УДК 004.056

Л. М. Куперштейн
Г. Л. Луцишин

АРХІТЕКТУРА СИСТЕМИ МОНІТОРИНГУ БЕЗПЕКИ ДАНИХ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вінницький національний технічний університет

Анотація

Запропоновано розробку архітектури системи моніторингу безпеки даних програмного забезпечення з метою підвищення захищеності допоміжного та кінцевого програмного забезпечення в процесі його функціонування за рахунок універсальності спроектованого рішення з можливостями узгодження процесів перевірки безпеки даних у клієнт-серверних взаємодіях та інтеграції у програмні технології розробки програмного забезпечення з використанням уніфікованих інтерфейсів задання правил перевірки.

Ключові слова: безпека застосунку, моніторинг даних, XSS, ін'єкції, перевірка введення, помилки валідації.

Abstract

Architecture of software data security monitoring system with the aim of increasing the security of auxiliary and end-user software in run-time based on the universal solution designed with the possibility of synchronization of data security validation processes in client-server interactions and integration into software development technologies using unified interfaces of validation rulesets definitions.

Keywords: application security, data monitoring, XSS, injections, input validation, validation errors.

Вступ

Сьогодні все більше процесів діяльності людини, держави та бізнесу відбуваються в електронному вигляді [1], тому потреба в безпеці різнотипних даних невпинно зростає. Проблема моніторингу безпеки та коректності даних програмного забезпечення сьогодні являється особливо актуальною, що підтверджується відомими стандартами безпеки, такими як PCI DSS [2], OWASP Top Ten [3], CWE Top 25 [4], а також методичними фреймворками імплементації безпеки застосунків, такими як OWASP Security Knowledge Framework [5] та OWASP Web Security Testing Guide [6].

Можливості моніторингу безпеки та коректності даних програмного забезпечення існуючих засобів мають частковий характер: орієнтація на окремі типи даних, платформна залежність, вузькі можливості інтеграції з іншими засобами розробки програмного забезпечення, обмеженість застосування, низький рівень повторного використання готових перевірених рішень та інші.

Метою дослідження є підвищення захищеності допоміжного та кінцевого програмного забезпечення в процесі його функціонування за рахунок універсальності архітектури з можливостями узгодження процесів перевірки безпеки даних у клієнт-серверних взаємодіях та інтеграції у програмні технології розробки програмного забезпечення з використанням уніфікованих інтерфейсів задання правил перевірки.

Результати дослідження

Встановлено, що в розробці програмного забезпечення використовуються різні типи даних, які воно обробляє, залежно від призначення [7]. Наступні категорії даних потребують перевірки безпеки:

- вхідні та вихідні дані кінцевого користувача;
- вхідні та вихідні дані прикладного програмного інтерфейсу (API) сервісів;
- моделі даних, що репрезентують бізнес рішення;
- моделі даних репозиторіїв програмного забезпечення;
- конфігураційні дані програмного забезпечення.

Визначено наступні методи забезпечення безпеки даних, згідно з дослідженими стандартами безпеки застосунків та методичними фреймворками:

- перевіряти усі вхідні дані: довжину, діапазон, формат і тип;
- реалізувати неявну перевірку вхідних даних, використовуючи такі строгі типи, як числа, логічні значення, дати, час або фіксовані діапазони даних;
- обмежити введення текстових даних регулярними виразами;
- відхиляти неочікуваний/некоректний вміст.

Встановлено, що популярні існуючі засоби моніторингу безпеки даних, такі як FluentValdation, Microsoft EnterpriseLibrary, Jakarta Bean Validation, Yup [8 – 11] мають наступні недоліки:

- обмеженість способів задання правил перевірки даних;
- підтримка лише клієнтської або лише серверної перевірки даних;
- дизайн та імплементація орієнтовані лише на одну програмну платформу;
- слабка інтегрованість із технологіями розробки програмного забезпечення;
- обмежені можливості способів зберігання наборів правил перевірки;
- фіксованість множини можливих результатів перевірки.

Було розроблено універсальну архітектуру програмного фреймворку як варіанту імплементації системи моніторингу, що представлено на рисунку 1.

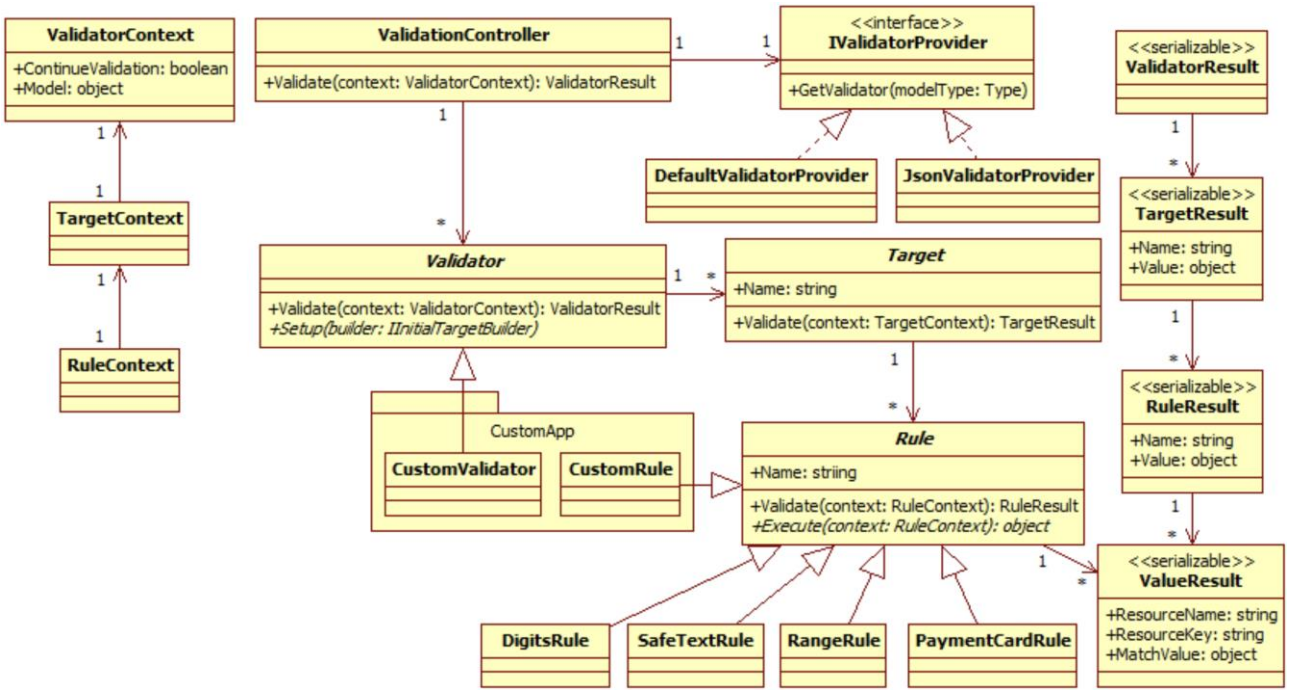

Рис. 1. Універсальна архітектура фреймворку у вигляді діаграми класів

Кожна модель даних перевіряється окремим набором правил перевірки. Клас Validator містить лише універсальну логіку виконання правил, заданих у методі Setup у похідних класах для перевірки моделей даних ПЗ. Це забезпечує однотипність перевірки для різнотипних даних та розширюваність набору правил. Для універсальності алгоритму застосування наборів правил до полів об’єктів введено клас Target, що представляє ціль перевірки, а похідні класи застосовують цей алгоритм відповідно до конкретного типу цілі (масиви елементів, їх окремі елементи та ін.). Також визначено базовий клас Rule, де похідні класи реалізують алгоритми згідно з рекомендованими методами перевірки безпеки даних. Управляючим компонентом є клас ValidationController, задачею якого є запуск та управління процесом моніторингу об’єктів даних. Вимога перевірки даних з урахуванням контексту виконання забезпечується набором зв’язаних класів ValidatorContext, TargetContext та RuleContext. Програмний код, який використовує засіб моніторингу, передає власний екземпляр класу ValidatorContext з потрібними параметрами. Результати моніторингу даних представлені набором серіалізованих класів ValidatorResult, TargetResult, RuleResult та ValueResult. Останній представляє об’єкти результатів як порушених, так і дотриманих правил. Таким чином, від розробників програмного забезпечення вимагається лише декларування валідаторів моделей даних з повторним використанням готових правил перевірки.

Програмну реалізацію розробленої архітектури було успішно виконано мовою C# для платформи .NET у вигляді фреймворку згідно з формалізованими вимогами до програмного засобу. Тестування було успішно проведено на прикладах найбільш популярних загроз, а саме SQL-ін'єкцій, Cross-site Scripting та Out-of-bounds Write. За рахунок імплементації рекомендованого підходу "білий список" програмний код задання правил перевірки та запуску процесу моніторингу залишався незмінним, що підтвердило універсальність розробленої архітектури.

Висновки

В ході дослідження було встановлено, що вразливості та загрози даних програмного забезпечення входять до списку найбільш актуальних і для безпечного функціонування програмного забезпечення необхідним є моніторинг безпеки усіх видів об'єктів даних. Було спроектовано універсальну архітектуру засобу моніторингу безпеки даних з можливістю його мультиплатформної реалізації. Обраний підхід на основі правил забезпечує відповідність засобу рекомендованим методам перевірки безпеки даних та розширюваність. Успішність розробленої архітектури було підтверджено реалізацією фреймворку для платформи .NET. Розробка дозволяє коротити час на інтеграцію методів перевірки безпеки даних у 10 разів на основі об'єму вихідного коду та у 4.5 рази на основі об'єму виконуваного коду.

Основними напрямками подальшого вдосконалення вбачаються розширення стандартного набору правил перевірки, розробка адаптерів для інтеграції засобу з відомими фреймворками та бібліотеками розробки програмного забезпечення та підтримка об'єктно-орієнтованого стилю задання наборів правил перевірки. Передбачається, що розроблена система моніторингу набуде широкого використання не лише в комерційній розробці програмного забезпечення, але і в навчальному та науковому застосуванні.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Digitalisation in Europe - 2023 edition [Electronic resource] // European Commission. – Mode of access: https://ec.europa.eu/eurostat/web/interactive-publications/digitalisation-2023 (date of access: 08.12.2023).
2. Official PCI Security Standards Council Site [Electronic resource] // PCI Security Standards Council. – Mode of access: https://www.pcisecuritystandards.org (date of access: 08.12.2023).
3. OWASP Top Ten | OWASP Foundation [Electronic resource] // OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. – Mode of access: https://owasp.org/www-project-top-ten (date of access: 08.12.2023).
4. CWE - 2023 CWE Top 25 Most Dangerous Software Weaknesses [Electronic resource] // CWE - Common Weakness Enumeration. – Mode of access: https://cwe.mitre.org/top25/archive/2023/2023_top25_list.html (date of access: 08.12.2023).
5. Security Knowledge Framework [Electronic resource] // Security Knowledge Framework. – Mode of access: https://www.securityknowledgeframework.org (date of access: 08.12.2023).
6. OWASP Web Security Testing Guide | OWASP Foundation [Electronic resource] // OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. – Mode of access: https://owasp.org/www-project-web-security-testing-guide (date of access: 08.12.2023).
7. Fowler M. Patterns of Enterprise Application Architecture / Martin Fowler. – [S. l.] : Pearsonn, 2012. – 560 p.
8. FluentValidation – FluentValidation documentation [Electronic resource] // FluentValidation – FluentValidation documentation. – Mode of access: https://docs.fluentvalidation.net/en/latest (date of access: 08.12.2023).
9. Banishing Validation Complication: Using the Validation Application Block [Electronic resource] // Microsoft Learn: Build skills that open doors in your career. – Mode of access: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/dn440720(v=pandp.60) (date of access: 08.12.2023).
10. Jakarta Bean Validation - Home [Electronic resource] // Jakarta Bean Validation - Home. – Mode of access: https://beanvalidation.org (date of access: 08.12.2023).
11. GitHub - jquense/yup at pre-v1 [Electronic resource] // GitHub. – Mode of access: https://github.com/jquense/yup/tree/pre-v1 (date of access: 08.12.2023).

Куперштейн Леонід Михайлович – к.т.н., доцент кафедри захисту інформації, Вінницький національний технічний університет, м. Вінниця, e-mail: kupershtein.lm@gmail.com

Луцишин Геннадій Леонідович – студент групи 1БС-22м, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: abitstudent16@gmail.com

Kupershtein Leonid M. – PhD, Associated Professor of Information Protection Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: kupershtein.lm@gmail.com

Lutsyshyn Hennadii L. – student of Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, e-mail: abitstudent16@gmail.com