UNIVERSITÁ POLITECNICA DELLE MARCHE

FACOLTÁ DI INGEGNERIA

Corso di Laurea Magistrale in
Ingegneria Informatica e dell'Automazione

Progettazione, sviluppo ed implementazione di un sistema per la gestione dell' impatto di CO_2 tramite BlockChain

Docente:
Dott. Spalazzi Luca

Realizzato da:
Antenucci Lucrezia
Mele Alessandro
Traini Davide

Anno Accademico 2021-2022

Indice

Capitolo 1

Introduzione

Il seguente progetto è interamente a scopo didattico: l'obiettivo è quello di progettare, sviluppare ed implementare un software per la gestione di un catalogo contenente materie prime e prodotti trasformati, ognuno dei quali è descritto da una specifica etichetta; all'interno della quale è presente il *CarbonFootprint*, ovvero il quantitativo di CO_2 derivante dalla creazione, o lavorazione, della risorsa. Si richiede la gestione di tre livelli di utenza, ognuno dei quali può svolgere diverse tipologie di azioni:

- Produttore, ha la possibilitá di inserire materie prime;
- Trasformatore, ha la possibilitá di inserire prodotti trasformati;
- Consumatore (Cliente), ha la possibilitá di acquistare i prodotti.

Tutti i livelli di utenza hanno la possibilità di consultare ed estrarre le informazioni relative alle materie prime ed i prodotti trasformati presenti nel catalogo.

Capitolo 2

Ingegneria dei requisiti

2.1 Analisi dei Requisiti

In questa sezione si definiscono i requisiti del sistema attraverso diagrammi i*, i quali consentono di definire i confini del progetto e le funzionalità implementate nel software finale.

2.1.1 Diagramma delle dipendenze

Di seguito, si individuano, tramite diagrammi i*, gli attori principali del sistema e le relative dipendenze.

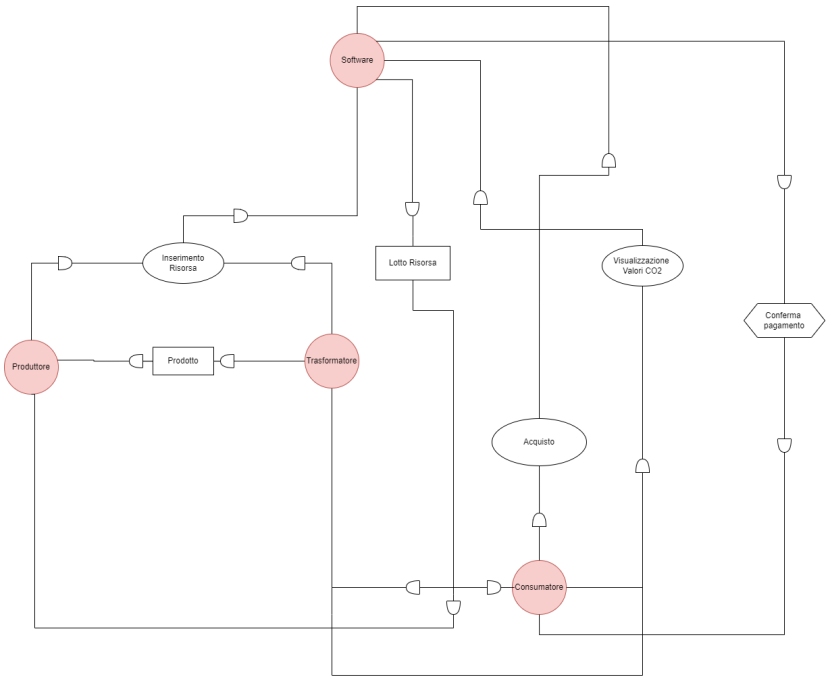

Figura 2.1: Early strategic dependency model

Gli attori del sistema, nello specifico, sono:

- *Produttore*: colui che produce le materie prime e le inserisce nel catalogo tramite il *software*;
- *Trasformatore*: colui che lavora le materie prime acquistate dal *produttore* e ottiene un prodotto trasformato da inserire nel catalogo tramite il *software*;
- *Consumatore*: colui che acquista il prodotto trasformato tramite il *software*;

Tutti e tre i livelli di utenza possono interrogare il *software* per ottenere informazioni sulle risorse presenti nel catalogo.

2.1.2 Diagramma Finale

Il diagramma delle dipendenze si raffina, per mostrare in maniera dettagliata le operazioni eseguibili dagli attori e le rispettive interazioni con il *software*.

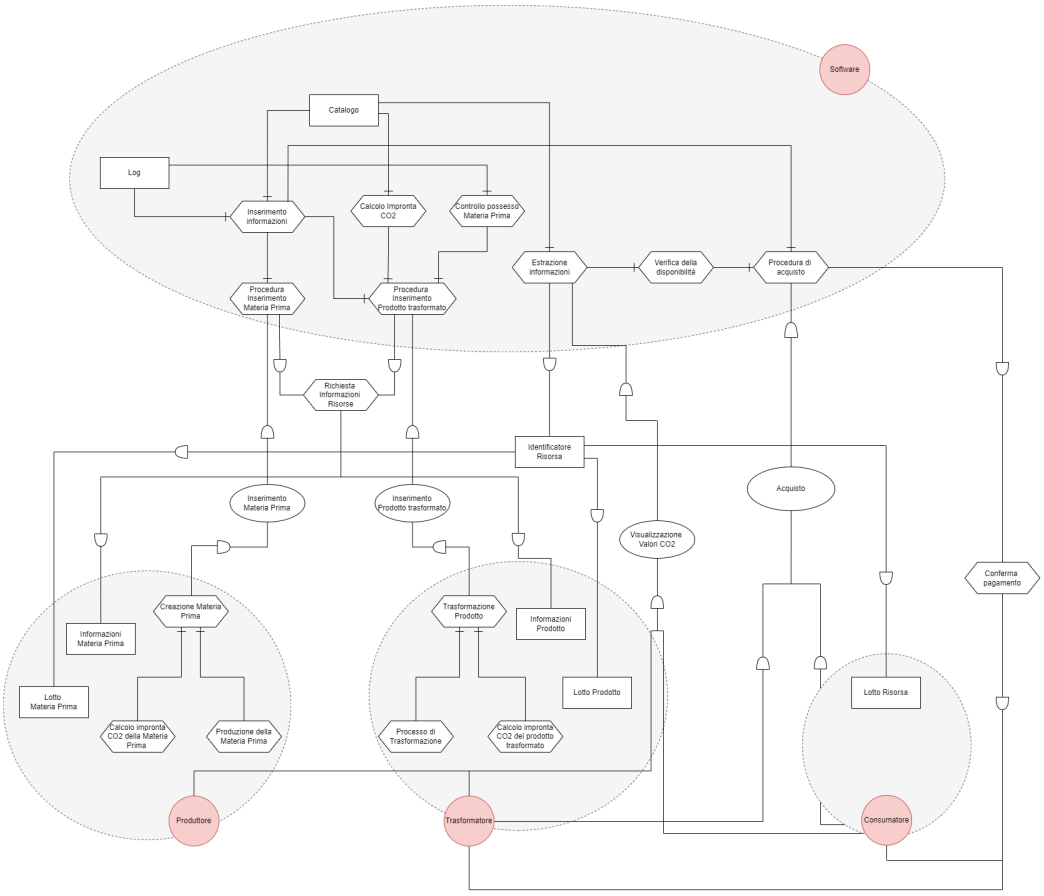

Figura 2.2: Diagramma Finale

Dalla figura 2.2 si estraggono i *workflow* delle operazioni eseguibili dagli attori. In particolare:

- Il *produttore*, per inserire una materia prima nel catalogo, fornisce al *software*: nome, lotto ed il *CarbonFootprint*.
  Il *software* verifica, durante l'inserimento, l'eventuale presenza di risorse con lo stesso lotto e, in caso di esito positivo, viene registrata in un file di log.

- Il *trasformatore*, successivamente all'acquisto di una, o più, materie prime, effettua delle lavorazioni al fine di ottenere un prodotto trasformato.
  Per procedere con l'inserimento nel catalogo di un prodotto trasformato, il *software*:

  - Richiede lotto e nome del prodotto trasformato, lotti delle materie prime utilizzate, nome e consumo di CO_2 delle lavorazioni;
  - Verifica:
    * La presenza di risorse con lo stesso lotto;
    * L'appartenenza delle materie prime al *trasformatore* in questione;
    * L'effettivo inutilizzo delle materie prime.
  - In caso affermativo, il *software* procede all'inserimento del prodotto trasformato, assegnando un *CarbonFootprint* che tiene conto del contributo delle materie prime utilizzate e delle relative attività per la trasformazione.

- Il *cliente*, per acquistare un prodotto, inserisce il lotto del prodotto trasformato nel *Software* e conferma il pagamento.

Per garantire la completa tracciabilità, e quindi uno storico delle operazioni svolte dagli utenti, si utilizza un file di log.
Per garantire la completa trasparenza, si consente agli utenti di visualizzare le informazioni relative a tutte le risorse presenti nel catalogo.

Durante la fase di progettazione viene valutata la possibilità di autenticare gli attori; non verrà implementata, poiché esula dall'ambito del progetto.

Inoltre, non viene implementata la funzionalità di acquisto, ma il trasferimento delle risorse da chi richiede il trasferimento ad un altro attore.

2.2 Identificazione dei rischi

In questa sezione si definiscono gli *asset*, le rispettive *policy* di sicurezza ed infine le tabelle di Jacobson relative ai casi d'uso degli *asset* identificati.

2.2.1 Identificazione degli asset

A partire dai diagrammi i*, si estraggono i seguenti *asset*:

- Log: contiene le operazioni eseguite dagli utenti;
- Catalogo: contiene tutte le informazioni relative alle materie prime e ai prodotti trasformati; deve essere trasparente e visibile ad ogni attore che ne chiede la visualizzazione;
- Acquisto di un bene: procedura attraverso la quale un trasformatore o un cliente possono acquistare risorse presenti nel catalogo.
- Inserimento materia prima: operazione riservata ai produttori che permette di inserire una materia prima specificando nome, lotto ed il *CarbonFootprint*.
- Inserimento prodotto trasformato: operazione riservata ai trasformatori; consente di inserire un prodotto trasformato specificando lotto e nome del prodotto trasformato, lotti delle materie prime utilizzate, nome e consumo di CO_2 delle lavorazioni;
- Dati personali: dati relativi agli attori registrati.
  Come anticipato, durante la progettazione si considerano come *asset*, ma nel software non si registreranno, poiché si ipotizzerà l'esistenza di una procedura di autenticazione esterna.

2.2.2 Policy di sicurezza e Valutazione degli asset

Asset	Descrizione	Valore	Policy	Impatto
File di Log	All'interno del file sono contenute tutte le operazioni svolte dagli attori (inserimento di informazioni, presa in carico di una trasformazione e acquisto di un prodotto). Una violazione dell'integrità del file di Log comporterebbe problemi violazione di non ripudio e autenticità delle operazioni svolte precedentemente dagli attori	3	Integrity	5
			Availability	5
Catalogo	All'interno del catalogo sono contenuti tutti i prodotti con le corrispondenti informazioni (CO_2, costo etc etc). Una violazione dell'integrità del Catalogo comporterebbe la presenza di prodotti con informazioni non corrette	5	Integrity	5
			Availability	5
Acquisto di un bene	Procedura che consente ad un Cliente o ad un Trasformatore di acquistare un bene dal catalogo, inserendo successivamente l'operazione nel file di Log. Il non ripudio è fondamentale poiché garantisce che sia sempre possibile rintracciare un certo pagamento. La violazione dell'autenticazione ha un alto impatto perché se l'attore non è autenticato non è possibile garantire il non ripudio. La violazione della disponibilità ha un impatto medio-basso poiché non consente a nessun attore l'acquisto; la resilienza ha impatto medio poiché è opportuno che il sistema resista agli attacchi o ai malfunzionamenti	5	Authentication	5
			Non-Repudiation	5
			Authorization	5
			Reliability	2
			Resilience	3
Inserimento Materia Prima	Procedura che consente ad un Produttore di inserire un bene nel catalogo, registrando successivamente l'operazione nel file di Log. Solo un produttore autenticato ed autorizzato deve poter inserire un prodotto, altrimenti è possibile che un attore esterno inserisca un prodotto errato e che un cliente lo acquisti, spendendo denaro, senza però ricevere alcun prodotto dal "falso produttore". Il non ripudio è fondamentale poiché altrimenti non è possibile risalire a chi abbia inserito un prodotto. La violazione della disponibilità ha un impatto medio-basso poiché non consente al produttore di inserire una materia prima; la resilienza ha impatto medio poiché è opportuno che il sistema resista agli attacchi o ai malfunzionamenti.	5	Authentication	4
			Non-Repudiation	5
			Authorization	4
			Reliability	2
			Resilience	3
Inserimento Prodotto Trasformato	Procedura che consente ad un Trasformatore di inserire un bene nel catalogo, registrando successivamente l'operazione nel file di Log. Solo un trasformatore autenticato ed autorizzato deve poter inserire un prodotto, altrimenti è possibile che un attore esterno inserisca un prodotto errato e che un cliente lo acquisti. Il non ripudio è fondamentale poiché altrimenti non è possibile risalire a chi abbia inserito un prodotto. La violazione della disponibilità ha un impatto medio-basso poiché non consente al trasformatore di inserire una materia prima; la resilienza ha impatto medio poiché è opportuno che il sistema resista agli attacchi o ai malfunzionamenti.	5	Authentication	4
			Non-Repudiation	5
			Authorization	4
			Reliability	2
			Resilience	3
Dati personali*	Durante la fase di registrazione sono memorizzati i dati dell'utente. La violazione della confidenzialità compromette il sistema inquanto consentirebbe ad un attore non fidato di compiere operazioni al posto di altri(acquisto di beni da parte di trasformatori/clienti o inserimento di materie prime/prodotti trasformati) (Noi non implementeremo la procedura di registrazione, autenticazione o	3	Confidentiality	5
			Integrity	2

Figura 2.3: Tabella della valutazione del valore e dell'esposizione degli asset.

Per la valutazione si utilizza una scala probabilistica che comprende valori da 1 a 5, con 5 valore massimo.

2.2.3 Casi d'uso

Nelle tabelle successive si mostrano i casi d'uso: essi sono composti da attori coinvolti, una breve descrizione, le risorse coinvolte, le precondizioni, il flusso d'azione, d'eccezione, le post condizioni, e i requisiti non funzionali coinvolti.

Case Type	Use Case	Case ID	CU-04
Case Name	CO-BUY		
Actors	Cliente/Trasformatore e Software		
Description	Un Cliente/Trasformatore chiede al Software la possibilità di acquistare un prodotto		
Data	ID del prodotto, valori di CO2, denaro/NFT		
Stimulus and preconditions	Il Cliente ha chiesto di visualizzare il prodotto con un dato ID (CU-03)		
Basic Flow	Il Cliente chiede al Software di poter comprare il prodotto visualizzato. Il Software verifica la disponibilità della risorsa e chiede al Cliente di inserire i dati relativi al metodo di pagamento**. Il cliente inserisce i dati e conferma il pagamento**. Se la transazione è andata a buon fine il Software inserisce in un file di Log l'operazione effettuata.		
Alternative Flow			
Exception Flow	Il Cliente chiede di acquistare un prodotto non più disponibile. Il Software nega tale procedura.		
Response and Postconditions	Il prodotto viene contrassegnato in modo che non possa essere più acquistato		
Non Functional Requirements	Reliability(affidabilità), Authenticity(autenticità), Accountability(Non Ripudio/Responsabilità)		
Comments			

Figura 2.4: Tabella di Jacobson dell'asset tabella acquisto

Case Type	Use Case	Case ID	CU-02
Case Name	CU-REGISTRATION*		
Actors	Attore generico e Software		
Description	Un attore che non è ancora registrato richiede al Software di potersi iscrivere inserendo i propri dati personali ed una password.		
Data	E-mail, Password, Numero di telefono, CF		
Stimulus and preconditions			
Basic Flow	Il Software fornisce la form di Registrazione. Il Produttore/Trasformatore/Cliente inserisce i dati personali. Il Software verifica la correttezza e inserisce il Produttore/Trasformatore/Cliente all'interno del Database con la corrispondente qualifica.		
Alternative Flow			
Exception Flow	Il Produttore/Trasformatore/Cliente inserisce i dati in modo errato. Il Software restituisce l'errore e consente al Produttore/Trasformatore/Cliente di reinserire i dati		
Response and Postconditions	Il sistema registra l'utente nel DB		
Non Functional Requirements	Reliability(il sistema si comporta nel modo giusto)		
Comments	Il cliente specifica nella form la sua qualifica		

Figura 2.5: Tabella di Jacobson dell'asset tabella registrazione

Case Type	Use Case	Case ID	CU-03
Case Name	CU-VISUALIZE		
Actors	Produttore/Trasformatore/Cliente e Software		
Description	Un attore si autentica* e chiede la visualizzazione dei dati relativi ad un prodotto.		
Data	ID, valori di CO2 e informazioni generali del prodotto		
Stimulus and preconditions			
Basic Flow	Il Produttore/Trasformatore/Cliente chiede al software di autenticarsi*. Il Produttore/Trasformatore/Cliente chiede al software di poter visualizzare un prodotto con un ID specifico. Il Software fornisce le informazioni richieste.		
Alternative Flow	1)Se l'utente è un Cliente/Trasformatore, allora il Software consente la possibilità di acquistare il prodotto; nel caso in cui il Cliente/Trasformatore scelga tale opzione, inizia il processo descritto nel caso d'uso CU-BUY		
Exception Flow	Il Produttore/Trasformatore/Cliente inserisce un ID non presente oppure inserisce credenziali errate*. Il Software restituisce l'errore e consente al Produttore/Trasformatore/Cliente di inserire un nuovo ID o reinserire le sue credenziali		
Response and Postconditions			
Non Functional Requirements	Reliability(Disponibilità)		
Comments			

Figura 2.6: Tabella di Jacobson dell'asset tabella visualizzazione

Case Type	Use Case	Case ID	CU-01
Case Name	CU-NEW_PRODUCT		
Actors	Produttore e Software		
Description	Il Produttore chiede al Software l'inserimento di una materia prima nel Catalogo.		
Data	Impronta CO2 calcolata dal Produttore, costo e nome del prodotto		
Stimulus and preconditions	Il produttore deve aver prodotto la materia prima e deve aver calcolato il corrispondente valore di CO2		
Basic Flow	Il Produttore chiede l'accesso al Software e si autentica*. Il Produttore inserisce il nome del prodotto, i valori di CO2 precedentemente calcolati ed il prezzo. Il Produttore conferma i valori inseriti. Il Software inserisce il prodotto nel catalogo e fa il Log dell'operazione.		
Alternative Flow			
Exception Flow	1)Il Produttore inserisce credenziali errate* Il Software restituisce l'errore e consente al Produttore di reinserire i dati 2)Il Produttore inserisce valori non ammissibili. Il Software restituisce l'errore e consente al Produttore di reinserire i valori. 3)Il Software non riceve i dati a causa di problemi esterni, perciò restituisce l'errore e consente al produttore di reinserire i dati.		
Response and Postconditions	Il sistema da conferma dalla transazione ottenuta		
Non Functional Requirements	Integrity(integrità), Accountability(Non Ripudio /Responsabilità), Reliability(affidabilità), Authenticity(autenticità)		
Comments			

Figura 2.7: Tabella di Jacobson dell'asset tabella prodotto

Case Type	Use Case	Case ID	CU-05
Case Name	CU-NEW_TRANSFORMED_PRODUCT		
Actors	Trasformatore e Software		
Description	Il Trasformatore chiede al Software l'inserimento di un nuovo prodotto trasformato nel Catalogo.		
Data	Impronta CO2 calcolata dal Trasformatore, costo e nome del prodotto		
Stimulus and preconditions	Il Trasformatore deve aver trasformato il prodotto e deve aver calcolato il corrispondente valore di CO2		
Basic Flow	Il Trasformatore chiede l'accesso al Software e si autentica*. Il Trasformatore inserisce il nome del prodotto, i valori di CO2 precedentemente calcolati, il prezzo, gli id delle materie prime che sono state usate per la trasformazione. Il Software conferma che le materie prime sono state comprate precedentemente dal Trasformatore e calcola i valori di CO2 complessivi. Il Software inserisce il prodotto nel catalogo e fa il Log dell'operazione.		
Alternative Flow			
Exception Flow	1)Il Trasformatore inserisce credenziali errate* Il Software restituisce l'errore e consente al Produttore di reinserire i dati 2)Il Trasformatore inserisce valori non ammissibili. Il Software restituisce l'errore e consente al Trasformatore di reinserire i valori. 3) Il Trasformatore inserisce id di materie prime che non ha comprato Il Software restituisce l'errore e consente al Trasformatore di reinserire i valori. 4)Il Software non riceve i dati a causa di problemi esterni, perciò restituisce l'errore e consente al produttore di reinserire i dati.		
Response and Postconditions	Il sistema da conferma di trasizione avvenuta		
Non Functional Requirements	Integrity(integrità), Accountability(Non Ripudio /Responsabilità), Reliability(affidabilità), Authenticity(autenticità)		
Comments			

Figura 2.8: Tabella di Jacobson dell'asset tabella prodotto trasformato

2.3 Identificazione delle minacce

La tabella sottostante si ottiene a partire dal modello **Dua-Stride**.

THREAT IDENTIFICATION									
property violated	*Authentication*	*Integrity*	*Non-Repudiation*	*Confidentiality*	*Availability*	*Authorization*	*Safety*	*Realiability*	*Resilience*
Asset	Spoofing	Tampering	Repudiation	Information disclosure	DOS	Elevation of privilege	Danger	Unreliability	Absence of Resilience
FILE DI LOG		X			X				
CATALOGO		X			X				
ACQUISTO DI UN BENE	X		X			X		X	X
INSERIMENTO MATERIA PRIMA	X		X			X		X	X
INSERIMENTO PRODOTTO TRASFORMATO	X		X			X		X	X
DATI PERSONALI		X		X					

Figura 2.9: Tabella delle minacce

In particolare:

- **S**poofing: violazione dell'autenticazione;
- **T**ampering: violazione di integrità;
- **R**epudiation: violazione del non-ripudio;
- **I**nformation Disclosure: violazione della confidenzialità;
- **D**oS: (Denial of Service) violazione della disponibilità;
- **E**levation of Privilege: violazione dell'autorizzazione;
- **D**anger: violazione della sicurezza;
- **U**nrealibility; violazione dell'affidabilità;
- **A**bsence of Resilience: violazione della resilienza.

2.4 Identificazione degli attacchi

La fase di identificazione degli attacchi si sviluppa tramite l'analisi dei casi di abuso, misuso e degli alberi di attacco.

2.4.1 Albero degli attacchi

Di seguito, nella figura 2.10, si riportano i possibili attacchi che possono colpire il sistema; l'identificazione è stata guidata dalla consultazione dei cataloghi Att&ck e Capec.
La semantica utilizzata è la stessa dei diagrammi i* presentati nei capitoli precedenti.

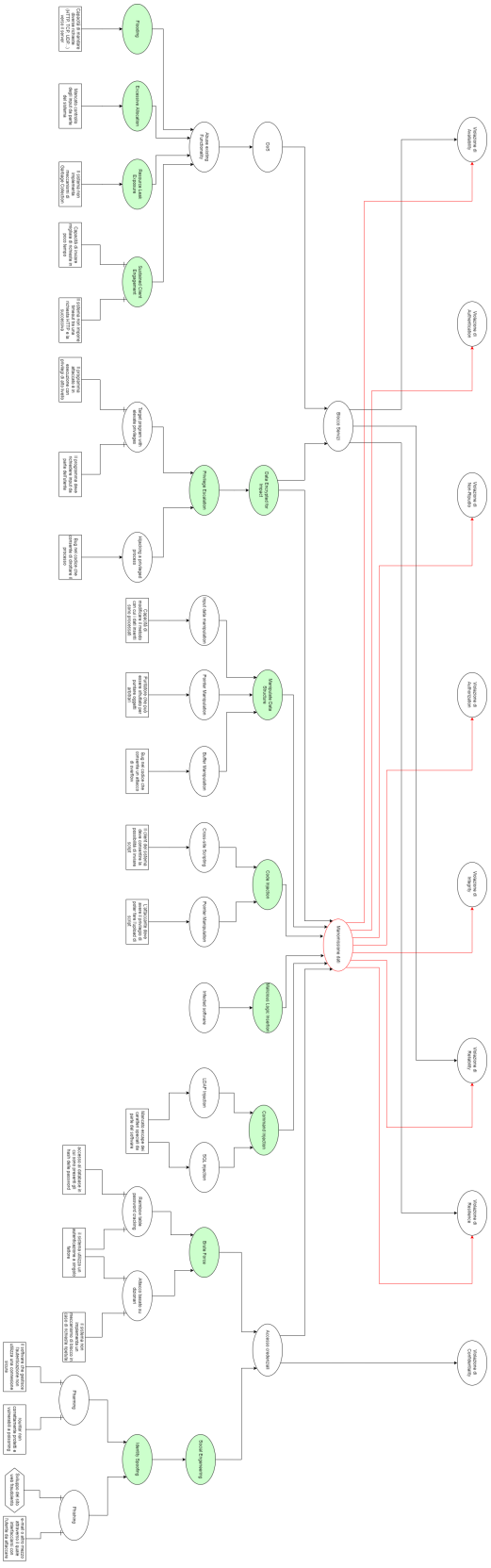

Figura 2.10: Albero degli attacchi

2.4.2 Casi di Abuso

Case Type	Abuse Case	Case ID	AT-01
Case Name	Flooding		
Actors	Software, Attaccante		
Description	L'attore malintenzionato accede al sistema da più dispositivi, per poi effettuare simultaneamente numerose richieste tramite una delle interfacce fornite dal sistema, in modo da saturare la capacità di risposta e rendendo impossibile l'accesso al sistema per gli altri utenti.		
Data	Catalogo, file di log		
Stimulus and preconditions	L'attaccante deve avere la possibilità di sfruttare più macchine contemporaneamente per portare l'attacco.		
Attack Flow 1	L'attaccante esplora il sistema manualmente o tramite tool automatizzati. Successivamente invia un quantitativo di richieste superiore alla capacità di elaborazione del sistema, in modo da saturarlo.		
Response and Postconditions	Il sistema si blocca rendendo i servizi non disponibili.		
Non Functional	Availability (disponibilità), Resilience (resilienza), Reliability		
Mitigations	Assicurarsi che i protocolli abbiano limiti di scala specifici configurati; Specificare quali comportamenti sono accettabili quando l'allocazione delle risorse raggiunge i limiti; Utilizzo rete distribuita; Ridondanza.		
Comments			

Figura 2.11: Flooding

Case Type	Abuse Case	Case ID	AT-02
Case Name	Excessive Allocation		
Actors	Software, Attaccante		
Description	Un attaccante fa in modo che il sistema allochi risorse eccessive per soddisfare la sua richiesta, riducendo così le risorse disponibile per i servizi legittimi, in maniera tale da renderli non disponibili.		
Data	Parametri della form inviata dall'attaccante		
Stimulus and preconditions	L'attaccante deve essere capace di controllare l'allocazione delle risorse associate alla sua richiesta		
Attack Flow 1	L'attaccante interagisce con il sistema inviando una o più richieste di allocazione di variabili, comportando un'allocazione eccessiva di risorse. Questo attacco può prende di mira le procedure che necessitano di molto tempo per essere eseguite, o allocano molto		
Response and Postconditions	Il servizio non è più disponibile, il sistema va in crash.		
Non Functional Requirements	Availability (disponibilità), Resilience (resilienza), Reliability (Affidabilità)		
Mitigations	Limitare la quantità di risorse disponibili per utenti non privilegiati; Sanificazione degli input; Utilizzo rete distribuita; Diversità.		
Comments			

Figura 2.12: Excessive Allocation

Case Type	Abuse Case	Case ID	AT-03
Case Name	Resource Leak Exposure		
Actors	Software, Attaccante		
Description	Un attaccante sfrutta una perdita di risorse da parte del sistema, per esaurire la quantità di risorse disponibili per soddisfare le richieste degli utenti legittimi.		
Data	Parametri della form inviata dall'attaccante		
Stimulus and preconditions	Il sistema deve avere una perdita di risorse che l'attaccante può sfruttare ripetutamente.		
Attack Flow 1	L'attaccante invia un gran numero di richieste al sistema, utilizzando un'interfaccia che attiva una procedura, che non prevvede deallocazione delle risorse utilizzate. In tal modo, all'aumentare delle richieste, il sistema saturerà la memoria disponibile.		
Response and Postconditions	Il servizio non è più disponibile, il sistema va in crash.		
Non Functional Requirements	Availability (disponibilità), Resilience (resilienza), Reliability (Affidabilità)		
Mitigations	Utilizzare linguaggi di programmazione che prevedono la gestione della memoria tramite Garbage Collector. Utilizzo rete distribuita;		
Comments			

Figura 2.13: Resource Leak Exposure

Case Type	Abuse Case	Case ID	AT-04
Case Name	Sustained Client Engagement		
Actors	Software, Attaccante		
Description	Un attaccante tenta di negare la disponibilità dei servizi mediante richieste che richiedono tempi di processamento molto elevati		
Data	Servizi del sistema.		
Stimulus and preconditions	L'attaccante ha necessità di uno script o programma capace di coinvolgere continuamente l'obiettivo e mantenere prolungato l'utilizzo di una specifica risorsa. Può essere necessario coinvolgere un network per aumentare il numero di richieste.		
Attack Flow 1	L'attaccante esplora il sistema manualmente o tramite tool automatizzati, ed individua un'interfaccia che attivi una procedura che richieda un tempo di processamento molto elevato. Successivamente invia le richieste in maniera tale da bloccare l'accesso ad altri utenti		
Response and Postconditions	Il servizio non è più disponibile per tutta la durata dell'attacco.		
Non Functional Requirements	Availability (disponibilità), Resilience (resilienza), Reliability (Affidabilità)		
Mitigations	Richiesta di login univoco per ogni richiesta di risorsa; Limitazione per un unico accesso agli indirizzi IP; Utilizzo rete distribuita; Ridondanza.		
Comments			

Figura 2.14: Sustained Client Engagement

Case Type	Abuse Case	Case ID	AT-05
Case Name	Privilege Escalation		
Actors	Software, Attaccante		
Description	Un avversario usa una debolezza che gli consente di elevare il proprio privilegio ed eseguire un'azione che non dovrebbe essere autorizzato a compiere.		
Data	Dati personali amministratore		
Stimulus and preconditions	Il sistema deve contenere vulnerabilità nel codice o nel sistema operativo		
Attack Flow 1	L'attaccante ispeziona il sistema manualmente o con tool automaticizzati, in cerca di debolezze; successivamente utilizza le vulnerabilità per poter ottenere privilegi di alto livello sfruttando le vulnerabilità del software o del sistema operativo.		
Response and Postconditions	L'attaccante dispone dei privilegi di amministratore/root ed ha la possibilità di alterare il contenuto del file di log, o ha la possibilità di aggiungere nuovi produttoi/consumatori o clienti.		
Non Functional Requirements	Confidentiality (confidenzialità), Authorization(autorizzazione), Integrity (integrità), Non-Repudiation (non ripudio)		
Stimulus and preconditions	Isolamento dell'applicazione (Sandboxing); controllo delle richieste; utilizzo di tool di analisi statica del codice per prevenire bug e code-		
Comments			

Figura 2.15: Privilege Escalation

Case Type	Abuse Case	Case ID	AT-06
Case Name	Data Encrypted for Impact		
Actors	Software, Attaccante		
Description	L'attaccante, dopo aver ottenuto privilegi di sistema, cripta i dati per poi chiedere un riscatto.		
Data	Dati catalogo e file di log		
Stimulus and preconditions	L'attaccante deve aver svolto una fase preliminare di escalation dei privilegi di sistema, in modo da avere accesso ai file che verranno		
Attack Flow 1	L'attaccante sfrutta i privilegi precedentemente ottenuti per cifrare i file relativi al catalogo ed al file di log, per poi chiedere un riscatto.		
Response and Postconditions	I dati del sistema risultano essere cifrati e quindi inaccessibili.		
Non Functional Requirements	Availability (disponibilità), Reliability (Fidatezza), Integrity (integrità), Non-Repudiation (non ripudio)		
Mitigations	Backup dei dati; Monitoraggio dei processi;		
Comments			

Figura 2.16: Data Encrypted for Impact

Case Type	Abuse Case	Case ID	AT-07
Case Name	Manipulate data Structure		
Actors	Software, Attaccante		
Description	L'attaccante sfrutta le debolezze relative alle strutture dati utilizzate dal sistema, in modo da modificare i dati.		
Data	Dati presenti nella memoria		
Stimulus and preconditions	Il sistema deve contenere vulnerabilità che consentono la manipolazione dei puntatori, delle strutture dati o del buffer		
Attack Flow 1	L'attaccante sfrutta una debolezza nella validazione degli input, oppure una debolezza introdotta dall'utilizzo di strutture dati dinamiche e/o puntatori. Successivamente l'attaccante riesce a manipolare le strutture dati, ottenendo la possibilità di modificare i dati in memoria		
Response and Postconditions	I dati nelle strutture non sono più considerati attendibili.		
Non Functional Requirements	Integrity (integrità), Authentication (autenticazione), Authorization (autorizzazione), Non-Repudiation (non ripudio)		
Mitigations	Utilizzo di un linguaggio di programmazione che evitino il buffer overflow; Sanificazione degli input; Adozione delle buone pratiche di programmazione.		
Comments			

Figura 2.17: Manipulate Data Structure

Case Type	Abuse Case	Case ID	AT-08
Case Name	Code Injection		
Actors	Software, Attaccante		
Description	Un avversario sfrutta una debolezza nella validazione dell'input sul bersaglio per iniettare nuovo codice in quello che è attualmente in esecuzione		
Data	Codice e dati del sistema		
Stimulus and preconditions	Mancata sanificazione degli input, possibilità per l'attaccante di fare deploy degli script		
Attack Flow 1	L'attaccante sfrutta le debolezze nelle validazioni degli input per inserire codice malevolo, che verrà mandato in esecuzione per modificare il comportamento del sistema; ciò può comportare una modifica dei dati nel catalogo o nei file di log i		
Response and Postconditions	Il sistema è compromesso.		
Non Functional Requirements	Integrity (integrità) , Availability (disponibilità), Authorization (autorizzazione), Non-Ripudiation (non ripudio)		
Mitigations	Sanificazione e monitoraggio degli input		
Comments			

Figura 2.18: Code Injection

Case Type	Abuse Case	Case ID	AT-09
Case Name	Command Injection		
Actors	Software, Attaccante		
Description	L'attaccante cerca di eseguire un comando a sua scelta o inserisce nuovi elementi in un comando già esistente, modificando così il funzionamento previsto dal sistema.		
Data	Dati nel sistema		
Stimulus and preconditions	L'applicazione di destinazione deve accettare l'input dell'utente e deve utilizzarlo nella costruzione dei comandi da eseguire. Il software non implementa un processo di escape dei caratteri speciali		
Attack Flow 1	L'attaccante sfrutta le debolezze nelle validazioni degli input per inserire comandi malevoli, in modo tale da modificare il comportamento del sistema. Ciò comporta una possibile modifica dei dati nel catalogo, o la possibiltà da parte dell'attaccante di bypassare il meccanismo di autenticazione.		
Response and Postconditions	Accesso a funzionalità riservate ad utenti autorizzati, possibilità di modificare il catalogo		
Non Functional Requirements	Integrity (integrità) , Availability (disponibilità), Confidentiality(Riservatezza), Authorization (autorizzazione),		
Mitigations	Sanificazione e monitoraggio degli input		
Comments			

Figura 2.19: Command Injection

Case Type	Abuse Case	Case ID	AT-10
Case Name	Malicious Logic Insertion		
Actors	Software, Attaccante		
Description	L'attaccante installa un malware in una componente del sistema.		
Data	Dati del sistema		
Stimulus and preconditions	L'attaccante deve poter accedere al software o all'hardware		
Attack Flow 1	L'attaccante sfrutta le debolezze software per aggiungere logica dannosa. Per poter accedere al software l'attaccante può utilizzare tecniche di ingegneria del software per spingere un utente a scaricare un file infetto, oppure può utilizzare hardware infetti (ad esempio una pennetta USB)		
Response and Postconditions	Esecuzione di comandi non autorizzati		
Non Functional Requirements	Authorization (Autorizzazione)		
Mitigations	Gestione dei privilegi e utilizzo di un antivirus		
Comments			

Figura 2.20: Malicious Logic Insertion

Case Type	Abuse Case	Case ID	AT-11
Case Name	Brute Force		
Actors	Software, Attaccante		
Description	L'attaccante cerca di rubare le credenziali di un utente autorizzato per poter svolgere operazioni contro il sistema		
Data	Password degli utenti		
Stimulus and preconditions	Il sistema deve fornire informazioni relative alle credenziali errate (e-mail non corretta o errore sulla password)		
Attack Flow 1	L'attaccante inizialmente cerca di ottenere un'email di un utente autorizzato, successivamente cerca di ottenere la corrispondente password utilizzando un dizionario. Se l'attacco va a buon segno l'attaccante ha la possibilità di bypassare il meccanismo di autenticazione, svolgendo operazioni sul catalogo		
Response and Postconditions	L'attaccante può svolgere le operazioni di un utente autorizzato		
Non Functional Requirements	Confidentiality , Authorization, Authentication, Non-Ripudiation		
Mitigations	Utilizzare un autenticazione multifattore e introdurre policy di accettabilità delle password(numero minimo di caratteri; presenza di numeri, caratteri speciali etc etc)		
Comments			

Figura 2.21: Brute Force

Case Type	Abuse Case	Case ID	AT-12
Case Name	Identify Spoofing		
Actors	Software , Attaccante		
Description	L'attaccante cerca di prendere l'identità di uno degli utenti registrati per poter svolgere azioni illecite o dannose per il sistema.		
Data	Dati Prodotti		
Stimulus and preconditions	L'attaccante ha accesso a credenziali di un utente autorizzato.		
Attack Flow 1	L'attaccante sfrutta un bug nella procedura di autenticazione per trafugare le credenziali di un utente. Successivamente procede a svolgere operazioni per conto dell utente autorizzato. In tal modo ha la possibiltà di modificare i dati nel catalogo.		
Response and Postconditions	L'attaccante ha svolto operazioni per conto di un utente autorizzato		
Non Functional Requirements	Confidentiality(Riservatezza), Integrity(Integrità), Authentication(Autenticazione), Non-Ripudiation (non ripudio)		
Mitigations	Utilizzare un autenticazione multifattore e introdurre policy di accettabilità delle password(numero minimo di caratteri; presenza di numeri, caratteri speciali etc etc)		
Comments			

Figura 2.22: Identify Spoofing

Case Type	Abuse Case	Case ID	AT-13
Case Name	Social Engeneering		
Actors	Utenti, Attaccante		
Description	L'attaccante adotta una serie di tecniche ingannevoli ai danni degli utenti, per indurli a rivelare informazioni riservate.		
Data	Informazioni riservate degli utenti		
Stimulus and preconditions	L'attaccante deve avere un canale di comunicazione aperto con l'utente		
Attack Flow 1	L'attaccante sfrutta i canali di comunicazione in possesso verso gli utenti per indurli a rivelare informazioni sensibili. Ciò può essere svolto mandando e-mail o link assumendo l'identità di un'autorità conosciuta		
Response and Postconditions	L'attaccante ottiene informazioni sensibili		
Non Functional Requirements	Confidentiality(Riservatezza)		
Mitigations	Utilizzare Antivirus e filtri anti-spam		
Comments			

Figura 2.23: Social Engeneering

2.4.3 Casi di Misuso

Case Type	Caso di misuso	Case ID	CMU-01
Case Name	CMU-WRONG_INPUT		
Actors	Trasformatore/Produttore/Consumatore e Software		
Description	Il Trasformatore/Produttore/Consumatore inserisce un input dannoso nella form che gli viene fornita dal server		
Data	Dati della form		
Stimulus and Preconditions	Il Software ha fornito la form al Trasformatore/Produttore/Consumatore		
Basic Flow	Il Trasformatore/Produttore/Consumatore inserisce un input errato nella form che gli viene fornita dal server. Il sistema non rileva l'errore, ma va in crash		
Alternative Flow			
Response and Postconditions	Il sistema è compromesso		
Non functional requirements	Reliability(affidabilità), Availability(disponibilità)		
Comments	Per evitare problematiche di questo tipo, sarà necessario introdurre dei controlli sugli input (sanificazione degli input)		

Figura 2.24: Caso di misuso: inserimento di un input dannoso

Case Type	Caso di misuso	Case ID	CMU-02
Case Name	CMU-INVALID_GOODS_DATA		
Actors	Trasformatore/Produttore e Software		
Description	Il Trasformatore/Produttore inserisce una merce con dei dati errati.		
Data	Dati della merce		
Stimulus and Preconditions	Il Software ha fornito la form al Trasformatore/Produttore per inserire la merce nel catalogo		
Basic Flow	Il Trasformatore/Produttore inserisce nella form dei dati errati relativi alla merce. Il sistema inserisce la merce nel catalogo.		
Alternative Flow			
Response and Postconditions	Il catalogo non è integro		
Non functional requirements	Integrity(integrità)		
Comments	Questo caso di misuso è problematico, poiché uno dei requisiti del sistema è che le operazioni svolte siano immutabili, perciò non è possibile modificare i dati di quella merce; perciò è solo possibile reinserire una nuova transazione che annulli gli effetti della precedente, previa consenso degli altri attori.		

Figura 2.25: Caso di misuso: inserimento di una risorsa con valori errati

2.4.4 Valutazione degli attacchi

La valutazione degli attacchi relativi agli *asset* consiste nel moltiplicare:

- La probabilità che un attacco violi una *policy* dell' *asset*, espressa tramite una scala a cinque valori;
- L'impatto derivante dalla violazione della policy stessa.

Si ottiene così il rischio **inerente**.

Project					
Year		Team ID		Team	
Sprint		Start		End	

Asset	Value	Spoofing	Tampering	Repudiation	Information disclosure	DOS	Elevation of privilege	Danger	Unreliability	Absence of Resilience	Exposure	Attack	Inherent Probability	Inherent Risk
FILE DI LOG	5		X								5	data encrypted for impact	5	25
			X											
			X											
			X									privilege escalation	3	15
			X											
			X									manipulate data structure	5	25
			X											
			X									code injection	4	20
			X											
			X									command injection	3	15
			X									identify spoofing	3	15
			X											
						X					5	Flooding	4	20
						X								
						X								
						X						Excessive Allocation	3	15
						X								
						X								
						X								
						X						Resource Leak Exposure	3	15
						X								
						X								
						X						Sustained Client Engagment	4	20
						X								

						X								
						X						DATA ENCRYPTED FOR IMPACT	5	25
						X								
						X								
			X											
			X									DATA ENCRYPTED FOR IMPACT	5	25
			X											
			X									PRIVILEGE ESCALATION	3	15
			X											
			X									MANIPULATE DATA STRUCTURE	5	25
			X								5			
			X									CODE INJECTION	4	20
			X											
			X									COMMAND INJECTION	3	15
			X									IDENTIFY SPOOFING	3	15
			X											
						X								
CATALOGO	5					X						Flooding	4	20
						X								
						X								
						X								
						X						Excessive Allocation	3	15
						X								
						X								
						X								
						X						Resource Leak Exposure	3	15
						X					5			
						X								
						X								
						X						Sustained Client Engagment	4	20
						X								
						X								
						X								
						X						DATA ENCRYPTED FOR IMPACT	5	25
						X								
		X										Brute-Force	4	20

ACQUISTO DI UN BENE	5	X									5	Brute-Force	4	20
		X										Identify Spoonfing	3	15
		X										Manipulate Data Structure	5	25
		X										Command Injection	3	15
		X												
				X							5	Privilege escalation	3	15
				X										
				X								Identity Spoofing	3	15
				X										
				X								Manipulate Data Structure	5	25
				X										
				X								Command Injection	3	15
				X								Brute-Force	4	20
				X										
							X				5	Privilege escalation	3	15
							X							
							X					Brute-Force	4	20
							X							
							X					malicious logic insertion	3	15
							X					Command Injection	3	15
							X							
							X					Code Injection	4	20
							X							
							X					Manipulate Data Structure	5	25
							X							
									X			Data encrypted for impact	5	10
									X					
									X					
									X			Flooding	4	8
									X					
									X			Excessive		

								X		2	Excessive Allocation	3	6
								X					
								X			Resource Leak Exposure	3	6
								X					
								X			Sustained Client Engagement	4	8
								X					
									X	3	Flooding	4	12
									X				
									X				
									X		Excessive Allocation	3	9
									X				
									X				
									X				
									X		Resource Leak Exposure	3	9
									X				
									X				
									X		Sustained Client Engagment	4	12
									X				
									X				
									X		Data encrypted for impact	5	15
									X				
		X								4	Brute-Force	4	16
		X											
		X									Identify Spoonfing	3	12
		X											
		X									Manipulate Data Structure	5	20
		X											
		X									Command Injection	3	12
				X							Privilege escalation	3	15
				X									
				X							Identity Spoofing	3	15

				X								Identity Spoofing	5	15
				X								Manipulate Data Structure	5	25
				X										
				X								Command Injection	3	15
				X								Brute-Force	4	20
				X										
							X					Privilege escalation	3	15
							X							
							X					Brute-Force	4	20
							X							
							X					malicious logic insertion	3	15
							X					Command Injection	3	15
							X					Code Injection	4	20
							X							
							X					Manipulate Data Structure	5	25
							X							
INSERIMENTO MATERIA PRIMA	5								X			Data encrypted for impact	5	10
									X					
									X					
									X			Flooding	4	8
									X					
									X					
									X			Excessive Allocation	3	6
									X					
									X					
									X			Resource Leak Exposure	3	6
									X					
									X					
									X			Sustained Client Engagement	4	8
									X					
									X					

									X				
									X		Flooding	4	12
									X				
									X				
									X		Excessive Allocation	3	9
									X				
									X				
									X				
									X				
									X	3	Sustained Client Engagement	4	12
									X				
									X				
									X				
									X		Data encrypted for impact	5	15
									X				
									X				
									X		Resource Leak Exposure	3	9
									X				
		X									Brute-Force	4	16
		X											
		X									Identify Spoonfing	3	12
		X								4			
		X									Manipulate Data Structure	5	20
		X											
		X									Command Injection	3	12
				X							Privilege escalation	3	15
				X									
				X							Identity Spoofing	3	15
				X									
				X							Manipulate Data Structure	5	25
				X						5			
				X							Command Injection	3	15
				X							Brute-Force	4	20
				X									

INSERIMENTO PRODOTTO TRASFORMATO	5												
							X				4	Privilege escalation	3	15
							X							
							X					Brute-Force	4	20
							X							
							X					malicious logic insertion	3	15
							X					Command Injection	3	15
							X					Code Injection	4	20
							X							
							X					Manipulate Data Structure	5	25
							X							
									X		2	Data encrypted for impact	5	10
									X					
									X			Flooding	4	8
									X					
									X			Excessive Allocation	3	6
									X					
									X			Resource Leak Exposure	3	6
									X					
									X			Sustained Client Engagement	4	8
									X					
										X		Flooding	4	12
										X				
										X		Excessive Allocation	3	9
										X				

										X				
										X				
										X	3	Sustained Client Engagement	4	12
										X				
										X				
										X		Data encrypted for impact	5	15
										X				
										X		Resource Leak Exposure	3	9
										X				
DATI PERSONALI	3				X						5	Privilege Escalation	3	15
					X									
					X							Command Injection	3	15
					X							Brute-Force	5	25
					X									
					X							Identify Spoofing	3	15
					X							Social engineering	3	15
			x								2	Data encrypted for impact	5	10
			X											
			X											
			x									Privilege escalation	3	6
			X											
			x									Manipulate Data Structure	5	10
			X											
			X									Code Injection	4	8
			X											
			X									Command Injection	3	6
			X									Identify Spoofing	3	6
			X											

2.5 Identificazione, valutazione e scelta delle misure di controllo

In questa sezione si identificano e classificano alcune possibili misure di controllo, con l'obiettivo di fronteggiare gli attacchi definiti precedentemente.
Attraverso il catalogo CAPEC, si ottengono alcune soluzioni per la mitigazione degli attacchi, che si mostrano di seguito:

- Backup dei dati: meccanismo di backup giornaliero, in maniera tale da evitare malfunzionamenti o attacchi che violino l'integrità dei dati;
- Sistema Distribuito: sistema *peer-to-peer* formato da più nodi, in grado di resistere a malfunzionamenti e attacchi DoS;
- Monitoraggio: sistema di monitoraggio del file di log, per la rilevazione di possibili azioni riconducibili a tentativi di attacco;
- ACLs: lista di gestione dei permessi degli utenti, in maniera tale da evitare accessi non autorizzati;
- Linguaggio di alto livello: linguaggi che diminuiscono la probabilità di avere *bug* o *code-smell* nel codice, come Java, Python, Javascript etc. ;
- Autenticazione a due fattori: utilizzo di due canali di autenticazione, in maniera tale da mitigare i furti di credenziali;
- Antivirus: software di individuazione dei possibili file infetti e bloccare le spam-mail destinate ad attori che interagiscono con il sistema;
- Blocco IP: meccanismo di blocco delle richieste in arrivo da uno specifico indirizzo IP;
- User Block: meccanismo di blocco delle richieste di autenticazione al server, nel caso in cui si inoltrino numerose richieste con esito negativo;

Per la valutazione e la classificazione delle misure di controllo si utilizzano due indici:

- Rischio residuo: si stimano la probabilità e l'impatto residui, per poi calcolarne il prodotto;
- Value-to-Cost: si calcola il rapporto tra il valore dell'asset ed il costo di implementazione della misura di controllo;

A partire dal *Value-to-Cost* le misure di controllo sono classificate in maniera qualitativa utilizzando la matrice definita nella figura 2.26.

Di seguito sono definite le fasce di valutazione:

- Alto: maggiore o uguale di 2;

- Medio: tra 0.8 e 1.66;

- Basso: tra 0.2 e 0.75.

Figura 2.26: Matrice value-to-cost

Di seguito è riportata la tabella di valutazione complessiva in cui è presente il rapporto *Value-to-Cost* per ogni soluzione analizzata:

ASSET	VALORE ASSET	MISURA DI CONTROLLO	COSTO	VALUE TO COST RATIO
file di log	3	Backup dei dati	3	1
		Sistema distribuito	5	0,6
		Monitoraggio	3	1
		ACLs	2	1,5
		livello	1	3
		Allow list	1	3
		Autenticazione a due fattori	2	1,5
		Antivirus	4	0,75
		Blocco IP	2	1,5
catalogo	5	Sistema distribuito	5	1
		Monitoraggio	3	1,666666667
		ACLs	2	2,5
		Utilizzo linguaggio di alto livello	1	5
		Allow list	1	5
		Autenticazione a due fattori	2	2,5
		Antivirus	4	1,25
		Blocco IP	2	2,5
acquisto di un bene	5	Autenticazione a due fattori	2	2,5
		User Block	1	5
		Antivirus	4	1,25
		Linguaggi di alto livello	1	5
		Monitoraggio	3	1,666666667
		ACLs	2	2,5
		Allow list	1	5
		Backup dei dati	3	1,666666667
		Sistema distribuito	5	1
		Blocco IP	2	2,5
inserimento materia prima	5	Autenticazione a due fattori	2	2,5
		User Block	1	5
		Antivirus	4	1,25
		Linguaggi di alto livello	1	5
		Monitoraggio	3	1,666666667
		ACLs	2	2,5
		Allow list	1	5
		Backup dei dati	3	1,666666667
		Sistema distribuito	5	1
		Blocco IP	2	2,5
inserimento prodotto trasformato	5	Autenticazione a due fattori	2	2,5
		User Block	1	5
		Antivirus	4	1,25
		Linguaggi di alto livello	1	5
		Monitoraggio	3	1,666666667
		ACLs	2	2,5
		Allow list	1	5
		Backup dei dati	3	1,666666667
		Sistema distribuito	5	1
		Blocco IP	2	2,5
Dati personali	3	Backup dei dati	3	1
		Sistema distribuito	5	0,6
		Monitoraggio	3	1
		ACLs	2	1,5
		Linguaggi di alto livello	1	3
		Allow list	1	3
		Autenticazione a due fattori	2	1,5

Figura 2.27: Tabella di valutazione complessiva per gli asset

Capitolo 3

Progettazione

3.1 Scelta tecnologie e linee guida progettazione sicura

Alla luce dell'analisi degli attacchi e della valutazione delle misure di controllo, si è deciso di implementare alcune delle tecnologie elencate precedentemente.

3.1.1 Blockchain

In particolare, il sistema utilizzerà le *blockchain*, poiché:

- Implementano distribuzione e ridondanza, aumentando l'affidabilità e la resilienza del sistema per evitare i *single point of failure* dei requisiti di Sommerville;

- Utilizzando una versione privata, è possibile semplificare il meccanismo di autenticazione impedendo l'accesso agli utenti sprovvisti di chiave.
  La chiave si genera in maniera pseudocasuale con una lunghezza tale da mitigare attacchi di *brute force*; il tutto contribuisce a garantire un solido meccanismo di autenticazione e rendere superflue strategie di autenticazione a più fattori.

- Implementano nativamente meccanismi di log, poiché le transazioni sono automaticamente scritte all'interno dei blocchi rendendo tracciabile ogni operazione; in tal modo si ha la possibilità di ripristinare il funzionamento del sistema in caso di malfunzionamenti, poiché lo stato della rete viene salvato.

L'utilizzo della *blockchain* introduce nuove problematiche implementative, di seguito sono riportate le più rilevanti:

- Consenso: ogni attore autenticato all'interno del sistema dovrebbe gestire un proprio nodo; in caso contrario, l'attore con più nodi potrebbe condurre un attacco a maggioranza qualificata (66%+1) , invalidando le transazioni a suo favore;

- Costi: per avere maggiore affidabilità è necessaria la presenza di un gran numero di nodi: ciò comporta sia un notevole investimento per l'acquisto delle macchine che un elevato consumo di corrente elettrica durante la validazione delle transazioni;

- Oracolo: i dati contenuti all'interno della blockchain sono immutabili e integri, come da specifiche; nulla però assicura che i dati inseriti dagli attori rispecchino i reali consumi di CO_2 delle risorse inserite.
  Questa problematica non viene risolta in questa sede: la soluzione consiste in un accordo tra le aziende riunite nel consorzio al fine di garantire ispezioni periodiche per la verifica dei dati immessi;

- Attacchi DoS: nonostante la distribuzione e la ridondanza, le *blockchain* private sono vulnerabili a questa tipologia di attacchi, poiché i nodi lavorano contemporaneamente per la validazione delle transazioni e necessitano di maggiori risorse rispetto ad un database tradizionale.
  I potenziali attaccanti risultano essere i clienti, i quali non spendono denaro, o token, per l'invio delle numerose richieste.

3.1.2 Altre Strategie di mitigazione

Oltre alla *blockchain*, si adottano:

- Un *software antivirus* per la presenza di eventuali file infetti;

- Un filtro *antispam* per le email che potrebbero indurre gli attori a fornire le proprie credenziali;

- Un *firewall* per il filtraggio dei pacchetti, in maniera tale da bloccare le richieste potenzialmente dannose per il sistema ed i tentativi di *flooding* che renderebbero il software inutilizzabile;

Relativamente ai linguaggi, si adottano:

- Solidity, per la scrittura degli *smart-contracts*, il quale:
  - fornisce primitive di alto livello;
  - fornisce meccanismi di verifica per le proprietà di *Safety* grazie al *Solidity Model Checker*;

  - consente, al verificarsi di errori durante le transazioni, di invalidarle e mantenere inalterato lo stato della *blockchain*, rispettando il *fail-safe default*.

- Node JS, per la realizzazione del *front-end* e parte del *back-end*.

In particolare, si implementano meccanismi di:

- Monitoraggio degli *input* esterni, in maniera tale da evitare iniezioni di codice o comandi;
- Controllo della tipologia dell'utente autenticato, in maniera tale da permettere le relative operazioni solamente se si è in possesso dei privilegi richiesti, garantendo la separazione dei ruoli ed i minimi privilegi.

Capitolo 4

Sviluppo e Implementazione

4.1 Tecnologie utilizzate

Per la creazione della *blockchain* si è utilizzato il tool *quorum-wizard*, attraverso il quale si è configurata una rete con versione di *Quorum* pari alla 20.10.0, con tre nodi e un algoritmo di consenso di tipo bizantino.
Successivamente si è utilizzato il *Runtime Environment Node JS* per:

- L'implementazione di parte del *back-end* e del *front-end*;
- La compilazione ed il deploy degli *smart-contract*, mediante *truffle*;
- L'interfacciamento con la *blockchain*, mediante *Web3*.

4.2 Struttura della directory

All'interno della directory di progetto sono presenti le seguenti cartelle:

- build: contiene i *metadati* relativi agli *smart-contract* compilati, tramite il comando *truffle compile*;
- contracts: contiene gli *smart-contract*:
  - CarbonFootprint.sol: contiene i metodi per l'inserimento delle risorse e l'estrazione delle informazioni;
  - Gestore_nft.sol: consente di gestire i ruoli degli account e fornisce l'interfaccia per il *CarbonFootprint* per l'esecuzione delle operazioni previste;
  - Generica.sol: libreria di supporto che implementa alcuni metodi per la manipolazione delle stringhe;
  - Migrations.sol: si genera automaticamente durante l'esecuzione di *truffle-config*.

- migrations: contiene i file Javascript per il *deploy* degli *smart-contract* sulla *blockchain*;
- implementazione: cartella contenente i file Javascript:
  - call.js: fornisce i metodi per l'estrazione delle informazioni dalla *blockchain*, tramite invocazioni delle *view* messe a disposizione dallo *smart-contract Gestore_nft*;
  - config.js: contiene i path relativi delle cartelle *node_modules* e *build/-contracts*.
  - form.js: contiene le funzioni richiamate dall'interfaccia *Gestore_nft* per fornire all'utente la *form* richiesta;
  - interfaccia.js: contiene l'implementazione dell'interfaccia da riga di comando;
  - send.js: contiene le funzioni per l'esecuzione delle transazioni nella *blockchain*, tramite l'invocazione di metodi pubblici dello *smart-contract Gestore_nft*.
- truffle-config.js: contiene le informazioni per la configurazione del *deploy* degli *smart-contract*.

I file e le cartelle non presenti nell'elenco si generano automaticamente tramite il tool *quorum-wizard* e rappresentano l'infrastruttura della *blockchain*.

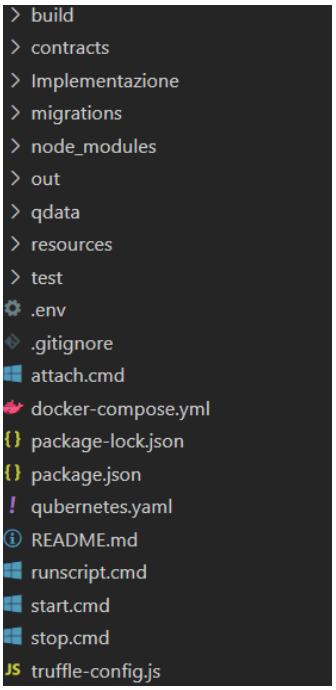

Figura 4.1: Struttura della directory di lavoro

4.3 Implementazione software

4.3.1 Osservazioni preliminari

Il *software* è stato implementato utilizzando alcune misure di sicurezza, in maniera tale da evitare intrusioni inaspettate.
sono state analizzate le vulnerabilità presenti nel catalogo *SWC Registry* (disponibile al seguente link) e sono state prese le seguenti precauzioni:

- Per evitare attacchi di tipo *Reentrancy* è stato utilizzato un pattern *Checks-Effects-Interactions*;
- All'interno dello *smart-contract Gestore_nft* sono presenti tre tabelle, una per ogni possibile ruolo; in questo modo è possibile assegnare uno o più ruoli agli account e controllare se vengono rispettati i requisiti per poter svolgere le operazioni richieste:
  - Il primo account è l'amministratore, perciò può assegnare ruoli a se stesso e agli altri attraverso l'operazione di inserimento di un attore;
  - Tutti gli account possono visionare il catalogo;
  - Solo i *produttori* possono inserire materie prime;
  - Solo i *trasformatori* possono inserire prodotti trasformati; durante l'operazione di inserimento, per evitare problematiche dovute ad un'allocazione eccessiva di memoria, si impone che il numero massimo di lotti di materie prime e di attività svolte sia compreso tra zero e cinque (estremi esclusi);
  - Sia *produttori* che *trasformatori* possono trasferire ad un *produttore/trasformatore* o *cliente* una propria materia prima.
  - L'account collegato al nodo uno è l'account *amministratore*, capace di assegnare ruoli sia a se stesso che agli altri account.
- Poiché si sta utilizzando un versione del compilatore maggiore alla 0.8, i problemi di *overflow* durante le operazioni di addizione sono direttamente controllati dallo stesso, fornendo uno specifico messaggio di errore; per poterlo personalizzare è presente una clausola *Require* all'interno di *CarbonFootprint*.
- I valori di CO_2 inseriti nella *blockchain* sono degli *uint32*, ciò consente di guadagnare in termini di gas, evitando inoltre di utilizzare numeri frazionari, i quali potrebbero introdurre vulnerabilità;
- Ogni *form* nella quale sono inseriti dati dall'utente è soggetta ad una verifica degli *input* tramite l'utilizzo delle *regular expression*:

- nei campi numerici si controlla che l'*input* contenga solamente valori numerici;
- per i campi testuali si controlla che il valore non contenga caratteri speciali.

4.3.2 Descrizione dei file

./Contracts/

- CarboonFootprint.sol

```
contract CarboonFootprint is ERC721{
struct Risorsa{} //struttura che contiene le informazioni
   della risorsa
constructor () // costruttore dello smart contract,
   eredita il costruttore di ERC721
function creaProdotto() // funzione che, a partire dai
   dati ricevuti in ingresso, crea una risorsa utilizzando
    il metodo "_safeMint"
function setTipologiaUtilizzato() // funzione che riceve
   un lotto e contrassegna la relativa risorsa come "
   utilizzata"
function currentToken() // funzione che restituisce il
   numero di token prodotti fino all'istante della
   chiamata
function getRisorsaByIdProdotto() // funzione che riceve
   un token e restituisce la relativa risorsa
function getRisorsaByLotto() // funzione che riceve un
   lotto e restituisce la risorsa la relativa risorsa
function transferFrom() // funzione che trasferisce la
   risorsa
```

- Generica.sol

```
function stringCompare() // funzione che confronta le
   stringhe in ingresso e fornisce "true" se sono uguali
function toString // funzione che riceve un uint256, lo
   trasforma in una stringa e lo restituisce al chiamante
function concatenate() // funzione che riceve due stringhe
    e restituisce al chiamante la stringa ottenuta dalla
   loro concatenazione
}
```

- Gestore_nft.sol

```
contract Gestore_nft is Ownable{
    function aggiungiAttore() // funzione con cui il
       creatore dello smartContract puo' aggiungere un
       nuovo produttore, trasformatore o cliente
    function creaMateriaPrima() // riceve nome, valore di
       CO2 e lotto, controlla l'ammissibilita' dei valori
       e chiama la funzione "creaProdotto" di "
       CarbonFootprint"
```

```
function trasferimentoRisorsa() // riceve indirizzo
   del destinatario e lotto della risorsa; controlla l
   'ammssibilita' dell'operazione e chiama il metodo "
   transferFrom" di "carbonFootprint"
function creaProdotto() //riceve un vettore di lotti
   di materie prime, un vettore di nomi di attivita'
   svolte, un vettore di consumi delle attivita'
   svolte, un nome ed un lotto; controlla l'
   ammissibilita' dell'operazione e chiama il metodo "
   creaProdotto" di "CarbonFootprint"
function getOwnerbyToken() // funzione che riceve un
   token, verifica l'ammissibilita' dell'operazione e
   chiama il metodo "ownerOf" di CarbonFootprint
   restituendone il valore ottenuto
function getInfoByToken() // funzione che riceve un
   token, verifica l'ammissibilita' dell'operazione e
   chiama il metodo "getRisorsaByIdProdotto" di
   CarbonFootprint restituendo i valori della risorsa
   ottenuta
function getInfoByLotto() // funzione che riceve un
   lotto, verifica l'ammissibilita' dell'operazione e
   chiama il metodo "getRisorsaByLotto" di
   CarbonFootprint restituendo i valori della risorsa
   ottenuta
function getInfoByNome() // funzione che riceve un
   nome, verifica l'ammissibilita' dell'operazione,
   chiama il metodo "getRisorsaByIdProdotto" di
   CarbonFootprint restituendo una stringa contenente
   le informazioni di tutte le risorse con il nome
   ricevuto
function esistente() // funzione che riceve un account
    ed un ruolo; se l'account ricopre gia' quel ruolo,
    restituisce "true"
//overloading
function esistente() // funzione che riceve un account
   , scorre i tre "mapping" relativi ai ruoli e
   restituisce "true" se l'account ricopre gia' uno
   dei 3 possibili ruoli
}
```

./Implementazione/

- call.js

```
class Call{
```

```
    constructor() // costruttore della classe, consente di
        collegarsi al nodo 0
    ottieniaccounts() // funzione che restituisce i 3
       account collegati ai 3 nodi
    getInfoByLotto() // fornisce le informazioni di una
       risorsa a partire dal lotto e dall'account
       richiedente
    getInfoByNome() // fornisce le informazioni di una
       risorsa a partire dal nome e dall'account
       richiedente
    getInfoByToken() // fornisce le informazioni di una
       risorsa a partire dal token e dall'account
       richiedente
    getOwnerByToken() // fornisce l'indirizzo del
       possessore di una risorsa a partire token
    controlloRuolo() // funzione che riceve un indirizzo
       ed un ruolo; fornisce "true" se l'account ha gia'
       quel ruolo
    stampa_risorsa() // funzione per formattare i
       risultati; viene richiamata all'interno degli altri
        metodi della classe
}
```

- form.js

```
function form_inserimento_attore() // form in cui
   scegliere un attore tra quelli disponibili ed un ruolo
   tra quelli assegnabili
function form_materia_prima() // form in cui inserire,
   nome, valore CO2 e lotto di una nuova materia prima;
function form_numero_prodotto() // form preliminare all'
   inserimento dei prodotti; consente l'inserimento del
   numero di materie prime ed attivita' relative ad un
   prodotto
function form_prodotto() // form in cui inserire il nome e
    il lotto del nuovo prodotto, i lotti delle materie
   prime utilizzate, le attivita' svolte durante la
   trasformazione del prodotto ed il loro consumo
function form_trasferimento() // form in cui inserire la
   risorsa da trasferire e l'account del ricevitore;
function form_by_token() // form in cui inserire il token
   relativo ad una risorsa
function form_by_lotto() // form in cui inserire il lotto
   relativo ad una risorsa
function form_by_nome() // form in cui inserire il nome di
    una risorsa
```

```
function form_operazione() //form che consente di
   scegliere l'operazione da effettuare
function form_account() // form che consente la scelta di
   un account con cui autenticarsi
function form_continua() // form in cui inserire "Y" o "N"
    scegliendo se continuare ad eseguire operazioni con l'
   account corrente o in alternativa terminare il
   programma.
```

- interfaccia.js

```
function startInterface() // mostra il messaggio di
   benvenuto
function getPrivilegiAttori() // estrazione dei privilegi
   degli attori
function checkPrivilegi() // controllo dei privilegi dell'
   account
function loop() // consente all'attore di scegliere se
   continuare o meno a svolgere operazioni
function esegui() // corpo del programma
```

- send.js

```
class Send {
    constructor() // costruttore della classe, consente di
        collegarsi al nodo passato come parametro
    aggiungiAttore() // funzione che riceve l'account
       richiedente, l'account a cui assegnare il ruolo ed
       il ruolo stesso
    aggiungiMateriaPrima() // funzione che crea la materia
        prima a partire dall'account richiedente, il lotto
       , il nome ed il valore di CO2
    trasferimentoRisorsa() //funzione che trasferisce la
       risorsa a partire dall'account richiedente, dal
       destinatario e dal lotto della risorsa da
       trasferire
    creaProdotto() //funzione che crea un proddoto
       trasformato a partire da: account richiedente,
       vettore con i nomi delle attivita', vettore con i
       consumi delle attivita', nome del nuovo prodotto,
       vettore con i lotti delle materie prime utilizzate
       per la produzione e lotto del nuovo prodotto
    }
```

4.4 Testing

Durante la fase di scrittura del codice è stato utilizzato *Solhint*, un'estensione di *Visual Studio Code* che consente di individuare i *code smell*, in modo da mitigare la presenza di vulnerabilità nel codice *solidity*.
La fase di testing dell'applicazione è stato diviso in tre fasi:

- Testing del codice *solidity* tramite *SMTChecker*: per l'individuazione di *code smell* nel codice; i *warning* rimanenti sono falsi positivi o sono relativi alle librerie importate nel programma;
- Testing dei metodi di interfaccia con la *blockchain*: tramite dei test semi-automatici, scritti utilizzando il modulo *mocha*, è stato possibile testare l'interfacciamento tra codice *Javascript* e codice *solidity*; inoltre sono state testate tutte le condizioni che portano i *require* a fallire, in modo da valutarne la correttezza; il codice relativo è presente nel file *test/transazioni.js*
- Testing della CLI: la validazione degli input inseriti dall'utente attraverso le form è stata svolta tramite dei test semi-automatici; il codice relativo è presente nel file *test/form.js*

Nel caso in cui si volessero lanciare i test, per verificarne la correttezza, è possibile farlo utilizzando il comando *npm test*; si consiglia di testare su una *blockchain* priva di transazioni, in quanto le funzioni interagiscono con essa ed il sistema potrebbe incorrere in errore se lo stato è modificato.
Al termine del testing sarà modificato lo stato della blockchain; per poter ottenere una blockchain priva di transazioni sarà necessario cancellare la cartella *build* e lanciare di nuovo il comando *truffle deploy*.

Capitolo 5

Manuale d'uso e sviluppi futuri

Il *software*, incluso il manuale d'uso e le istruzioni di configurazione, sono disponibili e consultabili al seguente link.

Come espresso in precedenza, alcune funzionalità sono state tralasciate; in particolare si immagina che alcuni possibili sviluppi del progetto riguardino:

- Implementazione di una procedura di autenticazione;
- Sviluppo del *front-end* con interfaccia grafica;
- Implementazione di una procedura di acquisto diretta da parte del cliente;
- Ottimizzazione delle procedure di inserimento dati all'interno della *blockchain*.

Elenco delle figure