UNIVERSITÀ DEGLI STUDI
DI SALERNO

MuSe Remix Plugin

Test Plan

Docente / Tutor

Andrea De Lucia

Gerardo Iuliano

Studenti

Carmine Calabrese (CC)
0522501853

Daniele Carangelo (DC)
0522501696

Data	Versione	Autori
02/09/2025	1.0	DC, CC

Table of Contents

1 Introduzione

MuSe (Mutation Seeding tool) è uno strumento basato su tecniche di mutation testing progettato per generare **benchmark di sicurezza** in ambito blockchain. Il tool è basato sul framework SuMo (SOlidity MUtator) e mira ad introdurre vulnerabilità note in smart contract scritti in solidity. MuSe sfrutta un motore di parsing AST per individuare pattern all'interno del codice e inserire specifiche mutazioni. Attualmente il tool è utilizzabile esclusivamente tramite interfaccia a **riga di comando (CLI)** in ambiente locale.

2 Testing Strategy

2.1 Unit Testing

I test di unità sono stati realizzati per verificare il corretto funzionamento dei singoli componenti dell'applicazione in modo isolato. È stato adottato un approccio di *black-box testing*, che ha permesso di progettare i casi di test considerando esclusivamente gli input e gli output attesi delle funzioni, senza basarsi sulla loro implementazione interna.In particolare, è stato applicato il metodo di Category Partition, che consente di individuare sistematicamente le diverse classi di input e le relative combinazioni. Per l'implementazione ed esecuzione dei test è stato utilizzato il framework **Jest**, scelto per la sua integrazione con l'ecosistema JavaScript/React. In questa tipologia di test sono coinvolte le `change request CR_01` e `CR_02`.

2.2 Integration Testing

I test di integrazione hanno verificato la corretta integrazione tra il tool MuSe e il backend del Plugin. La change request interessata è la CR_01. I tool utilizzati sono Jest e Supertest.

2.3 System Testing

Nei test di sistema è stato verificato che l'intero sistema si comporti come previsto. In questa fase non vengono utilizzati mock: viene testata l'applicazione completa, dal front-end fino all'integrazione con il tool MuSe. L'approccio adottato è di tipo black-box, utilizzando Playwright come framework di testing. Le change request interessate sono CR_01 e CR_02.

2.4 Regression Testing

Nel tool MuSe non sono presenti test case utilizzabili per il regression testing durante la fase di manutenzione. Invece di creare nuovi casi di test, è stato adottato un approccio basato sui golden file. L'idea consiste nell'eseguire MuSe su input standard e ben definiti e salvare i report generati come riferimento. Durante la fase di testing, MuSe viene eseguito nuovamente (questa volta attraverso il wrapper) sugli stessi input, e i nuovi report prodotti vengono confrontati con i golden file precedentemente salvati, verificando eventuali regressioni. Anche in questo caso il framework utilizzato per automatizzare i test è stato Jest.

3 Unit Testing

3.1 CopyTestConfig

Parametro: `FileList`	
Nome Categoria	**Scelta per la categoria**
Contenuto Lista [C]	1. Lista contiene degli elementi 2. Lista vuota 3. Lista non letta correttamente [**ERROR**]
Parametro: `isFile`	
Nome Categoria	**Scelta per la categoria**
File Type [T]	1. Contenuto della lista è un file 2. Contenuto della lista non è un file

Test Case ID	**Test Frame**	**Esito atteso**
TC-U_1.1	`C1, T1`	Tutti i file presenti nella cartella vengono copiati nella destinazione
TC-U_1.2	`C2`	Nessun file viene copiato, il metodo termina normalmente
TC-U_1.3	`C1, T2`	Gli elementi non file vengono ignorati, solo i file vengono copiati
TC-U_1.4	`C3`	[**ERROR**] Il processo termina con `process.exit(1)`

3.2 GetAllFiles

Parametro: `FileList`	
Nome Categoria	**Scelta per la categoria**
Contenuto Lista [C]	1. Lista contiene elementi 2. Lista vuota
Parametro: `isDirectory`	
Nome Categoria	**Scelta per la categoria**
Tipo elemento [T]	1. È una directory 2. Non è una directory (file)

Test Case ID	**Test Frame**	**Esito atteso**
TC-U_2.1	`C1, T1`	Restituisce una lita contenente tutti i file presenti nella cartella e nelle sottocartelle
TC-U_2.2	`C1,T2`	Restituisce una lista vuota

3.3 RunSumoCommand

Parametro: `command`	
Nome Categoria	**Scelta per la categoria**
Tipo comando [CMD]	1. `DISABLE` 2. `ENABLE` 3. `MUTATE` 4. `TEST` 5. Comando non valido
Parametro: `parameters`	
Nome Categoria	**Scelta per la categoria**
Parametri opzionali [P]	1. Array di stringhe 2. Array vuoto

Test Case ID	Test Frame	Esito atteso
TC-U_3.1	`CMD1`	Esegue il comando ‘npx sumo disable‘
TC-U_3.2	`CMD2, P1`	Esegue il comando ‘npx sumo enable PARAMS‘
TC-U_3.3	`CMD2, P2`	Restituisce "No operators to enable" senza eseguire il comando
TC-U_3.4	`CMD3`	Esegue il comando ‘npx sumo mutate‘
TC-U_3.5	`CMD5`	[**ERROR**] La Promise viene rifiutata con `"NO COMMAND FOUND"`

3.4 initializePlugin

Parametro: `ContractList`	
Nome Categoria	**Scelta per la categoria**
Contenuto Lista [CL]	1. Lista contiene degli elementi 2. Lista vuota 3. Lista non letta correttamente [**ERROR**]
Parametro: `Client`	
Nome Categoria	**Scelta per la categoria**
Client availability [CA]	1. Client creato con successo 2. Client non disponibile [**ERROR**]

Table 1: Vincoli tra le categorie CL, CA

Vincolo	Descrizione
V1	Se `CA => CA2`, la funzione termina subito. La categoria `CL` non è rilevante.

Test Case ID	Test Frame	Esito atteso
TC-U_4.1	`CA2`	[**ERROR**] console log con "Failed to initialize plugin"
TC-U_4.2	`CL1,CA1`	Plugin inizializzato, contratti caricati nel plugin e console log con "Load X contracts"
TC-U_4.3	`CA1, CL2`	Plugin inizializzato e console log con "No contracts found in contracts folder"
TC-U_4.4	`CA1, CL3`	[**ERROR**] console log con "Initialization error"

3.5 executeMutations

Parametro: `SelectedContract`	
Nome Categoria	**Scelta per la categoria**
Stato selezione contratto [SC]	1. Contratto selezionato 2. Contratto non Selezionato
Parametro: `selectedMutators`	
Nome Categoria	**Scelta per la categoria**
Mutanti selezionati [MS]	1. Mutanti selezionati = 0 2. Mutanti selezionati > 0
Parametro: `mutatorResponse (API)`	
Nome Categoria	**Scelta per la categoria**
API Response [AR]	1. response: OK e Mutanti generati = 0 2. response: OK e Mutanti generati > 0 3. response: KO [**ERROR**]

Table 2: Vincoli tra le categorie SC, MS e AR

Vincolo	Descrizione
V1	Se `SC => SC2`, la funzione termina subito. Le categorie `MS` e `AR` non sono rilevanti.
V2	Se `SC => SC1` ma `MS => MS1`, la funzione termina subito. La categoria `AR` non è rilevante.

Test Case ID	Test Frame	Esito atteso
TC-U_5.1	`SC1, MS2, AR2`	Console log con "File saved successfully"
TC-U_5.2	`SC1,MS1`	Console log con "Please select at least one mutation operator"
TC-U_5.3	`SC2`	Console log con "Please select a contract first"
TC-U_5.4	`SC1, MS2, AR1`	Console log con "No mutants generated..."
TC-U_5.5	`SC1, MS2, AR3`	Console log con "Error during mutation execution" [**ERROR**]

3.6 executeTesting

Parametro: `SelectedContract`	
Nome Categoria	**Scelta per la categoria**
Stato selezione contratto [SC]	1. Contratto selezionato 2. Contratto non Selezionato
Parametro: `testingFramework`	
Nome Categoria	**Scelta per la categoria**
Test file [TF]	1. Test file per il framework selezionato esiste 2. Test file per il framework selezionato non esiste
Parametro: `mutatorResponse (API)`	
Nome Categoria	**Scelta per la categoria**
API Response [AR]	1. response: OK e report generato 2. response: KO e report non generato 3. response: fetch KO [**ERROR**]

Table 3: Vincoli tra le categorie SC, TF e AR

Vincolo	**Descrizione**
V1	Se `SC => SC2`, la funzione termina subito. Le categorie `TF` e `AR` non sono rilevanti.
V2	Se `SC => SC1` ma `TF => TF2`, la funzione termina subito. La categoria `AR` non è rilevante.

Test Case ID	Test Frame	Esito atteso
TC-U_6.1	`SC1, TF1, AR1`	Console log con "Report saved to /MuSe/results/report.html"
TC-U_6.2	`SC1,TF2`	Console log con "No test files found for the selected contract and framework"
TC-U_6.3	`SC2`	Console log con "Please select a contract first"
TC-U_6.4	`SC1, TF1, AR2`	Console log con "Testing error"
TC-U_6.5	`SC1, TF1, AR3`	Console log con "Error during testing" [**ERROR**]

3.7 Utility Testing

ID	Funzione	Input	Output / Stato atteso
TC-U_7.1	`updateConsole`	"Messaggio di test"	Il messaggio viene aggiunto in `consoleMessages` con timestamp formattato.
TC-U_7.2	`clearConsole`	—	`consoleMessages` diventa vuoto (`[]`).
TC-U_7.3	`loadContracts`	client = null	`contracts = []`
TC-U_7.4	`getContractsFromRemix`	ignore.txt, keep.sol	`contracts = [keep.sol]`
TC-U_7.5	`getContractsFromRemix`	errore da fileManager	`consoleMessages` contiene il messaggio "Error reading contracts"
TC-U_7.6	`getTestFiles`	tests/a.js, tests/folder	`files = [a.js]`
TC-U_7.7	`getTestFiles`	errore da fileManager	`files = []`

4 Integration Testing

Durante la fase di *integration testing* è stata verificata la corretta configurazione degli endpoint delle API, unitamente al controllo della loro effettiva funzionalità attraverso test reali di comunicazione. Per la fase di testing sono stati creati due file ad-hoc funzionanti, mediante l'uso di LLM, per poter garantire un test che sia il più vicino ad un utilizzo reale:

- **SimpleToken.sol:** Uno smart contract che prevede un token ed anesse operazioni come lettura, scrittura, deploy, etc...
- **SimpleTokenTest-Brownie.py:** Un file di testing compatibile col framework *Brownie*

4.1 API: Import

4.2 Return di una lista di file - TC-I_001

Endpoint	`/api/files-to-import`
Method	`GET`
Body	`N/A`
Test Steps	1. GET su /api/files-to-import.
Expected Results	Stato 200, body contenente un array con i file presenti

4.3 API: Save

4.3.1 Save File Successfully - TC-I_002

Endpoint	`/api/save`
Method	`POST`
Body	`{ filename: "test.sol", content: "pragma solidity" }`
Test Steps	1. POST su /api/save con filename e content validi
Expected Results	Stato 200, body con `message` contenente "File saved"

4.3.2 Return Error for Missing Data - TC-I_003

Endpoint	`/api/save`
Method	`POST`
Body	`{}`
Test Steps	1. POST su /api/save senza fornire filename o content
Expected Results	Stato 400, body con `error` contenente "Missing filename"

4.4 API: Mutate

4.4.1 Mutation Success - TC-I_004

Endpoint	`/api/mutate`
Method	`POST`
Pre-Condition	Una copia del contratto deve essere presente nella cartella `/MuSe/contracts` (copiata durante l'inizializzazione dei test)
Body	`{ mutators: [ value: "VVR" ] }`
Test Steps	1. POST su /api/mutate con mutators validi
Expected Results	Stato 200, mutazione applicata correttamente

4.4.2 Error for Missing Mutators - TC-I_005

Endpoint	`/api/mutate`
Pre-Condition	Una copia del contratto deve essere presente nella cartella `/MuSe/contracts` (copiata durante l'inizializzazione dei test)
Method	`POST`
Body	`{}`
Test Steps	1. POST su /api/mutate senza fornire mutators
Expected Results	Stato 500, errore indicante che i mutators sono richiesti

4.5 API: Test

4.5.1 Corretta esecuzione del Mutation Testing - TC-I_006

Endpoint	`/api/test`
Pre-Condition	Esecuzione di `/api/mutate` e file di testing compatibile
Method	`POST`
Body	• `testingConfig: brownie` • `testingTimeout: 300` • `testFiles = "SimpleTokenTest-Brownie.py"`
Test Steps	1. POST su /api/test con file presenti nel body
Expected Results	• Stato 200 • `body.output` definito • `body.report` definito

5 System Testing

5.1 Nessun mutante generato - TC-S01

Test Case ID	TC-S_01
Test Case Scenario	Nessun mutante generato
Pre-condition	Il Plugin è correttamente caricato e l'utente ha inserito un contratto
Test Steps	1. Selezionare il contratto da mutare 2. Selezionare l'operatore da utilizzare 3. Cliccare su "Mutate"
Test Data	Contract: Simpletoken.sol operator: ER
Expected Results	L'utente riceve l'avviso in console: "No mutans generated"

5.2 Test completato con successo - TC-S02

Test Case ID	TC-S_02
Test Case Scenario	Test completato con successo
Pre-condition	Il Plugin è correttamente caricato e l'utente ha inserito un contratto ed un test file
Test Steps	1. Selezionare il contratto da mutare 2. Selezionare l'operatore da utilizzare 3. Cliccare su "Mutate" 4. Cliccare su "Test" 5. Scegliere il test framework da utilizzare 6. Cliccare su "RUN"
Test Data	Contract: Simpletoken.sol operator: ER Test: SimpleToken_brownie.py
Expected Results	L'utente riceve l'avviso in console: "report saved"

5.3 Plugin caricato con successo - TC-S03

Test Case ID	TC-S_03
Test Case Scenario	Plugin caricato con successo
Pre-condition	
Test Steps	1. Cliccare sul tasto del Plugin Manager 2. Cliccare su "Connect to an external plugin" 3. Inserire Plugin name 4. Inserire Display name 5. Inserire URL 6. Cliccare su "OK"
Test Data	Plugin name: muse Display name: muse URL: `https://carmineh.github.io/MuSe-Remix-Plugin/`
Expected Results	L'Iframe del plugin è visibile nel DOM

5.4 Mutation completata con successo - TC-S04

Test Case ID	TC-S_04
Test Case Scenario	Mutation completata con successo
Pre-condition	Il Plugin è correttamente caricato e l'utente ha inserito un contratto
Test Steps	1. Selezionare il contratto da mutare 2. Selezionare l'operatore da utilizzare 3. Cliccare su "Mutate"
Test Data	Contract: Simpletoken.sol operator: BOR
Expected Results	L'utente riceve l'avviso in console: "File saved successfully"

5.5 Nessun operatore selezionato - TC-S05

Test Case ID	TC-S_05
Test Case Scenario	Nessun operatore selezionato
Pre-condition	Il Plugin è correttamente caricato e l'utente ha inserito un contratto
Test Steps	1. Selezionare il contratto da mutare 2. Cliccare su "Mutate"
Test Data	Contract: Simpletoken.sol
Expected Results	L'utente riceve l'avviso in console: "Please select at least one mutation operator"

5.6 Nessun contratto selezionato - TC-S06

Test Case ID	TC-S_06
Test Case Scenario	Nessun contratto selezionato
Pre-condition	Il Plugin è correttamente caricato e l'utente ha inserito un contratto
Test Steps	1. Cliccare su "Mutate"
Test Data	
Expected Results	L'utente riceve l'avviso in console: "Please select a contract first"

5.7 Nessun file di test trovato - TC-S07

Test Case ID	TC-S_07
Test Case Scenario	Nessun file di test trovato
Pre-condition	Il Plugin è correttamente caricato e l'utente ha inserito un contratto
Test Steps	1. Selezionare il contratto da testare 2. Cliccare su "Test" 3. Selezionare il framework di testing da utilizzare 2. Cliccare su "RUN"
Test Data	Contract: SimpleToken.sol Framework: Truffle
Expected Results	L'utente riceve l'avviso in console: "No test files found"

6 Regression Testing

Test Case ID	TC-R_01
Test Case Scenario	Testing eseguito correttamente
Pre-condition	MuSe (No wrap) ha eseguito il test su *Test Data* e generato il report sumo-log.txt
Test Steps	1. Eseguita mutazione su Simple.sol 2. Eseguito testing con Simple_brownie.py 3. Generato report con MuSe (Wrapped)
Test Data	Contract: Simple.sol Test file: Simple_brownie.py
Expected Results	I due report risultano essere identici