Aalto University
School of Science

Ohjelmointiharjoituksien turvallisuuden automaattinen arviointi

Ferrix Hovi

Tietotekniikan laitos
Aalto-yliopisto, Perustieteiden korkeakoulu
ferrix.hovi@ferrix.fi

10. toukokuuta 2011

Motivaatio

- Ohjelmoinnin opiskelu vaatii käytännön harjoittelua.
- Opetus tapahtuu massakursseilla ja automaattisesta arvostelusta tuleva palaute tukee oppimista paremmin kuin manuaalinen palaute.
- Automaattinen arvostelu on tuntemattoman koodin suorittamista, mikä on aina riski.
- Haittaohjelmatutkimuksessa tutkitaan binäärimuotoisia haitallisia ohjelmia.
- Mitä yhteistä on ohjelmointiharjoituksella ja haittaohjelmalla?

Tutkimuskysymykset

- Voiko staattisella analyysillä varmistua lähdekoodin turvallisuudesta?
- Voiko kotitehtävätarkistimien turvallisuutta parantaa lisäämällä staattista analyysiä?
- Mikä on kurssilla opetettavan ohjelmointikielen vaikutus riskeihin?

Automaattinen arvostelu

- Hyvin rajattuja ongelmia, jotka palauttavat aina saman tuloksen samalla syötteellä.
- Sopivat parhaiten peruskursseille, joissa sovelletaan vähemmän kuin myöhemmissä opinnoissa.
- Toimintaperiaate: Käännetään, suoritetaan ja annetaan pisteet sekä palaute opiskelijalle.
- Osa kurssinhallintajärjestelmää, jolla on omat turvallisuushaasteensa.
- Tässä kandidaatintyössä keskitytään tarkistimeen.

Tarkistimet

- Työssä tutkitaan kahta tarkistinta SCHEME-ROBOa ja Goblin-järjestelmään kuuluvaa EXPACA:a.
- Molemmat on kehitetty Teknisessä korkeakoulussa.
- Toteutukset poikkeavat merkittävästi:
  - SCHEME-ROBO on metasirkulaarinen tulkki.
  - EXPACA antaa ohjelmalle syötettä ja tarkkailee sen tulostetta ja on siten kieliriippumaton.
  - SCHEME-ROBO määrittelee vain osan Scheme-kielestä ja EXPACA on suunniteltu täysimittaisten ohjelmien testaamiseen.

Tarkistimen turvallisuusriskit

- Käytännössä ohjelmoinnin keskimääräisen peruskurssin opiskelijan taidot eivät riitä tietomurtoihin.
- Tehtävät ovat helpompia kuin järjestelmän väärinkäyttö.
- Järjestelmässä esiinnytään omilla tiedoilla.
- Todennäköisin uhka tietoturvalle on ohjelmointivirhe.

Yleisimmät turvallisuusriskit

- Puskurin ylivuodot ovat yleisimpä haavoittuvuuden syitä.
- Ohjelman suorituksen kaappaamisen jälkeen voidaan yrittää päästä pidemmälle järjestelmän muiden haavoittuvuuksien kautta.
- C- ja C++ -kielten semantiikka ja kääntäjien toteutukset tekevät muistin käsittelystä helppoa ja ennustettavaa.
- Dynaamisissa kielissä tulkki tai kääntäjä huolehtii muistista eikä sen käsittelyyn ole välttämättä toimintoja.
- Funktionaalisissa kielissä muuttujan käsite on hieman erilainen eikä ongelmaa välttämättä ole määritelty.

SCHEME-ROBO

- Turvallisuusongelmat on rajattu ansiokkaasti tarkistimen ulkopuolelle:
  - Funktionaalinen ohjelmointikieli rajaa yleisimmät virheet.
  - Kielen tyypillisimmät turvallisuusriskit on tulkissa jätetty määrittelemättä.

Koodin tutkiminen

- Suorittamatta koodia – staattinen analyysi
- Suorituksen aikana – haittaohjelmatutkimusta

Staattinen analyysi

- Yleensä monipuoliset analysaattorit tuottavat paljon hälytyksiä.
- Hälytykset ovat usein vääriä – signaali hukkuu kohinaan.
- Olennaisen etsiminen vääristä hälytyksistä ei toimi hyvin automaatiossa.
- Väärät hälytykset eivät tue myöskään oppimista.
- Esimerkiksi päättyvyysongelma tekee käytöstä vaikeaa.
- Analysaattorin saa kuluttamalla aikaa tekemällä koodin tulkinnasta vaikeasti ratkeavaa.
- Tarkistimessa tämä on kuitenkin helppoa havaita.

Sopivat analysaattorit

- Kirjallisuustutkimuksella löytyi kaksi lupaavaa analysaattoria.
- Pscan etsii `printf`-tyyppisten kutsujen muotoiluparametrien riskialtista käyttöä.
- Esimerkiksi: `sprintf(output, text)` vaarallista, `sprintf(output, "%s", text)`
- UNO etsii alustamattomien muuttujien käyttöä, nollaosoittimeen viittauksia ja puskurin rajojen ulkopuolista viittaamista.
- Eivät kata kaikkia mahdollisia ongelmia.
- Rajaavat kuitenkin merkityksellisen joukon helposti vahingossa tapahtuvia virheitä.
- Parantavat opiskelijalle tulevaa palautetta.
- ”Segmentation fault” vastaan ”Virhe x rivillä y”

Käyttö Goblinissa

- EXPACA ei tue sellaisenaan lisätyökaluja.
- Se kuitenkin tutkii ajettavan koodin konsolitulostetta ja analysaattorit ovat komentorivityökaluja.
- Jos opiskelijan ohjelma ajetaan komentojonon osana, voidaan lähdekoodi esikäsitellä välissä.
- Virhetilanteessa analysaattorin tuloste näytetään opiskelijalle sellaisenaan.
- Jos odotettua tulostetta ei tule, pisteitä tulee nolla. Hienostuneemman arvostelun toteuttaminen pitää tehdä erikseen.

- Opetettava ohjelmointikieli vaikuttaa riskeihin merkittävästi.
- Staattiset analysaattorit eivät yleensä sovi automaattiseen arvosteluun väärien hälytysten takia.
- Analyysillä voidaan kuitenkin rajata ilmiselvimpiä ongelmia pois.
- Tärkein etu on opiskelijalle tuleva täsmällisempi palaute.
- Turvallisuusongelmat ratkeavat paremmin eristämällä järjestelmä hyvin tai toteuttamalla rajoitettuja kieliä.