SIGURNOST OPERACIJSKIH SUSTAVA I APLIKACIJA 2021./2022.
Treća laboratorijska vježba: DevSecOps

Uvod

DevSecOps podrazumijeva ugradnju razmatranja o sigurnosti u sve faze životnog ciklusa razvoja softvera. Ova laboratorijska vježba se fokusira na uzak podskup tog područja, automatizaciju kontinuirane integracije i kontinuirane isporuke (CI/CD) kroz automatsko testiranje i provjeru sigurnosnih propusta u kodu. Valja naglasiti da se isto može ostvariti s mnoštvom alata široke palete mogućnosti, no na vježbi će zbog vremenskog ograničenja biti obrađen samo jedan pojednostavljeni slučaj, u kojem se neposredno prije postavljanja koda u produkcijsko okruženje obavlja testiranje i analiza koda za pronalaženje čestih sigurnosnih propusta. U sklopu vježbe se neće ulaziti u sam proces razvoja softvera, već će se za potrebe demonstracije krenuti od postojećeg koda s pogreškama i lošim praksama.

Automatski testovi se često koriste u praksi jer provjeravaju rubne slučajeve u kojima se kod može ponašati neispravno, te osiguravaju da nove značajke i izmjene u kodu ne uvedu kritične greške i ne naruše funkcionalnost starijih verzija softvera. U metodologiji razvoja *test driven development* testovi se pišu prije samog koda softvera, na temelju zahtjeva, te se razvoj softvera svodi na pisanje koda koji zadovoljava zadane testne slučajeve.

Zadaci

U sklopu laboratorijske vježbe potrebno je ostvariti tri cilja. Prvo je pomoću Python modula *unittest*, u skladu s pripadajućom dokumentacijom, potrebno definirati testove za kod u Dodatku A. U tom kodu je jedna očevidna pogreška vidljiva u metodi *perform_division*, koja ne vraća vrijednost u skladu s opisom, na što bi mogao ukazati odgovarajući test. Preporuka je da studenti probaju pronaći još rubnih slučajeva i nedostataka i po potrebi definiraju dodatne testove za priloženi kod. Za potrebe demonstracije je preporučljivo i dopuniti kod iz Dodatka A s još nekim metodama koje sadrže namjerno ubačene pogreške, te u skladu s tim pogreškama definirati odgovarajuće dodatne testove.

Nakon definicije i provedbe testova potrebno je alatom Bandit provesti automatski sigurnosni pregled koda. Alat Bandit pronalazi učestale sigurnosne nedostatke u kodu, te o pronađenim nedostacima ispisuje izvještaj na temelju kojeg je moguće poboljšati kod.

Treći cilj vježbe je automatizirati proces testiranja i postavljanja koda. U tu svrhu je potrebno postaviti Git repozitorij, primjerice koristeći platformu GitHub, te u njega postaviti izvorni kod prikazan u Dodatku A. Na vlastitom računalu je potrebno izraditi mapu *test*, koja simulira testno okruženje, mapu *production*, koja simulira produkcijsko okruženje i mapu *logs* u koju se zapisuju greške pronađene testiranjem i izvješća o otkrivenim sigurnosnim nedostacima koda. Naposljetku je pomoću alata Jenkins potrebno na vlastitom računalu definirati cjevovod koji automatski pokreće testiranje i postavljanje nove verzije koda u mapu *production*. Jenkins cjevovod treba obavljati sljedeće:

1. Kada se pokrene, cjevovod treba automatski klonirati repozitorij u mapu *test* na vašem računalu. Nije potrebno postaviti cjevovod tako da se automatski pokreće kada detektira novi commit u repozitoriju, no oni koji žele znati više mogu opcionalno isprobati i takvu konfiguraciju.

2. Nakon kloniranja repozitorija u direktorij *test*, cjevovod mora automatski pokrenuti definirane testove i otkrivanje potencijalnih sigurnosnih propusta alatom Bandit.
3. Ako testovi prođu bez prijavljenih grešaka i Bandit ne pronađe sigurnosne propuste, cjevovod treba kopirati novu verziju koda iz mape *test* u mapu *production* na vašem računalu, čime se simulira postavljanje koda u produkcijsku okolinu.
4. Ako testovi otkriju pogreške ili sigurnosne propuste, cjevovod treba pohraniti izvješća u odgovarajuće datoteke u dodatnu mapu *logs*, u čijem je nazivu istaknuta trenutna vremenska oznaka (timestamp).
5. Na kraju postupka cjevovod treba izbrisati ranije stvorenu mapu *test*.

Na temelju rezultata neuspješnog prvog pokušaja isporuke, potrebno je popraviti kod i obaviti commit kako bi sljedeća isporuka bila uspješna.

Oni koji žele znati više mogu kao opcionalni zadatak pokušati postaviti testno okruženje u Docker container za testiranje, te umjesto produkcijske mape na izlazu iz cjevovoda izgraditi produkcijski Docker container. Uporaba containera za obavljanje testiranja osigurava jedinstvenu okolinu koja može biti istovjetna produkcijskoj okolini. Testni container se može definirati jednokratno, te se mapu *test* u njega može mapirati kao *volume*, kako ne bi bilo potrebno svaki puta iznova graditi container. Također, opcionalno se u cjevovod može uključiti i dodatne alate za analizu koda. Opcionalni zadaci se ne boduju.

Preporučeni alati i korisni materijali

Za izgradnju cjevovoda za testiranje i isporuku koda je predviđena uporaba alata Jenkins. Alat Jenkins ima plugine za brojne platforme, uključujući GitHub. Za obavljanje testova se preporučuje uporaba Python modula *unittest*, a za sigurnosnu analizu uporaba alata Bandit. Dozvoljeno je koristiti i druge alate po izboru, no u slučaju uporabe takvih alata potrebno ih je ukratko opisati u izvješću. Na sljedećim poveznicama možete pronaći službenu dokumentaciju preporučenih alata:

- https://docs.python.org/3/library/unittest.html
- https://bandit.readthedocs.io/en/latest/
- https://www.jenkins.io/doc/book/getting-started/
- https://www.jenkins.io/solutions/github/

Predaja

Rezultate vježbe potrebno je predati putem mailing liste predmeta sosa@fer.hr, pri čemu je kao predmet (subject) potrebno napisati "SOSA - Predaja 3. laboratorijske vježbe". Predaja se mora sastojati od sljedećih datoteka:

- *"3_lab_<JMBAG>.pdf"* je izvješće koje mora sadržavati sljedeće točke:
    - Opis eventualnih promjena obavljenih na kodu iz Dodatka A
    - Opis razvijenih testova
    - Opis razvijenog Jenkins cjevovoda
    - Opis prvog povlačenja koda od strane cjevovoda, koje rezultira izvješćem o pogreškama i odbija postavljanje koda u mapu *production*
    - Opis izvješća o pogreškama i popravka pogrešaka, te uspješnog postavljanja koda, tj. njegovog kopiranja u mapu *production*, nakon git commita

- *“3_lab_<JMBAG>.zip”* je arhiva koja mora sadržavati sve izvorne datoteke i datoteke cjevovoda, bez virtual environmenta i izvršnih datoteka, kao i generirana izvješća testova i sigurnosnih pregleda koda. Ukoliko se u tim datotekama nalaze vaša korisnička imena i lozinke za pristup repozitoriju, potrebno ih je zamijeniti zvjezdicama.

Rok za predaju vježbe je **6.6.2022. u 7:59 ujutro**. Za ispunjavanje minimuma i prolaz predmeta potrebno je poslati rješenje laboratorijske vježbe do navedenog roka. Međutim, postoji parcijalno bodovanje laboratorija. Za priznavanje vježbe potrebno je minimalno implementirati i demonstrirati testove i pokrenuti skeniranje alatom Bandit. U slučaju problema ili nedoumica prilikom izrade vježbe molimo da pravovremeno kontaktirate nastavno osoblje putem mailing liste predmeta sosa@fer.hr ili putem platforme MS Teams.

Važno: Dozvoljeno je i poželjno diskutiranje mogućih pristupa rješavanju vježbe između studenata. Međutim, samu laboratorijsku vježbu studenti moraju raditi samostalno. Nastavno osoblje će provesti provjere sličnosti predanih rješenja, a ponašanje koje nije u skladu s Kodeksom ponašanja studenata FER-a ćemo prijaviti Povjerenstvu za stegovnu odgovornost studenata te odrediti dodatne sankcije u sklopu predmeta.

Dodatak A: Izvorni kod za demonstraciju

```
import getpass

class OperationsManager():

    def __init__(self, a: float, b: float) -> None:
        self.a = a
        self.b = b

    def perform_division(self) -> float:
        """Divides a with b. If b is zero, returns NaN."""
        return self.a / self.b

if __name__ == "__main__":
    user = input("Username: ")
    password = getpass.getpass("Password: ")
    if user != "root" or password != "123":
        print("Wrong username or password!")
        exit(0)
    else:
        print("Login success!")
        a = float(input("A = "))
        b = float(input("B = "))
        ops_manager = OperationsManager(a, b)
        print(ops_manager.perform_division())
```