount, accessoNonAutorizzato).
%....
```

Il sistema notifica l'eventuale tentativo di intrusione, ma la scandenza temporale della chiave d'accesso incide in modo positivo sulla sicurezza stessa dell'*account*, può essere quindi considerato una misura di protezione effettiva. Procedendo attraverso un'analisi più approfondita, viene restituito il seguente *output*:

```
?- canbeComp(guestAccount, accessoNonAutorizzato).
%....
false.
```

Viceversa, qualora il malintezionato riesca a sottrarre le credenziali relative ad un *account* temporaneo dove la *password* è ancora valida per l'accesso (per comprendere il concetto si assume che il tempo di validità sia compromesso), la situazione cambia, concedendo una possibile opportunità ai danni della vittima:

```
t(intenzioniMalevoli).
assComp(timeValidity, intenzioniMalevoli).

?- canbeComp(guestAccount, accessoNonAutorizzato).
%....
true.
```

La regola di derivazione implementata da TAMELESS per giungere al risultato è la seguente, dove il tempo di validità risulta essere una misura di protezione non più valida:

$Control(credenzialiUtente, guestAccount) \wedge Comp(credenzialiUtente) \wedge Spread(credenzialiUtente, accessoNonAutorizzato) \wedge \neg Safe(guestAccount, accessoNonAutorizzato)$
$\rightarrow Comp(guestAccount, accessoNonAutorizzato).$

$\neg Safe(guestAccount) = assVul(guestAccount, accessoNonAutorizzato)$
$\wedge \neg Def(guestAccount, accessoNonAutorizzato).$

$\neg Def(guestAccount, accessoNonAutorizzato) = protect(timeValidity, guestAccount, accessoNonAutorizzato) \wedge \neg Valid(timeValidity).$

E' opportuno inoltre ricordare che i contenuti visualizzabili dall'utente, sono vincolati dai privilegi che gli amministratori concendono. Può capitare quindi che l'attaccante non abbia accesso a informazioni complete utili per conseguire gli obiettivi prefissati.

Procedendo con l'analisi, studiando nello specifico il caso dove la vittima è l'amministratore del sistema, si può notare che anche se le credenziali risultano compromesse, e il sistema centrale è vulnerabile ad un accesso non autorizzato, esiste una misura di protezione aggiuntiva che evita azioni dannose da parte di malintenzionati.

L'autenticazione a due fattori (2FA) verifica l'identità di un utente prima di consentirgli l'accesso ad un *account* o a un sistema. Invece di affidarsi solo ad una *password*, la 2FA aggiunge un secondo fattore di autenticazione che rende più difficile per l'aggressore accedere all'area riservata anche se riesce ad ottenere o indovinare la *password*.

```
?- canbeComp(sistemaCentrale,accessoNonAutorizzato).
%....
false.
```

$Def(sistemaCentrale, accessoNonAutorizzato) = protect(2FA$
$sistemaCentrale, accessoNonAutorizzato) \wedge Valid(2FA)$.

Concludendo, l'amministratore controlla le credenziali d'accesso alla rete domestica, la sua compromissione tramite *phishing* comporta una compromissione anche della chiave per accedere al *wifi*, attraverso il *router*:

```
?- canbeComp(accessPointA,T).
%....
T=accessoNonAutorizzato

?- canbeComp(accessPointB,T).
%....
T=accessoNonAutorizzato
```

$Control(admin, passwordWifi) \wedge Comp(admin) \wedge Spread(admin,$
$accessoNonAutorizzato) \wedge \neg Safe(passwordWifi, accessoNonAutorizzato)$
$\rightarrow Comp(passwordWifi, furtoCredenziali)$.

Conoscendo le vulnerabilità specifiche dell'entità, si può notare come risulta ragionevole per l'attaccante entrare in possesso della *password* per accedere alla rete, che rappresenta un punto vitale per il funzionamento dell'intero sistema. Cosi facendo l'entità malintenzionata può sfruttare questa debolezza per creare ulteriori spazi d'attacco. Si vedrà però successivamente come questo non sia possibile, siccome l'accesso alla rete domestica non garantisce un attacco certo.
Riassumendo, dalle analisi svolte emerge che l'attaccante deve sfruttare differenti percorsi, attuando un comportamento non omogeneo per poter rubare informazioni sensibili che viaggiano in rete o che sono in possesso dell'amministratore; utilizzate per coordinare l'attività ed i comportamenti dei dispositivi all'interno delle smart home.

Attacco DoS alla rete

Sfruttando le vulnerabilità dell'utente e le compromissioni derivate dai precedenti attacchi, ora l'entità malintenzionata può decidere di agire su differenti componenti del sistema. Un punto molto delicato è rappresentato dalla rete domestica, utilizzata per accedere al sistema centrale e far funzionare i dispositivi interconnessi. Lo scopo dell'attaccante è quello di impedire agli utenti di accedere ad una determinata risorsa o di rallentarne il funzionamento, creando problemi di disponibilità o *performance*. Attraverso l'attacco DoS *(Denial of Service)*, utilizzando un dispositivo connesso alla rete viene inviato elevato volume di traffico o richieste falsificate sovraccaricando il sistema.
Si assume quindi che la rete sia vulnerabile a questo tipo d'attacco ma allo stesso tempo, sfruttando le proprietà di alto livello nella modellazione, esistono delle misure di protezione per non permettere all'attaccante di eseguire operazioni malevoli in rete.

```
%....
e(firewallA).
t(dos).

/*l'utente utilizza la password wifi per connettersi
alla rete, la quale risulta protetta da un firewall.*/
connect(user,passwordWifi,rete).
spread(passwordWifi,dos).
protect(firewallA,rete,dos).
/*esistono delle vulnerabilità che possono permettere
all'attaccante di compiere azioni malevoli,
se correttamente sfruttate.*/
assVul(rete,dos)
%....
```

Considerando la smart home A come possibile entità vittima, anche se l'attaccante può accedere alla rete, in questo possibile scenario non può eseguire l'attacco DoS ai sistemi. Questi risultano sicuri all'eventuale minaccia siccome protetti da un'entità attualmente valida:

$\neg Safe(rete) = assVul(rete, dos) \wedge \neg Def(rete, dos)$.

$Def(rete, dos) = protect(rete, firewallA, dos) \wedge Valid(firewallA)$.

Modificando però la precedente modellazione, aggiungendo relazioni specifiche tra le entità coinvolte e le minacce, l'entità protettrice risulta essere compromessa. L'attaccante infatti potrebbe individuare vulnerabilità o debolezze nella configurazione del *firewall* per *"baypassare"* le regole di filtraggio e ottenere accesso diretto alla rete interna, sfruttando porte o protocolli non bloccati.

Viene sfruttata quindi una caratteristica fondamentale che lega il *firewall* alla rete, infatti questo è direttamente integrato nel *router* che fornisce la connessione a Internet.

```
%....
isContained(firewallA, rete).
spread(rete, intenzioniMalevoli).
assVul(firewallA, intenzioniMalevoli).
%....
```

Interrogando il sistema si nota come questo tipo d'attacco possa compromettere l'intera rete:

```
?-canbeComp(rete,T).
%....
T=dos
```

Le informazioni circa lo stato attuale del sistema in relazione alla minaccia sono ottenute attraverso la regola di derivazione 8, che specifica le condizioni di compromissione del *firewall*, e 7, che descrive il perchè la rete può essere compromessa da DoS: questa risulta non più sicura alla minaccia, siccome protetta da un'entità precedentemente violata e quindi non più valida.
Inoltre l'utente, anch'esso compromesso, è connesso tramite una chiave, la quale è stata precedentemente rubata e utilizzata per accedere alla rete permettendo la diffusione di questo attacco:

$Contain(firewallA, rete) \land Comp(rete) \land Spread(rete, intenzioniMalevoli) \land \neg Safe(firewallA, intenzioniMalevoli) \rightarrow Comp(firewallA, intenzioniMalevoli).$

$Connect(user, passwordWifi, rete) \land Comp(rete) \land Spread(rete, intenzioniMalevoli) \land \neg Safe(rete, dos) \rightarrow Comp(rete, dos).$

In aggiunta, è possibile utilizzare l'analisi per stabilire se l'entità è malfunzionante. L'*output* restituisce un valore positivo: secondo TAMELESS, la regola 9 deriva che la compromissione della rete da parte di una minaccia ne causa un possibile malfunzionamento.

```
?-canbeMalfun4Comp(rete).
%....
true.
```

$Comp(rete, dos) \rightarrow Malfun(rete).$

E' importante osservare che generalmente gli attacchi DoS vengono eseguiti e possono essere estremamente dannosi quando indirizzati verso scenari caratterizzati da molteplici dispositivi. Si consideri, come esempio, un contesto aziedale, dove i diversi dispositivi, collocati in reparti differenti, in base agli obiettivi, sono connessi tra loro in modo centralizzato. Questo tipo di attacco può essere dannoso per diverse ragioni: possono causare l'interruzione delle operazioni aziendali, la perdita della produttività e di reputazione, il possibile furto di dati ed un impatto negativo sulla continuità aziendale.
Contrariamente, in uno scenario smart home, con un numero di dispositivi limitati, a livello prestazionale, attuare un attacco DoS comporta un maggiore dispendio in termini di risorse, che può comportare una minimizzazione dell'efficacia ma non dei danni arrecati. Considerando questo possibile scenario d'attacco però, indipendentemente dal numero di dispositivi colpiti, l'entità malintenzionata può creare differenti percorsi sfruttando questa minaccia, in base agli obiettivi prefissati. Ad esempio, utilizzando il DoS per saturare i sensori di movimento, causandone uno spegnimento, l'attaccante può successivamente entrare nell'abitazione senza che l'utente venga allertato.
In definitiva la gravità degli attacchi DoS dipende principalmente dalla capacità di mitigazione di qualsiasi organizzazione, dalla tempestività nella risposta e dalle contromisure di sicurezza implementate.

Attacco Shadow Server

Dagli attacchi descritti precedentemente si può notare che sicuramente l'entità malintenzionata riesce ad eseguire un accesso non autorizzato nell'apposita area riservata all'utente, (analogamente anche nel caso in cui l'utente sia *"guest"* e la *password* non sia scaduta). Le informazioni più complete, sensibili e di controllo relative a tutto l'ecosistema sono in possesso solo dell'amministratore. L'analisi ora si concentra su una possibile soluzione d'attacco utile per eludere le misure di protezione attuate per garantire la sicurezza di un entità fondamentale. *"Shadow Server"* è la definizione di tecniche che coinvolgono attacchi quali *DoS* e *ipSpoofing*. Ampliando la modellazione dello scenario inizialmente descritto, l'analisi di minaccia permette di individuare un ulteriore spazio d'attacco coinvolgendo altre entità. Ai fini del corretto funzionamento del sistema, è opportuno inserire un *server*, il quale fornisce l'infrastruttura necessaria per ospitare il sistema e per la distribuzione dei contenuti.
Il primo *step* di questo attacco ha come obiettivo principale quello di indurre l'utente ad inserire l'autenticazione a due fattori su un *server* creato appositamente dal malintenzionato, in modo da ottenere le credenziali necessarie per poter entrare nel sistema.

Eseguendo un ulteriore modellazione dello scenario si ottiene:

```
%....
e(server).

/*il server ospita il sistema centrale ed è responsabile
dell'indirizzamento verso l'autenticazione a due fattori.*/
connect(admin, server, twoFA).
spread(server, intenzioniMalevoli).
assVul(twoFA, intenzioniMalevoli).
%....
```

Eseguendo l'analisi sulla precedente modellazione si può notare come l'attaccante sia riuscito a raggirare la misura di protezione aggiuntiva:

```
?-canbeComp(twoFA,intenzioniMalevoli).
%....
true.
```

Per permettere ciò è opportuno considerare il codice sottostante che descrive le relazioni che legano le entità coinvolte. Si osserva che esistono delle vulnerabilità note e ulteriori proprietà che negano all'attaccante la possibilità di penetrare inizialmente nel sistema.

```
%....
/*la rete permette il collegamento del
sistema centrale al server.*/
connect(sistemaCentrale, accessPointA, server).
depend(sistemaCentrale, server).
protect(firewall, server, dos).
spread(accessPointA, dos).
assVul(server, dos).
%....
```

Dalle precedenti analisi si può notare come l'attaccante può eludere la misura di protezione inserita; compromettendo il *firewall* infatti il *server* non risulta più sicuro, di conseguenza può essere sfruttata la vulnerabilità per compiere azioni malevoli ai danni della risorsa critica.
Tramite l'attacco DoS, l'aggressore mira ad inviare un'elevata quantità di pacchetti al *server* principale, in modo da interromperne il normale funzionamento, questo è possibile connettendosi alla rete della smart home A:

```
?-canbeComp(server,T).
%....
T=dos
```

TAMELESS valuta la possibile minaccia facendo riferimento alla regola 7 di derivazione:

$Connect(sistemaCentrale, accessPointA, server) \wedge Comp(accessPointA) \wedge Spread(accessPointA, dos) \wedge \neg Safe(server, dos) \rightarrow Comp(server, dos)$.

In questo modo l'utente è impossibilitato ad accedere alle proprie informazioni poichè la compromissione del *server* implica il suo malfunzionamento. Siccome il sistema centrale dipende da questa entità malfunzionante, anche quest'ultimo risulta inoperativo, come stabiliscono le regole di derivazione 9 e 10:

$Comp(server, dos) \rightarrow Malfun(server)$.

$Depend(sistemaCentrale, server) \wedge Malfun(server) \rightarrow Malfun(sistemaCentrale)$.

Così facendo, l'entità malintenzionata può sfruttare le precedenti vulnerabilità, umane e digitali, per "mascherare" un *server* falso, tramite la tecnica chiamata *"ipSpoofing"*, e indurre l'utente ad accedere a questo.
Più nel dettaglio,attraverso questa tecnica un aggressore potrebbe inviare un pacchetto di rete con un indirizzo ip *"spoffato"* per fingere di essere una fonte attendibile o un sistema di fiducia, al fine di ottenere un accesso non autorizzato o informazioni prima inaccessibili. Inoltre questo tipo di tecnica può essere utilizzata per nascondere l'identità dell'individuo con scopi dannosi e rendere difficile l'individuazione dell'origine dell'attacco.

```
%....
e(attaccante).

/*l'attaccante è riuscito ad ottenere
il controllo del server.*/
control(attaccante,server).
spread(attaccante,ipSpoofing).
assVul(server,ipSpoofing).
assComp(attaccante,intenzioniMalevoli).
%....
```

Si ipotizza ora che l'attaccante abbia il controllo di un nuovo *server*, che per l'utente risulta quello originale. Sfruttando quindi un percorso differente il comportamento del sistema evolve:

```
?-canbeComp(server,ipSpoofing).
%....
true.
```

$Control(attaccante, server) \wedge Comp(attaccante) \wedge Spread(attaccante, ipSpoofing) \wedge \neg Safe(server, ipSpoofing) \rightarrow Comp(server, ipSpoofing)$.

Si osserva come nelle precedenti analisi, l'aggressore ha sfruttato un'entità specifica come la rete Internet per creare differenti percorsi d'attacco ai danni delle componenti del sistema. Attraverso una vulnerabilità di natura umana, l'attaccante oltre ad ottenere l'accesso alla rete domestica, si è impossessato delle credenziali per entrare nel sistema centrale. Si ricorda che attraverso quest'importante entità, l'amministratore ha il controllo dell'intero sistema e delle informazioni sensibili riguardo le smart home.
Di seguito verranno analizzati nel dettaglio alcuni attacchi che si possono compiere nei confronti del sistema o sfruttare queste vulnerabilità per infliggere ulteriori danni sia di natura fisica che digitale alle componenti interconnesse.

Attacco Man in The Middle

Entrando più nello specifico, il compito principale di un *firewall* è quello di proteggere una rete controllando il flusso del traffico. Esso agisce come una barriera tra una rete privata e una pubblica o non fidata, come ad esempio Internet. Oltre al filtraggio del traffico, esaminando indirizzi IP, protocolli e porte di comunicazione, svolge un controllo degli accessi stabilendo politiche per limitare chi può connettersi ad una rete. Un ulteriore caratteristica rilevante del *firewall* implementato in questo scenario rappresenta l'abilità di identificare e bloccare tentativi di intrusioni o attacchi informatici.
Tuttavia, come analizzato in precedenza, l'insieme delle tecniche utilizzate dall'attaccante e alcune vulnerabilità note ne hanno comportato una sua compromissione.

```
%....
e(mqttConnectionA).

/*la comunicazione dei dispositivi
è controllata dal firewall.*/
control(firewallA,mqttConnectionA).
spread(firewallA,manInTheMiddle).
/*esistono delle vulnerabilità
che possono permettere all'attaccante
di ascoltare la comunicazione.*/
assVul(mqttConnectionA,manInTheMiddle).
%....
```

Analizzando le scelte architetturali utilizzate, questo rappresenta l'unica misura utilizzata per difendere la rete domestica della smart home A.
Considerando questa abitazione e le relative proprietà di sicurezza, si può notare come si verificano le condizioni ideali per un attaccante di accedere direttamente alle risorse interessate, siccome esiste una misura di protezione momentaneamente non più valida a difesa della rete.

Analizzato il possibile attacco, TAMELESS restituisce il seguente *output*:

```
?-canbeComp(mqttConnectionA,manInTheMiddle).
%....
true.
```

La risposta dell'analisi di minaccia, in relazione ai legami tra le entità coinvolte, viene generata attraverso le seguenti regole di derivazione:

$Control(firewallA, mqttConnectionA) \wedge Comp(firewallA) \wedge Spread(firewallA, manInTheMiddle) \wedge \neg Safe(mqttConnectionA, manInTheMiddle)$
$\rightarrow Comp(mqttConnectionA, manInTheMiddle)$.

Si consideri ora la smart home B: come riportato inizialmente è opportuno sottolineare che non sempre l'attacco analizzato possa rappresentare un possibile percorso che l'attaccante utilizza per causare il malfunzionamento dell'entità colpita. Si può notare come viene integrata una misura di sicurezza aggiuntiva in questa abitazione: IDS, acronimo di *Intrusion Detection System*, è un sistema progettato per identificare e rispondere agli eventi di intrusione o agli attacchi informatici all'interno di un sistema, monitorando costantemente il traffico di rete. Quando un'attività sospetta viene identificata, l'IDS genera un avviso o una notifica per avvertire gli amministratori della rete.

```
e(ids).

protect(ids, mqttConnectionB, manInTheMiddle).
```

Ripetendo l'analisi, si osserva che il sistema risulta protetto, siccome il malintenzionato riesce a raggirare il *firewall*, (con l'utilizzo delle medesime tecniche) ma non l'IDS. Deve quindi necessariamente sfruttare un percorso o un insieme d'attacchi differenti che siano in grado di soddisfare gli obiettivi prefissati.

```
?-canbeComp(mqttConnectionB,manInTheMiddle).
%....
false.
```

Di seguito le proprietà di alto livello che hanno determinato il risultato conseguito:

$Safe(mqttConnectionB, manInTheMiddle) = Def(mqttConnectionB, manInTheMiddle)$.

$Def(mqttConnectionB, manInTheMiddle) = \exists ids.Protect(ids, mqttConnectionB, manInTheMiddle) \wedge Val(ids)$.

$Val(ids) = \neg Comp(ids) \wedge \neg Malfun(ids)$.

Accesso fisico non autorizzato all'interno della smart home

Uno degli scopi dell'attaccante è quello di poter entrare all'interno dell'abitazione. Il compito dei sensori di rilevazione è monitorare eventuali presenze non autorizzate all'interno della struttura. Qualora accadesse il contrario, l'utente verrebbe immediatamente avvisato della presenza mediante notifica sul proprio dispositivo connesso. L'accesso all'abitazione avviene solamente tramite la porta d'ingresso la quale è dotata di un ulteriore sensore che ne rileva l'apertura. Esistono però alcune vulnerabilità note che possono rappresentare un vantaggio, se correttamente sfruttate, per l'aggressore.
Si consideri inzialmente lo scenario che coinvolge la smart home B, siccome sarà utile per una successiva analisi d'attacco.

```
%....
e(sensoriRilevazioniPresenzeB).
e(smartHomeB).
e(portaIngresso).

t(accessoFisicoNonAutorizzato).
t(furto).

/* i sensori sono installati per monitorare
le presenze all'interno della smart home.*/
control(sensoriRilevazioniPresenzeB,portaIngresso).
spread(sensoriRilevazioniPresenzeB,accessoFisicoNonAutorizzato).
/*esistono delle vulnerabilità
e delle relazioni che l'attaccante
può sfruttare per accedere all'abitazione.*/
assVul(portaIngresso,accessoFisicoNonAutorizzato).
protect(portaIngresso,smartHomeB,furto).
contain(smartHomeB,portaIngresso).
spread(portaIngresso,furto).
assVul(smartHomeB,furto).
%....
```

Studiando queste proprietà, oltre ai sensori utilizzati, si può notare come esistono delle misure di sicurezza fisiche per evitare che l'attaccante approfitti delle fragilità note del sistema, rendendolo così sicuro alle minacce:

$Def(smartHomeB, furto) = \exists portaIngresso.Protect(portaIngresso, smartHomeB, furto) \wedge Val(portaIngresso)$.

$Val(portaIngresso) = \neg Comp(portaIngresso) \wedge \neg Malfun(portaIngresso)$.

Tuttavia, dalle precedenti analisi si descrive come il sistema centrale può essere vittima di accesso non autorizzato, cosi facendo l'attaccante può compiere azioni malevoli ai sensori installati nell'abitazione, come ad esempio spegnere i sensori di rilevazione per provare ad accedere successivamente all'abitazione.

```
%....
t(intenzioniMalevoli).

/*il sistema centrale controlla i sensori.*/
control(sistemaCentrale,sensoriRilevazioniPresenzeB).
depend(sensoriRilevazioniPresenzeB,sistemaCentrale).
spread(sistemaCentrale,intenzioniMalevoli).
assVul(sensoriRilevazioniPresenzeB,intenzioniMalevoli).
%....
```

Modellando così il sistema si nota una vulnerabilità specifica dei sensori, la quale può essere sfruttata da una minaccia per compiere un determinato attacco, si deduce inoltre che non esistono protezioni per evitare direttamente il malfunzionamento dei sensori.
In termini di proprietà di alto livello si ottiene:

$\neg Def(sensoriRilevazioniPresenzeB, intenzioniMalevoli) = \nexists B.Protect(B, sensoriRilevazioniPresenzeB, intenzioniMalevoli)$.

Cosi facendo, i sensori di rilevazioni possono essere compromessi attraverso il sistema centrale, poichè questo controlla l'entità scopo dell'attacco e precedentemente compromesso. Secondo la regola 6 si deriva:

```
?- canbeComp(sensoriRilevazioniPresenzeB,T).
%....
T=intenzioniMalevoli
```

$Control(sistemaCentrale, sensoriRilevazioniPresenzeB) \wedge Comp(sistemaCentrale) \wedge Spread(sistemaCentrale, intenzioniMalevoli) \wedge \neg Safe(sensoriRilevazioniPresenzeB, intenzioniMalevoli) \rightarrow Comp(sensoriRilevazioniPresenzeB, intenzioniMalevoli)$.

Inoltre, i sensori sono legati al sistema centrale da una relazione di dipendenza, è possibile quindi causare un malfunzionamento di questi sfruttando il legame tra le due entità e la precedente compromissione del sistema centrale.
Il fatto viene stabilito dalla regola 9 di derivazione:

```
?- canbeMalfun4Comp(sensoriRilevazioniPresenzeB).
%....
true.
```

$Depend(sensoriRilevazioniPresenzeB, sistemaCentrale) \wedge Malfun(sistemaCentrale) \rightarrow Malfun(sensoriRilevazioniPresenzeB)$.

Sfruttando le vulnerabilità e le caratteristiche del sistema l'aggressore è riuscito a creare un possibile percorso d'attacco per introdursi all'interno dell'abitazione senza che l'utente venga avvisato della presenza. Questo è reso possibile poichè la porta d'ingresso che proteggeva la smart home da un possibile furto non risulta più sicura e quindi può essere compromessa; siccome l'entità che controllava quest'ultima, ovvero il sensore d'apertura è stato compromesso attraverso il sistema centrale.
Interrogando il sistema l'analisi restituisce *"true"*. L'*output* è definito attraverso la regola 8 di derivazione:

```
?- canbeComp(smartHomeB,T).
%....
T=furto
```

$Contain(smartHomeB, portaIngresso) \land Comp(portaIngresso) \land Spread($
$portaIngresso, furto) \land \neg Safe(smartHomeB, furto)$
$\rightarrow Comp(smartHomeB, furto).$

E' opportuno introdurre un importante osservazione: precedentemente si sottolineava come l'attacco DoS possa essere utilizzato per sovraccaricare i sensori di rilevazione delle presenze al fine di ostacolare il loro funzionamento. E' noto inoltre, dalle precedenti analisi, che questo non può essere realizzato considerando la smart home B, siccome esiste una misura di protezione che impedisce a malintenzionati di manipolare la rete domestica (IDS).
Supponendo che l'attaccante non riesca ad accedere al sistema centrale gestito dall'amministratore, può utilizzare le proprie risorse e capacità per introdursi all'interno della smart home A sfruttando le vulnerabilità della rete e l'attacco DoS (applicabile nel contesto della smart home A).
La precedente modellazione viene quindi arricchita con la seguente:

```
%....
/* la rete permette il funzionamento
dei sensori e lo scambio di comandi con
l'utente connesso.*/
connect(user,rete,sensoriRilevazioniPresenzeA).
spread(rete,dos).
assVul(sensoriRilevazioniPresenzeA,dos).
%....
```

Dal punto di vista architetturale, le due smart home risultano definite nel medesimo modo, quindi la compromissione dei sensori di rilevazione delle presenze permettono all'attaccante di accedere fisicamente all'abitazione.
TAMELESS analizza il caso e restituisce l'*output* seguente con le stesse regole di derivazione utilizzate per la smart home B.

```
?- canbeComp(smartHomeA,T).
%....
T=furto
```

In conclusione, analizzando questo scenario, si può notare come l'attaccante deve utilizzare obbligatoriamente due percorsi di attacco ben strutturati e in particolar modo differenti tra loro; anche se l'obiettivo iniziale è comune per entrambe le entità coinvolte.

Packet Sniffing

Considerando le molteplici iterazioni e connessioni dirette ed indirette tra le componenti all'interno del sistema descritto e i differenti percorsi d'attacco possibili sfruttando vulnerabilità fisiche, digitali e umane, un danno aggiuntivo che può essere arrecato al sistema riguarda i dati.
Attraverso l'attacco *Man In The Middle*, l'aggressore ha approfittato di alcune vulnerabilità note per poter intercettare le comunicazioni; il passo successivo consiste nel sottrarre concretamente i dati che viaggiano attraverso il canale. Si ricorda inoltre che l'analisi si focalizza sul possibile furto di dati riguardo solamente i consumi della smart home A, per poter accedere interamente alle informazioni sulla smart home B, verrà analizzata in seguito l'eventuale possibilità, sfruttando un percorso di attacco differente. All'interno del sistema i dati rappresentano informazioni specifiche che vengono archiviati, elaborati e trasmessi tra i componenti; inoltre svolgono un ruolo fondamentale nel prendere decisioni e fornire dei risultati agli utenti. Lo scambio di informazioni tra componenti, interfacce di visualizzazione e utenti è regolato da uno standard specifico utilizzato dalle tecnologie abilitanti all'Internet of Things (si veda sezione 3.1.3). Ai fini della corretta analisi, è bene differenziare la rappresentazione dei dati: tutte le informazioni che viaggiano dai sensori al sistema centrale vengono identificate come *"flusso dati"*, che comprendono sia informazioni di controllo che dati effettivi. Successivamente, una volta che i dati elaborati giungono all'interfaccia utente, in base agli algoritmi utilizzati e all'applicazione di filtri idonei secondo i privilegi concessi all'utente finale, vengono archiviati e prendono il nome di *"datiConsumoEnergetico"* o *"datiRilevazioniPresenze"*.
Nello specifico, l'analisi è incentrata sul capire se e come l'attaccante, sfruttando le vulnerabilità descritte e due percorsi d'attacco differenti, può ottenere le informazioni circa la stato energetico della smart home interessata.
Entità, relazioni e vulnerabilità considerate vengono così modellate:

```
%....
e(flussoDatiA).
e(datiConsumoEnergetico).
e(sensoriConsumoEnergetico).
e(symmetricEncryptionKey).

t(furtoDati).
t(packetSniffing).
```

```
/*il sistema centrale si occupa di ricevere
i dati rilevati.*/
control(sistemaCentrale,datiConsumoEnergetico).
depend(sensoriConsumoEnergetico,sistemaCentrale).
spread(sistemaCentrale,furtoDati).
/*i dati ricevuti possono essere rubati.*/
assVul(datiConsumoEnergetico,furtoDati).

/*le informazioni viaggiano in rete con le regole
stabilite dal protocollo MQTT.*/
connect(sensoriConsumoEnergetico,mqttConnectionA, flussoDatiA).
depend(datiConsumoEnergetico,flussoDatiA).
spread(mqttConnectionA,packetSniffing).
/*le informazioni risultano essere cifrate secondo
uno standard concordato.*/
protect(symmetricEncryptionKey,flussoDatiA,packetSniffing).
/*è utile considerare il fatto che i pacchetti
possono essere sottratti dall'attaccante.*/
assVul(flussoDatiA,packetSniffing).
%....
```

Osservando il primo caso è gia possibile notare che non esistono misure di protezione o filtraggio per impedire a chiunque abbia accesso al sistema centrale di visualizzare liberamente i contenuti:

$\neg Safe(datiConsumoEnergetico, furtoDati) = Vul(datiConsumoEnergetico, furtoDati) \wedge \neg Def(datiConsumoEnergetico, furtoDati)$.

$\neg Def(datiConsumoEnergetico, furtoDati) = \nexists B.Protect(B, datiConsumoEnergetico, furtoDati)$.

Interrogando il sistema, in base alle proprietà descritte e dedotte precedentemente, si osserva come esistano le totali condizioni per un attaccante di trarre vantaggio dalle vulnerabilità delle componenti. Infatti, secondo le regola 6 di derivazione si ottiene che i dati archiviati non risultano sicuri, e quindi esposti a questa minaccia. Di conseguenza l'attaccante può utilizzare il sistema centrale, precedentemente compromesso, per ottenere informazioni e monitorare i diversi consumi degli elettrodomestici presenti all'interno delle due abitazioni.

```
?-canbeComp(datiConsumoEnergetico,T).
%....
T=furtoDati
```

$Control(sistemaCentrale, datiConsumoEnergetico) \wedge Comp(sistemaCentrale) \wedge Spread(sistemaCentrale, furtoDati) \wedge \neg Safe(datiConsumoEnergetico, furtoDati) \rightarrow Comp(datiConsumoEnergetico, furtoDati)$.

Nel secondo caso, viene analizzato il vero e proprio attacco di *"packet sniffing"*, sfruttando l'infrastruttura di rete che permette la comunicazione.
Con *"packet sniffing"* si intende una tecnica mirata ad intercettare ed analizzare il traffico di rete. L'attacco specifico consiste nel catturare e ispezionare i pacchetti di dati che transitano su una rete, al fine di monitorare e analizzare il loro contenuto. A differenza del precedente caso, oltre ad accedere alle informazioni vere e proprie relative al consumo energetico, l'aggressore può estrarre dettagli come indirizzi IP, protocolli utilizzati e altre caratteristiche relative alla comunicazione di rete. Dalle precedenti analisi è emerso che i controlli di protezione di rete possono essere elusi e un aggressore tramite l'attacco *"Man In The Middle"* potrebbe ascoltare la comunicazione tra componenti situati nella smart home A. Attraverso le tecniche di crittografia è possibile proteggere i dati riservati e garantire la confidenzialità delle informazioni durante la comunicazione. Se l'attaccante dovesse intercettare i pacchetti crittografati, per accedere al loro contenuto avrebbe bisogno di conoscere la chiave utilizzata per cifrare i messaggi scambiati. Di conseguenza, senza la chiave corretta, i dati risultano sicuri siccome inaccessibili e incomprensibili per l'aggressore:

$$Safe(flussoDatiA, packetSniffing) = Def(flussoDatiA, packetSniffing).$$

$$Def(flussoDatiA, packetSniffing) = \exists symmetricEncryptionKey.Protect(symmetricEncryptionKey, flussoDatiA, packetSniffing) \wedge Val(symmetricEncryptionKey).$$

$$Val(symmetricEncryptionKey) = \neg Comp(symmetricEncryptionKey) \wedge \neg Malfun(symmetricEncryptionKey).$$

E' importante notare che la tecnica di crittografia simmetrica coinvolge l'utilizzo di una singola chiave segreta per cifrare e decifrare i dati.
Questa caratteristica può indurre l'attaccante a utilizzare tecniche d'attacco mirate ad ottenere la chiave. Ad esempio, cercando tutte le possibili combinazioni fino a trovare quella corretta. Aggiungendo un ulteriore modifica alla modellazione precedente e assumendo che l'entità utilizzata per crittografare i dati non risulti sufficientemente lunga e complessa e quindi facilmente reperibile in un tempo breve con un attacco a forza bruta, lo stato di sicurezza del sistema si evolve:

```
%....
t(bruteForceAttack).
assComp(symmetricEncryptionKey,bruteForceAttack).
%....
```

Questa situazione comporta un nuovo possibile percorso d'attacco: attraverso le caratteristiche definite e assumendo che il flusso dei dati sia vulnerabile ad un possibile monitoraggio attraverso la rete, si deriva che l'entità può essere compromessa.

L'attaccante intercetta i pacchetti che transitano in rete e utilizza la chiave appena sottratta per ottenere il testo in chiaro desiderato:

```
?-canbeComp(flussoDatiA,T).
%....
T=packetSniffing
```

$Connect(sensoriConsumoEnergetico, mqttConnectionA, flussoDatiA) \wedge Comp($
$mqttConnectionA) \wedge Spread(mqttConnectionA, packetSniffing) \wedge \neg Safe($
$flussoDatiA, packetSniffing) \wedge Comp(sensoriConsumoEnergetico)$
$\rightarrow Comp(flussoDatiA, packetSniffing).$

Considerando questo rilevante *output* restituito dall'analisi di minaccia (regola 7), si deriva che i componenti compromessi sono anche malfunzionanti. Infatti secondo ulteriori regole di derivazione si ottiene:

```
?-canbeMalfun4Comp(datiConsumoEnergetico).
%....
true.
```

$Comp(datiConsumoEnergetico, furtoDati) \rightarrow Malfun(datiConsumoEnergetico).$

```
?-canbeMalfun4Comp(flussoDatiA).
%....
true.
```

$Comp(flussoDatiA, packetSniffing) \rightarrow Malfun(flussoDatiA).$

E' importante osservare, considerando le precedenti proprietà introdotte nel sistema, che esiste una relazione di dipendenza tra:

- sistema centrale - sensori consumo energetico;
- flusso dati - dati consumo energetico.

Lo stato di sicurezza delle componenti del sistema può essere condizionato in relazione al fattore determinante: l'attaccante infatti può sfruttare la dipendenza per manomettere direttamente i sensori, senza sfruttare le vulnerabilità note ma "semplicemente" compromettendo il sistema centrale, ma ciò implica un malfuzionamento fisico di questi e non garantisce che l'aggressore riesca ad avere un quadro completo delle informazioni raccolte.
Inoltre la compromissione del flusso dati attraverso l'intercettazione della comunicazione sul canale di trasmissione dei dispositivi comporta un malfunzionamento anche dei dati archiviati dal sistema centrale, violando così importanti proprietà di sicurezza.

Di conseguenza la relazione introdotta tra i componenti comporta che lo stato di sicurezza dei sensori e dei dati archiviati sono influenzati da possibili azioni arrecate ai danni del sistema centrale e al flusso dati.

```
?-canbeMalfun4Dep(sensoriConsumoEnergetico).
%....
true.
```

$Depend(sensoriConsumoEnergetico, sistemaCentrale) \wedge Malfun(sistemaCentrale) \rightarrow Malfun(sensoriConsumoEnergetico)$.

```
?-canbeMalfun4Dep(sensoriConsumoEnergetico).
%....
true.
```

$Depend(datiConsumoEnergetico, flussoDatiA) \wedge Malfun(flussoDatiA) \rightarrow Malfun(datiConsumoEnergetico)$.

Si ipotizzi ora che lo scopo dell'attaccante sia quello di intercettare tutte le informazioni riguardo i consumi energetici e/o monitorare le presenze nella smart home B. Come sottolineato molteplici volte, l'entità protagonista dell'attacco non è riuscita in alcun modo a "penetrare" nella rete domestica, per via delle stringenti misure di sicurezza implementate. Ai fini dell'intera analisi al sistema è opportuno studiare ulteriori possibili percorsi che l'attaccante può scoprire con le proprie capacità o approffittando di alcune vulnerabilità delle componenti. Per garantire il funzionamento dei sensori vengono utilizzati dei programmi o *software embedded* all'interno di essi, che prendono in nome di *firmware*. Nello specifico, il *firmware*, è un tipo di *software* appositamente realizzato per controllare e gestire il funzionamento del sensore stesso. Ad esempio è responsabile di raccolta ed elaborazione dati, calibrazione per garantire la precisione e l'affidabilità delle misurazioni e comunicazione con altri dispositivi e sistemi. Modellando lo scenario appena descritto si ottiene:

```
%....
e(firmware).
t(intenzioneMalevoli).

/*si modella il firmware installato
nei sensori delle smart home.*/
control(firmware, flussoDatiB).
spread(firmware, packetSniffing).
/*esistono alcune vulnerabilità che possono
permettere all'attaccante di infliggere danni al sistema.*/
assVul(flussoDatiB, packetSniffing).
assVul(firmware, intenzioniMalevoli).
depend(sensoriConsumoEnergetico, firmware).
depend(sensoriRilevazioniPresenzeB, firmware).
%....
```

Inoltre, analizzando le caratteristiche dei sensori di rilevazione delle presenze e del *firmware* emerge che è possibile stabilire una connessione *"wired"* (es. usb), tra il dispositivo remoto (pc, *smarthome* ecc..) ed il dispositivo di monitoraggio, in modo da poter visualizzare le caratteristiche ed accedere al flusso informativo. Da queste proprietà iniziali, un'analisi preliminare al sistema restiusce un riscontro positivo, infatti anche conoscendo le vulnerabilità delle componenti, l'attaccante non riesce a trarne vantaggio. Attualmente il *firmware* che controlla la comunicazione tra i dispositivi e il sistema centrale non risulta compromesso:

```
?-canbeComp(firmware, intenzioneMalevoli).
%....
false.

?-canbeComp(flussoDatiB, packetSniffing).
%....
false.
```

Di conseguenza è necessario che il malintenzionato sia collocato ad una distanza ravvicinata in modo da poter stabilire il collegamento. Per fare ciò è quindi fondamentale accedere fisicamente all'interno della smart home B. Assumendo quindi che l'attaccante sia riuscito a penetrare nell'abitazione eludendo i controlli, e riesca a modificare il codice relativo al *firmware*, questo risulterà essere compromesso, favorendo il controllo della comunicazione da parte del malintenzionato.

```
%....
assComp(firmware,intenzioniMalevoli).
%....
```

Aggiungendo la precedente modifica alla modellazione dello scenario, è possibile individuare un nuovo percorso d'attacco che l'aggressore può sfruttare per infliggere danni al sistema, in particolar modo alle informazioni raccolte ed elaborate dai sensori. Eseguendo la medesima analisi, ora il risultato atteso risulta essere diverso:

```
?-canbeComp(firmware, intenzioneMalevoli).
%....
true.

?-canbeComp(flussoDatiB, packetSniffing).
%....
true.
```

Questa possibile situazione viene derivata dalla seguente regola:

$Control(firmware, flussoDatiB) \wedge Comp(firmware) \wedge Spread(firmware, packetSniffing) \wedge \neg Safe(flussoDatiB, packetSniffing) \rightarrow Comp(flussoDatiB, packetSniffing)$.

Ai fini del corretto apprendimento e applicazione dell'analisi, risulta importante conoscere che l'entità definita *"flussoDati"* rappresenta l'insieme dei dati che i sensori inviano al sistema centrale. Le informazioni riguardano sia i valori relativi ai consumi energetici, sia quelli inerenti al monitoraggio delle presenze. Pertanto lo scenario d' attacco appena descritto può essere replicato analogamente: una volta che l'attaccante ha intercettato il canale di comunicazione tra i dispositivi di rilevazione delle presenze e il sistema centrale, può ottenere informazioni riguardo i dati scambiati. L'analisi restituirà la medesima situazione precedentemente valutata ma considerando diverse entità coinvolte.

Packet Injection

Esaminando l'attacco precedente, l'aggressore ha approfittato di vulnerabilità e compromissioni di entità come la rete domestica, il sistema centrale e le caratteristiche del *firmware* dei sensori per compiere attacchi su specifici componenti del sistema, in particolare ha avuto accesso alla informazioni che viaggiano sui canali di comunicazioni tra i dispositivi connessi.
In questo possibile scenario d'attacco, attraverso la tecnica di *"packet injection"*, molto diffusa nelle *Wireless Sensor Network*, l'attaccante mira a sfruttare le vulnerabilità delle rete e dei protocolli di comunicazione per ottenere un controllo non autorizzato sulla rete stessa, compromettendo l'integrità dei dati in transito mediante l'inserimento di pacchetti malevoli o manipolati. Possono quindi essere immessi pacchetti nuovi o alterati all'interno del flusso dei dati in modo che il destinatario, rappresentato dall'utente finale o dai sensori, li consideri legittimi. Questo possibile attacco può essere utilizzato dall'aggressore per modificare la logica di rilevazione delle presenze all'interno all'abitazione, in modo che i sensori non riescano ad individuare entità al suo interno; oppure può essere utilizzato, in casi più estremi, dall'utente stesso della casa per modificare i parametri dei consumi in modo da alterare i costi stessi.
Si consideri come esempio il flusso dati inerente alla smart home A.

```
%....
t(packetInjection).

connect(sistemaCentrale, mqttConnectionA, flussoDati).
spread(mqttConnectionA, packetInjection).
assVul(flussoDati, packetInjection).
protect(symmetricEncryptionKey, flussoDati, packetInjection).
%....
```

Attraverso queste proprietà si possono derivare gli stati di sicurezza del sistema in relazione alla minaccia analizzata. Si conoscono le vulnerabilità delle componenti ma esiste una misura di protezione per evitare che l'attaccante riesca a trarre vantaggio da queste criticità.

Si assume però che con il precedente attacco l'aggressore abbia ottenuto la chiave utilizzata per cifrare i messaggi, questo implica che la misura preventiva non è più valida per difendere l'entità da una possibile minaccia.

$\neg Safe(flussoDati, packetInjection) = Vul(flussoDati,$
$packetInjection) \wedge \neg Def(flussoDati, packetInjection).$

$\neg Def(flussoDati, packetInjection) = Protect(symmetricEncryptionKey,$
$flussoDati, packetInjection) \wedge \neg Valid(symmetricEncryptionKey).$

Applicando l'analisi di minaccia si può notare come esistano le totali condizioni per l'aggressore di sfruttare le vulnerabilità scoperte per diffondere l'attacco:

```
?-canbeComp(flussoDati,packetInjection).
%....
true.
```

$Connect(sistemaCentrale, mqttConnectionA, flussoDati) \wedge Comp($
$mqttConnectionA) \wedge Spread(mqttConnectionA, packetInjection)$
$\wedge \neg Safe(flussoDati, packetInjection) \wedge Comp(sistemaCentrale)$
$\rightarrow Comp(flussoDati, packetInjection).$

Capitolo 5

Valutazione degli attacchi

Il lavoro è mirato a condurre un'analisi dettagliata in uno scenario IoT in ambito smart home. Nello specifico, le due abitazioni coinvolte raccolgono dati e offrono un'interfaccia per interazioni con gli utenti, indipendentemente dalle loro competenze tecniche ma in base ai privilegi d'accesso assegnati. L'obiettivo primario consiste nell'assicurare la protezione all'interno delle due smart home, con un'attenzione particolare alle minacce più note, plausibili e realizzabili in un contesto abitato da entità "ibride" e "intelligenti", includendo attacchi rivolti a compromettere:

- **Identità degli utenti**;
- **Confidenzialità delle informazioni**;
- **Integrità dei dati**;
- **Rete e dispositivi**.

La Figura 5.1 descrive in maniera riassuntiva i primi attacchi analizzati. Cosi come i successivi schemi, viene posta l'attenzione in particolar modo su cinque concetti fondamentali:

- **Entità:** componente oggetto dell'analisi.
- **Protezione:** misura preventiva per difendere l'entità coinvolta da un possibile attacco.
- **Vulnerabilità:** debolezze o "falle" di una componente del sistema o dell'insieme di entità coinvolte.
- **Minaccia:** azioni che potrebbero causare danni o problemi al sistema esaminato.
- **Conseguenze attacco:** effetti e risultati che una minaccia causa; danneggiando le proprietà di sicurezza delle entità coinvolte ed interconnesse.

Questo metodo di visualizzazione può rappresentare uno strumento aggiuntivo in supporto allo studio dettagliato dei casi, consentendo una successiva formulazione di strategie adeguate in risposta ai risultati ottenuti.

SMART HOME A e SMART HOME B				
ENTITA'	PROTEZIONE	VULNERABILITA'	MINACCIA	CONSEGUENZE ATTACCO
Utente	no protezioni attive	mancanza di consapevolezza	Phishing	vengono sottratte informazioni sensibili quali credenziali dei sistemi e chiave d'accesso alla rete
Area Utente	sistema d'auteticazione	misura di protezione debole, necessario conoscere solo username e password	Accesso non Autorizzato	possibile furto di dati e manipolazione dispositivi connessi
Sistema Centrale	sistema d'auteticazione, 2FA	sistema d'autenticazione debole	Accesso non Autorizzato	con 2FA attaccante non riesce ad accedere, anche se conosce credenziali
Guest Account	sistema d'autenticazione, validità temporale credenziali	sistema d'auteticazione debole, tempo di validità credenziali lungo	Accesso non Autorizzato	conoscendo le credenziali, l'attaccante può liberamente entrare in possesso di alcune info sensibili. Il tentativo viene bloccato se la password utilizzata risulta scaduta
Server	firewall	porte aperte non necessarie, configurazione errata regole d'accesso firewall	DoS	malfunzionamento del server e conseguente indisponibilità del sistema centrale
Server	firewall	porte aperte non necessarie, errori di configurazione del firewall	Ip Spoofing	il server originale non risulta più attivo, l'utente viene indirizzato verso un server non sicuro
Sistema Centrale	sistema d'auteticazione, 2FA	sistema d'auteticazione debole e server replicabile	Shadow Server	Creando un server illecito, l'attaccante sottrae le credenziali 2FA per accedere al sistema

Figura 5.1: Schema riassuntivo dei primi scenari d'attacco analizzati.

Esaminando l'aspetto umano, emerge chiaramente come un utente specifico possa risultare più esposto ad una minaccia di *"phishing"*.
La conseguenza dell'attacco varia in base alla vittima, con diversi possibili percorsi creati dalle azioni dell'attaccante stesso (in Figura 5.2 si può osservare una possibile rappresentazione dell'attacco che coinvolge un "utente base"). E' importante notare che non esistono contromisure di natura "digitale" che possano mitigare o prevenire questa situazione. In altre parole, quando si riceve una *mail* fraudolenta, è responsabilità dell'utente riconoscere se il *link* fornito è legittimo o meno. Questo tipo di attacco apre la porta ad un possibile accesso non autorizzato ai sistemi, poichè l'aggressore riesce ad ottenere le credenziali d'accesso. Tuttavia, esistono misure di protezione di sicurezza, come ad esempio l'autenticazione a due fattori (2FA), che proteggono l'accesso al sistema centrale riservato all'amministratore. Così facendo è compito dell'entità malintenzionata attuare un diverso percorso d'accesso.

La tecnica di *"shadow server"* potrebbe costituire una strategia adottata dall'aggressore al fine di eludere tali misure di sicurezza. L'analisi mette in evidenza come sia possibile rendere "occulto" il *server* che ospita il sistema, mediante l'interazione di tecniche offensive quali *"DoS"* e *"IP spoofing"*. Questa serie di azioni si traduce nell'indirizzare inconsapevolmente l'amministratore verso una piattaforma ospitata su un *server* illecito, creato appositamente dall'aggressore per sottrarre le credenziali di autenticazione a due fattori e, di conseguenza, per infiltrarsi all'interno del sistema centrale.
Dalle analisi effettuate si possono considerare questi scenari come tipi di attacchi *"black box"*. L'attaccante non è a conoscenza dei privilegi e delle informazioni dell'utente, il che lo costringe ad impiegare risorse, tempo e capacità in modo inefficiente. L'unica certezza è che il malintenzionato riesce ad ottenere le credenziali per accedere alla rete domestica.
Considerando l'attacco di *phishing*, una contromisura potenziale per mitigare le problematiche potrebbe essere l'implementazione di un *software* di intelligenza artificiale per il rilevamento delle *mail* sospette. Tuttavia, anche in questo caso, non si può garantire una protezione al 100%, poichè il fattore umano gioca un ruolo determinante nel prendere decisioni corrette. Inoltre è possibile adottare una soluzione di archiviazione basata su *cloud* per conservare in modo sicuro le chiavi d'accesso alla rete domestica. Queste chiavi sarebbero accessibili solo tramite una richiesta autorizzata e successivamente verificate dal *server*. Tale approccio aggiunge uno strato di sicurezza supplementare ad un'entità che attualmente manca di un sistema di protezione stabile e robusto.

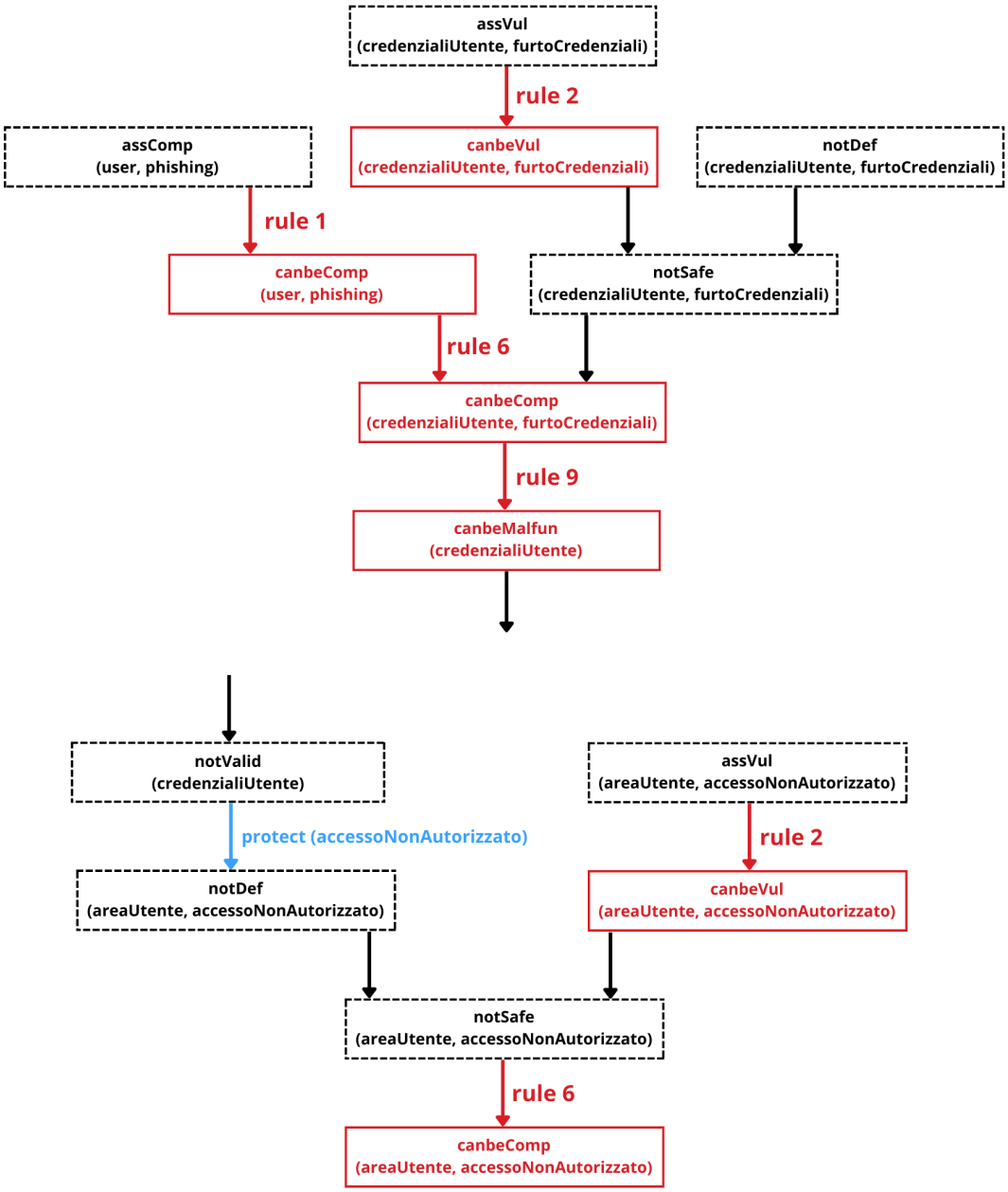

Figura 5.2: Dettaglio di *phishing attack path* e accesso nell'area utente.

Si osservi ora la Figura 5.3 che riepiloga schematicamente ulteriori risultati dell'analisi svolta.

SMART HOME A

ENTITA'	PROTEZIONE	VULNERABILITA'	MINACCIA	CONSEGUENZE ATTACCO
Rete	credenziali accesso, firewall	debolezza nella configurazione del firewall integrato nel router, vengono sfruttate porte non bloccate	DoS	invio di un elevata quantità di pacchetti in rete che comportano la saturazione di quest'ultima, causando problemi di performance e disponibilità delle componenti
Rete	credenziali accesso, firewall	sistema d'instrusione integrato nel firewall risulta debole e mal configurato	Man In The Middle	l'attaccante rimane in ascolto delle comunicazioni tra dispositivi attraverso la rete
Flusso Dati	Crittografia	crittografia simmetrica, una sola chiave facilmente reperibile.	Packet Sniffing, Brute Force Attack	sfruttando Man In The Middle vengono intercettati pacchetti che transitano in rete, decifrandoli in un tempo breve e compromettendo confidenzialità dei dati
Flusso Dati	Crittografia	crittografia simmetrica, una sola chiave facilmente reperibile.	Packet Injection	sfruttando Man in The Middle e Brute Force Attack vengono immessi pacchetti malevoli in rete, compromettendo integrità dei dati

SMART HOME B

ENTITA'	PROTEZIONE	VULNERABILITA'	MINACCIA	CONSEGUENZE ATTACCO
Rete	credenziali accesso, firewall	debolezza nella configurazione del firewall integrato nel router, vengono sfruttate porte non bloccate	DoS	invio di un elevata quantità di pacchetti in rete che comportano la saturazione di quest'ultima, causando problemi di performance e disponibilità delle componenti
Rete	credenziali accesso, firewall, IDS	sistema d'instrusione integrato nel firewall risulta debole e mal configurato	Man In The Middle	l'aggressore non riesce a concludere l'attacco, IDS risulta una misura ancora valida e attiva
Flusso Dati	firmware sensori	falla di sicurezza all'interno del codice firmware implementato nella rete di sensori	Packet Sniffing	stabilendo una connessione cablata con il dispositivo, l'attaccante sottrae i dati raccolti dai sensori, compromettendo confidenzialità
Flusso Dati	Crittografia	crittografia simmetrica, una sola chiave facilmente reperibile	Packet Injection	sfruttando Man in The Middle e Brute Force Attack vengono immessi pacchetti malevoli in rete, compromettendo integrità dei dati

Figura 5.3: Schema riassuntivo di ulteriori scenari d'attacco analizzati.

L'affidabilità della rete domestica e gli aspetti di natura "digitale" sono di assoluta importanza quando si progettano e analizzano sistemi *smart*. All'interno delle due abitazioni intelligenti, sono stati implementati *firewall* che fungono da barriera contro potenziali azioni dannose da parte di malintenzionati.

Sebbene l'attaccante sia in possesso delle credenziali per accedere alla rete, la presenza di misure di sicurezza limitano la sua attività alla mera navigazione. L'analisi delle minacce ha rivelato come l'attaccante sia in grado di aggirare il *firewall*, sfruttando vulnerabilità legate all'*access point* cui è connesso. Questo apre la strada a due possibili tipi di attacchi: *"Man in the Middle"* e *"DoS"*. Quest'ultimo mira a sovraccaricare i sistemi fino a causarne il malfunzionamento. In particolare, l'attaccante si concentra sulla saturazione dei sensori di movimento al fine di creare un percorso che gli consenta di penetrare successivamente all'interno delle abitazioni. Tuttavia, nella smart home B è presente un ulteriore strato di protezione che ostacola l'attaccante nel suo intento. Si introduce un sistema di rilevamento delle intrusioni (IDS), più rigoroso rispetto al *firewall*, il quale impedisce all'attaccante di infiltrarsi nella rete. Questi due tipi di attacco in sè non hanno una scopo definito, ma possono essere combinati per aprire diverse vie all'interno del sistema.
Analizzando con attenzione lo scenario dell'attacco con *"packet sniffing"*, comunemente utilizzato nelle reti di sensori, l'attaccante è in grado di intercettare le informazioni trasmesse attraverso il protocollo di comunicazione. Anche se queste informazioni risultano cifrate, attraverso una combinazione di attacchi per rivelare le chiavi, l'aggressore riesce a decifrare i dati. Ciò consente l'accesso a tutte le informazioni contenute nel pacchetto.
D'altra parte, nell'ambito della smart home B, risulta esplicito che la medesima procedura di attacco non può essere eseguita, impedendo all'entità malintenzionata di sfruttare le stesse risorse, come mostra l'*attack path* in Figura 5.4. Ora l'attaccante è forzato a studiare casi differenti per sottrarre dati e informazioni che viaggiano in rete. Infine, da ulteriori analisi approfondite è evidente come l'attaccante possa compromettere sia la confidenzialità che l'integrità dei dati, quest'ultima utilizzando un attacco noto come *"packet injection"*.

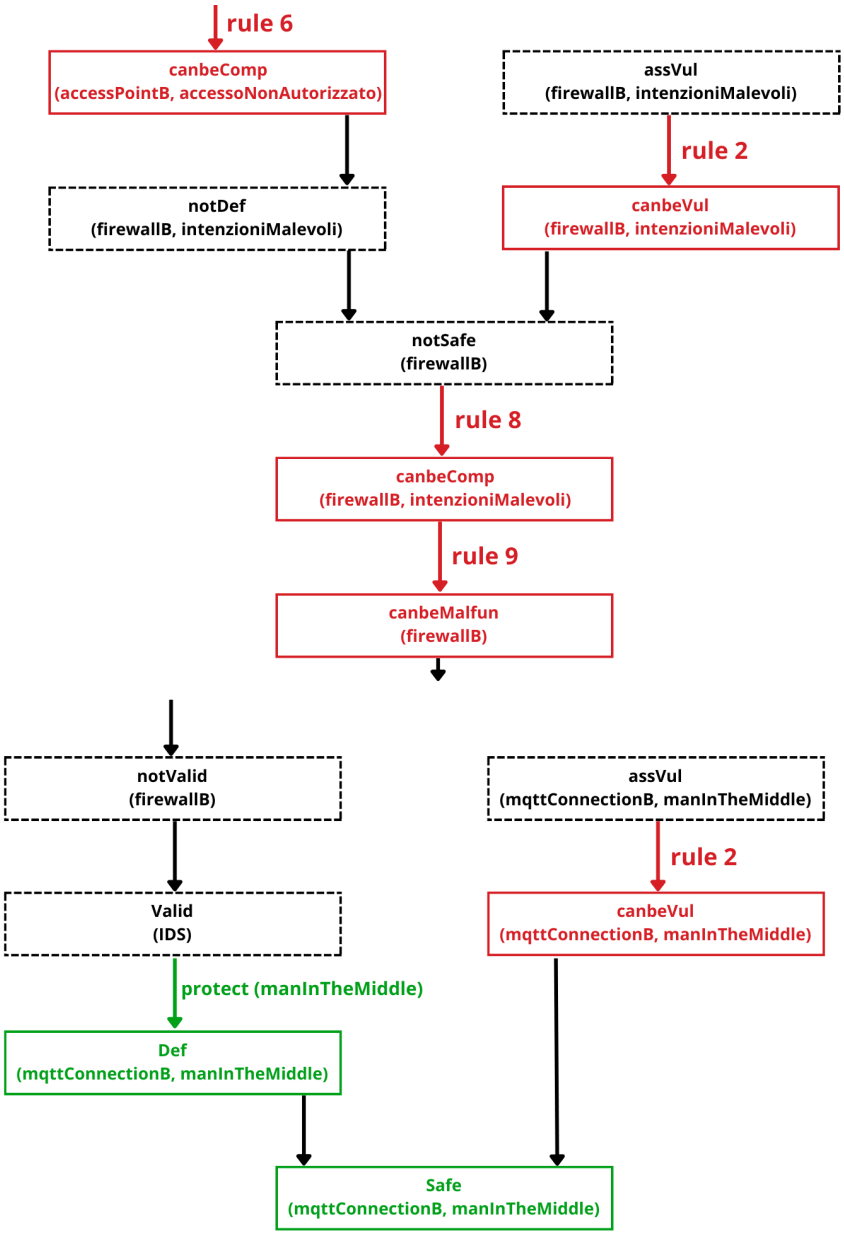

Figura 5.4: Dettaglio di *Man In The Middle attack path* nella smart home B.

Valutando ulteriori *performance*, includendo nell'analisi anche componenti di natura "fisica"; è opportuno osservare che le due smart home costituiscono un sistema integrato con diverse unità di intelligenza. L'interconnessione di molteplici dispositivi e le possibili esposizioni a rischi mettono in rilievo la potenziale capacità dell'attaccante di affettuare una *penetrazione fisica all'interno delle due abitazioni* (sintetizzato in Figura 5.5). E' noto che i sensori di rilevamento delle presenze siano realizzati e installati con lo scopo di prevenire queste eventualità, mantenendo un costante monitoraggio delle attività all'interno dell'ambiente *smart*.

SMART HOME A e SMART HOME B				
ENTITA'	PROTEZIONE	VULNERABILITA'	MINACCIA	CONSEGUENZE ATTACCO
Sensori di movimento	nessuna misura di protezione	vengono sfruttate le vulnerabilità note della rete domestica e/o del sistema centrale	DoS	i sensori di movimento vengono spenti o resi indisponibili, in modo tale da non avvertire l'utente di eventuali presenze.
Smart Home A	porta ingresso, sensori di rilevazione	si sfruttano le vulnerabilità che legano le connessioni tra sensori e rete	Furto	il sensore utilizzato per controllare la porta viene messo fuori uso, permettendo al malintenzionato di accedere all'abitazione
Smart Home B	porta ingresso, sensori di rilevazione	si sfruttano le vulnerabilità che legano le connessioni tra sensori e sistema centrale	Furto	il sensore utilizzato per controllare la porta viene messo fuori uso, permettendo al malintenzionato di accedere all'abitazione

Figura 5.5: Schema riassuntivo riguardante la compromissione fisica delle smart home.

Analizzando il contesto in maniera approfondita, emerge che l'aggressore deve elaborare e mettere in atto due approcci differenti, motivati dalle contromisure operative e dalla robustezza delle difese, al fine di raggiungere l'obiettivo. Nel caso della smart home A, trae vantaggio dall'utilizzo sinergico delle tattiche precedentemente delineate, in particolare ottenendo l'accesso alla rete, sfruttando l'attacco *DoS* e disabilitando i sensori di movimento.
Al contrario, nell'abitazione B, esaminando la Figura 5.6, l'aggressore può sfruttare un punto di ingresso al sistema centrale, procedendo con la disattivazione dei sensori e conseguente intrusione. Mediante quest'ultima modalità, l'attaccante apre una nuova via per attuare un metodo di *"packet sniffing"*, precedentemente irrealizzabile per via delle stringenti misure attuate. Di conseguenza, una volta guadagnato l'accesso alla smart home, l'ulteriore analisi eseguita rileva che è possibile stabilire una connessione fisica con il sensore, sfruttando le vulnerabilità all'interno del *firmware*, per poter monitorare le informazioni catturate dal sensore stesso e trasmesse al sistema centrale.

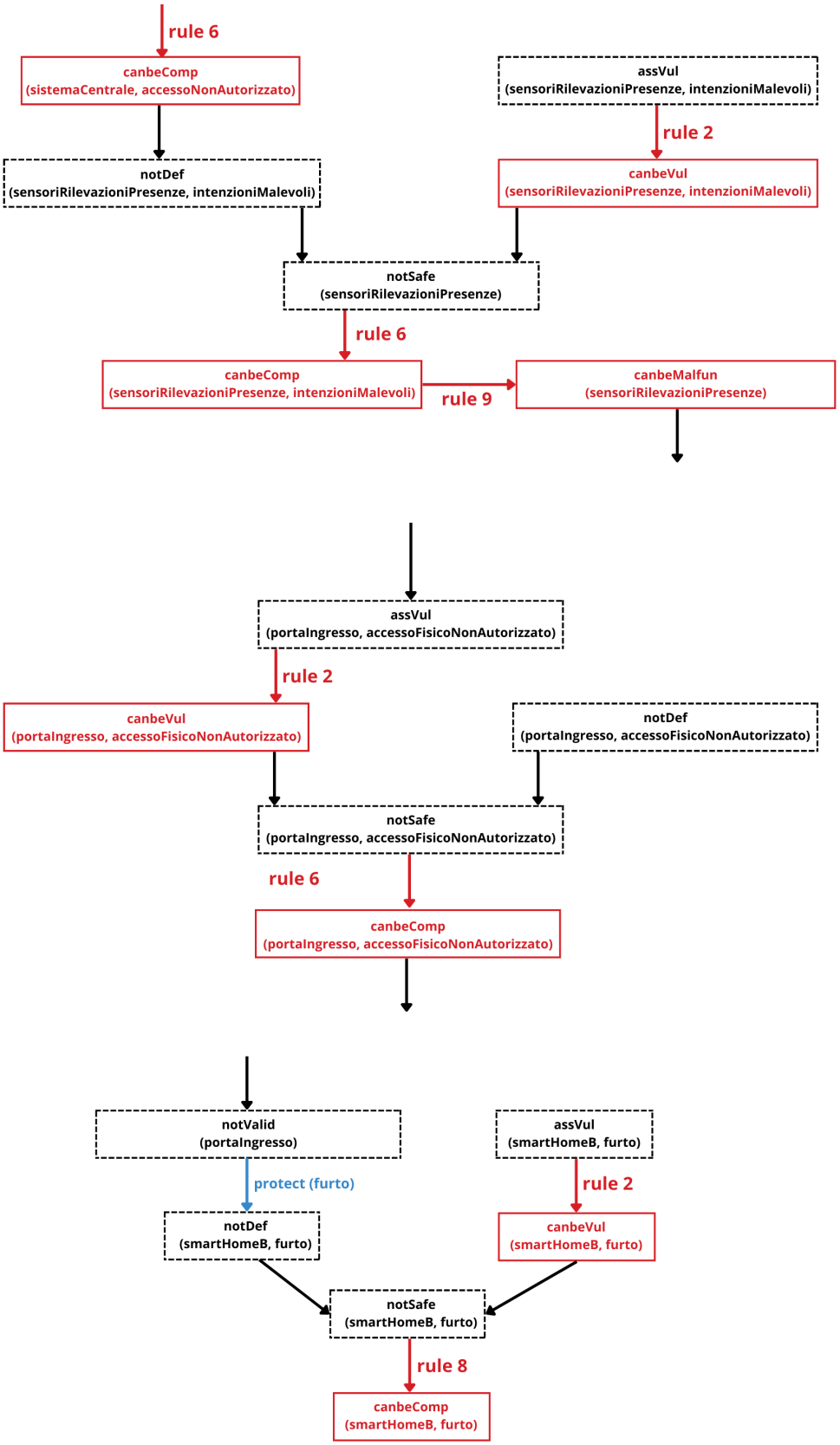

Figura 5.6: Dettaglio di *attack path* che permette l'intrusione fisica nella smart home B.

In conclusione, i risultati ottenuti dipendono dall'ambiente specifico, dagli attori coinvolti, dalle proprietà di sicurezza e dalle relazioni conosciute che insieme, determinano la definizione di un sistema vulnerabile a determinati attacchi, compromesso e malfunzionante, o contrariamente munito di adatte misure di protezione per prevenire o mitigare le minacce trattate.

Capitolo 6

Conclusione e sviluppi futuri

Questo lavoro di tesi si è concentrato sull'approfondimento della tematica IoT e *security management*. In particolare, è stato definito uno scenario smart home, che comprendeva diverse sorgenti dati per il monitoraggio dei consumi e delle presenze all'interno di due abitazioni. Alla luce delle iterazioni e delle autorizzazioni tra i sensori che hanno fornito i dati e gli utenti che li hanno visualizzati, si è definito un *threat model*, volto ad individuare eventuali vulnerabilità e attacchi al sistema a seguito di compromissione di uno o più dispositivi e/o delle identità degli utenti. L'*output* previsto consiste in una *threat analysis* derivante dal *threat model*.

Più nello specifico, il progetto è stato sviluppato dalla necessità di individuare possibili minacce nei sistemi *cyber-fisici* interconessi con l'ambiente IoT, che hanno combinato aspetti umani, fisici e informatici, con l'obiettivo finale di garantire la loro protezione. Questo studio è stato implementato con un *tool* automatico chiamato TAMELESS, ampiamente analizzato in [15], il quale ha introdotto un modello innovativo di minaccia per il sistema preso in considerazione, includendo in modo integrato tutti gli aspetti sopra descritti. Il modello ha illustrato le interconnessioni tra i componenti del sistema e le caratteristiche di sicurezza. Un metodo di analisi delle minacce ha accompagnato il modello, consentendo una valutazione completa della sicurezza dei componenti: ha esaminato le minacce, verificato le proprietà di sicurezza e prodotto risultati per conoscere lo stato dello scenario e facilitare le successive scelte di strategie adeguate di protezione.

Le funzionalità elaborate al fine di garantire il risultato atteso sono due. La prima ha consentito di modellare lo scenario preso in considerazione attraverso proprietà definite dal *tool*, specificando relazioni tra componenti e vulnerabilità note. La seconda funzionalità ha permesso di condurre la *threat analysis*, interrogando il terminale basato su Prolog. Si è individuato, per ogni entità e per ogni proprietà di sicurezza, l'insieme dei possibili attacchi volti a danneggiare il sistema, fornendo in *output*, attraverso regole di derivazione implementate da TAMELESS, una rappresentazione del percorso d'attacco e delle componenti interessate.

Per quanto riguarda le prospettive future esistono diverse opportunità da esaminare attentamente. In primo luogo, si potrebbe automatizzare l'identificazione delle relazioni e delle caratteristiche di sicurezza dei componenti del sistema. Nonostante tali informazioni siano state precedentemente fornite al modello delle minacce, molte di esse potrebbero essere estratte automaticamente dalle caratteristiche degli scenari, migliorando così la loro acquisizione e comprensione. E' importante notare che le relazioni, le caratteristiche di sicurezza e le regole utilizzate dal modello di analisi sono fisse e binarie nella loro valutazione. Tuttavia, considerando l'associazione di informazioni probabilistiche a queste diverse caratteristiche e vulnerabilità, si apre la strada ad un ragionamento più sofisticato sulla gestione del rischio, sia in fase di progettazione che durante un attacco. Nel prossimo futuro si provvederà ad utilizzare *dataset* reali che monitorano in *real-time* l'andamento delle smart home. Inoltre si prevederà di estendere il *tool* a sistemi dinamici, in cui nuovi componenti possono essere aggiunti o rimossi dal sistema. In questo contesto saranno disponibili informazioni dirette sulle possibili contromisure da adottare per mitigare o prevenire minacce future.
Questi approcci consentiranno così di ampliare il numero di smart home considerate ed estendere le funzionalità del modello, simulando l'utilizzo di dati e componenti in scenari più complessi, dinamici e autentici, con lo scopo di modellare in maniera più accurata i sistemi coinvolti.

Bibliografia

[1] S.Sicari et al. «A security-and-quality-aware system architecture for Internet of Things». In: *Inf Syst Front* 18 (2016), pp. 665–677. URL: https://doi.org/10.1007/s10796-014-9538-x (cit. a p. 3).

[2] Sabrina Sicari et al. «Security policy enforcement for networked smart objects». In: *Computer Networks* 108 (2016), pp. 133–147. URL: https://doi.org/10.1016/j.comnet.2016.08.014 (cit. alle pp. 3, 8).

[3] Tom Kirkham et al. «Risk driven Smart Home resource management using cloud services». In: *Future Generation Computer Systems* 38 (2014), pp. 13–22. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167739X13001714 (cit. a p. 3).

[4] Andreas Jacobsson, Martin Boldt e Bengt Carlsson. «A risk analysis of a smart home automation system». In: *Future Generation Computer Systems* 56 (2016), pp. 719–733. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167739X15002812 (cit. a p. 4).

[5] Thomas R Peltier. *Information security risk analysis*. CRC press, 2005. URL: https://books.google.it/books?hl=it&lr=&id=n8Z1RDjEKa0C&oi=fnd&pg=PR7&dq=Information+security+risk+analysis&ots=Salnn6cD3W&sig=n_8YUX0pIAJ-Z4Yx_2IU3KTTpcw&redir_esc=y#v=onepage&q=Information%20security%20risk%20analysis&f=false (cit. a p. 4).

[6] Laurens Lemaire et al. «Extracting vulnerabilities in industrial control systems using a knowledge-based system». In: *3rd International Symposium for ICS & SCADA Cyber Security Research 2015* (2015), pp. 1–10. URL: https://www.scienceopen.com/hosted-document?doi=10.14236/ewic/ICS2015.1 (cit. alle pp. 4, 5).

[7] Christos Tsigkanos et al. «Ariadne: Topology aware adaptive security for cyber-physical systems». In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* 2 (2015), pp. 729–732. URL: https://ieeexplore.ieee.org/abstract/document/7203054 (cit. a p. 4).

[8] Christos Tsigkanos et al. «On the interplay between cyber and physical spaces for adaptive security». In: *IEEE Transactions on Dependable and Secure Computing* 15.3 (2016), pp. 466–480. URL: https://ieeexplore.ieee.org/abstract/document/7542583 (cit. a p. 4).

[9] Ebenezer A Oladimeji, Sam Supakkul e Lawrence Chung. «Security threat modeling and analysis: A goal-oriented approach». In: *Proc. of the 10th IASTED International Conference on Software Engineering and Applications (SEA 2006)* (2006), pp. 13–15 (cit. a p. 4).

[10] Daniele Sgandurra e Emil Lupu. «Evolution of attacks, threat models, and solutions for virtualized systems». In: *ACM Computing Surveys (CSUR)* 48.3 (2016), pp. 1–38. URL: https://dl.acm.org/doi/abs/10.1145/2856126 (cit. a p. 4).

[11] Daniele Sgandurra, Erisa Karafili e Emil Lupu. «Formalizing threat models for virtualized systems». In: *Data and Applications Security and Privacy XXX: 30th Annual IFIP WG 11.3 Conference, DBSec 2016, Trento, Italy, July 18-20, 2016. Proceedings 30* (2016), pp. 251–267. URL: https://link.springer.com/chapter/10.1007/978-3-319-41483-6_18 (cit. alle pp. 4, 5).

[12] Xinming Ou, Wayne F Boyer e Miles A McQueen. «A scalable approach to attack graph generation». In: *Proceedings of the 13th ACM conference on Computer and communications security* (2006), pp. 336–345. URL: https://dl.acm.org/doi/abs/10.1145/1180405.1180446 (cit. a p. 4).

[13] Ravi Akella, Han Tang e Bruce M McMillin. «Analysis of information flow security in cyber–physical systems». In: *International Journal of Critical Infrastructure Protection* 3.3-4 (2010), pp. 157–173. URL: https://doi.org/10.1016/j.ijcip.2010.09.001 (cit. a p. 5).

[14] Marco Rocchetto e Nils Ole Tippenhauer. «On attacker models and profiles for cyber-physical systems». In: *Computer Security–ESORICS 2016: 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part II 21* (2016), pp. 427–449. URL: https://link.springer.com/chapter/10.1007/978-3-319-45741-3_22 (cit. a p. 5).

[15] Fulvio Valenza et al. «A hybrid threat model for smart systems». In: *IEEE Transactions on Dependable and Secure Computing* (2022). URL: https://ieeexplore.ieee.org/abstract/document/9916127 (cit. alle pp. 5, 29, 72).

[16] K. Ashton et al. «That 'internet of things' thing:» in: *RFID journal* 22.7 (2009), pp. 97–114. URL: http://www.itrco.jp/libraries/RFIDjournal-That%5C%20Internet%5C%20of%5C%20Things%5C%20Thing.pdf (cit. a p. 7).

[17] M. Khan e K. Han BN Silvia. «Internet of Things: A Comprehensive Review of Enabling Technologies, Architectures and Challenges». In: *IETE Tech.Rev.* 0.0 (2018), pp. 1–16. URL: https://doi.org/10.1080/02564602.2016.1276416 (cit. a p. 8).

[18] Hasan Ali Khattak et al. «Perception layer security in Internet of Things». In: *Future Generation Computer Systems* 100 (2019), pp. 144–164. URL: https://doi.org/10.1016/j.future.2019.04.038 (cit. a p. 8).

[19] Poorana Senthilkumar e Bhavadharani Subramani. «Study on IoT Architecture, Application Protocol and Energy Needs». In: *International*

Journal of Advanced Science and Technology 8.31 (2020), pp. 7–12. URL: https://www.researchgate.net/publication/352212549_Study_on_IoT_Architecture_Application_Protocol_and_Energy_needs (cit. a p. 9).

[20] Shivangi Vashi et al. «Internet of Things (IoT): A vision, architectural elements, and security issues». In: *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)* (2017), pp. 492–496. URL: https://ieeexplore.ieee.org/abstract/document/8058399 (cit. a p. 9).

[21] JeongGil Ko et al. «Wireless sensor networks for healthcare». In: *Proceedings of the IEEE* 98.11 (2010), pp. 1947–1960. URL: https://ieeexplore.ieee.org/abstract/document/5570866 (cit. a p. 11).

[22] Tobias Zillner e Sebastian Strobl. «ZigBee exploited: The good, the bad and the ugly». In: *Black Hat–2015. Available online: https://www. blackhat. com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly. pdf (accessed on 21 March 2018)* (2015) (cit. a p. 14).

[23] Firas Albalas et al. «Security-aware CoAP application layer protocol for the internet of things using elliptic-curve cryptography». In: *Power (mw)* 1333 (2018), p. 151. URL: https://www.researchgate.net/publication/325987571_Security-aware_CoAP_Application_Layer_Protocol_for_the_Internet_of_Things_using_Elliptic-Curve_Cryptography (cit. alle pp. 18, 20).

[24] Francesco Buccafurri, Vincenzo De Angelis e Roberto Nardone. «Securing mqtt by blockchain-based otp authentication». In: *Sensors* 20.7 (2020), p. 2002. URL: https://www.mdpi.com/1424-8220/20/7/2002 (cit. a p. 20).

[25] Abebe Diro et al. «Lightweight authenticated-encryption scheme for internet of things based on publish-subscribe communication». In: *IEEE Access* 8 (2020), pp. 60539–60551. URL: https://ieeexplore.ieee.org/abstract/document/9045934 (cit. a p. 20).

[26] Xueqi Fan et al. «Security analysis of zigbee». In: *MWR InfoSecurity* 2017 (2017), pp. 1–18 (cit. a p. 20).

[27] Frank Swiderski e Window Snyder. *Threat Modeling*. Microsoft Press, 2004. URL: https://dl.acm.org/doi/abs/10.5555/983226 (cit. alle pp. 22, 25).

[28] Guglielmo Yurcik Suvda Myagmar Adam J. Lee. «Threat Modeling as a Basis for Security Requirements». In: *Symposium on Requirements Engineering for Information Security (SREIS)* (2005). URL: http://d-scholarship.pitt.edu/id/eprint/16516 (cit. a p. 24).