Оцінка правила стосовного того, наскільки воно гарно підходить для характеристики клієнтів, виконує підсистема визначення характеристик бази правил. Основними її характеристиками є:

- пристосованість правила;
- сумарна пристосованість набору правил;
- оцінка достовірності правила;
- оцінка сумісності тестового шаблона для кожного правила;
- сумарна оцінка сумісності по кожному правилу.

Вхідними даними для підсистеми є набір правил, який потрібно охарактеризувати.

Вихідними даними є оцінена характеристика правила, яка найчастіше представляється дійсним числом (для одного правила) або масивом дійсних чисел (для набору правил).

Роботу системи було випробувано на тестовій вибірці, взятої з UCI Machine Learning Repository, CA: University of California [4]. В результаті перевірки роботи системи середня частка правильно класифікованих клієнтів є рівною 90.1 %.

Список використаних джерел:

1. Klir G. J. Fuzzy Sets and Fuzzy Logic. Theory and Applications / G. J. Klir, B. Yuan. – Prentice Hall, 1995. – 574 p.

2. Ishibuchi H. Comparison of the Michigan and Pittsburgh Approaches of the Design of the Fuzzy Classification Systems / H. Ishibuchi, T. Nakashima, T. Murata // Electronics and Communications in Japan (Part III: Fundamental Electronic Science). – 1997. – Vol. 80, Issue 12. – P. 10–19.

3. Ishibuchi H. Performance Evaluation of Fuzzy Classifier Systems of Multidimensional Pattern Classification Problem / H. Ishibuchi, T. Nakashima, T. Murata // IEEE Transactions on Systems, Man, and Cybernetics. – 1999. – C. 601–608.

4. Credit Approval Data Set [Електронний ресурс]. – Режим доступу: https://archive.ics.uci.edu/ml/datasets/Credit+Approval.

Пивоваров А.С.

студент,

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

МЕХАНІЗМИ ЗАХИСТУ ПРОГРАМНОГО КОДУ В IDE MICROSOFT VISUAL STUDIO

IDE Microsoft Visual Studio забезпечує надійні та безпечні обчислення (trustworthy computing), дозволяє проектувати, реалізовувати і тестувати реалізацію підсистеми безпеки на кожному етапі життєвого циклу програми з підтримкою принципів розробки безпечного коду [1].

При розробці прикладних програм створюється власний код та повторно використовується існуючий код. Для безпечного виконання коду, отриманого із невідомого джерела і захисту «довіреного» коду від навмисного або випадкового пошкодження, середовище Visual Studio на базі технології.NET Security Framework можна використати вбудовану систему безпеки з назвою управління доступом до коду CAS (Code Access Security) [2].

CAS – це система безпеки, яка дозволяє адміністраторам і розробникам виконувати авторизацію прикладних програм так же, як виконується авторизація користувачів. За допомогою CAS можна створювати правила використання для прикладних програм, встановлювати різні ступені довіри до модулів прикладної програми. Метод захисту на основі CAS може застосовуватися тільки для керованого коду (managed code) платформи.NET [3], тому що базується на можливостях загальномовного середовища виконання (Common Language Runtime, CLR). Для коду середовища CLR вводиться поняття зборки (assembly). Зборка – це готова до виконання прикладна програма, яка містить код та метадані з описом зборки, які розташовані у маніфесті зборки (assembly manifest). Маніфест зборки дозволяє ідентифікувати зборку, описує файли, які включені до реалізації зборки, описує типи та ресурси, які використовуються, відображає залежність від інших зборок, описує набори прав доступу, які необхідні зборці для коректної роботи. Ця інформація використовується під час виконання для визначення посилань, перевірки коректності версій та цілісності завантажених зборок. Кожна зборка має унікальне ім’я.

Метод захисту CAS концептуально базується на двох головних блоках компоновки: наборі дозволів та групі коду. Набір дозволів являє собою колекцію окремих дозволів на доступ до різних ресурсів комп’ютера. Група коду є засобом авторизації, який зв’язує зборки з набором дозволів.

Система безпеки CAS дозволяє обмежувати доступ коду до захищених ресурсів та операцій, визначати дозволи та набори дозволів для надання прав доступу до системних ресурсів, налаштовувати політику безпеки.

Для реалізації концепції CAS платформа.NET Security Framework пропонує великий набір класів, визначений у просторі імен System::Security. Клас SecurityManager використовується для задавання відповідності між свідоцтвами групи і дозволами, що визначає реальну політику безпеки. Для ідентифікації зборок використовується клас Evidence, який інкапсулює інформацію про свідоцтва для визначення характеристик зборки. Цей клас є визначеним у просторі імен System::Security::Policy. Пропонується множина похідних класів, які є похідними класу CodeAccessPermission і використовуються для перевірки дозволів зборок. Бібліотека криптографічних класів забезпечує доступ до більшості криптографічних алгоритмів: DSA, RSA, DES, RC2, KeyedHashAlgorithm, MD5, SHA, SHA256, SHA384, SHA512. Криптографічні класи реалізовані в просторі імен System::Security::Cryptography.

Для визначення набору дозволів та груп коду можна використовувати утиліту.NET Framework Configuration.

Утиліта.NET Framework Configuration дозволяє за замовчуванням налаштовувати 19 дозволів CAS: Directory Services, DNS, Environment Variables, Event Log, File Dialog, File IO, Isolated Storage File, Message Queue, Ole Db, Performance Counter, Printing, Reflection, Registry, Security, Service Controller, Socket Access, SQL Client, User Interface, Web Access. Ці дозволи забезпечують перегляд визначених областей служб каталогів, дозвіл/заборону доступу до DNS, змінних середовища, принтерів, контроль перезапису файлів та папок, дозвіл/заборону доступу до діалогових вікон, виконання аудиту журналів подій тощо. Утиліта.NET Framework Configuration дозволяє користувачу створювати власні набори дозволів а також пропонує 7 попередньо заданих наборів дозволу: FullTrust – дає необмежений доступ до всіх ресурсів з можливістю пропуску верифікації; SkipVerification – пропускає початковий процес верифікації, перевірки цифрового підпису; Execution – надає право на виконання коду; Nothing – відміняє всі привілеї в тому числі виконання; LocalInternet – містить права для локальних мережевих програм; Internet – містить права для Інтернет-програм; Everything – дає необмежений доступ до ресурсів без можливості пропуску етапу верифікації.

В середовищі Visual Studio окрім зазначених механізмів захисту пропонується використовувати засоби підтримки аспектно-орієнтованого програмування (АОП). АОП є однією із концепцій програмування, яка є розвитком процедурного та об’єктно-орієнтовного програмування. Пакети AspectJ, AspectC++, Aspect.NET [4] є простим та практичним розширенням відповідно мов Java, C++, C++/CLI, C#, які дозволяють додавати можливості АОП в рамках об’єктно-орієнтованих мов. Пакети AspectC++, Aspect.NET можуть вбудовуватися в систему розробки Visual Studio.

Використання АОП в середовищі Visual Studio дозволяє реалізувати підсистему безпеки у вигляді окремих компонентів і повторно використовувати їх в різних додатках. Програмну систему можна розглядати як сукупність різних компонентів. Кожен компонент відповідає за визначену функціональність. Можна виділити певні частини, або аспекти, що відповідають за функціональність, реалізація якої розосереджена в коді програми і складається із схожих фрагментів коду. Таку функціональність називають наскрізною функціональністю (cross – cutting concerns) [5]. Засоби АОП дозволяють виділити і реалізувати наскрізну функціональність в окремих модулях, які називаються аспектами. Аспекти вмонтовуються в точки приєднання цільової програмної системи за допомогою компонувальника аспектів (weaver) з використанням правил впровадження. Прикладами наскрізної функціональності є протоколювання (logging), безпека виконання програми у багатопотоковому обчислювальному середовищі (MT – safety), обробка помилок, реалізація підсистеми безпеки (security). Пакет АОП AspectC++ може використовуватися для безшовної інтеграції наскрізної функціональності у додатки і здійснює впровадження аспектів в цільову зборку на рівні бінарного коду без модифікації початкового коду цільової програмної системи.

Aspect є основною одиницею модульності АОП. В аспектах задаються зрізи точок виконання (Pointcut) та інструкції, які виконуються в точках виконання (Advice). JoinPoint – строго визначена точка виконання програми, яка пов’язана з контекстом виконання, наприклад, викликом функції, конструктора, обробника виключень. Pointcut – набір (зріз) точок JoinPoint, які задовольняють заданій умові. Advice – набір інструкцій мови C++, який виконується до, після або замість кожної із точок виконання (JoinPoint) заданого зрізу (Pointcut). Introduction – спроможність аспекту змінювати структуру або ієрархію класу. Pointcut і Advice визначають правила інтеграції.

У роботі пропонується реалізація авторизованого доступу до серверу даних. Ідентифікація принципала є однією із головних задач засобів захисту даних. Для прийняття рішення на доступ до певних елементів система повинна обов'язково встановити особу принципала. Функціональність авторизації і аутентифікації має бути вбудована у функції, які потребують захисту, що однозначно приведе до перемішування вимог у коді і втрати модульності компонентів бізнес-логіки.

При використанні пакету АОП на етапі проектування системи подібну наскрізну функціональність потрібно винести в аспектний модуль. Наприклад, в абстрактному аспекті AbstractAuthenticAspect визначається логіка аутентифікації та авторизації. Визначається набір інструкцій мови для ідентифікації before():authOperations(), який вставляється до коду, який має бути захищеним, і виконує закриту функцію заданого аспекту – authenticate(). В аспекті визначається набір інструкцій авторизації Object around(): authOperations(), який викликається замість коду, який захищається. Код який захи- щається виконується у випадку успішного виконання коду служби авторизації. В абстрактному аспекті AbstractAuthenticAspect також визначено принципал authenticatedSubject, для якого в конкретному аспекті визначені права за допомогою виклику заміщеної функції getPermission(). У конкретному аспекті визначається набір точок інтеграції аспектного коду authOperations. Такими точками є головні функції для виконання операцій з даними. У всі ці функції інтегрується функціональність, яка визначена в аспектному модулі. При цьому самі компоненти будуть містити тільки базовий код без наскрізної функціональності, що значно покращує модульність системи [5].

Платформа.NET Security Framework підтримує дві концепції безпеки створення надійних програм: безпека на базі ідентифікації користувача та його «ролі», безпека на базі контролю доступу до коду CAS. На відміну від традиційного підходу, CAS дозволяє контролювати виконання дій, виходячи із ідентифікації власне коду, а не ідентифікації користувача, який цей код виконує. CAS дозволяє вирішувати, які дії дозволено виконувати коду програми.

Список використаних джерел:

1. Сафонов В. О. Современные технологии разработки надежных и безопасных программ (Trustworthy Computing). // Компьютерные инструменты в образовании. – 2010. – JN2 6. – С. 25-33.

2. Торстейнсон П. Криптография и безопасность в технологии.NET/ П.Торстейн- сон Г.А. Ганеш; пер. с англ. – М.: БИНОМ. Лаборатория знаний, 2007. – 479 с.

3. Просиз Дж. Программирование для Microsoft.NET/ Дж. Просиз; пер. с англ. – М.: Издательско-торговий дом «Русская редакция», 2009. – 704 с.

4. AspectC++. [Електроний ресурс] // Режим доступу: http: //www.aspectc.org.

5. Нгуен Ван Доан. Средства аспектно-ориентированного программирования для разработки Web-приложений в системе Aspect.NET/ Нгуен Ван Доан, В. О. Сафонов // Вестн. С.-Петерб. ун-та. Сер. 10. –2011. – Вып. 1. – С. 85-105.

Сімоненко В.П.

доктор технічних наук, професор;

Баль В.В.

магістр,

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

ОЦІНКА УЗГОДЖЕНОСТІ ІНФОРМАЦІЇ В GRID СИСТЕМАХ

Інформаційна система Grid вирішує питання, які можуть зажадати розгляду всіх джерел інформації (Grid-сервісів), які широко поширені географічно, з тим щоб забезпечити ефективні функції Grid, які можуть використовувати кілька взаємодіючих сервісів. В основному це може бути досягнуто шляхом переміщення запиту до даних (відправлення запиту) або переміщення даних в запит (відправка даних).

Фундаментальний аспект розподілених обчислень є необхідність отримання інформації про структуру і стан мережевих послуг в Грід системах, які широко поширені географічно. Інформація, що описує кожну мережеву послугу Грід системи забезпечується само мережевою службою і, отже є основним джерелом інформації. Інформація відповідає інформаційній моделі яка представляє основні концепції служби Грід системи і відносин між ними, щоб чітко визначити їх семантику. Передбачається, що зміст джерела інформації є актуальним, тобто значення відображають реальний стан Грід системи.

Необхідно дозволити запити, які можуть враховувати деякі або всі джерела інформації, щоб забезпечити ефективні функції Грід системи, які можуть використовувати кілька взаємодіючих служб. Загалом, інформація в Грід системах надзвичайно розподілена, тобто кожен кортеж (екземпляр об'єкта) знаходиться в іншому місці [8]. Запити не є складними з точки зору структури, не більше одного предиката і немає об'єднань, тому результат зазвичай є підмножиною загальної інформації (наприклад, знайти найближчий обчислювальний сервіс для служби зберігання). Метою є мінімізація часу відповіді на запит для виконання багатьох запитів від багатьох клієнтів для багатьох джерел інформації (тобто для вирішення проблеми масштабованості).