Raport: Concepte și metode din articolul „TestLab: An Intelligent Automated Software Testing Framework”[1]

Stoinea Maria Miruna

Nazare Elena-Denisa

Ciurescu Irina Alexandra

Antonescu Ionut-Andrei

1. Introducere/Context

Deoarece sistemele software sunt peste tot, ele nu mai sunt doar opționale sau de nișă, ci parte fundamentală a vieții de zi cu zi. Astfel, aplicațiile au devenit din ce în ce mai mari și mai complicate. Nu mai e vorba de aplicații simple, ci de sisteme complexe, cu sute de mii de linii de cod și componente interconectate. Această complexitate are un cost: durează mai mult să proiectezi, implementezi, testezi și întreții software. [2]

Articolul abordează problema calității scăzute a software-ului, cauzată adesea de testarea insuficientă sau inexistentă, din dorința de a accelera ciclul de dezvoltare. În acest context, este propus TestLab, un cadru inteligent pentru testarea automată a software-ului, care îmbină diverse metode de testare cu tehnici de Inteligență Artificială (IA) pentru a asigura testarea continuă și eficientă a aplicațiilor software.

TestLab este gândit să funcționeze pe toată durata dezvoltării software:

- Dezvoltatori: verifică codul sursă.
- Testeri: generează cazuri de test.
- Utilizatori finali: validează comportamentul aplicației.
  Și face asta la toate nivelurile: unitate, integrare, sistem și acceptanță.

El este împărțit în 3 componente specializate, fiecare cu un rol: descoperă vulnerabilități în API-uri (FuzzTheREST), analizează codul pentru riscuri (VulnRISKatcher), generează automat teste (CodeAssert). Vom discuta mai pe larg aceste componente în capitolele următoare.

2. Concepte-cheie

Autorii subliniază cele două obiective fundamentale ale testării:

- Validare: sistemul construit răspunde cerințelor utilizatorului? (este realizată de utilizatori și de către client)
- Verificare: implementarea respectă specificațiile și regulile? (este realizată de către o echipă tehnică)

Testarea automată a software-ului (Automated Software Testing): Executarea testelor fără intervenție umană, cu scopul de a identifica erori și vulnerabilități în mod rapid și eficient.

Testarea pe mai multe niveluri:

testarea la nivel de unitate - pe componente izolate (unit testing),

integrare – testarea interacțiunii dintre module (integration testing),

sistem – testarea sistemului în ansamblu și cum funcționează acesta (system testing),

acceptanță - efectuată de utilizatori pentru a valida cerințele (acceptance testing).

Tipuri de testare:

- White-box: Analizează logica internă a codului.
- Black-box: Testează funcționalitatea fără a cunoaște implementarea internă.
- Grey-box: Combină cele două abordări de mai sus.

Inteligență Artificială în testare: Utilizarea tehnicilor de machine learning și reinforcement learning pentru generarea automată de cazuri de test și detectarea vulnerabilităților.

După compararea unora dintre cele mai folosite framework-uri de testare automată, autorii au ajuns la concluzia că, deși există tool-uri automate și metode AI, niciuna nu oferă o soluție completă care să acopere toate tipurile de testare (black, white, grey), să funcționeze la toate nivelurile (unit, integration, system, acceptance) și să integreze toate metodele într-un singur cadru automatizat. De aici rezultă necesitatea și originalitatea TestLab. [3]

3. Structura și modulele TestLab

TestLab este compus din trei module principale, ce au diferite roluri.

Modulele 1 și 2 (FuzzTheREST și VulnRISKatcher) se concentrează pe securitate, adică găsirea punctelor slabe în aplicație. O fac din perspective diferite: unul „din afară”, ca un utilizator; altul „din interior”, analizând codul.

Al treilea modul (CodeAssert) merge mai departe decât tool-urile clasice de testare automată: el creează singur testele, analizând direct codul. Nu mai ai nevoie de un om care să scrie testele manual.

a. FuzzTheREST

- Fuzzer inteligent pentru API-uri REST, bazat pe Reinforcement Learning.
- Funcționează în mod black-box, generând input-uri deformate și folosind feedback-ul API-ului pentru a găsi vulnerabilități.
- Poate fi adaptat la grey-box prin analiza execuției codului.

b. VulnRISKatcher

- Utilitar de detectare a vulnerabilităților în cod sursă, folosind modele de machine learning.
- Suportă mai multe limbaje de programare și funcționează chiar și cu porțiuni de cod, fără a avea nevoie de context complet.

c. CodeAssert

- Automatizează generarea de scripturi de test folosind analiza codului și Natural Language Processing.
- Creează cazuri de test cu acoperire 100% la nivel de cod, pentru testare unitară și de integrare (white-box).

4. Exemplu practic

a. Aplicația peste care vrem să aplicăm framework-ul de testare

Prezentare succintă a aplicației Developer Toolbox

Despre aplicație: Developer Toolbox este o platformă interactivă dedicată învățării și exersării abilităților tehnice, în special în domeniul programării. Aplicația oferă un mediu în care utilizatorii pot adresa întrebări tehnice, răspunde la întrebările altora și pot rezolva exerciții de programare. De asemenea, permite utilizatorilor să urmărească progresul lor prin statistici detaliate și să participe la provocări săptămânale pentru a-și îmbunătăți abilitățile.

Funcționalități principale:

- **Întrebări și răspunsuri tehnice**: Utilizatorii pot întreba și răspunde la întrebări tehnice, pot salva întrebările și răspunsurile și le pot evalua.
- **Exerciții de programare**: Platforma oferă o varietate de exerciții de codare pe diferite domenii și nivele de dificultate. Utilizatorii pot rezolva exerciții direct pe platformă, într-un editor de cod integrat.
- **Evaluare și recompense**: Utilizatorii pot câștiga puncte de reputație și medalii pentru contribuțiile lor la platformă. Acestea vor fi afișate într-un clasament global.
- **Provocări săptămânale**: Administratorii pot crea provocări de codare săptămânale, iar utilizatorii pot participa pentru a câștiga recompense suplimentare.

- **Administrare și moderare**: Administratorii și moderatorii pot adăuga sau elimina întrebări, răspunsuri și exerciții. Ei pot gestiona, de asemenea, utilizatorii și pot aplica reguli pentru a asigura respectarea ghidurilor comunității.

Beneficii pentru utilizatori:

- Începătorii pot învăța prin rezolvarea de exerciții și obținerea de răspunsuri la întrebările lor.
- Utilizatorii experimentați pot contribui la comunitate prin ajutorul oferit altora și pot rezolva exerciții avansate.
- Platforma încurajează competiția și învățarea continuă prin provocări și recompense.

Tehnologii utilizate:

- **ASP.NET Core**, **C#**: Pentru crearea aplicației web și integrarea cu bazele de date.
- **Entity Framework Core**, **LINQ**: Pentru gestionarea datelor.
- **SQL Server**: Pentru persistarea datelor.
- **JavaScript, HTML, CSS**: Pentru interfața utilizatorului.
- **Python**: Folosit pentru implementarea serverului backend și logica compilatorului.

Scopuri pentru testare: Aplicația va fi utilizată pentru a testa funcționalitățile sale, inclusiv logarea utilizatorilor, gestionarea întrebărilor și răspunsurilor, rezolvarea exercițiilor și integrarea sistemului de recompense. Testele vor ajuta la identificarea problemelor și la îmbunătățirea performanței platformei.

b. Implementarea primului modul: FuzzTheREST

RESTler[4] este un instrument automatizat de fuzzing destinat testării API-urilor REST și identificării de erori și vulnerabilități de securitate în aplicațiile care folosesc aceste API-uri. Este un instrument de fuzzing de tip „stateful” (cu stare), care poate înregistra și analiza interacțiunile anterioare ale aplicației pentru a găsi vulnerabilități legate de comportamentele statice ale acesteia.

Prin contrast cu alte instrumente de fuzzing, RESTler nu se bazează doar pe generarea aleatorie de inputuri, ci folosește învățarea prin recompensă (Reinforcement Learning - RL) pentru a ghida căutarea de valori de input corecte, care pot descoperi vulnerabilități. Acesta poate îmbunătăți eficiența procesului de fuzzing, evitând pierderile de timp în căutarea unor inputuri greșite. În plus, RESTler poate fi configurat să includă informații de tip „grey-box”, ceea ce înseamnă că poate integra analiza execuției codului pentru a obține metrice în timp real, cum ar fi acoperirea codului.

Cum funcționează RESTler:

- **Inputul utilizatorului**: Testerul furnizează informații precum fișierul de specificație OpenAPI, fișierul de scenarii și parametrii pentru algoritmul de RL.
- **Generarea de inputuri**: Algoritmul de RL este folosit pentru a genera inputuri, iar acestea sunt mutate pe baza feedback-ului primit de la API-ul testat.
- **Evaluarea vulnerabilităților**: Fiecare input generat este trimis la API pentru a verifica dacă există erori sau vulnerabilități. Algoritmul înregistrează aceste informații și ajustează căutarea în continuare, pentru a maximiza eficiența și acoperirea.
- **Raportul de testare**: După finalizarea testelor, instrumentul produce un raport care detaliază vulnerabilitățile descoperite, valorile de input generate și orice alte metrice relevante pentru evaluarea securității API-ului.

Acest instrument este deosebit de util pentru testarea securității API-urilor REST într-un mod automatizat, reducând efortul necesar pentru identificarea problemelor de securitate și îmbunătățind eficiența procesului de testare. De asemenea, poate contribui la identificarea unor vulnerabilități care nu ar fi ușor de depistat în testele manuale tradiționale.

5. Concluzii

TestLab este un cadru modern și inteligent care propune o abordare integrată a testării software, combinând metode clasice cu AI pentru a atinge:

- Testare continuă,
- Acoperire completă,
- Detectarea timpurie a erorilor și vulnerabilităților,
- Reducerea efortului uman prin automatizare extinsă.

6. Referințe

[1] Dias, T., Batista, A., Maia, E., & Praça, I. (2023). TestLab: An Intelligent Automated Software Testing Framework. Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development (GECAD), ISEP, Porto.

[2] "The prevalence of software systems has become an integral part of modern-day living."/"Software usage has increased significantly, leading to its growth in both size

and complexity."/"Consequently, software development is becoming a more time-consuming process." from TestLab article.

[3]**"The integration of Artificial Intelligence (AI) in the software testing process is promising..."/"...no prior work has addressed the development of a comprehensive framework comprising multiple testing methods..."** from TestLab article.

[4] Microsoft RESTler: Stateful REST API for fuzzing