Relazione Progetto Software Cybersecurity

SISTEMA DI TRACCIAMENTO DI UNA FILIERA PRODUTTIVA

AUTORI

MARCOLINI ALESSANDRO, PAOLINI DAVIDE, SALVONI SIMONE,
SOPRANZETTI LORENZO, TISENI LORENZO

INDICE

CONTESTO

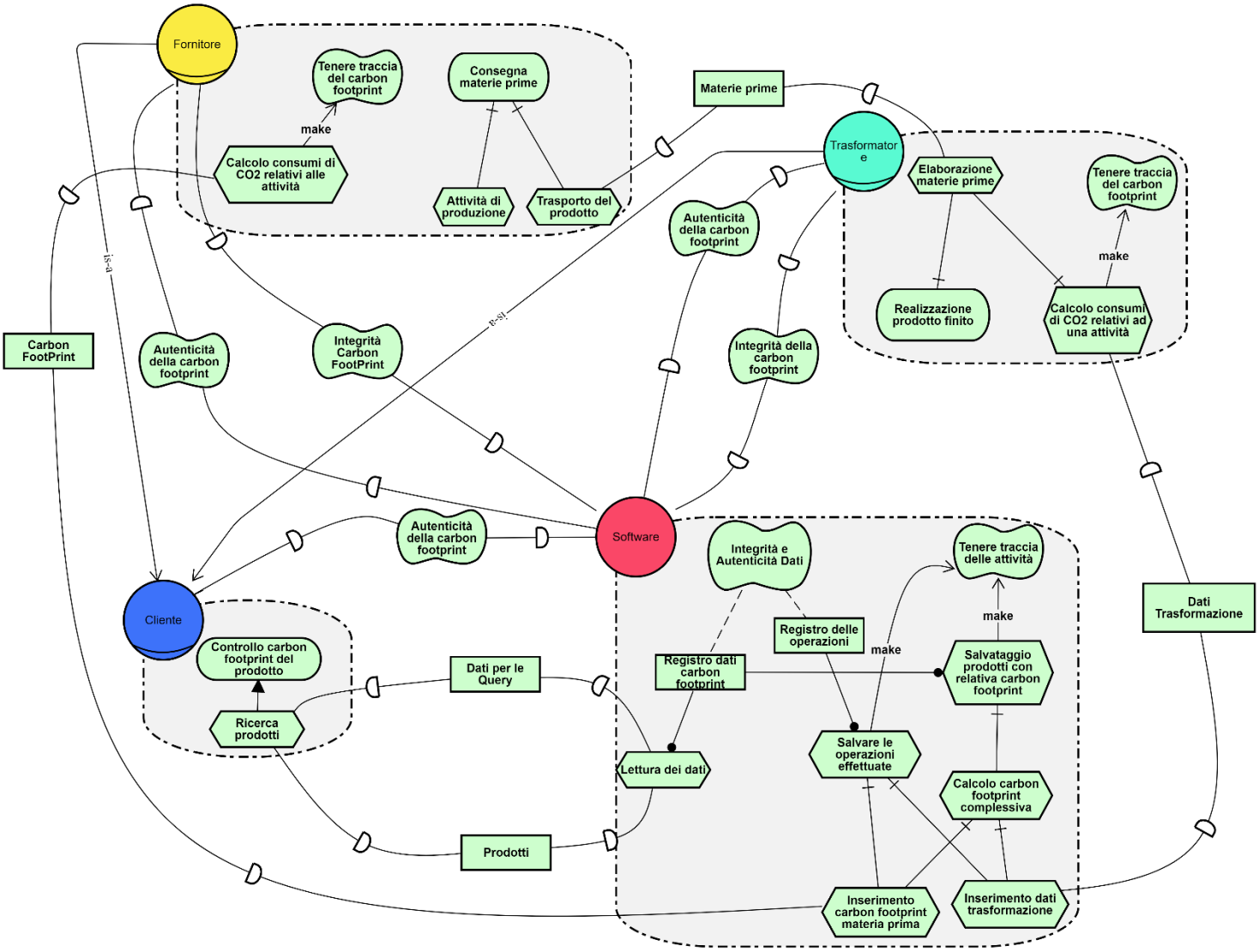
Viene qui presentato un sistema di tracciamento del lavoro di un consorzio di imprese specializzate nella produzione. L'obiettivo principale è quello di tracciare i processi produttivi delle varie aziende che entrano a far parte del consorzio, con particolare attenzione all'inquinamento che i processi producono. Le aziende possono essere dei fornitori di materie prime, oppure delle imprese che trasformano queste materie prime per far avanzare la catena di produzione, fino ad arrivare al prodotto finito. Della catena produttiva di ogni prodotto vengono registrate permanentemente e apertamente:

- Le varie materie prime che sono state utilizzate per far partire la catena produttiva. Per ognuna di esse viene registrato da che fornitore provengono, per che prodotto sono state utilizzate e la loro impronta di carbonio.
- Le varie trasformazioni che sono state eseguite partendo dalle materie prime, e per ognuna la relativa impronta di carbonio
- I prodotti finali e le loro impronte di carbonio complessive.

Gli utenti che dovranno interagire con il sistema sono di tre tipologie:

- **Clienti:** cioè gli utenti generici. I clienti possono ricercare i vari prodotti tracciati, controllandone le materie prime, le trasformazioni e le impronte di carbonio di ognuna di queste e del prodotto finito.
- **Fornitori:** Questa classe di utenti, oltre a poter eseguire ricerche come i clienti, ha anche la possibilità di inserire nuove materie prime nel sistema quando vengono consegnate a un certo trasformatore, su previo accordo di vendita.
- **Trasformatori:** Questa classe di utenti, oltre a poter eseguire ricerche come i clienti, ha anche la possibilità di creare nuovi prodotti, aggiungere trasformazioni a prodotti non conclusi e passare la proprietà di un prodotto ad un altro trasformatore che dovrà continuare la catena produttiva.

Lo scenario descritto sinteticamente nei paragrafi precedenti, è stato rappresentato, includendo anche il sistema di tracciamento da realizzare, utilizzando il linguaggio di modellazione **i***. La scelta di **i*** è utile perché esso permette di modellare non solo il sistema di tracciamento in sé, ma anche tutto il sistema socio-tecnico intorno al sistema. In questa maniera si riesce, oltre a dare una descrizione delle funzionalità di alto livello del software, anche a spiegare il modo in cui esso si inserisce nel contesto della filiera produttiva descritto in precedenza. La rappresentazione dello scenario è visibile nella figura a pagina successiva. In questo schema si vede come interagiscono le varie entità che lavorano nella filiera produttiva, utilizzando il sistema di tracciamento. Grande rilevanza viene posta sulle azioni che le entità eseguono durante il processo produttivo, e su come esse devono interagire con il sistema software. Oltre a questo grazie al linguaggio i* è possibile mettere in rilievo anche tutti gli obiettivi che hanno le varie entità nella loro attività quotidiana, e anche quali altre entità possono contribuire al raggiungimento degli obiettivi. Particolare attenzione va posta sugli obiettivi, il cui raggiungimento è legato al sistema software, i quali sono integrità e autenticità dei dati.



Partendo dallo scenario descritto, è iniziata la fase di progettazione del sistema software, in cui in tutte le fasi svolte, ovvero analisi dei requisiti, progettazione e implementazione, si è sempre tenuto conto del rischio cyber.

ANALISI DEI RISCHI PRELIMINARE

Nella prima fase di analisi dei requisiti del sistema software, è stata eseguita un'analisi delle possibili minacce o attacchi a cui il sistema stesso è sottoposto. Questa tipologia di analisi del rischio, essendo eseguita prima di effettuare scelte progettuali e tecnologiche, è un'analisi di alto livello, che cerca di prendere in considerazione problematiche di carattere generale, che prescindono dall'implementazione. Per questo, in tale fase preliminare si utilizzano il catalogo CAPEC®, per individuare possibili pattern di attacco e, in misura minore, ATT&CK®, per avere una panoramica su alcune tecniche di attacco.

Identificazione asset

Il primo passo è stato identificare gli asset principali del sistema suddividendoli in dati e procedure. Entrambe queste categorie vanno protette, in quanto la loro compromissione va

a minare la sicurezza o l'affidabilità del sistema, arrecando un danno a tutti gli utenti finali. Gli asset identificati sono i seguenti:

1. **DATI**
    a. Impronta di carbonio delle materie prime
    b. Impronta di carbonio delle trasformazioni
    c. Impronta di carbonio dei prodotti finiti
    d. Registro delle operazioni effettuate
    e. Dati per l'autenticazione
2. **PROCEDURE**
    a. Procedura di inserimento dati del prodotto
    b. Procedura di lettura dei dati
    c. Procedura di inserimento di un'operazione
    d. Procedura di calcolo dell'impronta di carbonio complessiva
    e. Procedura di autenticazione

Per ogni asset è stato stimato un valore, sono state definite le policy di sicurezza da soddisfare, ed è stato assegnato un valore di "esposizione" alle minacce, cioè una valutazione qualitativa della vulnerabilità dell'asset. Le principali policy di sicurezza identificate sono:

- **Integrità e autenticità per i dati inseriti:** ciò significa che i dati inseriti e poi mostrati all'utente devono corrispondere a operazioni realmente eseguite e alle reali impronte di carbonio dei vari prodotti.
- **Disponibilità e affidabilità delle procedure:** le azioni a disposizione degli utenti, quindi, devono essere sempre eseguibili, e si devono svolgere nella maniera prevista.
- **Responsabilità e Non Ripudio:** Le azioni degli utenti non devono essere ripudiabili dagli stessi
- **Autorizzazione:** solamente gli utenti con i corretti privilegi (fornitori e trasformatori) devono poter aggiungere nuovi dati nel sistema.

Di seguito i dettagli, esplicitati in delle tabelle realizzate con Microsoft Excel

Asset	Value (1-3)	Objective	Exposure (1-5)
Carbon footprint di ogni materia prima	2	L'asset non può essere modificato o cancellato (Integrità). Il dato deve essere corrispondente alla carbon footprint reale.(Autenticità)	(Integrità=5, Autenticità=2) 5
Carbon footprint di ogni trasformazione	2	L'asset non può essere modificato o cancellato (Integrità). Il dato deve essere corrispondente alla carbon footprint reale.(Autenticità)	(Integrità=5, Autenticità=2) 5
Carbon footprint di ogni prodotto finito	3	L'asset non può essere modificato o cancellato (Integrità). Il dato deve essere corrispondente al calcolo svolto con le carbon footprint parziali (Autenticità)	(Integrità=5, Autenticità=2) 5

Registro delle operazioni effettuate	1	Un'operazione salvata nel registro non può essere modificata o cancellata (Integrità). Le operazioni registrate devono essere realmente accadute (Autenticità).	(Integrità=3, Autenticità=3) 3
Dati per Autenticazione	2	Questo asset deve rimanere privato (Confidenzialità). I dati non devono essere modificati o cancellati (Integrità)	(Confidenzialità=2, Integrità=2) 2
Procedura di inserimento dati prodotto	3	Chi ha eseguito la procedura non può ripudiarla (Responsabilità). La procedura deve essere sempre eseguibile (Disponibilità). L'operazione deve essere svolta nella maniera prevista (Reliability). Solo chi è autorizzato può inserire i dati (Autorizzazione)	(Responsabilità=5, Disponibilità=1, Reliability=3, Autorizzazione =2) 5
Procedura di lettura dati	1	Il dato restituito in lettura deve essere veritiero (Autenticità). La procedura deve essere sempre eseguibile (Disponibilità). L'operazione deve essere svolta nella maniera prevista (Reliability).	(Autenticità=3, Disponibilità=1, Reliability=2) 3
Procedura di inserimento di un'operazione	2	Solo il software può richiamare e far eseguire la procedura (Autorizzazione). L'operazione deve poter essere sempre svolta dal sistema (Disponibilità)	(Autorizzazione=2, Disponibilità=2) 2
Procedura di calcolo della carbon footprint complessiva di ogni prodotto	3	Solo il software può richiamare e far eseguire la procedura (Autorizzazione). L'operazione deve essere svolta nella maniera prevista (Reliability).	(Reliability = 4, Autorizzazione = 3) 4
Procedura di autenticazione	1	L'operazione deve essere svolta nella maniera prevista (Reliability). Devono essere identificati come utenti registrati solo coloro che inseriscono le credenziali giuste (Autenticità). La procedura deve essere disponibile (Disponibilità)	(Reliability=3 Autenticità=2 Disponibilità=1) 3

Identificazione delle minacce e studio delle contromisure

Individuati gli asset da proteggere, si è proceduto con l'identificazione delle minacce che realisticamente avrebbero potuto comprometterli, creando quindi delle "security breach". Le minacce individuate sono state raccolte nella seguente tabella, in cui sono elencate insieme a tutti i riferimenti necessari per ritrovarle all'interno del catalogo CAPEC®.

Asset	Voce CAPEC		Policy
	ID	Name	
Procedura di inserimento dati prodotto	664	Server Side Request Forgery (SSRF)	Affidabilità, disponibilità
	94	Adversary in the middle	Autorizzazione
	194	Fake the Source of Data	Autorizzazione
	593	Session hijacking	Autorizzazione
	125	Flooding	Disponibilità
	196	Session Credential Falsification Through Forging	Responsabilità, Autorizzazione
	242	Code injection	Affidabilità, Disponibilità
Procedura di lettura dati	94	Adversary in the middle	Autenticità
	125	Flooding	Disponibilità
	664	Server Side Request Forgery (SSRF)	Affidabilità, Disponibilità
	242	Code injection	Affidabilità, Disponibilità
Procedura di inserimento di un'operazione	180	Exploiting Incorrectly Configured Access Control Security Levels	Autorizzazione
	664	Server Side Request Forgery (SSRF)	Disponibilità, (Autorizzazione)
Procedura di calcolo della carbon footprint complessiva	100	Buffer overflow	Affidabilità
	180	Exploiting Incorrectly Configured Access Control Security Levels	Autorizzazione
	74	Manipulating state	Affidabilità
Procedura di autenticazione	242	Code injection	Affidabilità, Disponibilità
Carbon footprint di ogni materia prima	194	Fake the Source of Data	Autenticità
	248	Command Injection	Integrità
	240	Resource Injection	Integrità
Carbon footprint di ogni trasformazione	194	Fake the Source of Data	Autenticità
	248	Command Injection	Integrità
	240	Resource Injection	Integrità
Carbon footprint di ogni prodotto finito	248	Command Injection	Integrità
	240	Resource Injection	Integrità
Registro delle operazioni effettuate	240	Resource Injection	Integrità
	664	Server Side Request Forgery (SSRF)	Autenticità
Dati per Autenticazione	112	Brute Force	Confidenzialità
	240	Resource Injection	Integrità

Identificate le minacce principali, si prosegue eseguendo l'analisi del rischio qualitativa per ciascuna minaccia e per ogni asset. Per prima cosa esse sono stati categorizzate utilizzando il modello STRIDE-DUAl, il quale rappresenta una versione estesa dello STRIDE model. Esso cataloga i vari rischi in delle macro-categorie riconducibili al tipo di policy che viene violata, la quale o fa parte dell'insieme delle policy di sicurezza (se si fa riferimento alla parte

STRIDE) o fa parte delle policy di dependability(se si fa riferimento alla parte DUA). Le macro-categorie del modello enunciato, relative alla parte STRIDE sono le seguenti:

- **Spoofing:** comprende tutti gli attacchi in cui l'attaccante ha lo scopo di impersonare un'altra persona, cioè tutti quelli che causano una violazione dell'autenticazione.
- **Tampering:** comprende tutte le minacce che compromettono l'integrità dei dati
- **Repudiation:** comprende tutte le minacce che permettono a malintenzionati di rendere irriconoscibile l'autore di una certa azione malevola, provocando una violazione della responsabilità.
- **Information disclosure:** comprende tutte le minacce che hanno lo scopo di rendere pubbliche delle informazioni riservate, provocando una violazione della confidenzialità.
- **Elevation of privilege:** comprende tutti gli attacchi che permettono di bypassare i meccanismi di autorizzazione dei sistemi, per ottenere privilegi più alti di quelli concessi alla categoria di utente a cui si appartiene. Tali attacchi provocano una violazione dell'autorizzazione.

Per quanto riguarda invece l'estensione DUA, le macro-categorie identificate sono:

- **Danger:** comprende tutte le minacce i cui effetti rischiano di danneggiare persone o cose appartenenti al sistema socio-tecnico nel quale è inserito il sistema software
- **Unreliability:** comprende le minacce che rendono imprevedibile il comportamento del sistema, danneggiando la sua usabilità da parte degli utenti finali.
- **Absence of resilence:** comprende le minacce che danneggiano la robustezza del sistema, cioè la sua capacità di ritornare operativo e ripristinare tutte le funzioni dopo un qualsiasi tipo di attacco.

Si ricorda, ovviamente che una minaccia non appartiene solo a una singola macro-categoria, ma può far parte di più macro-categorie contemporaneamente. Aderendo quindi al modello descritto si è operata una categorizzazione dei rischi rilevati, esplicitata nella tabella successiva:

Asset	Value	Spoofing	Tampering	Repudiation	Information disclosure	DOS	Elevation of privilege	Danger	Unreliability	Absence of Resilience	Exposure 1-5	Attack
Carbon footprint di ogni materia prima	2		X								(Integrità=5, Autenticità=2) 5	Resource Injection
		X										Fake the Source of Data
			X									Command Injection
Carbon footprint di ogni trasformazione	2		X								(Integrità=5, Autenticità=2) 5	Resource Injection
		X										Fake the Source of Data
			X									Command Injection
Carbon footprint di ogni prodotto finito	3		X								(Integrità=5, Autenticità=2) 5	Resource Injection
			X									Command Injection
Registro delle operazioni effettuate	1		X								(Integrità=3, Autenticità=3) 3	Resource Injection
		X										Server Side Request Forgery
Dati per Autenticazione	2				X						(Confidenzialità=2, Integrità=2) 2	Brute Force
			X									Resource Injection

Asset	Value	Spoofing	Tampering	Repudiation	Information disclosure	DOS	Elevation of privilege	Danger	Unreliability	Absence of Resilience	Exposure 1-5	Attack
Procedura di inserimento dati prodotto	3					X			X		(Responsabilità=5, Disponibilità=1, Reliability=3, Autorizzazione =2) 5	Server Side Request Forgery
							X					Adversary in the middle
							X					Fake the Source of Data
							X					Session Hijacking
						X						Flooding
				X			X					Session Credential Falsification Through Forging
						X			X			Code Injection

Asset	Value	Spoofing	Tampering	Repudiation	Information disclosure	DOS	Elevation of privilege	Danger	Unreliability	Absence of Resilience	Exposure 1-5	Attack
Procedura di lettura dati	1	X									(Autenticità=3, Disponibilità=1, Reliability=2) 3	Adversary in the middle
						X						Flooding
						X			X			Server Side Request Forgery
						X			X			Code Injection
Procedura di inserimento di un'operazione	2						X				(Autorizzazione=2, Disponibilità=2) 2	Exploiting Incorrectly Configured Access Control Security Levels
						X	X					Server Side Request Forgery

Asset	Value	Spoofing	Tampering	Repudiation	Information disclosure	DOS	Elevation of privilege	Danger	Unreliability	Absence of Resilience	Exposure 1-5	Attack
Procedura di calcolo della carbon footprint complessiva di ogni prodotto	3								X		(Reliability=4, Autorizzazione=3) 4	Buffer Overflow
							X					Exploiting Incorrectly Configured Access Control Security Levels
									X			Manipulating state
Procedura di autenticazione	1					X			X		(Reliability=3 Autenticità=2 Disponibilità=1) 3	Code Injection

Conclusa la categorizzazione dei rischi, è stato stimato il livello di rischio per ogni asset e per ogni minaccia relativa, stimando, nello specifico, l'impatto e la probabilità di accadimento. Sulla base delle stime effettuate si è proseguito, con l'identificazione di possibili contromisure applicabili, in maniera tale da ridurre il rischio inerente. Per ogni soluzione proposta è stimato il costo di esecuzione, la sua fattibilità, e una stima del rischio residuo.

Asset	Exposure 1-5	Attack	Inherent Probability	Inherent Risk	Control	Cost 1-5	Feasibility (bassa, media,alta)	Residual Probability	Residual Impact	Residual Risk
Carbon footprint di ogni materia prima	(Integrità=5, Autenticità=2) 5	Resource Injection	3	3	Sistema di validazione dell'input	1	Alta	1	5	2
		Fake the Source of Data	3	2	Autenticazione a due fattori	3	Media	1	2	1
		Command Injection	2	3	Sistema di validazione dell'input	1	Alta	1	5	2
					Blockchain	4	Media	2	2	2
Carbon footprint di ogni trasformazione	(Integrità=5, Autenticità=2) 5	Resource Injection	3	3	Sistema di validazione dell'input	1	Alta	1	5	2
		Fake the Source of Data	3	2	Autenticazione a due fattori	3	Media	1	2	1
		Command Injection	2	3	Sistema di validazione dell'input	1	Alta	1	5	2
					Blockchain	4	Media	2	2	2
Carbon footprint di ogni prodotto finito	(Integrità=5, Autenticità=2) 5	Resource Injection	3	3	Sistema di validazione dell'input	1	Alta	1	5	2
		Command Injection	2	3	Sistema di validazione dell'input	1	Alta	1	5	2
					Blockchain	4	Media	2	2	2

Asset	Exposure 1-5	Attack	Inherent Probability	Inherent Risk	Control	Cost 1-5	Feasibility (bassa, media,alta)	Residual Probability	Residual Impact	Residual Risk
Registro delle operazioni effettuate	(Integrità=3, Autenticità=3) 3	Resource Injection	3	2	Sistema di validazione dell'input	1	Alta	1	3	1
		Server Side Request Forgery	3	2	Blockchain	4	Media	1	3	1
Dati per Autenticazione	(Confidenzialità=2, Integrità=2) 2	Brute Force	2	2	Autenticazione a due fattori	3	Media	2	1	1
					Limite tentativi login	2	Alta	1	2	1
		Resource Injection	3	2	Sistema di validazione dell'input	1	Alta	1	2	1

Asset	Exposure 1-5	Attack	Inherent Probability	Inherent Risk	Control	Cost 1-5	Feasibility (bassa, media,alta)	Residual Probability	Residual Impact	Residual Risk
Procedura di inserimento dati prodotto	(Responsabilità=5, Disponibilità=1, Reliability=3, Autorizzazione =2) 5	Server Side Request Forgery	3	(1/3 e 3/3) 2	Blockchain	4	Media	1	3	1
		Adversary in the middle	3	2	Sistema di crittografia asimmetrica	3	Media	1	3	1
		Fake the Source of Data	3	2	Autenticazione a due fattori	3	Media	1	2	1
		Session Hijacking	3	2	Meccanismo di session timeout e gestione delle sessioni	3	Alta	2	1	1
		Flooding	3	1	Implementare dei limiti nelle richieste ai servizi e nell'utilizzo delle risorse di sistema	2	Alta	1	1	1
					Blockchain	4	Media	1	1	1
		Session Credential Falsification Through Forging	2	3	Gestione delle sessioni e session id di sufficiente complessità	3	Alta	1	3	1
		Code Injection	3	3	Sistema di validazione dell'input	1	Alta	1	3	1
					Blockchain	4	Media	1	3	1

Asset	Exposure 1-5	Attack	Inherent Probability	Inherent Risk	Control	Cost 1-5	Feasibility (bassa, media,alta)	Residual Probability	Residual Impact	Residual Risk
Procedura di lettura dati	(Autenticità=3, Disponibilità=1, Reliability=2) 3	Adversary in the middle	3	2	Sistema di crittografia asimmetrica	3	Media	1	3	1
		Flooding	3	1	Implementare dei limiti nelle richieste ai servizi e nell'utilizzo delle risorse di sistema	2	Alta	1	1	1
					Blockchain	4	Media	1	1	1
		Server Side Request Forgery	3	2	Blockchain	4	Media	1	3	1
		Code Injection	3	3	Sistema di validazione dell'input	1	Alta	1	3	1
					Blockchain	4	Media	1	3	1
Procedura di inserimento di un'operazione	(Autorizzazione=2, Disponibilità=2) 2	Exploiting Incorrectly Configured Access Control Security Levels	3	2	Progettare il controllo degli accessi in maniera adeguata	3	Media	1	2	1
					Blockchain	4	Media	1	2	1
		Server Side Request Forgery	3	2	Blockchain	4	Media	1	3	1

Asset	Exposure 1-5	Attack	Inherent Probability	Inherent Risk	Control	Cost 1-5	Feasibility (bassa, media,alta)	Residual Probability	Residual Impact	Residual Risk
Procedura di calcolo della carbon footprint complessiva di ogni prodotto	(Reliability = 4, Autorizzazione = 3) 4	Buffer Overflow	3	3	Utilizzare un linguaggio di programmazione che controlli automaticamente i limiti dei buffer	1	Alta	1	1	1
		Exploiting Incorrectly Configured Access Control Security Levels	3	2	Progettare il controllo degli accessi in maniera adeguata	3	Media	1	2	1
					Blockchain	4	Media	1	2	1
		Manipulating state	2	2						
Procedura di autenticazione	(Reliability=3 Autenticità=2 Disponibilità=1) 3	Code Injection	3	2	Sistema di validazione dell'input	1	Alta	1	3	1
					Blockchain	4	Media	1	3	1

Use case, Abuse case, Misuse case

Infine, sono stati stilati i vari casi d'uso, di abuso e di misuso. Li definiamo nei seguenti modi:

- I casi d'uso sono le interazioni previste che gli utenti possono avere con il sistema.
- I casi di abuso sono le interazioni non previste che un utente può avere con il sistema, e che ha lo scopo di violare le policy di sicurezza.
- I casi di misuso sono invece le interazioni non previste che utenti possono eseguire senza volerlo. Queste interazioni hanno la possibilità di violare le policy di sicurezza inavvertitamente, o di causare comportamenti inaspettati.

Per la descrizione dei casi d'uso, di misuso e di abuso si è deciso di aderire allo schema di Jacobson in quanto ritenuto completo e sufficiente per una descrizione dettagliata. Di seguito sono riportate le informazioni riguardanti i casi d'uso:

Case Type	Use Case	Case ID	0
Case Name	Inserimento dati prodotto		
Actors	Fornitore / Trasformatore e Software		
Description	Il fornitore/trasformatore inserisce nel sistema dei dati parziali sulla carbon footprint del prodotto		
Data	Dati parziali sulla carbon footprint. Se l'attore è il fornitore, si tratta della carbon footprint relativa alle materie prime e al trasporto di queste, mentre se è il trasformatore, si tratta della carbon footprint di un processo di trasformazione delle materie prime		
Stimulus and preconditions	Precondizione: Il fornitore deve aver calcolato la carbon footprint della materia prima consegnata al trasformatore, oppure il trasformatore deve aver calcolato la carbon footprint di un processo di trasformazione terminato. Stimolo: Viene eseguito il processo di inserimento dal fornitore o dal trasformatore.		
Basic Flow	1) Il fornitore/trasformatore inserisce i dati della materia prima/trasformazione nel sistema tramite un'apposita interfaccia di input. 2) Il sistema inserisce nel suo registro dei prodotti i dati immessi compresa la carbon footprint 3) Il sistema salva nel registro delle operazioni la transazione eseguita		
Alternative Flow	1) Il trasformatore inserisce i dati sulla trasformazione nel sistema tramite un'apposita interfaccia di input. 2) Il trasformatore indica che sta inserendo i dati relativo all'ultimo processo di trasformazione 3) Il sistema inserisce nel suo registro dei prodotti i dati immessi compresa la carbon footprint 4) Il sistema calcola la carbon footprint complessiva del prodotto 5) Il sistema salva nel registro prodotti il risultato complessivo per la carbon footprint 6) Il sistema salva nel registro delle operazioni l'operazione eseguita		
Exception Flow	1) Il fornitore/trasformatore inserisce i dati della materia prima/trasformazione nel sistema tramite un'apposita interfaccia di input. 2) I dati non vengono correttamente trasmessi 3) Il sistema restituisce un messaggio di errore ---------- 1) Il fornitore/trasformatore inserisce i dati della materia prima/trasformazione nel sistema tramite un'apposita interfaccia di input. 2) Si verifica un errore nel salvataggio dei dati 3) Il sistema restituisce un messaggio di errore		

Response and Postconditions	Il sistema restituisce un messaggio che conferma il corretto inserimento e salvataggio dei dati
Non Functional Requirements	Integrità, Responsabilità, Disponibilità, Reliability, Autorizzazione

Case Type	Use Case	Case ID	1
Case Name	Ricerca Prodotti		
Actors	Cliente e Software		
Description	Il cliente richiede e poi legge la carbon footprint di uno o più prodotti in base ai criteri di ricerca. Può leggere sia la footprint delle fasi parziali di produzione sia quella totale.		
Data	Dati della query eseguita dal cliente al software. Registro delle carbon footprint dei prodotti. Carbon footprint parziali e complessive dei prodotti selezionati.		
Stimulus and preconditions	Il cliente esegue la lettura tramite un'apposita interfaccia del sistema.		
Basic Flow	1) Il cliente esegue la ricerca di uno o più prodotti 2) Il sistema ricerca nel registro i prodotti conformi ai criteri di ricerca 3) Il sistema fornisce al cliente le informazioni relative ai prodotti trovati 4) Il cliente riceve le informazioni e le legge		
Alternative Flow			
Exception Flow	1) Il cliente esegue la ricerca di uno o più prodotti 2) La ricerca non va a buon fine 3) Il sistema avverte l'utente con un messaggio di errore		
Response and Postconditions	Il sistema restituisce le informazioni sui prodotti ricercati		
Non Functional Requirements	Autenticità, disponibilità, reliability		

Case Type	Use Case	Case ID	2
Case Name	Procedura di autenticazione		
Actors	Fornitore/Trasformatore e Software		
Description	I fornitori e i trasformatori devono eseguire un processo di autenticazione per essere riconosciuti come tali		
Data	Credenziali di autenticazione		
Stimulus and preconditions	Fornitori e trasformatori eseguono un'autenticazione tramite un'interfaccia apposita fornita dal sistema		
Basic Flow	1) L'utente inserisce i propri dati di autenticazione 2) Le credenziali sono riconosciute e il sistema autentica l'utente con successo 3) L'utente riceve accesso alle funzionalità di inserimento dati		
Alternative Flow	1) L'utente inserisce i propri dati di autenticazione 2) Il sistema non riconosce le credenziali inserite dall'utente 3) Il sistema avverte l'utente che l'autenticazione è fallita 4) Il sistema permette all'utente di riprovare		
Exception Flow	1) L'utente inserisce i propri dati di autenticazione 2) L'autenticazione non va a buon fine per via un errore interno 3) Il sistema avverte l'utente con un messaggio di errore		
Response and Postconditions	Il sistema autentica l'utente se le credenziali sono corrette, altrimenti lo avverte che sono sbagliate		
Non Functional Requirements	Autenticità, reliability		

I casi di abuso identificati sono invece:

Case Type	Abuse Case	Case ID	0
Case Name	Resource Injection (carbon footprint della materia prima, della trasformazione, del prodotto finito, dati autenticazione, registro operazioni).		
Actors	Attaccante		
Description	Un attaccante sfrutta le debolezze nella validazione dell'input manipolando gli identificatori delle risorse permettendogli di fare modifiche su di esse.		
Data	L'attaccante necessita di conoscere l'identificatore utilizzato dal sistema per accedere a determinate risorse.		
Stimulus and preconditions	Precondizione: L'applicazione target deve permettere all'utente di specificare gli identificatori usati per accedere a una risorsa di sistema.		

	Stimolo: l'utente sfrutta questo permesso inserendo gli identificatori della risorsa e prendendo l'accesso
Attack Flow 1	1) L'attaccante scopre che il sistema è vulnerabile a questo tipo di attacco 2) L'attaccante prova a modificare i parametri che contengono degli identificatori di risorse 3) Trovati i parametri vulnerabili, accede ad una risorsa differente da quella richiesta oppure protetta
Attack Flow 2	
Attack Flow 3	
Response and Postconditions	L'attaccante può accedere a dati riservati o modificare dati del sistema
Non Functional Requirements	Integrità
Mitigations	- Assicurarsi che l'input sia sanificato consentendo il solo inserimento di contenuti accettabili - Validare l'input per ogni contenuto - Correggere e controllare regolarmente il software
Comments	

Case Type	Abuse Case	Case ID	1
Case Name	Server side request forgery (inserimento e lettura dati prodotto, registro delle operazioni)		
Actors	Attaccante, sistema		
Description	Un attaccante sfrutta un'erronea validazione degli input fornendo al sistema un input mal formattato, con lo scopo di forzare il server a fare richieste a se stesso o a altri servizi web che sono in esecuzione sulla stessa rete interna del server, o a terze parti esterne. Se l'attacco ha successo, la richiesta dell'attaccante verrà eseguita con il livello di privilegi del server, bypassando i controlli di autenticazione.		
Data	L'attaccante deve costruire un input malevolo apposito per far eseguire al server la richiesta che desidera far eseguire.		
Stimulus and preconditions	Precondizione: la validazione degli input deve essere assente o malfunzionante. Il server deve essere un servizio web che gestisce richieste HTTP. Stimolo: l'attaccante spedisce l'input malevolo		

Attack Flow 1	1) L'attaccante scopre che il sistema è vulnerabile a questa tipologia di attacco eseguendo appositi test. 2)L'attaccante costruisce l'input malevolo a dipendenza dell'azione che vuole far eseguire al server 3) L'attaccante inserisce e spedisce questo input tramite una delle form disponibili: quella per l'inserimento dati o quella per la lettura dei dati. 4)La richiesta viene eseguita dal servizio che ha ricevuto l'input con i privilegi del server.
Attack Flow 2	
Attack Flow 3	
Response and Postconditions	La richiesta eseguita può causare vari effetti dannosi, tra cui la lettura di dati confidenziali o la scrittura di dati senza autenticazione.
Non Functional Requirements	Affidabilità, disponibilità
Mitigations	-Il sistema deve sfruttare una validazione degli input che riceve, che possa controllare e in caso sanificare gli input che arrivano. -Successivamente, il sistema deve controllare la risposta prima di inviarla al richiedente, assicurandosi che sia come ci si aspetta. -Inoltre il sistema dovrebbe accettare solo richieste di tipo HTTP e HTTPS, evitando invece quelle che sfruttano protocolli meno sicuri.
Comments	

Case Type	Abuse Case	Case ID	2
Case Name	Adversary in the middle (inserimento e lettura dati prodotto)		
Actors	Attaccante, sistema, trasformatore/fornitore/cliente		
Description	Un attaccante sfrutta la comunicazione tra utente e sistema per alterare le carbon footprint in entrata e in uscita. L'approccio più comune per questo attacco è quello di inserirsi all'interno del canale di comunicazione tra le due parti. Ogni qualvolta una delle due prova a comunicare con l'altra i dati raggiungono prima l'attaccante, che può modificarli.		
Data	Il canale di comunicazione su cui porsi. Il dato da modificare durante il transito.		

Stimulus and preconditions	Precondizioni: -L'attaccante deve essere in grado di riconoscere la natura e il meccanismo della comunicazione. -L'attaccante deve essere capace di porsi in mezzo al canale e intercettare i dati che sono trasmessi. -Non deve essere presente nessuna forma di forte autenticazione mutua tra le componenti, poiché ciò impedirebbe all'attaccante di frapporsi fra di esse. -La comunicazione deve essere eseguita in assenza di crittografia, o un sistema crittografico riconoscibile. Stimolo: Viene fatta partire una richiesta di lettura o di scrittura, che l'attaccante andrà quindi ad intercettare.
Attack Flow 1	1) L'attaccante studia il meccanismo di comunicazione, e scopre come funziona e come porsi in mezzo al canale. 2) L'attaccante si inserisce nel canale di comunicazione. 3) L'attaccante modifica i dati in scrittura o in lettura.
Attack Flow 2	
Attack Flow 3	
Response and Postconditions	I dati scritti nel registro non corrisponderanno più a quelli inseriti dai fornitori e dai trasformatori. Similmente, i dati ricevuti dai clienti non saranno quelli effettivamente presenti nel registro.
Non Functional Requirements	Autorizzazione, autenticità
Mitigations	-Controllare che le chiavi pubbliche siano firmate da un'autorità di certificazione. -Usare la crittografia nelle comunicazioni. -Scambiare le chiavi pubbliche in canali sicuri.
Comments	

Case Type	Abuse Case	Case ID	3
Case Name	Brute Force (dati per l'autenticazione)		
Actors	Attaccante, sistema		
Description	Un attaccante cerca di accedere al sistema attraverso tentativi "trial and error" sperando di indovinare delle credenziali corrette		
Data	Procedura di autenticazione, dati necessari per l'autenticazione		
Stimulus and preconditions	Precondizioni: l'attaccante deve possedere dei metodi automatici per generare tutti i possibili valori da utilizzare nella fase di "trial and error". L'attaccante deve essere in grado di determinare quando ha individuato le credenziali corrette.		

Attack Flow 1	1) L'attaccante deve determinare come è possibile testare la correttezza di un possibile valore provato. 2) L'attaccante deve trovare un modo per diminuire il più possibile lo spazio di ricerca. Più piccolo è lo spazio di ricerca maggiore è la possibilità di indovinare le credenziali. 3) L'attaccante aumenta le possibilità di individuare le credenziali corrette attraverso varie tecniche 4) L'attaccante tenta tutte le possibili combinazioni di credenziali fino ad ottenere quelle giuste
Attack Flow 2	
Attack Flow 3	
Response and Postconditions	Il sistema autentica l'attaccante, di conseguenza l'attaccante acquisisce i privilegi dell'utente a cui ha sottratto i dati.
Non Functional Requirements	Confidenzialità
Mitigations	- scegliere uno spazio di ricerca molto ampio, conosciuto e senza pattern noti che permettano di ridurre la dimensione - limitare il numero di tentativi di accesso per delle determinate credenziali
Comments	

Case Type	Abuse Case	Case ID	4
Case Name	Falsificare la fonte dei dati (inserimento dati prodotto, carbon footprint parziali)		
Actors	Attaccante, sistema, fornitore/trasformatore		
Description	Un attaccante riesce a prendere le veci di un altro utente per inserire dati falsi nel registro. Questo può essere eseguito, per esempio, entrando in possesso delle sue credenziali e autenticandosi con esse.		
Data	Le credenziali di autenticazione.		
Stimulus and preconditions	Precondizione: l'attaccante deve essere in grado di mascherarsi per un utente fornitore o trasformatore. Stimolo: L'attaccante impersona un fornitore/trasformatore inserendo un dato falso		
Attack Flow 1	1) L'attaccante identifica gli utenti vulnerabili al furto di credenziali. 2) L'attaccante ottiene le credenziali di un utente fornitore o trasformatore. 3) L'attaccante esegue l'accesso con quelle credenziali 4) L'attaccante inserisce nel registro un dato fasullo.		
Attack Flow 2			
Attack Flow 3			

Response and Postconditions	L'attaccante, mascherandosi per un utente con l'autorizzazione ad inserire i dati sulla carbon footprint dei prodotti, avrà quindi inserito nel registro un dato falso riguardante un prodotto esistente, o un dato riguardante un prodotto inesistente.
Non Functional Requirements	Autenticità
Mitigations	-Sfruttare tecniche di autenticazione a più passaggi, come il 2FA
Comments	

Case Type	Abuse Case	Case ID	5
Case Name	Session Hijacking (inserimento dati prodotto)		
Actors	Attaccante, sistema, fornitore/trasformatore		
Description	Un attaccante riesce a rubare o manipolare la sessione attiva di un altro utente, per ottenere un accesso non autorizzato al sistema e la possibilità di inserire dati falsi.		
Data	La sessione dell'utente fornitore/trasformatore.		
Stimulus and preconditions	Precondizioni: - Il sistema deve sfruttare le sessioni per gestire le autenticazioni - La vittima deve avere una sessione attiva - A dipendenza della modalità che l'attaccante decide di utilizzare per mettere in atto questo attacco, specifiche precondizioni devono essere soddisfatte, come, per esempio, l'utilizzo di token di sessione intercettabili o l'utilizzo di ID di sessione da parte del sistema. Stimolo: L'attaccante ruba o manipola la sessione e ottiene la possibilità di scrivere sul registro		
Attack Flow 1	1) L'attaccante individua un traffico di rete non criptato in cui viene scambiato un token di sessione. 2) L'attaccante cattura il token di sessione con tool di sniffing appositi. 3) L'attaccante inserisce il token catturato nella comunicazione con il sistema. 4) L'attaccante sfrutta la sua sessione rubata per inserire dati fasulli.		
Attack Flow 2	1) L'attaccante scopre che vengono utilizzati identificativi di sessione per autenticare gli utenti. 2) L'attaccante ruba l'ID di sessione da un utente fornitore o trasformatore 3) L'attaccante sfrutta l'ID rubato per scrivere dati fasulli sul registro.		
Attack Flow 3			
Response and Postconditions	L'attaccante, avendo ottenuto la sessione di un utente autorizzato a inserire dati sul registro, sarà riuscito ad inserire un dato falso riguardante un prodotto esistente, o un dato riguardante un prodotto inesistente.		

Non Functional Requirements	Autorizzazione
Mitigations	- Criptare i token di sessione durante le comunicazioni - Sfruttare un meccanismo di session timeout per chiudere le sessioni anche quando l'utente non esegue il logout. - Quando lo stesso utente si ri-autentica la sessione generata deve essere differente. - Sfruttare protocolli protetti per comunicare
Comments	

Case Type	Abuse Case	Case ID	6
Case Name	Command Injection (carbon footprint materia prima, trasformazione e prodotto finito)		
Actors	Attaccante, sistema		
Description	L'attaccante cerca di eseguire un comando arbitrario, attraverso l'inserimento di nuovi elementi in un comando già esistente, modificando così il comportamento inteso.		
Data			
Stimulus and preconditions	Precondizioni: - il sistema deve prevedere delle funzioni che costruiscano dei comandi usando l'input dell'utente		
Attack Flow 1	1) L'attaccante fa un inventario di tutte le funzionalità esposte dall'applicazione 2) L'attaccante identifica quali input sono suscettibili all'iniezione di comandi 3) L'attaccante identifica la struttura dei comandi che vengono eseguiti utilizzando gli input forniti 4) L'attaccante esegue comandi arbitrari sul sistema		
Attack Flow 2			
Attack Flow 3			
Response and Postconditions	L'attaccante può alterare i dati presenti nel sistema, violando la loro integrità.		
Non Functional Requirements	Integrità dei dati		
Mitigations	- sanificare sempre tutti gli input provenienti dall'utente - codificare gli input prima di essere usati come parte di un comando - parametrizzare gli input		
Comments			

Case Type	Abuse Case	Case ID	7
Case Name	Flooding (inserimento e lettura dati prodotto)		
Actors	Attaccante, sistema		
Description	L'attaccante esaurisce le risorse del sistema attraverso un grande numero di interazioni con esso. Impedendo ad altri utenti di poter usufruire di quei servizi.		
Data			
Stimulus and preconditions	Precondizioni: - il sistema deve rispondere alle richieste degli utenti - l'attaccante deve avere un programma per generare più richieste di quante il sistema ne possa gestire - in alternativa, l'attaccante può avere un network o cluster di oggetti capaci di effettuare richieste simultanee Stimolo: l'attaccante utilizza uno strumento a sua disposizione per effettuare un gran numero di richieste		
Attack Flow 1	1) L'attaccante esplora le possibili funzioni offerte dal sistema 2) L'attaccante sceglie la funzione più facile da attaccare e/o quella computazionalmente più pesante da gestire per il sistema 3) L'attaccante sfrutta lo script creato per richiedere molteplici invocazioni della funzione scelta		
Attack Flow 2	1) L'attaccante esplora le possibili funzioni offerte dal sistema 2) L'attaccante sceglie la funzione più facile da attaccare e/o quella computazionalmente più pesante da gestire per il sistema 3) L'attaccante comunica al network o cluster, il bersaglio a cui effettuare le richieste simultaneamente		
Attack Flow 3			
Response and Postconditions	L'attaccante riesce a consumare le risorse del sistema, causando un rallentamento o un blocco dei servizi offerti.		
Non Functional Requirements	Disponibilità		
Mitigations	- Assicurarsi che il protocollo implementi dei limiti sulle richieste - Specificare le aspettative per le capacità e dettare quali comportamenti sono accettabili quando l'allocazione delle risorse ha raggiunto il limite - gestire uniformemente le richieste per rendere più difficile consumare risorse più velocemente di quanto possano essere liberate		
Comments			

Case Type	Abuse Case	Case ID	8
Case Name	Session Credential Falsification through Forging (inserimento dati prodotto)		
Actors	Attaccante, sistema		
Description	L'attaccante crea delle false ma funzionali credenziali di sessione al fine di accedere ad un servizio. Attraverso questo metodo l'attaccante potrebbe bypassare l'autenticazione o sfruttare la sessione di un altro utente autenticato.		
Data			
Stimulus and preconditions	Precondizioni: - l'applicazione target deve usare credenziali di sessione per identificare gli utenti legittimi - le credenziali di sessione non devono cambiare quando cambia il livello di privilegi - gli identificatori di sessione devono essere prevedibili - l'attaccante potrebbe avere un programma per inviare messaggi con le credenziali da lui create Stimolo: - l'attaccante utilizza gli strumenti a sua disposizione per creare e utilizzare credenziali di sessione		
Attack Flow 1	1) L'attaccante scopre che l'applicazione obiettivo utilizza credenziali di sessione per identificare gli utenti legittimi 2) L'attaccante crea messaggi contenenti l'id di sessione (nelle richieste GET o POST, negli header HTTP o nei cookie) 3) Una volta che la vittima si è autenticata e ha quindi ottenuto un livello di privilegio più alto, l'attaccante può prendere il controllo della sessione utilizzando l'id di sessione da lui generato		
Attack Flow 2			
Attack Flow 3			
Response and Postconditions	L'attaccante assume l'identità di un utente registrato, bypassando il sistema di autenticazione e ottenendo privilegi più elevati.		
Non Functional Requirements	Responsabilità, autorizzazione		
Mitigations	- Usare id di sessione che sono difficili da indovinare: devono essere sufficientemente randomici - rigenerare e distruggere gli id di sessione quando si verifica un cambio di privilegi		
Comments			

Case Type	Abuse Case	Case ID	9
Case Name	Code injection (inserimento e lettura dati prodotto, autenticazione)		
Actors	Attaccante, sistema		
Description	Un attaccante sfrutta una debolezza nella validazione degli input per iniettare nuovo codice all'interno di quello attualmente in esecuzione		
Data	Lo script malevolo da iniettare.		
Stimulus and preconditions	Precondizioni: - Il sistema non deve eseguire validazione degli input, o questa deve essere inefficace. - Il sistema deve mettere in esecuzione degli script Stimolo: l'attaccante inserisce uno script malevolo all'interno di uno script pre-esistente del sistema		
Attack Flow 1	1) L'attaccante esplora gli input del sistema mediante tecniche di Spidering. 2) L'attaccante usa i punti di ingresso individuati nella prima fase e inietta diversi script non malevoli tentando di capire se quegli script vengono eseguiti, e quindi, quel particolare canale di input possiede delle vulnerabilità. A quel punto capisce come poterle sfruttare. 3) L'attaccante riesce a sfruttare una vulnerabilità trovata iniettando codice malevolo all'interno del codice pre-esistente del sistema.		
Attack Flow 2			
Attack Flow 3			
Response and Postconditions	Il sistema contiene al suo interno del codice malevolo che può impedirne il funzionamento o, ancor peggio, modificare il modo in cui il sistema svolge le sue funzioni.		
Non Functional Requirements	Affidabilità, Disponibilità		
Mitigations	- Assicurarsi che tutti gli input del sistema siano sanificati correttamente. - Validare l'input per tutti i contenuti - Aggiornare e correggere costantemente il software.		
Comments			

Case Type	Abuse Case	Case ID	10
Case Name	Buffer overflow (procedura di calcolo della carbon footprint complessiva)		
Actors	Attaccante, sistema		
Description	L'attaccante manipola l'interazione di un'applicazione con un buffer nel tentativo di modificare dati a cui non dovrebbe poter accedere o eseguire funzioni in maniera non autorizzata.		
Data			
Stimulus and preconditions	Precondizioni: - l'applicazione target deve utilizzare un linguaggio di programmazione vulnerabile alla manipolazione del buffer (C/C++) - l'applicazione target esegue in maniera non adeguata il controllo dei limiti dei buffer Stimolo: - l'attaccante identifica un buffer nell'applicazione target		
Attack Flow 1	1) l'attaccante esplora le possibili funzioni offerte dal sistema 2) l'attaccante sceglie una funzione che utilizza un buffer e che accetti input dall'utente 3) l'attaccante trova un canale di input dove inviare un contenuto che ecceda la dimensione del buffer 4) l'attaccante costruisce il contenuto da iniettare. Il contenuto deve essere costituito di dati generati casualmente in quantità sufficiente per causare il crash del sistema 5) utilizzando l'input vulnerabile, l'attaccante inietta il contenuto da lui generato		
Attack Flow 2	1) l'attaccante esplora le possibili funzioni offerte dal sistema 2) l'attaccante sceglie una funzione che utilizza un buffer e che accetti input dall'utente 3) l'attaccante trova un input dove inviare un contenuto che ecceda la dimensione del buffer 4) l'attaccante costruisce il contenuto da iniettare. Il contenuto deve essere generato in maniera tale da sovrascrivere l'indirizzo di return con uno scelto dall'attaccante 5) utilizzando l'input vulnerabile, l'attaccante inietta il contenuto da lui generato		
Attack Flow 3			
Response and Postconditions	Una procedura o un servizio del sistema opera in una maniera non prevista		
Non Functional Requirements	Disponibilità		

Mitigations	- Utilizzare un linguaggio o compilatore che limita le capacità del programmatore di agire oltre i limiti del buffer - Se si utilizzano funzioni non sicure, assicurarsi che si controlli adeguatamente i limiti dei buffer - Utilizzare meccanismi basati sul compilatore (analisi statica del codice) che identificano e risolvono potenziali problemi legati ai buffer
Comments	

Case Type	Abuse Case	Case ID	11
Case Name	Sfruttare livelli di sicurezza del controllo degli accessi non correttamente configurati (Procedura di inserimento di un'operazione, Procedura di calcolo della carbon footprint complessiva)		
Actors	Attaccante, sistema		
Description	Un attaccante sfrutta un punto debole nella configurazione dei controlli di accesso ed è in grado di aggirare la protezione prevista da queste misure e ottenere così l'accesso non autorizzato al sistema.		
Data			
Stimulus and preconditions	Precondizione: il sistema di configurazione dei controlli di accesso deve essere presente ma mal configurato Stimolo: l'attaccante ottiene l'accesso a funzionalità o dati che dovrebbero essere protetti		
Attack Flow 1	1) L'attaccante esamina l'applicazione di destinazione, possibilmente come utente valido e autenticato, oppure utilizzando attacchi di session hijacking. 2) L'attaccante sonda il controllo degli accessi a funzioni e dati identificati nella prima fase per identificare potenziali punti deboli nel modo in cui sono configurati i controlli degli accessi. 3) L'attaccante esegue la funzione o accede ai dati individuati nella prima fase aggirando il controllo degli accessi.		
Attack Flow 2			
Attack Flow 3			
Response and Postconditions	L'attaccante ottiene la possibilità di eseguire operazioni a lui illecite, come la scrittura di dati fasulli o la modifica di dati già presenti.		
Non Functional Requirements	Autorizzazione		
Mitigations	Configurare il sistema di controllo degli accessi correttamente.		
Comments			

Case Type	Abuse Case	Case ID	12
Case Name	Manipulating state (procedura di calcolo della carbon footprint complessiva)		
Actors	Attaccante, sistema		
Description	L'attaccante modifica informazioni sullo stato del sistema oppure causa una transizione di stato nell'hardware. Se l'attacco ha successo, il sistema eseguirà i servizi nello stato modificato in una maniera non prevista.		
Data			
Stimulus and preconditions	Prerequisiti: - gli stati degli utenti sono mantenuti in una locazione accessibile al livello degli utenti (ad esempio cookies o parametri URL) - è presente un problema di progettazione nella logica hardware - l'attaccante necessita un software che gli permetta di generare e creare input che aiutino nell'attacco		
Attack Flow 1	1) L'attaccante analizza la natura di gestione degli stati utilizzata dal sistema: determina la locazione e i possibili elementi che vengono salvati come parte dello stato degli utenti 2) L'attaccante prova quindi a modificare i contenuti nello stato degli utenti (indipendentemente dal fatto che siano offuscati o criptati) oppure causa una transizione di stato 3) L'attaccante osserva gli effetti derivanti dalla sua modifica 4) Avendo determinato come manipolare gli stati, l'attaccante può ora effettuare azioni illegittime		
Attack Flow 2			
Attack Flow 3			
Response and Postconditions	Una procedura o un servizio del sistema opera in una maniera non prevista.		
Non Functional Requirements	Affidabilità		
Mitigations	- non affidarsi solamente a locazioni che sono controllabili dall'utente per mantenere gli stati dell'utente - evitare di inserire informazioni sensibili, quali username, informazioni di autenticazione, nelle locazioni controllabili dall'utente - le informazioni sensibili dell'utente che fanno parte dello stato devono essere appropriatamente offuscate e/o criptate in modo da garantire la confidenzialità e l'integrità delle stesse - tutti i possibili stati devono essere gestiti dall'automa a stati finiti		
Comments			

Infine, i casi di misuso sono i seguenti:

Case Type	Misuse Case	Case ID	0
Case Name	Inserimento di dati con caratteristiche problematiche		
Actors	Fornitore, trasformatore, sistema		
Description	Durante la procedura di inserimento il fornitore/trasformatore inserisce un dato che non viene gestito in maniera corretta dal sistema.		
Data	Il dato inserito.		
Stimulus and preconditions	Precondizioni: -Il sistema non compie una corretta validazione degli input -Il fornitore/trasformatore non sa come formattare l'input correttamente Stimolo: il fornitore/trasformatore inserisce un dato non correttamente formattato		
Attack Flow 1	Il fornitore/trasformatore inserisce una carbon footprint negativa		
Attack Flow 2	Il fornitore/trasformatore inserisce una carbon footprint eccessivamente elevata che non viene gestita correttamente		
Attack Flow 3	Il fornitore/trasformatore inserisce una carbon footprint che presenta caratteri speciali che interferiscono con il normale funzionamento della procedura		
Response and Postconditions	La procedura di inserimento non viene eseguita nella maniera prestabilita, oppure il calcolo della carbon footprint complessiva potrebbe generare un valore scorretto o potrebbe anche portare ad un errore. Questo può portare ad un funzionamento scorretto o al totale stop del funzionamento del sistema		
Non Functional Requirements	Reliabilty, disponibilità, autenticità del dato		
Mitigations	- Effettuare un controllo sul dato inserito - Sanificare l'input dell'utente - Specificare all'utente le caratteristiche che il dato in input deve avere.		
Comments			

Case Type	Misuse Case	Case ID	1
Case Name	Errata segnalazione di conclusione del processo di trasformazione		
Actors	Trasformatore, sistema		
Description	Durante la procedura di inserimento il trasformatore comunica per errore troppo presto al sistema di aver terminato il processo di trasformazione del prodotto.		
Data			
Stimulus and preconditions	Precondizione: il sistema non prevede richieste di conferma per tale azione. Stimolo: il trasformatore seleziona per errore di aver inserito l'ultimo dato utile.		
Attack Flow 1	1) Il trasformatore segnala per errore di aver concluso l'inserimento dei dati per il processo di trasformazione del prodotto in corso 2) Il sistema esegue il calcolo della carbon footprint complessiva prima del tempo 3) Il sistema salva sul registro il risultato prematuro		
Attack Flow 2			
Attack Flow 3			
Response and Postconditions	Il sistema si ritrova ad avere salvato sul registro un valore di carbon footprint complessiva che non corrisponde a quella reale. Il sistema è incapace di segnalare l'errore.		
Non Functional Requirements	Autenticità del dato		
Mitigations	- L'interfaccia del sistema deve richiedere una conferma all'utente quando questo seleziona di aver concluso il processo di inserimento di dati parziali prima di permettere che questi dati vengano spediti al sistema.		
Comments			

Case Type	Misuse Case	Case ID	2
Case Name	Input mal formattato		
Actors	Cliente/trasformatore/fornitore, sistema.		
Description	Il cliente, quando esegue una query di ricerca dei prodotti la formatta in modo tale da creare problemi al sistema. Per esempio, l'utilizzo di alcune tipologie di caratteri, se non gestito, può essere mal interpretato. La stessa cosa può accadere con la procedura di autenticazione di un utente fornitore o trasformatore.		
Data	L'input inserito		
Stimulus and preconditions	Precondizioni: il sistema non esegue sanificazione e validazione degli input Stimoli: -il cliente inserisce una query non correttamente formattato - Il fornitore o il trasformatore inserisce dell'input mal formattato nella form di autenticazione		
Attack Flow 1	1) Il cliente, nello scrivere la sua query di ricerca, inserisce caratteri speciali che il sistema non sa gestire. 2) Il sistema, quando va ad interpretare la query, la gestisce erroneamente per via della presenza di questi caratteri		
Attack Flow 2	1) Il fornitore o il trasformatore, nell'inserire le credenziali di input, inserisce caratteri speciali che il sistema non sa gestire. 2) Il sistema, quando va ad interpretare l'input inserito, va incontro ad errori per via della presenza di questi caratteri		
Attack Flow 3			
Response and Postconditions	A dipendenza dei caratteri inseriti, il sistema potrebbe rispondere al cliente fornendo una risposta scorretta. In casi estremi ciò potrebbe anche portare al crash del sistema		
Non Functional Requirements	Reliability, disponibilità		
Mitigations	- Il sistema deve validare correttamente e, se necessario, sanificare l'input dell'utente		
Comments			

Case Type	Misuse Case	Case ID	3
Case Name	Inserimento di carbon footprint parziali per un prodotto segnalato come concluso		
Actors	Trasformatore/fornitore, sistema.		
Description	Un trasformatore o un fornitore possono erroneamente inserire valori parziali della carbon footprint relative ad un prodotto che era stato precedentemente identificato come finito.		
Data	Il nuovo dato da inserire, i dati del prodotto finito.		
Stimulus and preconditions	Precondizioni: il sistema non esegue controlli sullo stato dei prodotti Stimolo: il fornitore o il trasformatore inserisce un dato relativo a un prodotto già finito		
Attack Flow 1	1) Il fornitore o il trasformatore inserisce un dato relativo a un prodotto già finito 2) Il sistema, che considera il dato come già concluso, farà partire un nuovo calcolo della carbon footprint completa 3) Il risultato andrà a sovrascrivere il precedente, che era quello corretto.		
Attack Flow 2			
Attack Flow 3			
Response and Postconditions	Il dato ora presente nel registro non rappresenterà la reale carbon footprint del prodotto.		
Non Functional Requirements	Integrità del dato		
Mitigations	- Il sistema deve controllare se un prodotto è stato già segnalato come concluso prima di permettere un nuovo inserimento. Se questo è il caso deve bloccare l'utente.		
Comments			

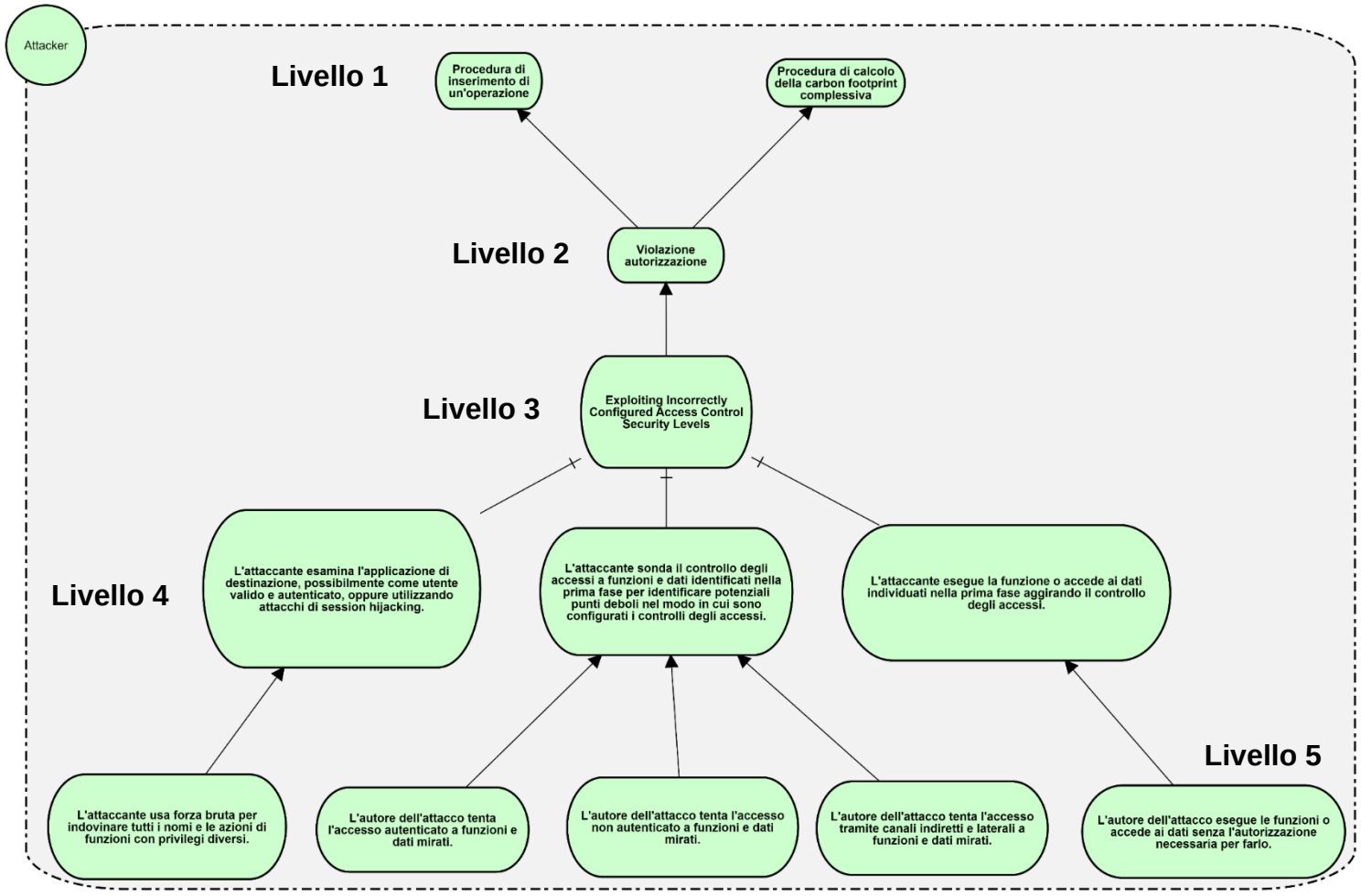
Realizzazione degli Attack Trees

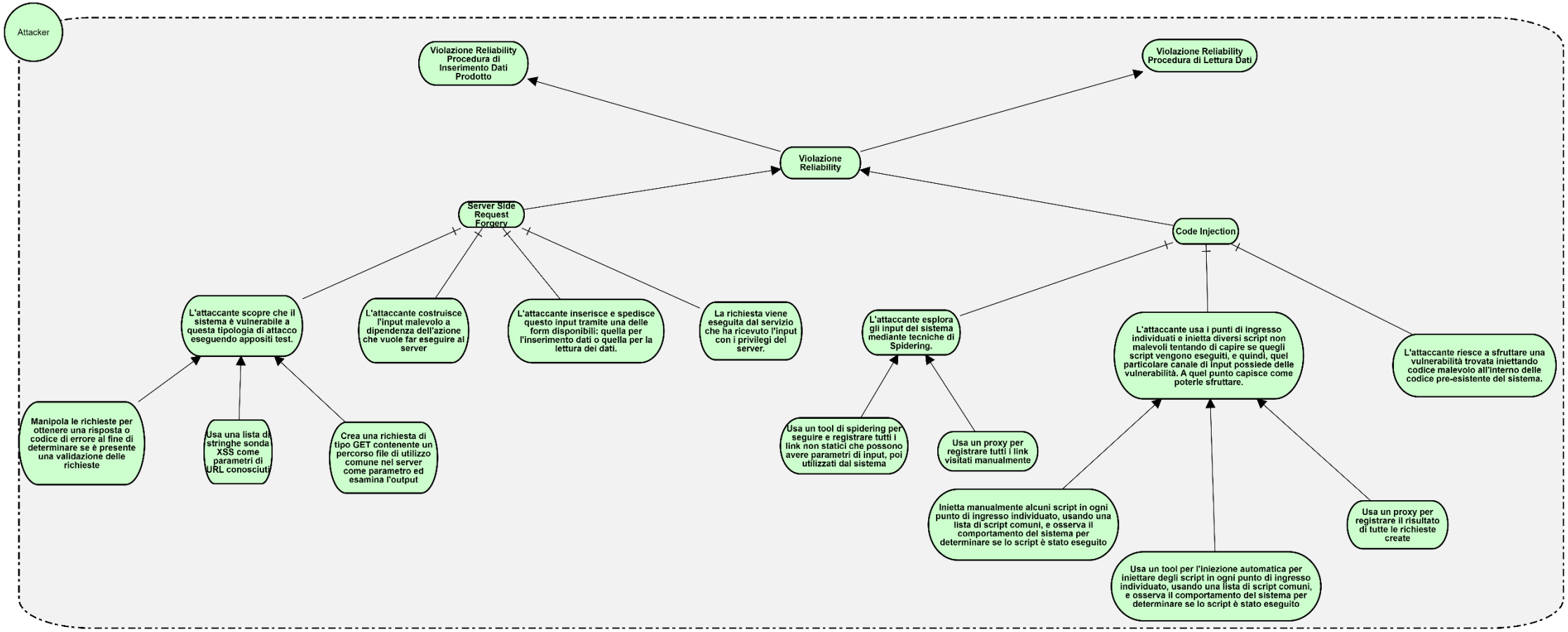
Nell'ultimo passaggio della fase di analisi del rischio sono stati creati gli **Attack Tree**. Un Attack Tree è un diagramma che rappresenta le varie modalità rilevate durante le analisi precedenti con cui un attaccante può arrivare a violare le policy di sicurezza. Esso, nel caso più semplice, considerato un singolo attacco, che può violare una o più policy di sicurezza stabilite per un dato asset, riesce a decomporre l'attacco nelle singole fasi, individuando tutte le possibili modalità di realizzazione dell'attacco. La costruzione degli Attack Tree è stata sviluppata sempre mediante diagrammi i*, che permettono di evidenziare al meglio quali sono le fasi dell'attacco e quali sono invece i modi diversi di implementazione

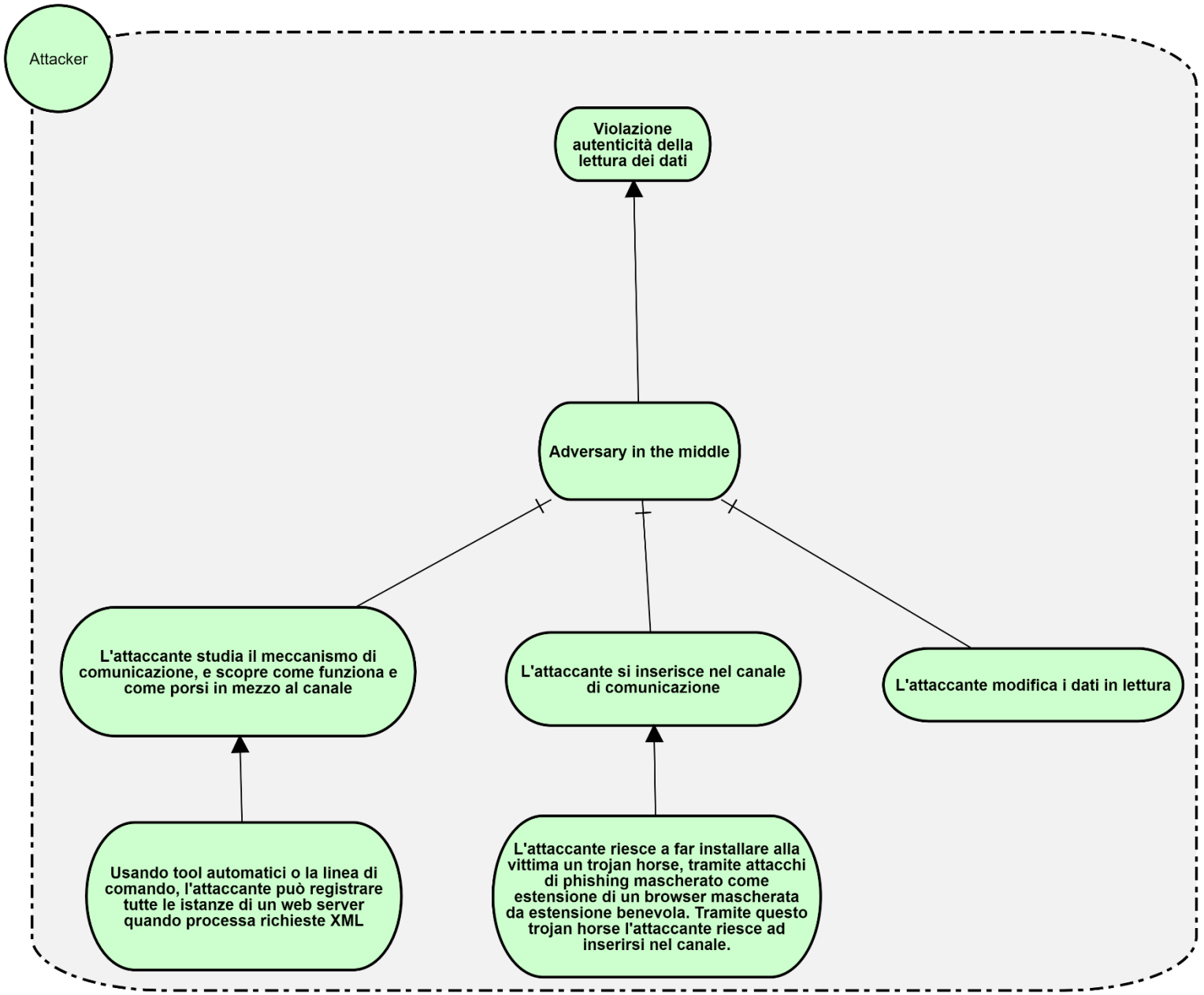
dell'attacco, mediante delle semplici decomposizioni AND e OR. Chiaramente in un Attack Tree per convenzione si possono raggruppare anche più attacchi, che violano più policy di sicurezza per più asset. Per chiarire come si legge letti un Attack Tree, si considera come esempio uno di quelli costruiti nell'analisi del rischio eseguita, quello che rappresenta possibili violazioni di autorizzazione delle procedure di calcolo della carbon footprint e inserimento di un'operazione. Nell'Attack Tree sono stati aggiunti dei numeri per indicare i vari livelli di lettura, che sono numerati:

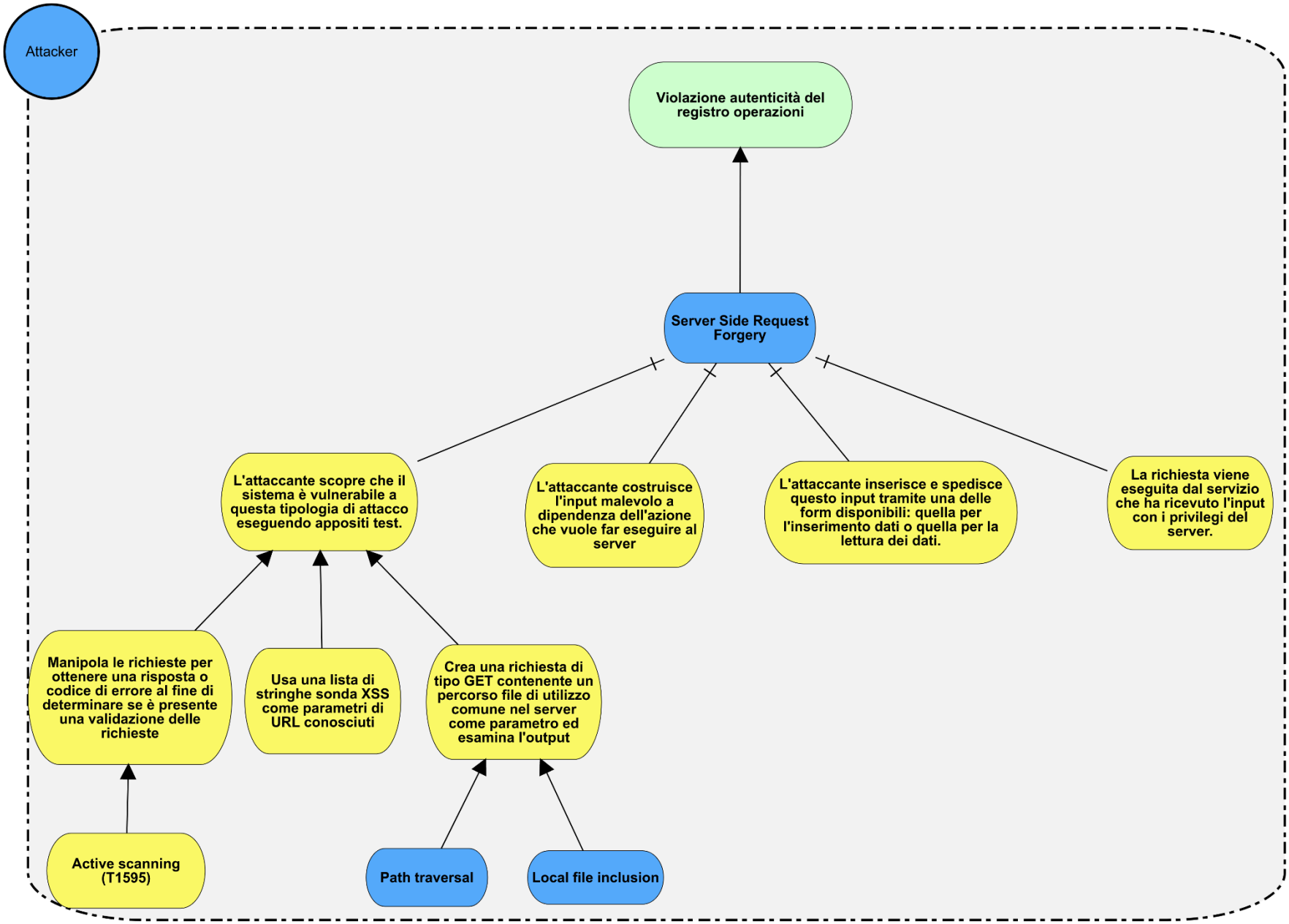
- **Livello 1:** Gli asset di cui vengono violate le policy
- **Livello 2:** Le singole policy che vengono violate, dagli attacchi
- **Livello 3:** I possibili attacchi eseguibili per violare la policy
- **Livello 4:** Le azioni da compiere, per eseguire gli attacchi al livello superiore. Esse devono essere eseguite tutte per portare a compimento l'attacco, in un ordine che è visibile nella voce corrispondente all'attacco nel catalogo CAPEC®
- **Livello 5:** Le possibili tecniche che attaccante può usare per compiere le azioni esplicitate al livello precedente

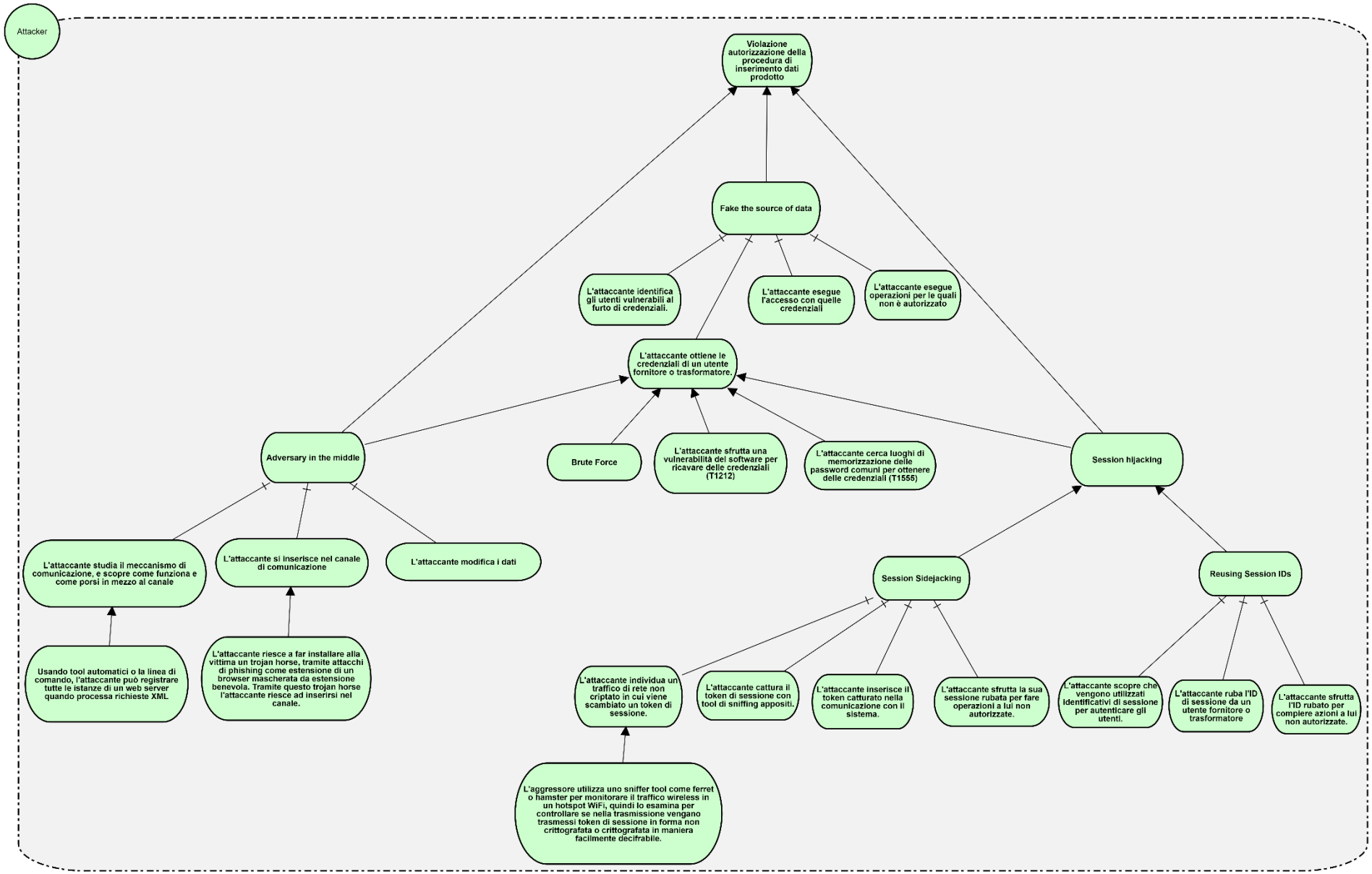
Non in tutti gli Attack Tree costruiti sono sempre presenti tutti questi livelli: per esempio in alcuni il primo e il secondo sono unificati, perché la violazione della policy riguarda solo un asset. Oltre all'Attack Tree in cui è presente la legenda descritta sopra di seguito si presentano i restanti Attack Tree prodotti in fase di analisi del rischio, ovvero tutti quelli riferiti ad attacchi per cui è stato possibile identificare in maniera precisa un possibile flusso di attacco.

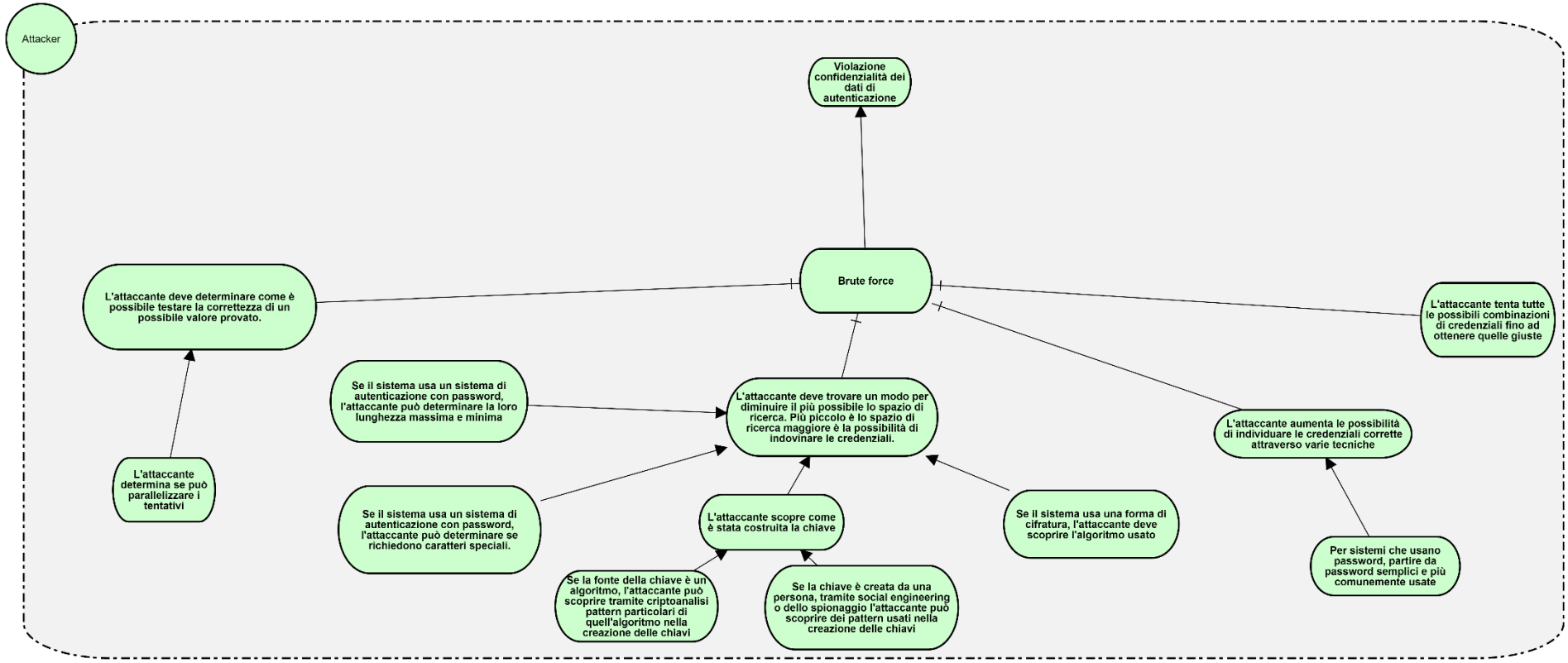


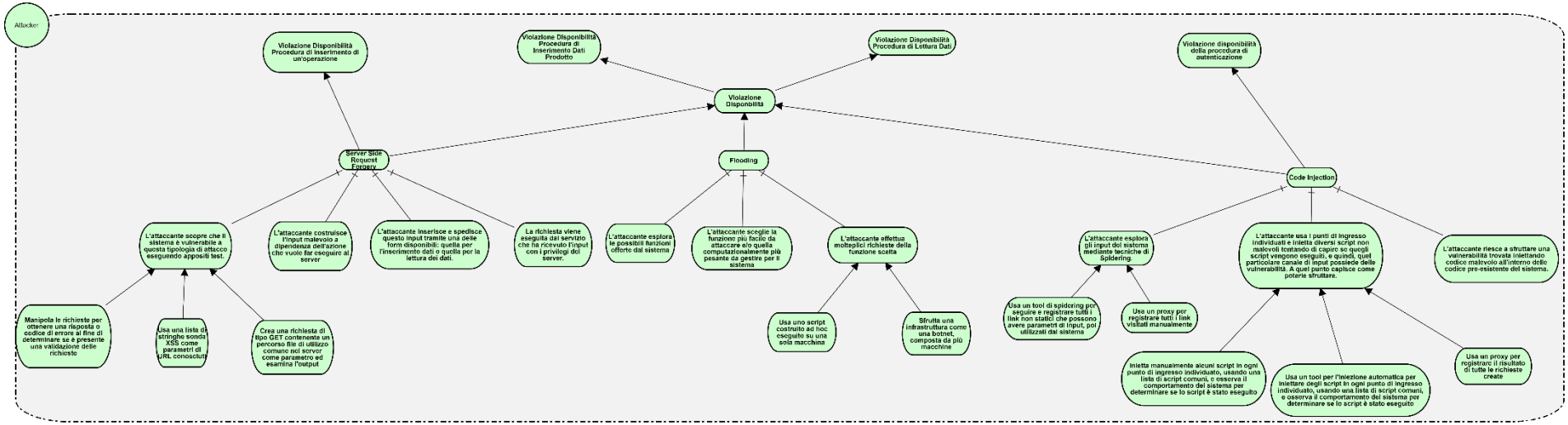

Attacker
Violazione Reliability Procedura di Inserimento Dati Prodotto
Violazione Reliability Procedura di Lettura Dati
Violazione Reliability
Server Side Request Forgery
Code Injection
L'attaccante scopre che il sistema è vulnerabile a questa tipologia di attacco eseguendo appositi test.
L'attaccante costruisce l'input malevolo a dipendenza dell'azione che vuole far eseguire al server
L'attaccante inserisce e spedisce questo input tramite una delle form disponibili; quella per l'inserimento dati o quella per la lettura dei dati.
La richiesta viene eseguita dal servizio che ha ricevuto l'input con i privilegi del server.
Manipola le richieste per ottenere una risposta o codice di errore al fine di determinare se è presente una validazione delle richieste
Usa una lista di stringhe sonda XSS come parametri di URL conosciuti
Crea una richiesta di tipo GET contenente un percorso file di utilizzo comune nel server come parametro ed esamina l'output
L'attaccante esplora gli input del sistema mediante tecniche di Spidering.
L'attaccante usa i punti di ingresso individuati e inietta diversi script non malevoli tentando di capire se quegli script vengono eseguiti, e quindi, quel particolare canale di input possiede delle vulnerabilità. A quel punto capisce come poterle sfruttare.
L'attaccante riesce a sfruttare una vulnerabilità trovata iniettando codice malevolo all'interno delle codice pre-esistente del sistema.
Usa un tool di spidering per seguire e registrare tutti i link non statici che possono avere parametri di input, poi utilizzati dal sistema
Usa un proxy per registrare tutti i link visitati manualmente
Inietta manualmente alcuni script in ogni punto di ingresso individuato, usando una lista di script comuni, e osserva il comportamento del sistema per determinare se lo script è stato eseguito
Usa un tool per l'iniezione automatica per iniettare degli script in ogni punto di ingresso individuato, usando una lista di script comuni, e osserva il comportamento del sistema per determinare se lo script è stato eseguito
Usa un proxy per registrare il risultato di tutte le richieste create

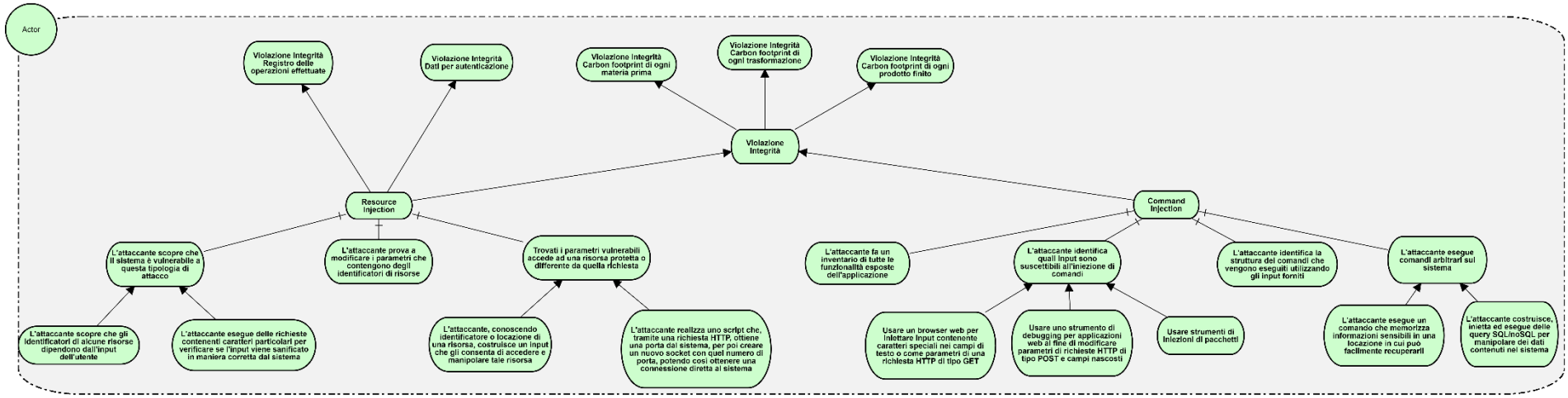

Attacker
Violazione autenticità della lettura dei dati
Adversary in the middle
L'attaccante studia il meccanismo di comunicazione, e scopre come funziona e come porsi in mezzo al canale
L'attaccante si inserisce nel canale di comunicazione
L'attaccante modifica i dati in lettura
Usando tool automatici o la linea di comando, l'attaccante può registrare tutte le istanze di un web server quando processa richieste XML
L'attaccante riesce a far installare alla vittima un trojan horse, tramite attacchi di phishing mascherato come estensione di un browser mascherata da estensione benevola. Tramite questo trojan horse l'attaccante riesce ad inserirsi nel canale.

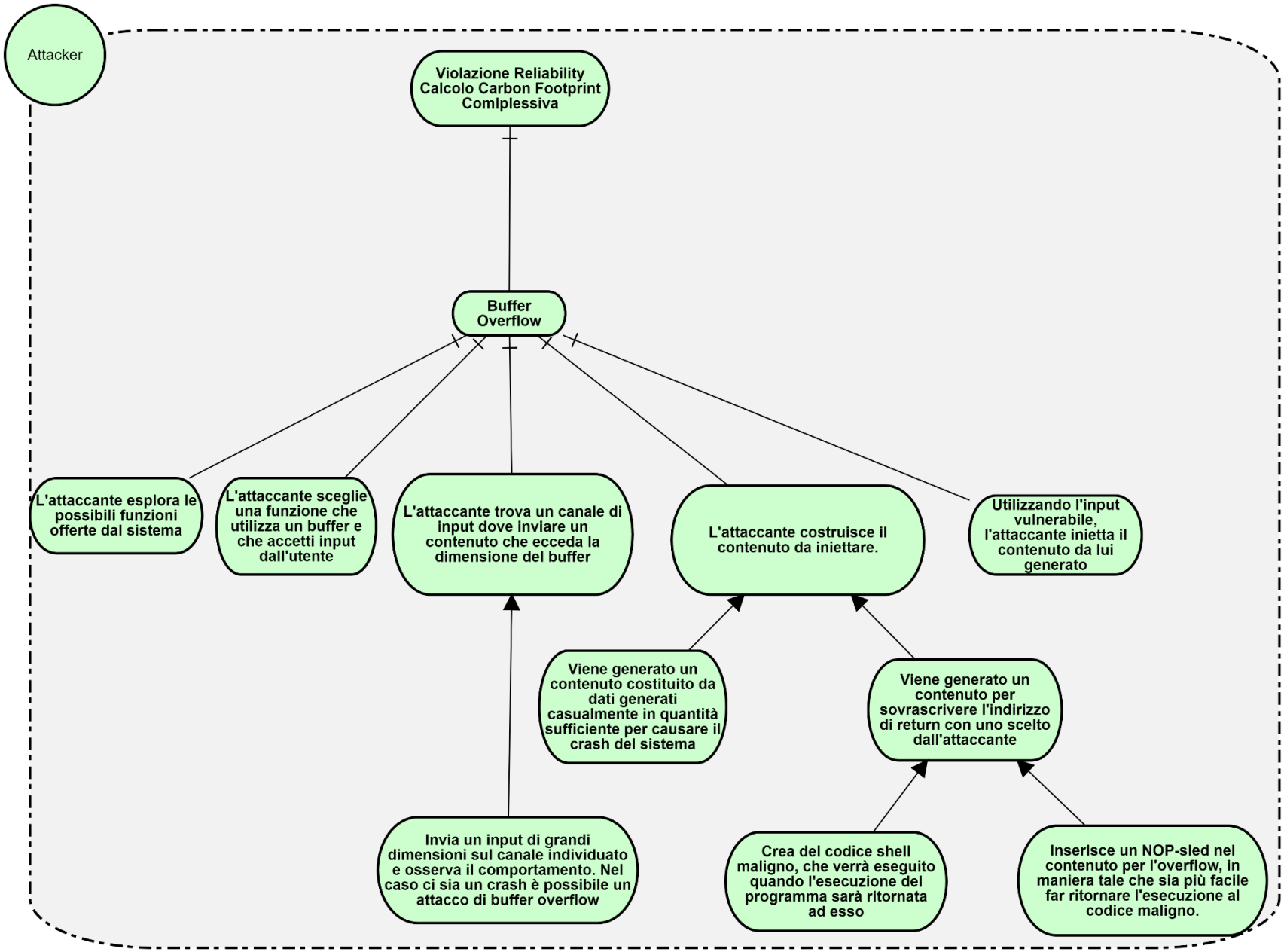

Attacker
Violazione autenticità del registro operazioni
Server Side Request Forgery
L'attaccante scopre che il sistema è vulnerabile a questa tipologia di attacco eseguendo appositi test.
L'attaccante costruisce l'input malevolo a dipendenza dell'azione che vuole far eseguire al server
L'attaccante inserisce e spedisce questo input tramite una delle form disponibili: quella per l'inserimento dati o quella per la lettura dei dati.
La richiesta viene eseguita dal servizio che ha ricevuto l'input con i privilegi del server.
Manipola le richieste per ottenere una risposta o codice di errore al fine di determinare se è presente una validazione delle richieste
Usa una lista di stringhe sonda XSS come parametri di URL conosciuti
Crea una richiesta di tipo GET contenente un percorso file di utilizzo comune nel server come parametro ed esamina l'output
Active scanning (T1595)
Path traversal
Local file inclusion

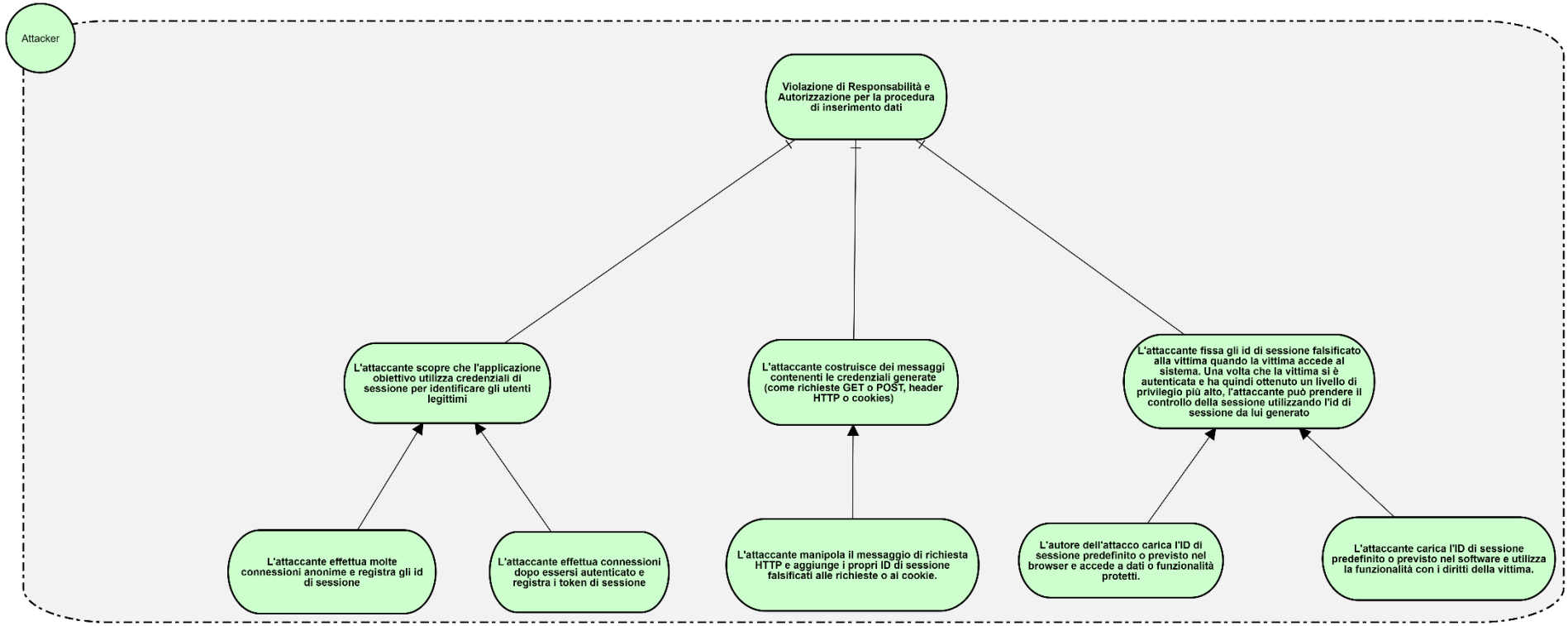

Attacker
Violazione autorizzazione della procedura di inserimento dati prodotto
Fake the source of data
L'attaccante identifica gli utenti vulnerabili al furto di credenziali.
L'attaccante esegue l'accesso con quelle credenziali
L'attaccante esegue operazioni per le quali non è autorizzato
L'attaccante ottiene le credenziali di un utente fornitore o trasformatore.
Adversary in the middle
Brute Force
L'attaccante sfrutta una vulnerabilità del software per ricavare delle credenziali (T1212)
L'attaccante cerca luoghi di memorizzazione delle password comuni per ottenere delle credenziali (T1555)
Session hijacking
L'attaccante studia il meccanismo di comunicazione, e scopre come funziona e come porsi in mezzo al canale
L'attaccante si inserisce nel canale di comunicazione
L'attaccante modifica i dati
Session Sidejacking
Reusing Session IDs
Usando tool automatici o la linea di comando, l'attaccante può registrare tutte le istanze di un web server quando processa richieste XML
L'attaccante riesce a far installare alla vittima un trojan horse, tramite attacchi di phishing come estensione di un browser mascherata da estensione benevola. Tramite questo trojan horse l'attaccante riesce ad inserirsi nel canale.
L'attaccante individua un traffico di rete non criptato in cui viene scambiato un token di sessione.
L'attaccante cattura il token di sessione con tool di sniffing appositi.
L'attaccante inserisce il token catturato nella comunicazione con il sistema.
L'attaccante sfrutta la sua sessione rubata per fare operazioni a lui non autorizzate.
L'attaccante scopre che vengono utilizzati identificativi di sessione per autenticare gli utenti.
L'attaccante ruba l'ID di sessione da un utente fornitore o trasformatore
L'attaccante sfrutta l'ID rubato per compiere azioni a lui non autorizzate.
L'aggressore utilizza uno sniffer tool come ferret o hamster per monitorare il traffico wireless in un hotspot WiFi, quindi lo esamina per controllare se nella trasmissione vengano trasmessi token di sessione in forma non crittografata o crittografata in maniera facilmente decifrabile.


Attacker
Violazione confidenzialità dei dati di autenticazione
Brute force
L'attaccante deve determinare come è possibile testare la correttezza di un possibile valore provato.
L'attaccante tenta tutte le possibili combinazioni di credenziali fino ad ottenere quelle giuste
L'attaccante deve trovare un modo per diminuire il più possibile lo spazio di ricerca. Più piccolo è lo spazio di ricerca maggiore è la possibilità di indovinare le credenziali.
Se il sistema usa un sistema di autenticazione con password, l'attaccante può determinare la loro lunghezza massima e minima
L'attaccante determina se può parallelizzare i tentativi
Se il sistema usa un sistema di autenticazione con password, l'attaccante può determinare se richiedono caratteri speciali.
L'attaccante scopre come è stata costruita la chiave
Se il sistema usa una forma di cifratura, l'attaccante deve scoprire l'algoritmo usato
L'attaccante aumenta le possibilità di individuare le credenziali corrette attraverso varie tecniche
Per sistemi che usano password, partire da password semplici e più comunemente usate
Se la fonte della chiave è un algoritmo, l'attaccante può scoprire tramite criptoanalisi pattern particolari di quell'algoritmo nella creazione delle chiavi
Se la chiave è creata da una persona, tramite social engineering o dello spionaggio l'attaccante può scoprire dei pattern usati nella creazione delle chiavi


Attacker
Violazione Disponibilità Procedura di inserimento di un'operazione
Violazione Disponibilità Procedura di Inserimento Dati Prodotto
Violazione Disponibilità Procedura di Lettura Dati
Violazione Disponibilità
Violazione disponibilità della procedura di autenticazione
Server Side Request Forgery
Flooding
Code Injection
L'attaccante scopre che il sistema è vulnerabile a questa tipologia di attacco eseguendo appositi test.
L'attaccante costruisce l'input malevolo a dipendenza dell'azione che vuole far eseguire al server
L'attaccante inserisce e spedisce questo input tramite una delle form disponibili: quella per l'inserimento dati o quella per la lettura dei dati.
La richiesta viene eseguita dal servizio che ha ricevuto l'input con i privilegi del server.
L'attaccante esplora le possibili funzioni offerte dal sistema
L'attaccante sceglie la funzione più facile da attaccare e/o quella computazionalmente più pesante da gestire per il sistema
L'attaccante effettua molteplici richieste della funzione scelta
L'attaccante esplora gli input del sistema mediante tecniche di Spidering.
L'attaccante usa i punti di ingresso individuati e inietta diversi script non malevoli tentando di capire se quegli script vengono eseguiti, e quindi, quel particolare canale di input possiede delle vulnerabilità. A quel punto capisce come poterle sfruttare.
L'attaccante riesce a sfruttare una vulnerabilità trovata iniettando codice malevolo all'interno dello codice pre-esistente del sistema.
Manipola le richieste per ottenere una risposta o codice di errore al fine di determinare se è presente una validazione delle richieste
Usa una lista di stringhe sonda XSS come parametri di URL conosciuti
Crea una richiesta di tipo GET contenente un percorso file di utilizzo comune nel server come parametro ed esamina l'output
Usa uno script costruito ad hoc eseguito su una sola macchina
Sfrutta una infrastruttura come una botnet, composta da più macchine
Usa un tool di spidering per seguire e registrare tutti i link non statici che possono avere parametri di input, poi utilizzati dal sistema
Usa un proxy per registrare tutti i link visitati manualmente
Inietta manualmente alcuni script in ogni punto di ingresso individuato, usando una lista di script comuni, e osserva il comportamento del sistema per determinare se lo script è stato eseguito
Usa un tool per l'iniezione automatica per iniettare degli script in ogni punto di ingresso individuato, usando una lista di script comuni, e osserva il comportamento del sistema per determinare se lo script è stato eseguito
Usa un proxy per registrare il risultato di tutte le richieste create


Actor
Violazione Integrità Registro delle operazioni effettuate
Violazione Integrità Dati per autenticazione
Violazione Integrità Carbon footprint di ogni materia prima
Violazione Integrità Carbon footprint di ogni trasformazione
Violazione Integrità Carbon footprint di ogni prodotto finito
Violazione Integrità
Resource Injection
Command Injection
L'attaccante scopre che il sistema è vulnerabile a questa tipologia di attacco
L'attaccante prova a modificare i parametri che contengono degli identificatori di risorse
Trovati i parametri vulnerabili accede ad una risorsa protetta o differente da quella richiesta
L'attaccante fa un inventario di tutte le funzionalità esposte dell'applicazione
L'attaccante identifica quali input sono suscettibili all'iniezione di comandi
L'attaccante identifica la struttura dei comandi che vengono eseguiti utilizzando gli input forniti
L'attaccante esegue comandi arbitrari sul sistema
L'attaccante scopre che gli identificatori di alcune risorse dipendono dall'input dell'utente
L'attaccante esegue delle richieste contenenti caratteri particolari per verificare se l'input viene sanificato in maniera corretta dal sistema
L'attaccante, conoscendo identificatore o locazione di una risorsa, costruisce un input che gli consenta di accedere e manipolare tale risorsa
L'attaccante realizza uno script che, tramite una richiesta HTTP, ottiene una porta dal sistema, per poi creare un nuovo socket con quel numero di porta, potendo così ottenere una connessione diretta al sistema
Usare un browser web per infettare input contenente caratteri speciali nei campi di testo o come parametri di una richiesta HTTP di tipo GET
Usare uno strumento di debugging per applicazioni web al fine di modificare parametri di richieste HTTP di tipo POST e campi nascosti
Usare strumenti di iniezioni di pacchetti
L'attaccante esegue un comando che memorizza informazioni sensibili in una locazione in cui può facilmente recuperarli
L'attaccante costruisce, inietta ed esegue delle query SQL/noSQL per manipolare dei dati contenuti nel sistema


Attacker
Violazione Reliability Calcolo Carbon Footprint Complessiva
Buffer Overflow
L'attaccante esplora le possibili funzioni offerte dal sistema
L'attaccante sceglie una funzione che utilizza un buffer e che accetti input dall'utente
L'attaccante trova un canale di input dove inviare un contenuto che ecceda la dimensione del buffer
L'attaccante costruisce il contenuto da iniettare.
Utilizzando l'input vulnerabile, l'attaccante inietta il contenuto da lui generato
Viene generato un contenuto costituito da dati generati casualmente in quantità sufficiente per causare il crash del sistema
Viene generato un contenuto per sovrascrivere l'indirizzo di return con uno scelto dall'attaccante
Invia un input di grandi dimensioni sul canale individuato e osserva il comportamento. Nel caso ci sia un crash è possibile un attacco di buffer overflow
Crea del codice shell maligno, che verrà eseguito quando l'esecuzione del programma sarà ritornata ad esso
Inserisce un NOP-sled nel contenuto per l'overflow, in maniera tale che sia più facile far ritornare l'esecuzione al codice maligno.


Attacker
Violazione di Responsabilità e Autorizzazione per la procedura di inserimento dati
L'attaccante scopre che l'applicazione obiettivo utilizza credenziali di sessione per identificare gli utenti legittimi
L'attaccante costruisce dei messaggi contenenti le credenziali generate (come richieste GET o POST, header HTTP o cookies)
L'attaccante fissa gli id di sessione falsificato alla vittima quando la vittima accede al sistema. Una volta che la vittima si è autenticata e ha quindi ottenuto un livello di privilegio più alto, l'attaccante può prendere il controllo della sessione utilizzando l'id di sessione da lui generato
L'attaccante effettua molte connessioni anonime e registra gli id di sessione
L'attaccante effettua connessioni dopo essersi autenticato e registra i token di sessione
L'attaccante manipola il messaggio di richiesta HTTP e aggiunge i propri ID di sessione falsificati alle richieste o ai cookie.
L'autore dell'attacco carica l'ID di sessione predefinito o previsto nel browser e accede a dati o funzionalità protetti.
L'attaccante carica l'ID di sessione predefinito o previsto nel software e utilizza la funzionalità con i diritti della vittima.

DESIGN DEL SISTEMA GUIDATO DAL RISCHIO

Una volta completata l'analisi dei requisiti guidata dal rischio si è poi proseguito con la fase di progettazione dell'applicazione, operando quindi delle scelte progettuali e tecnologiche importanti tenendo conto dei risultati della prima fase. Ogni scelta compiuta è stata sempre giustificata e corroborata da un'ulteriore analisi del rischio, che tiene però conto delle scelte implementative. Si sottolinea, inoltre, che durante la fase di progettazione si è fatto riferimento alle principali linee guida di riferimento ad oggi disponibili. Durante la spiegazione dell'architettura e delle scelte di design, queste linee guida verranno citate indicando il numero della linea guida poi riportata in maniera dettagliata in appendice al paragrafo. Lo stesso verrà fatto nei capitoli successivi.

Architettura generale dell'applicazione

Alla luce dei risultati ottenuti nella prima fase, considerate anche le mitigazioni scelte per molte delle minacce che potevano compromettere gli asset, si sono individuati una serie di requisiti fondamentali, che il sistema software implementato doveva avere. Si è deciso di sfruttare la tecnologia delle blockchain per costruire il sistema di tracciamento della carbon footprint delle catene produttive.

Il sistema di tracciamento è stato dunque suddiviso in due sezioni principali:

1. Il tracciamento in sé è gestito tramite la blockchain e l'esecuzione di smart contracts.
2. Gli utenti possono accedere alla blockchain, tramite un'applicazione sviluppata fuori dalla blockchain, che funge da interfaccia di interazione.

L'applicazione ricalca quindi la struttura di una DApp, che sta per "decentralized application", ovvero applicazione decentralizzata.

La blockchain è stata proposta come strumento per tracciare le catene produttive per via delle sue caratteristiche intrinseche. Essa, infatti, è una tecnologia che permette un tracciamento significativo, registrando qualsiasi operazione come una transazione permanente e leggibile da chiunque. Inoltre, a differenza di una normale transazione svolta in un sistema gestito da un'autorità centrale, ogni transazione sulla blockchain è validata da più macchine, ovvero da più nodi validatori: ognuno di questi nodi esegue la transazione localmente e sulla base del risultato vota in maniera favorevole o contraria l'approvazione della transazione stessa. L'effetto che si ottiene, perciò, è che più entità controllano se la transazione che sta per essere registrata è corretta o meno, offrendo così un meccanismo di verifica più robusto che si addice a un contesto molto delicato, come quello preso in esame. Per questo la blockchain ha i requisiti fondamentali per implementare un tracciamento pubblico dell'impronta di carbonio dei prodotti all'interno delle filiere produttive delle imprese appartenenti al consorzio che gestisce la blockchain.

Altro punto a favore della tecnologia scelta è che all'interno della blockchain le informazioni sono permanenti e, teoricamente, qualsiasi dato inserito nel registro distribuito non può essere modificato o cancellato. Infatti, anche se esistono modi con cui è possibile corrompere o alterare informazioni salvate su una blockchain, questi sono difficilmente

praticabili, quasi impossibili, in quanto richiedono la compromissione di un numero significativo di nodi validatori.

Per par condicio, è necessario anche osservare che la scelta di una blockchain può anche causare alcuni disagi, come un rallentamento delle transazioni, dato il meccanismo di consenso che si deve mettere in piedi, e un consumo di risorse non indifferente. Nel caso analizzato, tuttavia, questi due aspetti negativi non influiscono molto, in quanto la blockchain verrebbe affiancata nel contesto reale da altre tecnologie, messe a disposizione dalle aziende, che limiterebbero questi due aspetti negativi.

Dunque, grazie alle peculiarità sopra elencate, questa tecnologia consente di rispettare diverse policy di sicurezza stabilite per gli asset identificati, diminuendo notevolmente il livello di rischio a cui sono esposti i singoli asset. Per completezza si dettagliano di seguito quali policy la blockchain contribuisce a rispettare:

- **Integrità:** i dati inseriti sulle blockchain non possono essere modificati o cancellati, come sopra descritto.
- **Responsabilità:** tutte le operazioni che vengono svolte sulla blockchain sono tracciate, salvate e visibili a tutti dunque è sempre possibili risalire a chi ha compiuto determinate azioni. Questo permette il rispetto di questa policy, che è essenziale per un sistema di tracciamento come quello in questione.
- **Autenticità**: tutte le informazioni che sono originate all'interno delle blockchain sono considerate autentiche, perché approvate da più nodi. Per quanto riguarda informazioni inserite dall'esterno, è necessario assicurarsi con altre modalità che queste risultino veritiere.
- **Autorizzazione**: la tecnologia rende possibile l'implementazione di meccanismi sofisticati, che permettono solamente ad alcune classi di utenti di accedere a determinate funzionalità.
- **Affidabilità**: la natura distribuita delle blockchain consente alti livelli di affidabilità, in quanto tutte le procedure che implementano le funzionalità richieste sono eseguite da più macchine. Quindi nell'ipotesi, che una o poche macchine commettano un errore nell'esecuzione, questo non influisce in maniera pesante sul risultato della transazione
- **Disponibilità**: La forte ridondanza, diversità, in accordo con le linee guida (4) e (7), e distribuzione intrinseche nell'architettura della blockchain, consentono di resistere anche a malfunzionamenti più o meno diffusi sulla rete, garantendo sempre la fruibilità del servizio e anche la consistenza finale delle transazioni.

Di seguito sono presentate, nel dettaglio, le strutture, le caratteristiche e le peculiarità sia della parte dell'applicazione che "vive" sulla blockchain, sia della parte fuori dalla blockchain che si interfaccia con la prima.

Progettazione del Back-end sulla blockchain

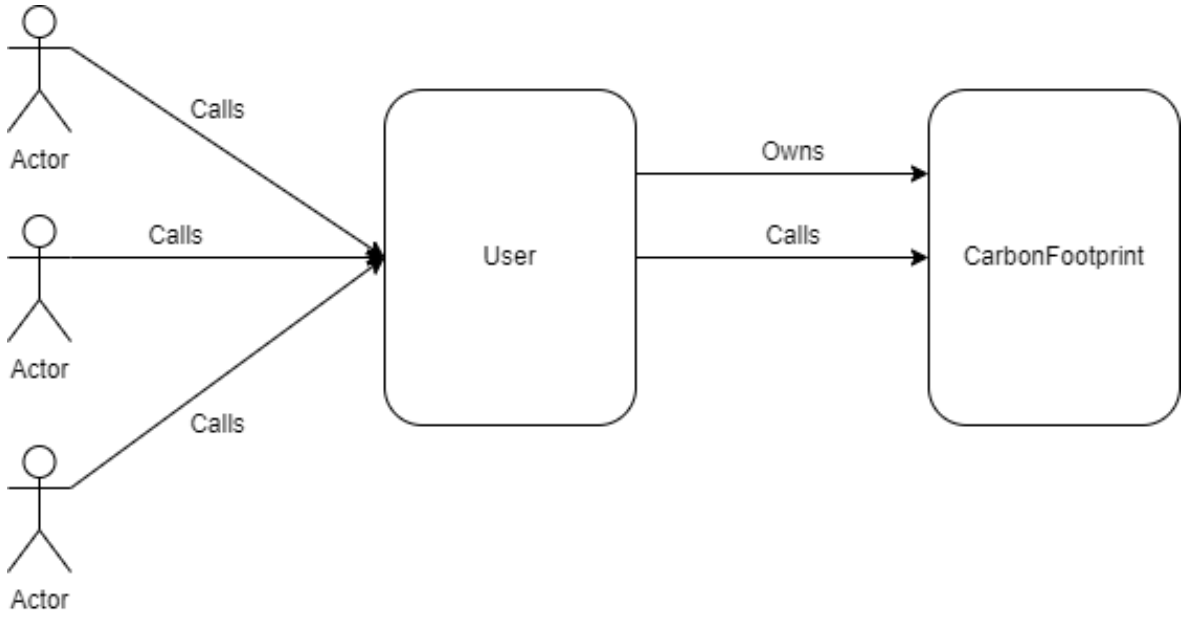
Design dei contratti

La parte del sistema che si trova sulla blockchain rappresenta la parte di back-end, ed è stata progettata per sfruttare i vantaggi che la blockchain offre, con l'intenzione, inoltre, di rispettare tutte le linee guida di buona progettazione, che puntano a minimizzare il rischio. Nonostante tutti i meccanismi di sicurezza che essa garantisce, comunque, anche la parte

sulla blockchain deve essere progettata in una maniera guidata dal rischio, per evitare di inserire all'interno del sistema vulnerabilità grossolane ed evidenti.

In generale, questa parte del sistema si occupa della gestione e del salvataggio di dati e informazioni riguardanti la carbon footprint dei prodotti e delle materie prime, e della gestione dell'interazione fra i vari attori del sistema. Tutto questo, in una blockchain, in accordo con la linea guida (6), può essere implementato attraverso dei moduli software che vengono chiamati contratti o smart contract, e sono assimilabili al concetto di classe tipico della Object-Oriented Programming, e trasparenti a chiunque.

Definite alcuni importanti concetti preliminari si illustra, di seguito, la struttura di questa sezione, composta da due contratti, User e CarboonFootprint tra loro comunicanti, secondo lo schema seguente:



In accordo le linee guida (5) si è adottato il design pattern proxy, in quanto il contratto User funge da proxy per il contratto CarbonFootprint. Infatti, ogni funzionalità esposta da CarboonFootprint viene richiamata dal contratto User se solo se tutti i controlli sui privilegi dell'utente chiamante vanno a buon fine, controlli fatti seguendo le linee guida (1) e (2). Il contratto CarboonFootprint, perciò, non è accessibile in altre vie, o attraverso canali collaterali. Tutti gli utenti che desiderano accedere alle funzionalità esposte da CarbonFootprint dovranno necessariamente passare per il contratto User, in sintonia con le linee guida (8), e questo rende il sistema decisamente più sicuro e difficile da violare. Oltre a questo ruolo, User offre anche delle funzionalità di gestione degli utenti, che permettono a questi ultimi, una volta ottenute le credenziali dal consorzio, di certificare il loro ingresso all'interno del sistema, nonché mantenere le informazioni riguardanti il loro ruolo.

Il contratto CarbonFootprint fornisce tutte le funzionalità necessarie per il tracciamento della carbon footprint e per l'interrogazione da parte degli utenti, implementando anche esso dei controlli sulla business logic delle funzionalità stesse. In esso è implementata tutta la business logic del sistema di tracciamento, e tutte le funzionalità necessarie per permettere a tutte le classi di utenti di svolgere le azioni loro permesse. In più esso si occupa anche di mantenere, attraverso il suo stato, tutte le informazioni riguardanti prodotti e materie prime, che sono gli oggetti principali su cui si focalizza il tracciamento.

La soluzione adottata per favorire l'esecuzione del tracciamento è stata quella di modellare ogni prodotto come un NFT, il che ha concesso numerosi vantaggi:

- Ogni prodotto, essendo un token non fungibile, è unico e perfettamente distinguibile dagli altri attraverso un'ID univoco.
- Il tracciamento e l'implementazione dei passaggi di proprietà, relativi a un prodotto, viene facilitato da meccanismi già esistenti all'interno delle blockchain, come l'emissione di eventi e l'uso di funzionalità predefinite
- I prodotti, essendo NFT, hanno un proprietario ben identificato, e sono associabili facilmente all'account di un utente trasformatore che li possiede.

A differenza dei prodotti, le materie prime non sono gestite tramite NFT, ma tramite una semplice struttura dati definita nel contratto. Una unità di materia prima, nello scenario descritto, è modellata come un oggetto di natura "consumabile", che, una volta utilizzato per un prodotto, non può essere riutilizzato ancora per la lavorazione di un altro prodotto. Infatti, si presuppone che un utente trasformatore, fuori dalla blockchain, indichi, nel momento in cui vuole avviare la lavorazione di un nuovo prodotto, a uno o più fornitori il numero di unità di materie prime necessarie. Questo comporta che il fornitore, una volta vendute le materie prime, le inserirà, un'unità alla volta, all'interno del sistema di tracciamento, assegnandone la proprietà al trasformatore che ne beneficia, che le potrà usare per una sola volta. Quindi, esse non saranno mai soggette ad altri passaggi di proprietà, oltre a quello iniziale. Inoltre, le uniche informazioni da tracciare relativamente alle materie prime sono l'uso che ne viene fatto, la carbon footprint, e il momento di inserimento. Quindi modellarle utilizzando un NFT, non porta nessun vantaggio significativo nell'applicazione, anzi creerebbe il problema dell'invalidazione del token non fungibile dopo l'uso, che non è facilmente governabile e risolvibile con le competenze attuali.

Infine, per tenere traccia delle azioni compiute dagli utenti vengono emessi sulla blockchain degli eventi, i quali permettono facilmente di ricostruire l'intera filiera del prodotto. Il sistema di emissione degli eventi è molto utile perché consente, in accordo alla linea guida (3), di operare un log completo ed esaustivo delle azioni fatte dagli utenti, usando meccanismi predefiniti delle blockchain. Nel contesto proposto, cioè quello di un sistema di tracciamento, l'emissione degli eventi sarà di carattere verboso e per questo viene operata per quasi tutte le operazioni compiute dagli utenti. In particolare, vengono creati degli eventi al momento della registrazione di un nuovo utente, all'inserimento di una nuova trasformazione su un prodotto, al termine del processo di lavorazione di un prodotto, all'inserimento e all'utilizzo di una unità di materia prima.

Design del controllo degli accessi

Secondo quanto definito nel contesto dell'applicazione, sono previste tre diverse classi di utenti: cliente, trasformatore, fornitore. Questi si interfacceranno con la blockchain attraverso il contratto User, ovvero l'unico entry point del backend. Quindi all'interno del contratto verrà anche costruito un sistema di controllo degli accessi che verificherà se l'utente che vuole accedere, è registrato tra quelli del consorzio, e se esso dispone di tutti i privilegi necessari per eseguire l'azione richiesta. Inoltre, essendo la blockchain utilizzata privata, esiste un meccanismo di controllo degli accessi di default che vieta a tutti gli utenti non pre-registrati sulla blockchain attraverso i loro portafogli di compiere una qualsiasi operazione su di essa, e quindi sui contratti al suo interno salvati.

Design del monitoraggio

La blockchain e gli smart contract mettono a disposizione dei meccanismi per l'implementazione di un runtime enforcement efficace. Nell'applicazione, all'interno dei contratti, sono previsti dei meccanismi di questo tipo per forzare delle proprietà di safety desiderate. In particolare, questi meccanismi verranno utilizzati per controllare se gli utenti non stiano cercando di compiere delle operazioni che violano i privilegi a loro assegnati. La loro implementazione verrà meglio discussa nelle sezioni successive.

Riferimenti alle linee guida adottate durante la fase di design del back-end e dell'applicazione in generale

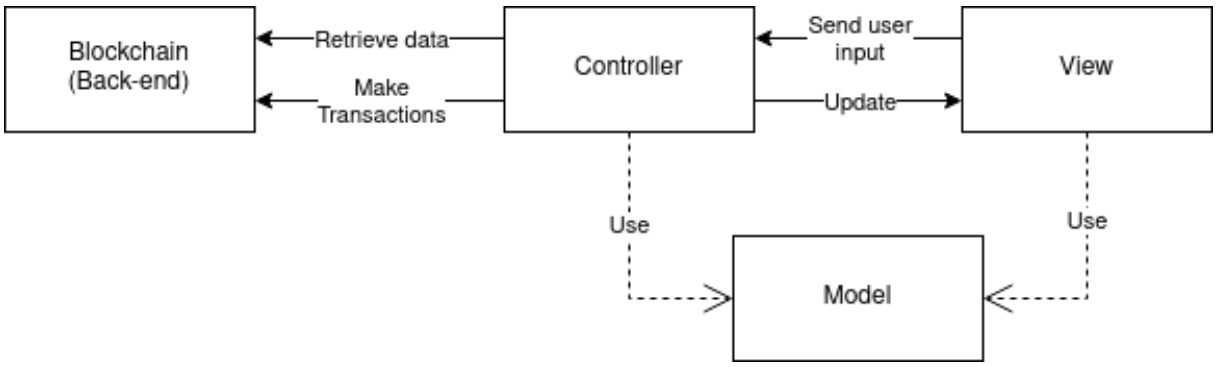
Durante la fase di progettazione si è cercato di adottare il più possibile le principali linee guida di design come quelle di OWASP, Saltzer e Schroeder o Sommerville. In particolare, per quanto riguarda la parte sulla blockchain:

1. **Separazione dei privilegi**: i livelli di privilegio sono stati ben divisi fra le tre classi di utenti. I clienti non possono accedere alle funzionalità dei trasformatori e dei fornitori, i quali non possono accedere alle reciproche funzioni. Questa divisione viene garantita nel contratto User.
2. **Privilegi minimi**: Grazie a diverse funzionalità offerte dagli smart contract, è possibile controllare strettamente i privilegi degli utenti limitandoli il più possibile.
3. **Fare il logging delle azioni degli utenti**: le azioni degli utenti possono essere semplicemente loggate tramite gli eventi disponibili di default sulla blockchain.
4. Evitare i single point of failure: grazie alla natura decentralizzata delle blockchain gli SPOF sono evitati.
5. **Minimizzare la superficie di attacco:** come già detto, User funge da proxy per le funzionalità di CarbonFootprint, alle quali non si può accedere in altro modo. Questo focalizza tutte le interazioni dell'utente su questo contratto, riducendo al minimo indispensabile la superficie di attacco.
6. **Design aperto**: i contratti, essendo su una blockchain, sono visibili a tutti. Ciò rende la loro revisione possibile a chiunque lo desideri, permettendo così di individuare tempestivamente eventuali errori e problemi.
7. **Usare ridondanza e diversità**: la blockchain implementa ridondanza e diversità naturalmente, dato che i blocchi sono salvati su tutti i nodi validatori contemporaneamente e i nodi stessi sono diversi l'uno dall'altro.
8. **Mediazione completa**: tutti gli accessi alla blockchain sono controllati mediante il contratto User.

Progettazione Off-Chain

Per quanto riguarda la parte "off-chain", si è deciso di utilizzare Python, scegliendo un'architettura MVC (Model-View-Controller) che è stata adattata alle esigenze del programma. L'MVC è stato scelto perché è un design pattern semplice e adatto alle funzionalità che andavano implementate. Permette infatti di separare la logica del programma dalle viste, rendendole indipendenti tra loro. Inoltre, questo consente di fornire funzionalità diverse per ogni tipologia di utente permettendo di separare il più possibile le componenti, i moduli e le operazioni del programma in base al livello di utenza in accordo con le linee guida (1) e (2).

La separazione, oltre a fornire maggiori garanzie di sicurezza di per sé, rende la manutenibilità del sistema sul lungo termine migliore, perché le modifiche su una parte del sistema non influenzeranno mai tutta l'applicazione. Modifiche alla view, ad esempio, non comporteranno alcun cambiamento al model dei dati, o ai controller stessi in quanto questi componenti sono comunicanti tra loro ma fortemente disaccoppiati. Inoltre, i modelli dei dati e le funzionalità dei controller difficilmente verranno modificati. Questo perché entrambi sono legati al back-end su blockchain, il quale è a sua volta difficilmente modificabile per le proprietà intrinseche degli smart contract su blockchain.



Nel dettaglio:

- Il Model si occupa di mappare le strutture dati presenti nella blockchain in modo da avere un'interfaccia comune
- La View ha il compito di mostrare una vista, differente per ogni tipologia di utente, dato che essi possono compiere azioni differenti. La vista sarà semplice e di facile comprensione come definito nella linea guida (3)
- Il Controller invece, ha il compito di interagire con la blockchain e quindi di recuperare i dati da essa e di compiere transazioni per conto degli utenti in sintonia con la linea guida (4)

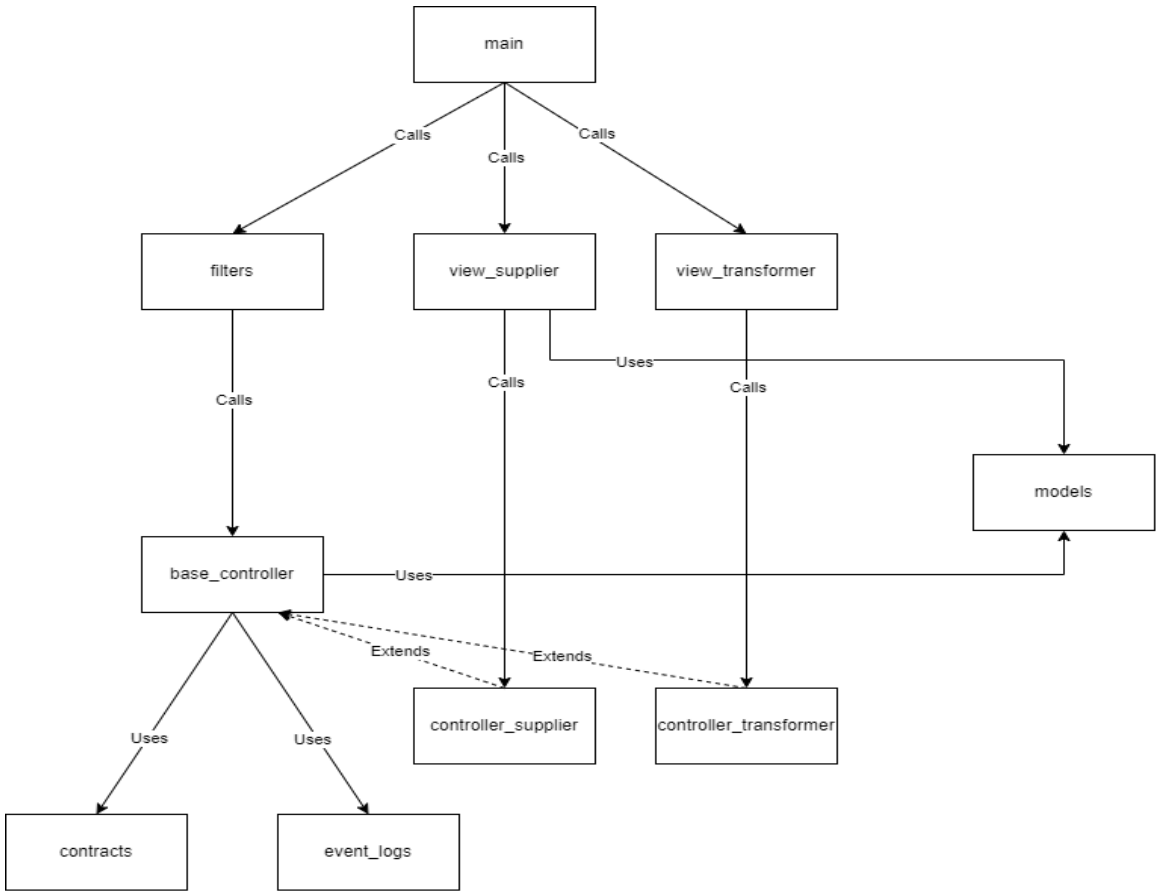
Visto che alcune azioni sono comuni a tutte le tipologie di utente, si è deciso di creare una classe controller base, che contiene tutti i metodi condivisi tra gli utenti, e delle classi controller specifiche per fornitore e trasformatore, che estendono il modulo base. Data la pericolosità che queste dipendenze possono avere, si è fatta molta attenzione ad evitare qualsiasi problema di sicurezza e a mantenere la catena di dipendenza molto breve e dunque facile da tenere sotto controllo.

Si vuole far notare che, nel nostro caso specifico, è il Controller che si occupa del recupero dei dati dalla blockchain e non il Model come nel classico MVC: questo è dovuto alla mancanza del classico accoppiamento che è spesso presente fra il model e un database come può accadere in applicazioni classiche che non fanno utilizzo di blockchain. Infatti, queste ultime, rispetto ai classici database, presentano al loro interno un'ulteriore business logic con altre funzioni, venendo così a mancare il ruolo di interfaccia con i dati che il model solitamente svolge. In questo caso, dunque, il model classico del MVC sarebbe stato un ulteriore strato software inutile, in quanto avrebbe semplicemente preso la chiamata di una funzione del controller e l'avrebbe rimbalzata sulla blockchain. Si potrebbe, dunque, arrivare ad affermare che, gli smart contract sulla blockchain siano di fatto l'elemento più simile al Model classico del pattern MVC perché contengono al loro interno diverse funzioni che

operano sui dati con inserimenti e modifiche, e soprattutto mantengono anche lo stato dell'applicazione salvando i dati in maniera persistente.

Struttura dei moduli dell'applicazione

Al livello di design, una possibile struttura dei moduli dell'applicazione e dell'interazione tra loro è rappresentata nella figura sottostante:



L'entry point dell'applicazione è il modulo "main", il quale, in base al ruolo dell'utente, passerà il controllo ad una delle viste disponibili. Queste ultime sono di tre tipologie, definite in base al livello di utenza: filters per il cliente, view_supplier per il fornitore e view_trasformer per i trasformatori. Fatto ciò, queste passano il controllo a uno dei controller, che si occuperanno di richiamare i metodi del contratto User salvato sulla blockchain, supportati dai moduli ausiliari indicati, cioè contracts e event_logs. I controller sono tre, anche in questo caso uno per ogni livello di utenza. Il modulo "models" mapperà le strutture dati che modellano prodotti, materie prime e trasformazioni.

Linee guida adottate durante la fase di progettazione della parte off-chain dell'applicazione

Anche durante la progettazione di questa parte di applicazione si è fatto riferimento alle linee guida, alcune di esse sono condivise con il back-end ma, in questo caso, sono implementate in maniera differenze e per scopi diversi. Di seguito sono riportate quelle più significative e che hanno un impatto più esplicito nell'applicazione:

1. **Separazione dei compiti**: il design pattern MVC garantisce la separazione delle varie funzionalità di interazione con l'utente e di logica interna. Inoltre, I ruoli degli

utenti sono ben definiti, e le operazioni che possono eseguire sono distinte a dipendenza dei ruoli grazie alla suddivisione delle viste e dei controller. Come spiegato nella sezione precedente, infatti, se le funzionalità comuni sono accessibili con una vista di base e sono gestite con un controller di base, le funzionalità uniche di fornitori e trasformatori sono invece accessibili e poi gestite da altre viste e controller che estendono quelli di base e a cui si accede solamente se il proprio ruolo è quello corretto.

2. **Privilegi minimi**: la suddivisione dei moduli permette anche di ridurre al minimo indispensabile le funzionalità a cui i vari utenti possono accedere.
3. **Facilità di utilizzo e accettabilità**: benché molto sicura, la blockchain nell'applicazione sarà di facile utilizzo perché mediato attraverso un front-end che espone le funzionalità in maniera semplice ed efficace all'utente.
4. **Non avere fiducia nei servizi**: l'interazione con la blockchain saranno gestite da moduli specifici, i controller, permettendo, in fase implementativa, di eseguire tutti i controlli del caso in maniera precisa e localizzata isolando il più possibile la blockchain dal resto del programma. Benché la parte applicativa sulla blockchain venga progettata insieme alla parte off-chain è bene considerarla come fosse un servizio esterno, in quanto, data la sua estensione e la sua natura distribuita non è sempre facile prevederne a pieno il comportamento.

IMPLEMENTAZIONE DEL SISTEMA DI TRACCIAMENTO

Implementazione del Back-end sulla blockchain

La blockchain utilizzata è Quorum, una blockchain di tipo permissioned. Tale scelta è motivata dal fatto che nel caso di studio considerato, i partecipanti alla blockchain devono essere predeterminati e approvati da un consorzio, in quanto appartenenti a una filiera produttiva. Inoltre, nonostante nasca come fork di Ethereum non prevede un costo per il gas utilizzato nell'esecuzione delle transazioni, ed infatti risulta molto adottata proprio nel campo della gestione delle catene di approvvigionamento. Nonostante ciò, il concetto di gas rimane, in maniera tale da prevenire comportamenti inaspettati, come cicli infiniti e occupazione di risorse per lungo tempo.

Il back-end è stato realizzato in Solidity, il linguaggio messo a disposizione da Ethereum, il quale permette di realizzare facilmente smart contract da eseguire su Ethereum Virtual Machine (EVM) e implementare tutte le specifiche definite nella fase di progettazione.

Scendendo ora nel dettaglio dell'implementazione dei contratti definiti durante la fase di design.

Contratto CarbonFootprint

In questo contratto viene definito il token non fungibile rappresentante i prodotti e la loro carbon footprint utilizzando l'implementazione dell'interfaccia ERC721 disponibile all'interno

della libreria di “Openzeppelin”. L’interfaccia IERC721 è lo standard per la costruzione di NFT tramite l’utilizzo di smart contract.

Dunque, la principale funzionalità di questo contratto è gestire gli NFT dei prodotti. Da sottolineare è, innanzitutto, come si è modellata l’entità prodotto: si è deciso di utilizzare una “struct” con diversi attributi.

```
struct Product {
    uint256 productId;
    string name;
    address currentOwner;
    uint256 CF;
    bool ended;
}
```

- **productId**: identificatore numerico del prodotto
- **name**: nome del prodotto
- **currentOwner**: indirizzo di colui che attualmente possiede questo prodotto.
- **CF**: carbon footprint totale associata a quel particolare prodotto
- **ended**: indica se il prodotto è stato completato.

Oltre alla classica implementazione dell’interfaccia, questo contratto gestisce anche altre operazioni essenziali per i particolari casi d’uso che questa applicazione deve consentire:

- **Gestione delle materie prime:** il contratto consente di aggiungere, memorizzare e leggere le materie prime necessarie a realizzare prodotti e trasformazioni. La materia prima viene modellata a livello software utilizzando una “struct”, come accade per i prodotti, che contiene tutti gli attributi che caratterizzano la materia prima.

```
struct RawMaterial {
    uint256 materialId;
    string name;
    uint256 lot;
    address supplier;
    address transformer;
    uint256 CF;
    bool isUsed;
}
```

  - **materialId:** identificatore univoco della materia prima
  - **name:** nome della materia prima
  - **lot:** numero che indica il lotto di quella particolare materia prima
  - **supplier:** indirizzo del fornitore che ha appunto fornito quella particolare materia prima
  - **transformer:** indirizzo del trasformatore che possiede quella particolare materia prima
  - **CF:** carbon footprint associato alla particolare materia prima.
  - **isUsed:** indica se quella particolare materia prima è stata utilizzata e non può più essere dunque riutilizzata.

  Attraverso tutti questi attributi è possibili descrivere e gestire l’intero ciclo di vita della materia prima secondo gli use case dell’applicazione. In particolare, rispetto ad un classico NFT, le materie prime sono consumabili e la loro vita è limitata e possono essere utilizzate una sola volta.

  Le materie prime strutturate in questo modo sono poi effettivamente gestite inserendole all’interno di un array, come avviene anche per i prodotti, ma con la differenza che le materie prime non sono state rese dei token, ovvero non sono stati

implementati tutti i metodi dell'interfaccia IERC721. Inoltre, nel contratto è presente un mapping che riporta il prodotto per cui ciascuna materia prima è stata usata, o il valore di default '0' se ancora non è stata impiegata per realizzare un prodotto. Infine, vengono emessi degli eventi durante le operazioni di inserimento e utilizzo delle materie prime che consentono di tenere traccia di tutte le informazioni relative ad esse e richieste dalle specifiche.

- **Gestione delle trasformazioni:** ogni prodotto, in seguito a delle trasformazioni, vede variare la propria carbon footprint e per questo il contratto CarbonFootprint consente di modificarne il valore associato.
- **Tracciamento:** tramite gli eventi si traccia tutto ciò che accade durante la catena produttiva. Questo permette la ricostruzione completa dell'itinerario seguito dal prodotto. Nello specifico, si tiene traccia:
    - dell'inserimento di nuovi prodotti.
    - di trasformazioni a prodotti già esistenti con relativa modifica al valore della carbon footprint.
    - dell'inserimento di nuove materie prime.
    - dell'utilizzo di materie prime, già presenti nel sistema, per la creazione di un prodotto
    - della conclusione della catena produttiva

Tutte queste operazioni vengono fatte solamente dopo aver passato una serie di controlli di varia natura gestiti dal contratto necessari per garantire il corretto funzionamento e la corretta logica dell'applicazione secondo le specifiche e i vincoli di dominio. I controlli effettuati sono di varia natura e dipendono dalla specifica funzione che si sta controllando. I principali controlli effettuati sono:

- **Controlli sulla validità dei parametri passati in input alle funzioni**: i dati che vengono passati in input alla funzione sono verificati accuratamente, in accordo con la linea guida(1) e (2), secondo le possibilità che solidity offre. I controlli esercitati, sono di vario genere e includono range check e size check, dei parametri. Purtroppo, in solidity non è possibile comparare stringhe con altre stringhe, nonostante sia una necessità per eseguire alcuni controlli nell'applicazione. Per ovviare al problema è stato necessario utilizzare una funzione di hash, keccak256(o SHA256), che genera il digest delle stringhe, eventualmente concatenate, che si vogliono confrontare. I digest, essendo in solidity di tipo "bytes20", possono essere confrontati facilmente con l'operatore "==" consentendo così il controllo di uguaglianza tra le stringhe. Tutti i controlli lato blockchain, essendo computazionalmente onerosi a causa del meccanismo di consenso e dell'esecuzione su più nodi, sono sempre di natura semplice e su dati di tipo primitivo. Al contrario i controlli su strutture dati di tipo reference sono limitati ai casi strettamente necessari, e richiesti dalle specifiche.
- **Controlli legati alla logica del programma**: questi controlli servono per verificare delle proprietà di safety richieste dalle specifiche stabilite in fase di analisi dei requisiti. Il contratto deve garantire, prima dell'esecuzione di qualsiasi transazione, che tutte le proprietà di safety, richieste per essa, siano verificate, e in caso non lo fossero deve bloccare l'esecuzione. Alcuni esempi di questi controlli sono: verifica che un prodotto desiderato esista, verifica che una materia prima che si intende utilizzare non sia stata già utilizzata, e così via.

- **Controlli sui permessi**: tutti i metodi del contratto possono essere invocati solamente dal possessore dello stesso, ovvero il contratto User. Questo è stato fatto per poter implementare il pattern Proxy.

Questi controlli vengono fatti attraverso la parola chiave “require” di solidity in accordo con il coding standard (1). Essi permettono di definire delle condizioni nel codice che devono essere verificate affinché una qualsiasi invocazione del contratto vada a buon fine. In caso di mancato soddisfacimento di queste condizioni viene ritornato e mostrato all’utente un messaggio di errore personalizzato che spiega quale condizione è stata violata in accordo con la linea guida(4).

Per assicurarsi che la funzione di proxy del contratto User non venga evitata si sfrutta un modificatore, chiamato “onlyOwner”, costruito appositamente per evitare che delle chiamate possano provenire da altre fonti. Come si può intuire dal nome, questo modificatore fa in modo che solamente il possessore del contratto potrà eseguire la funzione che lo specifica. Inoltre, nel caso in cui l’eccezione generata, corrisponda al fallimento della transazione, la blockchain riesce a ritornare a uno stato consistente in accordo con la linea guida (5).

Contratto User

Questo contratto svolge diversi compiti:

1. Gestione degli utenti dell’applicazione: memorizza attraverso un mapping e una variabile di tipo enum quale ruolo è assegnato ad ogni wallet registrato sulla blockchain. Oppure, attraverso una funzione, consente ad un utente non ancora registrato di associare un ruolo all’indirizzo del proprio wallet. Infine, mantiene traccia della registrazione degli utenti attraverso l’emissione di un evento.
2. Deployment del contratto “CarbonFootprint”: il contratto si occupa di fare il deploy del contratto CarbonFootprint, diventando un proxy per quest’ultimo. Infatti, i metodi di CarbonFootprint possono essere chiamati solo dal proprietario del contratto, così come descritto precedentemente. User si occupa di svolgere una serie di controlli di autorizzazione verificando che l’utente chiamante sia del ruolo giusto e abbia le giuste autorizzazioni per richiamare il particolare metodo del secondo contratto. Questo è gestito tramite una coppia di modificatori, che funzionano similmente a onlyOwner, ovvero onlySupplier e onlyTransformer. Questi modificatori sono applicati rispettivamente alle funzioni riservate agli utenti di tipo fornitore e a quelle riservate agli utenti di tipo trasformatore. Grazie ad essi, inoltre, è stato possibile implementare i meccanismi di monitoraggio, in particolare di runtime enforcement, descritti durante la fase di progettazione. I modificatori, infatti, si comportano come dei monitor andando a modificare l’esecuzione del programma qualora si verifichino degli eventi che non rispettalo determinate proprietà. In questo caso, le proprietà di safety sono del tipo: “un utente non trasformatore non può invocare determinate funzioni”, “un utente di tipo cliente non può invocare funzioni riservate agli utenti trasformatori o a quelli fornitori”.
3. Mapping delle funzioni di CarbonFootprint: esso mette a disposizione dei metodi per richiamare le funzionalità implementate nel contratto “CarbonFootprint”, non accessibili direttamente in accordo alla linea guida (3), in maniera tale da permettere agli utenti di svolgere le operazioni di:
    - Aggiunta di un nuovo prodotto
    - Aggiunta di un nuovo lotto di materia prima

- Aggiunta di una trasformazione su un prodotto esistente
- Trasferimento di un token relativo ad un prodotto al wallet di un altro trasformatore

Inoltre, sono presenti anche delle funzioni di lettura che permettono di recuperare la lista di tutti i prodotti o materie prime oppure i dettagli relativi ad uno specifico prodotto. È importante anche sottolineare che si riduce l'overhead delle funzioni generato dal proxy, grazie all'utilizzo della variabile tx.origin, che viene usata in maniera attenta in accordo al coding standard (2).

Linee guida di buona programmazione adottate

1. **Controllo della validità degli input**: qualsiasi dato proveniente dall'esterno della blockchain viene validato, anche attraverso l'utilizzo del costrutto require, verificando che esso aderisca al formato corretto. Ad esempio, si filtrano stringhe vuote per i nomi di prodotti o materie prime, si verifica che l'input inserito non sia corrispondente già a quello di altre entità e così via. Il controllo è applicato sia ai dati generati dalla parte off-chain dell'applicazione, sia ai dati inseriti attraverso canali diversi.
2. **Controllo sulla dimensione degli array**: sempre utilizzando il costrutto require, si verifica che gli array inseriti in input non siano array vuoti, oppure nel caso delle materie prime si verifica che il numero di nomi, lotti e carbon footprint inserite sia coerente con il numero di materie prime che si vuole inserire.
3. **Limitare la visibilità dell'informazione**: Tutto ciò che si trova all'interno del contratto CarbonFootprint è privato, al di fuori dell'indirizzo del possessore. Per poi assicurare che l'accesso a questo contratto avvenga in maniera completamente controllata si sfrutta il contratto User come proxy in aggiunta al modificatore onlyOwner.
4. **Provvedere un gestore per tutte le eccezioni**: il costrutto require oltre a bloccare l'inserimento di input mal formattati, blocca anche l'esecuzione di una transazione nel momento in cui sia violata una determinata proprietà di safety. La require innesca il meccanismo di gestione dell'eccezione, generata, tipicamente bloccando l'esecuzione e restituendo un messaggio di errore all'utente.
5. **Provvedere capacità di ripristino**: Il funzionamento intrinseco della blockchain fa in modo che se una transazione fallisce, lo stato della blockchain ritorna all'ultimo stato valido, permettendo così di ricominciare l'interazione.

Coding standard di solidity seguiti

1. **Checks-Effects-Interactions pattern**: nella scrittura delle funzioni contenute nei contratti è stato seguito il pattern indicato, il quale indica che l'ordine più sicuro in cui eseguire le istruzioni è il seguente:
    i. Controlli sul chiamante e sui parametri della funzione
    ii. Modifiche alle variabili di stato del contratto
    iii. Chiamate ad altri contratti
2. **Non usare la variabile tx.origin per l'autorizzazione:** nell'implementare il pattern proxy si è reso necessario l'utilizzo di tx.origin, ma si è posta particolare attenzione nel non eseguire mai su di esso controlli legati all'autorizzazione in quanto da questi potrebbero poi derivare dei problemi di sicurezza.
3. **Formato Natspec per la documentazione del codice:** si è utilizzato il formato Natspec per documentare il codice come raccomandato dalle linee guida di solidity.

Implementazione della parte off-chain in Python

Descrizione dei moduli

Per scrivere il codice è stato fatto uso di librerie di terze parti di python. Tre sono da sottolineare: una ovviamente è “web3”, necessaria per lavorare con le blockchain. Un’altra è la libreria “inquirer”, che offre vari metodi per gestire l’interazione con l’utente in forma testuale. Infine, è stata usata la libreria “tabulate” per gestire la stampa.

Segue una descrizione dei moduli e delle loro funzionalità, che sono stati implementati cercando di attenersi il più possibile alle linee guida (1) e (4).

- **Main**: nel main troviamo il primo livello di interazione con l’utente. L’interazione viene gestita facendo utilizzo della libreria inquirer. Per prima cosa all’utente è richiesto di selezionare il proprio ruolo fra cliente, fornitore e trasformatore e successivamente dovrà fornire il suo wallet, che sarà validato. Il modulo “**connection.py**” si occupa di connettere l’utente al nodo associato al ruolo. A dipendenza della scelta precedente viene poi presentata all’utente la lista delle azioni eseguibili. Tutto ciò è gestito tramite un dizionario, nel quale troviamo i ruoli selezionabili e le azioni eseguibili in base al ruolo.

```
role_dict = {
    "Client": {
        "num": "0",
        "actions": [
            ("Search one or more products", filters.filter_products),
            ("Exit", bye),
        ],
    },
    "Supplier": {
        "num": "1",
        "actions": [
            ("Search one or more products", filters.filter_products),
            ("Add new raw materials", view_supplier.insert_raw_material),
            ("Exit", bye),
        ],
    },
    "Transformer": {
        "num": "2",
        "actions": [
            ("Search one or more products", filters.filter_products),
            ("Create a new product", view_transformer.create_new_product),
            ("Add a new operation", view_transformer.add_transformation),
            ("Transfer the property of a product", view_transformer.transfer_product),
            ("Exit", bye),
        ],
    },
}
```

  Come mostrato in figura, la lista di azioni è una lista di tuple, formate da una stringa e da un riferimento ad una funzione: questo perché inquirer, nell’input di tipo lista, se è presente una tupla, mostra all’utente il primo valore e salva come risposta il secondo. Così facendo è possibile eseguire l’azione selezionata chiamando la funzione salvata come scelta dell’utente.

- **filters**: Nel modulo filters viene gestita l'operazione di ricerca prodotti tramite l'utilizzo di uno o più filtri. È possibile scegliere di filtrare i prodotti in base a diversi filtri predefiniti. Questi filtri sono combinabili sia con logica "and" sia con logica "or". La costruzione del filtro avviene per iterazioni successive, sfruttando il meccanismo di "tail recursion". Ad ogni step è possibile scegliere se stampare i risultati ottenuti o aggiungere un nuovo filtro.
- **view_supplier**: In view_supplier troviamo la logica d'interazione con l'utente di tipo fornitore. La funzione principale che svolge il fornitore è quella di inserire materie prime. Durante il processo di inserimento è possibile inserire molteplici materie prime contemporaneamente: questo modulo prevede una funzione di validazione, in accordo alla linea guida (2) che impedisce di inserire due materie prime identiche in un'unica operazione. Per effettuare la scrittura sulla blockchain il controllo passa al modulo controller_ supplier.py.
- **view_transformer**: In view_transformer, similmente al modulo precedente, troviamo la logica di interazione con l'utente di tipo trasformatore. In questo caso però le funzionalità gestite sono tre: l'aggiunta di un nuovo prodotto, l'aggiunta di una nuova trasformazione ad un prodotto già esistente e il trasferimento di un prodotto non concluso ad un nuovo trasformatore. Anche qui, per effettuare la scrittura sulla blockchain il controllo passa al controller specifico.
- **event_logs**: In event_logs è presente la classe EventLogs, che gestisce il recupero degli eventi dalla blockchain. Vengono recuperati gli eventi che tracciano:
    - La conclusione della catena produttiva dei prodotti
    - La creazione di nuovi prodotti
    - Il trasferimento di prodotti fra trasformatori
    - L'utilizzo di materie prime
    - Tutte le trasformazioni di tutti i prodotti o di un prodotto specifico

  Partendo da questi eventi, si sono costruite delle funzioni parziali, tramite l'utilizzo di partial(). Questo ha reso possibile il riutilizzo di queste funzioni, specificando di volta in volta argomenti diversi, per ricavare dati specifici dalla blockchain.
- **models**: Il modulo models contiene tre classi che mappano rispettivamente le materie prime, i prodotti e le trasformazioni. Le tre classi presentano costruttori diversi in base all'origine dei dati: o presi dalle strutture presenti nella blockchain (RawMaterial e Product), o ricavate attraverso gli eventi (RawMaterial e Transformation). Sono stati inoltre ridefiniti: il metodo __str__() per gestire la stampa di questi oggetti e, dove necessario, il metodo __eq__().
- **validation**: In validation troviamo tutti i metodi necessari per validare l'input dell'utente in accordo alla linea guida (2). Questi vengono passati alle funzioni di input di inquirer come valore del parametro "validate". La struttura di queste funzioni è predefinita dallo standard della libreria. Ad esempio, devono sollevare delle specifiche eccezioni nel caso in cui l'input non sia corretto, e i loro parametri sono definiti a priori. La validazione è un procedimento molto importante per evitare comportamenti indesiderati da parte del sistema. Le validazioni eseguite sono:
    - Validazioni sugli indirizzi. Per farlo si sfrutta la funzione "toChecksumAddress" di Web3, che lancia un'eccezione se la stringa passata come parametro non rappresenta un possibile indirizzo. La funzione di validazione stessa inoltre lancia un'eccezione se l'indirizzo è composto da soli zeri.
    - Validazione sul valore della carbon footprint. Deve risultare un intero positivo.

    - Validazione sul numero di lotto, che deve essere un intero non negativo.
    - Validazione sull'id del prodotto. Anch'esso deve essere un intero non negativo.
    - Validazione del nome di una materia prima o di un prodotto. Può contenere solo caratteri alfanumerici, e avere una lunghezza da 2 a 50 caratteri. Per questa validazione si ricerca il regex pattern: "^[a-zA-Z0-9]{2,50}$"
- **contracts**: in questo modulo sono implementate due funzioni per costruire le istanze dei due smart contracts. Vengono utilizzate le primitive messe a disposizione dal modulo web3. Nel particolare si costruisce il contratto principale (il contratto user) a partire dall'indirizzo presente nel file address.json (popolato precedentemente dallo script deploy_contracts.py). Questo file è presente per tenere in memoria l'indirizzo dello smart contract e per evitare di salvarlo all'interno del codice: questo permette di aggiornarlo facilmente in caso il contratto subisca un nuovo deploy.
- **base_controller**: in base_controller si trova una classe, chiamata Blockchain, che espone i metodi, comuni a tutti gli utenti per interagire con la blockchain. La classe si occupa di costruire le istanze dei contratti e contiene tutte le funzioni dei contratti che sono accessibili a tutti i livelli di utenza. Inoltre, è responsabile per la validazione del wallet inserito dall'utente: verifica che l'indirizzo appartenga al nodo a cui si vuole stabilire la connessione.
- **controller_supplier**: il modulo controller_supplier si occupa della logica di gestione delle funzionalità specifiche dei fornitori. Contiene una classe, Supplier, che estende la classe Blockchain presente in base_controller. Questa sottoclasse aggiunge il metodo per creare una nuova materia prima, prendendo in ingresso i dati inseriti dall'utente tramite il modulo view_supplier. Costruisce poi, il contenuto della transazione e la manda alla blockchain.
- **controller_transformer**: il modulo controller_transformer si occupa della logica di gestione delle funzionalità specifiche dei trasformatori.
  Come nel modulo precedente, anche questo contiene una classe che estende Blockchain, in questo caso chiamata Transformer. Questa sottoclasse aggiunge quattro nuovi metodi:
    - Il metodo per filtrare i prodotti ottenuti dalla blockchain che restituisce solo quelli posseduti dal trasformatore corrente
    - Il metodo per aggiungere una nuova trasformazione ad un prodotto
    - Il metodo per trasferire la proprietà di un prodotto
    - Il metodo per creare un nuovo prodotto

  Questi tre ultimi metodi hanno un funzionamento analogo fra loro: ottengono i dati inseriti dall'utente attraverso il modulo view_transformer, e li usano per effettuare le transazioni sulla blockchain.

Gestione dei dati presenti nella blockchain

Per evitare inconsistenza tra i dati proposti all'utente tramite l'interfaccia e i dati presenti sulla blockchain, si è deciso di non salvare i dati in locale, ma di prenderli direttamente dalla blockchain ogni qualvolta fosse necessario, sfruttando sia le strutture dati, sia gli eventi emessi durante le transazioni.

Gestione delle eccezioni

Si discute brevemente, la gestione delle eccezioni, sviluppata in accordo alla linea guida (3) dell'applicazioni distinguendo tra errori relativi alla logica degli smart contract, ed errori di connessione:

- **Errori relativi alla logica degli smart contract**: in base ai require definiti negli smart contract, è possibile che l'esecuzione della transazione venga annullata. Tramite delle eccezioni definite in web3 si può gestire questo comportamento e comunicare all'utente il messaggio di errore.
- **Errori relativi alla connessione alla blockchain**: essendo possibile una disconnessione dalla blockchain, in qualunque momento, viene gestita questa possibilità attraverso costrutti try-except specifici. Questa eccezione viene gestita in qualunque operazione che necessita di una connessione alla blockchain.

Linee guida di buona programmazione adottate

1. **Il sistema è di facile utilizzo**: le operazioni possibili sono semplici e chiaramente presentate all'utente. Se l'utente commette un errore il sistema lo avverte specificando cosa ha sbagliato, permettendogli di correggere l'errore. L'utente inoltre non vede mai le funzionalità che non competono al suo ruolo.
2. **Controllare la validità di tutti gli input**: tutti gli input degli utenti, non appena inseriti, sono prontamente validati con le funzionalità del modulo validation. La validazione aiuta a capire il formato corretto dell'input. In questo modo si evita di passare ai contratti nella blockchain dei dati mal formattati o senza senso.
3. **Fornire un gestore per tutte le eccezioni**: nel caso in cui si verifichi un qualunque errore, questi vengono sempre gestiti tramite eccezioni e messaggi di errore, evitando il crash dell'applicazione.
4. **Si è limitato al minimo l'utilizzo di costrutti error-prone:** L'applicazione non sfrutta parallelismo, interruzioni, goto, numeri in virgola mobile, aliasing. È presente un limitato utilizzo dell'ereditarietà, in quanto questa scelta rappresentava una strutturazione sensata dei controller. L'ereditarietà è ben controllata e si ferma al primo livello di sottoclassi, il che permette di evitare i problemi più gravi che possono scaturire da un suo uso improprio o esagerato.
5. **Il codice è ben documentato** in ogni sua funzione ed è stato utilizzato un code formatter, in modo da rendere la comprensione del codice e la sua manutenzione semplice.
6. **È stato fatto uso di un linter** per controllare la qualità di scrittura del codice.

TESTING

SMT Checker Formal Verification

È stato eseguito un processo di verifica formale del codice mediante il model checker messo a disposizione da Solidity. Queste verifiche non hanno evidenziato particolari problematiche, ma hanno sollevato perlopiù warning relativi al fatto che il model checker non supporta strutture dati create ad hoc dall'utente. Inoltre, dalla versione 0.8.0 di Solidity, i controlli su

overflow e underflow sono effettuati di default, e quindi non è stato necessario aggiungere dei require specifici.

Unit Test in Python

Sono inoltre presenti altri quattro moduli extra che contengono gli Unit test che sono stati eseguiti per assicurarsi che l'applicazione funzionasse correttamente. I quattro moduli di test sono:

- **test_filter:** In questo modulo vengono testate le funzioni di filtro che si trovano nel modulo filter, e quindi le funzionalità comuni a tutte le classi di utenti. Sono stati quindi testati tutti i singoli filtri che l'applicazione fornisce, e sono anche testati filtri combinati in and e in or.
- **test_supplier:** In questo modulo si testa la creazione di una nuova materia prima da parte di un fornitore, e si testano anche vari casi di misuso per assicurarsi che siano ben gestiti, attraverso delle eccezioni, e che non causino problemi, o fallimenti inaspettati. Nello specifico viene testato il comportamento del software quando avviene:
  - l'inserimento di un nome vuoto di una materia prima.
  - l'inserimento di una materia prima con valore di carbon footprint nullo.
  - l'inserimento di una materia prima già presente (due materie prime sono considerate la stessa se hanno lo stesso nome, numero di lotto e indirizzo del fornitore).
  - l'inserimento di una materia prima con un indirizzo di destinazione non associato ad un utente di tipo "trasformatore".
  - L'inserimento di un valore della carbon footprint che causa un errore di overflow.
- **test_transformer:** In questo modulo si testano le funzionalità a disposizione dei trasformatori, e i casi di misuso rilevati. I casi di misuso testati sono:
  - il trasferimento di un prodotto a sé stessi.
  - l'inserimento di un prodotto con nome vuoto
  - il tentativo di aggiunta di una nuova trasformazione ad un prodotto la cui lavorazione è terminata.
  - il tentativo di trasferimento di un prodotto di cui la lavorazione è terminata, ad un altro trasformatore.
  - l'aggiunta di una nuova trasformazione ad un prodotto non posseduto dall'utente corrente.
  - l'utilizzo di una materia prima non posseduta dal trasformatore corrente.
- **test_validation:** In questo modulo sono testate le validazioni sull'input inserito dall'utente. Sono testate varie possibili stringhe e valori numerici che non devono essere accettati dal sistema, come stringhe vuote, stringhe contenenti caratteri non alfanumerici o valori negativi per la carbon footprint.

Per eseguire questi test non è necessario connettersi effettivamente alla blockchain, tranne che per i test relativi alle funzionalità specifiche di trasformatore e fornitore, che necessitano di richiedere l'esecuzione di transazioni.

In totale quindi si sono scritti 31 unit test che sono stati eseguiti e sono terminati tutti con successo.

```
(venv) C:\Users\devoz\PycharmProjects\progetto-software-cybersecurity>test.bat
test_and_filter (test_filter.FilterTest) ... ok
test_cf_filter (test_filter.FilterTest) ... ok
test_ended_filter (test_filter.FilterTest) ... ok
test_name_filter (test_filter.FilterTest) ... ok
test_or_filter (test_filter.FilterTest) ... ok
test_owner_filter (test_filter.FilterTest) ... ok
test_rm_name_filter (test_filter.FilterTest) ... ok
test_supplier_filter (test_filter.FilterTest) ... ok
test_transformer_filter (test_filter.FilterTest) ... ok
test_create_raw_materials (test_supplier.SupplierTest) ... ok
test_misues1_creation_raw_material (test_supplier.SupplierTest) ... ok
test_misues2_creation_raw_material (test_supplier.SupplierTest) ... ok
test_misues3_creation_raw_material (test_supplier.SupplierTest) ... ok
test_misuse4_creation_raw_material (test_supplier.SupplierTest) ... ok
test_overflow_creation_raw_material (test_supplier.SupplierTest) ... ok
test_a_create_products (test_transformer.TransformerTest) ... ok
test_b_add_transformation (test_transformer.TransformerTest) ... ok
test_c_transfer_product (test_transformer.TransformerTest) ... ok
test_d_transfer_product_to_self_failed (test_transformer.TransformerTest) ... ok
test_e_final_transformation (test_transformer.TransformerTest) ... ok
test_f_create_product_failed_name_empty (test_transformer.TransformerTest) ... ok
test_g_add_transformation_failed_is_ended (test_transformer.TransformerTest) ... ok
test_h_transfer_product_failed (test_transformer.TransformerTest) ... ok
test_i_add_transformation_failed_is_not_owner (test_transformer.TransformerTest) ... ok
test_l_create_product_failed_not_rm_owner (test_transformer.TransformerTest) ... ok
test_carbon_fp_input_validation (test_validation.ValidationTest) ... ok
test_id_input_validation (test_validation.ValidationTest) ... ok
test_lot_input_validation (test_validation.ValidationTest) ... ok
test_name_input_validation (test_validation.ValidationTest) ... ok
test_supplier_address_validation (test_validation.ValidationTest) ... ok
test_transformer_address_validation (test_validation.ValidationTest) ... ok

----------------------------------------------------------------------
Ran 31 tests in 5.804s

OK
```