BAJARILADIGAN DASTUR KODINI DINAMIK VA STATIK TAHLILDAN HIMOYA QILISH USULLARI VA ALGORITMLARI

Musayev Sh.S.
"Kiberxavfsizlik markazi" davlat unitar korxonasi, O'zbekiston
Quryozov RB
Magistr, Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti,
Kiberxavfsizlik fakulteti

Ma'lumotlar tuzilmasi dasturlarda ajratish usuli bo'yicha statik va dinamikaga bo'lingan. Statik ma'lumotlar tuzilmasi - bu kompyuterning xotirasida joylashishi va elementlarning o'zaro aloqalari ular tomonidan amalga oshiriladigan sohada dasturni bajarish paytida o'zgarishsiz qoladigan ma'lumotlardir. Statik strukturaning ma'lumotlariga dasturda e'lon qilingan asosiy va mahalliy, ham global darajadagi o'zgaruvchilar kiradi. Dinamik ma'lumotlar tuzilmasi - bu kompyuterning xotirasiga joylashtirilishi va New va Dispose kabi tizim proseduralari yordamida dasturni bajarishda xotiradan o'chirilishi mumkin bo'lgan ma'lumotlar.

• Ma'lumotlar tuzilmasi dasturlarda ajratish usuli bo'yicha statik va dinamikaga bo'lingan. Statik ma'lumotlar tuzilmasi - bu kompyuterning xotirasida joylashishi va elementlarning o'zaro aloqalari ular tomonidan amalga oshiriladigan sohada dasturni bajarish paytida o'zgarishsiz qoladigan ma'lumotlardir. Statik strukturaning ma'lumotlariga dasturda e'lon qilingan asosiy va mahalliy, ham global darajadagi o'zgaruvchilar kiradi. Dinamik ma'lumotlar tuzilmasi - bu kompyuterning xotirasiga joylashtirilishi va New va Dispose kabi tizim proseduralari yordamida dasturni bajarishda xotiradan o'chirilishi mumkin bo'lgan ma'lumotlar.

Statik ma'lumotlar tuzilmasi vaqt o'tishi bilan o'z o'lchamini o'zgartirmaydi. Biz har doim dastur kodidagi statik ma'lumotlar tuzilmasiga qarab ularning o'lchamini bilishimiz mumkin. Bunday ma'lumotlarga teskari ravishda dinamik ma'lumotlar tuzilmasi mavjud bo'lib, bunda dastur bajarilishi davomida dinamik ma'lumotlar tuzilmasi o'lchamini o'zgartirishi mumkin. Dinamik ma'lumotlar tuzilmalari ikki shaklda bo'ladi: bog'liq bo'lmagan dinamik ma'lumotlar; bog'liq dinamik ma'lumotlar.

• Statik ma'lumotlar tuzilmasi vaqt o'tishi bilan o'z o'lchamini o'zgartirmaydi. Biz har doim dastur kodidagi statik ma'lumotlar tuzilmasiga qarab ularning o'lchamini bilishimiz mumkin. Bunday ma'lumotlarga teskari ravishda dinamik ma'lumotlar tuzilmasi mavjud bo'lib, bunda dastur bajarilishi davomida dinamik ma'lumotlar tuzilmasi o'lchamini o'zgartirishi mumkin. Dinamik ma'lumotlar tuzilmalari ikki shaklda bo'ladi: bog'liq bo'lmagan dinamik ma'lumotlar; bog'liq dinamik ma'lumotlar.

Hammasi bo'lib, dinamik ma'lumotlar tuzilishining 6 ta asosiy turi mavjud:

• Hammasi bo'lib, dinamik ma'lumotlar tuzilishining 6 ta asosiy turi mavjud:

• Stek
• Navbat
• Ro'yhat
• Daraxt
• Graf
• Ro'yxat. Ro'yxatning 3 turi mavjud: Bir bog'lamli (chiziqli)
• Ikki bog'lamli
• Siklik

Maqolaning asosiy g'oyasi shundaki, ikkilik kod tahlilining samaradorligini oshirish uchun dinamik va statik tahlil usullarini bir-biri bilan birgalikda qo'llash kerak. Maqolada mualliflar ikkilik kodni dekompilyatsiya qilish va dinamik bajarilishini tahlil qilish usullariga hissa qo'shadilar. Natijalar ruxsatsiz teskari muhandislikdan dasturiy ta'minotni himoya qilish muammosiga qo'llaniladi. Mualliflar dasturni chalkashtirib yuborish va uni tadqiqotdan himoya qilish uchun asosiy dasturni boshqarish oqimi algoritmini tahlil qilishdan foydalanganlar.

Dasturiy ta'minot kompaniyalari dasturiy ta'minot xavfsizligi bilan bog'liq jiddiy xavflarga duch kelishadi, chunki dasturiy ta'minotni ishlab chiqish jarayonida har doim ba'zi xatolar va xatolar mavjud. O'z navbatida, bu xatolar dasturiy ta'minotda jiddiy zaifliklarning paydo bo'lishi uchun qulay shart-sharoitlarni yaratishi mumkin. Zaifliklardan foydalanish nafaqat zaif mahsulotdan foydalanadigan tizim xavfsizligiga, balki dasturiy ta'minotni ishlab chiqish bilan shug'ullanadigan kompaniyaning raqobatdosh ustunliklariga ham ta'sir qiladi. Shuni ta'kidlash kerakki, zaifliklarni o'rganish jarayoni juda murakkab, chunki bu jarayonda bir qator fundamental texnologik muammolar mavjud. Bundan tashqari, amalda dasturiy ta'minot xavfsizligiga hujum qilishning har qanday usuli ikkilik kodni himoya qilish mexanizmini dastlabki qismlarga ajratish va tahlil qilishga asoslangan. Ushbu tadqiqotning maqsadi himoya algoritmini tiklash, uning zaif tomonlari va hujjatlashtirilmagan xususiyatlarini kelajakda o'zgartirish va (yoki) hujum jarayonini avtomatlashtirishdir. Ushbu muammolarni hal qilish uchun statik va dinamik ikkilik tahlil usullari keng qo'llaniladi. Ushbu maqolalarda mualliflar ikkilik kodda zaifliklarni aniqlash uchun dinamik va statik ikkilik tahlil qilish usullarini namoyish etdilar. Dinamik tahlil protsessorda dasturning bajarilishiga asoslanadi. O'z navbatida, statik tahlil xotirada PE-modul texnikasini tahlil qilishga asoslangan . Dasturiy ta'minotni bajarish oqimini tiklash uchun dinamik tahlil texnikasi qo'llaniladi. Mualliflar PE (portativ bajariladigan) modullarida zaiflik naqshlarini aniqlash uchun statik ikkilik tahlil qilish texnikasidan foydalanganlar. Mualliflar ikkilik ijro oqimining xiralashishi ruxsatsiz tadqiqotlarga qarshi dasturiy ta'minot xavfsizligi uchun samarali ekanligini ko'rsatdi. Mualliflar dasturiy ta'minotni chalkashtirish uchun dinamik ikkilik bajarish usullaridan foydalanadilar. O'z navbatida, muallif bajarilish va ma'lumotlar oqimi algoritmlarini tahlil qilish uchun statik analizatordan foydalanadi. Maqolada dinamik va statik

tahlilning original usullari, shuningdek, mavjudlarini optimallashtirish ko'rsatilgan. Himoya va zaifliklarni aniqlash samaradorligini oshirish uchun biz statik va dinamik tahlil usullari kombinatsiyasidan foydalandik. Statik analizatorni amalga oshirish uchun biz oldingi ishlarimizni dekompilyatsiya va oraliq tilni (IL) amalga oshirishdan foydalanishni taklif qilamiz. Shunday qilib, ushbu maqolada biz quyidagi hissa qo'shamiz:

1. IL tahlil texnikasi. Biz statik ikkilik tahlil uchun yangi sxemani taqdim etamiz — bu bizga oraliq tilda (IL) zaifliklarni qidirish imkonini beradi.

2. Dinamik bajarilishini tahlil qilish texnikasi. Maqolada dinamik ikkilik kodni bajarish uchun optimallashtirish algoritmlari keltirilgan.

3. Gibrid tahlil texnikasi. Texnika ruxsatsiz tadqiqotlardan dasturiy ta'minotni himoya qilish uchun statik va dinamik algoritmlarni birlashtiradi.

Statik tahlil texnikasini amalga oshirish uchun ikkilik kodni oraliq tilga talqin qilish kerak. Oldingi ishimizda biz ushbu ILni shakllantirgan va tavsiflagan edik. X86 mnemonikasini IL-ga tarjima qilish uchun biz 16 ta asosiy operatsiyalardan iborat bazadan foydalandik. Ushbu belgi bizga kodni manbaga yo'naltirilgan holda tasvirlash imkonini beradi. Ro'yxatga olish resursi resurslar soni kiritilgan bracerlar bilan belgilanadi. Xotira resursi manba manzili kiritilgan qavslar bilan ko'rsatilgan. Bunday resurslar soni qavs ichidan olib tashlangan. Natija «=» belgisi bilan ajratiladi. Amallar va konstantalar "xuddi shunday" yoziladi.

Shuni ta'kidlash kerakki, statik tahlil resurslarni ko'p talab qiladigan jarayondir, chunki tadqiqot tahlil qilingan dasturiy ta'minotning butun PE modulida amalga oshiriladi. Dinamik tahlil odatda statik tahlilga qaraganda aniqroqdir, chunki u ish vaqti rejimida haqiqiy qiymatlar bilan ishlaydi. Xuddi shu sababga ko'ra, dinamik tahlillar ko'pincha statik tahlillarga qaraganda ancha sodda.

Ushbu ikki yondashuv (statik va dinamik) bir-birini to'ldiradi. Texnikalarning kombinatsiyasi qo'llaniladi, chunki statik tahlil dasturning ish vaqti qiymatlarini tahlil qilishga imkon bermaydi, masalan: "Siz ko'rgan narsa siz bajarayotgan narsa emas ". Dinamik tahlilda dasturning barcha mumkin bo'lgan bajarish yo'llari va holatlarini tahlil qilish mumkin emas. Shunday qilib, ushbu maqolada biz yuqorida tavsiflangan usullarning kombinatsiyasidan foydalanamiz, ikkilik kodlar ketma-ketligi statik va dinamik analizator tomonidan tekshiriladi. Shuni ta'kidlash kerakki, yuqorida tavsiflangan texnologiyalarni amalga oshirishda asosiy muammo - bu unumdorlik. Statik tahlil paytida asosiy ishlash muammosi - butun dastur kodini tahlil qilish kerak, ammo katta dasturlarda bu muammoli. Dinamik tahlil jarayoni, shuningdek, ishlash bilan bog'liq jiddiy muammolarga ega, chunki dasturiy ta'minotdagi har bir ko'rsatmani ketma-ket tahlil qilish uchun texnikadan foydalaniladi. Keling, ushbu muammoni batafsil ko'rib chiqaylik. Ma'lumotlar oqimini tahlil qilish uchun samarali analizator imkon qadar ko'proq ma'lumot to'plashi kerak, u ma'lum bir bajarish yo'lining bir bosqichini bajarishi va dasturiy ta'minotni bajarishning har bir bosqichida registrlar

qiymatlarini saqlashi kerak. Bajariladigan modulda bir qadamni bajarishning bir necha usullari mavjud. Shuni ta'kidlash kerakki, ta'riflangan texnologiyalar dasturiy vositalar sifatida amalga oshirilgan.

1. «Rasmiy» interfeyslar orqali 32 API disk raskadrovkasini yutib oling. Buni amalga oshirish uchun biz avtomatik tuzatuvchini ishga tushirishimiz kerak, ijro kontekstida TF bit sozlamalari bo'yicha yagona qadamni majburlash va registrlar, xotira va bayroqlar haqida ma'lumot to'plash. Intel x86 protsessori murakkab ko'rsatmalar to'plami kompyuter (CISC) arxitekturasidan foydalanadi, ya'ni katta miqdordagi umumiy maqsadli registrlar o'rniga kam sonli maxsus maqsadli registrlar mavjud . Agar TF bayrog'i TRUE-ga o'rnatilgan bo'lsa, protsessor bitta buyruq bajarilgandan so'ng STATUS_SINGLE_STEP istisnosini ko'taradi. Biroq, bu sekin texnika, chunki har bir ko'rsatma va tuzatuvchidan so'ng kontekstni almashtirish kontekstni olishi va ijroni davom ettirishi kerak.

2. Dinamik ikkilik asboblar (DBI) texnikasi. Ikkilik asboblar - bu ikkilik dasturni kuzatish, kuzatish va o'zgartirish uchun bajarishdan oldin yoki bajarish vaqtida o'zgartiradigan texnika . Ta'riflanganidek, birinchi bosqichda ishga tushirish moslamasi sinov dasturini yuklaydi, DBI dll-ni asbob-uskunalar kodi bilan kiritadi, test ilovasining kirish nuqtasini hisoblab chiqadi, DBI dll- ga o'tish ko'rsatmalarini kiritadi va dasturni ishga tushiradi. O'tishdan keyin DBI kodi biroz tahlil qiladi va ikkinchi ko'rsatmada ikkinchi sakrashni kiritadi. Bu ikkilik kodni dinamik ravishda o'zgartirish imkoniyati bilan jarayon kontekstida samarali tahlil qilish imkonini beradi. Nosozliklarni tuzatish va DBI usullarini solishtirish uchun oddiy dastur yozildi, u ba'zi bir mezonlarni yig'ish uchun ma'lum bir necha marta tsiklni bajaradi. DBI orqali statik va dinamik tahlil usullari ikkilik kodni himoya qilish vazifasida sinovdan o'tkazildi. Ushbu texnikada himoyachi himoyalangan dasturiy ta'minotda qo'shimcha operatsiyalar va ko'rsatmalarni kiritadi.

Shuni ta'kidlash kerakki, statik tahlil resurslarni ko'p talab qiladigan jarayondir, chunki tadqiqot tahlil qilingan dasturiy ta'minotning butun PE modulida amalga oshiriladi. Dinamik tahlil odatda statik tahlilga qaraganda aniqroqdir, chunki u ish vaqti rejimida haqiqiy qiymatlar bilan ishlaydi. Xuddi shu sababga ko'ra, dinamik tahlillar ko'pincha statik tahlillarga qaraganda ancha sodda. Dinamik ikkilik tahlil texnikasini tavsiflab, keling, ikkilik kodning chiziqli chizmasi tushunchasini ko'rib chiqamiz. Ikkilik kodning bo'limi chiziqli bo'ladi, agar bo'limda ikkilik kodning boshqa bo'limiga boshqarish ko'rsatmalari mavjud bo'lmasa. Ko'rsatmalar , qaysi bor chiziqli bo'lmagan .

Ushbu maqolada tasvirlangan asosiy yondashuv dasturning bajarilishi to'g'risida ma'lumot berishdir, masalan: funktsiyalarni chaqirish, chiziqli uchastkalarda ishlatilgan resurslar. Ushbu yondashuv dasturning ba'zi bloklarini bajarishga va har bir bajarilgan chiziqli blokdan keyin CPU registrlari, bayroqlar va xotira ostida boshqarishni amalga oshirishga imkon beradi. Ushbu bajarish natijalari dasturning boshqaruvi va

ma'lumotlar oqimini kuzatish va ularning har birining natijasini saqlash bilan dastur bosqichlari ketma-ketligini tavsiflash imkonini beradi.

FOYDALANILGAN ADABIYOTLAR RO`YXATI:

1. Stiven R. Devis. Dummies uchun C++. Dialektik. Moskva - 2019 yil. 337 b.
2. Stroustrup B. C++ dasturlash tili. Maxsus nashr. - Moskva 2020. -1055 b.
3. Frolov A.B. Frolov G.B. C# tili bo'yicha qo'llanma. Moskva 2018.-559 b.
4. Gerbert Sh. C++ asosiy kursi. Moskva nashriyoti "Uilyams". 2020.-621 b.
5. Stenli Lippman. C++ dasturlash tili. Asosiy kurs. Uilyams - M.: 2021.
6. Nikita Kultin. Vazifalar va misollarda Microsoft Visual C++. BHV-Peterburg - Peterburg.: 2020 yil.
7. B. Stroustrup. C++ dasturlash tili. Maxsus nashr.-M.: “Binom-Press” MChJ, 2018.-1104 b.
8. Pavlovskaya T.A. C++. Yuqori darajadagi tilda dasturlash - Sankt-Peterburg: Peter. 2018.- 461 b.
9. Podbelskiy V.V. C++ tili - M.; Moliya va statistika - 2020 yil 562s.
10. Pavlovskaya T.S. Shchupak Yu.S. C/C++. Strukturaviy dasturlash. Practicum.-SPb.: Piter, 2022-240-yillar