UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Laurea Triennale in Ingegneria Informatica

Internet of Things: rilevare attacchi informatici mediante analisi del consumo energetico

Relatore:

Prof. Alessio Vecchio

Prof. Pericle Perazzo

Candidato:

Giorgio Cecchi

ANNO ACCADEMICO 2024/2025

Indice

Capitolo 1

Introduzione

L'avvento dei dispositivi IoT ha introdotto un nuovo modo di interagire con la tecnologia: oggi ognuno di noi è circondato da oggetti "intelligenti" in grado di semplificare o automatizzare diverse attività quotidiane. Si tratta per lo più di dispositivi di uso comune dotati di connettività Internet, capaci di raccogliere e scambiare dati sia tra loro sia con l'utente, con l'obiettivo di ottimizzare processi e attività. Negli ultimi anni la diffusione di tali dispositivi ha conosciuto una crescita esponenziale, raggiungendo decine di miliardi di unità a livello globale, impiegate in una vasta gamma di settori e contesti applicativi. Questo fenomeno rappresenta un grande vantaggio per gli utenti, ma allo stesso tempo apre sfide complesse in ambito di sicurezza informatica.

Per dispositivi come computer, tablet e smartphone la sicurezza è affidata a strumenti consolidati, come antivirus e antispyware, che monitorano in tempo reale l'attività del sistema rilevando e bloccando potenziali minacce. Queste soluzioni tradizionali, pur essendo ampiamente diffuse ed efficaci, non risultano applicabili ai dispositivi IoT. La ragione risiede nelle loro caratteristiche: forte eterogeneità, risorse computazionali e di memoria limitate, spesso alimentazione a batteria con capacità ridotta. Di conseguenza, l'adozione di meccanismi di difesa tradizionali può risultare impraticabile o comportare consumi energetici eccessivi, compromettendo la durata operativa del dispositivo. La sicurezza dei dispositivi IoT, quindi, rappresenta una sfida tuttora aperta.

Una delle strade esplorate per superare tali limiti è l'analisi del consumo energetico. Ogni processo in esecuzione su un dispositivo, infatti, genera un particolare pattern di consumo. Analizzando tali pattern è possibile individuare quale attività sia in corso e, nello specifico, identificare eventuali attività malevole. Un aspetto cruciale di questo approccio è che il monitoraggio dei consumi può essere effettuato da un dispositivo esterno, senza incidere sulle prestazioni del dispositivo osservato. Numerosi studi hanno indagato questa direzione, mostrando risultati promettenti, come illustrato anche nel Capitolo 2 — Related Works. Il presente lavoro si inserisce in tale contesto con l'obiettivo di fornire ulteriori evidenze sull'affidabilità di questo metodo.

L'attività sperimentale è consistita nel monitoraggio del consumo energetico di un dispositivo IoT simulato durante sei scenari. Tre di essi riproducono attività normali tipiche di un IoT: acquisizione e trasmissione di dati sensoriali, elaborazione video con rilevazione di presenza umana (telecamera smart) e funzionamento come server web. Gli altri tre scenari corrispondono invece ad attività malevole: un attacco brute force sull protocollo SSH subito dal dispositivo, e due attacchi DoS (tramite traffico DNS e HTTP) eseguiti dal dispositivo compromesso. In tutti i casi i dati sono stati raccolti con strumenti esterni ad alta precisione, mantenendo invariato lo stato operativo del dispositivo, ad eccezione del test oggetto di analisi, così da garantire la confrontabilità dei risultati.

I dati acquisiti sono stati elaborati e impiegati per l'addestramento di modelli di classificazione in tre differenti scenari:

- Primo scenario: modello addestrato a riconoscere quale attività è in corso tra le sei analizzate;
- Secondo scenario: modello addestrato a distinguere tra attività normali e malevole;

- Terzo scenario: modello addestrato con i soli dati relativi a attività normali con l'obiettivo di distinguere le attività malevole.

I risultati dei primi due scenari sono stati estremamente positivi, con un'accuratezza quasi completa. Anche nel terzo scenario il modello ha mostrato buone capacità discriminative, seppur con prestazioni leggermente inferiori, come atteso data la maggiore complessità del compito.

In conclusione, i risultati ottenuti sono promettenti, ma derivano da un campione limitato di attività; non è quindi possibile affermare con certezza che l'analisi dei consumi energetici rappresenti la soluzione definitiva alla sicurezza degli IoT. Tuttavia, la convergenza di diversi studi, incluso il presente, verso esiti positivi lascia intravedere la possibilità che in futuro questo approccio possa costituire la base per nuovi meccanismi di difesa, contribuendo a colmare una delle principali lacune oggi esistenti nella protezione dei dispositivi IoT, per i quali mancano soluzioni di sicurezza realmente efficaci.

Capitolo 2

Related Works

2.1 Leveraging power consumption for anomaly detection on IoT devices in smart homes

[1] https://link.springer.com/article/10.1007/s12652-022-04110-6

A seguito della crescita esponenziale del numero di dispositivi dell'IoT emergono preoccupazioni per quanto riguarda la sicurezza informatica; infatti, nel 2022 si stimava che ci fossero circa 2 milioni di dispositivi vulnerabili ad attacchi, destinati ad aumentare significativamente negli anni successivi.

Poiché i meccanismi di difesa convenzionali non sono efficaci nel contesto dell'IoT, dispositivi eterogenei e con risorse limitate, questo articolo vuole evidenziare come il rilevamento delle anomalie possa essere una buona contromisura per proteggersi dagli attacchi informatici.

Generalmente un dispositivo infettato da un malware consuma più energia; quindi, un suo comportamento anomalo può essere facilmente rilevato analizzando le caratteristiche del suo consumo energetico.

L'articolo analizza lo scenario di una smart home con alcune camere smart, per non sovraccaricarle l'Anomaly Detection System proposto presenta un server (interno o esterno) che si occupa dell'identificazione di anomalie tramite i consumi energetici dei dispositivi.

Nella fase di raccolta dati la camera smart è stata simulata con un Raspberry Pi e utilizzando una Power Measurement Unit sono stati raccolti i dati relativi al suo consumo. In particolare, sono stati raccolti dati sul consumo in vari casi di operatività della camera smart simulata:

- Attività normale
- Attacco Brute Force sulla smart camera
- La smart camera compromessa esegue un attacco brute force ad un altro dispositivo
- La smart camera compromessa esegue un attacco DDoS SYN Flood ad un altro dispositivo

I dati sul consumo energetico sono poi stati usati per addestrare vari algoritmi di machine learning, in particolare: Support Vector Machine (SVM), K-Nearest Neighbors (kNN), Naive Bayes (NB), Logistic Regression (LR), One-Class Support Vector Machine (OCSVM) e Deep Feed-forward Neural Network (DFNN).

Dopo l'addestramento tutti in modelli hanno dato risultati ottimi, producendo un'accuratezza e una precisione tra il 96% e il 99%.

In conclusione è stato dimostrato che il consumo energetico è un fattore affidabile nel rilevamento di anomalie nei dispositivi IoT e che gli algoritmi di Machine Learning sono idonei a tale scopo.

2.2 Smart-Power: A Smart Cyber-Physical System to Detect IoT Security Threat through Behavioral Power Profiling

[2] https://ieeexplore.ieee.org/abstract/document/9202734

Con l'aumento del numero di dispositivi e/o sensori wireless IoT nel contesto delle smart home, la sicurezza di tali dispositivi è un problema sempre più rilevante.

I dispositivi IoT potrebbero esser usati per vari tipi di azioni dannose come ottenere di informazioni riservate senza accesso autorizzato. L'idea è quella di monitorare il consumo di questi dispositivi per rilevare attività anomale e quindi possibili minacce. Avendo a disposizione un profilo di consumo energetico del dispositivo, si potrebbero confrontare i dati sul consumo in tempo reale con quelli del profilo e rilevare le anomalie.

L'articolo propone un sistema cyber-fisico basato su smartphone, chiamato "Smart Power". Il sistema di monitoraggio è composto principalmente da un Arduino Uno e uno smartphone. L'Arduino, tramite sensori di corrente e tensione, monitora gli assorbimenti di una presa elettrica alla quale si può collegare qualsiasi dispositivo IoT. Grazie ad un modulo bluetooth i dati letti dai sensori vengono inviati allo smartphone, sul quale è in esecuzione un'applicazione che li analizza. L'analisi avviene tramite il confronto dei dati in tempo reale con un profilo di consumo del dispositivo in un contesto normale e, se viene rilevata un'anomalia, l'applicazione lo notifica all'amministratore.

La raccolta dei dati è stata fatta su tre dispositivi IoT: Amazon Alexa Echo Dot, Wyze Camera e Avatar Mini Smart Light. Per ogni dispositivo sono stati effettuati test in sei scenari:

- Stato Inattivo
- Stato Attivo
- Stato Inattivo con attacco DDoS
- Stato Attivo con attacco DDoS
- Stato Inattivo con attacco MitM
- Stato Attivo con attacco MitM

Infine, per analizzare lo stato dei dispositivi IoT e identificare eventuali minacce è stato usato il classificatore K-Nearest Neighbor (K-NN). La classificazione è avvenuta in due fasi: nella prima fase è stata usata come unica feature la potenza media; nella seconda fase invece è stata aggiunta come feature anche la Power Spectral Density (PSD).

I risultati sull'accuratezza nell'individuare una minaccia sono: 85% (71% senza PSD) per l'Echo Dot, 71% (57% senza PSD) per la Smart Light e 65% (43% senza PSD) per la Wyze Cam. Tali risultati dimostrano come l'aggiunta della PSD abbia migliorato l'accuratezza predittiva del modello del 17%, facendo una media del miglioramento percentuale tra i 3 dispositivi IoT testati.

2.3 Energy Audition based Cyber-Physical Attack Detection System in IoT

[3] https://dl.acm.org/doi/abs/10.1145/3321408.3321588

Il concetto alla base di questo articolo è che i sistemi informatici utilizzano l'energia e che proprio l'energia ci può dare informazioni sul sistema, in particolare anche per quanto riguarda la sicurezza. Nel contesto dei sistemi dell'IoT gli attacchi informatici sono difficili da rilevare poiché il loro impatto può rivelarsi in un secondo momento, ma la questione della sicurezza per i sistemi IoT è diventata di rilevante importanza al giorno d'oggi.

Le tecnologie di attack detection si dividono in due categorie: metodi software-based e metodi data-driven. La categoria software-based non è abbastanza versatile da adattarsi alla continua evoluzione degli attacchi, invece la categoria data-driven, che si basa sull'osservazione delle statistiche e dell'utilizzazione del sistema, è adatta.

Il problema dei metodi software-based è che le statistiche utilizzate vengono riportate dal nucleo del dispositivo; pertanto, se il dispositivo è compromesso l'integrità dei dati non può esser garantita. Il consumo energetico è un parametro che può esser controllato all'esterno del nucleo del dispositivo, al sicuro da possibili manipolazioni da parte di un attacco, ecco che è quindi un buon parametro per fare attack detection.

Gli autori dell'articolo realizzano quindi un esperimento in cui simulano un dispositivo dell'IoT con un Raspberry Pi 3 Modello B, con il sistema operativo Raspbian. Nel dispositivo è installato un programma che va in esecuzione in background e che periodicamente preleva dati dai sensori, li salva localmente e li trasmette ad un server. Inoltre, è presente un misuratore di energia che memorizza i dati sul consumo durante i test.

Durante la raccolta dati viene memorizzato il segnale sul consumo energetico in varie condizioni: attività normale senza attacco e attività normale con attacco in corso. Gli attacchi analizzati sono sia software come Virus, Intrusione, Deny of Service (DoS), Trojan, che fisici come Riscaldamento e Interruzione della Linea Elettrica. I segnali vengono poi filtrati per ridurre il rumore e su di essi vengono estratte le seguenti feature: Mean, Standard deviation, Skewness, Kurtosis, Min, Max, Interquartile range, cumulative sum e Fast Fourier transform.

Infine, vengono applicati due algoritmi di classificazione: k-nearest neighbor (KNN) e neural network (NN). In entrambi i casi vengono fatte due classificazioni, breve termine e lungo termine, e viene utilizzata la 10-fold cross-validation. Nella classificazione a breve termine vengono usate finestre di 10 secondi e in input viene utilizzato solo il segnale filtrato, in quella a lungo termine invece le finestre sono da 180 secondi e vengono utilizzate anche le feature estratte oltre al segnale filtrato.

In conclusione, la classificazione a breve termine da come risultato un'accuratezza media del 82,48%, mentre quella a lungo termine del 98.64%.

2.4 Attacks on IoT: Side-Channel Power Acquisition Framework for Intrusion Detection

[4] https://www.mdpi.com/1999-5903/15/5/187

L'articolo inizia con una panoramica sulla crescita del numero di attacchi informatici negli ultimi anni: nel 2020 gli attacchi malware sono aumentati del 358% rispetto al 2019, e sono continuati ad aumentare negli anni successivi. Ad esempio, nella prima metà del 2022 si sono verificati circa 236 milioni di attacchi ransomware: un tipo di malware che dopo aver infettato il dispositivo, ne crittografa il disco rigido e lo "trattiene" a meno del pagamento di un riscatto.

Con l'avvento dei dispositivi IoT, vista anche la loro esponenziale diffusione in tutto il mondo, la superficie di attacco è cresciuta notevolmente. L'international Data Corporation (IDC) prevede che nel 2025 ci saranno circa 55,7 miliardi di dispositivi IoT nel mondo.

Nel settembre 2017 tramite la botnet Mirai è stato registrato uno degli attacchi DDoS più grandi mai registrato al mondo. Le stime dicono che tale botnet fosse composta da circa 380.000 dispositivi IoT ed è stata usata per attacchi con livelli di traffico senza precedenti, ad esempio, contro l'host web francese OVH con un traffico stimato di 1,1-1,5 terabit al secondo (Tbps). In conclusione, se 380.000 dispositivi sono riusciti a creare tale disservizio si prevede che in futuro con un pool di 55,7 miliardi di dispositivi attacchi di questo tipo saranno più potenti e diffusi.

Emerge quindi la necessità di migliorare la sicurezza dei dispositivi IoT, che per loro natura sono limitati in termini di risorse e pertanto inadatti ai sistemi di sicurezza convenzionali. I primi passi in questo ambito hanno sfruttato il traffico di rete del dispositivo IoT per rilevare attività dannose; tuttavia, un'altra area chiave per questo scopo è quella dei dati dell'alimentazione del dispositivo. L'articolo propone quindi un framework di acquisizione dati di potenza al fine di rilevare attacchi al dispositivo.

Lo studio è stato effettuato su due dispositivi: un Raspberry Pi 3 Modello B e un DragonBoard 410. Complessivamente si stima che ci siano circa 245 milioni di tali dispositivi in circolazione. Durante i test le condizioni di lavoro di entrambe le schede erano le seguenti: connessione ad Internet tramite Wi-Fi, Bluetooth attivo per comunicare con altri sensori e protocollo SSH abilitato. Infine, entrambe le schede ciclicamente ogni cinque secondi richiedono dati al sensore, li elaborano e li visualizzano. Tali condizioni sono fedeli a quelle di reale operatività un dispositivo IoT.

Durante lo studio le schede sono state sottoposte ad attacchi relativi a riservatezza, integrità e disponibilità. In particolare, gli attacchi effettuati sono stati: port-scanning, bruteforce SSH, DoS SYN Flood. Infine, è stato simulato anche uno scenario Capture The Flag (CTF) in cui l'attaccante cerca di accedere al sistema per ottenere un file segreto.

In conclusione, questo studio dimostra come i dati sull'alimentazione possano essere usati nell'ambito della sicurezza, poiché ogni attacco lascia una firma nella traccia di potenza.

2.5 WattsUpDoc: Power Side Channels to Nonintrusively Discover Untargeted Malware on Embedded Medical Devices

[5] https://www.usenix.org/conference/healthtech13/workshop-program/presentation/clark

L'articolo tratta la vulnerabilità dei dispositivi embedded, largamente utilizzati in contesti sanitari e industriali. Questi dispositivi spesso non sono compatibili con le classiche tecniche software di protezione, come ad esempio antivirus o i cosiddetti Network Intrusion Detection System (NIDS). Infatti, vengono largamente utilizzati firmware o sistemi operativi custom per i quali non esistono antivirus, inoltre questi dispositivi hanno risorse limitate e quindi è generalmente difficile implementare tecniche con antivirus. Infine, c'è un certo gruppo di questi dispositivi che sono teoricamente compatibili con antivirus e NIDS ma spesso i produttori non permettono modifiche software da parte degli utenti, rendendo quindi impossibile l'installazione di un antivirus. Questo succede perché i produttori sostengo che a seguito dell'alterazione del software non possono più garantire la sicurezza del dispositivo.

Viene proposta un'alternativa: WattsUpDoc ovvero un sistema di monitoraggio del comportamento per sistemi embedded. WattsUpDoc sfrutta il consumo energetico del sistema per rilevare comportamenti anomali, inoltre non richiede nessuna modifica né hardware né software. Poiché i sistemi embedded svolgono generalmente attività ripetitive è possibile modellare i loro livelli di consumo energetico, WattsUpDoc usa tecniche di machine learning per associare pattern di consumo energetico ad attività malevole o sospette.

WattsUpDoc è stato testato su due dispositivi basati su Windows XP Embedded:

- Baxa ExactaMix 2400 compounder, un compostatore farmaceutico;
- Schweitzer SEL3354 substation computer, un dispositivo SCADA usato in ambito industriale.

I classificatori che usa WattsUpDoc sono quelli che hanno performato meglio nella fase di testing, ovvero: 3-Nearest Neighbors (3-NN), Multilayer Perceptron e Random Forest. Tutti classificano i dati distinguendo in attività normali o anormali, ed utilizzano la validazione incrociata stratificata a 10 fold per garantire risultati equilibrati. Infine, WattsUpDoc usa feature sia nel dominio del tempo come media, varianza, skewness, kurtosis, minimo, massimo, root mean square (RMS), intervallo interquartile (IQR) che nel dominio della frequenza come Trasformata di Fourier (FFT) e l'energia nei 10 intervalli inferiori (da 0 a 2.5 KHz). Durante i test le features sono state estratte da finestre di ampiezza tra 1 e 60 secondi, i risultati migliori sono stati ottenuti con finestre di 5 secondi.

I risultati dimostrano come WattsUpDoc abbia un'accuratezza su tutti e tre i classificatori scelti del 94% circa per il compounder e del 95.5% per il substation computer. Inoltre, le cinque feature che maggiormente contribuiscono alla corretta classificazione sono, in ordine: media, RMS, skewness, varianza e massimo/minimo.

Capitolo 3

Ambiente di lavoro

L'obiettivo degli esperimenti è stato quello di acquisire una traccia del consumo di corrente del dispositivo il più possibile significativa e priva di interferenze. Particolare attenzione è stata posta nel mantenere invariato l'ambiente di lavoro tra un esperimento e l'altro, al fine di evitare variazioni che potessero influenzare i risultati successivi.

Ciascun esperimento ha avuto una durata di trenta minuti, con una frequenza di campionamento di 4000 campioni al secondo. Di conseguenza, ogni traccia è costituita da 7.200.000 campioni, garantendo un livello di dettaglio adeguato all'analisi.

3.1 Hardware coinvolto

I dispositivi IoT oggetto di test sono stati simulati utilizzando un Raspberry Pi 3 Model B+ [6], un single-board computer (SBC) ampiamente diffuso in ambito didattico e sperimentale. Il sistema è stato configurato con Raspberry Pi OS (64-bit), che rappresenta la distribuzione ufficiale e maggiormente supportata dalla comunità. La scelta di questo dispositivo si è rivelata particolarmente adatta alle esigenze del lavoro grazie alla sua versatilità: pur disponendo di risorse hardware limitate rispetto a piattaforme più potenti, esso si presta efficacemente a un'ampia gamma di progetti e applicazioni, rendendolo uno strumento ideale per attività di simulazione e validazione sperimentale.

Per quanto riguarda l'analisi del consumo energetico, è stato impiegato un Otii Arc Pro prodotto da Qoitech [7], uno strumento professionale progettato specificamente per lo studio e l'ottimizzazione dell'efficienza energetica di dispositivi IoT alimentati a batteria o di sistemi embedded. L'Otii Arc Pro offre diversi vantaggi che lo rendono particolarmente indicato in questo contesto, tra cui:

- la possibilità di unire in un unico dispositivo sia l'alimentazione sia l'analisi dei consumi;
- un'elevata precisione nelle misurazioni, requisito essenziale per valutazioni attendibili;
- un'interfaccia e un software dedicato (Otii 3 [8]) che garantiscono un utilizzo intuitivo e user-friendly.

Grazie a tali caratteristiche, la combinazione tra Raspberry Pi 3 Model B+ e Otii Arc Pro ha permesso di creare di un ambiente di test affidabile e riproducibile, capace di unire semplicità di implementazione e rigore nelle analisi energetiche.

3.2 Setup

Il setup usato per condurre gli esperimenti è illustrato in Figura 3.1.

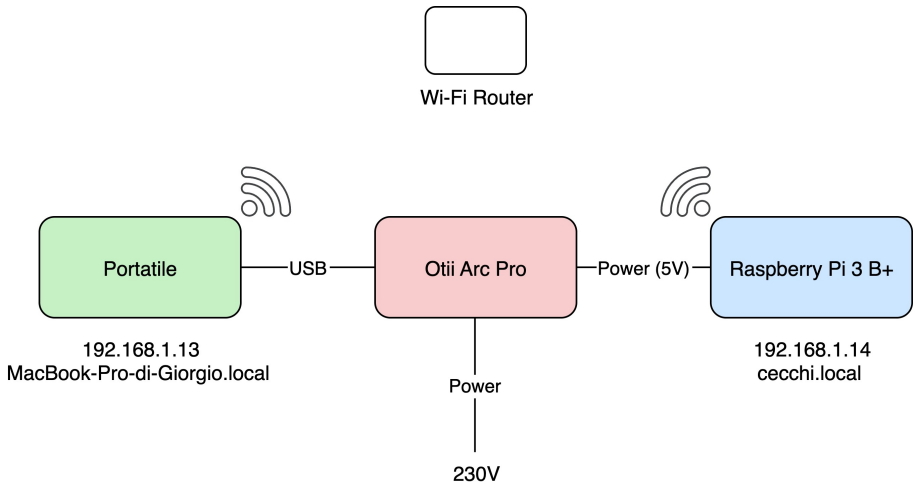

Figura 3.1: Setup

A sinistra dello schema sperimentale è collocato un portatile con in esecuzione il software Otii 3, che rappresenta il vero e proprio centro di controllo dell'esperimento. Attraverso questa applicazione, infatti, il computer gestisce sia il pilotaggio del dispositivo di alimentazione sia il controllo del dispositivo sotto test mediante connessione SSH. Contestualmente, sempre tramite Otii 3, il portatile svolge anche la funzione di raccolta e registrazione dei dati relativi al consumo energetico.

Il portatile è collegato all'Otii Arc Pro tramite connessione USB. Questo collegamento sarebbe già sufficiente a garantire il funzionamento dello strumento, permettendogli di erogare una tensione massima in uscita pari a 3.75 V. Tuttavia, poiché il dispositivo richiede una tensione di alimentazione superiore, l'Otii Arc Pro viene collegato anche alla rete elettrica domestica a 230 V. In questa configurazione, lo strumento è in grado di fornire fino a 5 V in uscita, valore che coincide con la tensione necessaria al funzionamento del Raspberry Pi 3 Model B+. L'uscita dell'Otii Arc Pro è quindi connessa direttamente ai pin di alimentazione del Raspberry Pi, assicurando un'alimentazione stabile e monitorata.

Infine, un ulteriore collegamento di tipo wireless tra il portatile e il Raspberry Pi 3 Model B+ consente sia il pilotaggio del dispositivo sia l'esecuzione dei test, garantendo così una comunicazione bidirezionale efficiente e completando l'architettura del banco prova.

Nella fase di setup di ciascun esperimento, il Raspberry Pi 3 Model B+ è stato inizialmente configurato tramite l'ausilio di periferiche esterne quali mouse, tastiera e monitor. Tuttavia, al fine di evitare assorbimenti di corrente indesiderati che avrebbero potuto alterare le misurazioni, tutte le periferiche sono state successivamente scollegate durante l'esecuzione del test. Il controllo del dispositivo è rimasto comunque possibile grazie alla connessione SSH, che consente l'avvio e la gestione dei test da remoto senza necessità di ulteriori collegamenti fisici.

Poiché l'alimentazione del Raspberry Pi 3 Model B+ è costante a 5 V, per il monitoraggio del suo consumo energetico è stato sufficiente misurarne l'assorbimento di corrente. Tale operazione è stata resa particolarmente agevole dall'impiego del software Otii 3, il quale integra nativamente una funzione dedicata alla registrazione dei consumi e permette inoltre di esportare i dati campionati in formato CSV. Questo ha reso possibile non solo la raccolta sistematica dei dati, ma anche la loro successiva elaborazione e analisi quantitativa in ambiente esterno.

Capitolo 4

Attività svolte

Gli esperimenti condotti sono stati progettati con l'obiettivo di simulare attività reali tipicamente eseguite da dispositivi IoT. In particolare, per quanto riguarda la simulazione di attività malevole, il riferimento principale è stato il noto attacco informatico noto come Botnet Mirai. Questo attacco, che ha avuto ampia risonanza mediatica, si basava sull'infezione di decine di migliaia di dispositivi IoT tramite malware, sfruttandoli per lanciare potenti attacchi DDoS (Distributed Denial of Service).

Di conseguenza, la prima attività sperimentale malevola è stata concepita per simulare l'infezione di un dispositivo, riproducendo le fasi iniziali dell'attacco Mirai. Le attività successive, invece, replicano il comportamento del dispositivo già compromesso, ovvero il suo impiego per generare traffico malevolo e partecipare a un attacco DDoS, consentendo così di osservare e analizzare il consumo energetico nelle diverse fasi operative di un attacco reale.

4.1 Attività normali

4.1.1 Simulazione di un sensore

Il primo test è stato progettato per simulare il comportamento di un sensore intelligente, ossia un dispositivo che ciclicamente raccoglie dati ambientali e li invia a un server. In questo caso, il comportamento è stato riprodotto prelevando alcuni parametri di sistema del Raspberry Pi 3 Model B+ tramite il comando vcgencmd, nello specifico: temperatura della CPU, tensione del core, frequenza della CPU e stato del throttling.

I dati raccolti sono stati successivamente inseriti nel payload di un pacchetto HTTP e inviati mediante il comando curl al portatile, il quale ha assunto il ruolo di server simulato, eseguendo uno script in ascolto in grado di ricevere le informazioni.

Codice completo disponibile nell' Appendice A.

4.1.2 Simulazione di una camera smart

Questo test è stato progettato per simulare il comportamento di una camera smart, ovvero un dispositivo in grado di registrare video e rilevare la presenza di esseri umani. Nella simulazione, il video non viene effettivamente registrato in tempo reale, ma viene analizzato un video già presente in locale sul Raspberry Pi 3 Model B+. I frame del video vengono prelevati e processati in modo da mantenere la velocità reale, e ogni 5 secondi il frame corrispondente viene analizzato per rilevare la presenza di persone.

Il rilevamento umano viene effettuato tramite un classificatore a cascata di Haar preaddestrato, specifico per il riconoscimento della parte superiore del corpo umano. Ogni volta che il classificatore individua un umano, l'evento viene notificato tramite una richiesta HTTP al portatile, sul quale è in esecuzione uno script che simula il comportamento di un server in ascolto. Vista la limitata

potenza di calcolo del Raspberry Pi 3 Model B+, il classificatore non riesce a riconoscere con precisione assoluta le presenze umane, tuttavia funziona comunque in modo più che adeguato ai fini di questo test.

Codice completo disponibile nell' Appendice B.

4.1.3 Esecuzione di un server Web

L'ultimo test di attività normale prevede che il Raspberry Pi 3 Model B+ assuma il ruolo di server web. A tale scopo sul dispositivo è stato installato un semplice server web, Lighttpd, sul quale è stata caricata una pagina contenente del testo e alcune immagini. Sul portatile è stato quindi avviato uno script che, ciclicamente, inviava una richiesta HTTP al server ogni secondo per scaricare la pagina.

4.2 Attività malevole

4.2.1 Subire un attacco bruteforce su SSH

Il primo test di attività malevola ha previsto il tentativo di forzare l'accesso al Raspberry Pi 3 Model B+ tramite un attacco brute force sul protocollo SSH. Questo tipo di attacco consiste nell'aprire ciclicamente connessioni SSH verso il dispositivo provando password diverse, con l'obiettivo di individuarne una corretta a forza di tentativi.

In questo test il portatile ha assunto il ruolo di attaccante, oltre che di monitor per l'esperimento, ed è stata usato lo strumento hydra per effettuare l'attacco. In particolare è stato eseguito un attacco a dizionario, usando una nota lista di password più frequenti: "rockyou.txt". Inoltre, per rendere l'attacco più efficace è stato usato il multithreading con 4 thread così da provare più combinazioni contemporaneamente.

Di seguito si può trovare il comando utilizzato:

```
1 sudo hydra -t 4 -w 60 -vV -l "cecchi" -P rockyou.txt 192.168.1.18 ssh
```

4.2.2 Effettuare attacco DoS tramite query DNS o richieste HTTP

I restanti due test di attività malevola sono stati progettati assumendo che il Raspberry Pi 3 Model B+ fosse già compromesso, ossia infetto da un malware in grado di eseguire task controllati da remoto. Lo scopo di questi test è stato analizzare il consumo energetico del dispositivo durante l'esecuzione di attacchi di tipo DoS.

Il primo scenario simulato consiste in un attacco DoS basato sull'invio massivo di richieste HTTP verso un server target.

Il secondo scenario riguarda invece un attacco di tipo DNS Amplification, in cui l'attaccante invia query a server DNS aperti falsificando l'indirizzo IP mittente (IP spoofing), sostituendolo con quello della vittima. In questo modo, i server DNS rispondono al dispositivo target con un elevato volume di risposte non richieste, generando un traffico amplificato che può saturarne la banda e le risorse, pur avendo l'attaccante iniziale inviato richieste di dimensioni molto contenute.

Durante entrambi i test, sul portatile è stato eseguito uno script che simulava il comportamento del server corrispondente, rispettivamente ascoltando richieste HTTP o DNS, in modo da ricevere e monitorare le richieste inviate dal Raspberry Pi 3 Model B+ durante l'esecuzione degli attacchi. Infine per eseguire l'attacco vero e proprio dal Raspberry Pi 3 Model B+ è stato usato il tool DNS-BOMBER impostato ad inviare 60 richieste al secondo in entrambi i casi, come in Figura 4.1.

Tool disponibile al seguente link: https://github.com/ciphersquid666/DNS-BOMBER/blob/main

Figura 4.1: Parametri usati negli attacchi HTTP e DNS con il tool DNS-BOMBER

4.3 Estrazione delle features

Al termine di ciascun test, i dati acquisiti tramite il software Otii 3 sono stati esportati in file formato CSV. Ogni test ha avuto una durata di 30 minuti e, considerando una frequenza di campionamento di 4000 campioni al secondo, ogni file contiene 7.200.000 campioni. Tali dati sono stati divisi in finestre di durata fissate a 1 secondo per il calcolo delle features, corrispondenti quindi a 4000 valori per finestra all'interno del file CSV.

Per l'estrazione delle features dai dati raccolti è stato utilizzato MATLAB. In particolare, per ogni esperimento il file CSV contenente le misure di corrente è stato importato come tabella, che è stata poi processata ciclicamente 4000 righe alla volta, analizzando cioè una finestra per volta.

Per ciascuna finestra sono state calcolate le seguenti features statistiche: media, deviazione standard, kurtosis, skewness, minimo, massimo, mediana, interquartile range (IQR) e root mean square (RMS). Successivamente, è stata costruita una nuova tabella, come da esempio in Tabella 4.1, nella quale ogni riga rappresenta una finestra e le colonne contengono le features sopra indicate, con una colonna aggiuntiva contenente un label che identifica il numero dell'esperimento analizzato. In questo modo, per ogni test sono state generate 1800 righe, ciascuna rappresentativa di una finestra di 1 secondo, pronta per le successive classificazioni.

Codice completo disponibile nell' Appendice C.

Tabella 4.1: Esempio tabella delle feature

Mean	**StdDev**	**Kurtosis**	**Skewness**	**Min**	**Max**	**Median**	**IQR**	**RMS**	**Activity**
0.2502	0.0098	42.8138	5.9013	0.2445	0.3410	0.2477	0.0027	0.2504	1
0.2508	0.0100	42.8912	5.7793	0.2443	0.3490	0.2479	0.0030	0.2510	1
0.2504	0.0105	48.6759	6.2592	0.2445	0.3542	0.2478	0.0032	0.2506	1
0.2531	0.0142	20.0177	3.8557	0.2445	0.3537	0.2484	0.0041	0.2535	1
0.2623	0.0334	12.9417	2.9498	0.2444	0.4659	0.2484	0.0074	0.2644	1
0.2508	0.0115	35.2953	5.3967	0.2448	0.3427	0.2478	0.0030	0.2511	1
0.2509	0.0122	48.8486	6.1875	0.2445	0.3816	0.2478	0.0030	0.2512	1
0.2505	0.0114	53.3760	6.5746	0.2446	0.3749	0.2477	0.0030	0.2507	1
0.2504	0.0106	42.4166	5.7961	0.2447	0.3506	0.2477	0.0030	0.2506	1
0.2634	0.0333	12.4956	2.8511	0.2451	0.4650	0.2491	0.0092	0.2655	1

Capitolo 5

Classificazione e Risultati

Per l'analisi è stato usato il Classification Learner di MATLAB, uno strumento che consente di eseguire la classificazione dei dati utilizzando diversi algoritmi di machine learning supervisionato. In particolare, è possibile testare contemporaneamente più tipologie di modelli, tra cui Decision Trees (alberi decisionali), Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Ensemble Methods (come Random Forest e Boosted Trees) e Reti Neurali (Neural Network). Lo strumento permette inoltre di valutare le prestazioni dei modelli tramite metriche standard quali confusion matrix, accuracy, precision e recall, facilitando così l'interpretazione e il confronto dei risultati ottenuti.

5.1 Scenario 1

L'obiettivo dello Scenario 1 è stato condurre un'analisi multiclasse, ossia addestrare i modelli di machine learning a identificare correttamente ciascuna delle sei attività considerate, con particolare attenzione alla capacità del modello di riconoscere le tre attività malevole.

Per utilizzare il Classification Learner di MATLAB, è stata preparata una tabella ottenuta concatenando le tabelle delle feature relative a tutte e sei le attività. Nella colonna delle label è stato inserito un numero da 1 a 6, corrispondente all'attività specifica, ottenendo così una tabella complessiva di 10.800 righe, conforme al formato richiesto dal tool.

Il codice utilizzato per la creazione della tabella è riportato nell'Appendice D.

La validazione dei modelli è stata effettuata su tutti gli algoritmi disponibili in Classification Learner, utilizzando lo schema di validazione 5-fold cross-validation. In questo schema, l'intero dataset viene suddiviso in cinque parti uguali, dette fold. Il modello viene addestrato cinque volte: in ciascuna iterazione, un fold diverso viene utilizzato come set di test, mentre i restanti quattro fold costituiscono il set di training. In questo modo, al termine dell'addestramento, ciascun fold è stato impiegato una volta per il testing e quattro volte per il training, garantendo una stima più affidabile delle prestazioni del modello.

In questo scenario numerosi modelli hanno mostrato prestazioni elevate nell'identificazione delle attività, raggiungendo in diversi casi un'accuratezza superiore al 99%. Tra questi, il modello che ha ottenuto i risultati migliori è stato l'Ensemble – Bagged Trees, che ha raggiunto un'accuratezza pari al 99,6%.

5.1.1 Ensemblee - Bagged Trees Scenario 1

Oltre al risultato numerico che indica l'accuratezza del modello è possibile avere un'interpretazione grafica del risultato tramite due strumenti, lo Scatter Plot e la Confusion Matrix.

Scatter Plot Scenario 1

Lo Scatter Plot è un grafico bidimensionale che rappresenta i dati in un piano cartesiano, ponendo sui due assi (X e Y) i valori di due feature, in questo caso la media e la deviazione standard. Questo tipo di rappresentazione è particolarmente utile per individuare pattern, relazioni tra variabili e l'eventuale presenza di anomalie nei dati. In particolare nel grafico i dati correttamente classificati dal modello vengono rappresentati con un "•", mentre i dati classificati in modo errato con una "x".

Dalla Figura 5.1 si osserva che le attività malevole (4, 5 e 6), se pur con qualche eccezione, tendono a concentrarsi in una regione ben delimitata del piano, risultando quindi chiaramente distinguibili e facilmente individuabili dal modello.

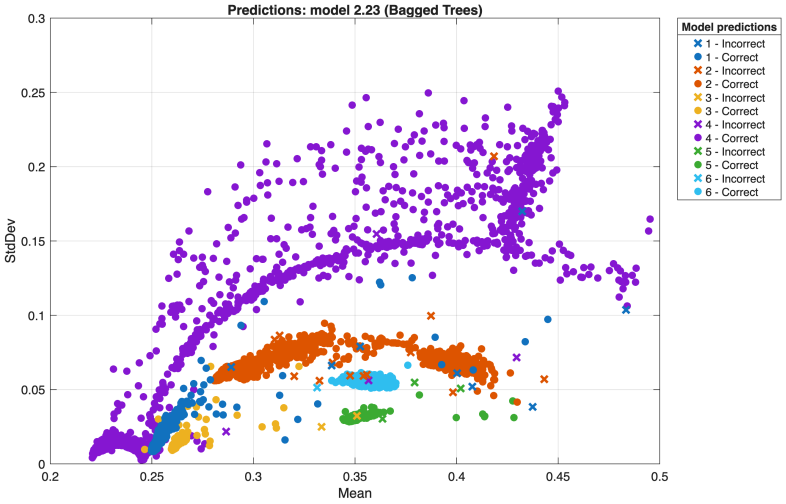

Figura 5.1: Scatter Plot Scenario 1

Confusion Matrix Scenario 1

La Confusion Matrix è una rappresentazione tabellare che confronta le etichette reali con quelle predette dal modello, consentendo di distinguere le predizioni corrette da quelle errate. I valori sulla diagonale principale indicano il numero di volte in cui una determinata classe è stata correttamente identificata, mentre i valori al di fuori della diagonale rappresentano le occorrenze in cui il modello ha commesso errori di classificazione.

Questo strumento risulta particolarmente utile poiché permette di valutare non solo l'efficacia complessiva del modello, ma anche la sua performance su ciascuna classe specifica, evidenziando eventuali confusioni tra classi. Inoltre, la Confusion Matrix costituisce una base fondamentale per il calcolo di metriche di valutazione chiave, quali accuracy, precision, recall e F1-score, quest'ultimo definito come la media armonica tra precision e recall.

Dalla Figura 5.2 si osserva che, salvo pochi errori, tutte le attività vengono correttamente riconosciute dal modello. Particolarmente rilevanti sono i casi relativi al quadrante in basso a sinistra, in cui attività malevole non vengono individuate e vengono classificate come normali (falsi negativi, FN), e al quadrante in alto a destra, in cui attività normali vengono erroneamente riconosciute come malevole (falsi positivi, FP). Tra queste due situazioni, quella dei falsi negativi rappresenta il rischio maggiore.

Tuttavia, come evidenziato dalla tabella, l'attività 4 non viene rilevata in 3 casi su 1800, l'attività 5 in 2 casi su 1800 e l'attività 6 in 5 casi su 1800. In conclusione, sebbene l'accuratezza complessiva del modello sia già elevata (99,6%), il dato più significativo è che, nel caso peggiore, un'attività malevola non viene rilevata solamente 5 volte su 1800, corrispondendo a una percentuale di rilevamento delle attività malevole (Recall) del 99,99%.

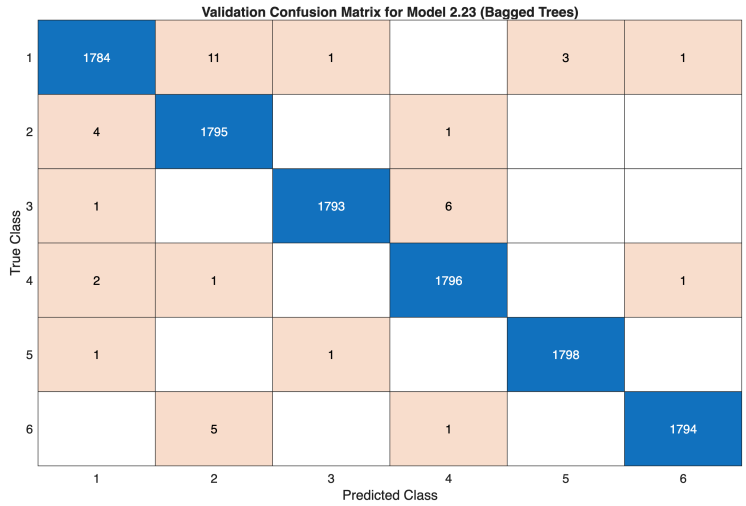

Figura 5.2: Confusion Matrix Scenario 1

5.2 Scenario 2

L'obiettivo dello Scenario 2 è stato quello di effettuare una classificazione binaria tra attività normali e attività malevole. A tal fine, le sei attività sono state raggruppate in due categorie soltanto: attività normali (attività 1, 2 e 3) e attività malevole (attività 4, 5 e 6). Analogamente a quanto fatto nello Scenario 1, è stata predisposta una tabella da utilizzare nel Classification Learner, contenente tutte e sei le attività con le rispettive feature e label; in questo caso, le label assumono solo due valori distinti (1 o 2) per distinguere tra attività normali e malevole.

Il codice utilizzato per la creazione della tabella è riportato nell'Appendice E.

Anche in questo scenario è stato utilizzato il Classification Learner, eseguendo la validazione su tutti gli algoritmi disponibili mediante uno schema di 5-fold cross-validation.

I risultati ottenuti sono stati nuovamente eccellenti: numerosi modelli hanno mostrato elevate prestazioni nella distinzione tra le due categorie di attività, raggiungendo un'accuratezza superiore al 99%. In particolare, il modello che ha conseguito le migliori prestazioni è stato l'Ensemble - Boosted Trees, con un'accuratezza pari al 99,7%.

5.2.1 Ensemblee - Boosted Trees Scenario 2

Scatter Plot Scenario 2

Dallo Scatter Plot dello Scenario 2, Figura 5.3, si osserva che le attività normali occupano una regione ben definita nel piano, con poche eccezioni rappresentate da alcuni punti leggermente spostati. Le attività malevole, invece, si distribuiscono su più regioni, pur rimanendo comunque distinte e riconoscibili. In generale, le aree occupate dai due gruppi risultano per lo più non sovrapposte e chiaramente separabili, il che consente al modello di distinguere efficacemente le due tipologie di attività e di classificare correttamente i dati.

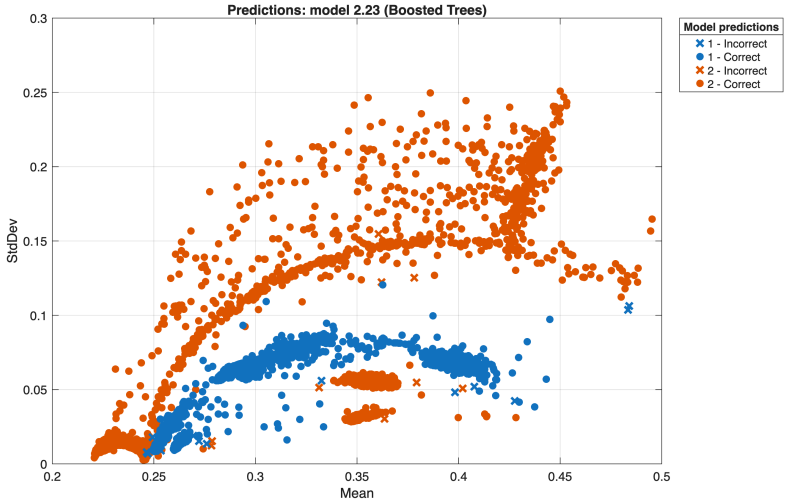

Figura 5.3: Scatter Plot Scenario 2

Confusion Matrix Scenario 2

Nella Confusion Matrix dello Scenario 2, Figura 5.4, risulta ancora più evidente il comportamento del classificatore nell'individuazione delle attività malevole. Si osserva infatti nella parte inferiore della matrice che sono state correttamente riconosciute come malevole 5385 istanze su un totale di 5400 (veri positivi, TP). Nei rimanenti 15 casi, invece, l'attività malevola non è stata rilevata (falsi negativi, FN). La parte superiore della matrice mostra, invece, come 5387 istanze siano state correttamente classificate come normali (veri negativi, TN), mentre in 13 casi l'attività normale è stata erroneamente interpretata come malevola (falsi positivi, FP).

Nonostante l'accuratezza complessiva del modello risulti già estremamente elevata (99,7%), è particolarmente rilevante osservare che l'errore più critico, ovvero la mancata rilevazione di un'attività malevola, si verifica soltanto in 15 casi su 5400. Ciò equivale a una capacità di identificazione delle attività malevole (Recall) pari al 99,99%, confermando l'elevata affidabilità del modello.

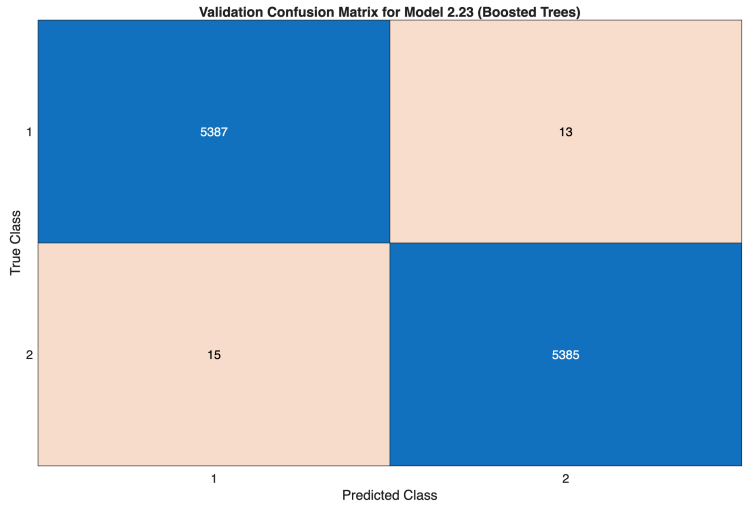

Figura 5.4: Confusion Matrix Scenario 2

5.3 Scenario 3

Scenario 3 ha perseguito il medesimo obiettivo dello scenario precedente, ovvero realizzare una classificazione binaria tra attività normali e attività malevole. In questo caso, tuttavia, si è voluto compiere un passo ulteriore: verificare se un algoritmo addestrato esclusivamente sui dati relativi ad attività normali fosse in grado, in fase di testing, di riconoscere come anomale le attività malevole. A tale scopo è stata adottata una tecnica di apprendimento automatico denominata Anomaly Detection, e in particolare l'algoritmo Isolation Forest disponibile in MATLAB.

Analogamente agli esperimenti precedenti, dai dati di consumo energetico sono state estratte le features relative alle sei attività, organizzandole in due tabelle: una contenente le attività normali e l'altra le attività malevole. A differenza degli scenari precedenti, tali tabelle non presentano etichette di classe, poiché non necessarie all'algoritmo impiegato in questo contesto.

L'algoritmo Isolation Forest è stato addestrato utilizzando la tabella delle sole attività normali, specificando un parametro denominato ContaminationFraction, che rappresenta la percentuale stimata di dati anomali presenti nel dataset di training. Dal momento che il dataset di addestramento era interamente composto da attività normali, il parametro è stato fissato a un valore molto basso (5%).

Successivamente, l'algoritmo è stato testato sulla tabella contenente le attività malevole, ottenendo un tasso di rilevamento delle anomalie di circa il 90%. Sebbene inferiore rispetto ai risultati conseguiti negli scenari precedenti, tale esito è in linea con le aspettative, considerando la maggiore complessità intrinseca di questo approccio.

Il codice utilizzato per questo scenario è riportato nell'Appendice F.

In questo scenario, il principale dubbio riguardava la scelta del valore del parametro Contamination Fraction. L'unica certezza iniziale era che, trattandosi di dati di test totalmente normali e privi di attività anomale, tale parametro dovesse assumere un valore basso. In seguito al test precedente, in cui il parametro è stato impostato arbitrariamente al 5%, è stato condotto un ulteriore esperimento per valutare come le prestazioni dell'algoritmo Isolation Forest variassero al variare di Contamination Fraction.

A tal fine, è stato implementato un algoritmo che replica il test precedente, eseguendo l'Isolation Forest con valori di Contamination Fraction compresi tra l'1% e il 20%, con step dell'1%. Per ciascun valore, sono state calcolate le metriche di Precision, Recall e F1-Score, quest'ultima combinando le due precedenti, e i risultati sono stati riportati in un grafico per facilitarne l'analisi comparativa.

$$\text{Precison} = \frac{\text{Attività Malevole Rilevate Correttamente}}{\text{Attivita Malevole Rilevate Correttamente} + \text{Attività Normali Classificate Come Malevole}}$$

$$\text{Recall} = \frac{\text{Attività Malevole Rilevate Correttamente}}{\text{Attivita Malevole Rilevate Correttamente} + \text{Attività Malevole Non Rilevate}}$$

$$F_1 Score = 2\,\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Dal grafico in Figura 5.5 si osserva che, all'aumentare del parametro Contamination Fraction, il Recall tende a crescere, indicando che tutte le attività malevole vengono rilevate. Tuttavia, contestualmente la Precision diminuisce, comportando un aumento dei cosiddetti "falsi allarmi", ossia attività normali erroneamente classificate come malevole.

Poiché l'obiettivo è identificare le attività malevole senza generare un numero eccessivo di falsi allarmi, il compromesso ottimale si ottiene con un valore di Contamination Fraction pari a 0,07, corrispondente al 7%.

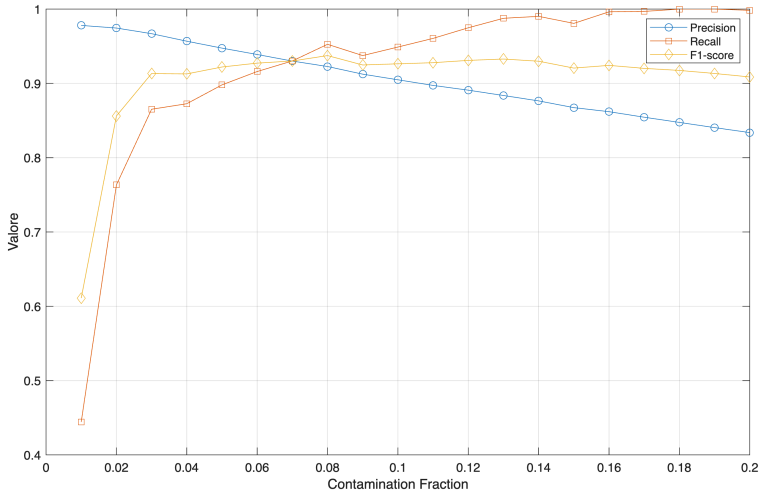

Figura 5.5: Metriche al variare della Contamination Fraction Scenario 3

Il codice utilizzato per questo test è riportato nell'Appendice G.

Capitolo 6

Conclusioni

I risultati ottenuti in questo studio confermano come l'analisi del consumo energetico rappresenti un approccio promettente per affrontare il problema della sicurezza informatica nei dispositivi IoT. Gli esperimenti condotti hanno infatti mostrato che numerosi algoritmi di machine learning sono in grado di distinguere attività malevole da attività lecite con un'accuratezza superiore al 99%.

Particolarmente significativo è il risultato relativo allo Scenario 3, che evidenzia la possibilità di ottenere prestazioni elevate anche addestrando i modelli esclusivamente su dati relativi ad attività normali. Tale evidenza è di fondamentale importanza in quanto apre la strada ad applicazioni reali: in un contesto operativo, infatti, sarebbe complesso disporre di dataset esaustivi riguardanti le molteplici tipologie di attacco, data la loro variabilità e continua evoluzione.

Nonostante l'efficacia dimostrata, emergono alcune criticità. La prima riguarda il numero di falsi positivi che, seppur ridotto, risulta comunque eccessivo in uno scenario di utilizzo reale, dove ogni rilevamento errato può comportare spreco di risorse o interruzioni indesiderate del servizio. La seconda riguarda la natura del dataset utilizzato: nonostante gli sforzi volti a bilanciare le diverse attività, esse rimangono in numero limitato e raccolte in un ambiente controllato. In un contesto operativo reale, più eterogeneo e rumoroso, la capacità di generalizzazione del modello potrebbe risultare ridotta.

In ottica di estensione, i successivi passi di questo studio potrebbero riguardare:

- ridurre ulteriormente il numero di falsi positivi, mantenendo al contempo un'elevata accuratezza complessiva;
- ampliare lo studio includendo un maggior numero di dispositivi e attività, sia normali che malevoli;
- analizzare il comportamento dei dispositivi in scenari misti, in cui attività legittime e malevole avvengono simultaneamente, al fine di valutare la capacità dei modelli di riconoscere situazioni più complesse e realistiche;
- costruire un prototipo hardware in grado di eseguire in tempo reale i modelli di classificazione addestrati, così da identificare automaticamente le attività e rilevare eventuali anomalie durante il normale funzionamento dei dispositivi IoT.

In conclusione, questo lavoro contribuisce a consolidare l'evidenza che l'analisi del consumo energetico può rappresentare una strategia efficace e non invasiva per il rilevamento di anomalie nei dispositivi IoT. Pur trattandosi di un approccio ancora da validare su larga scala, i risultati ottenuti forniscono un solido punto di partenza per lo sviluppo di soluzioni pratiche e innovative in ambito di sicurezza informatica, capaci di rispondere a una delle sfide più urgenti poste dalla crescente diffusione dell'Internet of Things.

Appendice A

Codice usato per la simulazione di un sensore.

```
#!/bin/bash

SERVER_IP="MacBook-Pro-di-Giorgio.local"
SERVER_PORT=5050
INTERVAL=5

while true; do
    TEMP=$(vcgencmd measure_temp | cut -d "=" -f2 | tr -d "'C")
    VOLT=$(vcgencmd measure_volts core | cut -d "=" -f2 | tr -d "V")
    FREQ=$(vcgencmd measure_clock arm | awk -F= '{print $2}')
    THROTTLED=$(vcgencmd get_throttled | cut -d "=" -f2)

    TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")

    PAYLOAD="{\"time\":\"$TIMESTAMP\",\"temp\":$TEMP,\"volt\":$VOLT,\"freq\":$FREQ
    ,\"throttled\":\"$THROTTLED\"}"

    # invio via HTTP POST
    curl -X POST -H "Content-Type: application/json" -d "$PAYLOAD" http://
    $SERVER_IP:$SERVER_PORT/

    sleep $INTERVAL
done
```

Appendice B

Codice usato per la simulazione di una telecamera.

```python
import cv2
import time
import requests
from datetime import datetime

cascade = cv2.CascadeClassifier("haarcascade_upperbody.xml")

cap = cv2.VideoCapture("videoplayback.mp4")
fps = cap.get(cv2.CAP_PROP_FPS)
if fps == 0:
    fps = 25

frame_duration = 1 / fps
analisi_interval_video_s = 5
prossima_analisi_s = 0

start_time = time.time()

frame_index = 0

while True:
    ret, frame = cap.read()
    if not ret:
        break

    video_time_s = cap.get(cv2.CAP_PROP_POS_MSEC) / 1000.0
    real_time_s = time.time() - start_time
    delta = video_time_s - real_time_s



    # Analisi ogni 5s
    if video_time_s >= prossima_analisi_s:
        prossima_analisi_s += analisi_interval_video_s

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        bodies = cascade.detectMultiScale(gray, 1.1, 4)



        if len(bodies) > 0:
            ora = datetime.now().strftime("%H:%M:%S")
            #print(f" Persona rilevata alle {ora} (tempo video: {video_time_s:.2f}s
    )")
            try:
                requests.post("http://MacBook-Pro-di-Giorgio.local:5050/", json={
                    "evento": "persona_rilevata",
                    "timestamp": ora
                })
            except Exception as e:
                print("Errore notifica:", e)
```

```
51          #else:
52              #print(f"Nessuno (tempo video: {video_time_s:.2f}s)")
53
54      # Sleep dinamico per sincronizzarsi al tempo reale
55      next_frame_time = start_time + (frame_index + 1) * frame_duration
56      sleep_duration = next_frame_time - time.time()
57      if sleep_duration > 0:
58          time.sleep(sleep_duration)
59
60      frame_index += 1
61
62  cap.release()
63  cv2.destroyAllWindows()
```

Appendice C

Codice usato per l'estrazione delle features.

```
function T = estraiFeature(filepath, activity_code, fs)

    data = readmatrix(filepath);
    signal = data(:,2);

    window_size = fs;
    num_windows = floor(length(signal) / window_size);

    features = zeros(num_windows, 7);
    labels = repmat(activity_code, num_windows, 1);

    for i = 1:num_windows
        idx_start = (i-1)*window_size + 1;
        idx_end = i*window_size;
        window = signal(idx_start:idx_end);
        features(i,1) = mean(window);
        features(i,2) = std(window);
        features(i,3) = kurtosis(window);
        features(i,4) = skewness(window);
        features(i,5) = min(window);
        features(i,6) = max(window);
        features(i,7) = median(window);
        features(i,8) = iqr(window);
        features(i,9) = rms(window);
    end

    T = array2table(features, 'VariableNames', ...
        {'Mean','StdDev','Kurtosis','Skewness','Min','Max','Median','IQR','RMS'});
    T.Activity = labels;
end
```

Appendice D

Codice usato per la creazione della tabella delle features di Scenario 1.

```
fs = 4000;
data_path = '/Users/giorgiocecchi/Desktop/Tesi/dati_final';
current_path = fileparts(mfilename('fullpath'));

esperimenti = {
    '1-sensor', 1;
    '2-cam',    2;
    '3-serverweb',       3;
    '4-bruteforceSSH_T4',   4;
    '5-DoS_HTTP',   5;
    '5-DoS_DNS',  6;
};

T_finale = table();

for i = 1:size(esperimenti,1)
    folder = esperimenti{i,1};
    code = esperimenti{i,2};

    filepath = fullfile(data_path, folder, 'Main current - Arc.csv');

    T = estraiFeature_1(filepath, code, fs);

    T_finale = [T_finale; T];
end

writetable(T_finale, fullfile(current_path, 'features_corrente_finale.csv'));

disp('Estrazione completata.');
```

Appendice E

Codice usato per la creazione della tabella delle features di Scenario 2.

```
fs = 4000;
data_path = '/Users/giorgiocecchi/Desktop/Tesi/dati_final';
current_path = fileparts(mfilename('fullpath'));

esperimenti = {
    '1-sensor', 1;
    '2-cam',    1;
    '3-serverweb',       1;
    '4-bruteforceSSH_T4',   2;
    '5-DoS_HTTP',   2;
    '5-DoS_DNS',  2;
};

T_finale = table();

for i = 1:size(esperimenti,1)
    folder = esperimenti{i,1};
    code = esperimenti{i,2};

    filepath = fullfile(data_path, folder, 'Main current - Arc.csv');

    T = estraiFeature_2(filepath, code, fs);

    T_finale = [T_finale; T];
end

writetable(T_finale, fullfile(current_path, 'features_corrente_finale.csv'));

disp('Estrazione completata.');
```

Appendice F

Codice usato per lo Scenario 3.

```
fs = 4000;
data_path = '/Users/giorgiocecchi/Desktop/Tesi/dati_final';
current_path = fileparts(mfilename('fullpath'));

esperimenti_normali = {
    '1-sensor', 1;
    '2-cam',    2;
    '3-serverweb',       3;
};

esperimenti_malevoli = {
    '4-bruteforceSSH_T4',   4;
    '5-DoS_HTTP',   5;
    '5-DoS_DNS',  6;
};

normalData = table();
maliciousData = table();

for i = 1:size(esperimenti_normali,1)
    folder = esperimenti_normali{i,1};
    code = esperimenti_normali{i,2};

    filepath = fullfile(data_path, folder, 'Main current - Arc.csv');

    T = estraiFeature_3(filepath, code, fs);

    normalData = [normalData; T];
end

writetable(normalData, fullfile(current_path, 'features_corrente_finale_normal.csv'
    ));

for i = 1:size(esperimenti_malevoli,1)
    folder = esperimenti_malevoli{i,1};
    code = esperimenti_malevoli{i,2};

    filepath = fullfile(data_path, folder, 'Main current - Arc.csv');

    T = estraiFeature_3(filepath, code, fs);

    maliciousData = [maliciousData; T];
end

writetable(maliciousData, fullfile(current_path, '
    features_corrente_finale_malicious.csv'));

disp('Estrazione completata.');

% TRAIN
[forest, tfTrain, scoresTrain] = iforest(normalData, ContaminationFraction=0.05);
```

```
50
51 % TEST
52 [tfMal, scoresMal] = isanomaly(forest, maliciousData);
53
54 % RISULTATI
55 detectionRate = mean(tfMal) * 100;
56 fprintf('Rilevati come anomalie: %.2f%%\n', detectionRate);
```

Appendice G

Codice usato per il test su Contamination Fraction di Scenario 3.

```
fs = 4000;
data_path = '/Users/giorgiocecchi/Desktop/Tesi/dati_final';
current_path = fileparts(mfilename('fullpath'));

esperimenti_normali = {
    '1-sensor', 1;
    '2-cam',    2;
    '3-serverweb',       3;
};

esperimenti_malevoli = {
    '4-bruteforceSSH_T4',   4;
    '5-DoS_HTTP',   5;
    '5-DoS_DNS',  6;
};

normalData = table();
maliciousData = table();

for i = 1:size(esperimenti_normali,1)
    folder = esperimenti_normali{i,1};
    code = esperimenti_normali{i,2};

    filepath = fullfile(data_path, folder, 'Main current - Arc.csv');

    T = estraiFeature_3(filepath, code, fs);

    normalData = [normalData; T];
end

writetable(normalData, fullfile(current_path, 'features_corrente_finale_normal.csv'
    ));

for i = 1:size(esperimenti_malevoli,1)
    folder = esperimenti_malevoli{i,1};
    code = esperimenti_malevoli{i,2};

    filepath = fullfile(data_path, folder, 'Main current - Arc.csv');

    T = estraiFeature_3(filepath, code, fs);

    maliciousData = [maliciousData; T];
end

writetable(maliciousData, fullfile(current_path, '
    features_corrente_finale_malicious.csv'));

disp('Estrazione completata.');

% Range di contamination fraction da testare
contFractions = 0.01:0.01:0.20;
```

```
50
51 precisionVals = zeros(size(contFractions));
52 recallVals    = zeros(size(contFractions));
53 f1Vals        = zeros(size(contFractions));
54
55 for i = 1:numel(contFractions)
56
57     cf = contFractions(i);
58
59     forest = iforest(normalData, ContaminationFraction=cf);
60
61     [isAnomNorm, ~] = isanomaly(forest, normalData);
62
63     [isAnomMal, ~] = isanomaly(forest, maliciousData);
64
65     TP = sum(isAnomMal == 1);   % malevoli correttamente rilevati
66     FN = sum(isAnomMal == 0);   % malevoli persi
67     FP = sum(isAnomNorm == 1);  % normali scambiati per malevoli
68     TN = sum(isAnomNorm == 0);  % normali corretti
69
70     precision = TP / (TP + FP + eps);
71     recall    = TP / (TP + FN + eps);
72     f1        = 2 * (precision * recall) / (precision + recall + eps);
73
74     precisionVals(i) = precision;
75     recallVals(i)    = recall;
76     f1Vals(i)        = f1;
77 end
78
79 figure;
80 plot(contFractions, precisionVals, '-o'); hold on;
81 plot(contFractions, recallVals, '-s');
82 plot(contFractions, f1Vals, '-d');
83 xlabel('Contamination Fraction');
84 ylabel('Valore');
85 legend('Precision','Recall','F1-score');
86 grid on;
```

Bibliografia

[1] K. Nimmy, M. Dilraj, S. Sankaran, and K. Achuthan. Leveraging power consumption for anomaly detection on iot devices in smart homes. *Journal of Ambient Intelligence and Humanized Computing*, 14:14045–14056, 2023.

[2] AKM Jahangir Majumder, Jared D. Miller, Charles B. Veilleux, and Amir A. Asif. Smartpower: A smart cyber-physical system to detect iot security threat through behavioral power profiling. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1041–1049, 2020.

[3] M. Ge, J. Hong, W. Guttmann, and D. Kim. Energy audition based cyber-physical attack detection system for iot devices. In *Proceedings of the ACM Turing Celebration Conference - China (ACM TURC '19)*, 2019.

[4] Dominic Lightbody, Duc-Minh Ngo, Andriy Temko, Colin C. Murphy, and Emanuel Popovici. Attacks on iot: Side-channel power acquisition framework for intrusion detection. *Future Internet*, 15(5):187, 2023.

[5] Shane S. Clark, Benjamin Ransford, Amir Rahmati, Shane Guineau, Jacob Sorber, Wenyuan Xu, and Kevin Fu. WattsUpDoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In *2013 USENIX Workshop on Health Information Technologies (HealthTech '13)*, Washington, D.C., August 2013. USENIX Association.

[6] Raspberry Pi Foundation. Raspberry pi 3 model b+. https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/, 2025.

[7] Qoitech AB. Otii arc pro. https://www.qoitech.com/otii-arc-pro/, 2024.

[8] Qoitech AB. Otii software. https://www.qoitech.com/software/, 2025.