Հակոբյան Ջիվան Անդրանիկի

**ՖԱԶԶ-ԹԵՍՏԱՎՈՐՄԱՆ ԱՐԴՅՈՒՆԱՎԵՏՈՒԹՅՈՒՆԸ ԲԱՐՁՐԱՑՆՈՂ ՄԵԹՈԴՆԵՐԻ ՀԵՏԱԶՈՏՈՒՄ ԵՎ ԻՐԱԿԱՆԱՑՈՒՄ**

Ատենախոսության շրջանակներում ուսումնասիրվել են ծրագրային ապահովումների հուսալիության բարձրացման և դրանցում սխալների հայտնաբերման դինամիկ մեթոդները։ Դիտարկվել է ֆազզ-թեստավորման մեթոդը և դրա համադրությունն այլ մեթոդների հետ։ Ուսումնասիրվել են ֆազզ-թեստավորում կատարող գործիքներն ու ի հայտ բերվել դրանցում առկա թերությունները:

Ուսումնասիրության արդյունքում պարզ դարձավ, որ գոյություն ունեցող թեստավորման գործիքները նախատեսված են միայն կոնկրետ խնդրի համար և չեն կարող համապարփակ կերպով արդյունավետորեն թեստավորել բարդ տրամաբանություն ունեցող տարբեր ծրագրային համակարգեր։

Java լեզվով գրված գրադարաններ թեստավորվող գործիքների ուսումնասիրության արդյունքում պարզվեց, որ գոյություն չունի այնպիսի թեստավորման համակարգ, որը կթեստավորի API ֆունկցիաների տարբեր կանչերի կոմբինացիներ՝ հաշվի առնելով տվյալների կախվածությունը։ Գոյություն ունեցող գործիքները սահմանափակվում են մի ֆունկցիայի կամ դասի անդամ ֆունկցիաների առանձին թեստավորմամբ։

Ծրագրային ապահովումների մեծամասնությունն օգտվում է մուտքային տվյալների գլխագրերից՝ դրանց ընդունելիությունն ստուգելու համար։ Պահանջներին չբավարարելու դեպքում նման մուտքային տվյալները անտեսվում են որպես խոտան։

Դիտարկվել են առկա թեստավորման գործիքներում մուտքային տվյալների գեներացման ալգորիթմները։ Ուսումնասիրության արդյունքում պարզ դարձավ, որ գրեթե բոլոր ալգորիթմները հաշվի չեն առնում թիրախային ծրագրի առանձնահատկությունները և մուտքային տվյալների գեներացիայի ժամանակ փոխում են գլխագրերը։ Իսկ այն գործիքները, որոնք հաշվի են առնում, կիրառելի են խիստ սահմանափակ քանակով տվյալների տիպերի համար։

Ելնելով վերոնշյալ թերություններից և սահմանափակումներից, ատենախոսության շրջանակում առաջ են քաշվել և լուծվել հետևյալ խնդիրները․

1. Նախագծել ֆազզ-թեստավորում կատարող համակարգ, որը կարող է հեշտությամբ ընդլայնվել նոր խնդիրներ լուծելու համար և թեստավորել նպատակային ծրագրի հետևյալ մուտքային աղբյուրները՝ ստանդարտ մուտքի հոսք, ֆայլեր, ցանցային տվյալներ, միջավայրի փոփոխականներ, մուտքային դրոշներ:
2. Հետազոտել և մշակել Java լեզվով գրված գրադարանների ֆազզ-թեստավորման համակարգ, որը կկարողանա թեստավորել փոխկապակցված API ֆունկցիաները։
3. Հետազոտել և մշակել նպատակային ծրագրի մուտքային տվյալների ձևափոխման արդյունավետ մեթոդ, որը հաշվի է առնում այդ տվյալների կառուցվածքային առանձնահատկությունները:

Ելնելով վերոնշյալ խնդիրներից, նախագծվել և իրականացվել է ISP-Fuzzer ֆազզ-թեստավորման համակարգը։ Ներդրված հավելվածների մեխանիզմի շնորհիվ թեստավորման համակարգն հնարավոր է ընդլայնել՝ ներառելով նոր ֆունկցիոնալություններ։ Համակարգում իրականացվել և ներդրվել են մեկ տասնյակից ավելի հավելվածներ՝ մուտքային տվյալների ձևափոխությունների, BNF (Backus–Naur form) կառուցվածք ունեցող տվյալների գեներացման, ԴՍԿ (դինամիկ սիմվոլիկ կատարում) գործիքի կիրառման, միջակայքային ձևափոխությունների, ցանցային հաղորդակցության թեստավորման, անոտացիաների գեներացման և մի շարք այլ հավելվածներ։

Համակարգի արդյունավետությունը բարձրացնելու նպատակով մշակվել է հավելյալ գործիք, որի միջոցով հնարավոր է համակարգը գործարկել բաշխված և զուգահեռ համակարգերում։

Իրականացված համակարգի արդյունավետությունը ստուգվել է մի շարք իրական ծրագրերի թեստավորմամբ, որի արդյունքում հայտնաբերվել են 280-ից ավելի սխալներ 14 տարբեր ծրագրերում։

Java լեզվով գրված գրադարանների վերլուծության համար իրականացվել և ISP-Fuzzer համակարգում ներդրվել են չորս հավելվածներ։ Առաջին հավելվածի միջոցով jar ֆայլից ավտոմատ դուրս են բերվում API ֆունկցիաների և դասերի մասին ամբողջական տվյալները՝ անոտացիաները։ Երկրորդի միջոցով անոտացիաների հիման վրա գեներացվում է API ֆունկցիաների կատարման ինտերպրետատոր։ Երրորդ հավելվածն իրականացնում է ISP-Fuzzer գործիքի և կատարող մեքենայի միջև հաղորդակցություն, իսկ վերջին՝ չորրորդ հավելվածը, սխալ հայտնաբերելու դեպքում գեներացնում է ամբողջական Java լեզվով գրված ծրագիր՝ սխալը վերարտադրելու համար։

API ֆունկցիաներ թեստավորելիս համակարգն ի վիճակի է գեներացնել միաժամանակ մի քանի ֆունկցիաների կանչեր, անգամ առանձին դասերին պատկանող։ Կոնֆիգուրացիոն ֆայլի միջոցով հնարավոր է սահմանել ֆունկցիաների միջև կապեր՝ մի ֆունկցիայի վերադարձի արժեքը փոխանցելով հաջորդին, ինչպես նաև սահմանել քայլերի հաջորդականություններ, որոնք կկատարվեն յուրաքանչյուր հաջորդականության մեկնարկից առաջ կամ հետո։ Հաղորդակցությունների հավելվածի միջոցով համակարգը կարող է միաժամանակ աշխատել մի քանի թեստավորման մեքենաների հետ։

 Առաջարկվող մեթոդը կիրառելի է իրերի համացանց (IoT) համակարգերի թեստավորման համար։ Մեթոդի միջոցով Samsung SmartThings ծրագրում հայտնաբերվել են 15 ունիկալ սխալներ։ Հայտնաբերված բոլոր սխալները հաստատվել են Samsung ընկերության Electronics R&D խմբի կողմից։

Մուտքային տվյալների ձևափոխության ժամանակ գլխագրերի կամ ծառայողական արժեքների ամբողջականությունն ապահովելու նպատակով մշակվել են միջակայքերի ձևափոխության ալգորիթմն ու ZC-DSE ստատիկ վերլուծության գործիքը։

Գործիքի միջոցով կապ է հաստատվում ծրագրում բազային բլոկի և մուտքային տվյալի առանձին հատվածի միջև։

Հայտնաբերված միջակայքերն ու արժեքները կիրառվում են մուտքային տվյալների միջակայքերի ձևափոխման ալգորիթմում։ Ձևափոխության ընթացքում նախ գեներացվում են պատահական տվյալներ, ապա հայտնաբերված միջակայքերի արժեքները ավելացվում են դրանցում։

Որպեսզի գնահատենք միջակայքային ձևափոխությունների ազդեցությունը մուտքային տվյալների գեներացիայի որակի վրա օգտագործվել են Ubuntu 18.04.2 LTS օպերացիոն համակարգի մեկ տասնյակից ավելի ծրագր։ Ծրագրերը նախ թեստավորվել են միջակայքային ձևափոխության ալգորիթմով ու առանց դրա, ապա համեմատվել ստացված արդյունքները։ Համեմատման արդյունքում պարզ դարձավ, որ միջակայքային ձևափոխության կիրառումն ունի բացահայտ առավելություն: Թեստավորման նույն պայմաններում մեթոդի կիրառմամբ հայտնաբերվել են 40% ավելի շատ կատարման ճանապարհներ և 37 սխալներ։

Ատենախոսության ընթացքում հայտնաբերված բոլոր սխալներն ստուգվել և հաստատվել են կրկնակի մեկնարկի միջոցով։