УДК 651.34 DOI: 10.31673/2412-4338-2018-0-2-81-90

Козелков С. В. *Державний університет телекомунікацій, Київ*
Лисиця Д. О., Семенов С. Г., Лисиця А. О. *НТУ «Харківський політехнічний інститут»*

ІМІТАЦІЙНА МОДЕЛЬ ПРОЦЕСУ ГЕНЕРАЦІЇ ТА РЕАЛІЗАЦІЇ ЗАСОБІВ «ТЕСТУ НА ПРОНИКНЕННЯ»

В результаті моделювання перевірена гіпотеза на можливість використання критерію Хі-квадрат Пірсона та проведені дослідження достовірності результатів математичної формалізації. Розглянуті основні шаблони безпечних програм, що можуть використовувати розроблений комплекс математичних моделей. Дано практичні рекомендації щодо використання методів і засобів тестування безпеки програмного забезпечення однорангової комп'ютерної системи.

Ключові слова: *імітаційне моделювання, безпека програмного забезпечення, критерій Хі-квадрат Пірсона, тест на проникнення.*

Kozelkov S. V. *State University of Telecommunications, Kyiv*
Lysytsia D. O., Semenov S. H., Lysytsia A. O. *NTU "Kharkiv Polytechnic Institute"*

SIMULATION MODEL OF THE GENERATION AND IMPLEMENTATION OF THE MEANS OF «PENETRATION TESTING»

During simulation of the process generation of cyberattacks of various formats was performed using the specialized IxChariot 9 package and software developed during the research. The collection of incoming information about the download of the network device was carried out using the standard software traffic analyzer ("Wireshark"). As a result of the simulation, the hypothesis has been tested for the possibility of using the Pearson Xi-square criterion and studies of the reliability of the results of mathematical formalization have been carried out. It is shown that the use of only the principles of secure architectures in the development of secure software is not enough. Must use templates. The main templates of safe programs are considered in which the developed complex of mathematical models of technologies of generation and realization of "penetration test" methods and the method of allocation of the algorithm from binary code for software security analysis can be used. Two main variants of use and improvement of the developed model are considered. The first option has a single centralized security element, which becomes the middle of the queries for all resources in the system. The second option shares the security element's responsibilities, so that there is a separate instance of the security element for each individual resource type.

The security of software in the modern world is becoming more and more difficult and difficult, since the value of information resources is constantly increasing. Ensuring the quality of software is a complex matter. The changing one of the characteristics of a computer system or software can easily lead to changes or create a need for change in other parts of the system. Practical recommendations on the use of methods and tools for testing software security of the temporary computer system are given.

Keywords: *simulation, software security, Pearson Xi-square criterion, penetration test.*

Козелков С. В. *Государственный университет телекоммуникаций, Киев*
Лисица Д. А., Семенов С. Г., Лисица А. А. *НТУ «Харьковский политехнический институт»*

ИМИТАЦИОННАЯ МОДЕЛЬ ПРОЦЕССА ГЕНЕРАЦИИ И РЕАЛИЗАЦИИ СРЕДСТВ «ИСПЫТАНИЕ НА ПРОНИКНОВЕНИЕ»

В результате моделирования проверена гипотеза на возможность использования критерия Хи-квадрат Пирсона и проведены исследования достоверности результатов математической формализации. Рассмотрены основные шаблоны безопасных программ, которые могут использовать

разработанный комплекс математических моделей. Даны практические рекомендации по использованию методов и средств тестирования безопасности программного обеспечения одноранговой компьютерной системы.

Ключевые слова: *имитационное моделирование, безопасность программного обеспечения, критерий Хі-квадрат Пирсона, тест на проникновение.*

Вступ. Постановка задачі

Безпека програмного забезпечення в сучасному світі стає все більш складною і важкою справою, оскільки цінність інформаційних ресурсів постійно збільшується. В той же час, забезпечення якості програмного забезпечення – це комплексна справа. Стає очевидним, що зміна однієї з характеристик комп'ютерної системи або програмного забезпечення може легко привести до змін або створити потребу в зміні в інших частинах системи. У зв'язку з цим одним з найбільш важливих і корисних знарядь аналізу складних процесів і систем стало імітаційне моделювання.

Аналіз літературних джерел [1-3] показав, що одним з ключових етапів наукового дослідження є синтез отриманих результатів в єдину методологічну систему практичного застосування технологій та розробка імітаційної моделі даної системи. При цьому їх метою має бути рішення наступних часткових завдань:

– перевірка адекватності розроблених моделей і методів;

– аналіз достовірності отриманих результатів в ході вирішення поставлених оптимізаційних задач;

– обгрунтований вибір показників, коефіцієнтів і параметрів функціонування системи;

– розробка науково-практичних рекомендацій щодо використання моделей і методів.

В роботах [4, 5] наведені матеріали математичного моделювання технологій генерації та реалізації засобів «тесту на проникнення», наукових розробок методу виділення алгоритму з двійкового коду для аналізу безпеки програмного забезпечення.

Метою та задачами дослідження є розробка імітаційної моделі процесу генерації та реалізації засобів «тесту на проникнення», а також обгрунтування практичних рекомендацій з використання методів та засобів тестування безпеки програмного забезпечення одноранговой комп'ютерної системи.

Для обґрунтування достовірності отриманих результатів математичного моделювання і оцінки ефективності розроблених методів було проведено імітаційне моделювання. Як інструментарій імітаційного моделювання використано середовище символьної математики MathCAD-14, спеціалізовані програми, що виконують функції як засобів захисту, так і засобів генерації кібератак різного формату (IxChariot 9, WireShark) [1].

Імітаційна модель тестування безпеки програмного забезпечення

Узагальнена структурна схема розробленої імітаційної моделі технології тестування безпеки програмного забезпечення представлена на рис. 1.

У процесі формування вимог до інформаційної безпеки програмного забезпечення та допустимих ймовірносно-часових показників його тестування емулювалися і розглядалися різні характеристики комп'ютерних систем, типи програмного забезпечення, різні способи архітектурної побудови комп'ютерних мереж. Зокрема розглядалися такі варіанти:

– різнотипність операційних систем;

– рангові системи;

– різновиди джерел вихідного коду та ін.

Генерація кібератак різного формату виконувалася з використанням спеціалізованого пакету IxChariot 9 і програмного забезпечення, розробленого в ході досліджень методу виділення алгоритму з двійкового коду для аналізу безпеки програмного забезпечення.

Збір вхідної інформації про завантаження мережевого пристрою здійснювався за допомогою стандартного програмного аналізатора трафіку («Wireshark»).

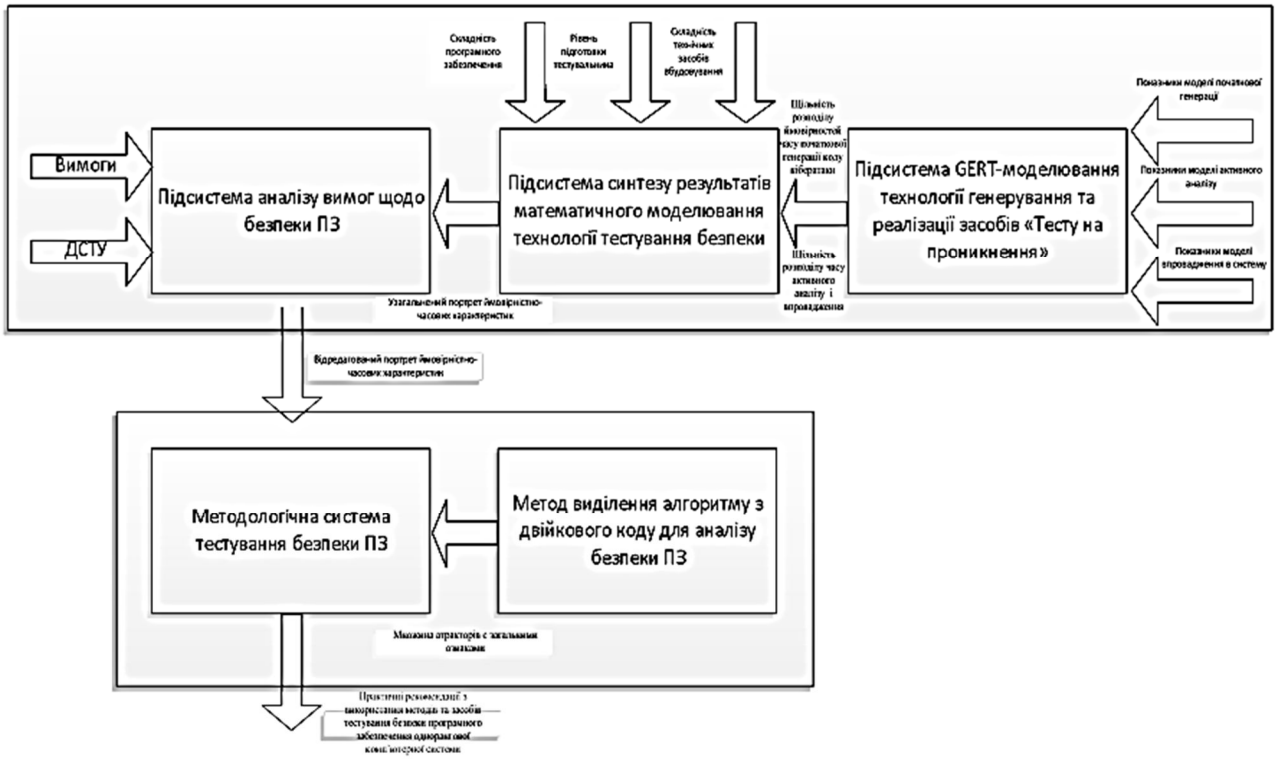

Рис. 1. Узагальнена структурна схема імітаційної моделі технології тестування безпеки програмного забезпечення

Обґрунтування достовірності результатів математичного моделювання

Для обґрунтування достовірності отриманих в [4, 5] результатів проведено ряд експериментів, відповідно до наступних умов:

- основна методологія управління розробкою програмного забезпечення є SCRUM;
- для тестування безпеки виділено окремих фахівців;
- число експериментів N^*=100.

За результатами експерименту отримана гістограма часу генерації та реалізації засобів «тесту на проникнення», яка представлена на рис. 2.

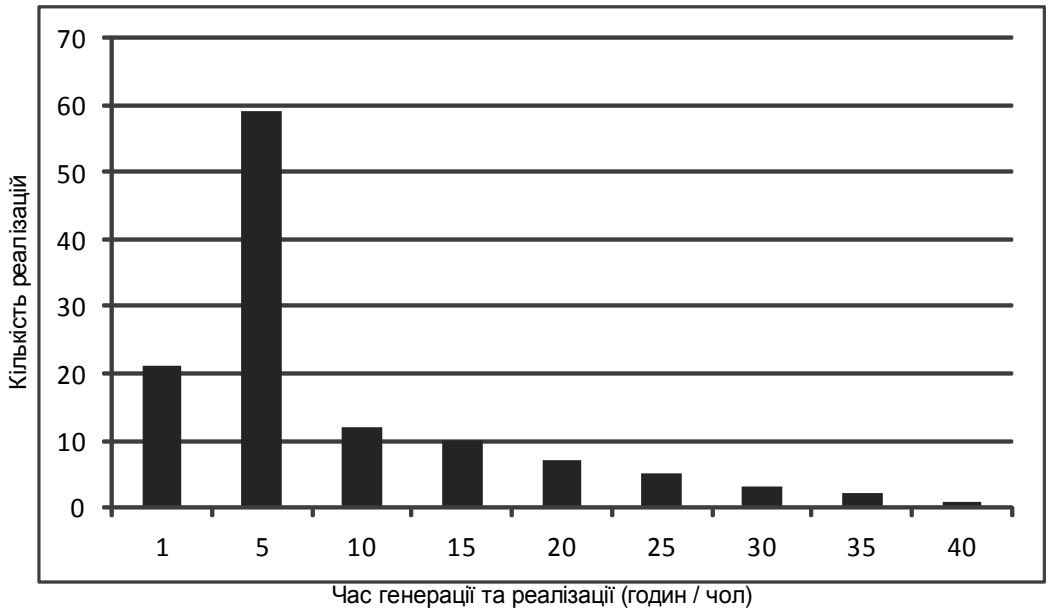

Рис. 2. Гістограма часу генерації та реалізації засобів «тесту на проникнення»

Висунута гіпотеза про нормальний розподіл цієї випадкової величини була перевірена за критерієм згоди χ^2 Пірсона [6]

$$\chi^2 = N^* \sum_{i=1}^{k} (P_i^* - P_i)^2 / P_i ,$$

де k – кількість розрядів (інтервалів) статистичного ряду;

P_i^* та P_i – статистична та теоретична ймовірності "попадання" заданого показника до *i*-го розряду.

Проведена перевірка довела правдоподібність гіпотези про те, що величина часу генерації та реалізації засобів «тесту на проникнення» розподілена за нормальним законом.

Отримано оцінки $\hat{t}_{\text{тест}}^{(i)}$ – математичного сподівання та $\hat{D}_{t_{\text{тест}}^{(i)}}$ – дисперсії ($\hat{\sigma}_{t_{\text{тест}}^{(i)}}$ середньоквадратичного відхилення) випадкової величини $t_{\text{тест}}^{(i)}$ часу тестування ПЗ [5, 6]:

$$\hat{t}_{\text{тест}}^{(i)} = \frac{\sum_{i=1}^{k} \hat{t}_{\text{тест}}^{(i)}}{\text{N}^*}; \quad \hat{D}_{\hat{t}_{\text{тест}}^{(i)}} = \frac{\sum_{i=1}^{k} \left(\hat{t}_{\text{тест}}^{(i)} - t_{\text{тест}}^{(i)} \right)^2}{N^*\text{-}1}; \quad \hat{\sigma}_{\hat{t}_{\text{тест}}^{(i)}} = \sqrt{\hat{D}_{\hat{t}_{\text{тест}}^{(i)}}} .$$

Скориставшись відомим виразом для розрахунку довірчої ймовірності відхилення відносної частоти від постійної ймовірності в незалежних випробуваннях [5, 6] визначимо довірчу ймовірність того, що отримане в результаті експерименту значення часу генерації та реалізації засобів «тесту на проникнення» "не відхилиться" від математичного сподівання $\hat{t}_{\text{тест}}^{(i)}$ більш ніж на 1:

$$P\left(\left| \hat{t}_{\text{тест}}^{(i)} - t_{\text{тест}}^{(i)} \right| < 1 \right) = 2\Phi(1 / \hat{t}_{\text{тест}}^{(i)}),$$

де Ф – функція Лапласа виду $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-t^2/2} dt$ [4, 5].

Результати проведених експериментів показали, що для всіх досліджуваних видів даних довірча ймовірність того, що значення статистичної величини $t_{\text{тест}}^{(i)}$ не відхиляється від математичного сподівання $\hat{t}_{\text{тест}}^{(i)}$ більш ніж на 1 дорівнює: $P \approx 0{,}94$.

За даними, отриманими в [4-6], в умовах, зазначених вище, проведено порівняльне дослідження результатів математичного моделювання і експерименту. Результати порівняння представлені на рис. 3 у вигляді графіка щільності розподілу ймовірностей часу $t_{\text{тест}}$ генерації та реалізації засобів «тесту на проникнення» та відповідних їм меж довірчого інтервалу:

$$I_\beta = \left[\hat{\bar{J}} - \varepsilon_\beta, \hat{\bar{J}} + \varepsilon_\beta \right],$$

в якій дійсне значення $\bar{J}$ потрапляє з довірчою ймовірністю $\beta = 0{,}94$ та оцінок його $\hat{t}_{\text{тест}}^{(i)}$ математичного сподівання.

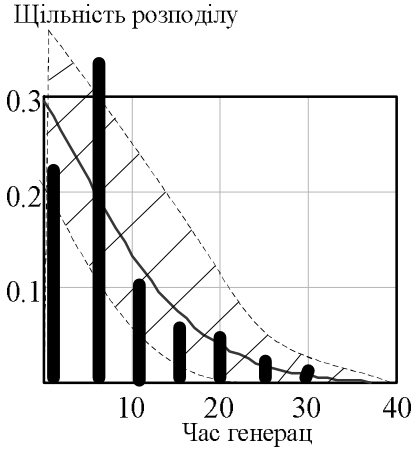

Рис. 3. Розподіл ймовірностей часу генерації та реалізації засобів «тесту на проникнення»

З графіків бачимо, що в ключовій тестовій ситуації (час $t_{\text{тест}} \approx 5$ год/люд) *розрахункова* крива J (суцільна крива), отримана відповідно до розробленої в роботі математичної моделі, в більшості практичних випадків потрапляє в *усереднени*й довірчий інтервал (заштрихована область).

Обґрунтування практичних рекомендацій з використання методів та засобів тестування безпеки програмного забезпечення однорангової комп'ютерної системи

При розробці програмного забезпечення необхідно враховувати, що виконання будь-якого з вимог показників якості може впливати на інші вимоги. І в такому випадку необхідно проаналізувати співвідношення вигод і втрат для сукупності багатьох показників якості [5, 6]. Серед інших показників якості найбільш вагомими є показники безпеки.

Як зазначено в [1] безпека – це здатність системи запобігати зловмисним або випадковим діям, що не передбачені при проектуванні, або не допускати розголошення або втрату даних. Тут ведеться мова про безпеку системи в загальному сенсі. У той же час підвищення безпеки програмного забезпечення веде також до підвищення надійності системи в цілому за рахунок зниження ймовірності успіху атак і їх негативного впливу на роботу системи.

Проведений аналіз літератури [1, 2, 7] і досвід програмування [2, 8] дозволили виділити ряд особливостей створення безпечних програмних додатків. Серед них можливо виділити:

- безпечне програмування;
- проектування безпеки.

Основні завдання безпечного програмування освітлені у ряді інтернет-ресурсів [1, 2, 7] та виданнях зарубіжних авторів [8]. Серед основних принципів безпечного програмування слід виділити:

- використання безпечних бібліотек;
- виконання самоперевірки;
- використання утиліт перевірки коду;
- тестування безпеки;
- виконання code review та ін.

Проектування безпеки програмного забезпечення, як відмічено у ряді джерел [1, 2, 7], повинно базуватись на таких основних принципах безпечних архітектур:

- простота механізмів (economy of mechanism);
- безпека за замовчуванням (fail-safe defaults);
- повне проникнення захисту (complete mediation);
- відкритий дизайн (open design);
- поділ повноважень (separation of privilege);
- мінімум привілеїв (least privilege);
- мінімізація поділу ресурсів (least common mechanism);
- психологічна прийнятність (psychological acceptability).

Але, як показали дослідження, використання лише принципів безпечних архітектур при розробці безпечного програмного забезпечення недостатньо. Необхідно використання шаблонів. Розглянемо основні шаблони безпечних програм, в яких можуть використовуватись розроблені комплекс математичних моделей технологій генерації и реалізації засобів «тесту на проникнення» та метод виділення алгоритму з двійкового коду для аналізу безпеки програмного забезпечення.

Використання моделі «тесту на проникнення» в шаблоні «Захищена система»

Основною рекомендацією при створенні шаблону «Захищена система» є структурування системи таким чином, щоб весь доступ клієнтів до ресурсів був опосередкований додатковим елементом захисту «security», який би застосовував політику безпеки.

Щоб переконатись, що всі запити доступу оцінюються відповідно до політики безпеки системи, має існувати механізм виконання цієї політики, що має такі властивості:

– механізм повинен бути викликаний при кожному запиті на доступ;
– механізм не повинен бути обхідним;
– механізм повинен правильно оцінити політику;
– функціонування механізму повинно бути надійним;
– попередні чотири властивості повинні бути перевірені «тестом на проникнення» до певного рівня довіри.

Слід зауважити, що модель «тесту на проникнення» повинна складатися з трьох елементів:

– "зовнішній", з якого походять всі запити доступу;
– "всередині", в якому розташовані всі ресурси;
– ефективно відтворена внутрішня система, що може тестуватися та забезпечувати виконання політики щодо всіх запитів за межами ресурсів та усередині.

Проведені дослідження показали, що існують два основних варіанти використання та вдосконалення розробленої моделі. Перший варіант має єдину централізований елемент безпеки, який стає посередині запитів на всі ресурси в системі.

Другий варіант розподіляє обов'язки елемента безпеки, так, що існує окремий примірник елемента безпеки для кожного окремого типу ресурсу.

Під час тестування запропонованої схеми необхідно враховувати декілька особливостей архітектури. Перш за все необхідно перевірити ізоляцію елементів захисту та їх непересічність. Багато конструкцій мікропроцесорів не підтримують повну ізоляцію адресного простору між програмами, що працюють у "стані системи", та програмами, що працюють у "користувацькому стані". Важко або неможливо розробляти ядра операційної системи, які не можна обійти на таких мікропроцесорах.

Віртуальні архітектури машин також страждають від невдач адресної ізоляції: наприклад, кілька версій віртуальної машини Java, коли загальні статичні змінні між потоками адресних просторів порушують властивість ізоляції потоку, передбачені моделлю безпеки Java.

При тестуванні також необхідно враховувати, що багато систем (включаючи багато Інтернет-серверів) є вразливими до атак переповнення буфера. Наслідки переповнення буфера є результатом нездатності елементів безпеки перевірити розмір вхідних параметрів, наданих клієнтом. Атака переповнення буфера змушує елементи безпеки виконувати шкідливий код, наданий клієнтом.

Деякі системи вразливі до атак отруєння даними; атаки на отруєння даних, які виникають у розробників елементів безпеки, які не можуть визначити відповіді на помилки для всіх можливих недійсних значень вхідних даних. Нападки на отруєння даних використовують непередбачувану реакцію елементів безпеки на "неправильну" вхідну цінність.

Шаблон «Захищена система» сприяє запевненню шляхом мінімізації кількості коду, який повинен бути забезпечений, і шляхом модуляції його до елементу безпеки.

Використання шаблону «Політика безпеки» при тестуванні на проникнення

Використання шаблону «Політика безпеки» доцільно коли бажано відокремити реалізацію політики безпеки від реалізації менеджеру ресурсів, необхідно зниження числа запитів політики щодо захисту до мінімального числа простих модулів, щоб спростити перевірку безпеки.

Загальний вид діаграми шаблону «Політика безпеки» при тестуванні на проникнення наведено на рис. 4. При цьому слід враховувати, що шаблон «Політика безпеки»:

– знижує зв'язок між впровадженням політики та впровадженням ресурсів. Це дозволяє керувати менеджерами ресурсів без розуміння автентифікації, атрибутів «збір», «оцінка політики» та «впровадження політики»;
– забезпечує єдину точку контролю та управління для діяльності, пов'язаної з політикою;

– локалізує операції, пов'язані із політикою, покращує надійність системи шляхом обмеження величини системного коду, яка повинна бути перевірена під час проведення тестування на проникнення та ін.

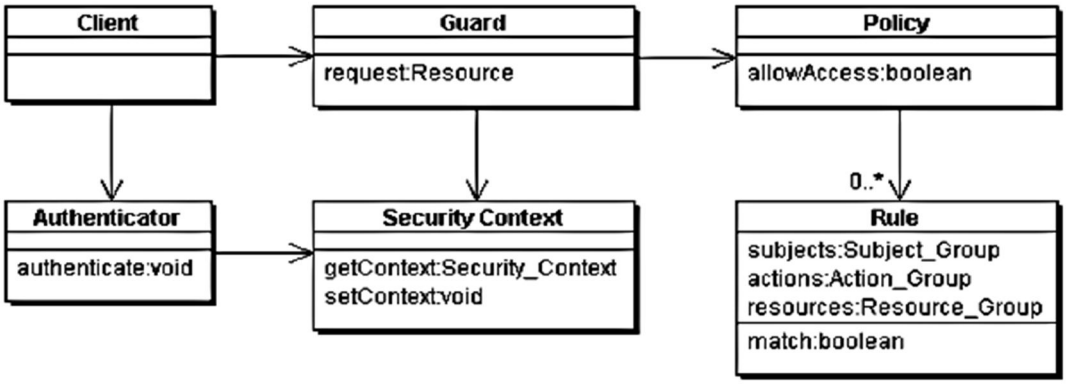

Рис. 4. Діаграми шаблону «Політика безпеки» при тестуванні на проникнення

Також може бути корисно включити окремий розширюваний шаблон у клас «Authenticator», згаданий у наведеній вище політиці. Цей розділ зарезервований як заповнювач.

Використання шаблону «Суб'єктивний дескриптор»

Існує багато артефактів, пов'язаних із безпекою, які можуть бути пов'язані з предметом; тобто суб'єкт (людина або програма). Артефакти можуть включати властивості та твердження про предмет, а також майно, пов'язане з безпекою, такими як ключі шифрування. Контроль доступу суб'єкта до різних ресурсів може залежати від різних артефактів суб'єкта. Деякі артефакти можуть самостійно містити конфіденційну інформацію, що потребує контрольованого доступу.

Тематичний дескриптор забезпечує доступ до предметних артефактів і полегшує управління та захист їх, а також забезпечує зручну абстракцію для передачі артефактів між підсистемами. Наприклад, підсистема автентифікації може встановлювати суб'єктні артефакти, що включають підтвердження ідентичності користувача, яке потім може бути використано та використано окремою підсистемою авторизації.

Рекомендується використання шаблону дескриптора теми, коли:

– підсистема, відповідальна за перевірку артефактів суб'єкта (наприклад, прав або облікових даних), не залежить від підсистеми, яка встановлює їх;

– кілька підсистем встановлюють артефакти, що застосовуються до одного і того ж предмету;

– різні типи або набори предметних артефактів можуть бути використані в різних контекстах;

– необхідний вибірковий контроль доступу до певних предметних артефактів.

– кілька предметних ідентифікацій потрібно управляти в рамках однієї операції.

Загальний вид діаграми шаблону «Суб'єктивний дескриптор» наведено на рис. 5, з якого слідує, що використання шаблону дескриптора теми виконує наступні функції:

1. Інкапсулює артефакти суб'єктів. Теговий дескриптор дозволяє копію артефактів обробляти як єдиний об'єкт. Нові типи артефактів можуть бути додані без зміни дескриптора предмету або коду, який використовує його.

2. Забезпечує контроль доступу. Тематичний дескриптор дозволяє створювати списки атрибутів, включаючи функцію контролю доступу, щоб гарантувати, що неавторизовані виклики не матимуть доступу до конфіденційних атрибутів (наприклад, токенів автентифікації).

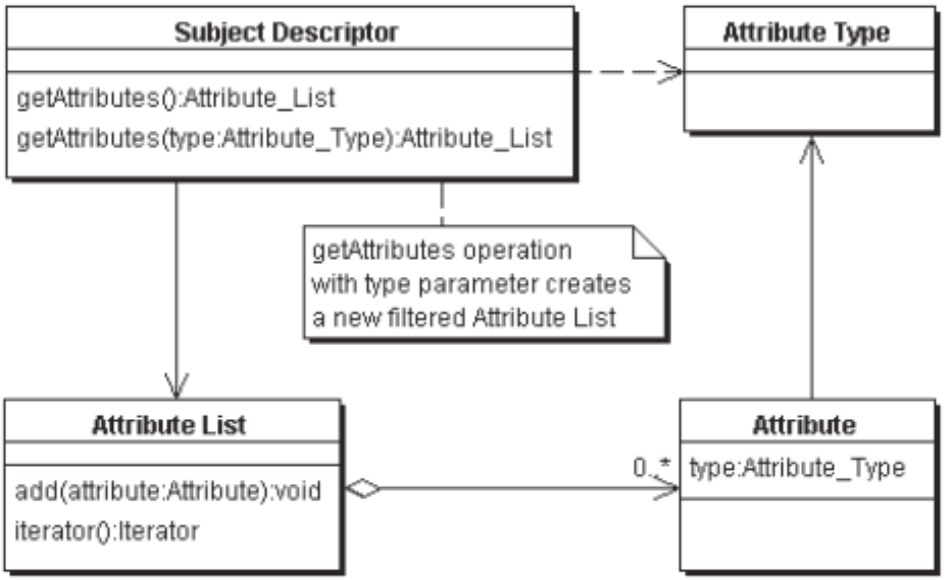

Рис. 5. Загальний вид діаграми шаблону «Суб'єктивний дескриптор»

Виділення алгоритму з двійкового коду на основі шаблону «Захищений вміст»

Досвід використання сучасних засобів реверсної інженерії, а також проведені дослідження показали, що якщо для окремого контексту виконання, програми або процесу потрібно діяти від імені декількох предметів, то предмети повинні бути диференційовані один від одного, а інформація про кожен предмет має бути доступною для використання. Коли контекст виконання, програма або процес повинен діяти від імені одного суб'єкта декілька разів протягом певного періоду часу, він повинен мати доступ до інформації про предмет, коли це необхідно зробити. Шаблон контексту безпеки забезпечує доступ до інформації суб'єкта у цих випадках.

Як практичні рекомендації застосування шаблону «Захищений вміст» можливо відмітити його доцільність коли:

– контекст процесу або виконання є активним від імені окремого суб'єкта протягом певного періоду часу, проте він повинен встановити безпечні зв'язки з різними партнерами від імені цього окремого питання;

– контекст процесу або виконання може діяти від імені різних предметів і потребує управління тим, який предмет в даний час активний.

Загальний вид діаграми шаблону «Захищений вміст» наведено на рис. 6.

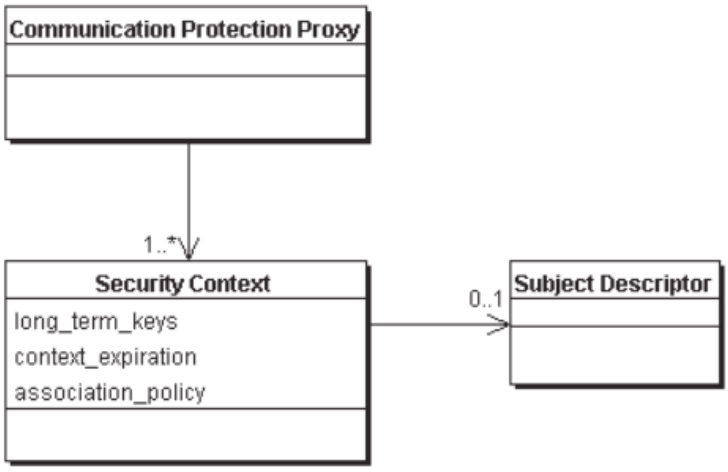

Рис. 6. Загальний вид діаграми шаблону «Захищений вміст»

Досліджено, що виділення алгоритму з двійкового коду з використанням шаблону «Захищений вміст»:

– інкапсулює атрибути безпеки, пов'язані з процесом та користувачем. Використання контексту безпеки дозволяє атрибутам безпеки, криптографічним клавішам і атрибутам безпеки процесу обробляти як окремий об'єкт;

– забезпечує точку контролю доступу.

Контекст безпеки включатиме атрибути або додатки, які дозволяють абонентам отримувати надзвичайно конфіденційну інформацію (наприклад, довготривалі криптографічні ключі, що належать суб'єкту). Ця інформація повинна бути захищена від розголошення або неправильного використання. Таким чином, реалізація контексту безпеки повинна буде захищати конфіденційну інформацію, що міститься в ній.

Контроль доступу може бути неявним, якщо система архітектурована таким чином, що лише авторизовані абоненти можуть отримати посилання на контекст безпеки. Якщо неавторизованим абонентам можливо виявити посилання на контексти безпеки, для реалізації потрібно буде надавати доступ, які перевіряють авторизацію абонента перед поверненням конфіденційної інформації.

Висновки

Проведена оцінка достовірності отриманих результатів і обґрунтування практичних рекомендацій по їх використанню. Для цього проведено імітаційне моделювання та проведено порівняльне дослідження результатів математичного моделювання та експерименту.

Результати проведених експериментів показали, що для всіх досліджуваних видів даних довірча ймовірність того, що значення статистичної величини $t_{\text{тест}}^{(i)}$ “не відхиляється” від математичного очікування $\hat{t}_{\text{тест}}^{(i)}$ більш ніж на 1 дорівнює: $P \approx 0,94$.

На основі отриманих при імітаційному моделюванні результатів проведено їх синтез в єдину методологічну систему практичного застосування технологій тестування безпеки програмного забезпечення однорангової комп'ютерної системи.

Запропоновано впровадження моделі «тесту на проникнення» та методу виділення алгоритму з двійкового коду при розробці шаблонів «Захищена система», «Політика безпеки», «Суб'єктивний дескриптор» та «Захищений вміст».

Список використаної літератури

1. Кибербезопасность 2017–2018: цифры, факты, прогнозы. – // https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/cybersecurity-2017-2018-rus.pdf.

2. Путь к киберустойчивости: прогноз, защита, реагирование. 19-е международное исследование EY в области информационной безопасности за 2016-2017 годы. – // http://www.ey.com/Publication/vwLUAssets/EY-giss-2016-rus/$File/EY-giss-2016-rus.pdf.

3. Cisco 2018. Годовой отчет по информационной безопасности. – // https://www.cisco.com/c/dam/global/ru_ru/assets/offers/assets/cisco_2018_acr_ru.pdf

4. Lysytsia D. Model of data preparation for allocation of algorithm from binary code for the safety analysis of the software / D. Lysytsia, S. Semenov, A. Lysytsia // Advanced information system. – 2018. – Vol.2, No.2. – P. 53-57.

5. Lysytsia D. Optimization of algorthm allocation from binary code / D. Lysytsia, S. Semenov, A. Lysytsia // News of science and education. – 2018/ – Vol. 3, No.59. – P. 94-100.

6. Lysytsia D. Gert-model of processes of active analysis of the system resource management and implementation in the computer system / D. Lysytsia, S. Semenov, A. Lysytsia // Středoevropský věstník pro vědu a výzkum. – 2018. – № 6(50). – P. 103-110.

7. Загрузка и моделирование данных. – // https://help.qlik.com/ru-RU.

8. Филиппов А. Н. Метод нумерации значений и использование его результатов в оптимизациях программ / А. Н. Филиппов // Информационные технологии. – 2009. – №4. – С. 43-49.

References (MLA)

1. Cyber Security 2017-2018: Figures, Facts, Forecasts. https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/cybersecurity-2017-2018-rus.pdf. Web. 13 May 2018.

2. The Path to Cyber Stability: Forecast, Protection, Response. The 19th EY International Research in the Field of Information Security for 2016-2017. http://www.ey.com/Publication/vwLUAssets/EY-giss-2016-rus/$File/EY-giss-2016-rus.pdf. Web. 13 May 2018.

3. Cisco 2018. Annual Information Security Report. https://www.cisco.com/c/dam/global/ru_ru/assets/offers/assets/cisco_2018_acr_ru.pdf. Web. 13 May 2018.

4. Lysytsia D., Semenov S., and Lysytsia A. "Model of Data Preparation for Allocation of Algorithm from Binary Code for the Safety Analysis of the Software." *Advanced Information system* 2(2) (2018): 53-57. Print.

5. Lysytsia D., Semenov S., and Lysytsia A. "Optimization of Algorthm Allocation from Binary Code." *News of Science and Education* 3(59) (2018): 94-100. Print.

6. Lysytsia D., Semenov S., and Lysytsia A. "Gert-Model of Processes of Active Analysis of the System Resource Management and Implementation in the Computer System." *Středoevropský Věstník pro Vědu a Výzkum*. 6(50) (2018): 103-110. Print.

7. Data Loading and Modeling. https://help.qlik.com/ru-RU. Web. 13 May 2018.

8. Filippov A. N. "The Method of Numbering Values and the Use of its Results in Optimizing Programs." *Information Technology* 4 (2009): 43-49. Print.

Автори статті

Козелков Сергій Вікторович – доктор технічних наук, професор, директор Навчально-наукового інституту телекомунікацій та інформатизації, Державний університет телекомунікацій, Київ. Тел. +380 (44) 249 2926. E-mail: nniti_dut@ukr.net.

Лисиця Дмитро Олександрович – аспірант кафедри обчислювальної техніки та програмування, Національний технічний університет «Харківський політехнічний інститут», Харків. Тел.: +380 66 284 2009. E-mail: lysytsia-mail@ukr.net.

Семенов Сергій Геннадійович – доктор технічних наук, професор, завідувач кафедри обчислювальної техніки та програмування, Національний технічний університет «Харківський політехнічний інститут», Харків. Тел.: +380 50 300 7647. E-mail: s_semenov@ukr.net.

Лисиця Аліна Олександрівна – студентка, кафедра інформаційних систем, Національний технічний університет «Харківський політехнічний інститут», Харків. Тел.: +380 50 342 3000. E-mail: a.lisitsa@profkom-khpi.org.

Authors of the article

Kozelkov Serhiy Viktorovych – doctor of sciences (technical), director of the Educational-Scientific Institute of Telecommunications and Informatization, State University of Telecommunications, Kyiv. Phone: +380 (93) 542 27 73. E-mail: nniti_dut@ukr.net.

Lysytsia Dmytro Oleksandrovych – postgraduate student of the Department of Computer Science and Programming, National Technical University "Kharkiv Polytechnic Institute". Phone: +380 66 284 2009. E-mail: lysytsia-mail@ukr.net.

Semenov Serhii Hennadiiovych – doctor of sciences (technical), head of the Department of Computing Technology and Programming, National Technical University "Kharkiv Polytechnic Institute". Phone: +380 50 300 7647. E-mail: s_semenov@ukr.net.

Lysytsia Alina Oleksandrivna – student at the Department of Information Systems, National Technical University "Kharkiv Polytechnic Institute". Phone: +380 50 342 3000. E-mail: a.lisitsa@profkom-khpi.org.

Дата надходження
в редакцію: 16.03.2018 р.

Рецензент:
доктор технічних наук В. В. Онищенко
Державний університет телекомунікацій, м. Київ,