Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in
Ingegneria Informatica, Elettronica e delle Comunicazioni

Elaborato finale

Cyber Resilience Act: progettazione di un framework di verifica dei requisiti ed applicazione ad un caso reale

Supervisore
Prof. Bruno Crispo

Co-supervisore
Dott. Carlo Ramponi

Laureando
Jago Revrenna

Anno accademico 2024/2025

Indice

Sommario

La recente diffusione capillare dei dispositivi con elementi digitali, come i dispositivi IoT (Internet of Things), ha prodotto la necessità di garantire sicurezza e affidabilità lungo il loro intero ciclo di vita. In particolare, il Cyber Resilience Act (CRA), Regolamento (UE) 2024/2847 [31], introduce requisiti essenziali di cibersicurezza per i prodotti immessi sul mercato europeo.
Questa tesi ha come scopo la trasformazione di indicazioni normative in pratiche di verifica concrete, ripetibili e utili a chi progetta, valuta o utilizza dispositivi con elementi digitali.

L'obiettivo della tesi è duplice. Da un lato viene proposto un framework operativo che rende testabili alcuni requisiti del CRA: questi vengono analizzati, tecnicizzati e per alcuni di essi viene proposta una procedura pratica di test. Dall'altro, il framework viene validato su un caso reale, scelto tra i dispositivi consumer più diffusi, per verificarne la praticabilità e mettere in evidenza eventuali criticità. Il tutto avviene in un ambiente controllato ed impiegando strumenti open source per l'osservazione del traffico, la valutazione delle interfacce, l'analisi della procedura di aggiornamento, l'analisi firmware e la verifica della resilienza rispetto a scenari d'attacco realistici.

Il contributo personale consiste nella costruzione della mappatura requisiti–metodi, nella progettazione dell'ambiente di test e nell'esecuzione dei test stessi, oltre all'analisi critica degli esiti ottenuti. Il framework proposto si è dimostrato utile nella guida a una valutazione strutturata, dimostrando versatilità.

I Introduzione

Questa tesi nasce dall'esigenza di dare concretezza al Cyber Resilience Act [31], offrendo un insieme di metodi ripetibili ed applicabili ad un'ampia varietà di dispositivi. L'idea di fondo è che una norma è efficace quando può essere tradotta in pratiche operative chiare, che possano essere testate in modo cristallino.

La prima parte introduce il quadro di riferimento: motivazioni, obiettivi, elementi chiave e struttura del CRA. A partire da queste basi vengono definiti gli obiettivi specifici della tesi e il metodo di lavoro. Il risultato di questa prima parte è una tabella di mappatura dei requisiti tecnici (allegato A). Ad alcuni di essi viene associato un caso di test sintetico con i relativi passi essenziali per l'esecuzione. L'intento è passare da principi generali a verifiche che possano essere eseguite e documentate in modo tecnico.

La seconda parte mette alla prova il framework su una videocamera IP di uso domestico (Wyze Cam v3, descritta al capitolo 3). Dopo una breve descrizione del dispositivo e del setup, il testing è organizzato per aree tematiche in linea con il CRA: osservazione del traffico di rete, verifica delle porte aperte, analisi del comportamento durante gli aggiornamenti, valutazione del comportamento in caso di attacchi, prove di intercettazione del canale di comunicazione ed analisi del firmware. L'obiettivo non è esaurire ogni possibile tecnica, ma mostrare come il framework consenta di condurre verifiche ordinate, comparabili e ripetibili.

Le conclusioni raccolgono gli esiti del caso di studio, discutendone i punti di forza e i limiti dell'approccio proposto. Si auspica che il percorso individuato possa essere utile come base di lavoro per chi deve valutare la conformità di determinati dispositivi alle normative proposte dal CRA o, più in generale, desidera impostare verifiche di sicurezza su dispositivi in modo sistematico.

II Cyber Resilience Act (CRA)

1 Introduzione e contesto

Il Regolamento (UE) 2024/2847, noto come **Cyber Resilience Act (CRA)** [31], si propone di affrontare in modo sistematico le crescenti minacce alla sicurezza informatica, derivanti dalla diffusione di dispositivi connessi e di software vulnerabili.
Il CRA introduce un insieme di requisiti essenziali di cibersicurezza per tutti i prodotti con elementi digitali immessi sul mercato europeo, con l'obiettivo di migliorarne la sicurezza lungo l'intero ciclo di vita, dalla progettazione fino alla dismissione.

1.1 Motivazioni e obiettivi del CRA

Lo scopo è rispondere a una crescente preoccupazione per la sicurezza informatica, data l'esponenziale crescita dei dispositivi connessi e l'aumento degli attacchi ad essi, che comportano conseguenze gravi in termini economici, sociali e di sicurezza per imprese e cittadini.

Il CRA si impone di:

- rafforzare la ciberresilienza dei prodotti digitali
- garantire un livello minimo di sicurezza per tutti i dispositivi connessi
- migliorare la trasparenza lungo la filiera produttiva
- facilitare la conformità normativa da parte dei produttori

2 Elementi chiave del regolamento

I punti principali sono:

- Sicurezza dei prodotti: i dispositivi devono essere progettati per ridurre i rischi informatici fin dalla fase di sviluppo e garantire che restino sicuri durante tutto il loro ciclo di vita.
- Gestione delle vulnerabilità: i produttori sono obbligati a monitorare i loro prodotti per rilevare eventuali vulnerabilità e a fornire aggiornamenti (patch) per correggerle, anche dopo la vendita.
- Trasparenza: sono richieste informazioni dettagliate sui componenti del prodotto (tramite la distinta base del software, SBOM, descritta al capitolo 3.2).
- Certificazione e controllo: i prodotti devono essere certificati per garantire che soddisfino gli standard di sicurezza. Le autorità competenti sorvegliano il rispetto delle norme, non rilasciando certificazioni in caso di non conformità.

2.1 Struttura del regolamento

Il Cyber Resilience Act è articolato in due componenti principali:

- Una parte normativa, composta da articoli suddivisi in capi (titoli), che definisce gli obblighi legali, i soggetti coinvolti, le procedure di conformità ecc.
- Una parte tecnica, costituita da allegati, con l'elenco dei requisiti concreti che i prodotti devono rispettare per essere conformi al CRA.

La parte tecnica è composta da allegati, ai quali ci si riferirà in questo modo:
Allegato X, Parte X, Punto X (ad esempio *Allegato I, Parte 1, Punto 2*).

Ad esempio, l'Allegato I specifica i requisiti essenziali di cibersicurezza ed è suddiviso in due parti:

- Parte I – Requisiti relativi al prodotto: include requisiti che riguardano direttamente il design, lo sviluppo e la produzione sicura del dispositivo. I punti coprono aspetti come configurazione sicura, gestione degli accessi, integrità e riservatezza dei dati, resilienza agli attacchi e logging delle attività.
- Parte II – Requisiti di gestione delle vulnerabilità: stabilisce obblighi specifici per i fabbricanti relativi alla gestione delle vulnerabilità, come la redazione della distinta base del software (SBOM), la divulgazione responsabile delle vulnerabilità stesse, la divulgazione di aggiornamenti di sicurezza e la comunicazione efficace agli utilizzatori.

2.2 Obiettivi

L'obiettivo è definire e sviluppare procedure operative per la verifica della conformità dei dispositivi elettronici ai requisiti previsti dal CRA.
Sono stati analizzati gli allegati presenti nel documento e sono stati tecnicizzati i più interessanti.

Si prevede di:

- progettare un framework di verifica a livello procedurale e di processo
- definire metodi di test ripetibili, tracciabili e standardizzati
- validare il framework attraverso l'analisi di un caso concreto: la videocamera Wyze v3 (capitolo 3)

2.3 Metodo di lavoro

Poiché è necessario avere una struttura solida per affrontare la fase sperimentale, è stata sviluppata una tabella (allegato A) che raccoglie i requisiti tecnici del CRA. In particolare, la sua struttura è la seguente:

- **Allegato / Parte / Punto**: identificativo normativo del requisito.
- **Requisito**: descrizione testuale riassunta direttamente dal CRA.
- **Metodo di implementazione**: proposta di strategia tecnica o procedurale per la verifica.
- **Nello specifico**: descrizione operativa concreta e strumenti suggeriti.

III Device Testing

3 Wyze Cam v3

3.1 Descrizione

Questo capitolo descrive la Wyze Cam v3 [39], mostrata in Figura 3.1. Si tratta di una videocamera IP di fascia consumer, prodotta da Wyze Labs e pensata per il monitoraggio domestico/di piccoli spazi esterni. Essa rappresenta bene la categoria grazie alla sua diffusione e al suo prezzo relativamente contenuto (dai 30/40€ nei mercati statunitensi e cinesi agli 80/90€ dei mercati europei). Inoltre la Wyze Cam v3 offre un insieme variegato di funzioni e servizi, andando a costituire quindi un buon candidato per questo caso di studio.

Il dispositivo registra in 1080p, rileva movimento e suono ed integra una piccola sirena. È presente uno slot per microSD, utile per la registrazione locale continua o su evento. In alternativa, ci si può affidare ai servizi cloud del produttore su abbonamento (dai 2€ ai 30€ al mese). L'involucro è compatto e resistente agli agenti atmosferici (grado IP65). L'alimentazione avviene tramite USB a 5 V e la connettività è esclusivamente Wi-Fi a 2.4 GHz. Non è prevista una porta Ethernet o USB per la trasmissione di dati.

La gestione avviene dall'app mobile ufficiale (Wyze App [38]), disponibile per Android e iOS, mentre non è presente una classica interfaccia web. Sulla base della videocamera si trovano le informazioni identificative: modello, indirizzo MAC e numero di serie, oltre alle marcature regolamentari. Microfono e altoparlante sono integrati nel corpo del dispositivo e permettono il dialogo in tempo reale, mentre un LED indica lo stato della videocamera ed un pulsante di setup consente di interagire fisicamente con essa (per la prima configurazione).

La fruizione dello streaming avviene tramite l'app mobile. Il supporto a protocolli come RTSP [24] **non** è abilitato di default: dipende dalla versione del firmware installato. Questa caratteristica, insieme alla registrazione su microSD e all'integrazione con i servizi cloud, rende la Wyze Cam v3 interessante anche dal punto di vista della varietà di interfacce e servizi esposti.

In sintesi, si tratta di una videocamera economica ma completa. Le sue caratteristiche permettono di osservare diversi comportamenti rilevanti per la sicurezza, che verranno analizzati nei capitoli successivi.

Figura 3.1: Wyze Cam v3

3.2 SBOM

La **SBOM** (Software Bill of Materials) è l'inventario dettagliato di tutti i componenti software presenti in un prodotto con elementi digitali: librerie, framework, driver, pacchetti e dipendenze varie. Una SBOM tipica include, per ogni componente, informazioni quali nome, versione, fornitore ed identificatori standard per il riconoscimento nei database di vulnerabilità (ad esempio CVE di MITRE [28]).

Per la Wyze Cam v3 non è disponibile una SBOM pubblica. Wyze Labs (il produttore) non ha rilasciato questo tipo di documentazione attraverso i canali ufficiali. Le sole informazioni tecniche accessibili riguardano la certificazione FCC (ID 2AUIUWYZEC3F) [23]. Nella "Confidentiality Letter" presente sul portale FCC, il fabbricante ha richiesto un trattamento confidenziale per schematics, block diagrams e operational descriptions: queste informazioni tecniche, potenzialmente correlate ai componenti software, non sono quindi pubblicamente accessibili. Sono invece disponibili le certificazioni e i risultati dei test di conformità radio e conformità elettromagnetica, che attestano il rispetto delle relative normative.

L'assenza di una SBOM pubblica limita significativamente la valutazione della sicurezza: molti requisiti del CRA prevedono di basarsi sulla SBOM, con lo scopo di ottenere informazioni sui componenti presenti nel software del dispositivo.

3.3 Setup

La configurazione della Wyze Cam v3 avviene esclusivamente tramite l'applicazione mobile ufficiale menzionata sopra. Il processo di setup della videocamera richiede:

1. la creazione di un account Wyze (o il login ad esso)
2. il login del dispositivo alla rete Wi-Fi domestica
3. l'associazione del dispositivo all'account utente

Durante la fase iniziale della procedura guidata, l'applicazione richiede di premere il pulsante di setup (alla base della videocamera). Viene attivata quindi una modalità di configurazione in cui la videocamera si prepara a ricevere le credenziali di rete. Il trasferimento delle credenziali avviene attraverso un meccanismo basato su QR code: l'applicazione genera un codice QR, contenente le credenziali Wi-Fi criptate, che viene inquadrato dalla videocamera tramite il suo sensore ottico. Questo approccio elimina la necessità di interfacce web locali o procedure di associazione Bluetooth.

Una volta completato il riconoscimento del QR code, la videocamera si connette alla rete Wi-Fi indicata e stabilisce una comunicazione con i server Wyze per completare l'attivazione. Durante questa fase vengono inoltre proposti eventuali aggiornamenti firmware. La procedura prevede anche la configurazione di parametri base, quali il nome del dispositivo e le preferenze di notifica.

Per l'analisi descritta nei capitoli successivi è stato predisposto un ambiente di test controllato. La videocamera è stata configurata per connettersi a un hotspot Wi-Fi generato direttamente dal computer di analisi (con sistema operativo macOS), che funge contemporaneamente da access point e da gateway verso Internet. Questa configurazione permette di intercettare e analizzare tutto il traffico di rete generato dal dispositivo, mantenendo comunque la connettività ad internet (necessaria per il corretto funzionamento dei servizi offerti dalla videocamera).

L'hotspot è stato configurato con crittografia WPA2 ed opera sulla banda 2.4 GHz, unica frequenza supportata dalla Wyze Cam v3. Il computer host, collegato a Internet tramite connessione cablata, condivide la connettività attraverso il bridge di rete. In questo modo si è potuto osservare il comportamento del dispositivo in condizioni realistiche, pur mantenendo il controllo completo del traffico di rete.

Si è scelto di utilizzare un hotspot anziché una rete domestica già esistente per la necessità di isolare il traffico del dispositivo in questione: in questo modo possono essere applicati strumenti di intercettazione (sniffing) senza interferenze da parte di altri dispositivi connessi.

4 Packet sniffing

4.1 Analisi di pacchetti

Per procedere con l'intercettazione dei pacchetti è sufficiente collegare la videocamera all'hotspot precedentemente configurato ed utilizzare un network protocol analyzer. In questo caso è stato utilizzato Wireshark [33], selezionando l'interfaccia `ap1`, che corrisponde all'hotspot creato dal sistema operativo macOS.

È importante specificare che in ambiente macOS l'interfaccia `ap1` e `bridge100` sono sostanzialmente sinonimi dal punto di vista funzionale. Il dispositivo `bridge100` è di fatto un'interfaccia virtuale che il sistema operativo utilizza internamente, al fine di creare un ponte tra l'interfaccia di rete principale, che fornisce la connettività Internet, e quella dedicata all'hotspot (`ap1`). Tuttavia, parlando di analisi del traffico, `bridge100` non consente l'accesso diretto ai pacchetti di rete a livello hardware, necessario per strumenti di analisi come `nmap` [13] o Wireshark. `ap1` è quindi l'interfaccia di riferimento per l'intercettazione di pacchetti.

È inoltre indispensabile assicurarsi che la videocamera riceva un indirizzo IP dal servizio DHCP (Dynamic Host Configuration Protocol) dell'hotspot `ap1`. Il controllo dei dispositivi connessi all'hotspot può essere effettuato consultando il file di sistema `/var/db/dhcpd_leases`, dove l'apposito daemon registra i lease assegnati durante l'utilizzo della funzione Internet Sharing di macOS.

Il file appena menzionato rivela informazioni dettagliate sui dispositivi connessi all'hotspot:

```
sudo cat /var/db/dhcpd_leases

{
        name=WYZE_CAKP2JFUS-80482C413C5E
        ip_address=192.168.2.2
        hw_address=1,80:48:2c:41:3c:5e
        identifier=1,80:48:2c:41:3c:5e
        lease=0x6886312c
}
```

Nel caso specifico della Wyze Cam v3 utilizzata per l'analisi, il sistema ha registrato il dispositivo con:

- nome: `WYZE_CAKP2JFUS-80482C413C5E`
- indirizzo IP: `192.168.2.2`
- MAC address: `80:48:2c:41:3c:5e`

Il dispositivo è quindi correttamente isolato nell'ambiente di test.

4.2 Protocolli utilizzati

L'analisi del traffico di rete ha mostrato come vengano utilizzati diversi protocolli, ciascuno con finalità specifiche per il corretto funzionamento della parte di comunicazione della videocamera.

4.2.1 Protocolli di configurazione e monitoraggio

Al momento della connessione alla rete, la videocamera utilizza il protocollo DHCP (Dynamic Host Configuration Protocol) per ottenere automaticamente un indirizzo IP. Nel caso specifico dell'ambiente di test, il servizio DHCP dell'hotspot ha assegnato al dispositivo l'indirizzo IP `192.168.2.2`.

Immediatamente dopo l'ottenimento dell'indirizzo IP, il dispositivo genera una serie di query DNS per la risoluzione dei nomi di dominio dei servizi necessari al funzionamento. Le query osservate includono domini come `api.wyzecam.com`, `google.com`, `clock.fmt.he.net` e `wyze-general-api.wyzecam.com`. Queste richieste vengono trasmesse in chiaro mediante protocollo UDP verso la porta 53, rendendo facilmente identificabili i servizi esterni con cui la videocamera intende stabilire comunicazioni.

La videocamera implementa diversi meccanismi di monitoraggio dello stato della connessione. Sono presenti pacchetti ARP (Address Resolution Protocol) in chiaro, attraverso i quali il dispositivo mantiene aggiornata la propria tabella di associazione tra indirizzi IP e MAC address. Contemporaneamente vengono generati pacchetti ICMP (Internet Control Message Protocol) di tipo `echo request` diretti verso il gateway (`192.168.2.1`), allo scopo di verificare la continuità della connessione di rete.

La sincronizzazione del tempo cronologico viene gestita mediante il protocollo NTP (Network Time Protocol), con comunicazioni dirette verso server pubblici di riferimento temporale. Nell'ambiente di test analizzato la videocamera ha stabilito comunicazioni con `clock.fmt.he.net` (IP: `216.218.192.202`).

4.2.2 Comunicazioni sicure e crittografia

Per tutte le comunicazioni con i servizi Wyze la videocamera implementa il protocollo TLS (Transport Layer Security) al fine di garantire l'integrità dei dati all'interno del canale di comunicazione.

L'handshake TLS (figura 4.1) osservato segue la procedura standard, iniziando con il `Client Hello` in cui il dispositivo propone le versioni supportate del protocollo (TLS 1.2 e TLS 1.3) insieme a un set di 31 diverse cipher suites disponibili. Il messaggio include inoltre il Server Name Indication (SNI) che specifica esplicitamente il dominio di destinazione.

Il server risponde con il `Server Hello`, selezionando la versione del protocollo TLS e la cipher suite più appropriata tra quelle proposte. Nelle sessioni analizzate è stata frequentemente osservata la selezione di `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`: una configurazione crittografica robusta, che combina lo scambio di chiavi basato su Elliptic Curve Diffie-Hellman Ephemeral [41] (le chiavi generate sono temporanee per ogni sessione), autenticazione mediante RSA e cifratura AES a 128 bit in modalità GCM (fornisce sia cifratura che autenticazione in un'unica operazione).

La procedura prosegue con l'invio del certificato del server, firmato da un'autorità di certificazione riconosciuta (CA). In seguito il `Server Key Exchange` stabilisce un segreto condiviso. I parametri trasmessi includono il tipo di curva ellittica, il nome specifico della curva utilizzata, l'algoritmo di firma e la chiave pubblica del server, necessaria per stabilire un segreto condiviso e completare lo scambio.

Il `Client Key Exchange` completa la fase di negoziazione delle chiavi, permettendo a entrambe le parti di derivare un segreto simmetrico condiviso. I messaggi di `Change Cipher Spec`, trasmessi in entrambe le direzioni, servono a segnalare l'inizio dell'utilizzo della cifratura simmetrica per tutti i dati successivi. L'handshake si conclude con i messaggi `Encrypted Handshake Message (Finished)`, che verificano l'integrità dell'intera procedura di negoziazione.

Una volta completato l'handshake, tutte le comunicazioni viaggiano attraverso HTTPS sulla porta 443, utilizzando il canale crittografato appena stabilito. Questo approccio garantisce tre proprietà fondamentali di sicurezza: **confidentiality** attraverso la cifratura dei dati trasmessi, **integrity** mediante checksum e hash che prevengono modifiche non autorizzate, **autentication** tramite il certificato digitale che assicura l'identità del server.

4.2.3 Streaming video

L'analisi del traffico di streaming video ha rivelato come vengano utilizzati protocolli differenti dai protocolli standard per la trasmissione multimediale. All'inizio dello streaming, si osserva un significativo

incremento del traffico UDP caratterizzato da pacchetti di lunghezza variabile.

Tuttavia, tentando di decodificare questi flussi UDP attraverso il protocollo RTP (Real-time Transport Protocol), comune per la trasmissione di contenuti multimediali in tempo reale, i pacchetti risultano malformati. L'analisi dettagliata dei dati rivela che, dopo i campi standard dell'header UDP (source port, destination port, lunghezza e checksum), invece del tipico campo versione dell'header RTP (che dovrebbe presentare valori `01` o `10`) sono presenti sequenze di byte apparentemente casuali, come `cc aa ab ab` oppure `3e 6f ad a8`.

L'assenza di handshake DTLS (Datagram Transport Layer Security) o altri meccanismi (visibili) di negoziazione delle chiavi nel traffico intercettato, fa pensare all'implementazione di un protocollo proprietario per la trasmissione del flusso video. Questa soluzione proprietaria presumibilmente integra meccanismi di cifratura per proteggere il contenuto multimediale durante la trasmissione.

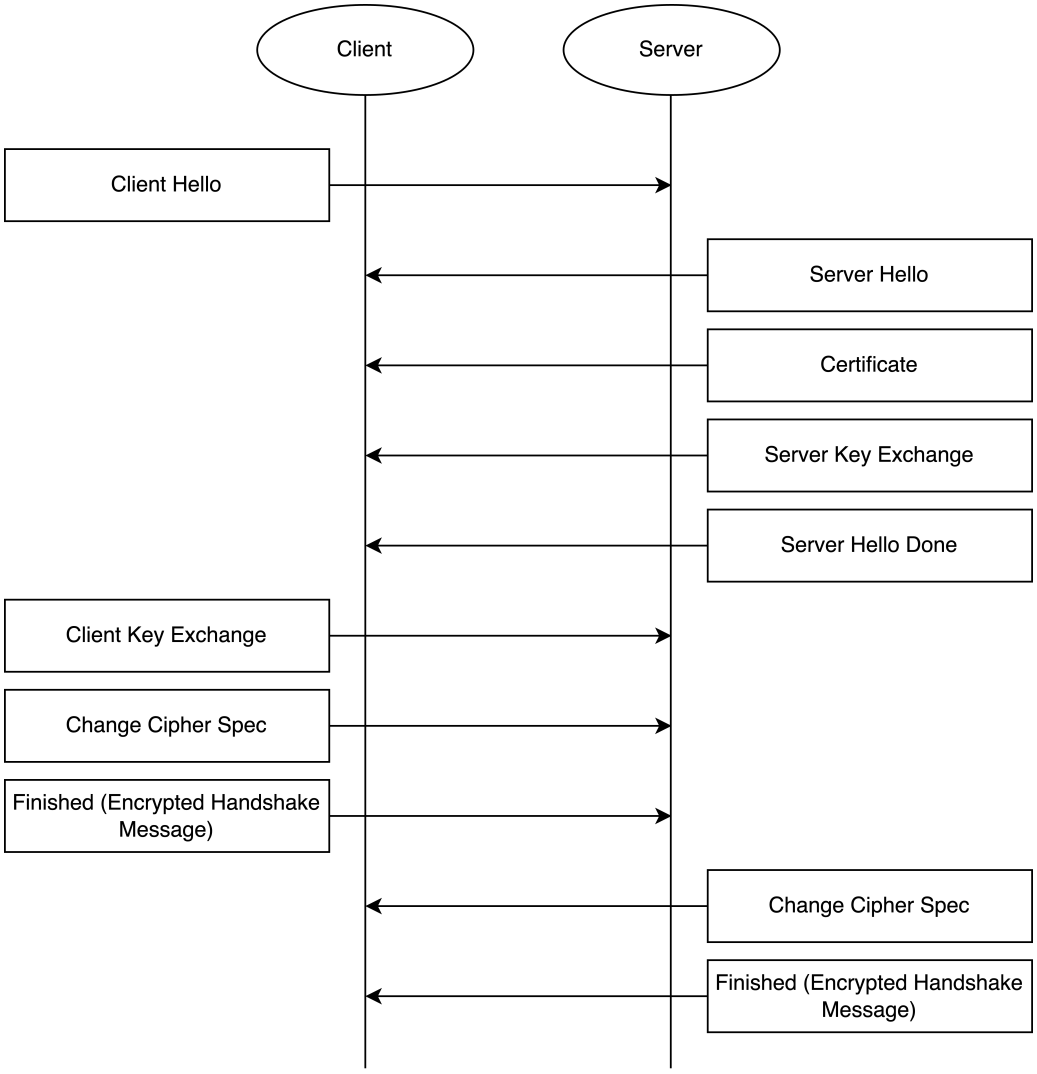

Figura 4.1: Handshake TLS

5 Testing

5.1 Scan

Per procedere con l'analisi di sicurezza del dispositivo, è stata condotta una scansione delle porte (eventualmente esposte) della Wyze Cam v3. Queste operazioni sono state eseguite utilizzando `scan` [27], uno script sviluppato da Matteo Mariotti [26]. `scan` è basato su `nmap` [13], che permette di condurre scansioni TCP e UDP, oltre ad ottenere informazioni in merito al dispositivo che si sta scansionando (ad esempio la versione del sistema operativo in utilizzo).

Le principali opzioni disponibili nello script (`scan -h`) includono:

```
Usage: scan [OPTIONS]
    -u              perform udp scan
    -a              perform attack with default scripts
    -t              perform tcp scan
    -f              full scans
    -d              discovery mode (only -i and -o relevant)
    -v              scan for OS version
    -o  FILENAME    redirect output to FILENAME
    -i  IP          scan specified IP
    -m  MODE        tcp scan mode
```

L'opzione `-d` è usata per identificazione di host attivi sulla rete, mentre `-i` permette di specificare l'indirizzo IP target o il range di rete in notazione CIDR. Le opzioni `-t` e `-u` abilitano rispettivamente scansioni TCP e UDP e possono essere combinate. L'opzione `-f` estende la scansione a tutte le 65535 porte invece di limitarsi alle 1000 più comuni, mentre `-v` abilita il rilevamento del sistema operativo. L'opzione `-a` attiva l'esecuzione degli script di default di `nmap` sulle porte aperte e `-o` permette di esportare l'output su file.

5.1.1 Scansione con firmware standard

Le scansioni sono state effettuate sulla videocamera configurata con il firmware ufficiale più recente (versione 4.36.14.3497 [36]).

Esecuzione della scansione completa

La scansione è stata avviata utilizzando tutti i flag disponibili, così da eseguire un'analisi più ampia possibile:

```
sudo ./scan -futav -o results.txt -i 192.168.2.2
```

I comandi eseguiti dallo script includono una scansione iniziale con rilevamento di sistema operativo e versioni, seguita da scansioni TCP e UDP complete.

Rilevamento del sistema operativo

La prima fase della scansione ha tentato di identificare il sistema operativo e le versioni dei servizi in esecuzione:

```
First scan and os version
***Executing command: nmap -O -sV 192.168.2.2

Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-30 11:03 CEST
Nmap scan report for 192.168.2.2
```

```
Host is up (0.0059s latency).
All 1000 scanned ports on 192.168.2.2 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 80:48:2C:41:3C:5E (Wyze Labs)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop
```

Il risultato mostra che tutte le 1000 porte più comuni risultano chiuse, con il dispositivo che risponde con pacchetti di reset TCP. Il MAC address conferma l'identità del dispositivo, mentre il tentativo di fingerprinting del sistema operativo non produce risultati definitivi.

Scansione TCP completa

La seconda parte dello script prevede una scansione TCP su tutte le 65535 porte disponibili:

```
Full TCP scan
***Executing command: nmap -p- -sS 192.168.2.2

Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-30 11:03 CEST
Nmap scan report for 192.168.2.2
Host is up (0.0094s latency).
All 65535 scanned ports on 192.168.2.2 are in ignored states.
Not shown: 65535 closed tcp ports (reset)
MAC Address: 80:48:2C:41:3C:5E (Wyze Labs)

Nmap done: 1 IP address (1 host up) scanned in 57569.55 seconds
```

La scansione TCP completa, portata a termine in circa 16 ore, ha confermato l'assenza totale di porte TCP aperte. Tutte le 65535 porte risultano chiuse con risposta di reset, indicando che il dispositivo riceve le connessioni ma le rifiuta attivamente, non disponendo di servizi configurati per gestirle.

Scansione UDP completa

La terza parte dello script prevede una scansione UDP completa, configurata con parametri per ridurre i tempi di esecuzione:

```
Full UDP scan
***Executing command: nmap -sU -p- 192.168.2.2 --max-rtt-timeout 20ms --max-retries 0

Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-31 03:02 CEST
Warning: 192.168.2.2 giving up on port because retransmission cap hit (0).
Nmap scan report for 192.168.2.2
Host is up (0.010s latency).
All 65535 scanned ports on 192.168.2.2 are in ignored states.
Not shown: 62621 open|filtered udp ports (no-response), 2914 closed udp ports (port-
    unreach)
```

La scansione UDP ha prodotto risultati misti a causa dei parametri di timeout aggressivi (20ms). La maggior parte delle porte (62621) risulta in stato `open|filtered` a causa della mancanza di risposta entro il timeout impostato, mentre 2914 porte risultano esplicitamente chiuse.

Risultati finali della scansione

Il completamento dello script ha prodotto un output finale che riassume i risultati ottenuti:

```
Scanning results:

Not shown: 62621
--------------------
Starting to attack discovered ports
```

Il messaggio finale indica che non sono stati identificati servizi attivi sui quali eseguire gli script di attacco, confermando l'architettura client-only del dispositivo.

I risultati della scansione dimostrano che la Wyze Cam v3 con firmware standard è sicura dal punto di vista della rete locale. Il dispositivo non espone alcun servizio di rete direttamente accessibile, essendo di fatto esclusivamente un client per le comunicazioni verso i servizi cloud del produttore.

5.2 RTSP

Per verificare il comportamento in presenza di servizi attivi è stato installato il firmware modificato ufficiale di Wyze che abilita il supporto RTSP (4.61.0.3). Questo firmware, disponibile nella sezione dedicata del sito del produttore [36], espone funzionalità di streaming video per l'integrazione con sistemi di videosorveglianza di terze parti.

5.2.1 Il protocollo RTSP

RTSP (Real Time Streaming Protocol) [24] è un protocollo di livello applicativo che serve a gestire il controllo della trasmissione di flussi audio/video in tempo reale. RTSP opera tipicamente sulla porta TCP 554, permettendo di stabilire e controllare sessioni di streaming audio e video.

Il protocollo RTSP è di tipo stateful, mantenendo uno stato della sessione tra client e server durante lo streaming. RTSP prevede di poter controllare flussi multimediali continui, supportando operazioni come play, pause, stop, fast-forward e seek temporale. Il protocollo non trasporta direttamente i dati multimediali, ma coordina la loro trasmissione attraverso protocolli complementari (ad esempio RTP).

5.2.2 Installazione del firmware RTSP

Il firmware RTSP (versione 4.61.0.3 [20]) è stato scaricato dalla sezione firmware ufficiale di Wyze ed installato manualmente (mediante la scheda microSD). Una volta completata l'installazione, il servizio RTSP è stato attivato attraverso l'apposita sezione dell'app mobile, configurando le credenziali di accesso per l'autenticazione.

Questo firmware rappresenta un caso particolare: vengono esposti alcuni servizi di rete per permettere l'accesso diretto al flusso audio/video, in contrasto con l'architettura proposta dal firmware standard. L'attivazione del servizio RTSP trasforma la videocamera da dispositivo cloud-dipendente a telecamera IP standard, mantenendo tuttavia le funzionalità di base offerte da Wyze.

5.2.3 Scansione con servizio RTSP attivo

L'esecuzione dello script `scan` [27] sulla videocamera (configurata con firmware RTSP) ha prodotto risultati diversi rispetto alla configurazione con il firmware standard:

```
sudo ./scan -futav -o resultsRTSP.txt -i 192.168.2.2

First scan and os version
***Executing command: nmap -O -sV 192.168.2.2

Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-31 14:52 CEST
Nmap scan report for 192.168.2.2
Host is up (0.0099s latency).
Not shown: 999 closed tcp ports (reset)
PORT    STATE SERVICE VERSION
554/tcp open  rtsp    DoorBird video doorbell rtspd
MAC Address: 80:48:2C:41:3C:5E (Wyze Labs)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.13
Network Distance: 1 hop
Service Info: Device: webcam
```

```
OS and Service detection performed. Please report any incorrect results at https://
    nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.13 seconds
```

In questo caso, lo script `scan` ha condotto un fingerprinting accurato del sistema operativo, identificando correttamente il kernel Linux nella versione 2.6.32-3.13. Questa informazione è coerente con quanto dichiarato nella documentazione open source di Wyze [40], che specifica l'utilizzo di Linux 3.10.14 (modificato) come base del sistema operativo del dispositivo.

`scan` ha inoltre identificato il daemon RTSP come `DoorBird video doorbell rtspd`.

5.2.4 Attacco brute-force

Al fine di valutare la sicurezza del servizio RTSP esposto è stato sviluppato uno script dedicato, `attackRTSP` (B.1), per l'esecuzione di test di brute-force specifici per il protocollo RTSP.

Lo script utilizza il modulo NSE (Nmap Scripting Engine) `rtsp-url-brute` [29], con lo scopo di tentare un attacco brute-force sugli URL più comunemente utilizzati dai dispositivi di videosorveglianza.

Risultati del test di brute force

```
nmap -Pn -sV -p554 --script rtsp-url-brute 192.168.2.2

PORT    STATE SERVICE VERSION
554/tcp open  rtsp    DoorBird video doorbell rtspd
| rtsp-url-brute:
|   other responses:
|     401:
|       rtsp://192.168.2.2/
|       rtsp://192.168.2.2/live
|       rtsp://192.168.2.2/live.sdp
|       rtsp://192.168.2.2/live/ch0
|       rtsp://192.168.2.2/livestream
|       rtsp://192.168.2.2/video
|       rtsp://192.168.2.2/video.h264
|       rtsp://192.168.2.2/cam/realmonitor?channel=1&subtype=00
|       [... oltre 150 percorsi testati ...]
```

Il test ha generato tentativi di accesso su oltre 150 URL tipici dei dispositivi di videosorveglianza, basandosi sulle convenzioni di nome utilizzate dai principali produttori. Tutti i tentativi hanno ricevuto una risposta `HTTP 401 (Unauthorized)`: il servizio RTSP richiede correttamente l'autenticazione per l'accesso alle risorse video.

I risultati del test dimostrano come, nonostante la porta dedicata all'RTSP sia aperta, i requisiti di sicurezza di base vengano rispettati. Infatti l'assenza di endpoint accessibili senza credenziali valide previene l'accesso non autorizzato.

5.2.5 Injection

È stato sviluppato uno script (B.2) per effettuare test di injection (di tipo path traversal e header injection) al fine di identificare possibili falle nella validazione degli input. In particolare, il path traversal [14] è una vulnerabilità che permette di accedere a directory (e files) al di fuori della directory `root` tramite l'uso di sequenze di directory traversal come `../`. Nel contesto RTSP questo tipo di attacco tenta di sfruttare il modo in cui vengono gestiti gli URI, allo scopo di accedere a file di sistema.

Lo script `injection` (B.2) implementa un attacco di Path Traversal semplificato, la cui esecuzione ha prodotto il seguente risultato:

```
=== Path Traversal Test (/etc/passwd) ===
RTSP/1.0 401 Unauthorized
CSeq: 1
Date: Thu, Jul 31 2025 13:38:07 GMT
WWW-Authenticate: Digest realm="LIVE555 Streaming Media", nonce="2
    ee31ce572c0a73b960b6fba54b4edf8"

=== Header Injection Test ===
RTSP/1.0 200 OK
CSeq: 2
Date: Thu, Jul 31 2025 13:38:08 GMT
Public: OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, GET_PARAMETER, SET_PARAMETER
```

È stata ottenuta una risposta `401 Unauthorized`, indicando che il servizio richiede autenticazione anche per percorsi non validi, impedendo l'accesso diretto a files e directory di sistema. La risposta include un digest con realm `LIVE555 Streaming Media` e un nonce univoco per la sessione.

Identificazione di vulnerabilità note

Sebbene i test diretti di injection non abbiano rivelato vulnerabilità, dai risultati si può notare l'utilizzo di `LIVE555 Streaming Media` [11]. Questa libreria è nota per essere soggetta a diverse vulnerabilità, documentate nel database CVE (Common Vulnerabilities and Exposures) [12].

In particolare LIVE555 ha presentato falle di sicurezza con punteggi CVSS compresi tra 7.5 e 9.8: queste vulnerabilità rappresentano un rischio significativo.

La sicurezza del servizio RTSP dipende (anche) dal mantenimento degli aggiornamenti della libreria analizzata sopra. Questo evidenzia quanto sia importante mantenere aggiornate le dipendenze software nei dispositivi IoT, dove le vulnerabilità delle librerie di terze parti possono compromettere la sicurezza dell'intero sistema.

5.3 Rate Limiting

È stato condotto un test specifico di rate limiting sull'endpoint di autenticazione principale ai servizi Wyze. Questo tipo di test è fondamentale per verificare l'implementazione di meccanismi di protezione contro attacchi DoS, che potrebbero compromettere la disponibilità del servizio.

Metodologia del test

Il test è stato condotto sull'endpoint di login principale ai servizi Wyze: `https://auth-prod.api.wyze.com/api/user/login`. È stato sviluppato uno script Python, `rateLimitingTest` (B.4), che simula un attacco DoS.

Lo script implementa un test che:

- utilizza credenziali non valide
- invia richieste `HTTP POST` con un intervallo di 0.2 secondi tra i tentativi
- monitora i codici di stato `HTTP` per riconoscere l'attivazione del meccanismo di rate limiting
- si interrompe automaticamente al ricevimento del codice `429 Too Many Requests`
- è configurato per un massimo di 200 tentativi, così da evitare impatti prolungati sul servizio

Risultati del test

L'esecuzione del test ha dimostrato l'efficacia dei meccanismi di rate limiting implementati da Wyze:

```
python3 rate_limiting_test.py
Flooding https://auth-prod.api.wyze.com/api/user/login for rate-limit test (max 200
    attempts)

[001] HTTP 400 Bad Request
[002] HTTP 400 Bad Request
[003] HTTP 400 Bad Request
[004] HTTP 400 Bad Request
[005] HTTP 400 Bad Request
[006] HTTP 400 Bad Request
[007] HTTP 400 Bad Request
[008] HTTP 400 Bad Request
[009] HTTP 400 Bad Request
[010] HTTP 400 Bad Request
[011] HTTP 400 Bad Request
[012] HTTP 400 Bad Request
[013] HTTP 400 Bad Request
[014] HTTP 400 Bad Request
[015] HTTP 400 Bad Request
[016] HTTP 400 Bad Request
[017] HTTP 400 Bad Request
[018] HTTP 400 Bad Request
[019] HTTP 400 Bad Request
[020] HTTP 400 Bad Request
[021] HTTP 400 Bad Request
[022] RATE LIMITED --> HTTP 429
```

I risultati mostrano che:

- Le prime 21 richieste hanno ricevuto il codice di stato `400 Bad Request`, indicando che le credenziali fornite non sono valide. Alla 22ª richiesta il sistema ha attivato il rate limiting restituendo il codice `429 Too Many Requests`.
- Il meccanismo si attiva dopo circa una ventina di tentativi consecutivi, indipendentemente dalla validità delle credenziali.

Test ripetuti hanno dimostrato che il rate limiting opera su una finestra temporale di circa un minuto. Esecuzioni successive del test, immediatamente dopo la prima volta, hanno causato l'attivazione immediata del rate limiting. Invece test condotti dopo un intervallo di tempo superiore a circa un minuto hanno presentato un comportamento normale.

Conformità ai requisiti di sicurezza

La gestione del rate limiting da parte di Wyze soddisfa il relativo requisito di sicurezza del Cyber Resilience Act (CRA).

La soglia di circa venti richieste rappresenta un compromesso ragionevole che:

- permette agli utenti di gestire errori occasionali di login senza impatti
- riduce il carico sui server di autenticazione durante tentativi di attacco

La finestra temporale di circa un minuto per il reset del rate limiting è appropriata per la maggior parte dei casi d'uso, consentendo agli utenti di ritentare l'accesso dopo aver risolto eventuali problemi di credenziali.

6 Procedura di aggiornamento

6.1 Descrizione

Gli aggiornamenti firmware della Wyze Cam v3 vengono distribuiti da un server centrale. L'infrastruttura utilizza diversi endpoints per gestire la procedura di aggiornamento, dalla verifica della disponibilità fino all'installazione sul dispositivo. E' possibile, tuttavia, scaricare manualmente (dall'apposita sezione del sito internet di Wyze [36]) il file `.bin` contenente l'immagine del firmware, per poi procedere ad installarla sul dispositivo mediante una scheda microSD.

Modalità operative

Il dispositivo è settato di default in modalità di aggiornamento automatico [34], così da poter verificare periodicamente la disponibilità di nuove versioni del firmware e installarle autonomamente. In questo modo si punta a far sì che il dispositivo rimanga sempre aggiornato con le ultime patch di sicurezza.

Tuttavia, in conformità ai requisiti del Cyber Resilience Act, è offerta agli utenti la possibilità di disattivare gli aggiornamenti automatici e di gestirli manualmente [34]. In questa modalità l'app notifica la disponibilità di aggiornamenti attraverso popup informativi e permette di far sì che sia l'utente a scegliere quando (e se) procedere con l'installazione (fig. 6.2).

Come menzionato prima, il dispositivo supporta anche l'aggiornamento manuale tramite scheda microSD. Questa modalità permette di installare sul dispositivo i firmwares scaricati dal sito ufficiale Wyze [36], offrendo libertà nella scelta della versione.

Processo di aggiornamento

Il workflow di aggiornamento inizia con la verifica della disponibilità di nuove versioni. Il dispositivo contatta regolarmente i server Wyze per confrontare la propria versione firmware con l'ultima disponibile: questo controllo avviene indipendentemente dalla configurazione degli aggiornamenti automatici.

Il trasferimento del firmware avviene esclusivamente tramite connessioni HTTPS protette da certificati SSL/TLS, utilizzando il protocollo TLS con SNI (Server Name Indication) per garantire l'autenticità del server di origine.

Durante il processo di aggiornamento viene fornito un feedback appropriato attraverso l'applicazione mobile. In ogni caso è presente una documentazione dettagliata delle modifiche, incluse in ogni aggiornamento, nell'apposita sezione del sito internet di Wyze.

6.2 Analisi di un aggiornamento firmware legittimo

Per comprendere meglio il processo di aggiornamento e controllare l'implementazione delle misure di sicurezza descritte sopra, è stato condotto un test di aggiornamento dal firmware di versione `4.36.13.0416` alla versione `4.36.14.3497`.

Procedura di installazione firmware `4.36.13.0416` [35]

È stato necessario installare manualmente la versione `4.36.13.0416`, seguendo questa procedura:

1. Download del firmware `4.36.13.0416` dal sito ufficiale Wyze (rilasciato il 18/07/2024).

2. Formattazione di una scheda microSD con file system `FAT32`.
3. Copia del file firmware (`.bin`) sulla scheda microSD.
4. Inserimento della microSD nella videocamera.
5. Tenendo premuto il pulsante SETUP è stata collegata l'alimentazione ed il pulsante stesso è stato mantenuto premuto fino a quando il LED di stato è diventato viola.
6. Configurazione del dispositivo tramite l'app Wyze.

Processo di aggiornamento tramite applicazione

Una volta configurata la videocamera con il firmware `4.36.13.0416`, il sistema ha proposto automaticamente l'aggiornamento alla versione più recente tramite un popup nell'app.

La richiesta di aggiornamento ha generato questo avviso:

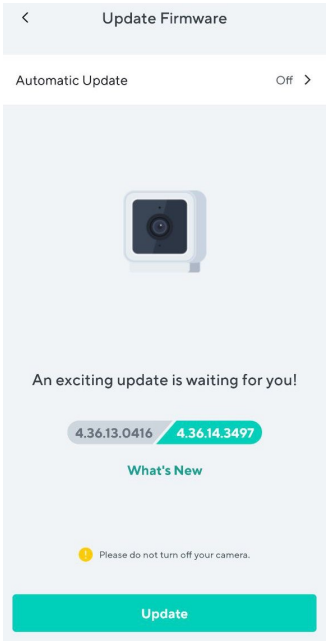

Figura 6.1: Avviso di aggiornamento firmware nell'applicazione Wyze

Analisi del traffico di rete durante l'aggiornamento

L'analisi del traffico di rete durante l'aggiornamento è stata condotta utilizzando Wireshark [33], così da monitorare tutte le comunicazioni tra la videocamera e i server Wyze.

Fasi della comunicazione

Il processo di aggiornamento ha seguito una sequenza strutturata di comunicazioni:

1. **Query iniziale ai servizi di aggiornamento**: Richiesta standard a `wyze-firmware-upgrade-service.wyzecam.com` per verificare la disponibilità di aggiornamenti.
2. **Stabilimento della connessione sicura**: Invio del `Client Hello` con SNI=`wyze-firmware-upgrade-service.wyzecam.com`, seguito dall'handshake TLS.
3. **Download sicuro del firmware**: Trasferimento dei dati del firmware attraverso connessione TCP protetta da TLS.

4. **Verifica finale**: Query aggiuntiva a `wyze-firmware-upgrade-service.wyzecam.com` prima del riavvio del dispositivo.

5. **Riavvio del sistema**: Disconnessione dall'hotspot e riavvio della videocamera.

6. **Verifica post aggiornamento**: Dopo il riavvio, nuove query ai servizi `wyze-firmware-upgrade-service.wyzecam.com`, `wyze-device-log` e altri endpoint.

Verifica periodica degli aggiornamenti

Un aspetto importante emerso dall'analisi è che vengono effettuate regolarmente query a `wyze-firmware-upgrad` `wyzecam.com` per confrontare la versione firmware corrente con l'ultima disponibile. Questo comportamento si verifica indipendentemente dal fatto che gli aggiornamenti automatici siano attivati o disattivati.

6.3 Attacco MITM

La gestione degli aggiornamenti firmware è da considerarsi un elemento critico per la sicurezza dei dispositivi IoT. Una procedura di aggiornamento inadeguata può esporre il dispositivo a numerose tipologie di attacco, inclusi gli attacchi **man-in-the-middle (MITM)**. In questo capitolo viene analizzata la procedura di aggiornamento della Wyze Cam v3, con particolare attenzione alla validazione dei certificati SSL/TLS.

La validazione dei certificati SSL/TLS è cruciale, poiché altrimenti sarebbe possibile un attacco di questo tipo:

1. L'attacker intercetta le richieste DNS della videocamera.

2. Il dominio degli aggiornamenti viene risolto verso un IP sotto il controllo dell'attacker.

3. L'attacker configura un webserver (con certificato self-signed) che distribuisce un firmware maligno.

4. La videocamera scarica e installa l'aggiornamento compromesso se non verifica correttamente l'autenticità.

6.3.1 Attacco MITM con DNS Spoofing

Il primo approccio ha previsto la configurazione di un server DNS locale utilizzando `dnsmasq` [5], in modo da reindirizzare le richieste verso `wyze-firmware-upgrade-service.wyzecam.com` ad un server HTTPS locale.

Configurazione dell'ambiente di test

È stato configurato un server HTTPS locale in ascolto sulla porta 443 con certificati SSL self-signed generati tramite OpenSSL [30], così da simulare un server di aggiornamento compromesso. Allo stesso tempo è stata impostata una configurazione DNS locale, così da risolvere il dominio degli aggiornamenti Wyze verso l'indirizzo IP locale del server di test.

Tuttavia la videocamera ha continuato ad utilizzare le configurazioni DNS di sistema predefinite, ignorando la configurazione DNS locale impostata in precedenza, anche a seguito di un blocco esplicito delle configurazioni DNS pubbliche (ad esempio `8.8.8.8`, utilizzato dalla videocamera per risolvere gli alias di `wyze-firmware-upgrade-service.wyzecam.com`)

Limitazioni dell'approccio

Durante i test sono emersi due scenari problematici:

- **Regole troppo permissive**: La videocamera ignorava la configurazione DNS locale e continuava a utilizzare le configurazioni DNS di sistema.
- **Regole troppo restrittive**: Le configurazioni di rete troncavano completamente la connettività ad internet della videocamera, che non riusciva nemmeno a risolvere `api.wyzecam.com` per le operazioni di base.

Solo in un caso il reindirizzamento di `wyze-firmware-upgrade-service.wyzecam.com` verso `192.168.2.100` (IP del server HTTPS configurato dall'attacker) è riuscito. Tuttavia il server HTTPS configurato non ha ricevuto alcuna richiesta.

6.3.2 Attacco MITM con `bettercap`

Per superare le limitazioni dell'approccio basato solo su DNS spoofing è stato implementato un attacco man-in-the-middle utilizzando `bettercap` [1], una versione moderna del tool `ettercap` [6] descritto da Alessandro Brighenti [21] per attacchi ARP spoofing.

Attacco ARP spoofing

L'attacco ARP spoofing sfrutta le vulnerabilità del protocollo ARP (Address Resolution Protocol), che non richiede autenticazione per gli aggiornamenti delle tabelle ARP. Come descritto da Brighenti [21], il protocollo ARP traduce indirizzi IP in indirizzi MAC attraverso due tipi di query:

- **Request-reply**: un computer invia in broadcast "`who is SOME_IP`" per conoscere l'indirizzo MAC associato.
- **Gratuitous**: un host invia spontaneamente "`SOME_IP is at SOME_MAC`" senza richiesta.

L'assenza di autenticazione permette a un attacker di posizionarsi come MITM, inviando continuamente messaggi ARP gratuitous che "avvelenano" le tabelle ARP dei dispositivi target.

La configurazione dell'attacco ha previsto i seguenti passaggi:

Generazione di certificati self-signed

```
openssl req -x509 -newkey rsa:4096 -keyout wyze.key -out wyze.crt -days 365 -nodes \
-subj "/CN=api.wyzecam.com" \
-addext "subjectAltName=DNS:api.wyzecam.com,DNS:*.wyzecam.com,DNS:wyze-firmware-
    upgrade-service.wyzecam.com"
```

Il comando OpenSSL [30] genera un certificato X.509 auto firmato (quindi forgiato), così da simulare un server compromesso. Il comando crea una private key RSA a 4096 bit ed il corrispondente certificato con validità di 365 giorni, configurato con `api.wyzecam.com` (Common Name) e `api.wyzecam.com`, `*.wyzecam.com` e `wyze-firmware-upgrade-service.wyzecam.com` (Subject Alternative Names): in questo modo il certificato forgiato può essere considerato valido per tutti gli endpoint critici Wyze durante l'attacco man-in-the-middle. Il certificato, non essendo firmato da una Certificate Authority (CA) riconosciuta, dovrebbe essere rifiutato da dispositivi che implementano correttamente la validazione SSL/TLS.

Avvio del server HTTPS con certificato forgiato

```
sudo openssl s_server -cert wyze.crt -key wyze.key -port 443 -HTTP
```

Il server HTTPS di test viene avviato sulla porta 443, utilizzando il certificato forgiato precedentemente. Il parametro `-HTTP` permette al server di rispondere a richieste HTTP di base.

Configurazione dell'attacco MITM

```
sudo bettercap -iface bridge100
net.recon on
net.show
set arp.spoof.targets 192.168.2.2
arp.spoof on
set http.proxy.sslstrip true
http.proxy on
net.sniff on
set dns.spoof.domains api.wyzecam.com,*.wyzecam.com,wyze-firmware-upgrade-service.
    wyzecam.com
set dns.spoof.address 192.168.2.1
dns.spoof on
```

Dopo l'avvio del tool sull'interfaccia `bridge100`, viene attivata la ricognizione di rete per identificare i dispositivi connessi. L'attacco procede con l'ARP spoofing verso il target `192.168.2.2` (la videocamera), il DNS spoofing per reindirizzare i domini Wyze verso `192.168.2.1` (l'attacker) e l'attivazione di un proxy HTTP con SSL stripping.

6.4 Risultati dell'attacco MITM

L'esecuzione dell'attacco ha prodotto risultati che dimostrano l'efficacia dei meccanismi di validazione SSL/TLS implementati dalla videocamera.

Intercettazione DNS

Il sistema ha intercettato con successo le richieste DNS della videocamera, come dimostrato dai log di `bettercap`:

```
[21:31:33] [sys.log] [inf] dns.spoof sending spoofed DNS reply for api.wyzecam.com
    (->192.168.2.1) to 192.168.2.2 : 80:48:2c:41:3c:5e (Wyze Labs Inc).
[21:31:33] [net.sniff.dns] dns 8.8.8.8 > 192.168.2.2 : api.wyzecam.com is local
[21:31:33] [sys.log] [inf] dns.spoof sending spoofed DNS reply for api.wyzecam.com
    (->192.168.2.1) to 192.168.2.2 : 80:48:2c:41:3c:5e (Wyze Labs Inc).
```

I log confermano che la videocamera ha tentato di risolvere `api.wyzecam.com` utilizzando diversi server DNS (`8.8.8.8`, `8.8.4.4`, `208.67.222.123`, `208.67.220.123`), che queste richieste sono state intercettate con successo e sono state reindirizzate verso l'indirizzo IP controllato dall'attacker.

Validazione dei certificati SSL/TLS

Nonostante l'intercettazione DNS sia riuscita, la videocamera ha dimostrato un comportamento di sicurezza coerente durante il processo di validazione SSL/TLS. I log del server HTTPS mostrano il rifiuto del certificato self-signed attraverso il codice TLS Alert 48.
Il TLS Alert 48 corrisponde a "`unknown_ca`" secondo la RFC 5246 [22], indicando che il client (la videocamera) ha rifiutato il certificato perché emesso da una CA non riconosciuta.

Si può quindi confermare che la Wyze Cam segue best practice di sicurezza corrette, utilizzando solamente connessioni crittografate per il trasferimento del firmware, validando l'identità del server attraverso certificati SSL/TLS, effettuando controlli di integrità pre e post installazione e verificando periodicamente la disponibilità di aggiornamenti.

Sebbene i test di attacco non abbiano permesso di raggiungere l'endpoint specifico degli aggiornamenti firmware (`wyze-firmware-upgrade-service.wyzecam.com`), il comportamento osservato per `api.wyzecam.com` suggerisce che gli stessi meccanismi di validazione vengano applicati a tutti gli endpoint Wyze.

7 Firmware

Come menzionato in precedenza, tutte le versioni del firmware della Wyze Cam v3 sono scaricabili dall'apposita sezione del sito di Wyze [36]. Per ogni versione è presente una breve descrizione delle modifiche effettuate rispetto alla versione precedente.

7.1 Componenti open source

Come specificato nella pagina "Open Source Software" [40], i firmware della Wyze Cam v3 incorporano numerosi componenti open source, che costituiscono la base del firmware stesso. Tra i componenti principali possiamo trovare:

- **Linux (kernel)** [17]: Sistema operativo di base (versione 3.10.14 modificata)
- **BusyBox** [3]: Collezione di tools Unix ottimizzata per sistemi embedded
- **OpenSSL** [30]: Libreria crittografica per comunicazioni sicure
- **libcurl** [10]: Utile per la gestione delle comunicazioni HTTP/HTTPS
- **wpa_supplicant** [19]: Utile per l'autenticazione wireless mediante WPA/WPA2
- **FFmpeg** [7]: Framework per elaborazione audio/video

La presenza di questi componenti open source offre vantaggi in termini di stabilità e supporto, ma introduce potenziali rischi di sicurezza legati a vulnerabilità note in versioni obsolete (come nel caso del firmware analizzato).

7.2 Analisi statica

L'analisi statica è una tecnica di analisi firmware che prevede di esaminare il codice sorgente, i file binari e la struttura del firmware senza eseguirlo (al contrario dell'analisi dinamica, che richiederebbe l'estrazione fisica del firmware dal dispositivo e la sua esecuzione).

In questo modo è possibile identificare componenti e filesystem, esaminare le varie configurazioni, verificare impostazioni di sicurezza, credenziali e parametri di sistema, identificare librerie, versioni software e componenti utilizzati, per poi procedere a confrontare i componenti identificati con database di vulnerabilità note.

In questo caso si è preferito affidarsi all'approccio statico, analizzando l'ultimo firmware disponibile per Wyze Cam v3 scaricato dal sito del produttore [36] (versione 4.36.14.3497).

7.3 Binwalk

Come specificato da Simone Rossi [32], i vendor spesso comprimono o cifrano il firmware per risparmiare spazio e nascondere informazioni sensibili, rilasciando un file binario. La parte complessa sta nel trovare quale sezione del file binario contiene l'archivio e quale algoritmo di compressione è stato utilizzato. Qui entra in gioco Binwalk [2], un tool che utilizza il pattern matching per identificare strutture dati attraverso la ricerca di magic header specifici.
In particolare il tool riconosce i principali algoritmi di compressione (LZMA, GZIP, BZIP2, ZIP) e supporta l'estrazione ricorsiva con approccio *matryoshka* (come spiegato da Alessandro Brighenti [21]). La signature di un file è una sequenza di byte identificativi nell'header, unica per ogni tipo di file, che

permette a Binwalk di distinguere tra diversi formati di archivio.
Per poter utilizzare Binwalk è sufficiente eseguire i seguenti comandi:

```
sudo apt install binwalk
binwalk firmware.bin
binwalk -e firmware.bin  #estrazione automatica
```

7.4 Estrazione del firmware

L'analisi del firmware è iniziata utilizzando Binwalk per identificare la struttura interna del file principale scaricato dalla sezione firmware del sito di Wyze (`demo_wcv3.bin`) [32]. Questa prima scansione ha identificato il firmware come un'immagine `uImage`, formato standard per kernel Linux sviluppato da U-Boot. Come menzionato nella pagina Open Source Software di Wyze [40], si può confermare che è stato utilizzato Linux 3.10.14 per architettura MIPS32.

A seguito dell'estrazione del contenuto principale, l'analisi del file `jz_fw.bin` (unico file estratto prima) ha rivelato una struttura più articolata:

```
binwalk jz_fw.bin
0        0x0       Linux-3.10.14__isvp_swan_1.0__, compression: lzma
2031616  0x1F0000  SquashFS file system, inode count: 362
6029312  0x5C0000  SquashFS file system, inode count: 160
```

Si può notare una struttura organizzata in tre componenti distinte: il kernel Linux 3.10.14, compilato per il target `isvp_swan_1.0` (ISVP, Ingenic Smart Video Platform [8], indica che il kernel è stato ottimizzato per gestire le funzionalità di elaborazione video del SoC, System on Chip) e i due filesystem SquashFS [16] identificati agli offset `0x1F0000` e `0x5C0000`, che rappresentano rispettivamente il sistema operativo base (RootFS) e le applicazioni (AppFS).

Inizialmente il tentativo di estrazione automatica di `jz_fw.bin` non è andato a buon fine: Binwalk ha fallito nell'estrarre i filesystem SquashFS a causa dell'assenza del tool `sasquatch` [15], specializzato nell'estrazione di filesystem SquashFS. Dopo l'installazione del tool menzionato sopra e della riesecuzione del comando di estrazione sono stati ottenuti i due filesystem `rootfs.squashfs` e `appfs.squashfs`, decompressi successivamente con il comando `unsquashfs`. Infine si dispone quindi di due directory complete, `rootfs_extracted` e `appfs_extracted`.

7.5 Analisi del firmware

L'analisi dei filesystem estratti mostra come il firmware preveda una separazione tra il sistema operativo base (RootFS) e le applicazioni specifiche del dispositivo (AppFS), consentendo di aggiornare componenti specifici senza dover ricompilare l'intero OS.

AppFS - Applicazioni specifiche

Il filesystem `appfs_extracted/` contiene tutti i componenti proprietari sviluppati per implementare le funzionalità della Wyze Cam:

`/bin/` — applicativi
Contiene tutti i software che implementano le funzionalità principali della videocamera, in particolare gestione video, comunicazione cloud, elaborazione AI e gestione degli eventi. Include anche tools per monitorare processi, logging e gestione della sicurezza.

`/driver/` — driver hardware
Sono presenti i drivers necessari per il controllo dell'hardware del dispositivo. Ogni driver è compatibile con le specifiche del SoC utilizzato nel dispositivo.

`/etc/` — **configurazioni**
Contiene i file di configurazione specifici per i servizi cloud e l'hardware della videocamera, come gli endpoint dei server Wyze.

`/init/` — **scripts di inizializzazione**
Directory dedicata agli script necessari per l'avvio del sistema e delle applicazioni Wyze. Questi script gestiscono la sequenza di inizializzazione, il caricamento dei driver appropriati e l'avvio di tutti i servizi.

`/lib/` — **librerie**
Raccolta delle librerie sviluppate da Wyze per implementare funzionalità avanzate non disponibili nelle librerie standard.

`/local/` — **certificati**
Contiene i certificati CA necessari per la validazione delle connessioni con i server Wyze e servizi cloud esterni.

RootFS - Sistema operativo base

Il filesystem `rootfs_extracted/` implementa il sistema operativo Linux:

`/bin/` e `/usr/bin/` — **utilities di sistema**
Implementa comandi Unix standard attraverso BusyBox [3], una software suite ottimizzata per sistemi embedded che consente di avere centinaia di utilities Unix in un singolo eseguibile, riducendo l'occupazione di RAM e storage.

`/lib/` — **librerie di sistema**
Ossia le librerie fondamentali per il runtime del sistema operativo Linux. Utilizza uClibc [18]come implementazione della libreria C, ottimizzata per dispositivi IoT. Include supporto per threading, gestione della memoria, operazioni e interfacce di sistema.

`/thirdlib/` — **librerie di terze parti**
Ossia librerie esterne necessarie per la funzionalità del dispositivo. OpenSSL [30] fornisce funzionalità crittografiche necessarie per comunicazioni sicure e validazione certificati. `libcurl` [10] gestisce le comunicazioni HTTP/HTTPS (usate per gli aggiornamenti e per le comunicazioni con i server Wyze, come visto precedentemente).

`/etc/` — **configurazioni**
Contiene tutti i file di configurazione fondamentali per il sistema operativo. Gestisce l'autenticazione, le configurazioni di rete (ad esempio la configurazione DNS) e i parametri di inizializzazione del sistema.

`/usr/share/` — **risorse audio**
Contiene files audio `.wav` utilizzati durante il setup e operazioni normali.

`/sbin/` — **system binaries**
Directory contenente i binari di sistema essenziali per l'amministrazione e la gestione del dispositivo.

7.6 Analisi delle vulnerabilità

7.6.1 Gestione delle credenziali di sistema

Essendo Linux il kernel su cui è basato il funzionamento della Wyze Cam, è necessario controllare la presenza dei file di autenticazione del sistema. Infatti è stata trovata la presenza di credenziali configurate per l'utente root, nei file `/etc/passwd` e `/etc/shadow`:

```
root:x:0:0:root:/:/bin/sh
root:$6$wyzecamv3$8gyTEsAkm1d7wh12Eup5MMcxQwuA1n1FsRtQ ... 0WI0:0:0:99999:7:::
```

(l'hash è stato accorciato per praticità)

Analisi dell'algoritmo di hashing

Per comprendere il livello di sicurezza implementato è necessario analizzare l'algoritmo crittografico utilizzato. Come descritto da Alessandro Brighenti [21], le funzioni hash crittografiche trasformano un messaggio di input di dimensione variabile in un output di dimensione fissa, soddisfacendo (idealmente) alcuni requisiti:

- **One-Way**: deve essere computazionalmente impossibile recuperare il messaggio di input dato l'hash.
- **Strong collision resistance**: deve essere impossibile trovare due messaggi diversi con lo stesso hash.
- **Weak collision resistance**: per un dato messaggio deve essere impossibile trovare un altro messaggio con lo stesso hash.

Per un algoritmo hash sicuro che produce un digest di n bit, un attacker potrebbe trovare una collisione con probabilità del 50% generando $2^{n/2}$ messaggi. Quindi, maggiore è la dimensione del digest, più alta è la sicurezza garantita.

Formato del file /etc/shadow

Nei sistemi operativi basati su Linux, il file `/etc/shadow` contiene la lista degli username e i corrispondenti hash delle password. Solo l'utente root e il gruppo root possono leggere questo file, per evitare accessi agli hash da parte di utenti ordinari. Le righe di questo file seguono il formato:

`[username]:$HashAlgorithm$[salt]$[password hash]:17736:0:99999:7:::`

Nel firmware Wyze Cam v3, la struttura dell'hash rivela:

- **Algoritmo**: SHA-512 (identificato dal prefisso `$6$`)
- **Salt**: `wyzecamv3`
- **Hash**: `8gyTEsAkm1d7wh12Eup5MMcxQwuA1n1FsRtQLUW8dZGo1b1pGRJg ... P0WI0`

L'utilizzo di SHA-512 rappresenta una scelta sicura, producendo hash di 512 bit che offrono un'elevata resistenza ai brute-force attack. Per valutare la robustezza della password sono stati condotti tentativi di cracking utilizzando `John the Ripper` [9], un tool specializzato per attacchi a hash di password. Sono state utilizzate liste di password comuni (dictionary attack) e sono state generate password basate sul contesto (simili a "wyzecam", "wyze", varianti numeriche), tuttavia i vari tentativi non hanno prodotto risultati.

7.6.2 Configurazione Wi-Fi hardcoded

Analizzando i file di configurazione wireless è stata rilevata la presenza di credenziali Wi-Fi hardcoded nel firmware, in particolare nel file `appfs_extracted/bin/wpa.conf`:

```
network={
    ssid="hualaikeji"
    key_mgmt=WPA-PSK
    pairwise=CCMP TKIP
    group=CCMP TKIP
    psk="hualaikeji.."
```

```
    scan_ssid=1
    priority=2
}
```

Questa pratica potrebbe essere problematica poiché, dopo la configurazione iniziale del dispositivo con le credenziali Wi-Fi dell'utente, le quali potrebbero venire memorizzate nello stesso file in chiaro. Tutto ciò comporterebbe l'esposizione delle credenziali Wi-Fi dell'utente, in caso di dump della memoria o estrazione fisica del firmware dal dispositivo.

7.6.3 Esposizione degli endpoint Wyze

L'analisi dei file di configurazione cloud (`appfs_extracted/etc/cloud_profile.ini`) ha rivelato l'esposizione degli endpoint Wyze:

```
[DOMAIN]
UpgradeBeta=beta-firmware-upgrade-service.wyzecam.com
UpgradeProd=wyze-firmware-upgrade-service.wyzecam.com
CloudBeta=beta-api.wyzecam.com
CloudProd=api.wyzecam.com
AIBeta=beta-ai-asm-api.wyzecam.com
AIProd=ai-asm-api.wyzecam.com
DeviceBeta=beta-device-api.wyzecam.com
DeviceProd=device-api.wyzecam.com
EventServiceBeta=event-service-beta.wyzecam.com
EventServiceProd=event-service.wyzecam.com
...
```

L'esposizione degli endpoint, in particolare quelli beta, può costituire un problema di sicurezza. Gli endpoint beta, essendo legati a sviluppo e testing, potrebbero presentare funzionalità sperimentali non testate a sufficienza (inclusa la loro parte di sicurezza). Inoltre, queste informazioni in merito alla configurazione dell'architettura Wyze, facilitano la pianificazione di attacchi MITM (capitolo 6.3), nonostante la presenza di validazione dei certificati SSL/TLS.

7.6.4 Vulnerabilità negli script di inizializzazione

L'analisi dello script principale di inizializzazione `app_init.sh` ha rivelato diverse potenziali problematiche di sicurezza:

- Lo script implementa una logica che attiva una modalità debug se presente il file `/configs/.debug_flag` (non presente di default). In modalità debug il sistema potrebbe potenzialmente esporre funzionalità di debugging non sicure, con tutte le conseguenze del caso.
- Il codice presenta servizi critici commentati, tra cui `telnet` [25]: la riattivazione di `telnet` tramite modifica dello script esporrebbe il dispositivo ad accessi non autenticati.
- Alcuni script (commentati) per la gestione dei core dump potrebbero, se riattivati, scrivere informazioni sensibili in merito alla memoria del dispositivo nella scheda microSD.

7.6.5 `test.sh` e factory mode

Analizzando il file `appfs_extracted/init/factory.sh`, chiamato nello script di inizializzazione menzionato prima, si è potuto notare che, se dovesse esistere la directory `/tmp/factory`, verrebbe eseguito uno script di test (`test.sh`) non controllato. Questo accadrebbe se fosse presente l'archivio `Test.tar` in storage esterni (microSD): in quel caso il file verrebbe estratto automaticamente in `/tmp` e verrebbe creata la directory `/tmp/factory`, portando quindi all'esecuzione di `test.sh` (informazioni trovate analizzando il file `appfs_extracted/bin/factorycheck`).

Questo meccanismo permette teoricamente l'esecuzione di codice inserendo una scheda microSD contenente un archivio `Test.tar` . Questo metodo è stato testato con uno script `test.sh` (allegato B.3)

(salvato all'interno di un archivio `Test.tar`) che crea un file all'interno della microSD e invia dei pacchetti in rete (`ping` e `HTTP GET requests`): probabilmente sono presenti alcune verifiche allo scopo di assicurare che lo script venga da una sorgente trusted, poiché nessun file è stato effettivamente salvato nella microSD e nessuno dei pacchetti è stato rilevato mediante Wireshark [33], di conseguenza lo script `test.sh` non è stato eseguito dalla Wyze Cam.

7.6.6 Gestione certificati e chiavi private

Analizzando `appfs_extracted/bin/iCamera` e `appfs_extracted/bin/iotclient` è stata trovata la presenza di percorsi hardcoded per certificati AWS IoT, in particolare per la chiave privata del dispositivo:

```
/tmp/aws_iot/rootCA.crt
/tmp/aws_iot/cert.pem
/tmp/aws_iot/privkey.pem      # Private key
```

Normalmente la directory `tmp` viene generata a runtime, quindi sarebbe necessario effettuare un dump della memoria per ricavarne il contenuto. È stata inoltre identificata la presenza di un certificato CA Mozilla completo in `appfs_extracted/local/cacert.pem`

7.6.7 Vulnerabilità CVE identificate

Sulla base della descrizione effettuata da Simone Rossi [32], si è proceduto alla verifica delle potenziali vulnerabilità contro le CVE note utilizzando il tool `cve-bin-tool` [4] (sviluppato da Intel per identificare vulnerabilità note nei file binari). Il tool scansiona i file binari per cercare signature di librerie o numeri di versione, confrontandole con CVE noti. Infine viene generato un report che elenca tutte le dipendenze non sicure trovate, specificandone il grado di rischio. Common Vulnerabilities and Exposures (CVE) [28] è un database mantenuto dalla MITRE Corporation, nel quale viene fornito un identificatore unico per ogni vulnerabilità. L'impatto delle vulnerabilità è descritto utilizzando il Common Vulnerability Scoring System (CVSS), che assegna un punteggio che varia da 0 a 10. Più alto è il punteggio, più pericolosa è la vulnerabilità .

Risultati dell'Analisi CVE

L'analisi è stata effettuata, separatamente, per AppFS e per RootFS, generando due report:

- **AppFS:** 12 CVE Critical, 569 CVE High, 1475 CVE Medium, 35 CVE Low e 154 CVE Unknown.
- **RootFS:** 4 CVE Critical, 13 CVE High, 18 CVE Medium, 1 CVE Low, 1 CVE Unknown.

La maggior parte delle vulnerabilità nell'AppFS sono associate al kernel Linux 3.10.14, una versione datata con numerose CVE note. Nel RootFS, le vulnerabilità riguardano principalmente componenti open source come `libcurl` [10], `OpenSSL` [30], `BusyBox` [3] e `wpa_supplicant` [19].

IV Conclusione

A seguito delle analisi effettuate relativamente a vari aspetti della Wyze Cam è possibile procedere al controllo di quali requisiti del CRA [31] siano stati rispettati e quali non lo siano. E' necessario sottolineare come la mancanza di una SBOM pubblica (capitolo 3.2) abbia limitato significativamente la valutazione della sicurezza: molti requisiti del CRA prevedono di basarsi sulla SBOM, allo scopo di ottenere informazioni sui componenti presenti nel software del dispositivo. A causa di questa mancanza i requisiti *1.1.2a*, *1.1.2b*, *1.1.2d* sono verificabili solo parzialmente, mentre il requisito *1.2.1* non è verificabile (se non, in parte, con l'analisi firmware). Inoltre:

- il requisiti *1.1.2a* e *1.2.1* sono parzialmente rispettati e verificabili con l'analisi firmware, che consente di controllare eventuali CVE o falle all'interno del software come descritto nel capitolo 7.
- il requisito *1.1.2b* è parzialmente rispettato: infatti sono presenti articoli nel sito di Wyze che spiegano come effettuare gli aggiornamenti ed impostare gli aggiornamenti automatici, così come è spiegato il modo corretto per effettuare il reset della videocamera. Per quanto riguarda le intercettazioni delle comunicazioni e la verifica delle porte aperte, il tutto è descritto nel capitolo 4 e capitolo 5.
- il requisito *1.1.2c* è rispettato: infatti l'analisi dei pacchetti scambiati durante un aggiornamento firmware (e la frequenza degli stessi) è descritta nel capitolo 6.2. Inoltre, al momento della notifica di un aggiornamento, il dispositivo non impedisce all'utente l'utilizzo dello stesso, invece notifica la possibilità di effettuare un aggiornamento mediante un popup nell'app Wyze.
- il requisito *1.1.2e* è parzialmente rispettato: l'analisi dei pacchetti, delle porte e della sicurezza nella comunicazione sono descritti nei capitoli 4 e 5 come un esempio di attacco MITM (6.3). Tuttavia è impossibile ottenere informazioni sulle politiche di conservazione e protezione delle chiavi di cifratura.
- il requisito *1.1.2h* è parzialmente rispettato: sebbene non siano presenti Captcha al momento del login, sono previsti metodi per la protezione contro attacchi DoS, come descritto nel capitolo 5.3 (Rate limiting test). È stato impossibile interrogare le API di backup per ottenere informazioni in merito ad un recovery plan.
- il requisito *1.2.2* è parzialmente rispettato: infatti non viene indicato il periodo di supporto, né la frequenza delle patch, così come gli intervalli massimi tra release di sicurezza e il tempo obiettivo per il rilascio di patch critiche. In aggiunta, sebbene si possano consultare informazioni in merito a tutte le releases del firmware della Wyze Cam, non è presente la categorizzazione in feature o security. La policy di aggiornamento di Wyze prevede che sia previsto un anno di bug fixes, maintenance releases, workarounds o patches per bugs critici, come specificato alla pagina EOL del sito Wyze [37].

Il presente caso di studio ha dimostrato l'efficacia dell'approccio proposto, partendo da una serie di norme, tecnicizzandole ed andando successivamente ad identificare vulnerabilità concrete. Sebbene alcuni requisiti siano difficilmente tecnicizzabili o verificabili, l'eterogeneità degli aspetti analizzati ha consentito di ottenere risultati interessanti, evidenziando come dispositivi IoT apparentemente sicuri possano presentare vulnerabilità che compromettono potenzialmente la privacy e la sicurezza degli utenti.

Bibliografia

[1] Bettercap. https://www.bettercap.org/.

[2] Binwalk. https://github.com/ReFirmLabs/binwalk.

[3] BusyBox. https://busybox.net/.

[4] cve-bin-tool. https://github.com/intel/cve-bin-tool.

[5] Dnsmasq. http://www.thekelleys.org.uk/dnsmasq/.

[6] Ettercap. https://www.ettercap-project.org/.

[7] Ffmpeg. https://ffmpeg.org/.

[8] Isvp. https://jmichault.github.io/ipcam-100-dok/en/includes.en/html/.

[9] John the Ripper password cracker. https://www.openwall.com/john/.

[10] libcurl. https://curl.se/libcurl/.

[11] Live555 LiveMedia Library. http://www.live555.com/liveMedia/.

[12] Live555 (SM) CVEs. https://app.opencve.io/cve/?vendor=live555&product=streaming_media.

[13] Nmap. https://nmap.org/.

[14] Path Traversal Attack. https://owasp.org/www-community/attacks/Path_Traversal.

[15] Sasquatch. https://github.com/devttys0/sasquatch.

[16] SquashFS. https://docs.kernel.org/filesystems/squashfs.html.

[17] The Linux Kernel Archives. https://www.kernel.org/.

[18] uclibc. https://www.uclibc.org/.

[19] wpa_supplicant. https://w1.fi/wpa$_s$*upplicant*/.

[20] Wyze Firmwares Repository. https://github.com/kohrar/Wyze-Firmwares, 2025.

[21] Alessandro Brighenti. A framework for the security analysis of IoT devices, 2022.

[22] Rescorla Eric Dierks Tim. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, 2008. Request for Comments: 5246.

[23] Federal Communications Commission. FCC ID: 2AUIUWYZEC3F. https://fcc.report/FCC-ID/2AUIUWYZEC3F/, 2020.

[24] R. Lanphier-M. Westerlund M. Stiemerling H. Schulzrinne, A. Rao. RFC 7826: Real-Time Streaming Protocol Version 2.0. https://tools.ietf.org/html/rfc7826, 2016.

[25] J. Postel, J. Reynolds. Telnet Protocol Specification. RFC 854, 1983.

[26] Matteo Mariotti. Automated Framework for IoT Security Testing, 2023.

[27] Matteo Mariotti. scan. https://github.com/Matteo0301/Thesis-scripts/blob/70a65447712b535c74cfc45da659d32c3535ee5d, 2023.

[28] MITRE Corporation. Common Vulnerabilities and Exposures (CVE). https://cve.mitre.org/.

[29] Nmap. Script rtsp-url-brute. https://nmap.org/nsedoc/scripts/rtsp-url-brute.html.

[30] OpenSSL Software Foundation. OpenSSL. https://www.openssl.org/.

[31] Parlamento europeo e Consiglio dell'Unione Europea. Regolamento (UE) 2024/2847 del Parlamento e del Consiglio europeo del 18 settembre 2024 relativo ai requisiti orizzontali di cybersicurezza per i prodotti con elementi digitali e che modifica il regolamento (UE) 2019/1020 (regolamento sulla resilienza informatica). https://eur-lex.europa.eu/legal-content/IT/TXT/?uri=CELEX:32024R2847, settembre 2024.

[32] Simone Rossi. Firmware security analysis on IoT devices, 2024.

[33] Wireshark Development Team. Wireshark. https://www.wireshark.org/.

[34] Wyze Labs Inc. How do I manage automatic firmware updates on Wyze Cam v3. https://support.wyze.com/hc/en-us/articles/20367091002523-How-do-I-manage-automatic-firmware-updates-on-Wyze-Cam-v3.

[35] Wyze Labs Inc. Webcam Firmware Instructions. https://support.wyze.com/hc/en-us/articles/360041605111-Webcam-Firmware-Instructions.

[36] Wyze Labs Inc. Wyze Cam v3 Firmware Release Notes. https://support.wyze.com/hc/en-us/articles/360024852172-Release-Notes-Firmware#Accordion-wyze-cam-v3-firmware-32.

[37] Wyze Labs, Inc. Wyze End of Life Policy. https://www.wyze.com/pages/end-of-life-policy.

[38] Wyze Labs, Inc. Wyze app. https://www.wyze.com/pages/wyze-app, 2020.

[39] Wyze Labs, Inc. Wyze Cam v3. https://www.wyze.com/products/wyze-cam-v3, 2020.

[40] Wyze Labs Inc. Wyze Open Source Software. https://support.wyze.com/hc/en-us/articles/360012546832-Open-Source-Software, 2025.

[41] M. Pegourie-Gonnard Y. Nir, S. Josefsson. RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier. https://datatracker.ietf.org/doc/html/rfc8422, 2018.

Durante lo sviluppo di questo elaborato sono stati utilizzati strumenti di intelligenza artificiale generativa (Policy generale: uso di strumenti di intelligenza artificiale generativa, Università degli Studi di Trento, paragrafo 5).

Allegato A Tabella di mappatura dei requisiti tecnici

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	1	1	I prodotti devono essere progettati, sviluppati e prodotti in modo da garantire un livello adeguato di cibersicurezza in base ai rischi	• Analizzare (se possibile) il ciclo di sviluppo del prodotto (Secure Software Development Lifecycle) • Verificare la presenza di analisi dei rischi (con tabella rischio = impatto * probabilità) • Controllare se sono previsti/effettuati test di sicurezza di vario genere	Non analizzato poiché difficilmente tecnicizzabile.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	1	2a	I prodotti sono messi a disposizione senza vulnerabilità sfruttabili note	• Controllare che le tecnologie utilizzate non presentino vulnerabilità note (effettuare un confronto con un database contenente queste vulnerabilità, il quale viene tenuto aggiornato) • Controllare che la documentazione presenti certificazioni e/o risultati di test effettuati	1. Richiedere al produttore il file SBOM 2. Verificare che elenchi tutti i componenti (librerie, moduli, sistema operativo) con relativa versione e licenza 3. Usare uno strumento di vulnerability scanning che controlla, per ogni componente della SBOM, se la sua versione ha delle criticità (il confronto viene effettuato con il database NVD o un generico database CVE) 4. Riportare l'indice CVSS (Common Vulnerability Scoring System) per ogni componente confrontato, andando ad evidenziare quelli più critici 5. Controllare se sono state rilasciate patch per il componente in questione 6. Controllare quali ambiti coprono i test effettuati dal produttore e di quali certificazioni dispone il prodotto in questione, anche in merito alle eventuali vulnerabilità trovate al punto 3 e 4

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	1	2b	I prodotti sono messi a disposizione con configurazione sicura in modo predefinito e con la possibilità di effettuare il ripristino	• Analizzare le impostazioni di default (ad esempio: password robuste, porte chiuse, aggiornamenti automatici attivati) • Verificare se c'è la possibilità di effettuare il reset ai dati di fabbrica • Controllare che i servizi non necessari siano disabilitati (di default)	1. Utilizzare uno script o un tool (ad esempio ho trovato Selenium per il test automatico) che consenta di testare la complessità della password, inviando password ed analizzando la risposta 2. Controllare nella documentazione ufficiale se viene menzionata la parola "password", in modo da poter ottenere ulteriori informazioni in merito (lunghezza, caratteri speciali, ...) 3. Eseguire una scansione delle porte TCP/UDP esposte dal prodotto all'avvio, ad esempio con il tool nmap o simili 4. Controllare nella documentazione ufficiale se vengono menzionate parole riguardanti l'argomento porte, in modo da poter ottenere ulteriori informazioni in merito (quali porte sono esposte, quali servizi sono previsti, ...) 5. Intercettare le richieste HTTP/HTTPS che vengono effettuate, usando sniffer di rete come wireshark, con lo scopo di trovare chiamate a server di aggiornamento automatico (quindi qualcosa in stile polling). 6. Nuovamente, controllare la documentazione ufficiale e verificare se viene menzionato l'argomento "aggiornamenti automatici", in modo da poter ottenere ulteriori informazioni in merito (frequenza del controllo aggiornamenti, ...) 7. Controllare la documentazione ufficiale e verificare se viene menzionato l'argomento "reset ai dati di fabbrica", in modo da poter ottenere ulteriori informazioni in merito (come effettuarlo, ...) 8. Per quanto riguarda la verifica dei servizi non necessari disabilitati, oltre a controllare la documentazione come nei punti precedenti, si potrebbe (nel caso in cui il dispositivo sia online / disponga di una cli) vedere cosa è in ascolto sulla rete (usando ad esempio nmap/netstat come prima, oppure ps/top se è possibile usare la cli). In base ai risultati, classificarli come necessari o non necessari.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	1	2c	Garantire la gestione delle vulnerabilità tramite aggiornamenti di sicurezza (preferibilmente automatici) con possibilità di disattivarli e notificarli	• Verificare se è prevista la possibilità di effettuare gli aggiornamenti in modo automatico • Controllare se è possibile disattivare gli aggiornamenti automatici (se si, con che facilità) • Verificare in che modo gli aggiornamenti vengono notificati all'utente (notifiche push, notifiche che impediscono l'utilizzo del dispositivo, ...)	1. Intercettare le richieste HTTP/HTTPS effettuate automaticamente dal dispositivo, usando uno sniffer di rete (es. Wireshark, tcpdump,...) per verificare se il dispositivo contatta server di aggiornamento subito dopo l'avvio (come descritto prima). 2. Controllare nella documentazione ufficiale se viene menzionato l'argomento "aggiornamenti automatici", così da poter capire se gli aggiornamenti automatici sono abilitati di default e con quale frequenza viene effettuato il controllo. 3. Utilizzare strumenti come Selenium, curl, ... per poter effettuare test mediante (ad esempio) uno script che accede al pannello di configurazione, per poi cercare un'opzione/flag per disattivare gli aggiornamenti automatici. 4. Valutarne la complessità. In alternativa, analizzare la documentazione ufficiale e cercare termini come "disattiva aggiornamenti, per capire se e come è prevista la disattivazione da parte dell'utente. 5. Cercare nella documentazione ufficiale se e come viene notificato un aggiornamento all'utente, ad esempio tramite messaggi sullo schermo, popup, notifiche push, oppure se il dispositivo impedisce l'uso fino all'aggiornamento.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	1	2d	Garantire la protezione dall'accesso non autorizzato tramite adeguati controlli di login e in che modo viene verificata l'identità dell'utente	• Verificare in che modo viene implementata l'autenticazione (es. password che devono essere complesse, MFA, ...) • Controllare sistemi di gestione degli accessi e logging accessi non autorizzati. • Analizzare policy di gestione account e privilegi.p.	1. Verificare in che modo viene implementata l'autenticazione: se è disponibile un'interfaccia web o CLI, utilizzare uno script (ad esempio Selenium o expect, come descritto prima) che invia tentativi di login con credenziali errate o deboli. Osservare quindi se il sistema impone vincoli di sicurezza (es. lunghezza minima della password, caratteri speciali, ...). Analizzare la documentazione ufficiale cercando frasi come "password complexity" o "MFA"/"autenticazione a due fattori", per capire se sono previsti meccanismi di autenticazione avanzati. 2. Analizzare la documentazione ufficiale per individuare la presenza di un sistema di gestione degli accessi. Verificare se sono previsti account con permessi differenti, come ad esempio utenti normali e amministratori. Controllare inoltre se viene registrata l'attività di login, cercando parole chiave come "access/security logs" o "login audit". Se questi log sono accessibili, si può simulare un tentativo di accesso errato per vedere se l'evento viene registrato nei log o segnalato in qualche modo. 3. Simulare un attacco brute-force per verificare se il sistema applica contromisure, eseguendo tentativi di login errati con lo stesso account tramite strumenti di testing descritti sopra. Osservare se il sistema reagisce bloccando temporaneamente l'account, introducendo un ritardo crescente tra i tentativi o generando un log di qualche tipo. 4. Verificare se vengono specificate regole come il numero massimo di tentativi di login falliti prima del blocco, il timeout automatico della sessione dopo inattività e l'obbligo di cambiare le credenziali periodicamente. Si potrebbe confrontare i valori dichiarati con quelli raccomandati da standard noti, ad esempio NIST. 5. Consultare la documentazione o testare direttamente il comportamento del sistema per capire se esistono ruoli distinti e se l'accesso a funzionalità critiche è riservato solo agli utenti autorizzati. Se disponibile un'interfaccia, provare ad accedere con un utente non privilegiato e verificare se le funzioni amministrative sono nascoste o inaccessibili.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	1	2e	Prevedere riservatezza dei dati conservati/inviati/trattati tramite la crittografia (o altri mezzi tecnici)	• Verificare la presenza di crittografia dei dati conservati e in transito (ad esempio criptandoli con AES e inviandoli con TLS) • Controllare se vengono utilizzati protocolli sicuri per le comunicazioni • Controllare come avviene la gestione delle chiavi di cifratura.	1. Catturare il traffico di rete generato dal dispositivo durante il funzionamento utilizzando strumenti come Wireshark o tcpdump. 2. Analizzare i pacchetti per verificare che i dati sensibili non siano trasmessi in chiaro, identificando i protocolli usati e controllando che non siano non cifrati (es HTTP invece di HTTPS, ...). Successivamente esaminare i certificati TLS per versione e validità della cifratura. 3. Eseguire una scansione automatizzata verso le porte di rete del dispositivo usando strumenti come nmap con lo script ssl-enum-ciphers per raccogliere informazioni sulle versioni TLS supportate, le cipher suite disponibili e la configurazione di sicurezza, confrontando i risultati con best practice di sicurezza. 4. Estrarre informazioni sulla crittografia tramite API, interfacce web o endpoint accessibili, interrogando le risorse disponibili per rilevare configurazioni o report che indicano se i dati sono protetti da cifratura e quali algoritmi sono utilizzati. Analizzare le risposte (ma anche la documentazione) cercando parole chiave come “encryption”, “AES”, ... e valutare la presenza e il tipo di cifratura. 5. Interrogare le API o interfacce di gestione chiavi esposte dal dispositivo per verificare come avviene la conservazione e la protezione delle chiavi di cifratura, controllando se sono previste politiche di rotazione, backup sicuro e restrizioni di accesso. 6. Integrare un proxy Man-in-the-Middle (ad esempio mitmproxy o Burp Suite) che intercetti il traffico cifrato: questo verificherà che la connessione TLS fallisca in presenza di corretta validazione certificati.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	1	2f	Prevedere integrità dei dati, dei programmi e delle configurazioni da manipolazioni e modifiche non autorizzate, segnalando le corruzioni	• Verificare l'uso della firma digitale per i dati sensibili. • Utilizzare il log per tracciare tutte le modifiche che vengono effettuate (e da chi) • Utilizzare funzioni di hashing per effettuare check sull'integrità dei dati (specialmente quelli trasportati)	Non analizzato poiché richiederebbe reverse engineering o dump della memoria.
1	1	2g	Trattare solo dati adeguati, pertinenti e limitati a quanto necessario (minimizzazione dei dati).	• Verificare che i dati raccolti siano limitati alla funzionalità del prodotto (principio del privilegio minimo), cosi anche come i privilegi che vengono assegnati alle varie parti del software • E' presente documentazione della gestione dati?	Non analizzato poiché difficilmente tecnicizzabile.
1	1	2h	Proteggere la disponibilità delle funzioni essenziali dopo un incidente, anche contro attacchi DoS	• Verificare se sono presenti accorgimenti/sistemi per limitare attacchi DoS/DDoS, come i Captcha oppure metodi di rate limiting • Verificare se ci sono piani di resilienza (come server ridondanti) oppure piani di disaster recovery (per ripristinare la situazione iniziale dopo eventuali attacchi/emergenze)	1. Interrogare le API o l'interfaccia web per rilevare la presenza di sistemi anti DoS/DDoS, verificando nei metadati delle risorse critiche (login, API di scrittura, form) l'attivazione di CAPTCHA o flag di rate limiting. 2. Inviare un numero elevato di richieste sequenziali agli endpoint critici e analizzare le risposte HTTP. In caso di risposta Too Many Requests o errori di blocco temporaneo, è confermata l'efficacia del rate limiting. 3. Interrogare le API di backup, se presenti, per ottenere informazioni in merito a backup e recovery plan.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	1	2i	Ridurre al minimo l'impatto (negativo) che potrebbe avere il prodotto in questione sulla disponibilità dei servizi di altri dispositivi o reti	• Verificare in che modo il prodotto consuma risorse, sia in condizioni normali che in caso di errori • Controllare che il dispositivo non generi traffico di rete anomalo	Non analizzato poiché difficilmente tecnicizzabile.
1	1	2j	Limitare le superfici di attacco, comprese le interfacce esterne.	• Verificare che siano esposte solamente le interfacce necessarie (ovviamente con le dovute precauzioni) • Controllare che porte e protocolli non necessari siano disabilitati (principio del privilegio minimo anche qui), cosi come assicurarsi che la configurazione delle API eventualmente utilizzate avvenga in modo sicuro	Non analizzato poiché richiederebbe reverse engineering o dump della memoria.
1	1	2k	Ridurre l'impatto degli incidenti utilizzando meccanismi di attenuazione dello sfruttamento.	• Verificare la presenza di protezioni come ASLR, DEP, stack canaries (non conosco queste tecnologie, mi sono informato da internet) • Controllare che ci siano meccanismi di sandboxing e rate limiting, così come controllare che gli errori vengano gestiti in modo corretto	Non analizzato poiché richiederebbe reverse engineering o dump della memoria.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	1	2l	Fornire informazioni sulla sicurezza registrando/monitorando le attività effettuate, con possibilità di disattivazione da parte dell'utente	• Verificare se è presente un sistema di log delle operazioni sensibili (o delle operazioni in generale) e che questo sistema sia sicuro (dal punto di vista crittografico) • Controllare che l'utente possa attivare/disattivare il sistema di log descritto sopra	Non analizzato poiché richiederebbe reverse engineering o dump della memoria.
1	1	2m	Offrire possibilità all'utente di rimuovere permanentemente dati e impostazioni in modo sicuro e che possa trasferirli in modo sicuro	• Verificare se sono implementate funzioni di wipe sicuro dei dati e se l'utente può usufruirne • Controllare se esiste una funzione di esportazione sicura dei dati (in che modo li esporta, verso dove, . . .) • Controllare che i dati cancellati non siano recuperabili facilmente	Non analizzato poiché richiederebbe reverse engineering o dump della memoria.
1	2	1	I fabbricanti identificano e documentano vulnerabilità e componenti, riassumendole in una distinta base software. Questa deve includere le dipendenze di primo livello del prodotto	• Verificare la presenza di una Software Bill of Materials (SBOM) aggiornata, il cui formato deve essere leggibile da dispositivi automatici • Controllare se esiste un sistema per tracciare le vulnerabilità interne descritte nella SBOM	1. Interrogare l'API/endpoint dedicato per scaricare la SBOM in formato leggibile da dispositivi automatici (ad es. XML/JSON, ...): la SBOM deve essere conforme agli standard, disponendo dei campi obbligatori e dello schema JSON/XML (cosa da controllare). 2. Estrarre dalla SBOM la lista dei componenti e delle dipendenze (quantomeno di primo livello), controllando che ogni voce includa nome, versione e identificatori univoci. 3. Interrogare le API del sistema di gestione vulnerabilità (se presenti) per ciascun componente elencato nella SBOM, raccogliendo elenco di CVE e relativi punteggi.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	2	2	I fabbricanti correggono tempestivamente le vulnerabilità e rilasciano aggiornamenti di sicurezza separati dalla funzionalità.	• Verificare la politica di gestione degli aggiornamenti di sicurezza, controllando anche la frequenza di rilascio delle patch: per quanto tempo, ogni quanto tempo, ... • Controllare che gli aggiornamenti di sicurezza sono separati dagli aggiornamenti funzionali	1. Interrogare l'API o consultare la documentazione ufficiale per recuperare la policy di gestione degli aggiornamenti, individuando il periodo di supporto e la frequenza delle patch. 2. Estrarre dalla policy i valori specifici di inizio supporto, fine supporto, intervallo massimo tra release di sicurezza e tempo obiettivo per il rilascio di patch critiche, confrontandoli con i requisiti normativi (per esempio, entro 30 giorni dalla scoperta della vulnerabilità). 3. Interrogare l'API dei rilasci o sfogliare il changelog ufficiale per ottenere la lista cronologica di tutte le release. Parsare per ciascuna release i metadati di rilascio, tipo e identificativo univoco. Verificare se è specificato il tipo di update (ad esempio feature o security). 4. Effettuare una richiesta di download di un security update o scaricare il pacchetto dalla documentazione, quindi ispezionare il manifest interno / note di rilascio per accertare che il changelog contenga solo correzioni di sicurezza e non nuove funzionalità.
1	2	3	Vengono effettuate prove e riesami efficaci e periodici della sicurezza del prodotto	• Verificare se sono descritte le tipologie di test effettuati e la loro frequenza • Controllare report con risultati dei test • Accertarsi che esista un calendario ufficiale di riesami	Non analizzato poiché difficilmente tecnicizzabile.
1	2	4	Pubblicare e divulgare pubblicamente le informazioni sulle vulnerabilità risolte	• Esaminare le Release Notes pubblicate sul sito del produttore: devono contenere descrizione vulnerabilità, prodotti interessati, gravità e istruzioni di mitigazione • Controllare date di pubblicazione in relazione alla disponibilità della patch	Non analizzato poiché difficilmente tecnicizzabile.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
1	2	5	Divulgazione in modo coordinato le vulnerabilità	• Verificare che sia descritto il modo in cui si intende gestire la divulgazione coordinata delle vulnerabilità (ad esempio tramite database delle vulnerabilità, sito/forum ufficiale, apposita pagina, ...)	Non analizzato poiché difficilmente tecnicizzabile.
1	2	6	Devono esserci misure prese per facilitare la segnalazione di vulnerabilità e contatto dedicato	• Cercare nel manuale o sul sito l'indirizzo e-mail o link dedicato per segnalazioni • Verificare la presenza di istruzioni chiare e tempi di risposta stimati	Non analizzato poiché difficilmente tecnicizzabile.
1	2	7	Meccanismi per distribuire in modo sicuro gli aggiornamenti (inclusi automatici)	• Controllare la documentazione in merito alla sicurezza nella distribuzione degli aggiornamenti e modalità di fallback • Verificare che siano descritti i criteri di roll-out e i requisiti di autenticazione per ottenere l'aggiornamento • Accertarsi della presenza di istruzioni per l'attivazione/disattivazione del meccanismo automatico	Non analizzato poiché difficilmente tecnicizzabile.
1	2	8	Garanzia di disponibilità tempestiva e gratuita degli aggiornamenti di sicurezza (con messaggi di avviso agli utilizzatori)	• Controllare i "Security Bulletins" per i tempi di rilascio (ad es. entro X giorni dalla scoperta) • Verificare esempi di comunicazioni (e-mail, notifiche) in merito alla distribuzione degli update	Non analizzato poiché difficilmente tecnicizzabile.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
7	/	1a	Descrizione generale del prodotto con elementi digitali: finalità prevista	• Controllare la descrizione delle finalità previste per il prodotto	Non analizzato poiché difficilmente tecnicizzabile.
7	/	1b	Descrizione delle versioni software rilevanti per la conformità ai requisiti essenziali di cibersicurezza	• Assicurarsi che sia mantenuta una SBOM aggiornata con tutte le librerie (e relative versioni) • Controllare la presenza di un registro delle release con indicazione delle patch di sicurezza che ogni release include/prevede	Non analizzato poiché difficilmente tecnicizzabile.
7	/	1c	Documentazione hardware: fotografie, illustrazioni e schemi delle caratteristiche esterne e interne del prodotto	• Controllare se sono presenti foto/illustrazioni/schemi delle viste esterne e interne • Prestare particolare attenzione a porte, antenne e numero seriale	Non analizzato poiché difficilmente tecnicizzabile.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
7	/	1d	Informazioni e istruzioni dettagliate per l'utilizzatore, inclusi installazione, configurazione e aggiornamenti (basandosi su allegato II)	Controllare la presenza di: • Dati fabbricante (nome/marchio, contatti, sito) • Punto unico segnalazioni e modo in cui il fabbricante divulga in modo coordinato le vulnerabilità • Identificativo univoco (modello e numero seriale) • Finalità, funzionalità chiave e rischi d'uso • Tipo di supporto sicurezza + data fine assistenza • Istruzioni per: avvio sicuro, installazione aggiornamenti, smantellamento sicuro (eliminazione sicura dei dati degli utilizzatori), disattivazione degli aggiornamenti automatici e integrazione con altri sistemi	Non analizzato poiché difficilmente tecnicizzabile.
7	/	2a	Documentazione di progettazione, sviluppo e architettura del sistema: componenti, flussi dati e dipendenze software	• Controllare se l'architettura viene descritta mediante diagrammi di componenti (component diagram) e sequence (flussi dei dati, diagrammi BPMN, API utilizzate e routes/endpoint esposti, dipendenze software) • Assicurarsi che punti di integrazione (protocolli usati per la comunicazione, interfacce, formati dei dati, meccanismi di autenticazione) e librerie di terze parti utilizzate siano elencati e descritti	Non analizzato poiché difficilmente tecnicizzabile.

All.	Parte	Punto	Requisito	Metodo per implementarlo	Nello specifico
7	/	2b	Gestione delle vulnerabilità: SBOM, policy di disclosure e canale dedicato per la segnalazione e aggiornamenti	Assicurarsi che sia: • Descritta la SBOM nella documentazione • Definito il processo di rilascio degli aggiornamenti (ad esempio OTA firmati e distribuiti via HTTPS, ...) • Presente un link/contatto per la segnalazione di vulnerabilità • Descritto il modo in cui si intende gestire la divulgazione coordinata delle vulnerabilità (ad esempio tramite database delle vulnerabilità, sito/forum ufficiale, apposita pagina, ...)	Non analizzato poiché difficilmente tecnicizzabile.
7	/	2c	Processi di produzione, monitoraggio e convalida dei controlli qualità	Verificare che siano presenti: • Sezioni che elencano le fasi di assemblaggio e collaudo (dall'inizio alla fine) • Elenco degli strumenti/metodologie impiegate • Descrizione delle procedure di gestione dei log • Date e versioni del software/hardware testato	Non analizzato poiché difficilmente tecnicizzabile.
7	/	3	Valutazione completa dei rischi di cibersicurezza e definizione di un piano di mitigazione	• Verificare che sia stato effettuato un Risk Assessment con matrice impatto x probabilità • Controllare che siano stati allineati i controlli richiesti dall'Allegato I parte I ai rischi identificati • Controllare che sia stato redatto un piano di mitigazione	Non analizzato poiché difficilmente tecnicizzabile.

Allegato B Scripts

B.1 attackRTSP

```
#!/usr/bin/env bash
set -euo pipefail

# IP target
IP="192.168.2.2"
OUTPUT_FILE="resultsAttackRTSP.txt"

# Pulizia output all'inizio
: > "${OUTPUT_FILE}"

print_header(){
  local header="$1"
  echo -e "\n==================== ${header} ====================\n" \
    | tee -a "${OUTPUT_FILE}"
}

run_command(){
  local msg="$1"
  local cmd="$2"

  print_header "${msg}"
  echo "\$ ${cmd}" | tee -a "${OUTPUT_FILE}"
  eval "${cmd}" 2>&1 | tee -a "${OUTPUT_FILE}"
}

attack(){
  print_header "Starting RTSP attack on port 554"
  PORTS="554"

  for p in ${PORTS}; do
    run_command "Testing RTSP port ${p}" \
      "nmap -Pn -sV -p${p} --script rtsp-url-brute ${IP}"
  done
}

# Viene lanciato l'attacco
attack

echo -e "\nDone. Results in ${OUTPUT_FILE}\n"
```

B.2 injection

```
#!/usr/bin/env bash
set -euo pipefail

IP="192.168.2.2"
AUTH="..."
OUT="injection.txt"

: > "$OUT"
echo "=== Path Traversal Test (/etc/passwd) ===" | tee -a "$OUT"
printf 'DESCRIBE rtsp://'"$IP"'/../../../etc/passwd RTSP/1.0\r\nCSeq: 1\r\nAuthorization: Basic '"$AUTH"'\r\n\r\n' \
  | nc "$IP" 554 2>&1 | tee -a "$OUT"
echo >>"$OUT"

echo "=== Header Injection Test ===" | tee -a "$OUT"
printf 'OPTIONS rtsp://'"$IP"'/ RTSP/1.0\r\nCSeq: 2\r\nUser-Agent: InjectTest\r\nX-Injected: bar\r\n\r\n' \
  | nc "$IP" 554 2>&1 | tee -a "$OUT"

echo -e "\nDone. Results in $OUT"
```

B.3 test.sh

```
#!/bin/sh

ping -c 1 8.8.8.8
ping -c 1 -p $(echo -n "wyzecam_test" | xxd -p | tr -d '\n') 8.8.8.8
ping -c 1 20.20.20.20

# HTTP GET
wget -O /dev/null "http://example.com/factory_mode_proof" 2>/dev/null
```

B.4 rateLimitingTest

```
#!/usr/bin/env python3
import os
import time
import json
import requests

# Parametri
URL = "https://auth-prod.api.wyze.com/api/user/login"
EMAIL    = 'asgsdfgasgf@example.com' # casuale
PASSWORD = '£$WTGJQ$£TJG£sdagsdfh' # casuale
KEY_ID   = '042609823048602'
API_KEY  = '439680234986034'

MAX_ATTEMPTS = 200
DELAY        = 0.2  # secondi

headers = {
    "Content-Type": "application/json",
    "Accept":       "application/json",
}

payload = {
    "email":    EMAIL,
    "password": PASSWORD,
    "key_id":   KEY_ID,
    "api_key":  API_KEY
}

print(f"Flooding {URL} for rate-limit test (max {MAX_ATTEMPTS} attempts)\n")

for i in range(1, MAX_ATTEMPTS + 1):
    resp = requests.post(URL, headers=headers, json=payload)
    code = resp.status_code

    # Si ferma al 429
    if code == 429:
        print(f"[{i:03d}] RATE LIMITED --> HTTP {code}")
        break

    # Logga gli altri errori
    if code != 200:
        print(f"[{i:03d}] HTTP {code} {resp.reason}")
    else:
        print(f"[{i:03d}] Unexpected success (200 OK)")

    time.sleep(DELAY)
else:
    print(f"\nNessun 429 rilevato dopo {MAX_ATTEMPTS} tentativi")
```

Ringraziamenti

Al prof. Crispo e al dott. Ramponi per avermi supportato durante la stesura di questo elaborato.

Ai miei genitori, Filippo e Giovanna, che mi hanno sempre sostenuto in tutto, ci sono sempre stati e non hanno mai smesso di credere in me. Senza di voi non avrei potuto raggiungere questo traguardo: grazie per come mi avete cresciuto e per non avermi mai fatto mancare nulla.
A mia sorella Anita, che con la sua gentilezza e premura mi ha sostenuto durante questo percorso, dandomi il suo parere anche nei momenti più tosti. È impossibile trovare qualcuno come te e sicuramente sei la miglior sorella che potessi desiderare.
A mio fratello Leo, che mi ha strappato sorrisi e colorato giornate, scherzando e ridendo spensieratamente, senza mai curarsi del parere degli adulti.
Alla nonna Lina, che mi ha sempre voluto un bene sincero in tutti questi anni, interessandosi e chiedendomi di tutto. Mai una volta è mancato il classico "fìn ché se vedémo se tira 'vanti", emblematico delle nostre telefonate.
Al nonno Lino, alla nonna Concettina e allo zio Tony, la cui saggezza mi ha sempre ispirato, così come la loro serenità e la loro gioia spontanea, autentici in tutto ciò che mi hanno sempre tramandato.
Agli zii Nicola e Alessandra, che hanno sempre avuto fiducia in me, aiutandomi nei momenti di bisogno ed insegnandomi moltissimo senza aspettarsi nulla in cambio.
Agli zii Paolo e Silvia e Margherita, che non mi hanno mai negato una parola, contagiandomi sempre con la loro felicità.
A Martina, che è come una sorella per me, mi è stata accanto nei momenti migliori e nei peggiori e mai si è tirata indietro. Con la sua onestà, il suo incoraggiamento e la sua benevolenza è senza dubbio la migliore amica che potessi desiderare. Grazie per non essertene mai andata.
A Francesca, personificazione della gentilezza, che riesce a farmi sorridere anche quando tutto il resto sembra tetro. Grazie per essere tornata nella mia vita, per il sostegno che mi dai ogni giorno, per le emozioni che mi fai vivere e per il forte legame che ci unisce, tanto intenso quanto unico.
Ad Elena, con cui ho condiviso questo percorso universitario dal giorno zero. L'esserci sostenuti a vicenda davanti ogni ostacolo ci ha permesso di arrivare fin qui: grazie per non esserti mai tirata indietro e per non avermi mai chiuso la porta in faccia.
Ad Enrico, che mi ha accompagnato durante tutta la vita, mi ha permesso di vedere molti aspetti di essa con occhi diversi dandomi preziosi pareri e saggi consigli. Grazie per essere sempre stato tanto sincero quanto disponibile durante tutti questi anni.
A Gemma, che con la sua disponibilità, la sua pazienza e la sua ironia unica mi ha spesso motivato e supportato, anche nei giorni più tristi.
A Gioia, che indipendentemente dalla situazione e dal periodo non mi ha mai negato un sorriso e una parola, si è sempre interessata nei confronti di ogni cosa e mi ha dato il coraggio di andare avanti.
A Marco, amico d'infanzia (da ciò che si dice), con cui ho condiviso infiniti ricordi e che mi ha aiutato a staccare nelle situazioni più complesse, passando dalla pesante serietà alla leggera Serie A.

A Leo(nardo), che mi ha visto crescere e cambiare, che mi ha sempre supportato ed ascoltato senza pretendere nulla in cambio, non si è mai tirato indietro nel momento del bisogno. Grazie per tutti i momenti condivisi: dalle corse, alle esplorazioni della Ronda, a tutte le cazzate sparate.
A Francesco, personaggio unico nel suo genere, che dietro alla sua maschera nasconde sincerità e gentilezza, interessandosi e facendo domande quando tanti altri sorvolerebbero. Grazie per essere stato al mio fianco anche nei periodi più duri.
Ad Alessandro, compagno di calcio, di scuola, di stadio, di gruppo, di compagnia, di esperienze vissute: grazie per tutte le chiacchierate che non sono mai mancate.
Ad Eric, per la tua onestà e la tua cortesia con cui mi hai sempre trattato. Grazie per aver avuto il coraggio di dirmi ciò che pensavi anche in momenti scomodi.
A Luca, per aver reso più felici e leggeri i diversi periodi che abbiamo condiviso durante la nostra vita: grazie per essere sempre stato così aperto e scherzoso, ma allo stesso tempo per non essertene mai andato.
A Tommaso, con cui ho condiviso pensieri, timori, successi e fallimenti: grazie per le lunghe chiacchierate in merito a qualsiasi possibile argomento (serio o divertente che fosse) e per le infinite cazzate sparate. Mai avrei immaginato di poter trovare qualcuno così simile a me.
A Giulio, che mi ha visto ridere e piangere, sapendo sempre cosa dirmi. Grazie per tutti gli spunti di riflessione che mi hai dato in questi anni, per non avermi mai negato un confronto e per essere sempre stato così disponibile.
A Jacopo, la precisione in persona ma un tenerone dentro. Grazie per tutti i bei momenti condivisi, per la gentilezza e la purezza d'animo che ti ha sempre contraddistinto, mi hai insegnato molto.
A Giulia, per il tuo modo unico di interfacciarti con la gente, per come ci fai sentire a nostro agio indipendentemente dal contesto. Grazie per aver reso più colorati e sorridenti questi anni di università.
A Matilde, che ha sopportato me e il mio accento per tre lunghi anni, ma che allo stesso tempo mi ha sempre dato un parere, specialmente quando più ne avevo bisogno.
A Cristiano, guru della finanza e della tecnologia, uno spasso nella serietà. Grazie per aver rallegrato le giornate a Trento e per tutte le opinioni che ci siamo condivisi.
A tutti i ragazzi del collegio, in particolar modo Dani, Lore, Frigo, Track, Tommy, Chesi, Albi, Matte, Gabri ed il mitico Roman, con cui ho condiviso le mie giornate tridentine, ridendo, scherzando, condividendo e parlando.
A chi se n'è andato, grazie per aver contribuito a scrivere una pagina della mia vita.