POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Riprogettazione sicura dell'applicazione e implementazione del protocollo di comunicazione per il sistema embedded di un rover

Relatore
prof. Antonio Lioy

Candidato
Marta Cattaneo

Supervisore aziendale
MCA Engineering S.R.L.
ing. Leonardo Forese

Anno accademico 2022-2023

Sommario

Negli ultimi anni la guida autonoma ha fatto grandi progressi grazie all'evoluzione delle tecnologie di intelligenza artificiale, di sensoristica e di elaborazione dei dati in tempo reale. Ci sono stati importanti investimenti da parte delle aziende automobilistiche e di tecnologia per sviluppare veicoli in grado di guidare da soli, senza l'intervento umano. Nonostante i progressi, ci sono ancora sfide importanti da affrontare prima che la guida autonoma possa diventare una realtà ancora più concreta. Tra queste, riveste un ruolo importante la natura sempre più connessa di questi veicoli, che ne determina l'incremento delle vulnerabilità da attacchi informatici e dei rischi che ne derivano. Per prevenire questi rischi, bisogna adottare delle misure di sicurezza sempre più avanzate, senza le quali, un hacker, potrebbe arrivare anche ad assumere il pieno controllo del veicolo.

In questo contesto, si sviluppa un progetto, nato all'interno dell'azienda MCA, il cui obiettivo è realizzare un rover terrestre in grado di muoversi autonomamente. L'idea nasce dal lavoro di tesi degli ingegneri L. Forese e A. Caliò, dal titolo "Deep Learning-Based Real-Time Detection and Single-Object Tracking on a GPU based embedded device". A questo scopo è stato realizzato un software, per un sistema embedded, che utilizza l'intelligenza artificiale per effettuare il riconoscimento e il tracciamento degli oggetti, utilizzando una rete neurale pre-trained e una fotocamera collegata alla scheda Nvidia Jetson Nano.

La naturale conseguenza è che, anche se autonomo, il sistema deve essere in grado di comunicare da remoto con l'utente che ne ha il controllo. Questo, deve essere in grado di vedere in tempo reale ciò che viene ripreso dalla videocamera, e scegliere la modalità con cui il rover deve lavorare, tra quelle disponibili. La sfida risiede nel far sì che questa comunicazione avvenga in sicurezza, ma cercando di ottenere una prestazione a bassa latenza.

Dopo un'attenta analisi del software di base, condotta grazie all'ausilio di strumenti automatici, standard e linee guida, è seguita una fase di riprogettazione del codice con lo scopo di eliminare le vulnerabilità potenzialmente utilizzabili da un utente malintenzionato. In particolare sono state confrontate le tecniche utili all'analisi di codice, tra cui analisi statica e l'analisi dinamica, effettuate con strumenti automatici e non. Grazie a questo, è stato scelto di procedere prima con l'utilizzo di tecnologie automatiche per l'analisi statica e, in seguito, con una revisione manuale, realizzata grazie alla conoscenza del funzionamento del software e all'utilizzo di linee guida utili allo scopo. I risultati hanno portato ad effettuare dei cambiamenti all'interno del codice con lo scopo di renderlo più resistente ad attacchi esterni.

In seguito, sono state vagliate le soluzioni migliori, per prestazioni e sicurezza, in grado di realizzare un sistema capace di comunicare attraverso reti pubbliche e non sicure. È stata effettuata un'attenta analisi delle soluzioni possibili, valutandone per ognuna i vantaggi e gli svantaggi, alla ricerca di quella più adatta alle esigenze del progetto. Per la scelta sono stati presi in considerazione sia il livello di sicurezza dei protocolli e dei sistemi in esame, sia i risultati ottenuti nella

velocità di trasmissione. Sono anche state studiate le maggiori vulnerabilità del panorama di cybersecurity mondiale, in modo da implementare, fin dall'inizio le tecniche necessarie a mitigarne i rischi. Durante lo sviluppo inoltre è stato possibile verificare l'efficacia di tutte queste scelte e, in alcuni casi, valutare la necessità di ulteriori misure.

Il risultato è stata l'implementazione di un'applicazione web, servita da un server ospitato sullo stesso sistema, accessibile tramite browser. L'utente comunicherà utilizzando il protocollo sicuro TLS, con mutua autenticazione, e il protocollo applicativo HTTP, ma con soluzioni utili a rendere minima la latenza nella trasmissione.

Nella fase finale è stata valutata l'effettiva sicurezza delle soluzioni implementate, tramite l'utilizzo di strumenti automatici e tecniche di penetrazione dell'applicazione. In particolare si è cercato di testare le soluzioni adottate, ma anche di agire come potrebbe fare un hacker. Dopo una prima raccolta di informazioni, realizzata utilizzando strumenti di scanning automatici e messaggi diretti con il server, sono state valutate le vulnerabilità trovate, evidenziando anche la conseguente corretta configurazione delle soluzioni di sicurezza analizzate precedentemente.

Ringraziamenti

Vorrei ringraziare il mio tutor, Leonardo Forese, che ha creduto in me e mi ha supportato fin dall'inizio di questo progetto.

Un ringraziamento speciale va anche alla mia famiglia, ai miei amici, a Rocco e a tutte le persone che mi sono state accanto, in un modo o nell'altro, durante questi anni, perché, senza il loro sostegno, non sarei arrivata fino a questo momento.

Indice

Capitolo 1

Introduzione

1.1 Azienda

MCA è una società europea di ingegneria e consulenza ad alta tecnologia che opera in tutti i settori industriali e dell'innovazione. Nasce in Francia e apre la prima filiale italiana a Torino nel 2016. Ad oggi offre alla realtà torinese servizi di consulenza in campo aerospaziale, biomedicale, energetico, dell'automazione industriale e, soprattutto, automobilistico.

1.2 Contesto

Il mondo automotive degli ultimi anni si è focalizzato molto sulla digitalizzazione e sull'automazione. In particolare sono tanti gli studi e i progetti che mirano a creare veicoli capaci di muoversi autonomamente, senza l'ausilio di conducenti o di controlli remoti. Negli ultimi anni sono stati fatti passi enormi in questo settore.

L'aumento di componenti connesse, elettriche o elettroniche, ha però aumentato considerevolmente la superficie d'attacco attraverso la quale potenziali malintenzionati possono prendere controllo del sistema o arrecargli danni. Con le innovazioni è quindi aumentata la consapevolezza di dover integrare nella produzione anche la gestione della sicurezza del veicolo, non solo più a livello fisico, ma anche a livello informatico. Per le aziende che lavorano in questo campo è diventato essenziale integrare la cybersecurity fin dalle primissime fasi di progettazione e lungo tutta la vita del prodotto, aderendo ai principi del security by design. Questa necessità si riflette anche lungo tutta la supply chain.

In questo contesto, prende vita il progetto di MCA, ovvero la realizzazione di un software che permetta ad un veicolo terrestre di tipo rover di viaggiare autonomamente, senza più l'ausilio di controlli da remoto. L'idea nasce grazie al lavoro svolto dall'ingegnere Leonardo Forese, laureato al Politecnico di Torino e attualmente Business Developer presso MCA, per la sua tesi di laurea magistrale. Il lavoro, realizzato nel 2020 insieme all'ingegnere Andrea Caliò presso l'Universidad Nacional de Còrdoba, ha titolo 'Deep Learning-Based Real-Time Detection and Single-Object Tracking on a GPU based embedded device [1]. Lo studio si è concluso con lo sviluppo di un software per un sistema integrato che utilizza l'intelligenza artificiale per riconoscere e tracciare differenti oggetti attraverso la telecamera.

1.3 Obiettivo

Il primo obiettivo di questa tesi è quello di rendere sicuro il codice già scritto, adottando tecniche di analisi e mitigazione delle minacce. Questo step è necessario per intraprendere un processo di sviluppo conforme agli standard della cybersecurity. Partendo dal software riprogettato in modo sicuro, è prevista l'implementazione di un protocollo sicuro che permetta la comunicazione

remota tra il rover e l'utilizzatore. Quest'ultimo non dovrà guidare il rover, ma essere in grado di visualizzare le immagini riprese e comunicare al veicolo la modalità in cui dovrà lavorare. Infine è obiettivo di questa tesi analizzare il software in questo stadio di sviluppo, utilizzando tecniche di penetration testing.

1.4 Struttura

Di seguito si elencano gli step attraverso i quali è stato condotto il lavoro:

- Studio del codice e dell'architettura del progetto;
- Studio della documentazione relativa allo sviluppo sicuro di codice e alla cybersecurity in campo automotive;
- Analisi e re-design sicuro del codice già esistente;
- Implementazione di un protocollo sicuro che permetta la comunicazione con l'esterno;
- Penetration testing e analisi dei risultati ottenuti

Capitolo 2

Background

Nell'ultimo decennio l'industria in generale è andata incontro ad una digitalizzazione sempre più importante, cosa che ha favorito l'aumento degli attacchi di tipo informatico. Infatti, la crescita del numero di componenti connesse, ha influito nell'aumento della superficie vulnerabile esposta ad eventuali malintenzionati.

Ogni giorno vengono scoperti nuovi e pericolosi attacchi, pronti a minare la sicurezza e la privacy di persone, aziende e stati. Per questo motivo si è resa necessaria l'adozione di misure di sicurezza per la protezione da tali minacce.

Purtroppo è impossibile creare qualcosa di inattaccabile, poiché nuove tecniche in grado di sfruttare vulnerabilità vengono scoperte ogni giorno, anche da persone male intenzionate. Quello che si può fare, però, è adottare tutte le misure necessarie a proteggersi da ciò che si conosce, in modo da anticipare e prevenire la maggior parte degli attacchi. In alcuni casi non ci si potrà proteggere, ma si potranno mitigare gli effetti. Per ciò che ancora non si conosce, invece, è opportuno adottare tecniche di continuo monitoraggio in modo da identificare eventuali attacchi in corso e aumentare le misure di sicurezza nel momento in cui nuovi attacchi vengono scoperti. Per tutti questi motivi sempre più dispositivi hanno la necessità di includere la gestione della cybersecurity all'interno del proprio ciclo di vita, a partire dallo sviluppo.

2.1 Cybersecurity e automotive

Il settore automobilistico, nell'ultimo decennio, si è indirizzato verso la produzione di veicoli sempre più connessi e autonomi. Questo aumento ha portato ad un'esposizione maggiore a minacce di tipo informatico con conseguenze anche gravi.

Lo standard SAE J3016 [2] definisce sei livelli di automazione per i veicoli su strada da cui i seguenti:

- Livello 4 di automazione si riferisce a veicoli completamente automatizzati che sono in grado, grazie alla moltitudine di sensori, di effettuare tutte le funzioni di guida sotto certe condizioni [3], cioè con operational design domain definito;
- Livello 5 di automazione, invece, è in grado di effettuare tutte le funzioni di guida autonomamente e in qualunque condizione [3].

L'obiettivo è quello di inglobare il sistema in un veicolo di tipo rover che rientri in una di queste due categorie.

I veicoli autonomi sono difficili da inglobare in un modello di riferimento data la grande varietà di tipologie e di produttori. Per analizzare i problemi di sicurezza e fornire la protezione necessaria, c'è bisogno di identificare gli asset sui quali lavorare. Nel caso dei veicoli autonomi questi sono molto diversificati. ENISA, l'agenzia dell'unione europea per la cybersecurity, ha realizzato uno

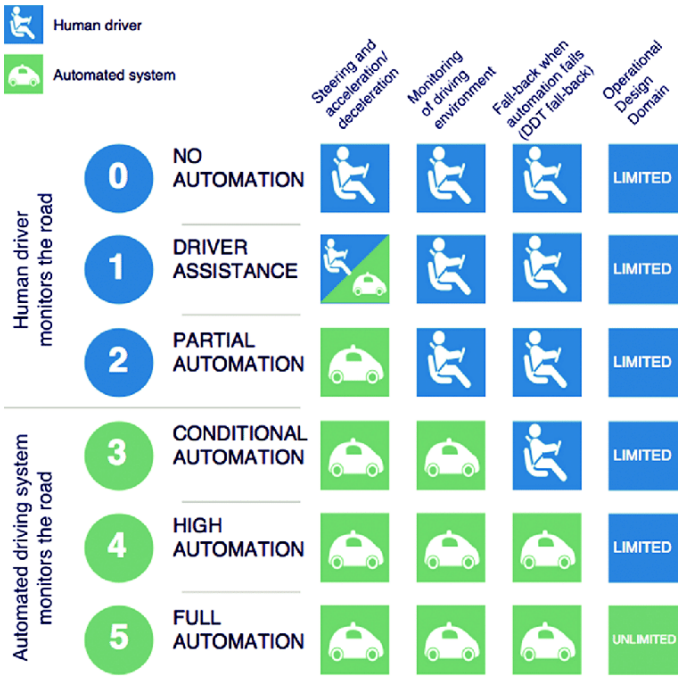

Figura 2.1. Classificazione dei veicoli autonomi secondo lo standard SAE J3016 (fonte: Research Gate).

studio sui veicoli smart [3], di cui i veicoli autonomi in esame fanno parte. In questo studio viene definita un'idea generale di quelli che possono essere gli asset di un veicolo nel momento in cui si deve analizzare la sua sicurezza. Si riporta l'elenco nell'annesso A.

Sulla base di queste informazioni sarà più facile identificare gli asset interessati, in base ai quali è possibile determinare le minacce a cui è esposto. Questo è un passo cruciale in ogni ciclo di sviluppo orientato alla sicurezza.

Nello stesso studio sono state identificate delle 'good practices' per la sicurezza delle auto intelligenti che dovrebbero essere implementate per proteggere sia le auto che i sistemi back-end associati.

Tra tutte queste misure si identifica la necessità di implementare un sistema di rilevamento delle intrusioni, IDS. Questo consentirebbe il rilevamento di attività dannose o di violazioni delle politiche sia a livello di veicolo, che di back-end. Sempre in questo contesto bisogna mantenere dei registri di audit, ma proteggerli in modo da evitare che entità non autorizzate possano accedervi.

È importante anche proteggere le comunicazioni remote e le interfacce attraverso meccanismi di mutua autenticazione e di controllo degli accessi. Questo può impedire l'accesso ai veicoli, proteggere l'integrità e l'autenticità di tutte le comunicazioni interne ed esterne, tra i veicoli e tutte con le diverse entità con cui è necessario comunicare.

Si tratta di un'ulteriore misura identificata da ENISIA garantire che la modalità di funzionamento più sicura sia quella di default. Si dovrebbe inoltre garantire l'autenticità e l'integrità del software prima che questo venga installato, per esempio attraverso un secure boot, per assicurare che venga utilizzato solo software legittimo. Così com'è necessario proteggere gli aggiornamenti del firmware per evitare che questo venga manipolato.

Viene anche identificato come necessario proteggere tutti i dati sensibili crittografandoli, in modo da impedirne la divulgazione ad entità illegittime. Inoltre, viene consigliato l'utilizzo di crittografia autenticata, con schemi e protocolli standardizzati, ampiamente considerati sicuri, per evitare che i dati personali vengano modificati e garantire la loro riservatezza.

2.1.1 Standard e Regolamentazioni in campo automotive

Per poter omologare un veicolo connesso, i car maker (OEM), devono seguire le regole predisposte nelle normative UNECE R155 [4] e R156 [5]. Per poter rispettare questo regolamento, il costruttore deve imporre ai propri fornitori di essere a sua volta conformi ad esso.

È stato recentemente pubblicato uno standard dedicato alla cyber security dei sistemi elettrici ed elettronici dei veicoli: l'ISO 21434 [6]. Il fornitore di un componente del veicolo, se rispettoso di questo standard, sarà a sua volta conforme alla normativa R155, poiché potrà fornire all'OEM la documentazione necessaria per omologare il veicolo.

- UNECE R155: requisiti generali per la sicurezza informatica e la sicurezza del sistema di gestione del veicolo [4];
- UNECE R156: disposizioni per l'aggiornamento software e gli aggiornamenti software del veicolo [5];
- ISO 21434: requisiti per la gestione dei rischi di sicurezza informatica riguardanti concept, sviluppo, produzione, manutenzione e smantellamento di sistemi E/E nei veicoli [6].

2.1.2 Intelligenza artificiale nella guida autonoma

L'intelligenza artificiale gioca un ruolo cruciale nella guida autonoma. Questa permette ai veicoli di percepire, comprendere e agire rispetto l'ambiente circostante. Si elencano di seguito alcuni modi in cui in cui l'AI viene utilizzata nella guida autonoma:

- Percezione: Gli algoritmi di intelligenza artificiale vengono utilizzati per elaborare i dati provenienti da vari sensori, come telecamere, LIDAR e radar, per identificare e seguire gli oggetti nell'ambiente di guida.
- Processo decisionale: Gli algoritmi di intelligenza artificiale vengono utilizzati per analizzare i dati provenienti dai sensori e prendere decisioni sulle azioni del veicolo, come sterzare, accelerare e frenare.
- Pianificazione del percorso: Gli algoritmi di intelligenza artificiale vengono utilizzati per generare percorsi di guida sicuri ed efficienti, tenendo conto del tracciato stradale, delle condizioni del traffico e di altri fattori ambientali.
- Cruise control adattivo: Gli algoritmi di intelligenza artificiale vengono utilizzati per controllare la velocità del veicolo e mantenere una distanza di sicurezza dagli altri veicoli.
- Rilevamento e classificazione degli oggetti: Gli algoritmi di intelligenza artificiale vengono utilizzati per rilevare e classificare gli oggetti presenti nell'ambiente, come veicoli, persone e segnali.

L'uso di algoritmi di intelligenza artificiale all'interno di veicoli presenta vari rischi. Prima di tutto questi veicoli utilizzano sistemi software complessi che possono essere vulnerabili a hacking, malware e altri attacchi informatici. Inoltre i veicoli autonomi si affidano a sensori per rilevare e reagire all'ambiente circostante. Un malfunzionamento o un guasto di questi sensori può portare a risultati impredicibili. Questi sistemi utilizzano algoritmi per prendere decisioni e controllare i movimenti. Gli errori in questi ultimi possono portare a comportamenti inaspettati e potenzialmente pericolosi.

Creando dei veicoli intelligenti con l'uso del machine learning poi, ci si deve approcciare ad una nuova minaccia: gli adversial attack. Questa è una categoria di attacchi che mira ad ingannare i modelli di apprendimento inducendoli a fare previsioni errate, quindi compromettendo il normale e sicuro funzionamento del sistema. In questa categoria si possono trovare:

- Adversial example: si tratta di creare input specificamente progettati per ingannare un modello di intelligenza artificiale e indurlo a fare previsioni errate.

- Data poisoning: comporta l'alterazione dei dati di addestramento utilizzati per formare un modello di intelligenza artificiale, in modo da distorcerlo e far si che faccia previsioni errate.
- Model inversion: si tratta di effettuare reverse engineering di un modello di AI in modo da capire come questo opera e creare input che portino a risultati specifici.
- Side-channel attack: si tratta di attaccare un sistema che utilizza l'AI sfruttando le vulnerabilità dell'hardware o del software sottostante, piuttosto che il modello stesso.

Per questo motivo, nonostante l'utilizzo di modelli pretrained apporti molti vantaggi, è essenziale essere consapevoli che si dipende da terze parti, che devono a loro volta esse consci delle minacce e mitigarne i possibili rischi. È particolarmente importante che i modelli siano propriamente messi in sicurezza durante il training. È necessario inoltre eseguire valutazioni periodiche del modello per valutarne il corretto funzionamento.

2.2 Sviluppo di software e sicurezza

Lo sviluppo di software sicuro è il processo di progettazione, implementazione e manutenzione di sistemi software sicuri. Si tratta di seguire una serie di pratiche e procedure per ridurre al minimo il rischio di vulnerabilità e minacce alla sicurezza del software e dei dati che elabora. Gli obiettivi principali sono la protezione della riservatezza, dell'integrità e della disponibilità delle informazioni elaborate dal software, la prevenzione dell'accesso non autorizzato e della manipolazione dei dati.

La sicurezza deve essere presa in considerazione durante tutto il ciclo di vita del software. Per fare questo, esistono linee guida e standard da seguire in base all'ambito applicativo del software. Ne sono un esempio le linee guida dell'agenzia europea per la cybersecurity in ambito automotive, sopracitate, o quelle redatte dall'AGID, Agenzia per l'Italia digitale [7], per lo sviluppo di applicazioni sicure.

Esiste anche uno standard a livello europeo, ETSI EN 303 645 [8]. Questo fornisce una serie di requisiti di base per la sicurezza dei dispositivi dell'Internet of Things (IoT). Questo standard, destinato a essere integrato da altri più specifici, contiene le buone prassi in materia di sicurezza per i dispositivi di consumo connessi a Internet. Poiché molti dispositivi IoT di consumo gestiscono informazioni di identificazione personale, i PII, l'implementazione dello standard aiuta anche a conformarsi al Regolamento generale sulla protezione dei dati, il GDPR [9], nell'UE.

Questo standard definisce disposizioni specifiche tra cui l'obbligo di non utilizzare password predefinite universali, istituire un sistema per gestire le segnalazioni di vulnerabilità, aggiornare regolarmente il software, memorizzare in modo sicuro i parametri di sicurezza sensibili e garantire comunicazioni sicure. Inoltre indica di ridurre al minimo le superfici di attacco esposte, mantenere l'integrità del software, proteggere i dati personali e garantire la resilienza del sistema durante le interruzioni. Per essere conforme a tale standard bisognerebbe anche analizzare i dati telemetrici del sistema, consentire la cancellazione dei dati da parte dell'utente, semplificare l'installazione e la manutenzione del dispositivo e soprattutto convalidare sempre e in modo corretto i dati di input.

Un'applicazione deve essere messa in sicurezza prendendo in considerazione tutti i livelli logici e fisici che la compongono. Gli elementi principali di cui può essere composta sono modellati in figura 2.2.

Security by Design è un approccio che mira a garantire che la sicurezza sia un requisito fondamentale di progettazione per qualsiasi sistema o applicazione, e che venga integrata fin dalle prime fasi del ciclo di vita del software. Ciò riduce i rischi e garantisce che i sistemi finali siano sicuri e protetti contro le minacce esterne. È quindi importante adottare pratiche di progettazione sicura attraverso l'identificazione dei requisiti di sicurezza e delle contromisure in linea con i principi di security by design.

È possibile, ad esempio, adottare un approccio di 'defense in depth', che mira a limitare i danni in caso di attacco riuscito creando vari livelli di sicurezza. In questo modo, nel caso in cui un attaccante riesca a superare il primo livello di difesa, devono essere applicate ulteriori misure

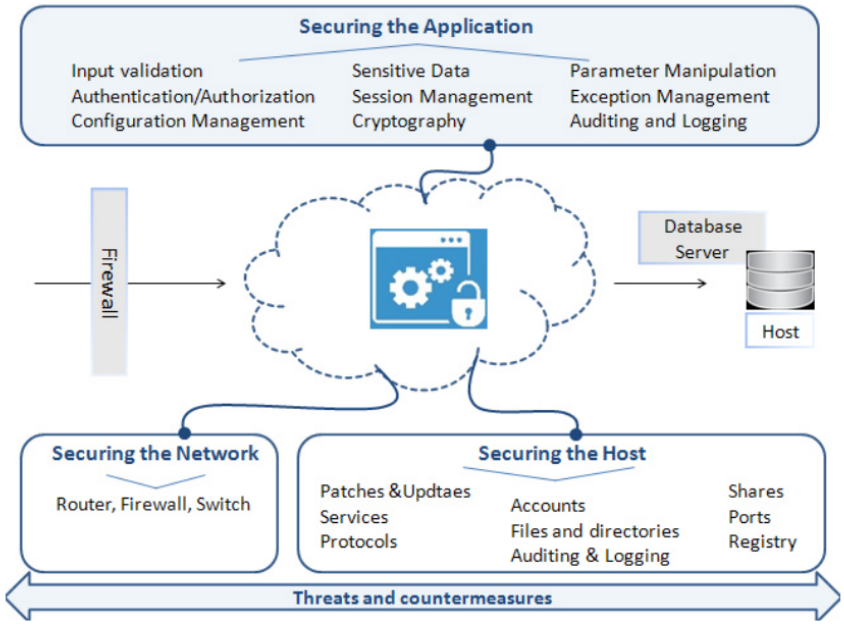

Figura 2.2. Schema per la sicurezza di un'applicazione [Linee guida per lo sviluppo di software sicuro]

più restrittive per impedirgli di procedere ulteriormente. Queste misure potrebbero includere la restrizione dei privilegi d'accesso alle risorse o l'isolamento dell'applicazione per impedire la propagazione dell'attacco all'intero sistema. Questo tipo di approccio però, deve essere preso in considerazione fin dall'inizio, per evitare costi e lavoro inutili.

Criteri generali

Durante lo sviluppo del codice di un'applicazione, esistono dei criteri generali da tenere in considerazione.

È importante considerare l'impatto che l'applicazione potrebbe avere sulle risorse del sistema in cui verrà eseguita. Ci sono diverse soluzioni di programmazione che possono aiutare a minimizzare questo impatto, come ad esempio preferire l'ottimizzazione del compilatore, tendenzialmente più efficace, all'ottimizzazione manuale. Oppure evitare di avere puntatori multipli ad una stessa risorsa ed utilizzare i data types appropriati. Inoltre, in alcuni casi, è da preferire lo switch/case alle strutture di if nidificati. Inoltre, è importante deallocare la memoria il prima possibile, quando non pregiudica la funzionalità dell'applicazione.Tutte queste pratiche possono contribuire a migliorare l'efficienza e la sicurezza del software durante l'esecuzione.

Inoltre, è importante evitare di inserire all'interno dei sorgenti, i dati di accesso ai database o ad altri sistemi sensibili, come username, password o nome del database. Se ciò non è possibile, è importante cifrare tali dati per evitare che siano facilmente accessibili da parte di terzi. Inoltre, per la gestione delle chiavi di cifratura e di altre informazioni riservate, valgono le stesse indicazioni.

È altresì importante che l'applicazione non venga avviata con i privilegi amministrativi. Questi concederebbero all'applicazione l'accesso a risorse e funzionalità critiche del sistema operativo, aumentando il rischio di attacchi informatici e di possibili danni al sistema stesso. Pertanto, l'applicazione dovrebbe essere avviata con i privilegi minimi necessari per il suo corretto funzionamento e senza concessioni di accesso a risorse o funzionalità non necessarie.

Il concetto di integrità del software, inoltre, è cruciale per garantire che questo funzioni come previsto e che le informazioni trattate siano protette. Ciò significa che il software deve essere resiliente agli attacchi informatici e alle violazioni della privacy, ma anche che bisogna impedire le modifiche non autorizzate . Per garantire l'integrità del software, sia la fase di progettazione, che quella di implementazione, devono prestare particolare attenzione alla gestione degli errori e

delle eccezioni che si possono verificare durante l'elaborazione dei dati in ingresso. È necessario che tali situazioni siano gestite correttamente, in modo che non ci sia una perdita di integrità delle informazioni. In questo modo, si può garantire che il software continui a funzionare in modo affidabile e che le informazioni trattate siano al sicuro.

Un altro aspetto importante da prendere in considerazione sviluppando un software è la gestione e la validazione degli input.

La corretta gestione dei parametri in input è un aspetto fondamentale per garantire la sicurezza del software. L'applicazione deve essere progettata per convalidare i dati inseriti dall'utente, in modo da evitare la possibilità di inserire valori potenzialmente pericolosi. A tal fine, è necessario implementare meccanismi che effettuino un controllo sui caratteri o sui valori inseribili dall'utente, sulla base di quelli richiesti dai campi corrispondenti. In questo modo, si possono annullare gli effetti dovuti ad errori come valori 'out of range', dati mancanti o incompleti e caratteri invalidi negli stream. Anche i caratteri speciali possono rappresentare una minaccia per la sicurezza del software, in quanto la loro combinazione può innescare diverse vulnerabilità. Pertanto, è necessario prestare particolare attenzione alla gestione di questi caratteri, implementando meccanismi di convalida che ne limitino l'uso solo ai casi in cui sia strettamente necessario.

L'applicazione deve essere progettata in modo da fornire solo le informazioni richieste e rilevanti per l'utente. In altre parole, l'applicazione non deve divulgare alcuna informazione che non sia strettamente necessaria per soddisfare la richiesta dell'utente, al fine di evitare la raccolta di informazioni non autorizzate o la divulgazione di dati sensibili. L'obiettivo è quello di proteggere la privacy dell'utente e di prevenire la divulgazione di informazioni riservate.

Inoltre, l'applicazione deve essere in grado di tracciare attività anomale ed eccezioni che si verificano durante l'utilizzo del sistema. Ci si riferisce in particolare ad eventi con esito positivo o no, errori di sistema e violazioni delle policy. Gli eventi che in genere necessitano di tracciamento includono l'autenticazione, i processi ad essa correlati, l'avvio e l'arresto delle componenti, le violazioni dei criteri, eventuali modifiche nella configurazione dell'applicativo, oltre che all'accesso ai dati, ai file e alle risorse dell'applicazione, nonché al tipo di accesso.

Per quanto riguarda l'autenticazione, l'autorizzazione e la gestione degli accessi all'interno di un'applicazione, esistono delle buone pratiche da seguire per limitare la superficie d'attacco. In particolare, è consigliabile adottare la politica standard 'tutto è generalmente proibito a meno che non sia espressamente concesso' per il controllo degli accessi, oltre che non assegnare alcun privilegio all'utente finché non vengono completate l'autenticazione e l'autorizzazione. Inoltre, riducendo al minimo le informazioni fornite agli utenti che non sono ancora autenticati durante la procedura di accesso, e utilizzando procedure di blocco momentaneo dell'account, per esempio dopo una serie di tentativi di accesso non andati a buon fine, si potrebbe bloccare un aggressore in fase di raccolta di dati o durante un attacco di tipo brute force sul sistema. Inoltre è buona pratica non utilizzare account con password di default, così come sarebbe necessario fare un controllo sui privilegi d'accesso dell'utente prima di autorizzarlo a fare qualsiasi tipo di operazione.

2.3 Cybersecurity nelle applicazioni web

La sicurezza delle applicazioni web è un aspetto critico della cybersecurity, dato il ruolo significativo che queste svolgono nel moderno panorama digitale. Poiché sempre più aziende si affidano al web per fornire i propri servizi, diventa sempre più importante garantire che queste applicazioni siano protette dalle minacce informatiche.

Esistono una serie di misure volte a prevenire attacchi quali cross-site scripting (XSS), SQL injection e altre forme di attacchi molto comuni. Tra queste troviamo pratiche di codifica, meccanismi di autenticazione e controllo degli accessi adeguati, crittografia dei dati sensibili, test di sicurezza ed un continuo monitoraggio delle vulnerabilità.

Una delle sfide principali in questo campo è rappresentata dal fatto che questo tipo di applicazioni sono spesso complesse e interconnesse. Questo le rende di base vulnerabili ad una serie di attacchi. Inoltre, le applicazioni web, sono in genere costruite utilizzando una serie di linguaggi di programmazione, framework e librerie, che possono creare contenere potenziali falle nella sicurezza se non adeguatamente protette.

Come se non bastasse, il panorama mondiale delle vulnerabilità sul web è in continua evoluzione, poiché gli attaccanti cercano sempre nuovi modi per sfruttare le debolezze di queste applicazioni. Si può avere una panoramica di questa tendenza grazie anche al progetto Top 10 OWASP [10].

L'Open Web Application Security Project è una comunità aperta, dedicata al miglioramento della sicurezza del software, con l'obiettivo di fornire una conoscenza condivisa e strumenti per identificare e mitigare i rischi di sicurezza nelle applicazioni web. La lista Top 10 Owasp è un elenco delle dieci vulnerabilità più comuni che possono compromettere la sicurezza delle applicazioni web. Questa lista, aggiornata periodicamente, è il risultato di una vasta ricerca e di un'ampia consultazione di esperti di sicurezza informatica provenienti da tutto il mondo. L'obiettivo è quello di fornire una guida ai professionisti della sicurezza e agli sviluppatori di software per aiutarli a prevenire e mitigare le vulnerabilità più comuni che possono essere sfruttate dagli attaccanti per compromettere la sicurezza delle applicazioni web.

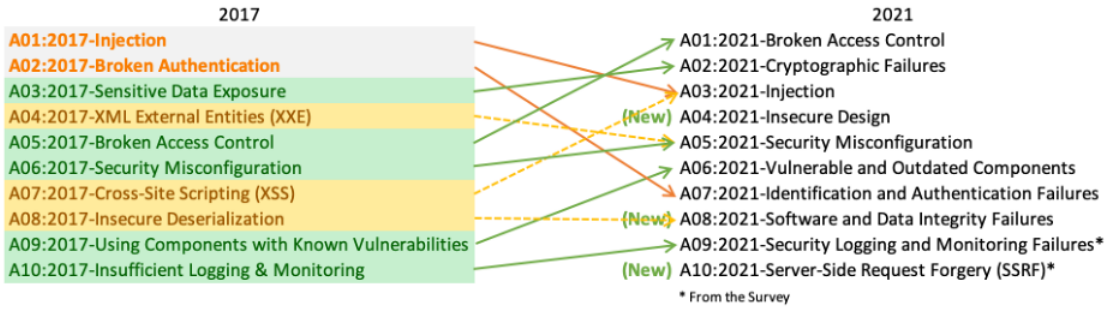

Figura 2.3. Confronto tra le 10 vulnerabilità più comuni tra il 2017 e il 2021 [OWASP Top 10]

2.3.1 Overview delle principali vulnerabilità

Controllo sugli accessi violato

Il controllo degli accessi è un aspetto cruciale nelle applicazioni, perché aiuta a garantire che gli utenti possano solo accedere alle risorse ed eseguire le azioni per le quali sono autorizzati. Se i meccanismi di controllo degli accessi non sono implementati correttamente, possono verificarsi gravi problemi di sicurezza, come la divulgazione, la modifica o la distruzione non autorizzata di dati, nonché l'accesso non autorizzato a funzioni aziendali sensibili.

Alcune delle vulnerabilità più comuni nel controllo degli accessi includono la violazione del principio del minimo privilegio o del 'deny by default', in cui l'accesso viene concesso a chiunque invece di essere limitato a capacità, ruoli o utenti specifici. Gli aggressori potrebbero, in determinati casi, anche aggirare i controlli modificando l'URL o le richieste API. Un altro problema comune è la vulnerabilità Insecure Direct Object References, IDOR, che può consentire l'accesso ad oggetti direttamente utilizzando dell'input utente.

Un altro problema è l'accesso alle API con controlli mancanti per i metodi POST, PUT e DELETE, che possono causare la manomissione o la distruzione dei dati. Anche l'elevazione dei privilegi è un rischio, in quanto un utente malintenzionato può agire come un utente autorizzato senza aver effettuato l'accesso, così come potrebbe agire come un amministratore mentre è connesso come un utente normale. Anche la manipolazione dei metadati è un problema, insieme alla riproduzione o alla manomissione dei token di controllo dell'accesso o l'utilizzo di cookie o campi nascosti per elevare i privilegi.

La configurazione errata di CORS, Cross-Origin Resource Sharing, è un'altra vulnerabilità che consente l'accesso non autorizzato da origini non attendibili. Infine, la navigazione forzata, può consentire agli aggressori di accedere come utenti non autenticati a pagine che la richiedono, o a pagine privilegiate come utenti standard.

È fondamentale garantire che i meccanismi di controllo degli accessi siano implementati correttamente e rivisti regolarmente per mitigare queste vulnerabilità e ridurre il rischio di accesso non autorizzato e perdita di dati.

Fallimenti Crittografici

Questa categoria riguarda tutte quelle situazioni in cui i dati sensibili possono essere esposti a rischi di vario genere a causa di problemi di crittografia. In questa categoria rientrano numerose tipologie di attacchi informatici, che possono portare all'esposizione di dati sensibili come password, numeri di carta di credito e record medici.

Tra le vulnerabilità più comuni in questo ambito troviamo ad esempio l'utilizzo di password non protette, l'utilizzo di algoritmi crittografici deboli o obsoleti, e la mancanza di sufficiente entropia per generare le chiavi crittografiche. Inoltre, possono esserci problemi di sicurezza legati all'utilizzo di protocolli di trasmissione di dati non sicuri, come HTTP, SMTP o FTP, e alla mancata implementazione di meccanismi di crittografia nei server web o nei backend dei sistemi.

Per prevenire queste vulnerabilità, è fondamentale adottare una serie di misure di protezione specifiche, come l'utilizzo di algoritmi di crittografia sicuri, la corretta gestione delle chiavi, la verifica della validità dei certificati dei server, e la generazione di vettori di inizializzazione sufficientemente sicuri. Inoltre, è importante evitare l'utilizzo di password in chiaro, e garantire che tutte le funzioni crittografiche siano progettate per soddisfare i requisiti di sicurezza specifici. Infine, è importante tenere sempre aggiornati tutti i software e i protocolli utilizzati, e monitorare costantemente l'infrastruttura per individuare eventuali anomalie o tentativi di attacco informatico.

Injection

L'Injection, ovvero iniezione, è una grave vulnerabilità che si verifica quando un aggressore è in grado di iniettare codice o dati dannosi in un'applicazione, facendola comportare in modi non previsti. In molti casi, ciò può portare all'accesso non autorizzato, alla modifica o alla distruzione di dati sensibili, o addirittura consentire a un aggressore di prendere il controllo dell'intero sistema.

Esistono diversi tipi di attacchi a iniezione, tra cui SQL, NoSQL, LDAP, Object Relational Mapping (ORM), o Object Graph Navigation Library (OGNL). Tutti questi attacchi funzionano sfruttando il modo in cui un'applicazione gestisce l'input dell'utente. Ciò può accadere quando i dati forniti dall'utente non sono adeguatamente convalidati o sanificati, oppure quando vengono utilizzate query dinamiche o chiamate non parametrizzate senza un adeguato escape consapevole del contesto.

Uno dei motivi principali per cui gli attacchi di questo tipo sono così pericolosi, è che possono essere sfruttate in vari modi. Ad esempio, gli aggressori possono utilizzare dati ostili all'interno di parametri di ricerca ORM per estrarre record sensibili o utilizzarli direttamente in query dinamiche.

Il modo migliore per rilevare se un'applicazione è vulnerabile alle iniezioni è eseguire una revisione del codice sorgente cercando in particolare gli input e le validazioni effettuate. Inoltre, è utile un controllo di tutti i parametri, delle intestazioni, degli URL, dei cookie e degli input di dati JSON, SOAP e XML.

Design insicuro

Questa cateogria di vulnerabilità, sottolinea l'importanza di affrontare i rischi legati ai difetti di progettazione nello sviluppo del software, oltre alla necessità di utilizzare la modellazione delle minacce, i modelli di progettazione sicuri e le architetture di riferimento.

È importante notare che la progettazione insicura si distingue dall'implementazione insicura. Mentre un progetto sicuro può comunque presentare dei problemi di implementazione che conducono a vulnerabilità, un progetto insicuro non può essere risolto da un'implementazione perfetta.

La progettazione sicura è una metodologia che valuta in modo continuativo le minacce, in modo da prevenire i attacchi noti. La modellazione delle minacce deve essere integrata nello sviluppo per analizzare le modifiche ai flussi di dati, al controllo degli accessi e a quelli di sicurezza.

Per lo sviluppo sicuro del software è necessario un ciclo di vita di sviluppo sicuro, che comprenda modelli di progettazione sicuri, librerie sicure, strumenti e modellazione delle minacce.

Configurazione della sicurezza scorretta

Questa categoria di rischio è salita in classifica negli ultimi anni, data la tendenza verso un software altamente configurabile.

Una configurazione errata può rendere le applicazioni vulnerabili agli attacchi, in particolare se manca l'hardening in qualsiasi parte dello stack applicativo o se le autorizzazioni sui servizi sono configurate in modo improprio. Altri fattori che possono aumentare il rischio di configurazione errata sono l'abilitazione di funzionalità non necessarie o il fatto di lasciare invariati account e password predefiniti. Anche la gestione degli errori che rivela tracce di stack o messaggi troppo esplicativi per gli utenti, così come la disattivazione o la configurazione errata delle funzioni di sicurezza più recenti sui sistemi aggiornati, possono aumentare il rischio.

Le impostazioni di sicurezza di application server, framework, librerie, database e altri componenti devono essere impostate su valori sicuri. Le intestazioni e le direttive di sicurezza devono essere inviate dal server e impostate su valori affidabili. Infine, è importante assicurarsi che il software sia aggiornato e non vulnerabile, poiché anche i componenti obsoleti o vulnerabili possono contribuire a una configurazione errata e aumentare il rischio di attacchi, anche se questo ricade in una categoria aprofondita in seguito.

Per ridurre il rischio di configurazione errata, è essenziale disporre di un processo di configurazione della sicurezza delle applicazioni concreto e ripetibile. Senza questo processo, i sistemi sono più esposti al rischio di attacchi.

Componenti vulnerabili e datati

Si tratta di una categoria che comporta la dipendenza e la valutazione di componenti di terze parti. Questo include tutti i componenti utilizzati direttamente e le dipendenze annidate.

Se il software, comprensivo di sistema operativo, server web, DBMS, API, applicazioni, ambienti di runtime e librerie, non è supportato, non è aggiornato o è vulnerabile, si va incontro a potenziali rischi. Una regolare scansione delle vulnerabilità e la sottoscrizione dei bollettini di sicurezza relativi ai componenti utilizzati sono essenziali per identificare e ridurre le vulnerabilità. Il mancato aggiornamento tempestivo del sistema, dei framework e delle dipendenze sottostanti, potrebbe portare ad un'esposizione non necessaria alle vulnerabilità precedentemente risolte. Anche la configurazione sicura dei componenti è essenziale per evitare errori di sicurezza.

Fallimenti nell'identificazione e nell'autenticazione

Si tratta di una categoria precedentemente nota come Broken Authentication, ora inclusiva delle vulnerabilità relative alle mancanze di identificazione, come la convalida impropria di un certificato con una mancata corrispondenza dell'host e l'autenticazione impropria.

È fondamentale confermare l'identità dell'utente, autenticarlo e gestire correttamente le sessioni per proteggersi dagli attacchi legati all'autenticazione. Un debolezza in questo ambito è data dalla possibilità di attacchi di brute forse, come il credential stuffing, in cui l'attaccante recupera un elenco di nomi utente e password, li inserisce in uno script e li sfrutta confidando nel fatto che spesso gli utenti utilizzano le stesse credenziali in più di un'applicazione. Allo stesso modo permettere password predefinite, deboli o note come "Password1" o "admin/admin", rendono il sistema vulnerabile. Un'altro punto debole in questo ambito è rappresentato dall'uso di processi di recupero delle credenziali deboli o inefficaci. L'applicazione può essere vulnerabile anche se utilizza archivi di password in chiaro o crittografati con hash debole, se l'autenticazione a più fattori è assente o inefficace, se espone l'identificativo di sessione nell'URL, se riutilizza o se non invalida correttamente gli ID di sessione.

Fallimenti nell'integrità del software e dei dati

Questa categoria di vulnerabilità si concentra su aggiornamenti del software, sui dati critici e sulle pipeline CI/CD di cui non viene verificata l'integrità. Rappresenta una di quelle vulnerabilità che hanno un impatto maggiore tra i dati delle Common Vulnerability and Exposures, CVE. Un esempio è CWE-494, ovvero il download di codice senza controllo precedente sull'integrità [11].

I fallimenti relativi all'integrità del software e dei dati sono legati a codice e ad infrastrutture che non proteggono adeguatamente contro le violazioni dell'integrità. Un esempio è un'applicazione che si basa su librerie, plugin o moduli provenienti da fonti e repository non affidabili. Inoltre, attualmente, molte applicazioni prevedono l'attivazione della funzionalità di auto-aggiornamento, in cui questi aggiornamenti vengono scaricati senza essere verificati e quindi con potenziali falle nella loro integrità. Gli attaccanti potrebbero quindi approfittarne e caricare le proprie versioni, da distribuire ed eseguire. Un altro esempio di vulnerabilità in questa categoria è quello in cui oggetti o dati sono vulnerabili alla de-serializzazione perché vengono codificati o serializzati in un modo che un attaccante può vedere e modificare.

Fallimenti nel logging e nel monitoraggio della sicurezza

Il logging e il monitoraggio della sicurezza sono essenziali per rilevare e rispondere alle violazioni attive. Se inadeguati, possono portare a una mancanza di visibilità e allerta sugli incidenti. Questa categoria è salita nella classifica, sottolineando la sua attuale importanza. Comprende debolezze come logging insufficiente, omissione di informazioni rilevanti per la sicurezza e inserimento di informazioni sensibili nel file di log.

Questo tipo di situazioni possono verificarsi quando eventi importanti non vengono registrati, gli avvisi e gli errori generano messaggi di registro poco chiari o inadeguati, i registri non vengono monitorati per rilevare attività sospette, o vengono archiviati solo a livello locale.

L'attività di logging e di monitoraggio sono quindi fondamentali per rilevare e rispondere alle violazioni. È essenziale garantire però, che gli eventi di registrazione e di avviso non siano visibili agli utenti non autorizzati o agli aggressori.

Falsificazione della richiesta lato server

La Server-Side Request Forgery, SSRF, è una vulnerabilità in cui un aggressore può manipolare l'applicazione per facendole inviare richieste a una risorsa non prevista. L'aggressore può sfruttare questa vulnerabilità per accedere a risorse interne o per eseguire azioni per conto dell'applicazione. Ciò può portare alla perdita di dati, all'escalation dei privilegi o alla completa compromissione del server.

Per prevenire le vulnerabilità SSRF, è necessario utilizzare tecniche di convalida e sanifica dell'input per garantire che gli URL forniti dall'utente siano legittimi e sicuri da utilizzare. Inoltre, è necessario utilizzare le regole del firewall e un ferreo controllo degli accessi per limitare l'accesso alle risorse sensibili.

2.4 Progetto base

Il software implementato da Forese e Caliò, di cui si può apprezzare il lavoro grazie alla tesi a supporto [1], è in grado di rendere indipendente un veicolo di tipo rover, sfruttando una rete neurale artificiale in grado di riconoscere e differenziare oggetti e rilevare l'ambiente circostante in modo da evitare collisioni e gestire casi limite. Il modulo di object detection utilizza la rete neurale MobilenetSSD v2 implementata in Python per rilevare differenti oggetti con una velocità tale da permettere di lavorare quasi in tempo reale. Il sistema è in grado di rilevare anche l'angolo e il modulo della posizione dell'oggetto, informazioni necessarie affinché un veicolo si muova autonomamente.

Il sistema richiede una minima interazione esterna per funzionare. Questa prevede la scelta della funzionalità tra 5 disponibili:

- Most central mode: selezione, da parte della rete, dell'oggetto più vicino al centro dell'immagine;
- Farthest mode: selezione, da parte della rete, dell'oggetto più lontano dal centro dell'immagine;
- Static target selection mode: selezione, da parte della rete, di un oggetto del tipo specificato dall'utilizzatore;
- Follow Me Mode: selezione e deselezione, grazie ad un movimento del corpo di una persona rilevata dalla rete, di un target da seguire;
- Reach target mode: selezione di un oggetto da seguire, del tipo specificato dall'utilizzatore.

L'interazione avviene attraverso una semplice interfaccia grafica.

2.4.1 Jetson Nano

Il kit è un piccolo computer, molto potente e in grado di eseguire reti neurali, progettato per l'implementazione di IA [12]. Rispetto alla versione ufficiale progettata da Nvidia, quella che utilizzata differisce solamente per la presenza di una memoria EMMC da 16 GB al posto dello slot per la scheda TF. Il computer single-board è provvisto di un ambiente Linux For Tegra, basato su Ubuntu 18.04 già presente al momento dell'accensione, ma reinstallabile tramite Nvidia SDKManager o tramite U disk.

2.4.2 Il software

L'applicazione è scritta in linguaggio Python e utilizza la rete SSD MobileNet V2. Il software si compone di differenti moduli:

1. **Object Detection Module** contenente la rete neurale e le cinque funzioni di sistema;
2. **Tracking Module** per gestire dinamicamente gli oggetti trovati dalla rete e seguirli;
3. **Follow Me Module** per permettere ad un soggetto di attivare e disattivare, attraverso un movimento specifico, la funzione di tracciamento;
4. **Safety Module** per la gestione del comportamento in alcuni casi limite attraverso funzioni che controllano la posizione dell'obiettivo e degli oggetti nell'immagine;
5. **GUI** che fornisce un'interfaccia che permette all'utilizzatore di scegliere la funzionalità desiderata. È l'unico modulo che si interfaccia con l'utente.

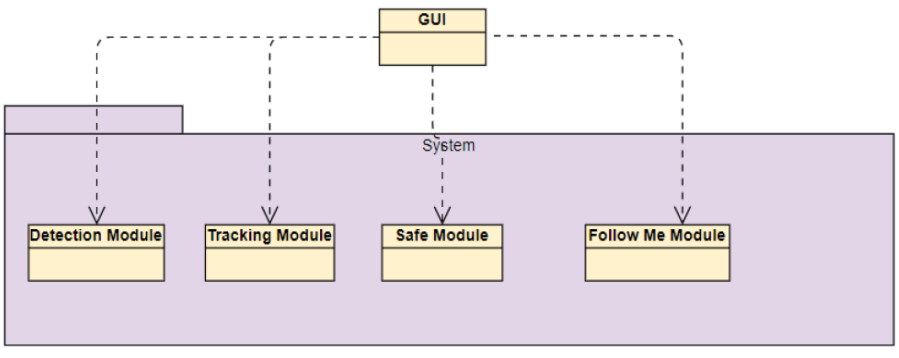

Figura 2.4. Diagramma delle classi di alto livello [1]

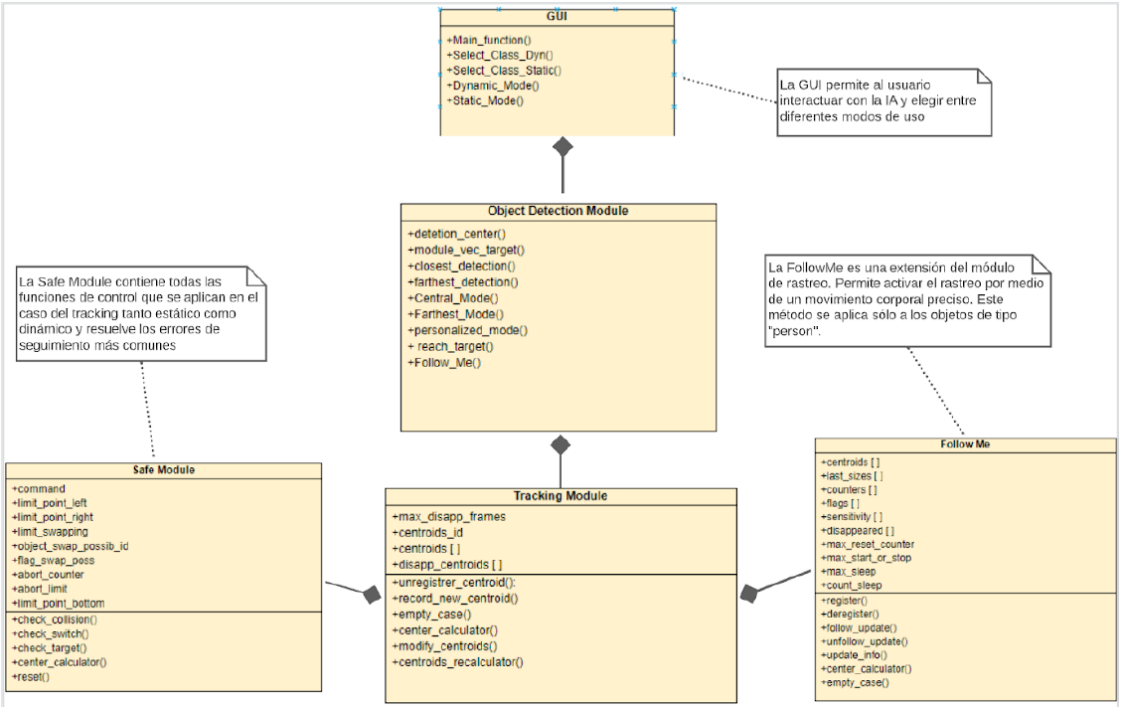

Figura 2.5. Diagramma delle classi completo [1]

2.4.3 Setup iniziale

Per eseguire l'applicazione correttamente è stato necessario installare alcune componenti del sistema poiché non presenti nell'immagine del sistema operativo alla prima accensione. Questo può essere fatto tramite Nvidia SDKManager su un host con sistema Linux o, in alternativa, è possibile reinstallare il sistema tramite U Disk utilizzando un'immagine con i pacchetti necessari preinstallati.

Ciò che deve essere installato è il pacchetto Jetpack fornito da Nvidia con la versione corrispondente a quella del sistema, in questo caso la versione 4.6.

Il pacchetto è composto dei seguenti:

- CUDA;
- cuDNN;
- TensorRT;
- OpenCV;
- VisionWorks;
- VPI;
- Altri componenti Nvidia.

A seguire deve essere scaricata la rete neurale utilizzata: SSD MobileNet v2. Per farlo è necessario clonare il progetto `jetson-inference` da https://github.com/dusty-nv/jetson-inference e seguire le istruzioni riportate nella documentazione di seguito riportate:

```
git clone https://github.com/dusty-nv/jetson-inference
cd jetson-inference
git submodule update --init
```

```
sudo apt-get install libpython3-dev python3-numpy
cd jetson-inference
mkdir build
cd build
cmake ../
```

Eseguendo download-models.sh si apre una finestra attraverso la quale è possibile scaricare la rete SSD MobileNet v2. In seguito sono necessari i seguenti comandi:

```
cd jetson-inference/build
make
sudo make installl
sudo ldconfig
```

Una volta completati questi passaggi è possibile importare il progetto. La scelta di utilizzare l'editor Visual Studio Code con l'aggiunta dell'estensione per il linguaggio Python è stata veicolata principalmente dalla facilità d'uso e dalla necessità di utilizzare strumenti di analisi della sicurezza del codice che sono forniti come estensioni per l'IDE. Installato l'editor e aperto il progetto correttamente, sono da risolvere eventuali errori di dipendenze dovuti a pacchetti di sistema non installati. In particolare potrebbe essere necessario installare il pacchetto imutils da terminale con i seguenti comandi:

```
python3 -m pip install -upgrade pip setuptools wheel
python3 -m pip install imutils
```

Capitolo 3

Analisi della sicurezza e riprogettazione del software

In questo capitolo si vuole analizzare dal punto di vista della sicurezza il software di cui si è discussa la realizzazione nel capitolo precedente. L'analisi avrà come scopo definire eventuali vulnerabilità e riprogettare l'applicazione con le misure di sicurezza adatte.

3.1 Metodologia di lavoro

Il primo passo è stato definire l'asset da prendere in considerazione in questo capitolo. Un asset è definito come un oggetto che ha valore, o contribuisce ad un valore, con una o più proprietà di sicurezza informatica la cui compromissione può portare ad uno o più scenari di danno [6].

In seguito è stato effettuato un confronto tra le maggiori tecniche di analisi della sicurezza volte ad identificare vulnerabilità presenti all'interno del codice già scritto. Il confronto ha permesso di identificare le tecniche migliori da utilizzare in questo contesto.

In questa fase è stata presa anche la decisione di utilizzare l'editor di codice sorgente Visual Studio Code. Questa scelta è conseguenza della presenza di numerosi strumenti di analisi statica disponibili anche come estensioni di VS Code. L'uso di queste estensioni ha permesso l'analisi del codice lungo tutta la fase di analisi e riprogettazione.

Dopo un'analisi effettuata con strumenti automatici, è iniziata la fase di revisione, composta dai seguenti step:

- Definizione degli obiettivi;
- Revisione del codice;
- Valutazione dei risultati;

Infine i problemi evidenziati dall'analisi sono state risolti come possibile.

3.2 Definizione dell'asset

Per gestire in modo corretto i problemi di cybersecurity di un veicolo è necessario inizialmente identificarne gli asset da proteggere e i limiti entro cui effettuare l'analisi. Un elenco generale dei possibili asset identificabili in un veicolo di questo tipo si può trovare nell'appendice A.

È oggetto di analisi di questo capitolo l'applicazione software descritta nel capitolo precedente, il cui codice esegue algoritmi di intelligenza artificiale, riceve dati dal sensore, la videocamera, e interagisce con l'utilizzatore.

Parti importanti dell'applicazione da proteggere sono i seguenti:

- Dati provenienti dalla videocamera e il sensore in sè;
- Password, chiavi e certificati;
- Informazioni relative al device;
- Punti di accesso al sistema;
- Informazioni relative allo user;
- Algoritmi di intelligenza artificiale;

L'ultimo punto si cita per completezza, ma non sarà oggetto approfondito dall'analisi effettuata in questa tesi. Infatti l'algoritmo realizzato utilizza una rete neurale pretrained. Questo significa che non è possibile attuare le mitigazioni opportune al fine di mettere in sicurezza l'asset dagli attacchi ad esso mirati. Sono particolarmente rilevanti al fine di quest'analisi e di questa decisione i report realizzati da ENISA, uno specifico sulle sfide di sicurezza informatica dell'intelligenza artificiale nella guida autonoma [13] e uno generale sulle minacce alla sicurezza dell'AI [14].

3.2.1 Confronto tra tecniche di analisi

L'analisi statica del codice e l'analisi dinamica del codice sono due tecniche diverse per valutare la sicurezza delle applicazioni software.

La prima, è spesso più veloce ed efficiente della seconda, ed è anche più facile da automatizzare. Tuttavia, potrebbe non essere in grado di identificare tutte le vulnerabilità presenti nell'applicazione, poichè non esamina il comportamento durante la sua esecuzione.

L'analisi dinamica del codice, invece, è in grado di identificare le vulnerabilità che potrebbero non essere rilevabili attraverso l'analisi statica, poichè esamina il comportamento dell'applicazione mentre è in esecuzione. Tuttavia, richiede più tempo e più risorse.

Per lo scopo di questo capitolo si utilizzano tecniche di analisi statica, più utili ed efficienti in questa fase. Questo perchè il codice dovrà essere integrato all'interno di un'applicazione web, come discusso nei capitoli 4 e 5. Quindi risulta dispendioso e poco vantaggioso eseguire un'analisi dinamica su un sistema non ancora completo. L'analisi statica, al contrario, risulta vantaggiosa perchè permetterà di utilizzarne i risultati all'interno del sistema completo.

Di seguito sono elencate alcune tecniche di analisi statica del codice:

- Revisione manuale del codice: si esamina manualmente il codice sorgente per identificare eventuali vulnerabilità.
- Analisi automatica del codice: utilizza strumenti di analisi del codice che analizzano automaticamente il codice sorgente per identificare le vulnerabilità. Questi strumenti utilizzano regole e schemi predefiniti per identificare le vulnerabilità.

L'analisi automatica presenta i seguenti vantaggi e svantaggi:

- Vantaggi
  - Coerenza: garantisce che il codice venga analizzato e valutato in modo coerente e standardizzato, riducendo il rischio di errore umano o di distorsione.
  - Efficienza: può elaborare grandi volumi di codice molto più velocemente dell'analisi manuale del codice, rendendola più efficiente ed economica.
  - Rilevamento precoce: può rilevare vulnerabilità di sicurezza e altri problemi nelle prime fasi del processo di sviluppo, consentendo di risolverli prima che l'applicazione venga distribuita.
  - Scalabilità: può essere facilmente scalata per soddisfare le esigenze di applicazioni e team di sviluppo su larga scala.

  - Monitoraggio continuo: può essere integrata nel processo di sviluppo per fornire un monitoraggio e un feedback continui, migliorando la qualità complessiva e la sicurezza del codice.

- Svantaggi
  - Falsi positivi: può produrre falsi positivi, quando vengono segnalati problemi che in realtà non sono vulnerabilità.
  - Falsi negativi: può produrre anche falsi negativi, quando non vengono segnalati problemi che in realtà sono vulnerabilità.
  - Comprensione limitata: è limitata dalla sua capacità di comprendere il codice, il che significa che possono sfuggire vulnerabilità di sicurezza o altri problemi che non sono facilmente rilevabili.
  - Accuratezza limitata: dipende dalla qualità e dalla complessità delle regole e degli algoritmi utilizzati dallo strumento, il che significa che potrebbe non essere sempre accurata come l'analisi manuale del codice.
  - Limitazioni dello strumento: gli strumenti sono limitati dai linguaggi e dalle piattaforme che supportano, per cui potrebbero non essere adatti a tutte le applicazioni.

La revisione manuale presenta i seguenti vantaggi e svantaggi:

- Vantaggi
  - Maggiore accuratezza: consente a un essere umano di analizzare a fondo il codice e di comprenderne il contesto, con conseguente maggiore precisione nell'identificazione delle vulnerabilità di sicurezza e di altri problemi.
  - Migliore comprensione del codice: offre l'opportunità di comprendere a fondo il codice, migliorando la qualità complessiva e la sicurezza dell'applicazione.
  - Personalizzazione: può essere adattata alle esigenze specifiche di un progetto, consentendo una valutazione più completa ed efficace del codice.

- Svantaggi
  - Richiede tempo: può richiedere tempo e lavoro, rendendola meno efficiente e più costosa dell'analisi automatica del codice.
  - Errore umano: è soggetta a errori umani, il che significa che i problemi possono essere mancati o identificati in modo errato.
  - Pregiudizio: può essere influenzata dalle opinioni soggettive del revisore, con conseguenti risultati incoerenti o imprecisi.
  - Scalabilità limitata: è limitata nella sua capacità di scalare per soddisfare le esigenze di applicazioni e team di sviluppo su larga scala.
  - Monitoraggio continuo: non è in grado di fornire un monitoraggio e un feedback continui, rendendo più difficile mantenere la qualità e la sicurezza complessiva.

3.3 Analisi statica del codice con strumenti automatici

Il codice è stato inizialmente analizzato adottando tecniche di analisi statica automatica con l'obiettivo di identificare prima e più velocemente le vulnerabilità più comuni. Questo passo è stato realizzato con l'obiettivo di incominciare la revisione manuale con le maggiori e potenziali vulnerabilità già identificate e da valutare.

3.3.1 Analisi preliminare

La scelta di utilizzare più strumenti di analisi è stata fatta in modo da permettere un'analisi più estesa, poiché è possibile che alcune vulnerabilità vengano rilevate da uno strumento e non dall'altro sulla base degli algoritmi usati.

Strumenti

La scelta degli strumenti utilizzati per l'analisi statica del codice si è basata sui seguenti criteri:

- Tipo di licenza: sono stati selezionati solo programmi Open Source;
- Piattaforma: è stato necessario scegliere strumenti compatibili con la piattaforma Linux;
- Linguaggio del codice: non tutti gli strumenti sono in grado di effettuare la scansione di codice scritto in Python, per questo sono stati scelti solo quelli che supportano questo linguaggio.

Gli strumenti che sono stati scelti sono:

- **Bandit**: programma eseguito da terminale creato per trovare problemi di sicurezza comuni in codice Python. Questo lo fa processando ogni file del progetto, costruendo un AST (Abstract Syntax Tree), eseguendo i plugin appropriati per i nodi dell'albero per poi generare un report alla fine della scansione [15]. Questo strumento non ha trovato alcuna vulnerabilità;
- **HCL Appscan CodeSweep extension for Visual Studio Code** effettua SAST, Static Application Security Testing, del codice semplicemente procedendo al salvataggio del file da esaminare e rilevando eventuali vulnerabilità del codice [16]. Anche questo strumento non ha rilevato vulnerabilità;
- **Horusec** può essere eseguito sia da terminale che come estensione di Visual Studio Code realizzando un'analisi statica del codice [17]. Non sono state riscontrate vulnerabilità.

Questo tipo di strumenti è in grado di rilevare varie vulnerabilità come cross site scripting XSS, SQL injection, buffer overflow, credenziali inserite nel codice, utilizzo di funzioni non sicure di I/O e altre, differenziando anche in base al linguaggio.

Per esempio sono in grado di rilevare se l'output generato dal software riceve come parametro l'input inserito dall'utente senza che questo venga validato in alcun modo, come potrebbe succedere nel caso delle modalità di 'select target' e 'reach target'. Infatti se quanto inserito dall'utente non fosse gestito tramite whitelist, e venisse preso come parametro di una funzione di stampa, allora il programma rivelerebbe una possibile vulnerabilità. Aggiungendo una funzione di stampa di test prima del controllo sul valore infatti, sia Horusec, che CodeSweep, evidenziano la presenza di una possibile vulnerabilità. È possibile anche verificare se il codice è soggetto a path traversal, ovvero se l'applicazione inserisce l'input dell'utente nelle operazioni del filesystem senza la dovuta attenzione. Questi strumenti sono anche in grado di rilevare per esempio l'utilizzo di alcune librerie deprecate e pericolose come Pickle, o se vengono utilizzate funzioni deprecate.

Risultati

Nessuno dei tre strumenti ha rilevato vulnerabilità, al momento della scansione. Questo tipo di strumenti rileva i problemi più comuni, relativi principalmente agli user input. In questo software gli user input sono richiesti solamente in due specifiche modalità, ovvero:

- 'Select detection class' in modalità statica;
- 'Reach target' in modalità dinamica.

In entrambi i casi l'input viene validato correttamente perché deve corrispondere ad una delle classi elencate in modo statico all'interno del programma, ovvero ci si appoggia ad una whitelist. Questo fa si che non rappresentino delle vulnerabilità. Anche se non rilevate, non vuol dire che non esistano vulnerabilità. Infatti questo tipo di strumenti rileva solo una parte di quelle che possono presentarsi in un'applicazione. Per questo è necessaria un'analisi manuale in modo da avere un quadro più completo.

3.3.2 Analisi Continua

Lo scanning realizzato gagli strumenti automatici utilizzati come estensione di Visual Studio Code ha permesso di effettuare un controllo di sicurezza attraverso tutti gli step a seguire. Il vantaggio di questa tecnica è stato quello di evitare lungo tutto il ciclo di sviluppo l'inserimento di vulnerabilità note.

3.4 Revisione manuale del codice

La revisione del codice, anche in combinazione con programmi automatici, permette di effettuare un'analisi più completa ed efficace.

Gli obiettivi di questa analisi sono:

- Verificare la sicurezza del codice, identificando eventuali vulnerabilità e garantendo che il codice sia protetto da eventuali attacchi;
- Migliorare la qualità e leggibilità del codice;
- Identificare eventuali opportunità di ottimizzazione.

Il primo è indiscutibilmente l'obiettivo principale dell'analisi, ma la sicurezza di un software è direttamente legata ad altri aspetti del codice.

Migliorare la qualità del codice può aiutare a prevenire alcune vulnerabilità di sicurezza poichè un codice ben scritto e organizzato è più facile da comprendere e mantenere, e meno propenso a contenere errori. Inoltre, un codice di qualità superiore è più probabile che segua linee guida di sicurezza e best practice, riducendo così il rischio di vulnerabilità conosciute. Infine, un codice più pulito e ben documentato rende più facile identificare eventuali problemi di sicurezza e aiutare a correggerli.

Identificare opportunità di ottimizzazione nel codice può aiutare a migliorare la sicurezza poichè un codice più efficiente è meno propenso ad essere soggetto a problemi di prestazioni che potrebbero creare opportunità per gli attaccanti. Inoltre, un codice ottimizzato utilizza risorse di sistema in modo più efficiente, riducendo il rischio di eventuali problemi di sovraccarico o di esaurimento delle risorse, ulteriore opportunità per agenti malevoli.

La revisione è stata condotta su livelli differenti:

- Sviluppo del codice: l'analisi si focalizza sulla progettazione del codice, valutando il rispetto di buone pratiche di sicurezza o l'inserimento di istruzioni o comportamenti vulnerabili;
- Formattazione del codice: l'analisi si concentra sulla formattazione corretta con lo scopo di identificare modifiche in grado di aumentarne la leggibilità e la comprensibilità;
- Tracciamento di eventi: l'analisi si concentra sul monitoraggio e sulla registrazione di eventi, in particolare di alarm detection, durante l'esecuzione dell'applicazione;
- Compilazione del codice: l'analisi si pone di valutare opzioni utilizzate in fase di compilazione, cercando le opzioni di sicurezza appropriate per prevenire eventuali attacchi.

3.4.1 Sviluppo del codice

Memoria

Python è un linguaggio di programmazione che utilizza un sistema di gestione automatica della memoria trasparente al programmatore. La memoria viene automaticamente assegnata e liberata dal sistema di garbage collector, che monitora continuamente gli oggetti in memoria e libera quelli che non sono più utilizzati. Ciò rende il codice più semplice da scrivere e mantenere, ma può anche

aumentare il consumo di memoria poiché gli oggetti inutilizzati non sono necessariamente liberati subito. Si deve valutare l'ottimizzazione del codice attraverso l'utilizzo delle funzioni della libreria Python `gc`, in grado di personalizzare l'uso del garbage collector.

Si ipotizza che l'uso della funzione `gb.collect()` inserita in specifici punti del codice possa portare ad un minore ed ottimizzato utilizzo della memoria. In particolare si ipotizza che possa essere utile al termine dell'utilizzo di ogni modalità, poiché molte variabili e strutture dati vengono rilasciati. Si utilizza lo strumento `memory_profiler` per valutare le performance.

Si è testato inizialmente il funzionamento del programma senza alcun intervento sul funzionamento del garbage collector. La percentuale di utilizzo della memoria, utilizzata da ciascuna modalità, indipendentemente dalla durata della ripresa, risulta in una media approssimata del 12,5%. Testando il sistema aggiungendo l'istruzione `gc.collect()` al termine della funzione che gestisce ciascuna modalità, la percentuale di memoria utilizzata risulta nuovamente di poco superiore al 12%. Per questo motivo risulta una soluzione poco efficace.

Tipi di dato

Python è un linguaggio di programmazione non tipicizzato, meno sicuro rispetto ai linguaggi di programmazione tipizzati poiché non effettua automaticamente controlli sui tipi dei dati in modo rigoroso. Questo rende necessari i seguenti accorgimenti:

- Utilizzare funzioni e librerie sicure;
- Convalidare i dati di input;
- Limitare i privilegi dell'account;
- Implementare controlli di autenticazione robusti;

Sono da valutare per questo scopo le librerie esterne e le relative funzioni utilizzate all'interno del programma in modo da valutarne eventuali vulnerabilità note. I seguenti dati sono stati rilevati sul National Vulnerability Database [18].

- Jetson Inference: non risultano esserci vulnerabilità note;
- Jetson Utils: non risultano esserci vulnerabilità note;
- Tkinter: non risultano esserci vulnerabilità note;
- math: non risultano esserci vulnerabilità note;
- re: non risultano esserci vulnerabilità note;
- time: non risultano esserci vulnerabilità note;
- logging: non risultano esserci vulnerabilità note;
- OrderedDict from collections: non risultano esserci vulnerabilità note;
- distance from scipy.spatial: non risultano esserci vulnerabilità note;
- OpenCV 4.1.1: risultano 2 possibili vulnerabilità della versione utilizzata;
- NumPy 1.13.1: risultano 4 possibili vulnerabilità della versione utilizzata;

Per tutte le vulnerabilità identificate si sono analizzate le implicazioni all'interno del programma. Le vulnerabilità di OpenCV sono le seguenti.

CVE-2019-5064 [19] : si tratta di una vulnerabilità che può essere sfruttata per un attacco di tipo buffer overflow, il cui score CVSS è 8.8; questa non rappresenta una minaccia in questo caso perché le variabili in cui si salvano i valori dati in input dall'utente non sono preallocate, ma allocate in automatico dal gestore di memoria.

CVE-2019-16249 [20] è una vulnerabilità, con CVSS 5.3, dovuta ad una lettura out-of-bounds all'interno di computeSSDMeanNorm, funzione non utilizzata all'interno del programma in esame.

Per quanto riguarda NumPy, 4 funzioni della libreria sono risultate affette da vulnerabilità.

CVE-2017-12852 [21] : si tratta di una mancata validazione di input all'interno della funzione numpy.pad, con score 7.5; questa funzione non è utilizzata dall'applicazione.

CVE-2019-6446 [22] : vulnerabilità della funzione numpy.load che utilizza il modulo di Python pickle in modo insicuro, con CVSS 9.8; anche in questo caso la funzione non viene mai utilizzata.

CVE-2021-34141 [23] : vulnerabilità di numpy.core con punteggio 5.3; non interessa questo progetto.

CVE-2021-41496 [24] : vulnerabilità con score 5.5, di tipo buffer overflow, dovuta alla funzione array_from_pyobj; la funzione interessata non è utilizzata all'interno del software.

Di conseguenza nessuna libreria è risultata affetta da vulnerabilità. Nonostante ciò, bisogna costantemente valutare l'insorgere di nuove vulnerabilità ad esse relative, in modo da essere pronti ad aggiornare il software per mitigarne gli effetti.

La convalida dell'input viene presa in esame più nel dettaglio nella sezione 3.4.1.

È importante che l'applicazione venga anche eseguita con i privilegi minimi, secondo il principio del least privilege. Infatti è buona norma assegnare solo le autorizzazioni necessarie al software per poter eseguire le proprie funzioni, in modo da limitare l'estensione della superficie d'attacco. Questo software, oltre ad essere eseguito all'interno della Jetson Nano da una sessione utente senza privilegi amministrativi, non ne necessita alcuno per essere eseguito. Il principio del privilegio minimo è quindi rispettato.

Autenticazione

Per quanto riguarda i controlli di autenticazione, nel progetto iniziale non ne è previsto alcun tipo. Non ci sono né chiavi né certificati da proteggere. Viene lasciata l'implementazione alla fase seguente, in cui verranno presi in esame più nel dettaglio il contesto e le misure adeguate.

Costrutti nidificati

Sono presenti all'interno del codice due costrutti nidificati if-else. Questi hanno conseguenze sia per la sicurezza che per la performance. Infatti, se non ottimizzati correttamente, i costrutti nidificati possono rallentare il codice e rendere l'applicazione più lenta. In questi casi è consigliato, dove possibile, convertirli in costrutti switch-case che, lavorando su valori interi, permettono una traduzione più veloce in codice macchina. Per quanto riguarda la sicurezza, i costrutti nidificati devono essere implementati correttamente per evitare di rappresentare vulnerabilità sfruttabili da attacchi come code injection. Per fare questo, è necessario verificare che i dati di input siano controllati e sanificati.

Al contrario di altri linguaggi però, Python non mette a disposizione al programmatore un costrutto switch-case. Inoltre i costrutti nidificati all'interno del programma sono solo due.

```
[...]

if(target == -1): #Waiting for a target
    [...]
else: #Target selected
     if(END_OF_THE_MISSION!=1): [...]
```

```
[...]

if(target == -1): #Waiting for a target
    [...]
else: #Target selected
    if(sf.check_target(objects,target) == 3): [...]
    else: [...]
```

In entrambi i costrutti si può ritenere l'utilizzo delle condizioni corretto. Infatti tutte le variabili, `target, END_OF_THE_MISSION, objects` sono create e assegnate dal programmatore e non subiscono l'intervengo dell'utilizzatore.

Validazione dei dati in input

La corretta validazione dei dati in input riveste un ruolo importantissimo per la sicurezza delle applicazioni. Questo processo consiste nella verifica della corretta forma e della validità dei dati forniti dall'utente, prima che vengano elaborati o archiviati, per proteggersi da attacchi come code injection, buffer overflow e SQL injection.

Nel codice sono presenti due sezioni di codice in cui è richiesto un input da parte dell'utente:

```
select_class()
select_class_dyn()
```

Queste due funzioni si trovano nel modulo principale, `object_detection_module`, dove viene gestita l'interfaccia grafica. Queste contengono il codice che richiede un valore all'utente, corrispondente alla classe desiderata, che viene confrontato con una lista di classi rese disponibili dall'applicazione. Se questo valore trova una corrispondenza, allora la categoria selezionata sarà usata come target della modalità prescelta.

Quando viene inserito l'input, questo deve essere validato nel modo corretto in modo da verificare che ciò che viene letto sia coerente con i requisiti. In questo caso il valore deve corrispondere con una delle parole di una lista statica presente all'interno del programma, ovvero si utilizza una whitelist dei valori in input, soluzione più sicura che si possa utilizzare. Infatti questo fa si che il programma debba gestire solo un insieme finito di valori. Tutti gli altri vengono scartati. Per scartare gli input troppo lunghi o con caratteri speciali, quindi potenzialmente malevoli, prima di scansionare l'intera lista, è possibile aggiungere una validazione iniziale. Le parole non superano i 15 caratteri, per cui è coerente effettuare un controllo sulla lunghezza. Inoltre il programma si aspetta solo caratteri alfabetici, spazi o underscore.

```
target = entry1.get()
pattern = re.compile("^[a-zA-Z\s_]{1,15}$")
if re.match(pattern, target) is None :
    logMain.warning('Input invalid: wrong syntax')
    tk.messagebox.showinfo("Input Error", "Input invalid: wrong syntax")
else:
    if not any( (target == label) for label in label_vetctor):
        logMain.warning('Input invalid: no class found named '+
            str(target))
        tk.messagebox.showinfo("Input Error", "Input invalid: no class
            found.")
    else:
        logMain.info('Valid input target:' + str(target))
```

Come mostrato in questo codice, l'input viene inizialmente validato tramite regex. Se non rispetta i requisiti questo viene immediatamente scartato. In caso contrario viene scansionata la lista in cerca di una corrispondenza. Il codice è uguale in entrambe le funzioni.

Validazione dei dati in output

Sono presenti numerose istruzioni `print` all'interno del codice. L'utilizzo di questa istruzione non rappresenta in sé un problema di sicurezza, se, come in questo caso, non viene richiesto di stampare dati privati o sensibili. È consigliabile però validare il contenuto delle variabili che sono stampate, valutandone i caratteri e la lunghezza, in modo da evitare eventuali fuoriuscite involontarie. Inoltre, stampare grandi quantità di dati può risultare controproducente per il sistema, soprattutto per applicazioni in tempo reale. È buona pratica utilizzare questa funzione solo per debug. In fase di produzione sono preferibili strumenti più sicuri e performanti, come il registro degli eventi di log.

Sono presenti 13 istruzioni `print`, di cui 9, con alcune ripetizioni, contenenti valori dinamici e potenzialmente sensibili.

```
print( angle, dist_target)
print("Target Lost:" + str(target))
print(str(target_object.Bottom))
print("END OF THE MISSION" + str(END_OF_THE_MISSION))
print(str(id) + " current: " + str((dim)) + " - old: " +
    str(self.last_sizes[id]) + " count:" + str(self.counters[id]))
print("Activated " + str(centroidID))
print("Abort counter: " + str(self.abort_counter))
```

In realtà si tratta di valori validati precedentemente, come target, o di valori numerici che non possono essere considerati valori privati. È buona pratica eliminare tutte queste istruzioni e, se necessario, inserire dei log a riguardo.

Le informazioni che sono rilasciate in output all'utente sono invece scritte sull'immagine ripresa dalla videocamera. Questi sono essenziali per il funzionamento e sono valori, principalmente numerici, che non rappresentano informazioni sensibili. Si tratta infatti delle percentuali di sicurezza nella rilevazione di un oggetto, della sua distanza o degli identificativi dei centroidi. Inoltre sono utilizzate delle stringhe statiche per comunicare all'utente determinate informazioni, come la fine della missione.

3.4.2 Formattazione del codice

La formattazione del codice è il processo di rendere il codice sorgente più leggibile e presentabile, rendendo più facile la sua comprensione, la manutenzione e l'identificazione di eventuali problemi di sicurezza. Ciò comprende l'utilizzo di spaziatura, indentazione, commenti e altre convenzioni di stile.

In Python, la formattazione del codice è supportata dalla PEP8 [25], che è una guida per lo stile di codifica ufficiale per il linguaggio Python. Il codice è in buona parte conforme allo standard, ma è consigliabile aggiungere delle intestazioni complete del file e delle singole funzioni.

Le pratiche consigliate da PEP8 sono:

- utilizzare 4 spazi per ogni livello di indentazione;
- se possibile, limitare le righe a 79 caratteri e se lo supera cercare di dividerla in più righe in modo coerente;
- usare parole minuscole separate da trattini bassi per i nomi di variabili e funzioni, usare CamelCase per i nomi delle classi;
- mantenere tutte le importazioni all'inizio del file, raggruppando le importazioni dalla libreria standard, dalle librerie di terze parti e dalle importazioni locali.
- usare gli spazi intorno agli operatori e dopo le virgole ed evitare di usare più di due righe vuote di seguito.

- scrivere commenti chiari, concisi e informativi per tutte le parti del codice.
- usare le docstring per fornire la documentazione di funzioni, classi e moduli.
- usare i blocchi try-except per gestire le eccezioni in modo pulito e coerente.
- tenere insieme il codice correlato in funzioni, classi e moduli, evitando funzioni lunghe e complesse.

3.4.3 Tracciamento di eventi

Il tracciamento degli eventi è un processo che consente di monitorare e registrare gli eventi che si verificano in un sistema, sia quelli andati a buon fine, che non.

Il software non presenta un sistema di tracciamento degli eventi interni. Un sistema di logging aiuta a comprendere meglio come il software sta funzionando e a identificare eventuali problemi, come errori o performance insufficienti. Inoltre, consente di monitorare l'utilizzo del software e di apportare eventuali miglioramenti per garantirne la qualità. È utile soprattutto per identificare e risolvere problemi, monitorare le prestazioni e migliorare la sicurezza dell'applicazione, non solo durante lo sviluppo, ma durante tutta la vita del prodotto.

È stata utilizzata la libreria fornita da Python `logging` per implementare un sistema di logging all'interno dell'applicazione. Questa permette di capire cosa succede all'interno dell'applicazione. Non serve solo per debug, ma anche per rilevare eventuali attacchi.

Il sistema che salva i log durante il funzionamento del software è stato così organizzato. Per i quattro moduli sono stati creati quattro file di log differenti. In questo modo, anche se più difficili da gestire, è più facile identificare eventuali problemi. Il livello minimo di dettaglio è stato impostato a `INFO` in modo tale da avere un quadro del funzionamento completo. Ogni log ha il formato:

$$\langle data \rangle + \langle ora \rangle + \langle filename \rangle + \langle livello \rangle + \langle messaggio \rangle$$

Si riportano di seguito due piccoli esempi dei file di log del modulo principale e del modulo FollowMe.

```
2023-01-03 16:26:00,075 __main__ INFO Most Central Object: started
2023-01-03 16:26:00,078 __main__ INFO Most Central Object: detecting neural
    network
2023-01-03 16:26:21,019 __main__ INFO Most Central Object: neural network
    detected
2023-01-03 16:26:21,149 __main__ INFO Camera: camera mode setted
2023-01-03 16:26:22,602 __main__ INFO Camera: camera initialiazed
2023-01-03 16:26:22,611 __main__ INFO Most Central Object: loop starting
2023-01-03 16:26:22,624 __main__ ERROR Camera: something went wrong when
    reading the camera
```

```
2023-02-06 15:56:27,345 others_modules.follow_me_module INFO FollowMe
    initiated
2023-02-06 15:56:38,636 others_modules.follow_me_module INFO
    center_calculator starting
2023-02-06 15:56:38,639 others_modules.follow_me_module INFO
    center_calculator finished
2023-02-06 15:56:38,665 others_modules.follow_me_module INFO register starting
2023-02-06 15:56:38,667 others_modules.follow_me_module INFO register finished
```

In particolare il primo esempio evidenzia come, in quel caso, la fotocamera non si sia attivata. Con questa informazione si può facilmente risalire al problema.

3.4.4 Compilazione del codice

La compilazione del codice rappresenta un passo importante nella sua sicurezza perché consente di rilevare errori di sintassi e di logica che possono essere il trampolino di lancio per eventuali attacchi. La compilazione è il processo di traduzione dal sorgente all'eseguibile, che può essere eseguito dal computer. Il compilatore esegue controlli di sicurezza come la verifica della tipizzazione o la verifica che eventuali array non vengano utilizzati al di fuori dei loro limiti, che possono aiutare a prevenire vulnerabilità comuni come code injection o buffer overflow.

La compilazione iniziale del codice ha evidenziato i seguenti problemi: le importazioni delle librerie `jetson.utils` e `jetson.inference` effettuate con il carattere speciale '.' risultano deprecate. Non è necessario che la compilazione avvenga senza warning, anche se altamente consigliato, purché correttamente gestiti e analizzati. I warning possono indicare problemi di codice che, se non risolti, potrebbero compromettere la sicurezza e la stabilità del software. In alcuni casi, i warning possono anche essere ignorati se ritenuti non rilevanti per il progetto specifico, ma ciò dovrebbe essere deciso con attenzione e documentato. In generale, è buona pratica cercare di evitare warning nella compilazione e risolverli se presenti, come in questo caso.

Le istruzioni `import` presenti nel modulo principale `object_detection_module` sono stati sostituiti rispettivamente da `jetson_util` e `jetson_inference`

3.5 Conclusione

L'analisi, condotta grazie all'ausilio di strumenti automatici, standard e linee guida, ha permesso di evidenziare alcune debolezze del codice e le conseguenti soluzioni. Queste ultime sono quindi state integrate all'interno del software, permettendo di migliorarne la sicurezza e la manutenibilità per future analisi forensi. Le modifiche effettuate, sono state verificate durante l'implementazione con l'ausilio degli strumenti di analisi statica citati precedentemente, in modo da non aggiungere ulteriori vulnerabilità.

Quest'analisi è solo una piccola parte dell'intero ciclo di vita di un'applicazione, ma ha permesso di proseguire il lavoro utilizzando del codice che si può considerare sicuro. Questo passaggio è importante perché facilità gli step successivi e migliora la robustezza del prodotto finale.

Capitolo 4

Progettazione del sistema di comunicazione

L'obiettivo di questo capitolo è quello di introdurre il sistema di comunicazione sicuro creato allo scopo di far comunicare il veicolo con l'esterno e permettere all'utente di interagire con esso. Si vuole spiegare in queste pagine quali sono stati i problemi e le motivazioni che hanno portato a determinate scelte e all'utilizzo delle soluzioni proposte.

4.1 Requisiti

Il sistema di comunicazione deve essere in grado di soddisfare le seguenti specifiche.

- Il sistema deve essere in grado di comunicare remotamente su reti non sicure e non condivise;
- L'utente deve essere in grado di scegliere la modalità di utilizzo del rover, tra le cinque elencate nella sezione 2.4, e visualizzare le immagini riprese dalla fotocamera;
- Il video deve essere riprodotto con una latenza bassa, ma anche con affidabilità;
- Il sistema deve essere pilotato da un unico utente e non da più utenti in parallelo;
- L'utente deve essere fortemente autenticato.

4.2 Analisi dei requisiti

Il sistema che si vuole proporre in questo progetto è un'applicazione servita da un web server ospitato sulla Jetson Nano. Si prende in considerazione questa soluzione perché la comunicazione non richiede di gestire un grande livello di traffico. Infatti questa deve essere unica, tra l'utente incaricato di assegnare modalità al rover, e il rover stesso.

Si ipotizza che un web server ospitato sulla Jetson Nano permetta di mantenere una latenza minore nella comunicazione con l'utente, che potrà accedere da qualunque browser, permettendo così di personalizzare il livello di sicurezza e offrire un controllo maggiore su di esso, e di eseguire la logica applicativa non appena il frame della camera viene catturato. Vari studi [26, 27] hanno dimostrato come un cloud server, alternativa alla soluzione proposta, anche se vantaggioso per scalabilità e manutenzione, sia influenzato dal carico in background di altre macchine virtuali, che con anche piccoli ritardi nella rete possono influenzare significativamente le prestazioni. Nel caso di applicazioni real-time questi effetti possono ridurre considerevolmente le performance.

4.2.1 Web framework e server

Framework

L'utente deve avere a disposizione un'interfaccia attraverso la quale scegliere le modalità e visualizzare il video ripreso dalla fotocamera. L'utente deve essere in grado di accedervi tramite una rete non sicura e con una trasmissione del video con una latenza bassa. Un'applicazione web permette all'utente di accedere all'interfaccia, dovunque si trovi e con tecniche efficaci e facilmente implementabili per proteggere la comunicazione.

Per fare questo bisogna scegliere un web framework. Scegliere un web framework offre molti vantaggi per lo sviluppo di un'applicazione web, come una maggiore produttività, grazie agli strumenti che mette a disposizione, la standardizzazione del contenuto , la sicurezza integrata, la scalabilità e il supporto della comunità.

Basandosi sul linguaggio dell'applicazione, Python, per semplicità di integrazione, si è limitata la scelta tra i due maggiori framework sviluppati con lo stesso linguaggio: Django e Flask.

Django [28] è un framework che fornisce una vasta gamma di funzionalità e strumenti integrati per lo sviluppo di applicazioni web. È molto adatto per progetti di grandi dimensioni e fornisce una forte protezione contro le minacce alla sicurezza web.

Flask [29] è un framework con un approccio più leggero e flessibile. È più adatto per progetti di piccole o medie dimensioni o per lo sviluppo di prototipi rapidi.

Django include una maggiore attenzione alla sicurezza rispetto a Flask grazie alla sua struttura rigida e alle sue funzionalità integrate di gestione della sicurezza. L'obiettivo principale di questo lavoro è quello di creare un sistema il più possibile sicuro, per cui questa caratteristica ha avuto un peso importante sulla scelta finale. Django fa della sicurezza un punto chiave del suo utilizzo, perché, grazie ai middleware che mette a disposizione, permette di proteggersi intrinsecamente da numerose vulnerabilità come SQL injection, cross-site scripting, cross-site request forgery e clickjacking. Django include funzionalità di autenticazione e autorizzazione integrate, mentre in Flask queste funzionalità devono essere configurate manualmente, con librerie esterne e possibili errori di configurazione.

Sia Django che Flask hanno una community attiva e un supporto adeguato. Questa caratteristica è molto importante quando si vuole creare un prodotto durevole e sicuro. Quando un sistema viene periodicamente controllato e aggiornato, ha una probabilità più bassa di presentare delle vulnerabilità. Essere revisionato da numerose persone ed essere al passo con le nuove minacce, permette di creare un prodotto più affidabile.

Flask è più leggero, adatto per progetti di piccole dimensioni, al contrario di Django che ha una struttura più pesante per progetti di piccole dimensioni come questo. La sua struttura rigida e al contempo solida, offre un vantaggio nel caso di progetti più complessi. Anche se per lo scopo di questa tesi non è necessario creare un grande progetto, lo sarà in futuro quando lo si completerà delle altre componenti necessarie per creare un rover funzionante. Una struttura come quella di Django sarà facilmente comprensibile e integrabile, diminuendo il rischio di configurazioni sbagliate e implementazioni poco efficienti.

La conclusione di questo confronto è stato scegliere Django come framework per lo sviluppo dell'applicazione.

Server

Per ospitare un'applicazione web, è necessario configurare un server. In base al ragionamento proposto all'inizio si è dovuto scegliere un web server in grado di funzionare all'interno della Jetson Nano, ma che non risulti pesante per occupazione di risorse e tempi di risposta. Uno studio [30] realizzato prendendo in considerazione due tra i web server più performanti, ha valutato la superiorità di Nginx per le stesse metriche proposte.

Nginx è un server web open source e un reverse proxy che viene spesso utilizzato per ospitare siti e applicazioni web, con una lunga storia di utilizzo a livello di produzione grazie alla sua affidabilità e stabilità. Noto per la sua elevata velocità e prestazioni. Nginx è progettato per scalare facilmente e gestire grandi quantità di traffico. Offre una serie di funzionalità avanzate, come la gestione delle connessioni persistenti e la gestione della cache, che lo rendono una soluzione versatile per molte esigenze.

4.2.2 Comunicazione

La comunicazione su reti non sicure e non condivise può comportare rischi significativi per la sicurezza. Infatti, la rete può essere compromessa da soggetti malintenzionati che possono intercettare e accedere alle informazioni sensibili trasmesse attraverso la rete. Per ridurre questi rischi, è importante implementare misure di sicurezza adeguate.

Protocolli di comunicazione

Il sistema deve garantire la latenza minore possibile. Per questo bisogna utilizzare un protocollo adatto allo streaming di contenuti e utilizzabile insieme ad un protocollo di sicurezza, che abbia la latenza minore possibile, ma che al contempo fornisca affidabilità allo streaming.

UDP è un protocollo connectionless e inaffidabile che fornisce una comunicazione veloce ed efficiente, ma senza un controllo degli errori. Inoltre viene facilmente bloccato dai firewall. UDP è molto vantaggioso per lo streaming multicast.

TCP è un protocollo affidabile e orientato alla connessione, che garantisce la consegna accurata dei dati ritrasmettendo i pacchetti persi o danneggiati. Stabilisce una connessione affidabile tra due dispositivi e garantisce che tutti i dati inviati vengano ricevuti nello stesso ordine in cui sono stati inviati. Permette solo connessione unicast.

È richiesto che i pacchetti utilizzino un protocollo di trasporto affidabile, poiché necessitano di attraversare una rete non sicura con informazioni potenzialmente sensibili. Non è necessario supportare il multicast, perché solo uno user può accedere alla telecamera. Affinché questo sia rispettato , è consigliato l'utilizzo di TCP, che offre più garanzie del primo. Poiché questa scelta va a pesare sulla latenza, è importante scegliere delle tecniche che siano in grado di attutirne la penalizzazione.

Esistono vari protocolli utilizzati per lo streaming di video online basati su TCP.

RTSP (Real-Time Streaming Protocol): è un protocollo di rete utilizzato per controllare la distribuzione di contenuti multimediali su Internet. È ampiamente supportato dalla maggior parte dei lettori multimediali ed è comunemente utilizzato nei sistemi di telecamere IP e in altre applicazioni di streaming a circuito chiuso. Non è direttamente utilizzabile per lo streaming sul browser, perché richiede l'utilizzo di HTTP.

HLS (HTTP Live Streaming): è un protocollo di streaming utilizzato per distribuire contenuti video e audio su Internet. Funziona dividendo il contenuto in segmenti più piccoli, che vengono poi consegnati al client tramite HTTP. Offre diversi vantaggi rispetto ad altri protocolli, come la scalabilità, l'adattabilità e la compatibilità con la maggior parte dei dispositivi e delle piattaforme.

DASH (Dynamic Adaptive Streaming over HTTP): è uno standard internazionale di streaming video utilizzato per la distribuzione di contenuti video e audio su Internet, simile a HLS (HTTP Live Streaming) e progettato per fornire vantaggi analoghi, oltre a un migliore supporto per lo streaming a bitrate adattivo, una migliore compatibilità e un migliore supporto per diversi formati.

Il problema di questi ultimi due protocolli, è che necessitano dei file già disponibili per realizzare lo streaming. È possibile implementare queste soluzioni creando i file attraverso OpenCV e la classe `VideoWriter()`, che prende in input i frame e crea dei blocchi temporanei di qualche secondo, che sono poi inviati e visualizzati sequenzialmente. Questo metodo, seppur personalizzabile grazie alla scelta della dimensione dei chunks, portata fino al minimo di 1 secondo, introduce una latenza non indifferente, sommata a quella dovuta alla trasmissione.

Per questo motivo è stata implementata un'altra soluzione, che permette di inviare il frame ripreso non appena elaborato da OpenCV. Questo permette di ridurre il carico sulla memoria e diminuire la latenza, affidandosi ad un protocollo sicuro come HTTP.

StreamingHttpResponse: è una classe Django che fornisce il supporto per lo streaming di risposte HTTP di grandi dimensioni nel framework web Django. Viene utilizzata per inviare una risposta in pezzi, invece di aspettare che l'intera risposta sia generata prima di inviarla al client. È utile per servire file di grandi dimensioni, inviare aggiornamenti in tempo reale o elaborare dati in modo incrementale. La risposta viene generata in pezzi non appena i dati sono disponibili, riducendo l'uso della memoria e migliorando i tempi di risposta per le risposte di grandi dimensioni.

Da sola però non offre gli stessi vantaggi che in combinazione con un algoritmo di compressione. Questo infatti permette di inviare la risposta compressa diminuendone la dimensione e il tempo di trasmissione. Infatti aggiungendo la compressione `gzip` a livello applicativo, solo ai frame del video che vengono trasmessi, la trasmissione è visibilmente più veloce.

Compressione	*latenza(s)*	*throughput(Mbps)*
Nessuna	3,56	40,42
GZIP	1,87	21,35

Tabella 4.1. Latenza e throughput medi calcolati con ciphersuite TLS_AES_256_GCM_SHA384.

Protocolli di sicurezza

Crittografia: la crittografia dei dati trasmessi in rete aiuta a proteggerli dall'intercettazione e dall'accesso da parte di utenti non autorizzati. I metodi utilizzati più comuni includono l'utilizzo di SSL/TLS per il traffico web e di VPN con IPSec per il traffico di rete.

Autenticazione: L'autenticazione dell'identità degli utenti e dei dispositivi che accedono alla rete può aiutare a prevenire gli accessi non autorizzati. Ciò può essere ottenuto con metodi quali l'autenticazione tramite password o l'autenticazione a due fattori.

Controlli di integrità: L'integrità dei dati trasmessi in rete garantisce che i dati non siano stati manomessi o modificati durante la trasmissione. Questo aspetto è importante nei casi in cui i dati trasmessi sono sensibili o critici, poiché anche piccole modifiche ai dati potrebbero avere conseguenze significative.

In questo caso i dati devono essere confidenziali, cioè non deve essere possibile visionare le immagini catturate dalla videocamera e le varie informazioni scambiate durante la trasmissione se non una volta arrivate all'utente o al veicolo. Se, per esempio, il rover dovesse lavorare in ambienti in cui le riprese contengono informazioni sensibili, queste non devono essere visionabili da chi non autorizzato. I dati devono essere integri, cioè è necessario che i pacchetti trasmessi non siano stati in alcun modo modificati. Se un pacchetto non integro fosse in grado di raggiungere il veicolo, per esempio, potrebbe prenderne il controllo. Inoltre sia l'utente che la Jetson Nano devono autenticarsi. Questa proprietà deve essere mantenuta da entrambi i capi della comunicazione, in modo da garantire all'uno di stare parlando con l'altro, e non con un eventuale attaccante non autorizzato.

Tutte queste proprietà possono essere garantite grazie all'utilizzo di un protocollo di sicurezza per la comunicazione. La scelta del giusto protocollo di sicurezza dipende da diversi fattori, tra

cui il tipo di dati trasmessi, le dimensioni e la complessità della rete, il livello di sicurezza richiesto, le risorse disponibili per l'implementazione e la manutenzione. I protocolli presi in considerazione per lo scopo sono elencati in seguito.

TLS (Transport Layer Security): TLS è un protocollo di sicurezza molto diffuso che garantisce la riservatezza, l'autenticazione e l'integrità dei dati mediante la crittografia dei dati e l'uso di certificati digitali per autenticare le parti coinvolte.

IPSec (Internet Protocol Security): IPSec è una suite di protocolli che garantisce la sicurezza delle comunicazioni IP. Garantisce la riservatezza, l'autenticazione e l'integrità dei dati mediante la crittografia dei dati e l'utilizzo di firme digitali per verificarne l'autenticità.

SRTP (Secure Real-Time Transport Protocol): è un protocollo di sicurezza per la protezione dei dati in tempo reale trasmessi su reti IP, come voce e video. SRTP garantisce la riservatezza, l'autenticità e la protezione dell'integrità dei dati, utilizzando meccanismi di crittografia, autenticazione dei messaggi e protezione da replay. È un'estensione di RTP (Real-time Transport Protocol), un protocollo di trasporto per fornire comunicazioni multimediali sicure in tempo reale.

Il problema di SRTP, come esplicitato nello standard [31] che lo definisce, è che utilizza per l'autenticazione dei messaggi la trasformazione predefinita HMAC-SHA1. SHA1 è deprecato nel 2011, ma ancora molto diffuso. Per questo motivo è stata esclusa la possibilità di utilizzare questo protocollo. Ciò che rende diversi IPSec e TLS è il livello in cui sono implementati. Il primo si appoggia sul livello 3, e rappresenta un insieme di strumenti che sono utilizzati per proteggere il protocollo IP. Sostanzialmente IPSec proteggere tutto il traffico di rete, proprio a causa del suo livello nello stack protocollare. Al contrario, TLS, lavora sopra al protocollo di trasporto, TCP. Questo rende il sistema più vulnerabile ad attacchi DoS, perché i controlli di sicurezza sono effettuati più in alto nello stack protocollare, di conseguenza i pacchetti malevoli sono scartati dopo. Il problema di IPSec, come evidenziato anche in [32], è che introduce un overhead maggiore al pacchetto, introducendo anche ritardi nella trasmissione. TLS è un compromesso tra velocità e sicurezza. Per proteggere il sistema da eventuali attacchi di tipo DoS, si prevede l'utilizzo di strumenti di filtering a livelli inferiori.

4.2.3 Protezioni aggiuntive

Un'ulteriore misura che permette di proteggere un sistema che necessita di comunicare con l'esterno è la configurazione di uno o più firewall.

Firewall: i firewall possono essere utilizzati per limitare l'accesso alla rete e prevenire gli accessi non autorizzati. Possono quindi aiutare a prevenire l'ingresso di traffico dannoso nella rete o limitare il traffico in uscita.

Un firewall è un sistema di sicurezza di rete che monitora e controlla il traffico di rete in entrata ed in uscita in base a regole di sicurezza predefinite. Servono a impedire l'accesso non autorizzato ad una rete e a proteggerla da potenziali minacce alla sicurezza, come malware, virus e tentativi di hacking. I firewall possono essere basati su hardware, software o una combinazione di entrambi. Funzionano ispezionando i pacchetti di dati trasmessi su una rete e consentendo o negando l'accesso in base alle regole di sicurezza definite. Esistono diversi tipi di firewall in base al livello su cui agiscono, con vantaggi e svantaggi.

Firewall a livello di rete: sono relativamente semplici e veloci, poiché ispezionano solo l'intestazione dei pacchetti di dati, rendendoli adatti a reti di grandi dimensioni con alti livelli di traffico; vengono utilizzati per controllare l'accesso in base ad attributi specifici dei pacchetti di dati, come gli indirizzi IP di origine e di destinazione, i numeri di porta e il tipo di protocollo; hanno lo svantaggio che ispezionano solo l'intestazione dei pacchetti di dati, il che può limitare la loro capacità di rilevare e prevenire alcuni tipi di minacce alla sicurezza.

Firewall a livello di trasporto: ispezionano sia l'intestazione che il payload dei pacchetti di dati, consentendo di decidere in modo più dettagliato e informato se consentire o negare l'accesso; mantengono un registro di stato di ogni connessione di rete, consentendo di comprendere il contesto della comunicazione e di prendere decisioni più informate sull'opportunità di consentire o negare l'accesso; risultano più complessi dei precedenti e possono richiedere una maggiore potenza di elaborazione per eseguire le ispezioni.

Firewall a livello applicazione: consentono il controllo più granulare sul traffico di rete, in quanto possono ispezionare e prendere decisioni in base al tipo di applicazione utilizzata e possono proteggere dalle minacce alla sicurezza che sono specifiche di un particolare tipo di applicazione; possono richiedere molte risorse, perché richiedono la capacità di ispezionare e comprendere il contenuto dei pacchetti di dati a livello di applicazione.

L'utilizzo di firewall a più livelli può fornire una difesa più completa contro le minacce alla sicurezza.

Una misura utile nel momento in cui si configura un server è il reverse proxy server.

Reverse Proxy Server: funge da intermediario tra server e client. Riceve le richieste dai client e le inoltra ai server appropriati. Un reverse proxy può essere utilizzato per migliorare le prestazioni, la sicurezza e l'affidabilità della rete, fornendo funzioni come controllo del traffico in entrata, bilanciamento del carico e servizi di crittografia.

Un reverse proxy può fornire diversi vantaggi in questo contesto, tra cui:

- distribuire il traffico in entrata su più server, migliorando l'affidabilità e la scalabilità dell'infrastruttura web;
- memorizzare nella cache i contenuti statici, come immagini, file CSS e JavaScript, riducendo il carico sui server di backend e migliorando le prestazioni del sito web;
- funzionare da barriera tra l'Internet pubblico e i server di backend, fornendo funzioni di sicurezza come la terminazione SSL/TLS, il filtraggio delle richieste HTTP e il blocco degli indirizzi IP;

Quando usati insieme, un firewall ed un reverse proxy server, possono fornire molteplici livelli di sicurezza per la rete. Un firewall che opera al livello di rete serve a controllare il traffico in entrata e in uscita in base ai protocolli del livello, come IP. Un reverse proxy server opera al livello applicativo e serve a controllare il traffico in entrata verso il server, filtrando per esempio sul tipo di richiesta o sull'URL.

In generale però, non è necessario nascondere un server dietro un reverse proxy se questi si trovano sulla stessa macchina e il server è unico. In questo caso, i vantaggi del suo utilizzo, come il filtraggio del traffico e la scalabilità, non sono così rilevanti come lo sarebbero in una rete più complessa con più server. Inoltre, l'uso di un proxy inverso in questo scenario aumenta la complessità della rete.

Nginx [33] è un HTTP e reverse proxy server che permette di gestire più efficacemente le richieste provenienti dall'esterno servendo i file statici ai client quando richiesto. In particolare Nginx, come reverse proxy server, rappresenta un layer aggiuntivo tra il mondo esterno e l'applicazione in sè. Permette di gestire più efficacemente il carico, di aumentare la tolleranza in caso di errori e fornisce il supporto per il protocollo TLS.

4.3 Ulteriori misure di sicurezza nelle applicazioni web

Quest'analisi è stata svolta prendendo in esame la lista OWASP Top 10 [10]. Questa fa parte di un progetto che si occupa di redarre ogni anno un documento, riconosciuto a livello mondiale contenente le categorie dei rischi più critici del periodo per le applicazioni web. Sono stati presi in esame i rischi, valutandone l'importanza e le misure consigliate da implementare per proteggere il sistema da essi. Sulla base delle indicazioni sono state applicate al sistema determinate misure.

4.3.1 Controllo degli accessi compromesso

Si tratta dei rischi dovuti a delle configurazioni sbagliate nel sistema di autenticazione che permettono ad eventuali malintenzionati di accedere al sistema senza averne i permessi. Ci sono alcune tecniche che si possono utilizzare per proteggersi da queste vulnerabilità.

Si devono implementare meccanismi di autenticazione, eliminando l'utilizzo di risorse pubbliche e limitando l'utilizzo di Cross-Origin Resource Sharing. Bisogna controllare che l'utente sia autenticato in ogni momento e che le risorse a cui tenta di accedere appartengano ad esso. Dopo il logout da un server stateless, inoltre, l'identificatore dovrebbe essere invalidato.

Analisi

Come anche anticipato nel capitolo precedente, si è dovuto aggiungere un sistema di autenticazione. Poiché è richiesto un grado abbastanza elevato di sicurezza nell'accesso all'applicazione da parte del client, l'utilizzo del solo sistema di autenticazione tramite password non soddisfa i requisiti, oltre ad essere scoraggiato.

L'autenticazione a più fattori, MFA, è un processo di sicurezza che richiede più di un metodo di autenticazione da categorie diverse per verificare l'identità dell'utente per un login o un'altra transazione. L'MFA aiuta a garantire che una persona che dichiara di essere un utente sia effettivamente tale e a prevenire l'accesso non autorizzato a informazioni sensibili. Prevede la combinazione di 2 o più tra le tre di seguito elencate:

- qualcosa che si conosce;
- qualcosa che si possiede;
- qualcosa che si è.

Il primo corrisponde ad un'autenticazione come quella con password, mentre la seconda si riferisce al possedimento materiale di qualcosa, come una smart card contenente un certificato. L'ultima categoria ingloba il tipo di autenticazione che prevede l'utilizzo di dati biometrici, come le iridi o le impronte digitali.

Vari studi hanno comprovato l'efficacia dell'autenticazione a due fattori, tra cui [34], [35], [36], e valutato diverse combinazioni tra le categorie elencate [34]. In base alle risorse disponibili e agli studi che ne certificano la validità, si è optato per l'autenticazione a due fattori tramite l'utilizzo di certificato di chiave asimmetrica e password. La scelta di utilizzare un certificato è possibile anche perché il sistema prevede un numero limitato di utenti con l'autorizzazione per accedervi. Come futuro sviluppo del progetto si potrebbe prevedere l'utilizzo di una smart-card per contenere la chiave asimmetrica.

In ogni momento, prima renderizzare la pagina, il sistema deve controllare che l'utente sia autenticato, e che sia autorizzato ad accedere alle risorse richieste. Per fare questo si effettua un controllo sullo user all'interno della view in Django. Se rileva un utente non autenticato, allora lo renderizza alla pagina di login. In questo modo tutte le risorse sono limitate ad utenti autenticati.

Il Cross-Origin Resource Sharing (CORS) è un meccanismo di sicurezza che consente a una pagina web di effettuare richieste ad un dominio diverso da quello di origine. I browser vietano di default di effettuare richieste ad un dominio diverso per motivi di sicurezza. CORS fornisce al server un modo per allentare tale politica e consentire alla pagina di effettuare richieste a un dominio diverso. All'interno dell'applicazione non è necessario effettuare richieste a domini differenti.

4.3.2 Fallimenti crittografici

I fallimenti crittografici sono situazioni in cui un sistema crittografico non è più in grado di garantire la sicurezza dei dati che dovrebbe proteggere. Le ragioni possono essere molte, come l'utilizzo di chiavi deboli o algoritmi rotti.

È importante identificare con precisione i dati sensibili, in modo da proteggerli adeguatamente e garantire la conformità alle leggi e alle normative vigenti. Bisogna assicurarsi di criptarli se immagazzinati e non immagazzinarli se non necessario. In particolare è sconsigliato l'utilizzo della cache per risposte che ne contengano, per evitarne l'utilizzo malevolo.

È necessario proteggere i dati sensibili durante le trasmissioni utilizzando protocolli di crittografia solidi, tenendo conto anche della vita del prodotto e per quanto tempo deve potenzialmente essere protetto. Per proteggere le trasmissioni sono consigliati protocolli come TLS, con forward secrecy, indicando una priorità dei cifrari e dei parametri di sicurezza nella contrattazione tra client e server. Utilizzando feature come HSTS (HTTP Strict Transport Security), che permette di obbligare a fornire il servizio tramite HTTPS, reindirizzando qualunque tentativo di connessione con HTTP. Al posto della sola crittografia, sarebbe da utilizzare la crittografia autenticata, che, non solo cripta i dati, ma fornisce anche un meccanismo per rilevare se i dati sono stati manomessi. Ciò si ottiene includendo un codice di autenticazione del messaggio (MAC) nei dati crittografati. Il MAC viene generato utilizzando una chiave segreta e viene utilizzato per verificare l'integrità dei dati al momento della decifrazione.

È importante che le password vengano memorizzate non in plain text, ma con salt e con funzioni di hashing forti.

Analisi

La password, usata per l'autenticazione dell'utente, deve essere memorizzata in modo sicuro. Una funzione hash è una funzione matematica unidirezionale che prende un input, la password, e lo converte in una stringa irreversibile di caratteri di lunghezza fissa. Lo stesso input produrrà sempre lo stesso valore e anche la minima modifica produrrà un hash completamente diverso. Il salt è un valore casuale che viene generato e memorizzato insieme all'hash di una password. Quando un utente inserisce la propria password, lo stesso sale viene utilizzato in combinazione ad essa, per eseguirne l'hash. Il valore viene confrontato con quello memorizzato. In questo modo si limita l'efficacia di tabelle precalcolate, le rainbow table, per decifrare le password. Infatti si dovrebbe applicare la funzione di hash per ogni singolo sale. Le linee guida del NIST [37] consigliano di utilizzare funzione di derivazione della chiave PBKDF2 con HMAC-SHA256 come hashing interno.

Ai sensi del GDPR [9], la regolamentazione europea sulla privacy, sono informazioni private quelle che possono identificare direttamente una persona. Si parla di informazioni sensibili se queste rivelano convinzioni religiose, origine razziale o etnica, opinioni politiche o orientamento sessuale. Poiché il rover possiede la capacità di individuare persone, le immagini devono essere protette in transito tra il server e il client in modo da evitare fuga di dati sensibili. Per questo motivo è necessario utilizzare un protocollo sicuro per il trasferimento. Questa analisi è stata fatta nella sezione 4.2.2.

TLS, scelto come protocollo di sicurezza, ha quattro versioni, di cui due obsolete. Inoltre, la versione 1.2 [38], presenta varie vulnerabilità conosciute [39] e risolte nella sua versione successiva, TLSv1.3[40, 41]. Questa versione non solo è più sicura, ma è anche più veloce, grazie all'handshake velocizzato.

Per evitare che un malintenzionato effetti un downgrade alla versione 1.2, o peggio, alla 1.1 o 1.0, per poter sfruttare le loro vulnerabilità, si limita la funzionalità del server all'utilizzo del protocollo TLSv1.3. In questo modo, se un utente vuole comunicare con il server, sarà costretto ad utilizzare il protocollo più sicuro. Anche se non ancora completamente compatibile con tutti i browser, questa versione è supportata da quelli più diffusi e sicuri. Inoltre è previsto il raggiungimento della piena compatibilità nei prossimi anni.

Quando un client contatta un server, inizia una contrattazione tra i due sulla chiphersuite da utilizzare. Per evitare che vengano utilizzati degli algoritmi obsoleti e vulnerabili, si prende la precauzione di inserire solo combinazioni raccomandate nella lista di quelle supportate. Le ciphersuite di algoritmi che il protocollo può utilizzare sono state selezionate tra quelle definite in RFC8446 [40] e nelle linee guida nel NIST [42]. Inoltre è possibili indicare che il server abbia precedenza rispetto al client sulla scelta dei cifrari. I cifrari sono i seguenti:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

Il primo è obbligatorio, gli altri due sono raccomandati. AES_128 e AES_256 si riferiscono all'algoritmo Advanced Encryption Standard (AES) con una chiave rispettivamente di 128 bit e di 256 bit. AES è un algoritmo di crittografia simmetrica ampiamente utilizzato e considerato altamente sicuro. GCM è l'acronimo di Galois/Counter Mode, una modalità di funzionamento della crittografia AES che garantisce l'autenticazione e la riservatezza. SHA256 e SHA384 si riferisce al Secure Hash Algorithm (SHA), una funzione di hash crittografico ampiamente utilizzata che fornisce un output di dimensioni fisse, 256 o 384 bit, per qualsiasi dato in ingresso. L'hash SHA_256 è utilizzato per verificare l'integrità dei dati trasmessi tra client e server. CHACHA20 è un cifrario a flusso utilizzato per crittografare i dati e considerato altamente sicuro. POLY1305 è un algoritmo che fornisce l'autenticazione dei messaggi. Insieme formano un algoritmo di crittografia con algoritmo di dati aggiuntivo definito in [43].

La velocità è stata valutata utilizzando sequenzialmente i tre algoritmi nello stesso ambiente di test e realizzando dieci misurazioni ciascuno, valutando i dati trasmessi, i secondi durante i quali sono stati inviati e la latenza del video trasmesso. I valori sono ottenuti mediando il ritardo approssimando alla cifra più significativa. 4.2

chiphersuite	*latenza(s)*	*throughput(Mbps)*
TLS_AES_128_GCM_SHA256	1,84	23,42
TLS_AES_256_GCM_SHA384	1,87	21,35
TLS_CHACHA20_POLY1305_SHA256	1,80	25,38

Tabella 4.2. Latenza e thorughput medi in base agli algoritmi utilizzati.

Le misurazioni complete si possono trovare nell'appendice B Queste misure sono da tenere in conto nella configurazione del server, in cui l'ordine di apparizione ne determina la preferenza nella contrattazione con il client.

4.3.3 Injection

L'injection è un tipo di vulnerabilità di sicurezza che si verifica quando un aggressore è in grado di iniettare codice dannoso in un'applicazione software, causando l'esecuzione di un comportamento non previsto. Possono presentarsi in diverse forme, tra cui SQL injection, command injection e code injection.

Per prevenire questo tipo di attacchi, bisogna sempre validare gli input, e farlo sul server. Inoltre è buona norma usare query parametrizzate invece di quelle dinamiche, e nel secondo caso validare e sanificare i valori.

Analisi

L'unico accesso al database viene effettuato durante il login. Per quanto riguarda gli input, è stato utilizzato lo stesso tipo di validazione discussa nel capitolo precedente, nella sezione 3.4.1, all'interno della funzione che riceve il parametro dal server, prima di renderizzare. La validazione in questo modo viene fatta sul server, e si può confidare in essa. Avendo aggiunto il sistema di autenticazione, anche i valori dello username e della password devono essere controllati.

Lo username può corrispondere alla mail o ad una sequenza di caratteri alfanumerici. Nel primo caso il controllo deve essere fatto sui caratteri, sulla lunghezza e sulla formattazione, seguendo i criteri forniti dallo standard [44]. L'espressione regolare utile per validarne il contenuto è la seguente:

```
pattern = re.compile("^[\w-\.]{1,64}@([\w-]+\.){1,253}[\w-]{2,4}$")
```

È necessario definire alcuni criteri basilari per la validazione di una password. In particolare vengono utilizzati all'interno del codice i seguenti criteri, in accordo con le linee guida del NIST [37]:

- la password deve essere lunga almeno 8 caratteri, con un massimo di almeno 64 caratteri;
- la password deve contenere almeno un numero, una lettera minuscola, una lettera maiuscola ed un carattere speciale.

```
pattern = re.compile("^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)
       (?=.*[@$!%*?&\s+.-])[A-Za-z\d@$!%*?&\s+.-]{8,64}$")
```

Se questi controlli non vanno a buon fine, non si arriva ad interrogare il database.

4.3.4 Design insicuro

La progettazione insicura non è una vulnerabilità o un problema specifico, ma piuttosto una mancanza di controlli di sicurezza nella progettazione del software. Ciò può comportare vulnerabilità e punti deboli che possono essere sfruttati dagli aggressori. Per ridurre il rischio di una progettazione non sicura, è importante considerare le minacce e i rischi potenziali per il software o il sistema durante la fase di progettazione e implementare i controlli di sicurezza necessari per affrontare tali rischi. Quanto affermato rafforza l'importanza dell'argomento affrontato in questa tesi. Prendere in considerazione la sicurezza lungo tutto il ciclo di vita di un prodotto è di vitale importanza, perché, in assenza di un'analisi approfondita, i rischi aumentano.

4.3.5 Configurazione errata della sicurezza

La configurazione errata della sicurezza può rappresentare un rischio per l'applicazione. Ciò può accadere se non vengono applicate le appropriate misure di sicurezza dell'applicazione o se le autorizzazioni sulle risorse sono configurate in modo non corretto. Inoltre, se sono abilitate o installate funzionalità non necessarie, come porte, servizi, pagine, account, l'applicazione potrebbe essere vulnerabile. L'utilizzo di account e password predefiniti rappresenta un altro rischio. Inoltre, se l'applicazione rivela informazioni troppo dettagliate agli utenti, potrebbe essere vulnerabile.

Il processo di hardening è l'insieme di attività volte a rendere più sicuro un sistema o un'applicazione riducendone la superficie di attacco. Ciò si ottiene rimuovendo o disabilitando le funzioni, i componenti e i servizi non necessari che possono essere sfruttati dagli aggressori. L'invio di direttive di sicurezza ai client è un aspetto importante nel processo di hardening dell'applicazione. Le intestazioni di sicurezza sono un tipo di intestazione di risposta HTTP che può essere utilizzata in questo caso. Forniscono una protezione aggiuntiva contro varie minacce alla sicurezza, tra cui cross-site scripting (XSS), cross-site request forgery (CSRF) e clickjacking.

Analisi

Le intestazioni che possono essere aggiunte alla risposta HTTP da prendere in esame sono:

- X-XSS-Protection: è utile nel prevenire gli attacchi di cross-site scripting impedendo la renderizzazione della pagina con codice malevolo;
- X-Content-Type-Options: aiuta a prevenire gli attacchi di tipo MIME sniffing impedendo al browser di interpretare male il contenuto;
- X-Frame-Option: per evitare attacchi di clickjacking, indicando al browser di non renderizzare pagine contenute all'interno di un frame;

- HTTP Strict Transport Security (HSTS): aiuta a proteggere dagli attacchi man-in-the-middle imponendo l'uso di connessioni sicure HTTPS tra un sito web e i suoi utenti;
- Content Security Policy (CSP): aiuta a prevenire gli attacchi di tipo cross-site scripting e altri attacchi di tipo code injection, specificando quali fonti di contenuto possono essere caricate ed eseguite da una pagina web.

Inviando queste intestazioni, il server può informare il browser del client sul comportamento che deve mantenere durante il rendering della pagina e delle restrizioni da applicare alle richieste effettuate da essa. Come protezione aggiuntiva si aggiungono queste intestazioni.

La prima intestazione, se abilitata, può far si che, se identificato un attacco di cross-site scripting, sanifichi o blocchi la pagina. In realtà questo approccio, utilizzato per proteggere i vecchi browser, in quelli nuovi può portare a vulnerabilità XSS. Per questo motivo non viene più supportato dalla maggioranza di essi, ma è stato rimpiazzato dal Content Security Policy, con l'opzione di non ammettere `unsafe-inline`.

Per quanto riguarda il X-Content-Type-Option, l'opzione `nosniffing` permette di bloccare richieste di tipo stile che non sono di tipo css o script.

X-Frame-Options invece è un header reso obsoleto dalla direttiva inseribile all'interno di CSP `frame-ancestors`.

HSTS, RFC-6797 [45], è un'intestazione che indica al browser che il dominio deve essere raggiungibile solo tramite HTTPS. Si tratta di una direttiva molto più forte della semplice ridirezione da HTTP ad HTTPS. Infatti in quest'ultimo caso la connessione iniziale è comunque insicura. Questo header permette di imporre la trasformazione prima di iniziare.

Il CSP ha una grande rilevanza in questo contesto perché permette, con la policy corretta, di identificare e bloccare svariati attacchi. Non servono risorse esterne, per questo motivo bastano `default-source` e `self`. In più si aggiunge la direttiva `always` per far si che questo header venga sempre inserito.

4.3.6 Componenti vulnerabili e datati

I componenti vulnerabili e obsoleti si riferiscono a librerie, framework e strumenti software che non sono più mantenuti o supportati dai loro creatori e che presentano vulnerabilità di sicurezza note. Questi componenti possono essere utilizzati in applicazioni e sistemi software, rendendoli vulnerabili agli attacchi. L'uso continuato di questi componenti può rappresentare un rischio significativo per la sicurezza dell'applicazione software e dei suoi utenti. È importante tenere traccia di questi componenti e aggiornarli o sostituirli con versioni più recenti e sicure.

Analisi

Fin dall'inizio si è presa in seria considerazione la sicurezza dell'ambiente di sviluppo, ma la versione della Jetson Nano utilizzata, realizzata dal produttore Yahboom, per problemi di compatibilità, obbliga ad utilizzare un sistema operativo che non è aggiornato all'ultima versione. Infatti si basa ancora su una versione di Ubuntu 18.04. Questo è un grave rischio per l'intero sistema, che impone di mettere in atto tutte le misure di sicurezza necessarie per proteggerlo, aggiornando tutti i pacchetti alle ultime versioni e impegnandosi ad aggiornare il sistema non appena rilasciata la versione successiva.

4.3.7 Fallimenti nell'identificazione e nell'autenticazione

L'autenticazione è un processo importante per garantire che solo gli utenti autorizzati abbiano accesso alle risorse protette dell'applicazione. Per garantire ciò, bisogna avere la conferma dell'identità, possibilmente utilizzando un'autenticazione a due fattori e credenziali forti, di cui si è parlato precedentemente. Inoltre è necessario controllare che qualunque accesso ad una risorsa sia preceduta da un controllo sull'utente.

Analisi

L'autenticazione è stata analizzata precedentemente. Prima di attivare la videocamera inoltre, si effettua un controllo sull'utente e sulla sua autenticazione. In questo modo si limita il suo utilizzo ad utenti autorizzati.

4.3.8 Fallimenti nell'integrità dei dati e del software

I problemi di integrità del software e dei dati si riferiscono a problemi relativi ai repository, alle librerie e ad altri fallimenti relativi all'infrastruttura. Per mitigare questi problemi bisogna assicurarsi ti stare utilizzando risorse sicure e fidate, utilizzando meccanismi di verifica e meccanismi di controllo lungo la vita del prodotto.

Analisi

È stata effettuata un'analisi delle vulnerabilità delle librerie utilizzate all'interno del software nella sezione 3.4.1. Inoltre sono state valutate le vulnerabilità che affliggono il web framework, Django. La versione 4.1.7, utilizzata all'interno del progetto, non presenta vulnerabilità conosciute secondo il National Vulnerability Database [18]. Neanche le librerie aggiuntive utilizzate ne contengono attualmente.

4.3.9 Security logging e monitoraggio dei fallimenti

Si tratta dei problemi derivanti dalla configurazione sbagliata, o l'assenza, di log e informazioni di monitoraggio. Bisogna provvedere ad un sistema di log che indichi abbastanza informazioni per identificare i problemi o gli utenti che li hanno causati, ma senza rivelare informazioni sensibili.

Analisi

I file di log, di cui si è parlato anche nel capitolo precedente, sono importanti per il monitoraggio del funzionamento, ma anche della sicurezza. Sono particolarmente importanti i log generati durante la fase di login dell'utente. Per esempio, il tentativo fallito ripetuto dovrebbe essere comunicato all'amministratore di sistema, per esempio via mail. È possibile prevedere di inserire tale meccanismo per un lavoro futuro.

4.3.10 Falsificazione della richiesta lato server

La Server-Side Request Forgery (SSRF) è un tipo di vulnerabilità di sicurezza che permetta ad un attaccante di obbligare il server a fare una richiesta non intenzionale, per esempio a servizi interni. Per limitare la possibilità di eseguire questo tipo di attacco si possono utilizzare delle tecniche di limitazione del traffico, per esempio con firewall con policy 'deny by default'. Bisogna inoltre assicurarsi che gli tutti gli input siano validati, che gli URL accessibili siano limitati, possibilmente con una whitelist, disattivare tutti gli indirizzamenti HTTP.

Analisi

Le mitigazioni sopra elencate sono state prese già prese in esame nelle sezioni precedenti. Sono stati presi in considerazione tecniche di filtraggio del traffico nella sezione 4.2.3 valutando la possibilità di inserire firewall e reverse proxy server. La validazione dell'input è stata discussa nella sezione 3.4.1 del capitolo precedente. Per quanto riguarda gli URL disponibili, si limita la vulnerabilità facendo si che il server sia abilitato solo per `https://`. Se si riceve qualunque altro tipo di richiesta, questa viene bloccata.

Capitolo 5

Configurazione del sistema di comunicazione

L'obiettivo di questo capitolo è mostrare come è avvenuta la configurazione del sistema applicando le misure prese in esame nel capitolo precedente.

5.1 Configurazione

Il lavoro è stato condotto utilizzando uno user senza privilegi amministrativi. Questa scelta si riconduce al principio dei privilegi minimi citato negli scorsi capitoli. Infatti in questo modo si preclude l'accesso diretto all'account di root, proteggendolo da eventuali accessi non autorizzati.

Il sistema operativo installato sulla Jetson Nano si accende inizialmente con l'utente root automaticamente disabilitato. Per questo motivo non è stato necessario creare un ulteriore utente. Un passo necessario, al contrario, è stato quello di modificare le credenziali di accesso, in modo da eliminare quelle di fabbrica e fornire una password più sicura.

5.1.1 Firewall

In seguito è stato configurato un firewall per il traffico in entrata. In questo modo si possono bloccare pacchetti estranei prima di raggiungere il reverse proxy. La policy adottata prevedere una whitelist basata sulle porte e sui servizi offerti, ovvero tutto il traffico deve essere bloccato, a meno di provenire dalle porte 443 e 22, rispettivamente in ascolto di pacchetti HTTPS e SSH.

Per fare questo è stato utilizzato lo strumento di configurazione per il firewall di Ubuntu, `Uncomplicated Firewall` [46]. Questo strumento offre un front-end che permette di configurare efficacemente e facilmente la tabella degli indirizzi IP e dei servizi offerti. Per la configurazione sono stati utilizzati i comandi elencati in seguito.

```
sudo ufw default deny incoming
sudo ufw default deny outgoing
sudo ufw allow 22
sudo ufw allow 443
```

Questi hanno permesso di raggiungere una configurazione in grado di bloccare qualunque traffico in ingresso, tranne il traffico SSH, necessario in fase di sviluppo per accedere al terminale remotamente, e il traffico HTTPS, poiché l'applicazione deve comunicare attraverso questo protocollo. Il firewall è da abilitare con il seguente comando:

```
sudo ufw enable
```

```
jetson@jetson-desktop:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
443                        ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
```

Figura 5.1. Uncomplicated Firewall status

5.1.2 Virtual enviroment

Un buona pratica quando si lavora ad un'applicazione in Python è quello di creare un ambiente virtuale. Questo permette di gestire le dipendenze tra pacchetti ed evitare eventuali conflitti derivanti dalle loro installazioni. Uno dei vantaggi che questo meccanismo apporta è che qualunque pacchetto installato all'interno dell'ambiente virtuale rispetta l'isolamento e non verrà installato a livello globale, evitando per esempio la possibile corruzione di ciò che viene utilizzato per le funzioni interne al sistema operativo.

È stato utilizzato per questo proposito il modulo venv [47]. L'ambiente è stato creato utilizzando i comandi a seguire.

```
mkdir thesisproject
cd thesisproject
python3 -m venv app_env --system-site-packages
```

Quest'ultimo comando, in particolare, crea l'ambiente con l'opzione –system-site-packages, che fa si che la variabile include-system-site-packages sia settata a 'true'. Questa variabile permette all'ambiente di avere accesso ai pacchetti già installati all'interno della cartella site-packages. In alternativa si possono installare nuovamente i pacchetti utilizzando il comando `pip` Per iniziare a lavorare all'interno dell'ambiente virtuale è necessario attivarlo con

```
source app_env/bin/activate
```

Una volta attivato l'ambiente, si potrà procedere alla creazione dell'applicazione.

5.1.3 Django

Per creare il progetto di Django si è seguita la seguente procedura:

```
pip install django
django-admin startproject remotecommunication ~/thesisproject
```

dove remotecommunication è il nome dato alla cartella creata. La cartella contente il progetto di Django e quella contente i file dell'ambiente virtuale si devono trovare entrambi nella stessa directory.

Il software, citato e modificato nei capitoli precedenti, è stato adattato in modo da funzionare all'interno del web framework. Se ne approfondirà la struttura all'interno della sezione 5.2.

All'interno della cartella remotecommunication, si può trovare il file settings.py. In questo sono elencate varie informazioni e impostazioni di Django che possono essere configurate in base alle necessità. Di seguito si elencano i cambiamenti effettuati sulla configurazione.

Alla creazione dell'applicazione `remotecommunication` è necessario aggiungerla tra le applicazioni installate nel framework

```
INSTALLED_APPS = [
    ...
    'livestreaming',
    ...
]
```

Un'altra impostazione, `ALLOWED_HOST`, è stata modificata per proteggere il sistema da attacchi di tipo HTTP Host Header Injection. Infatti, se l'header non viene correttamente validato, un attaccante potrebbe modificarne il valore inserendo del codice malevolo al suo interno. Questa parola chiave può essere seguita da una lista di indirizzi IP e domini validi per l'applicazione.

```
ALLOWED_HOSTS = ['192.168.1.235', 'localhost']
```

La chiave segreta creata ed inserita automaticamente all'interno di `settings.py` alla creazione, è stata sostituita da una chiave generata come segue all'interno dell'ambiente virtuale:

```
python3 manage.py shell
```

All'interno della shell aperta si utilizzano i comandi

```
from django.core.management.utils import get_random_secret_key
print(get_random_secret_key()) #Genera una nuova chiave segreta
```

Questa deve essere inserita in un file `.env` creato all'interno del progetto con `SECRET_KEY = <chiave segreta>` al suo interno Nel file `settings.py`

```
SECRET_KEY = config('SECRET_KEY')
```

Questo processo permette di avere una chiave sicura e non salvata in plaintext all'interno del progetto.

I file statici, quali .css e .js, sono stati inseriti all'interno di una cartella nominata `static` con path `thesisproject/remotecommunication/static`. All'interno del file di configurazione sono state aggiunte le seguenti direttive nell'ordine in cui appaiono:

```
STATICFILES_DIRS = [ os.path.join(BASE_DIR, 'remotecommunication/static/')]
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

Dopo aver aggiornato queste impostazioni è possibile eseguire i seguenti comandi all'interno dell'ambiente virtuale. `cd thesisproject`

```
python3 manage.py makemigrations
```

```
python3 manage.py migrate
```

```
python3 manage.py collectstatic
```

I comandi precedenti applicano le modifiche effettuate al database e collezionano i file statici all'interno della cartella definita in `STATIC_URL`

```
python3 manage.py createsuperuser
```

Dopo aver fornito i dati richiesti quali username, email e password, questo comando attiva l'interfaccia admin automatica. Quest'ultima è accessibile tramite l'URL `<dominio/IP>/admin/` e permette allo user appena definito di gestire il contenuto del sito.

Per questo progetto è stato realizzato un sistema di login per utenti senza privilegi di admin. Le specifiche si possono trovare nella sezione 5.2. Si è scelto di creare un sistema in cui i nuovi utenti vengono registrati manualmente dall'amministratore, che fornirà loro anche un certificato per la crittografia asimmetrica, come deciso nella sezione 4.3.1 e spiegato nella sezione 5.1.6. Questo è accettabile poiché l'applicazione, creata ad hoc per poter comunicare a distanza con il veicolo di tipo rover, deve essere accessibile da un numero ristretto di utenti.

5.1.4 Gunicorn

A seguire è stato necessario installare e configurare Gunicorn. Django viene fornito con un Web Server Gateway Interface (WSGI) Server integrato. Questo però deve essere sostituito per ragioni di sicurezza, poiché si tratta di un server per il quale non è stato condotto un security audit o un penetration test. Per questo motivo il server integrato è stato sostituito da un Python WSGI HTTP Server chiamato Gunicorn [48]. I motivi per cui è stato scelto questo application sever sono i seguenti:

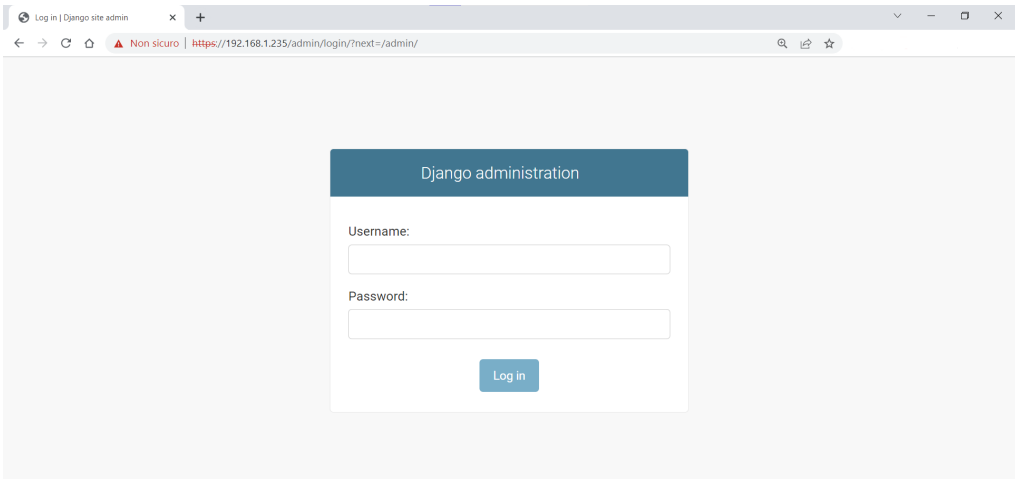

Figura 5.2. Admin login interface

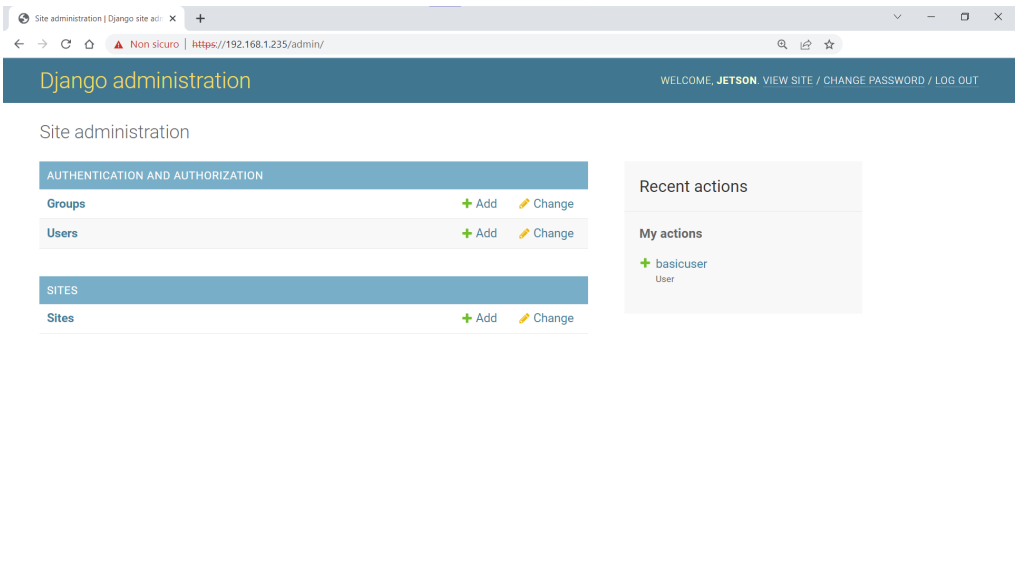

Figura 5.3. Admin interface

- Supporta nativamente Django
- È veloce ed ottimizzato
- Permette un controllo più incisivo sulle funzionalità dell'applicazione
- Ha un sistema di logging integrato e completo

All'interno dell'ambiente virtuale `app_env` si installa il WSGI server:

```
pip install gunicorn
```

Dopo aver installato il pacchetto, si procede creando due file all'interno della cartella di sistema /etc/systemd/system/. Systemd è uno strumento di linux che permette di gestire i processi, le risorse e i permessi per i servizi di sistema. Questi due file devono contenere le configurazioni e le istruzioni necessarie a systemd a creare un socket unix e ad avviare un processo di gunicorn in risposta al traffico in entrata.

- `gunicorn.socket`: contiene le informazioni che descrivono il socket e la sua location

```
[Unit]
Description=gunicorn socket
[Socket]
ListenStream=/run/gunicorn.sock
[Install]
WantedBy=sockets.target
```

- `gunicorn.service`: contiene
  - una sezione con i metadata e le dipendenze, tra i quali quella di gunicorn.socket;
  - una sezione che specifica l'utente e il gruppo sotto il quale deve girare il processo, la cartella di lavoro, il path assoluto dell'eseguibile all'interno dell'ambiente virtuale e di particolare rilevanza il numero di processi paralleli, chiamati worker;
  - una sezione che indica a cosa deve essere puntare il servizio quando viene avviato.

```
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target
[Service]
User=jetson
Group=www-data
WorkingDirectory=/home/jetson/thesisproject
ExecStart=/home/jetson/thesisproject/app_env/bin/gunicorn \
          --access-logfile - \
          --workers 3 \
         --timeout 600 \
          --bind unix:/run/gunicorn.sock \
          remotecommunication.wsgi:application
[Install]
WantedBy=multi-user.target
```

Per abilitare il socket si utilizzano i seguenti comandi:

```
sudo systemctl start gunicorn.socket
sudo systemctl enable gunicorn.socket
```

`gunicorn.service` verrà avviato in automatico non appena una connessione viene effettuata. Per controllare che il servizio sia attivo si utilizza il comando

```
sudo systemctl status gunicorn
```

5.1.5 Nginx

Infine è stato necessario installare e configurare Nginx, utilizzato per passare traffico al processo agendo da reverse proxy server.

```
sudo apt install nginx
```

Dopo aver installato il server, si è passati alla configurazione. Il file creato allo scopo è chiamato `remotecommunication` ed inserito all'interno della cartella `/etc/nginx/sites-available/`.

La scelta implementativa discussa nel capitolo precedenteè stata quella di comunicare utilizzando HTTPS attraverso una connessione criptata grazie al protocollo crittografico TLS.

In questo file sono stati specificati i server block, ovvero dei blocchi che incapsulano i dettagli di configurazione di ciascun server.

Prima di questi si aggiunge però l'opzione `server_tokens off;`, che permette di disabilitare l'informazione sulla versione di Nginx usata, molto utile in caso di utilizzo di versioni vecchie e vulnerabili. In questo caso è stata utilizzata l'ultima versione del server, ma si tratta di un ostacolo in più che rallenta un potenziale attaccante.

Il primo blocco ascolta sulla porta 80, HTTP, e se riceve del traffico la reindirizza automaticamente sulla porta 443, HTTPS. Questo rende obbligatoria la creazione di un canale criptato per la comunicazione, che altrimenti non viene iniziata.

```
server {
    listen 80;
    listen [::]:80;
    server_name 192.168.1.235;
    return 301 https://$server_name$request_uri;
}
```

Il secondo blocco è quello che il server utilizza per captare e gestire la comunicazione di un canale sicuro sulla porta 443.

```
server {
   listen 443 ssl;
   listen [::]:443 ssl;
```

Avendo optato per l'utilizzo del protocollo TLS con autenticazione bilaterale, bisogna specificare sia il certificato del server, che il certificato della CA competente. Questi sono trattati nello specifico nella sezione 5.1.6. La verifica del client è configurata come opzionale. Se il client non possiede un certificato valido, allora gli viene mostrato l'errore 403.

```
#SERVER AUTHENTICATION
ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
#CA per CLIENT AUTHENTICATION
ssl_client_certificate /etc/ssl/certs/ca.crt;
ssl_verify_client optional;
```

Si limita la funzionalità a TLSv1.3, la versione più sicura e veloce del protocollo. Anche se non ancora completamente compatibile con tutti i browser, questa versione è supportata da quelli più diffusi. Inoltre è previsto il raggiungimento della piena compatibilità nei prossimi anni. Le ciphersuite sono state selezionate tra quelle definite in RFC8446 [40] e nelle linee guida nel NIST [42]. Si aggiunge anche l'opzione di preferenza dei cifrari del server su quelli del client.

```
ssl_protocols TLSv1.3;
ssl_prefer_server_ciphers on;
ssl_ciphers TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:
          TLS_AES_256_GCM_SHA384;
```

Si fornisce la sorgente dei parmetri Diffie-Hellman e le curve per l'algoritmo di key-agreement ECDH, Diffie-Hellman a curva ellittica. Le curve specificate sono quelle raccomandate dal NIST [42]

```
ssl_dhparam /etc/nginx/dhparam.pem;
ssl_ecdh_curve secp384r1:prime256v1;
```

Viene specificato un periodo di tempo entro il quale l'utilizzatore può utilizzare la stessa sessione, con i parametri salvati in cache per lo stesso periodo di tempo. Si disabilita la ripresa della sessione, poiché non necessaria per le performance, ma possibile fonte di vulnerabilità.

```
ssl_session_timeout 5m;
ssl_session_cache shared:SSL:5m;
ssl_session_tickets off;
ssl_stapling on;
ssl_stapling_verify on;
```

```
        resolver 8.8.8.8 8.8.4.4 valid=300s;
        resolver_timeout 5s;
```

Si aggiungono anche vari header, approfonditi nella sezione 4.3.5

```
        add_header Strict-Transport-Security "max-age=63072000;
            includeSubDomains; preload";
        add_header X-Content-Type-Options nosniff;
        add_header Content-Security-Policy "default-src frame_ancestor
            'self';" always;

        server_name 192.168.1.235;
        location = /favicon.ico { access_log off; log_not_found off; }
        location /static/ {
         root /home/jetson/thesisproject;
        }
        location / {
        if ($ssl_client_verify != SUCCESS) {
         return 403;}
         include proxy_params;
         proxy_pass http://unix:/run/gunicorn.sock;
        }
    }
```

5.1.6 Autenticazione

Autenticazione del server

Per scopi implementativi è stato creato ed utilizzato un certificato di chiave asimmetrica self-signed per il server, rilevato come insicuro da qualunque browser. Questo certificato, in fase di produzione, deve obbligatoriamente essere sostituito da un certificato prodotto da un ente certificatore. Infatti utilizzando quello creato in questo paragrafo, un normale browser può aprire la connessione HTTPS, ma avvisa l'utente che il certificato utilizzato dal server non è affidabile.

Per crearlo è stato utilizzato il software OpenSSL, un kit di strumenti utili per scopi crittografici e di comunicazione sicura [49]. Per creare il certificato server-side è stato necessario generare una chiave privata di 2048 bit per il protocollo di crittografia asimmetrica RSA.

```
sudo openssl genrsa -out /etc/ssl/private/nginx-selfsigned.key 2048
```

In seguito è stato generato il certificato X.509, standard per chiavi e certificati per TLS. La durata è stata settata ad un anno, mentre come chiave è stata utilizzata quella precedentemente creata.

```
sudo openssl req -x509 -nodes -days 365 -key
    /etc/ssl/private/nginx-selfsigned.key -out
    /etc/ssl/certs/nginx-selfsigned.crt
```

È buona pratica utilizzare all'interno del certificato il dominio collegato al web server, poiché fisso, al contrario dell'indirizzo IP che può variare nel tempo, ma, per il solo lavoro di questa tesi, è stato utilizzato l'indirizzo IP del sever, mantenuto fisso per tutta la durata del progetto.

Autenticazione del client

Per garantire l'autenticazione del client, dei certificati devono essere generati. Per fare questo è stato necessario generare una CA, certificate authority, che potesse firmare, con il suo certificato, le richieste dei clienti.

```
sudo openssl genrsa -aes256 -out /etc/ssl/private/ca.key 2048
```

Per questo scopo è stata creata una chiave privata di 2048 bit per RSA. In questo caso la chiave è stata criptata con l'algoritmo crittografico AES, con chiave a 256 bit. Questa ulteriore opzione fa si che venga richiesta una passphrase alla creazione della chiave, che sarà necessaria ogni qual volta si voglia utilizzare il certificato creato in seguito.

```
sudo openssl req -new -x509 -days 365 -key /etc/ssl/private/ca.key -out
    /etc/ssl/certs/ca.crt
```

Creato il certificato per la CA è possibile proseguire con la creazione del CSR, il certificate signing request. Questo passaggio, in fase di produzione, deve essere realizzato dall'utente, al contrario del precedente che deve essere eseguito dall'amministratore.

```
sudo openssl genrsa -aes256 -out /etc/ssl/private/client.key 2048
sudo openssl req -new -key /etc/ssl/private/client.key -out client.csr
```

La richiesta deve poi essere inoltrata, approvata e firmata dall'autorità competente, in questo caso l'amministratore.

```
sudo openssl x509 -req -days 365 -in client.csr -CA /etc/ssl/certs/ca.crt
    -CAkey /etc/ssl/private/ca.key -set_serial 01 -out
    /etc/ssl/certs/client.crt
```

Il seriale permette di rinnovare il certificato una volta scaduto, semplicemente incrementandolo, senza bisogno di crearne uno nuovo.

L'utente, ricevuto il certificato dovrà crearne l'archivio .pfx, ovvero il bundle della chiave e del certificato X.509 secondo il formato `PKCS#12`

```
openssl pkcs12 -export -out client.pfx -inkey /etc/ssl/private/client.key
    -in /etc/ssl/certs/client.crt -certfile /etc/ssl/certs/ca.crt
```

Il file in formato .pfx è facilmente installabile all'interno del browser.

Diffie-Hellman group

Un'ulteriore misura di sicurezza è stata configurare il server per fare in modo che, lo scambio di chiavi Diffie-Hellman abbia dei parametri forti.

Si tratta di un protocollo crittografico che consente a due parti di stabilire un segreto condiviso, ovvero una chiave, scambiandosi messaggi attraverso una rete insicura. Questa chiave viene poi utilizzata per rendere confidenziali le comunicazioni successive utilizzando la crittografia simmetrica.

Il vantaggio dal punto di vista della sicurezza è che le due parti possono stabilire la chiave condivisa senza scambiarsi direttamente il segreto. Ciò significa che anche se un terzo soggetto intercetta la comunicazione tra le due parti, non è in grado di ottenere la chiave condivisa.

Tuttavia, è importante notare che lo scambio di chiavi Diffie-Hellman non è autenticato, ma è alla base di numerosi protocolli crittografici che performano l'autenticazione, come il protocollo utilizzato in questo caso, TLS.

A questo scopo è stato utilizzato il seguente comando per far si che vengano utilizzati dei parametri robusti per DH. Viene utilizzata una lunghezza di chiave 2048, attualmente considerata robusta. Lunghezze inferiori sono considerate insicure.

```
sudo openssl dhparam -out /etc/nginx/dhparam.pem 2048
```

5.2 Applicazione

5.2.1 Integrazione del software

I moduli del software, descritti nella sezione 2.4.2 sono stati inseriti all'interno del progetto Django con minime modifiche. Mentre i moduli Follow Me, Safety e Tracking sono stati incorportati senza variazioni, il modulo di Object Detection è stato adattato per l'uso interno al progetto.

È stata realizzata una classe, `VideoCamera`, che implementa le cinque funzionalità del modulo sopracitato.

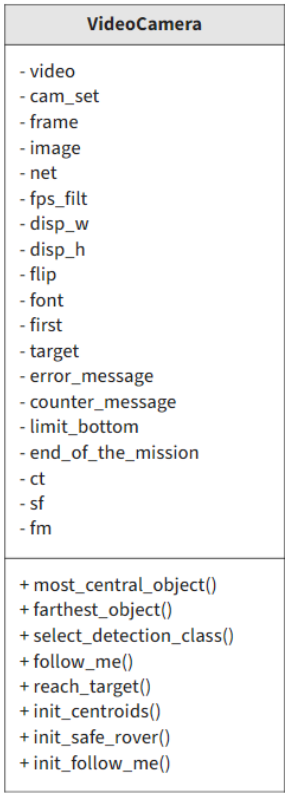

Figura 5.4. Object Detection Module

I moduli si interfacciano tra loro come mostrato in figura 2.5.

Live Streaming della fotocamera

Per fare in modo che il video ripreso dalla fotocamera possa essere visionato dall'utente, è stato usato `StreamingHttpResponse`, un tipo di HTTP response che permette di inviare il corpo della risposta del server in molteplici pezzi. Questo ha permesso di creare un sistema in grado di inviare il video, prodotto da OpenCV in tempo reale, al client, che potrà visualizzarlo come mostrato in figura 5.7.

Lo svantaggio è che non si tratta di un protocollo pensato appositamente per lo streaming di video. Infatti, poiché i frame fanno parte tutti di un'unica risposta HTTP, il processo worker di gunicorn rimane occupato fino alla fine dello streaming. Il motivo per cui, in questo caso, è possibile adottare questa soluzione, è che solo un utente alla volta può accedere al video e scegliere la modalità con cui lavorare. Si deve però aumentare il numero di processi worker paralleli all'interno di `gunicorn.socket`, per permettere che il server riceva altre richieste, come quella di stop.

Per minimizzare eventuali jitter dovuti a latenze del network, è stato limitato il buffer di OpenCV aggiungendo `self.video.set(cv2.CAP_PROP_BUFFERSIZE, 2)` durante l'inizializzazione della videocamera. In questo modo, si ottiene sempre l'ultimo frame, rendendo la trasmissione più veloce e fluida.

5.2.2 Interfaccia Utente

L'interfaccia grafica si compone di tre schermate principali.

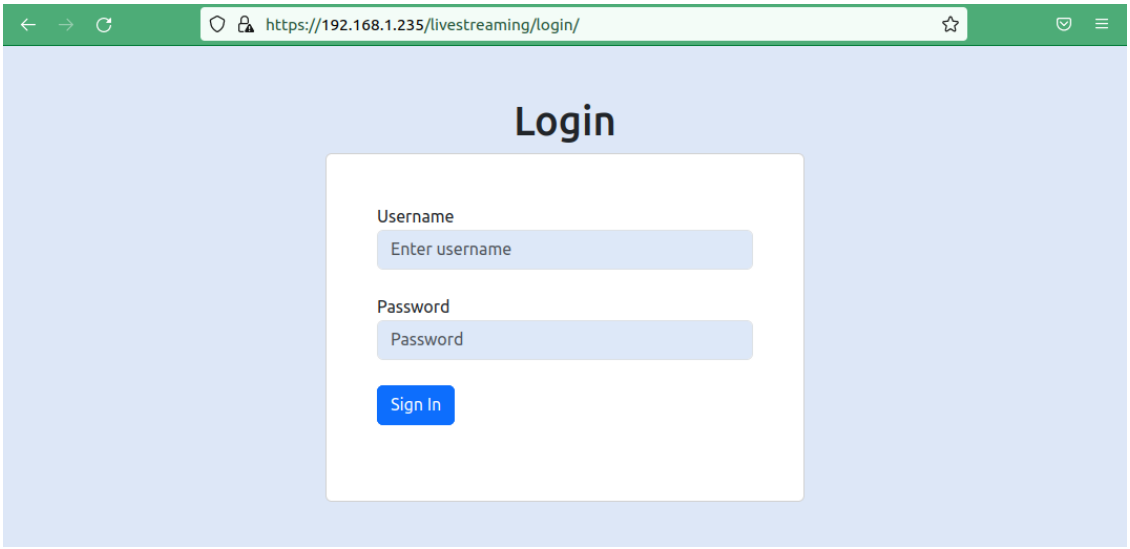

Figura 5.5. Login interface

La prima schermata per l'utente è quella di login. L'URL corrispondente è `https://<dominio o serverIP>/livestreaming/login/` Le credenziali sono fornite dall'amministratore di sistema, sulla base della policy definita in fase di produzione, come spiegato nel paragrafo 5.1.3.Le credenziali vengono inviate al server tramite POST request e sono validate tramite regex.

Se il processo di login va a buon fine, l'utente viene reindirizzato alla homepage, `https://<dominio o serverIP>/livestreaming/`, da dove può controllare il rover, definendone la modalità.

In questa pagina l'utente è in grado di scegliere la modalità, attivando la videocamera e il live streaming. Alcune modalità, 'target selection' e 'reach target', richiedono che venga inserito del testo. I due valori sono inviati al server tramite POST request e validati al suo interno.

Quando una modalità viene attivata, viene inviata una POST request contente un valore dipendente dalla scelta fatta. In base a questo valore, verrà mostrato un video che avrà come sorgente un URL diverso. Quando OpenCV attiva la fotocamera montata sulla jetson nano, viene mostrata l'immagine ripresa. In questa schermata è possibile terminare l'attività della videocamera tramite il pulsante apposito. Questa operazione reindirizzerà l'utente all'interfaccia di selezione della modalità.

In entrambe le precedenti schermate è possibile notare il pulsante di logout all'interno della navigation bar. Questo permette all'utente di terminare la propria attività e la propria sessione di login. L'utente viene reindirizzato all'interfaccia di login.

Qualora un utente non loggato tentasse di accedere tramite url ad una delle precedenti schermate, verrebbe immediatamente reindirizzato alla pagina di login.

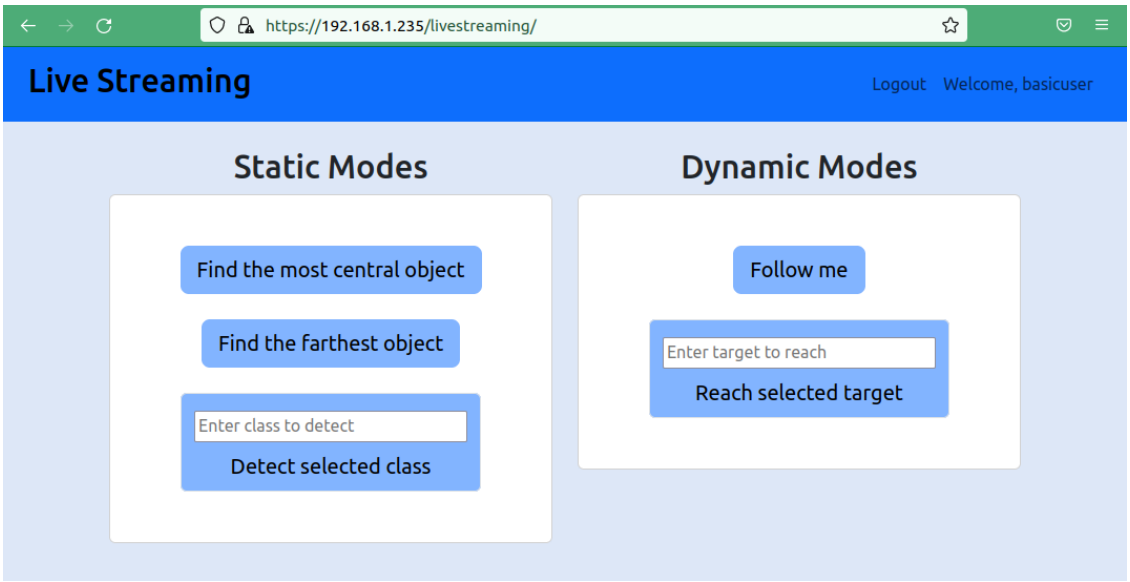

Figura 5.6. Homepage

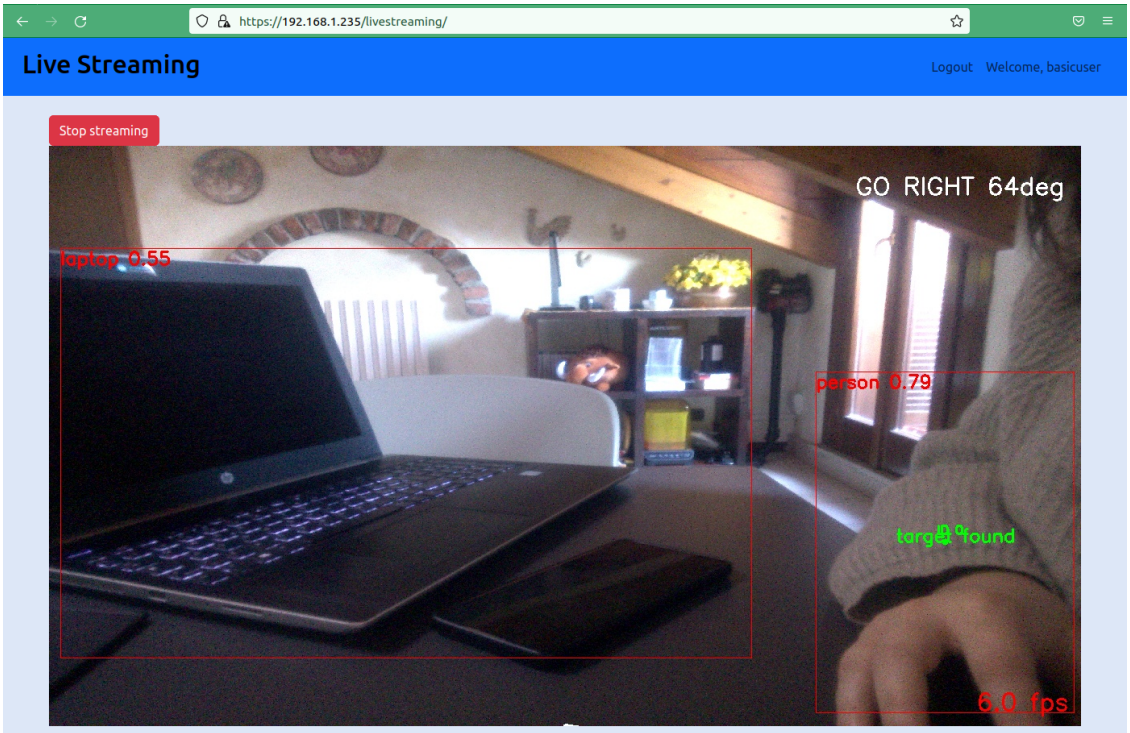

Figura 5.7. Streaming interface

Capitolo 6

Testing

Lo sviluppo del sistema di comunicazione è stato completato con una verifica dell'efficacia delle misure di sicurezza implementate. Sono di seguito elencati la metodologia utilizzata e i test realizzati, con i risultati ottenuti da essi.

6.1 Introduzione

Misurare la sicurezza di un sistema non è semplice. È una questione complessa, che coinvolge livelli di protezione multipli e minacce in continua evoluzione. Per questo bisogna valutare che le misure adottate siano efficaci nel rendere i sistemi sicuri, protetti da accessi non autorizzati, violazioni dei dati e altre minacce.

Anche se la sicurezza è stata presa in considerazione fin dall'inizio dello sviluppo, realizzare dei test in questo momento permette di verificare la correttezza e l'efficacia dell'analisi e dell'implementazione delle misure precedentemente discusse. In aggiunta, è possibile individuare vulnerabilità nascoste, permettendo di valutarne il rischio e prendere precauzioni adatte. Si tratta quindi di realizzare dei test di penetrazione, ovvero simulare il comportamento di un hacker in cerca di debolezze del sistema attraverso cui montare un attacco.

Esistono vari standard e linee guida utili per condurre questo tipo di simulazione su un'applicazione web.

- OWASP Web Security Testing Guide [50];
- Penetration Testing Execution Standard (PTES) [51];
- Technical Guide to Information Security Testing and Assessment (NIST 800-115) [52];
- Open Source Security Testing Methodology Manual (OSSTMM) [53].

Tutti questi documenti forniscono linee guida e suggerimenti per portare a termine efficacemente un penetration test.

Si possono evidenziare delle fasi attraverso cui si deve sviluppare il lavoro. Dopo aver definito i requisiti, i limiti entro cui rimanere, i potenziali rischi e gli strumenti da utilizzare, si procede alla raccolta di quante più informazioni possibili sul sistema in esame, in modo da identificare e validare eventuali vulnerabilità del sistema. Queste ultime vengono poi sfruttate per cercare di ottenere l'accesso al sistema. Infine bisogna raccogliere e definire i risultati, individuando soluzioni da adottare.

6.2 Descrizione del test

Prima di iniziare con il test vero e proprio, è importante definirne in modo chiaro e preciso le caratteristiche e i requisiti.

Obiettivi

L'obiettivo del test è quello di verificare il rispetto dei requisiti e l'efficacia delle misure di sicurezza prese in esame nei capitoli precedenti, valutandone anche la corretta configurazione. Inoltre con questa pratica, si vuole cercare vulnerabilità residue e gli exploit che potrebbero essere messi in atto grazie a loro. Bisogna verificare che l'attaccante non sia in grado di autenticarsi nell'applicazione. Nel qual caso il sistema si deve ritenere compromesso.

Limiti

Il sistema da esaminare comprende l'applicazione e il server web sviluppati precedentemente. Il test deve essere realizzato in un ambiente isolato, poiché il sistema non è ancora provvisto di IP pubblico o di un dominio associato raggiungibile attraverso internet, ma che rispecchi una rete non sicura.

Metodologia

È importante definire la metodologia di test che sarà utilizzata, ovvero il livello di conoscenza pregressa del sistema e l'area di applicazione.

Esistono diversi approcci sulla base del livello di informazioni di partenza.

- black-box, in cui non si ha accesso ad alcuna informazione pregressa;
- grey-box, in cui si ha una conoscenza basilare del sistema;
- white-box, in cui si ha una conoscenza approfondita del sistema.

Poiché in questo caso il tester e lo sviluppatore coincidono, la scelta di un test di tipo white-box è obbligata. In più è possibile definire due tipi di test sulla base della area da esaminare:

- testing esterno;
- testing interno.

Gli external penetration test hanno l'obiettivo di identificare le vulnerabilità della rete, del sito web o delle applicazioni web esposte su internet, che potrebbero essere utilizzate da un hacker per entrare nella rete dell'azienda. Il penetration tester simula gli attacchi di un hacker esterno, cercando di scoprire e sfruttare qualsiasi vulnerabilità scoperta. Questi test possono includere la scansione delle porte, la ricerca di vulnerabilità note e la tentata penetrazione tramite backdoor, phishing, social engineering e altre tecniche di attacco. La valutazione dell'infrastruttura DNS, dei siti e delle applicazioni web è particolarmente importante in questi test, in quanto rappresentano le porte d'ingresso principali per gli attacchi esterni. Il DNS può essere soggetto a attacchi di spoofing e DNS hijacking, mentre il sito e le applicazioni web possono essere vulnerabili ad attacchi di injection, cross-site scripting, vulnerabilità nella gestione degli account utente e altre minacce.

Gli internal penetration test sono volti a valutare la sicurezza delle reti, dei sistemi e delle applicazioni interne dell'organizzazione, simulando gli attacchi di un utente malintenzionato che abbia già ottenuto l'accesso al sistema. Si testa questa casistica perché un malintenzionato, potrebbe cercare di sfruttare falle interne per accedere a dati riservati o compromettere la sicurezza dell'organizzazione.

I risultati dei test di penetrazione interni vengono utilizzati per identificare le vulnerabilità e per migliorare la sicurezza dei sistemi e delle applicazioni interne dell'organizzazione, al fine di prevenire gli attacchi informatici interni e di garantire la sicurezza dei dati sensibili dell'azienda.

Il tipo di test che deve essere eseguito è di tipo esterno, ovvero l'attaccante non deve essere in grado di guadagnare l'accesso ai dispositivi. Poiché il sistema non è ancora accessibile da internet, non è possibile verificare come un attaccante potrebbe agire sulla base delle informazioni riscontrabili online, ma bisogna fare delle ipotesi. Infatti bisogna presupporre che l'attaccante sia in possesso del dominio e dell'indirizzo IP del rover.

Strumenti

Come strumento verrà utilizzata una macchina con sistema operativo Kali Linux, una distribuzione Linux open-source, basata su Debian, potente e versatile, che fornisce un'ampia gamma di strumenti e utilità che possono essere utilizzati per identificare e sfruttare le vulnerabilità, condurre ricerche sulla sicurezza e svolgere altre attività legate ad essa. In particolare verrà utilizzata la versione Live 2022.4, in Forensic Mode. Gli strumenti utilizzati per i seguenti test sono:

- Nmap: popolare strumento di esplorazione della rete e di verifica della sicurezza che consente agli utenti di scoprire host e servizi su una rete di computer e di identificare potenziali vulnerabilità e problemi di sicurezza;
- Nikto: scanner open-source per server web in grado di eseguire test completi per oltre 6700 file o programmi potenzialmente pericolosi sui server web, nonché di identificare altri problemi di sicurezza come software obsoleti, configurazioni insicure e vulnerabilità comuni come XSS, SQL Injection e altro;
- Masscan: scanner di porte ad alta velocità in grado di analizzare rapidamente ampi intervalli di indirizzi IP e di identificare le porte aperte sui sistemi di destinazione;
- Sslyze: strumento utilizzato per analizzare le connessioni SSL/TLS e identificare potenziali punti deboli e configurazioni errate. È in grado di rilevare potenziali vulnerabilità nei protocolli, cifrari deboli e problemi di certificati;
- Sslscan: è uno strumento a riga di comando che consente agli utenti di identificare problemi di certificati, cifrari deboli e potenziali vulnerabilità nei protocolli SSL/TLS.

6.3 Raccolta di informazioni

La raccolta di informazioni è un primo passo fondamentale nei test di penetrazione, infatti corrisponde a ciò che fa un attaccante prima di costruire un attacco. L'obiettivo di questa fase è comprendere l'ambiente da testare, identificare le potenziali vulnerabilità e sviluppare un piano per condurre il resto del test. Esistono diverse tecniche e strumenti che possono essere utilizzati per la raccolta di informazioni.

L'open-source intelligence consiste nel raccogliere informazioni da fonti pubblicamente disponibili, come i social media, i forum e i motori di ricerca. Anche se utile nel contesto di un'applicazione già raggiungibile attraverso internet, questo tipo di raccolta non verrà effettuata. Al contrario verrà utilizzata come data l'informazione sull'indirizzo IP. Allo stesso modo non verranno utilizzate tecniche di social engineering. Inoltre, è necessario tenere in considerazione che la pagina non è raggiungibile tramite un URL standard `/`. Anche se è più difficile per un attaccante conoscere l'indirizzo corretto, questo potrebbe comunque apparire nelle ricerche. Per questo bisogna assicurarsi di testarne la sicurezza allo stesso modo.

È possibile effettuare diverse indagini. La scansione di rete consiste nell'inviare richieste a una serie di indirizzi IP su una rete per determinare quali sono attivi, quali servizi sono in esecuzione su ciascun host, nonché le informazioni sul software e sulla versione di tali servizi. Per eseguire questo tipo di scansione esistono degli strumenti automatici molto validi, in grado di identificare diversi tipi di informazioni, come porte aperte, servizi in esecuzione e sistemi operativi. Allo stesso modo esistono strumenti in grado di scansionare gli host ed identificare potenziali vulnerabilità nei servizi e nelle applicazioni in esecuzione su una rete. Sono stati usati degli strumenti in grado di analizzare la rete e le risposte degli host presenti. Nell'appendice C si possono trovare i risultati completi delle scansioni effettuate.

Come primo passo si è cercato di identificare il sistema operativo e il tipo di server, così come i servizi disponibili.

Per fare questo esistono strumenti come Nmap, che grazie all'indirizzo IP della macchina è in grado di eseguire vari tipi di scansione. È possibile anche effettuare un'analisi manuale intercettando gli header delle risposte del server e le pagine che vengono visualizzate.

Queste informazioni sono utili durante la raccolta di informazioni perché possono aiutare nell'identificazione delle vulnerabilità note e delle potenziali falle di sicurezza in quel particolare ambiente. Ad esempio, se si scopre che il server utilizza una versione obsoleta di un software, un attaccante potrebbe essere in grado di trovare exploit noti per quella versione. Queste informazioni possono essere utilizzate per indirizzare i propri attacchi. Ad esempio, se il server utilizza Apache come web server, l'attaccante potrebbe concentrare i propri sforzi sulla ricerca di vulnerabilità specifiche per Apache, così come in questo caso potrebbe fare per Nginx. Anche conoscere la versione specifica del server può aiutare nell'identificazione di vulnerabilità, in particolare se si dovesse trattare di versioni obsolete.

Nmap e Masscan sono stati utilizzati per identificare le porte e i servizi disponibili.

Conoscere questa informazione può aiutare ad identificare quali servizi sono esposti alla rete e do conseguenza, quali sono potenzialmente vulnerabili ad attacchi. Ad esempio, se si scopre che il server dispone di un servizio SSH attivo, un malintenzionato potrebbe concentrare i propri sforzi sulla ricerca di vulnerabilità specifiche per questo protocollo. Conoscere i servizi e le porte aperte può anche aiutare a identificare l'eventuale presenza di non necessari o non utilizzati, che potrebbero essere potenzialmente pericolosi. Ad esempio, se il sistema dispone di una porta aperta ma non utilizzata, bisognerebbe procedere a chiudere quella porta per evitare potenziali attacchi. Inoltre, se per esempio si scopre che il sistema utilizza una versione obsoleta di un servizio, un attaccante potrebbe essere in grado di trovare exploit noti per quella versione, così come nel caso di servizi notoriamente meno sicuri, come Telnet.

Sulla base delle informazioni reperite è possibile effettuare ulteriori analisi, prendendo in esame le specifiche porte.

Come secondo passo è stato quindi utilizzato uno scanner di vulnerabilità per cercare di identificare quelle presenti nei servizi e nelle applicazioni in esecuzione sulla rete utilizzando tecniche che possono includere la scansione delle porte, l'analisi dei pacchetti, la ricerca di exploit noti e la valutazione della conformità alle normative di sicurezza.

Una volta individuate le vulnerabilità, gli scanner possono fornirne un report dettagliato. Questo report può aiutare a comprendere i rischi a cui il sistema è esposto. Tuttavia si tratta di strumenti non infallibili e che possono non possono individuare tutte le vulnerabilità presenti su un sistema. Inoltre è possibile che forniscano anche dei falsi positivi, ovvero vulnerabilità di elementi che in realtà non ne hanno.

In particolare sono stati utilizzati gli strumenti Nikto, SSLscan e SSLyze.

Nikto è uno scanner per server Web in grado di identificare potenziali vulnerabilità e configurazioni errate nei server Web e nelle applicazioni Web. È in grado di rilevare software obsoleto, password deboli e altri problemi di sicurezza comuni. Tuttavia, non è progettato specificamente per i test su TLS, per cui sono stati utilizzati altri strumenti.

Poiché viene utilizzato il protocollo TLS, è necessario verificare che la configurazione sia stata fatta correttamente e che, pur ricavando determinate informazioni, un possibile attaccante non trovi algoritmi deboli o spazio per attacchi mirati.

SSLScan, come SSLyze, è uno strumento progettato specificamente per questo tipo di test. È in grado di testare rapidamente le configurazioni del protocollo di sicurezza su un server web tentando di connettersi al server utilizzando vari protocolli SSL/TLS e suite di cifratura. È stato progettato per fornire una panoramica semplice e veloce dei problemi di configurazione.

SSLyze è uno strumento che fornisce un'analisi dettagliata delle configurazioni SSL/TLS, verificando la presenza di vulnerabilità come Heartbleed, oltre a configurazioni errate e problemi con le catene di fiducia dei certificati. Può essere eseguito in modalità riga di comando o come interfaccia grafica ed è progettato per fornire un'analisi completa della sicurezza di TLS.

6.4 Valutazione delle vulnerabilità

Le informazioni ricavate con i test elencati precedentemente possono essere ricavate da un attaccante, che quindi sarà in grado di utilizzarle per costruire degli attacchi mirati. Se vengono identificate delle potenziali vulnerabilità, allora è possibile determinarne un percorso di attacco.

Versione del server

Grazie ad una scansione di Nmap e all'analisi degli header dei pacchetti inviati dal server è possibile individuare alcune importanti informazioni.

Come indicato dalla scansione C.1, un attaccante è in grado di avere l'informazione che il server lavora su una macchina general purpose, prodotta da Nvidia, con sistema operativo Linux. L'architettura della macchina e il sistema operativo utilizzato sono generalmente considerati informazioni pubbliche e non riservate, quindi potrebbero essere facilmente dedotti da fonti come il sito web dell'azienda di produzione o altre fonti pubbliche di informazione. Tuttavia, per scopi di sicurezza, è sempre consigliabile limitare l'accesso alle informazioni sul sistema e utilizzare procedure di sicurezza adeguati per proteggere la macchina e le informazioni contenute al suo interno.

Anche se attualmente non si sta utilizzando una versione del web server obsoleta, è una buona pratica nascondere eventuali informazioni in previsione di un utilizzo futuro e rendere più difficile il compito di un attaccante.

Per verificare che il server non riveli alcuna informazione sulla versione di Nginx, ovvero che l'impostazione `server_tokens off` lavori correttamente, si possono individuare informazioni utili nella scansione di Nmap e negli header di risposta dei server.

Nmap permette di rilevare solo l'utilizzo di Nginx, ma non la sua versione. Anche nelle risposte del server, investigando le informazioni dell'header, questa è l'unica informazione che viene divulgata. Lo stesso comportamento si trova nelle pagine di default ritornate dal server per codici come 403.

Figura 6.1. Pagina di default visualizzata per codice di errore 403

Se si volesse nascondere anche questa informazione, si potrebbero modificare gli header e introdurre l'utilizzo di pagine di errore personalizzate.

Porte e servizi

Sfruttando ulteriormente le capacità di Nmap, è stato possibile scansionare il sistema in cerca di porte aperte.

Le porte possono essere aperte, chiuse o filtrate. Le porte aperte sono quelle che consentono il traffico di dati in entrata e/o in uscita senza alcun tipo di restrizione. Questo significa che qualsiasi connessione che tenta di accedere a quella porta sarà accettata dal sistema. Le porte chiuse, al contrario, sono accessibili, ma non hanno alcuna applicazione in ascolto su di essa. Possono essere utili per mostrare che un host è disponibile su un indirizzo IP e nel rilevamento del sistema operativo. Bloccando queste porte con un firewall, apparirebbero nello stato filtrato. Le porte filtrate sono quelle che non inviano alcuna risposta quando viene inviato un pacchetto. Questo di solito significa che il pacchetto di richiesta è stato filtrato o bloccato dal firewall, come nel caso in esame.

In questo ci si aspetta che siano aperte solo le due porte 22, utilizzata per lo sviluppo, e la porta 443, su cui il sistema comunica.

Le scansioni di Nmap e Masscan hanno infatti rilevato solo 2 porte aperte:

- 22/tcp con servizio ssh e versione OpenSSH 7.6

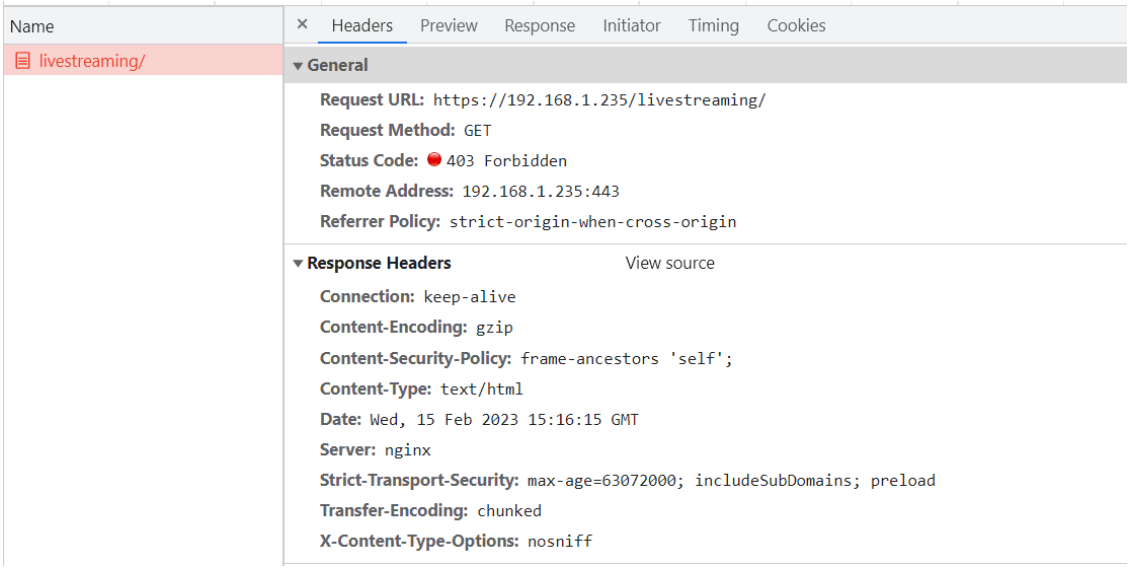

Figura 6.2. Esempio di intestazione di una risposta con codice 403

- 443/tcp con servizio ssl/http e nginx

Le altre 65333 sono risultate filtrate. I risultati ottenuti hanno confermato l'efficacia delle misure prese in esame nei capitoli precedenti. Infatti gli unici servizi disponibili e non bloccati dal firewall UFW sono quelli sulle porte esplicitamente abilitate. Inoltre è stato verificato che tutte le altre porte non restituiscono alcun risultato ad un attaccante.

La porta 22 dovrà essere esplicitamente bloccata in fase di produzione, mentre la porta 443, usata per la comunicazione, necessita di essere testata.

Vulnerabilità del web server

È stato utilizzato il software Nikto per scansionare il server, in ascolto sulla porta 443, in cerca di vulnerabilità.

```
GET The anti-clickjacking X-Frame-Options header is not present.
```

Questa intestazione è stata sostituita dalla direttiva frame-ancestors all'interno del CSP, come analizzato nella sezione 4.3.5. Un attaccante potrebbe comunque cercare di sfruttare questa opzione. Per valutare che la configurazione sia in grado di proteggersi da attacchi di tipo clickjacking questa verrà testata in seguito.

```
GET The X-XSS-Protection header is not defined. This header can hint to
    the user agent to protect against some forms of XSS
```

Questa è una vulnerabilità che un attaccante potrebbe cercare di sfruttare. Avendo una conoscenza del sistema approfondita, invece, si è a conoscenza del fatto che questa intestazione non è più presente perché sostituita dalla configurazione del CSP. Anche in questo caso, però, verrà testata la protezione contro questo tipo di attacco.

```
GET The site uses SSL and Expect-CT header is not present.
```

Lo scopo dell'intestazione Expect-CT era quello di consentire ai siti web di applicare volontariamente la Certificate trasparency (CT) prima che diventasse obbligatoria. Tuttavia, a partire dal 2018, i principali browser web hanno iniziato a richiedere il CT per tutti i nuovi certificati SSL/-TLS per impostazione predefinita. Ciò significa che i nuovi certificati SSL/TLS devono contenere i Signed Certificate Timestamp (SCT) per poter essere considerati affidabili dai browser web. A seguito di questa modifica, l'uso dell'intestazione Expect-CT 'e stato deprecato, quindi è corretto che questa intestazione non sia presente.

```
GET The Content-Encoding header is set to "deflate" this may mean that
    the server is vulnerable to the BREACH attack.
```

L'intestazione Content-Encoding specifica la codifica applicata al corpo della risposta. Viene utilizzata per comprimere o modificare il contenuto inviato dal server al client. Il valore 'deflate' nell'intestazione Content-Encoding indica che il server utilizza l'algoritmo DEFLATE per comprimere il corpo della risposta.

L'attacco BREACH, Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext, è una vulnerabilità di sicurezza che può essere sfruttata quando un sito web utilizza la compressione HTTP. L'attacco consente a un aggressore di estrarre informazioni sensibili dalla risposta compressa inviata dal server. La vulnerabilità è legata al fatto che gli algoritmi di compressione HTTP funzionano trovando schemi ripetuti nel corpo della risposta e gli aggressori possono sfruttare questa proprietà per dedurre informazioni sui dati segreti della risposta. Questo tipo di attacco può essere mitigato disabilitando la compressione HTTP o randomizzando il corpo della risposta per rendere più difficile per gli aggressori trovare gli schemi.

È stata verificata la corretta configurazione delle altre intestazioni analizzando le risposte ottenute dal sever.

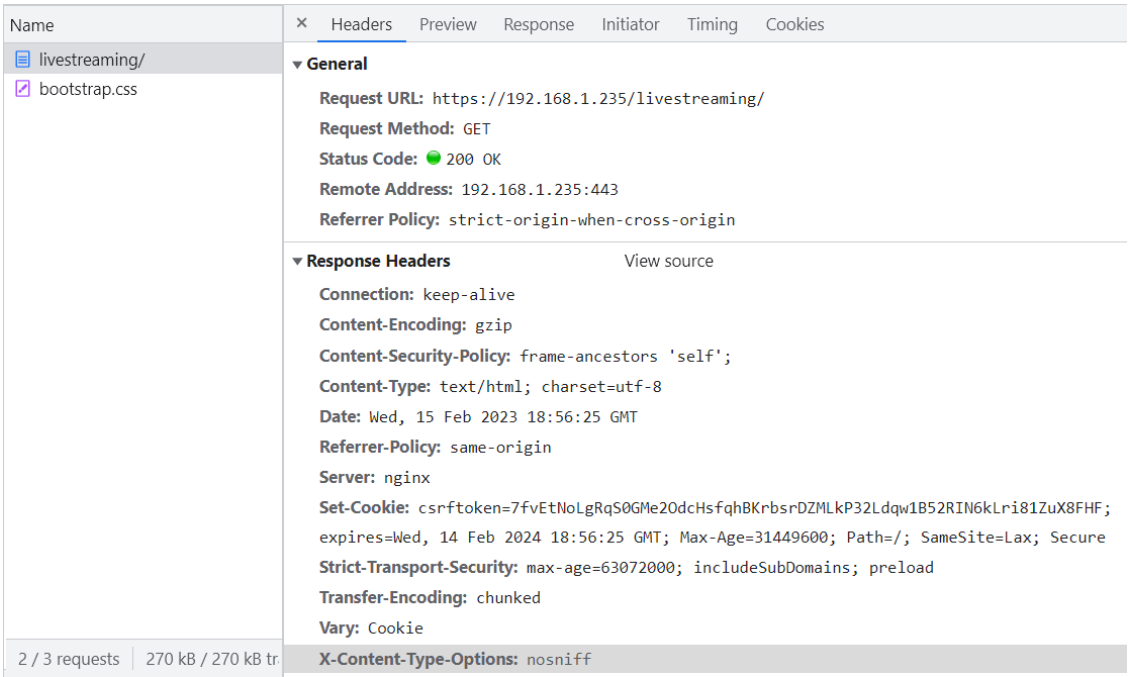

Figura 6.3. Intestazione della risposta del server

Configurazione del protocollo di sicurezza

Grazie alle scansioni di SSLyze e SSLscan, si è potuto verificare che le impostazioni discusse nei capitoli precedenti sono state configurate in modo corretto.

In particolare si è potuto verificare che l'unica versione del protocollo supportata è TLSv1.3, con ciphersuite attualmente ritenute sicure. Questo indica che il sistema è protetto da attacchi di tipo downgrade.

Un attacco di questo tipo consiste nel forzare un sistema o un canale di comunicazione a utilizzare una versione più vecchia o meno sicura di un protocollo, invece della versione più recente e sicura, con l'obiettivo di sfruttare le vulnerabilità dei protocolli più vecchi. Ad esempio, un hacker può intercettare una comunicazione tra due sistemi che supportano la versione più recente di un protocollo, come TLS 1.3, e cercare di forzarli a utilizzare una versione più vecchia,

come TLS 1.1 e sfruttare le debolezze della versione. In questo caso l'attacco non può essere realizzato perché la configurazione rifiuta le versioni più vecchie e meno sicure.

Inoltre è presente il TLS Fallback SCSV (Signaling Cipher Suite Value), un meccanismo utilizzato nel protocollo TLS per impedire agli aggressori di forzare un livello di crittografia inferiore a quello normalmente utilizzato in una connessione TLS, costringendo ad utilizzare il livello di crittografia negoziato, che di solito è più forte. Quando un client si connette a un server, avvia un processo di handshake per stabilire una connessione sicura. Durante questo processo, il client e il server concordano una suite di cifratura. Se un aggressore intercetta l'handshake e forza un livello di crittografia inferiore, la connessione può diventare vulnerabile agli attacchi. Funziona aggiungendo un valore speciale all'elenco delle suite di cifratura inviate dal client durante l'handshake. Se il server riceve questo valore, si rifiuterà di stabilire una connessione utilizzando una suite di cifratura più debole. È un'importante funzione di sicurezza che aiuta a proteggere dagli attacchi di downgrade e a mantenere l'integrità e la riservatezza delle connessioni TLS.

Inoltre la scansione ha potuto verificare al protezione attiva a diversi tipi di attacco mirato, dovuti a vulnerabilità quali la possibilità di rinegoziazione e compressione TLS.

La rinegoziazione TLS è una funzione del protocollo precedente a TLS 1.3, che consente di avviare, lato client, un nuovo handshake all'interno di una sessione esistente. Questo comportamento, introduce rischi per la sicurezza poiché un attaccante potrebbe sfruttare questo scambio per performare un attacco di tipo DoS, iniziando molteplici handshake, o un Man-in-the Middle inserendo dei dati malevoli sfruttando la vulnerabilità CVE-2009-3555 [54], che permette di fare una richiesta non autenticata al server. Oggi la rinegoziazione non è più supportata a partire dall'ultima versione del protocollo, infatti dalla scansione non risulta supportata.

La compressione TLS, anch'essa una funzione di TLS nelle versioni precedenti a 1.3, consente di comprimere i dati prima che vengano crittografati e trasmessi in rete, per poi essere decompressi dal destinatario prima di essere elaborati. Ciò può migliorare le prestazioni delle connessioni TLS riducendo la quantità di dati da trasmettere, il che può contribuire a ridurre la latenza della rete e a migliorare il throughput complessivo. È stata scoperta una vulnerabilità nella funzione di compressione TLS che ha permesso ad un aggressore di eseguire un attacco chiamato CRIME, acronimo per Compression Ration Info-leak Made Easy. Questo attacco sfruttava il fatto che i dati compressi possano far trapelare informazioni sui dati in chiaro, rendendo possibile per un aggressore dedurre informazioni sensibili. Per questo motivo l'uso della compressione è stata rimossa dall'ultima versione del protocollo. Per migliorare le prestazioni delle connessioni di rete, si utilizzano altre tecniche, come la compressione HTTP, invece della compressione TLS. Applicare queste tecniche a livello applicativo, piuttosto che al livello di trasporto, permette di ridurre al minimo il rischio di perdita di dati.

Nella scansione è possibile notare anche che il sistema risulta protetto da attacchi di tipo heartbleed, di tipo ROBOT e di tipo CCS Injection.

Heartbleed, CVE-2014-0160 [55], è una vulnerabilità resa pubblica nel 2014 della libreria crittografica OpenSSL, ampiamente utilizzata da TLS. Questa vulnerabilità è causata da un difetto nell'implementazione in OpenSSL dell'estensione TLS heartbeat, estensione che consente ad un client di inviare un messaggio ad un server chiedendogli di restituire una risposta contenente una copia di una determinata quantità di memoria. In una versione vulnerabile di OpenSSL, il server potrebbe essere indotto a inviare una quantità di memoria superiore a quella richiesta, esponendo potenzialmente informazioni sensibili memorizzate nella memoria del server.

ROBOT, acronimo di Return Of Bleichenbacher's Oracle Threat, si riferisce a una vulnerabilità, inizialmente scoperta nel 1998 da Daniel Bleichbacher, che affligge sistemi che supportano versioni vulnerabili nella crittografia con RSA. Questa vulnerabilità permette ad un utente malintenzionato di eseguire un tipo di attacco noto come Bleichenbacher oracle attack, che prevede l'invio di richieste accuratamente create ad un server TLS e l'analisi delle sue risposte per ottenere informazioni sulle chiavi di crittografia utilizzate e di conseguenza potenzialmente decifrare le informazioni sensibili trasmesse tramite una connessione TLS. Il sistema in esame non è vulnerabile perché non supporta ciphersuite con RSA.

La CCS Injection, che sta per Cross-Protocol Scripting Injection, è un tipo di vulnerabilità di sicurezza che colpisce alcune applicazioni Web che utilizzano entrambi i protocolli HTTP e

HTTPS. Questa vulnerabilità consente ad un aggressore di iniettare script dannosi nel contenuto di una risposta HTTPS sfruttando una debolezza nel modo in cui l'applicazione gestisce il contenuto misto. Questa vulnerabilità può essere attenuata implementando criteri più severi per i contenuti misti e utilizzando intestazioni CSP, Content Security Policy.

6.5 Sfruttamento delle vulnerabilità

Dopo aver raccolto informazioni, valutato e verificato l'assenza di alcune vulnerabilità è stato possibile constatare l'assenza di due intestazioni, X-Frame-Header e X-XSS-Protection, e la presenza di compressione a livello di protocollo applicativo HTTP. Poiché un utente malintenzionato potrebbe tentare di sfruttare queste vulnerabilità, si è proceduto a testare l'effettivo rischio che ne deriva.

Clickjacking attack

L'intestazione X-Frame-Options è una intestazione di sicurezza che, se aggiunto, aiuta a proteggere le applicazioni web dagli attacchi di tipo clickjacking. Il clickjacking è un tipo di attacco in cui un aggressore inganna l'utente inducendolo a fare clic su un elemento mascherato o nascosto di una pagina Web, facendogli compiere involontariamente un'azione che non intendeva compiere, per esempio utilizzando un elemento HTML iframe. L'intestazione X-Frame-Options specifica se una pagina web può essere visualizzata all'interno di questo tipo di elementi. Se ne è discussa nei capitoli precedenti l'efficacia dell'intestazione e il motivo per cui non è stata aggiunta alla configurazione di Nginx.

La soluzione alternativa, più sicura, è l'utilizzo della direttiva `frame_ancestors` all'interno della content security policy del sito. Per testarne l'efficacia nei confronti di attacco di tipo clickjacking si è proceduto a creare una pagina HTML con all'interno un elemento iframe. Il codice è il seguente:

```
<html>
    <body>
        <iframe src="https://192.168.1.235/">
    </body>
</html>
```

Questa pagina dovrebbe permettere, in assenza di protezioni, di visualizzare, all'interno di un riquadro, la pagina sorgente definita nel frame. Visualizzando la pagina web creata, si è potuto verificare quanto ipotizzato, ovvero che, anche se non è presente l'header X-Frame-Option, in caso di tentativo di attacco, la pagina rifiuterà qualsiasi tipo di connessione grazie alla direttiva `frame_ancestors`

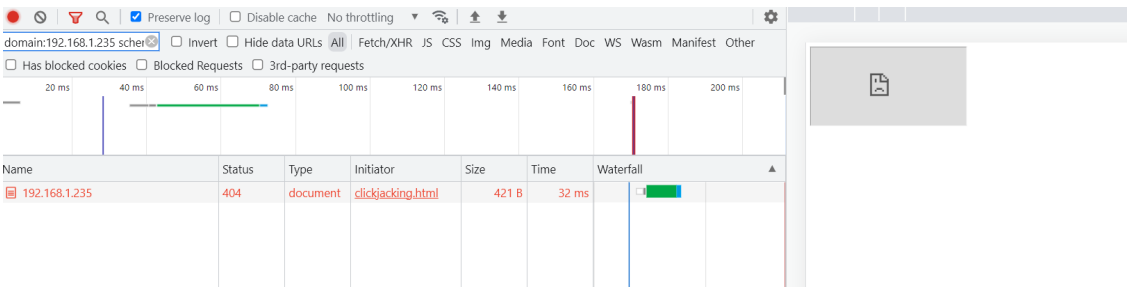

Figura 6.4. Pagina non caricata all'interno del frame creato

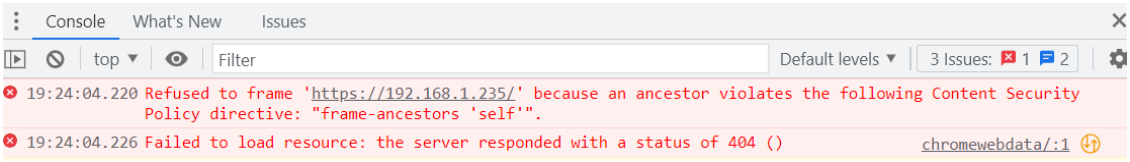

Figura 6.5. Errore rilevato in seguito alla connessione rifiutata

Cross-site scripting attack

Il cross-site scripting (XSS) è un tipo di vulnerabilità che consente ad un utente malintenzionato di iniettare codice dannoso in una pagina Web visualizzata da altri utenti. Questo può essere utilizzato per rubare informazioni sensibili, come le credenziali di accesso o i dati personali, o per eseguire azioni per conto dell'utente vittima, come effettuare acquisti non autorizzati o pubblicare contenuti dannosi.

Esistono due tipi principali di attacchi XSS:

- Reflected XSS: si verifica quando un aggressore inietta codice dannoso in una richiesta HTTP, che viene poi inviato all'utente nella risposta del server senza che questo venga validato correttamente.
- Stored XSS: si verifica quando viene iniettato codice dannoso in una pagina web, come prima, ma in questo caso viene poi memorizzato sul server e servito a tutti gli utenti che visualizzano la pagina.

L'intestazione X-XSS-Protection filtra le informazioni sospette per bloccare gli attacchi reflected XSS. Quando l'intestazione identifica XSS, impedisce il caricamento della pagina senza sanificare gli input all'interno della pagina. In realtà questo approccio è deprecato, poiché a sua volte vulnerabile, in favore del Content Security Policy con l'opzione `unsafe-inline` disabilitata.

Per testare questa funzionalità è stata disabilitata la mutua autenticazione TLS, che dovrebbe comunque essere bypassata da un utente malevolo prima di poter arrivare ad utilizzare questa funzione. Avendo inserito la validazione dell'input nel form di login, il tentativo di XSS, con lo script `<script>alert(123)</script>` al posto del nome utente, viene bloccato sul nascere. Inoltre l'input non viene utilizzato dal server nella risposta, quindi si tratta di un rischio inesistente.

Per completezza, si è voluta testare l'efficacia di CSP al posto dell'header in esame, facendo in modo di visualizzare il nome utente nella risposta del server, verificando che il contenuto sia sanificato e gli script non abbiano effetto sul browser del client.

Figura 6.6. Risposta del server con input sanificato alla riga 15

Sono stati verificati diversi tipi di script, ma, come si può vedere dalle immagini proposte, il testo viene sempre sanificato e non ha più alcun effetto.

Lo Stored XSS è invece un tipo di attacco per cui questa intestazione non serve, ma necessita che l'input utente venga sanificato. Questo è stato preso in esame nei capitoli precedenti.

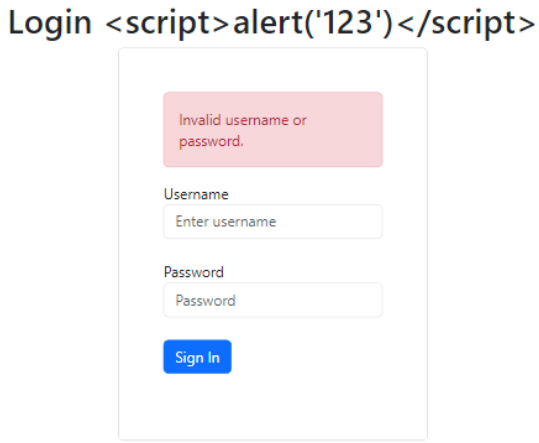

Figura 6.7. Interfaccia utente con visualizzazione dell'input sanificato

BREACH attack

L'attacco BREACH è una categoria di vulnerabilità dovute all'utilizzo della compressione HTTP, dello stesso tipo di CRIME, citato precedentemente. L'attacco consente a un aggressore di estrarre informazioni sensibili dalla risposta compressa inviata dal server. La vulnerabilità è legata al fatto che gli algoritmi di compressione HTTP funzionano trovando schemi ripetuti nel corpo della risposta e gli aggressori possono sfruttare questa proprietà per dedurre informazioni sui dati segreti della risposta. L'unico modo veramente efficace per proteggersi da questo tipo di attacchi è disabilitare la compressione.

Quello che si è potuto evincere, è che il server utilizza la compressione HTTP in tutte le sue risposte, e non solo durante la trasmissione del video, come esplicitamente configurato. Da questo, si è potuto verificare che la configurazione a più basso livello di Nginx, nel file `nginx.conf`, fa si che venga applicata la compressione di default. È quindi possibile limitare la minaccia rimuovendo questa configurazione. Purtroppo non è possibile eliminare completamente questa vulnerabilità perché la compressione `gzip` è necessaria per diminuire la latenza di trasmissione del video. Si possono però adottare tecniche che randomizzano la compressione, che aggiungono padding in modo da standardizzare la lunghezza del payload o altre. Inoltre anche la protezione CSRF, Cross Site Request Forgery, già implementata, permette di limitare la superficie di questo tipo di attacco aggiungendo un token, ovvero un valore segreto unico e imprevedibile inserito dal server nelle risposte HTTP.

6.6 Risultati

Testare il sistema ha permesso di verificare l'efficacia delle soluzioni adottate durante la sua progettazione. In particolare, è stato possibile simulare un utente malintenzionato, volenteroso di accedere al sistema senza possederne il certificato di chiave asimmetrica e le credenziali necessarie. La limitazione dovuta alla mancanza di un dominio associato all'applicazione web e all'ambiente isolato in cui sono stati eseguiti i test, non hanno permesso di valutare la presenza dell'applicazione sui motori di ricerca, così come l'utilizzo di strumenti di analisi che richiedono un dominio in input. Nonostante questo, è stato possibile testare tutte le misure implementate nel sistema ed effettuare una vasta analisi. Inoltre, per ovviare alla limitata esperienza nel penetration test, i test manuali sono stati affiancati da strumenti automatici.

Le vulnerabilità identificate sono in numero limitato. Mentre per le prime due non possono essere sfruttate, l'attacco di tipo BREACH può essere però mitigato adottando le tecniche corrette.

È possibile quindi concludere che le soluzioni adottate per proteggere il sistema sono effettivamente valide nella protezione dagli attacchi presi in analisi, o possono essere mitigate con

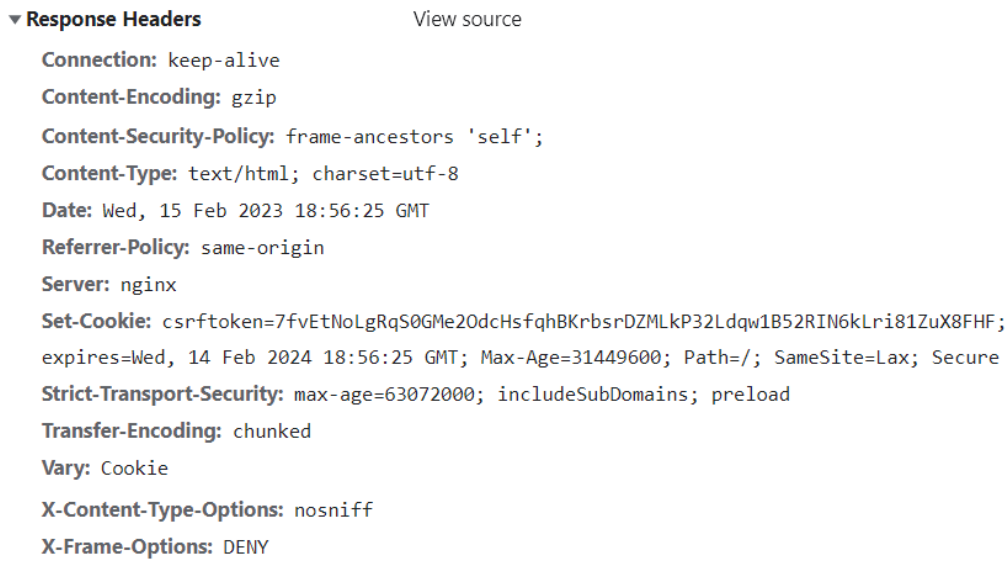

Figura 6.8. Header di una risposta HTTP con token CSRF

determinate misure. Per poter affermare che il sistema è veramente sicuro, si dovrà verificarne lo stato in un ambiente reale, con un sistema completo. Inoltre, per rendere un penetration test valido, questo dovrebbe essere eseguito e certificato da terze parti, con una conoscenza nulla o limitata del sistema in modo da agire esattamente come potrebbe fare un hacker.

Capitolo 7

Conclusioni

Ci sono ancora sfide importanti da affrontare prima che la guida autonoma prenda veramente il sopravvento. L'incremento della connettività e dei rischi da un punto di vista informatico che ne derivano, fanno si che si debbano adottare delle misure di sicurezza sempre più avanzate, fin dall'inizio del progetto.

Per questo motivo, il lavoro proposto, riveste un ruolo importante nello sviluppo del progetto di un rover terrestre a guida autonoma, e non solo.

Grazie al lavoro di tesi dal titolo "Deep Learning-Based Real-Time Detection and Single-Object Tracking on a GPU based embedded device" [1], è stato possibile lavorare sul codice sviluppato dagli ingegneri Caliò e Forese. Su questo è stata effettuata un'analisi orientata alla sicurezza, che ha permesso di evidenziare alcune vulnerabilità. La sfida è stata quella di superare il limite dato dall'esperienza, spesso fondamentale per identificare i problemi all'interno di un software, ed utilizzare la documentazione, gli studi e gli strumenti di analisi a disposizione. Ne è seguita una fase in cui sono stati risolte le vulnerabilità identificate all'interno del codice. Un passo in avanti potrebbe essere fatto in futuro prendendo in considerazione anche le vulnerabilità da cui è affetta la rete pre-trained utilizzata dal programma, anche sulla base dell'ambiente in cui il rover dovrà agire.

Per far comunicare il rover con l'esterno, attraverso reti non sicure, è stato progettato un sistema di comunicazione su browser. La sfida nella sua realizzazione, è stata quella di trovare un compromesso tra prestazioni e sicurezza. Infatti, nonostante esistano protocolli in grado di gestire meglio lo streaming in real-time, questi ne risentono abbondantemente dal lato della sicurezza, così come esistono protocolli che si trovano a lavorare in maniera contraria. Nel lavoro proposto, sono state prese in considerazione delle soluzioni che danno priorità alla sicurezza, ma che permettono di ottenere dei buoni risultati anche nello streaming del video, se configurati in maniera corretta. Durante il lavoro è stato preso in considerazione fin dall'inizio l'utilizzo di un server web ospitato direttamente sulla scheda. Questo è stato fatto ipotizzando che questa soluzione potesse offrire vari vantaggi da un punto di vista della sicurezza e della latenza. Tutto ciò apre la strada a degli approfondimenti futuri, in cui si potrebbero studiare gli effettivi vantaggi e svantaggi di questa soluzione, rispetto all'utilizzo di un web server su cloud con comunicazione client-client, normalmente utilizzata per servizi di videochiamata. Questo, alla luce della diffusione di nuovi protocolli come WebRTC, che potrebbero portare a migliorare le prestazioni della comunicazione del rover.

Nella fase finale è stata valutata l'efficacia delle soluzioni implementate, tramite l'utilizzo di strumenti automatici e tecniche di penetrazione dell'applicazione. È stato possibile verificare che ciò che è stato fatto ha portato ad implementare un sistema sicuro, protetto da tutta una serie di attacchi comuni e non. Bisogna però tenere in considerazione che il sistema dovrà essere testato nuovamente ad ogni aggiunta di funzionalità e non appena disponibile in un ambiente reale. Inoltre bisogna tenere conto che in fase di produzione, questo sistema dovrà essere testato anche da enti esterni, in grado di certificarne l'effettiva sicurezza, con l'aiuto sopratutto dell'esperienza, caratteristica importante in questo tipo di analisi.

Bibliografia

[1] A. Caliò and L. Forese, "Deep learning-based real-time detection and single-object tracking on a gpu based embedded device", 2020, Tesi di laurea magistrale, Universidad Nacional de Còrdoba

[2] On Road Automated Driving Committee, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles", April 2021, DOI 10.4271/J3016_202104

[3] ENISA, "Good practices for security of smart cars", November 2019, DOI 10.2824/17802

[4] UNECE, "UN regulation no. 155 - cyber security and cyber security management system", March 2021, https://unece.org/sites/default/files/2021-03/R155e.pdf

[5] UNECE, "UN regulation no. 156 - software update and software update management system", March 2021, https://unece.org/sites/default/files/2021-03/R156e.pdf

[6] ISO and SAE, "Iso/sae 21434:2021 - road vehicles - cybersecurity engineering", August 2021, https://www.iso.org/standard/70918.html

[7] AGID, "Linee guida per lo sviluppo del software sicuro", March 2020, https://www.agid.gov.it/sites/default/files/repository_files/allegato_2_-_linee_guida_per_lo_sviluppo_sicuro_di_codice.pdf

[8] ETSI, "En 303 645 - cyber security for consumer internet of things: Baseline requirements", June 2020, https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf

[9] Unione Europea, "General data protection regulation (gdpr) - regolamento (ue) n. 2016/679", April 2016, https://gdpr.eu/tag/gdpr/

[10] Owasp Top 10 Team, "Owasp top 10 vulnerabilties", 2021, https://owasp.org/Top10/

[11] The MITRE Corporation, "CWE-494", July 2006, https://cwe.mitre.org/data/definitions/494.html

[12] Nvidia Developer, "Nvidia jetson nano manual." https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[13] G. Dede, R. Hamon, H. Junklewitz, R. Naydenov, A. Malatras, and M. J. Sanchez, "Cybersecurity challenges in the uptake of artificial intelligence in autonomous driving", February 2021, DOI 10.2760/551271

[14] European Union Agency for Cybersecurity, A. Malatras, and G. Dede, "AI cybersecurity challenges : threat landscape for artificial intelligence", December 2020, DOI 10.2824/238222

[15] Bandit Developers, "Bandit tool documentation." https://bandit.readthedocs.io/en/latest/#

[16] HCL Software, "Hcl appscan codesweep tool documentation." https://www.hcltechsw.com/it/products/appscan/codesweep

[17] Zup, "Horusec tool documentation." https://horusec.io/site/

[18] NIST, "National vulnerability database." https://nvd.nist.gov/vuln/

[19] NIST, "CVE-2019-5064", January 2020, https://nvd.nist.gov/vuln/detail/cve-2019-5064

[20] NIST, "CVE-2019-16249", September 2019, https://nvd.nist.gov/vuln/detail/cve-2019-16249

[21] NIST, "CVE-2017-12852", August 2017, https://nvd.nist.gov/vuln/detail/cve-2017-12852

[22] NIST, "CVE-2019-6446", January 2019, https://nvd.nist.gov/vuln/detail/cve-2019-6446

[23] NIST, "CVE-2021-34141", December 2021, https://nvd.nist.gov/vuln/detail/cve-2021-34141
[24] NIST, "CVE-2021-41496", December 2021, https://nvd.nist.gov/vuln/detail/cve-2021-41496
[25] G. van Rossum, B. Warsaw, and N. Coghlan, "Pep8 - style guide for python code", August 2013, https://peps.python.org/pep-0008/
[26] S. K. Barker and P. Shenoy, "Empirical evaluation of latency-sensitive application performance in the cloud", Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems, February 2010, pp. 35–46, DOI 10.1145/1730836.1730842
[27] D. A. Popescu, N. Zilberman, and A. Moore, "Characterizing the impact of network latency on cloud-based applications' performance", November 2017, DOI 10.17863/CAM.17588
[28] Django Software Foundation, "Django framework documentation." https://www.djangoproject.com/
[29] Pallets, "Flask framework documentation." https://flask.palletsprojects.com/en/2.2.x/
[30] D. Kunda, S. Chihana, and M. Sinyinda, "Web server performance of apache and nginx: A systematic literature review", Computer Engineering and Intelligent Systems, vol. 8, 2017, pp. 43–52. https://www.researchgate.net/publication/329118749_Web_Server_Performance_of_Apache_and_Nginx_A_Systematic_Literature_Review
[31] E. Carrara, K. Norrman, D. McGrew, M. Naslund, and M. Baugher, "The Secure Real-time Transport Protocol (SRTP)." RFC-3711, March 2004, DOI 10.17487/RFC3711
[32] A. Hussein, I. H. Elhajj, A. Chehab, and A. Kayssi, "Securing diameter: Comparing tls, dtls, and ipsec", 2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET), November 2016, pp. 1–8, DOI 10.1109/IMCET.2016.7777417
[33] I. Sysoev and Nginx Inc, "Nginx documentation." https://nginx.org/en/docs/
[34] E. De Cristofaro, H. Du, J. Freudiger, and G. Norcie, "A comparative usability study of two-factor authentication", February 2014, DOI 10.14722/usec.2014.23025
[35] J. Williamson and K. Curran, "Best practice in multi-factor authentication", Semiconductor Science and Information Devices, vol. 3, May 2021, DOI 10.30564/ssid.v3i1.3152
[36] T. Zink and M. Waldvogel, "X.509 user certificate-based two-factor authentication for web applications", March 2017, DOI 10.13140/RG.2.2.31620.94083
[37] P. Grassi, J. Fenton, E. Newton, R. Perlner, A. Regenscheid, W. Burr, J. Richer, N. Lefkovitz, J. Danker, Y. Choong, K. Greene, and M. Theofanos, "Digital identity guidelines: Authentication and lifecycle management", March 2020, DOI 10.6028/NIST.SP.800-63b
[38] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2." RFC-5246, August 2008, DOI 10.17487/RFC5246
[39] K. K. Vasan and P. A. R. Kumar, "Taxonomy of ssl/tls attacks", International Journal of Computer Network and Information Security, vol. 8, February 2016, pp. 15–24, DOI 10.5815/ijcnis.2016.02.02
[40] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3." RFC-8446, August 2018, DOI 10.17487/RFC8446
[41] A. Alqattaa and A. Abmuth, "Analysis of the internet security protocol tls version 1.3", January 2019. https://www.researchgate.net/publication/341680184_Analysis_of_the_Internet_Security_Protocol_TLS_Version_13
[42] K. McKay and D. Cooper, "Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations", August 2019, DOI 10.6028/NIST.SP.800-52r2
[43] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols." RFC-8439, July 2018, DOI 10.17487/RFC8439
[44] D. John and C. Klensin, "Simple Mail Transfer Protocol." RFC-5321, October 2008, DOI 10.17487/RFC5321
[45] J. Hodges, C. Jackson, and A. Barth, "HTTP Strict Transport Security (HSTS)." RFC-6797, November 2012, DOI 10.17487/RFC6797
[46] Ubuntu Documentation Team, "Ufw documentation." https://help.ubuntu.com/18.04/serverguide/firewall.html
[47] Python Software Foundation, "Venv library documentation." https://docs.python.org/3/library/venv.html
[48] Benoit Chesneau, "Gunicorn documentation." https://docs.gunicorn.org/

[49] OpenSSL Project Authors, "Openssl documentation." https://www.openssl.org/
[50] The OWASP Foundation, "Owasp web security testing guide", December 2020, https://owasp.org/www-project-web-security-testing-guide/v42/
[51] The PTES Team Revision, "Ptes technical guidelines", April 2016, http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines
[52] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, "Nist 800-115 - technical guide to information security testing and assessment", April 2021, DOI 10.6028/NIST.SP.800-115
[53] P. Herzog, "Open source security testing methodology manual", December 2010, https://www.isecom.org/OSSTMM.3.pdf
[54] NIST, "CVE-2009-3555", November 2009, https://nvd.nist.gov/vuln/detail/cve-2009-3555
[55] NIST, "CVE-2014-0160", April 2014, https://nvd.nist.gov/vuln/detail/cve-2014-0160

Appendice A

Tassonomia degli asset in un veicolo autonomo

Di seguito l'elenco e degli asset identificati da ENISA [3] per un veicolo autonomo.

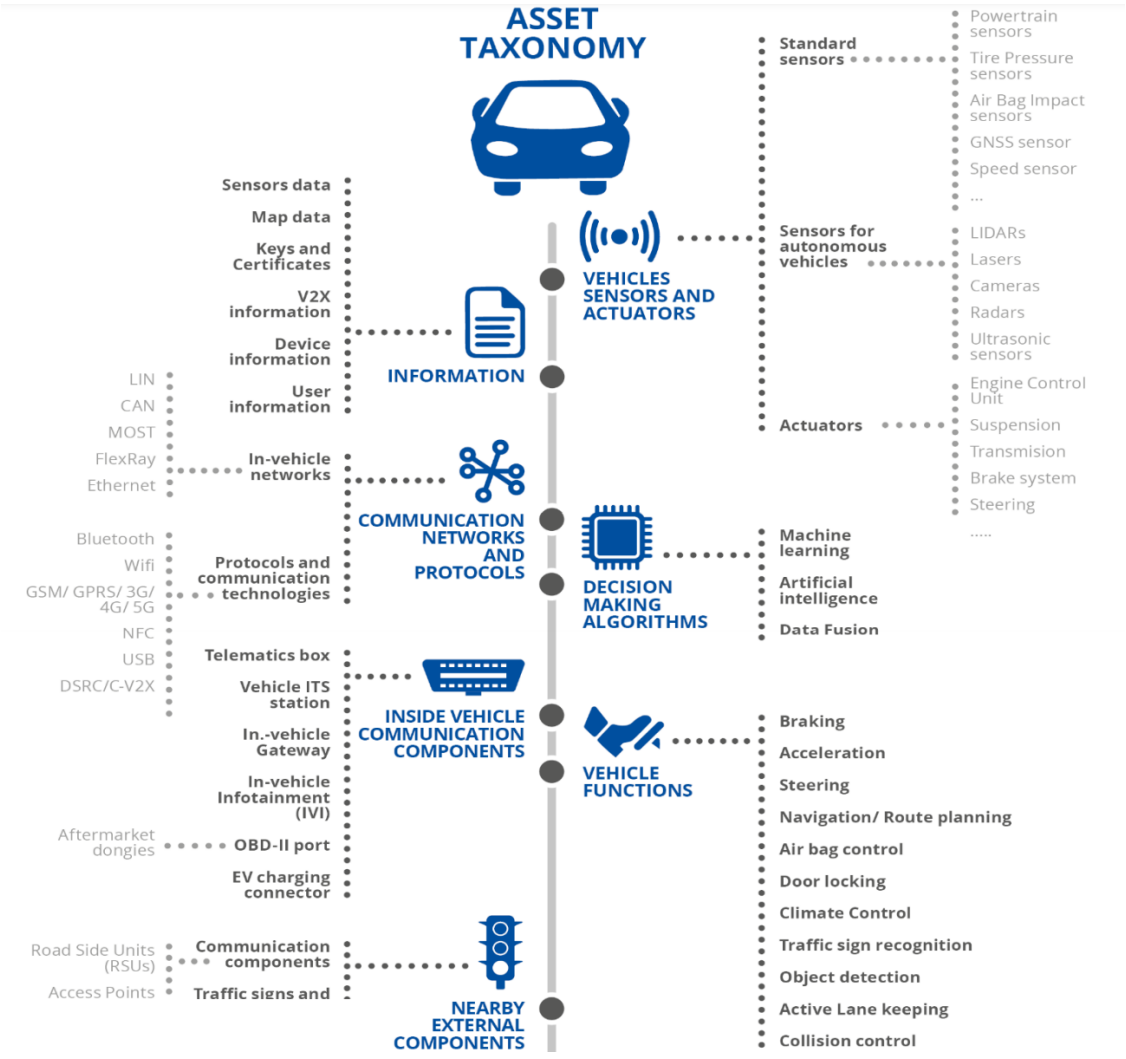

Figura A.1. Asset identificati da ENISA in un veicolo autonomo

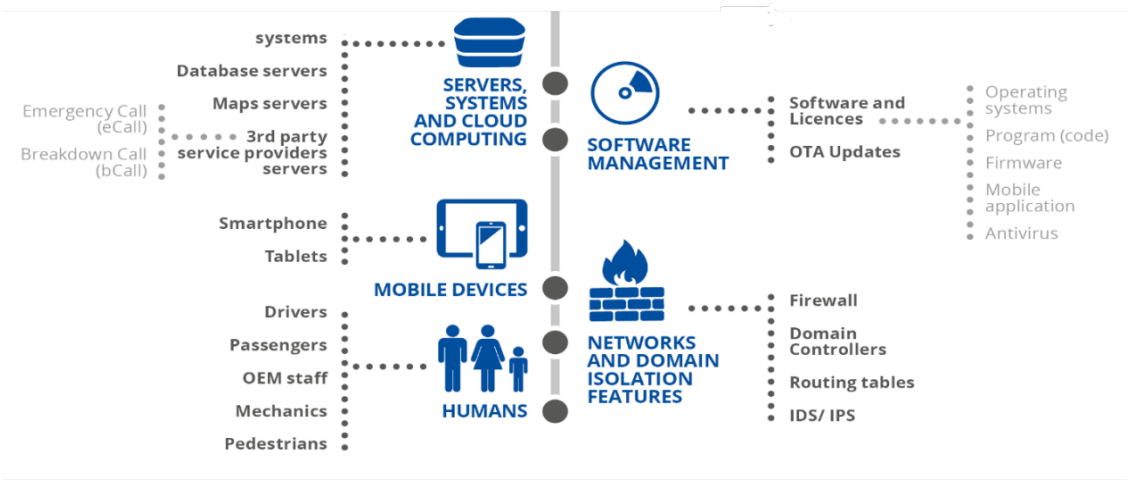

Figura A.2. Asset identificati da ENISA in un veicolo autonomo

Appendice B

Misurazioni di throughput e latenza

dati trasmessi(MB)	*tempo (s)*	*throughput(Mbps)*	*latenza(s)*
71,8	24,90	23,07	1,83
73,9	26,20	22,56	1,84
75,4	25,70	23,47	1,84
80,1	27,30	23,47	1,84
71,3	25,60	22,28	1,85
72,5	24,90	23,29	1,83
74,0	26,20	22,60	1,85
85,4	25,70	26,58	1,79
79,2	27,30	23,21	1,84
75,6	25,60	23,63	1,84

Tabella B.1. Misurazioni con TLS_AES_128_GCM_SHA256

dati trasmessi(MB)	*tempo (s)*	*throughput(Mbps)*	*latenza(s)*
67,3	25,40	21,20	1,88
68,1	24,91	21,87	1,87
78,9	28,79	21,92	1,87
65,2	26,50	19,68	1,89
80,5	29,90	21,54	1,88
70,9	26,43	21,46	1,88
68,7	24,90	22,07	1,86
80,0	27,80	23,02	1,85
64,9	26,47	19,61	1,89
80,5	29,90	21,54	1,88

Tabella B.2. Misurazioni con TLS_AES_256_GCM_SHA384

dati trasmessi(MB)	*tempo (s)*	*throughput(Mbps)*	*latenza(s)*
73,1	22,26	26,27	1,79
86,9	27,14	25,62	1,79
80,4	25,40	25,32	1,80
81,1	26,35	24,62	1,81
80,5	25,70	25,06	1,80
83,2	24,26	27,44	1,78
86,9	26,20	26,53	1,78
79,4	25,45	24,96	1,80
81,6	26,35	24,77	1,81
79,5	25,73	24,72	1,81

Tabella B.3. Misurazioni con TLS_CHACHA20_POLY1305_SHA256

chiphersuite	*latenza(s)*	*throughput(Mbps)*
TLS_AES_128_GCM_SHA256	1,84	23,42
TLS_AES_256_GCM_SHA384	1,87	21,35
TLS_CHACHA20_POLY1305_SHA256	1,80	25,38

Tabella B.4. Valori mediati di throughput e latenza

Appendice C

Penetration test report

C.1 Nmap

nmap -p - -A 192.168.1.235

```
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-20 16:49 ora solare Europa
    occidentale
NSOCK ERROR [0.5600s] ssl_init_helper(): OpenSSL legacy provider failed to
    load.

Nmap scan report for jetson-desktop (192.168.1.235)
Host is up (0.033s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 2048 864c8593a83eddce050fa728b797c349 (RSA)
| 256 aa55c1570ecc3bbbfc25dbbda9560f41 (ECDSA)
|_ 256 f5d64b81d322620be216f81340032c81 (ED25519)
443/tcp open ssl/http nginx
|_http-title: 403 Forbidden
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=192.168.1.235/organizationName=Politecnico
    Torino MCA/stateOrProvinceName=Piedmont/countryName=IT
| Not valid before: 2023-01-20T11:59:44
|_Not valid after: 2024-01-20T11:59:44
MAC Address: 48:B0:2D:5B:03:9C (Nvidia)
Warning: OSScan results may be unreliable because we could not find at least
    1 open and 1 closed port
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.10 - 4.11, Linux 3.16 - 4.6, Linux 3.2 - 4.9, Linux 4.4
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT ADDRESS
1 32.79 ms jetson-desktop (192.168.1.235)

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 115.81 seconds
```

C.2 Masscan

masscan -p0-65535 192.168.1.235

```
Discovered open port 22/tcp on 192.168.1.235
Discovered open port 443/tcp on 192.168.1.235
```

C.3 Nikto

```
---------------------------------------------------------------------------
                        Information
---------------------------------------------------------------------------
Test ID:  999102
OSVDB ID:  0
Message:  The X-XSS-Protection header is not defined. This header can hint to
    the user agent to protect against some forms of XSS
Reason:
---------------------------------------------------------------------------
                        Request
---------------------------------------------------------------------------
GET / HTTP/1.1
Connection: Keep-Alive
Host: 192.168.1.235
User-Agent: Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:map_codes)

---------------------------------------------------------------------------
                        Response
---------------------------------------------------------------------------
HTTP/1.1 403 Forbidden
server: nginx
date: Wed, 15 Feb 2023 20:34:11 GMT
content-type: text/html
content-length: 162
connection: keep-alive

<html>
<head><title>403 Forbidden</title></head>
<body bgcolor="white">
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>

---------------------------------------------------------------------------
                        Information
---------------------------------------------------------------------------
Test ID:  999955
OSVDB ID:  0
Message:  The site uses SSL and Expect-CT header is not present.
Reason:
---------------------------------------------------------------------------
                        Request
```

```
---------------------------------------------------------------------------
GET / HTTP/1.1
Connection: Keep-Alive
Host: 192.168.1.235
User-Agent: Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:map_codes)

---------------------------------------------------------------------------
                      Response
---------------------------------------------------------------------------
HTTP/1.1 403 Forbidden
server: nginx
date: Wed, 15 Feb 2023 20:34:11 GMT
content-type: text/html
content-length: 162
connection: keep-alive

<html>
<head><title>403 Forbidden</title></head>
<body bgcolor="white">
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>

---------------------------------------------------------------------------
                      Information
---------------------------------------------------------------------------
Test ID:  999957
OSVDB ID:  0
Message:  The anti-clickjacking X-Frame-Options header is not present.
Reason:
---------------------------------------------------------------------------
                      Request
---------------------------------------------------------------------------
GET / HTTP/1.1
Connection: Keep-Alive
Host: 192.168.1.235
User-Agent: Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:map_codes)

---------------------------------------------------------------------------
                      Response
---------------------------------------------------------------------------
HTTP/1.1 403 Forbidden
server: nginx
date: Wed, 15 Feb 2023 20:34:11 GMT
content-type: text/html
content-length: 162
connection: keep-alive

<html>
<head><title>403 Forbidden</title></head>
<body bgcolor="white">
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>

---------------------------------------------------------------------------
```

```
                        Information
---------------------------------------------------------------------------
Test ID:  999966
OSVDB ID:  0
Message:  The Content-Encoding header is set to "deflate" this may mean that
    the server is vulnerable to the BREACH attack.
Reason:
---------------------------------------------------------------------------
                        Request
---------------------------------------------------------------------------
GET / HTTP/1.1
Accept-Encoding: deflate, gzip
User-Agent: Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:headers: BREACH
    Test)
Connection: Keep-Alive
Host: 192.168.1.235

---------------------------------------------------------------------------
                        Response
---------------------------------------------------------------------------
HTTP/1.1 403 Forbidden
server: nginx
date: Wed, 15 Feb 2023 20:35:35 GMT
content-type: text/html
transfer-encoding: chunked
connection: keep-alive
content-encoding: gzip
```

C.4 Sslscan

sslscan 192.168.1.235:443

```
    Version: 2.0.15-static
OpenSSL 1.1.1q-dev xx XXX xxxx
Connected to 192.168.1.235
Testing SSL server 192.168.1.235 on port 443 using SNI name 192.168.1.235

 SSL/TLS Protocols:
SSLv2 disabled
SSLv3 disabled
TLSv1.0 disabled
TLSv1.1 disabled
TLSv1.2 disabled
TLSv1.3 enabled

 TLS Fallback SCSV:
Server supports TLS Fallback SCSV

 TLS renegotiation:
Session renegotiation not supported

 TLS Compression:
Compression disabled

 Heartbleed:
TLSv1.3 not vulnerable to heartbleed
```

```
 Supported Server Cipher(s):
Preferred TLSv1.3 256 bits TLS_AES_256_GCM_SHA384 Curve P-384 DHE 384
Accepted TLSv1.3 256 bits TLS_CHACHA20_POLY1305_SHA256 Curve P-384 DHE 384
Accepted TLSv1.3 128 bits TLS_AES_128_GCM_SHA256 Curve P-384 DHE 384

 Server Key Exchange Group(s):
TLSv1.3 128 bits secp256r1 (NIST P-256)
TLSv1.3 192 bits secp384r1 (NIST P-384)

 SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength: 2048
Subject: 192.168.1.235
Issuer: 192.168.1.235
Not valid before: Jan 20 11:59:44 2023 GMT
Not valid after: Jan 20 11:59:44 2024 GMT
```

C.5 Sslyze

sslyze 192.168.1.235

```
CHECKING CONNECTIVITY TO SERVER(S)
 ----------------------------------
   192.168.1.235:443 => 192.168.1.235 WARNING: Server requested optional
       client authentication

SCAN RESULTS FOR 192.168.1.235:443 - 192.168.1.235
 ---------------------------------------------------
 * Certificates Information:
       Hostname sent for SNI: 192.168.1.235
       Number of certificates detected: 1

     Certificate #0 ( _RSAPublicKey )
       SHA1 Fingerprint: 0c710e5ccfae8f3c63e73e5fbb754b06364cc076
       Common Name: 192.168.1.235
       Issuer: 192.168.1.235
       Serial Number: 457088304146916272779220945961849622987231994265
       Not Before: 2023-01-20
       Not After: 2024-01-20
       Public Key Algorithm: _RSAPublicKey
       Signature Algorithm: sha256
       Key Size: 2048
       Exponent: 65537
       DNS Subject Alternative Names: []

     Certificate #0 - Trust
       Hostname Validation: OK - Certificate matches server hostname
       Android CA Store (13.0.0_r8): FAILED - Certificate is NOT Trusted:
           self signed certificate
       Apple CA Store (iOS 15.1, iPadOS 15.1, macOS 12.1, tvOS 15.1, and
           watchOS 8.1):FAILED - Certificate is NOT Trusted: self signed
           certificate
       Java CA Store (jdk-13.0.2): FAILED - Certificate is NOT Trusted: self
           signed certificate
```

```
      Mozilla CA Store (2022-09-18): FAILED - Certificate is NOT Trusted:
          self signed certificate
      Windows CA Store (2022-08-15): FAILED - Certificate is NOT Trusted:
          self signed certificate
      Symantec 2018 Deprecation: ERROR - Could not build verified chain
          (certificate untrusted?)
      Received Chain: 192.168.1.235
      Verified Chain: ERROR - Could not build verified chain (certificate
          untrusted?)
      Received Chain Contains Anchor: ERROR - Could not build verified chain
          (certificate untrusted?)
      Received Chain Order: OK - Order is valid
      Verified Chain contains SHA1: ERROR - Could not build verified chain
          (certificate untrusted?)

    Certificate #0 - Extensions
      OCSP Must-Staple: NOT SUPPORTED - Extension not found
      Certificate Transparency: NOT SUPPORTED - Extension not found

    Certificate #0 - OCSP Stapling
                    NOT SUPPORTED - Server did not send back an OCSP response

* SSL 2.0 Cipher Suites:
    Attempted to connect using 7 cipher suites; the server rejected all
        cipher suites.

* SSL 3.0 Cipher Suites:
    Attempted to connect using 80 cipher suites; the server rejected all
        cipher suites.

* TLS 1.0 Cipher Suites:
    Attempted to connect using 80 cipher suites; the server rejected all
        cipher suites.

* TLS 1.1 Cipher Suites:
    Attempted to connect using 80 cipher suites; the server rejected all
        cipher suites.

* TLS 1.2 Cipher Suites:
    Attempted to connect using 156 cipher suites; the server rejected all
        cipher suites.

* TLS 1.3 Cipher Suites:
    Attempted to connect using 5 cipher suites.

    The server accepted the following 3 cipher suites:
       TLS_CHACHA20_POLY1305_SHA256 256 ECDH: secp384r1 (384 bits)
       TLS_AES_256_GCM_SHA384 256 ECDH: secp384r1 (384 bits)
       TLS_AES_128_GCM_SHA256 128 ECDH: secp384r1 (384 bits)

* Deflate Compression:
                                      OK - Compression disabled
* OpenSSL CCS Injection:
                                      OK - Not vulnerable to OpenSSL CCS
                                          injection
* OpenSSL Heartbleed:
                                      OK - Not vulnerable to Heartbleed
```

```
* ROBOT Attack:
                                        OK - Not vulnerable, RSA cipher suites
                                            not supported.
* Session Renegotiation:
      Client Renegotiation DoS Attack: OK - Not vulnerable
      Secure Renegotiation: OK - Supported
* Elliptic Curve Key Exchange:
      Supported curves: prime256v1, secp384r1
      Rejected curves: X25519, X448, prime192v1, secp160k1, secp160r1,
          secp160r2, secp192k1, secp224k1, secp224r1, secp256k1, secp521r1,
          sect163k1, sect163r1, sect163r2, sect193r1, sect193r2, sect233k1,
          sect233r1, sect239k1, sect283k1, sect283r1, sect409k1, sect409r1,
          sect571k1, sect571r1
SCANS COMPLETED IN 6.974262 S
---------------------------

COMPLIANCE AGAINST MOZILLA TLS CONFIGURATION
--------------------------------------------
   Checking results against Mozilla's "intermediate" configuration. See
       https://ssl-config.mozilla.org/ for more details.

   192.168.1.235:443: FAILED - Not compliant.
       * certificate_path_validation: Certificate path validation failed for
       1.2.840.113549.1.9.1=martacatto@gmail.com,CN=192.168.1.235,
       OU=Thesist,O=Politecnico Torino MCA,L=Turin,ST=Piedmont,C=IT.
```