Praktični seminarski rad u okviru kursa Verifikacija softvera *Super Mare Analysis*

Miljan Bakić

24. avgust 2024.

Sadržaj

1 Uvod

U ovom radu analizirani su rezultati korišćenja različitih alata za verifikaciju i optimizaciju softvera, uključujući `clang-tidy`, `Memcheck`, `Perf i FlameGraph`, `Flawfinder` i `Cachegrind`. Svaki alat ima specifične karakteristike koje omogućavaju detaljan uvid u kvalitet, performanse i sigurnost softvera. U daljem tekstu će biti predstavljene analize dobijene korišćenjem ovih alata na projektu `SuperMare`.

2 Clang-tidy

2.1 Opis alata

`Clang-tidy` je statički analizator koda koji pomaže u identifikaciji potencijalnih problema i stilskih nepravilnosti u C++ kodu. On omogućava analizu različitih aspekata koda, uključujući usklađenost sa standardima, performanse, bezbednost i upotrebljivost.

2.2 Analiza

Na osnovu dostavljenog izveštaja alata `clang-tidy`, može se zaključiti da postoji nekoliko ključnih oblasti u kodu koje bi trebalo unaprediti kako bi se postigla bolja usklađenost sa modernim C++ standardima i specifičnim smernicama projekta. Pre svega, primećeno je da više fajlova koristi zaštitu zaglavlja (header guards) koja ne prati preporučeni način imenovanja, što bi trebalo ispraviti kako bi se postigla konzistentnost. Pored toga, alat je ukazao na problem sa sistemskim uključivanjima (system includes) kao što su `QGraphicsTextItem` i `QGraphicsPixmapItem`, koji nisu direktno uključeni u fajlovima gde su potrebni ili nisu dozvoljeni po pravilima projekta. Takođe, identifikovani su problemi sa specijalnim funkcijama članovima (special member functions), gde više klasa definiše destruktore, ali ne i konstruktore za kopiranje ili premeštanje, niti operatore dodele, što može dovesti do neočekivanih ponašanja ili bagova. Postoji i potreba da se određene deklaracije obuhvate u `LIBC_NAMESPACE` prostoru imena (namespace), što bi trebalo proveriti i ispraviti prema zahtevima projekta. Korišćenje tzv. "magičnih brojeva" (magic numbers) u kodu, kao što su `16`, `400` i `3.0`, treba zameniti imenovanim konstantama radi bolje čitljivosti i održivosti koda. Takođe, više funkcija bi moglo koristiti povratne tipove sa zarezom (trailing return types) kako bi se poboljšala čitljivost, posebno kod kompleksnijih povratnih tipova. Prijavljeni su i problemi sa višestrukim nasleđivanjem nečistih

virtuelnih klasa (multiple inheritance), što je generalno obeshrabreno i može dovesti do komplikacija, pa bi trebalo razmotriti redizajniranje hijerarhije klasa. Pored toga, uključivanja (includes) u nekim fajlovima nisu poređana prema pravilima reda uključivanja (include order) projekta, što bi trebalo ispraviti. Takođe, alat predlaže optimizaciju performansi korišćenjem manjih tipova za `enum`-e, kao što su `GameOverOptions` i `Speed`, umesto trenutnih veličina tipa `int`. Na kraju, uočeni su i dodatni problemi, kao što su javne promenljive članice (public member variables), kratki nazivi parametara i neiskorišćena uključivanja, koje bi trebalo rešiti refaktorisanjem koda prema najboljim praksama.

3 Memcheck

3.1 Opis alata

`Memcheck` je alat koji je deo Valgrind okruženja, a koristi se za detekciju grešaka u radu sa memorijom, uključujući curenje memorije, neinicijalizovanu memoriju i nepravilne pristupe memoriji. Alat je veoma koristan za identifikaciju problema koji mogu izazvati neočekivano ponašanje aplikacije.

3.2 Analiza

Na osnovu analize dva Memcheck izveštaja generisana korišćenjem alata Valgrind, primećeni su sledeći problemi u radu programa `SuperMare`:

Prvo, oba izveštaja ukazuju na ozbiljan problem sa neinicijalizovanim bajtovima tokom sistemskog poziva `writev`. Konkretno, funkcija `__writev`, koja se nalazi u sistemskom fajlu `writev.c`, prijavljuje da vektor u pozivu sadrži neinicijalizovane bajtove, što može uzrokovati neočekivane rezultate ili greške. Ovaj problem se manifestuje u bibliotekama `libxcb.so.1.1.0` i `libQt6XcbQpa.so`, koje su deo X Window sistema za grafičke operacije, što može ukazivati na nepravilno korišćenje ovih biblioteka prilikom inicijalizacije grafičkog interfejsa ili ekrana.

Osim ovog ključnog problema, izveštaji takođe pokazuju dodatne potencijalne rizike:

- **Neinicijalizovani bajtovi:** Postoji više instanci gde se neinicijalizovani bajtovi prosleđuju funkcijama koje vrše operacije sa sistemskim resursima. Ovaj problem može dovesti do nepredvidivog ponašanja aplikacije, posebno u okruženjima sa ograničenim resursima.

- **Sistemske biblioteke:** Problemi su posebno uočeni u funkcijama kao što su `xcb_wait_for_reply` i `QXcbConnection::initializeScreensFromMonitor`, što sugeriše da bi način inicijalizacije i korišćenja ovih funkcija trebalo detaljno ispitati.

4 Perf i FlameGraph

4.1 Opis alata

Perf je alat za profilisanje performansi koji prikuplja statističke podatke o korišćenju CPU resursa od strane aplikacije. FlameGraph je vizualizacija koja prikazuje rezultate profilisanja kao trake (flames) koje predstavljaju funkcije koje troše CPU vreme. Širina trake odgovara količini procesorskog vremena koje funkcija troši.

4.2 Analiza

Perf i FlameGraph alati su korišćeni za analizu performansi aplikacije `SuperMare`. Na osnovu generisanog FlameGraph-a, slede ključne tačke:

- **Najviše vremena potrošene funkcije:** Funkcije `adler32_z` i `crc32_z`, koje su deo zlib biblioteke, zauzimaju značajan deo vremena. To sugeriše da operacije kompresije ili dekompresije podataka mogu biti usko grlo u performansama.
- **Dubina poziva i složenost:** FlameGraph pokazuje da postoji nekoliko funkcija sa dubokim lancima poziva, što može ukazivati na rekurzivne algoritme ili složene operacije koje troše puno vremena.
- **Grafičke operacije i keširanje:** Funkcije povezane sa grafikom, poput `QPainterPath::computeControlPointRect()`, troše značajan deo CPU resursa, što može biti rezultat složenih grafičkih operacija ili neefikasnog keširanja grafičkih elemenata.

5 Flawfinder

5.1 Opis alata

`Flawfinder` je alat za statičku analizu koda koji identifikuje potencijalne sigurnosne ranjivosti u C/C++ kodu. On se fokusira na prepoznavanje opasnih funkcija koje mogu dovesti do sigurnosnih problema kao što su prelivanje bafera ili upotreba neproverene memorije.

5.2 Analiza

Flawfinder je pregledao više fajlova iz projekta `SuperMare`, identifikujući sledeće ključne tačke:

- **Upotreba opasnih funkcija:** Funkcije poput `strcpy`, `sprintf`, `gets`, koje su identifikovane u izveštaju, mogu lako izazvati prelivanje bafera ako se koriste bez odgovarajućih provera. Ove funkcije su identifikovane u kritičnim fajlovima kao što su `coin.cpp`, `score.cpp`, `health.cpp`, i `main.cpp`.
- **Rangiranje ranjivosti:** Flawfinder rangira svaku pronađenu ranjivost na skali od 0 do 5, gde viši rang označava veći rizik. Funkcije sa višim rangom predstavljaju veći rizik i zahtevaju prioritetno rešavanje.

6 Cachegrind

6.1 Opis alata

Cachegrind je alat za profilisanje koji simulira ponašanje keša procesora i pomaže u identifikaciji delova koda koji uzrokuju visoku potrošnju procesorskih resursa. Ovaj alat omogućava programerima da identifikuju i optimizuju delove koda koji su najzahtevniji za procesor.

6.2 Analiza

Cachegrind je korišćen za analizu performansi programa `SuperMare`, identifikujući sledeće ključne tačke:

- **Najveći troškovi procesorskog vremena:** Glavni deo procesorskog vremena potrošen je u neidentifikovanim funkcijama, koje su označene sa `???`.
- **Funkcije visoke potrošnje:** Funkcija `adler32_z` i funkcija `__memcpy_avx_unaligned_erm` zauzimaju značajan deo CPU vremena.
- **Grafičke operacije:** Nekoliko funkcija povezanih sa QPainter i QGraphics objektima ima visoku potrošnju procesorskih resursa.
- **Operacije memorije:** Operacije alokacije i dealokacije memorije, uključujući funkcije kao što su `malloc` i `free`, takođe su identifikovane kao značajni troškovi.

7 Zaključak

Na osnovu analize rezultata dobijenih korišćenjem različitih alata, potrebno je preduzeti sledeće korake kako bi se poboljšao kvalitet, performanse i sigurnost projekta `SuperMare`:

- Projekat treba popraviti unapređenjem konzistentnosti u korišćenju zaštite zaglavlja, pravilnom upotrebom sistemskih uključivanja, kao i pravilnim definisanjem specijalnih funkcija članova.
- Potrebno je identifikovati i optimizovati nepoznate funkcije koje troše značajan deo CPU vremena, optimizovati grafičke operacije i poboljšati efikasnost upravljanja memorijom.
- Neophodno je zameniti opasne funkcije sigurnijim alternativama i sprovesti rigoroznu validaciju unosa kako bi se smanjio rizik od sigurnosnih ranjivosti.
- Potrebno je dalje optimizovati performanse pomoću alata kao što su Perf i FlameGraph, fokusirajući se na funkcije sa najvećim troškovima CPU resursa i smanjenje dubine poziva.
- Preporučuje se sprovođenje dodatnih testova sigurnosti kako bi se osiguralo da su sve potencijalne ranjivosti otklonjene i da je aplikacija bezbedna za korišćenje.