МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Одеська юридична академія»

ЄВРОПЕЙСЬКІ ОРІЄНТИРИ РОЗВИТКУ УКРАЇНИ: НАУКОВО-ПРАКТИЧНИЙ ВИМІР В УМОВАХ ВОЄННИХ ВИКЛИКІВ

МАТЕРІАЛИ
Міжнародної науково-практичної конференції

Одеса, 26 квітня 2024 року

Одеса
«Фенікс»
2024

УДК 005.332.2(4):316.42(477)“364”“20”(062.552)
Є 24

Рекомендовано до друку вченою радою
Національного університету «Одеська юридична академія»
(протокол № 8 від 15.05.2024 р.)

За загальною редакцією ***С. В. Ківалова***

Відповідальний за випуск ***М. Р. Аракелян***

Є 24 **Європейські орієнтири розвитку України: науково-практичний вимір в умовах воєнних викликів** : матеріали Міжнар.наук.-практ. конф. (Одеса, 26 квітня 2024 р.) / за заг. ред. С. В. Ківалова. – Одеса : Фенікс, 2024. – 1000 с.
ISBN 978-617-8430-05-4

У збірнику відображено наукові напрацювання вчених, практиків, військовослужбовців у теоретичній та емпіричній площині у сферах філософських основ, загальної теорії та історичних досліджень держави і права, актуальних проблем світових соціально-політичних процесів, соціології та психології в умовах повномасштабного військового вторгнення. Висвітлено питання загроз національній безпеці в їх конституційному вимірі у рамках міжнародного та європейського права, трудового права та права соціального забезпечення, земельного, аграрного та екологічного права, пов’язані з функціонуванням економіки та підприємництва в умовах європейського вибору України. Розглянуто проблеми інформатизації та цифровізації суспільства, захисту інформації та кібербезпеки в умовах військового вторгнення, питання методики викладання іноземних мов, теорії та практики перекладу, проблеми лінгвістики та журналістики. Відображено наукові напрацювання у сферах адміністративного права та процесу, фінансового, морського та митного права, організації та вдосконалення судоустрою, прокуратури, інших правоохоронних органів, адвокатури. Висвітлено питання кримінального права, кримінально-процесуальних аспектів кримінального провадження та кримінологічних особливостей протидії злочинності в умовах воєнного стану, криміналістики, судової експертизи, психології та медицини у забезпеченні судочинства, цивільного та сімейного права, інтелектуальної власності та патентної юстиції, цивільного судочинства, господарського права та процесу. Розглянуто актуальні питання діяльності сучасних бібліотек у закладах вищої освіти, фізичної підготовки здобувачів вищої освіти.

Збірник розраховано на наукових і науково-педагогічних працівників, здобувачів вищої освіти, практичних працівників у сферах юридичної, економічної, соціологічної, політологічної, психологічної, філологічної наук, журналістики та кібербезпеки тощо.

УДК 005.332.2(4):316.42(477)»364»»20»(062.552)

Матеріали видано в авторській редакції

ISBN 978-617-8430-05-4

СЕКЦІЯ 24. КІБЕРБЕЗПЕКА ОСОБИ, СУСПІЛЬСТВА ТА ДЕРЖАВИ ЯК ФУНДАМЕНТАЛЬНИЙ ЕЛЕМЕНТ ЗАБЕЗПЕЧЕННЯ ОБОРОНОЗДАТНОСТІ УКРАЇНИ

СЕКЦІЯ 25. ПИТАННЯ МЕТОДИКИ ВИКЛАДАННЯ ІНОЗЕМНИХ МОВ, ТЕОРІЇ ТА ПРАКТИКИ ПЕРЕКЛАДУ ЯК ЗАСОБУ НАЛАГОДЖЕННЯ КОМУНІКАЦІЇ

Як висновок можна відзначити те, що прогрес удосконалення систем ШІ за швидкістю явно випереджає процес створення відповідного нормативно-правового забезпечення, що в результаті може призвести до абсолютно непередбачуваних наслідків.

Список використаних джерел

1. Dario Amodei. For a hearing on "Oversight of A. I.: Principles for Regulation". URL: https://www.judiciary.senate.gov/imo/media/doc/2023-07-26_-_testimony_-_amodei.pdf..
2. Чанишев Р. І. Проблеми розвитку та правового регулювання штучного інтелекту. *Європейські орієнтири розвитку України в умовах війни та глобальних викликів XXI століття: синергія наукових, освітніх та технологічних рішень* : матеріали Міжнар. наук.-практ. конф. (м. Одеса, 19 трав. 2023 р.) : у 2 т. / за заг. ред. С. В. Ківалова. Одеса, 2023. С. 603–606.
3. Чанишев Р. І. Проект Ai Act: перша спроба правового регулювання використання технологій штучного інтелекту в Європейському Союзі. *Інформаційне суспільство: проблеми та перспективи :* матеріали VIII Всеукр. наук.-практ. конф. (м. Одеса, 2 черв. 2023 р.) /відп. ред. Н. І. Логінова. Одеса, 2023. С. 42–44. DOI: https://doi.org/10.32837/11300.25263.
4. Stempel J. Nvidia is sued by authors over AI use of copyrighted works. URL: https://www.reuters.com/technology/nvidia-is-sued-by-authors-over-ai-use-copyrighted-works-2024-03-10.
5. Stempel J. NY Times sues OpenAI, Microsoft for infringing copyrighted works. URL: https://www.reuters.com/legal/transactional/ny-times-sues-openai-microsoft-infringing-copyrighted-work-2023-12-27.
6. David E., Weatherbed J. The EU AI Act passed – here's what comes next. URL: https://www.theverge.com/2023/12/14/24001919/eu-ai-act-foundation-models-regulation-data

Ключові слова: штучний інтелект, проблеми, що пов'язані із системами штучного інтелекту, правове регулювання систем штучного інтелекту, безпека людства, законодавство Європейського Союзу

Keywords: artificial intelligence, problems related to artificial intelligence systems, legal regulation of artificial intelligence systems, safety of mankind, European Union legislation

Чикунов Павло Олександрович

Національний університет «Одеська юридична академія»,
доцент кафедри інформаційних технологій,
кандидат технічних наук, доцент

ВІДСТЕЖЕННЯ ПОШИРЕННЯ ПО ПРОГРАМІ НЕПЕРЕВІРЕНИХ ЗОВНІШНІХ ДАНИХ

Taint-аналіз (taint checking) [1] – це метод аналізу текстів програм на мовах високого рівня, який дозволяє відстежувати поширення неперевірених зовнішніх даних за програмою для виявлення потенційних безпекових загроз. Суть його полягає у відстеженні потенційно небезпечних або ненадійних даних, що вводяться у програму і можуть бути використані для атаки.

В процесі виконання програми taint-аналіз визначає, які дані можуть бути забруднені або підозрілими, і як вони можуть вплинути на безпеку програми. Основні заходи taint-аналізу включають відстеження потоку даних, визначення зв'язків між даними та їх джерелами, виявлення можливих точок витоку даних тощо. В результаті він допомагає підвищити безпеку програмного забезпечення шляхом виявлення та усунення можливих шляхів атаки.

Потрапляння неперевірених даних у критичні секції коду може призвести до різних вразливостей, включаючи SQL-ін'єкцію (виконання зловмисником небажаних SQL-запитів), міжсайтовий скриптинг XSS (впровадження шкідливого коду у веб-сторінки, що відвідують інші

користувачі) та багато інших. Зловмисники можуть використовувати ці вразливості для порушення коректної роботи системи, отримання конфіденційних даних або проведення інших несанкціонованих операцій.

Ключова ідея taint-аналізу полягає в тому, що будь-яка змінна, яка може бути змінена зовнішнім користувачем, становить потенційну загрозу безпеці. Якщо ця змінна використовується у певному вираженні, то значення цього виразу також стає підозрілим. Далі алгоритм відслідковує і сповіщає про ситуації, коли будь-яка з цих позначених змінних використовується для виконання небезпечних команд, наприклад, прямих запитів до бази даних або операційної системи комп'ютера.

Для своєї роботи taint-аналіз потребує конфігурації, в якій методам програми можна призначити одну з ролей: taint-source (джерело неперевірених даних) або taint-sink (приймач, деяка критична секція програми).

Джерелом неперевірених даних може бути метод зчитування даних користувача або метод отримання параметра HTTP-запиту. Змінні, до яких записується результат виконання taint-source, називаються позначеними. Приймачем може бути метод прямого звернення до бази даних або файлової системи, а також будь-яка інша потенційно небезпечна операція.

Також є дві допоміжні сутності, які потрібні для розширення можливостей алгоритму. Taint-pass – це функція, яка позначає значення, що повертається з урахуванням міток у її аргументах. Залежно від реалізації, мітки можуть накладатися не тільки на результат методу, а й на сам об'єкт, на вхідні параметри методу також. Taint-cleaner – це функція, яка видаляє заданий набір міток із переданих аргументів. Найчастіше це деякий метод валідації, який перевіряє, чи дійсно користувач ввів дані в очікуваному форматі.

Алгоритм taint-аналізу сканує граф потоку даних, намагаючись виявити маршрут між методом з множини taint-sources і методом з taint-sinks. Основною проблемою класичного алгоритму, як і в усіх методів неточного аналізу, є велика кількість хибно-позитивних спрацьовувань.

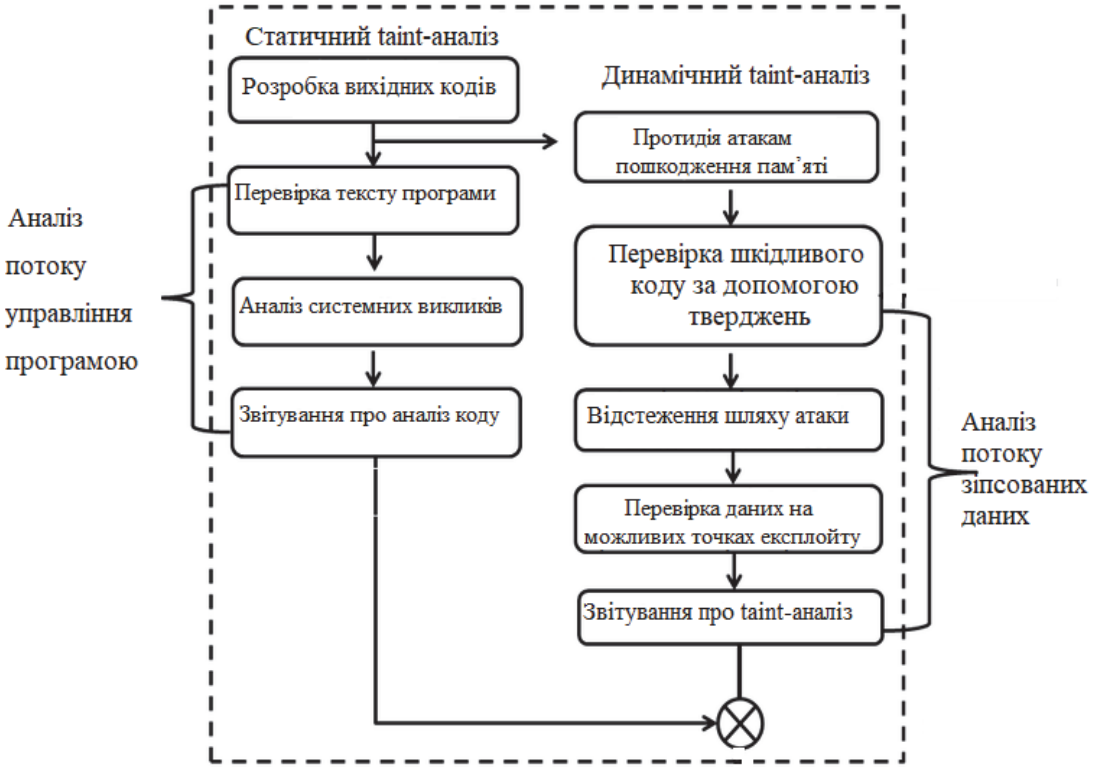

Рис. 1. Загальний алгоритм проведення taint-аналізу [1]

Наразі відомі дослідження [2–3], у яких ця проблема вирішується за допомогою застосування техніки символьного виконання коду, яка дозволяє знайти всі набори вхідних даних, що сприяють виконанню кожного з його можливих шляхів.

Статичний аналіз – це аналіз програмного забезпечення, що проводиться (на відміну динамічного аналізу) без реального виконання досліджуваних програм [4–5]. Найчастіше аналіз проводиться над вихідним кодом продукту, хоча іноді задіяний і якийсь вид об'єктного коду, наприклад, байт-код мови Java.

У зв'язку з високою затребуваністю теми автоматичного виявлення дефектів у програмах, на ринку існує безліч різних статичних аналізаторів. Список найвідоміших інструментів, що дозволяють аналізувати Java-код, включає SpotBugs, Coverity, CodeQL, Infer, а також багато інших [6]. Перелічені інструменти непогано справляються з пошуком простих помилок або друкарських помилок, проте здебільшого не виконують глибокого дослідження коду, тобто не виявляють рідкісні сценарії виконання, що призводять до некоректної поведінки програми. Крім того, часто допускається велика кількість хибних спрацьовувань.

Для оцінки якості аналізатора коду зазвичай використовується реєстр найнебезпечніших вразливостей програмного забезпечення, що включає зокрема такі помилки, як запис або читання поза межами масиву, SQL-ін'єкція, використання неперевіреного введення користувача, розйменування нульового покажчика, а також багато інших вразливостей. Символьне виконання здатне знаходити далеко не всі з перелічених помилок, проте його можливості можуть бути суттєво розширені за допомогою методу taint-аналізу.

Зазначені аспекти визначили необхідність розробки інструменту, який би виконував глибокий аналіз текстів програм на мові Java та знаходив серйозні проблеми, що призводять до несподіваної поведінки програми.

Запропонований авторами метод полягає у відстеженні поширення неперевірених зовнішніх даних за програмою і дозволяє виявляти порушення безпеки, наприклад, ін'єкції в базу даних, операційну або файлову систему. Таким чином, якщо вбудувати taint-аналіз безпосередньо в ядро символьної віртуальної машини, то можна поєднати переваги кожного з підходів, а саме отримати аналізатор з низьким рівнем хибних спрацьовувань, який, крім усього іншого, вміє знаходити проблеми в безпеці.

У рамках дослідження як символьну віртуальну машину взято UnitTestBot [7]. UnitTestBot може знаходити тестові випадки, що призводять до таких дефектів у програмному забезпеченні, як викидання необробленого виключення, переповнення примітивних чисельних типів, зависання методів та деякі інші. Однак самі по собі знайдені значення вхідних параметрів не є прийнятним результатом статичного аналізу, оскільки якість аналізатора вимірюється не тільки в його здатності знаходити неочевидні сценарії виконання, а й у зручності використання для програміста. У зв'язку з цим було вирішено додати функціональність виконання статичного аналізу, а також відображення його результатів безпосередньо до плагіну UnitTestBot.

Загальний вид алгоритмів обробки правил taint-source taint-sink представлений на рис. 2. У випадку правила taint-source, для кожної конфігурації правила (rule), яке відноситься до поточного методу, та для кожної сутності (entity) всередині нього, алгоритм отримує відповідну йому символьну змінну (symbol), після чого додає до її набору міток (marks[symbol]) мітку rule. Алгоритм обробки правила taint-sink (2) виглядає також з тією лише різницею, що замість додавання, символьна машина перевіряє наявність заборонених міток (rule.marks) у безлічі marks[symbol], якщо пересічення множин не порожнє, то повідомляє систему про знайдену вразливість.

Algorithm 1: Обробка джерела позначених даних

```
ProcessTaintSource (config, stmt)
    Data: config – набір правил taint-аналізу
    Data: stmt – поточна інструкція (виклик методу)
    rules = config.getSourcesBy(stmt);
    foreach rule∈ rules do
        foreach entity∈ rule.entities do
            symbol ← getSymbolicValue(stmt, entity);
            marks[symbol] =marks[symbol] ∪ rule.marks;
        end
    end
```

Algorithm 2: Обробка приймача позначених даних

```
ProcessTaintSink (config, stmt)
    Data:config – набір правил taint-аналізу
    Data:stmt – поточна інструкція (виклик методу)
    rules = config.getSinksBy(stmt);
    foreach rule∈ rules do
        foreach entity∈ rule.entities do
            symbol ←getSymbolicValue(stmt, entity);
            if marks[symbol] ∩ rule.marks≠∅ then
                reportProblem();
            end
        end
    end
```

Рис. 2. Приклади алгоритмів обробки правил

Список використаних джерел

1. Wang P., Chao W. J., Chao K. M., Lo C. C. Using Taint Analysis for Threat Risk of Cloud Applications. *11th IEEE International Conference on E-Business Engineering*. Guangzhou, China, 2014. P. 185–190.

2. Baldoni R., Coppa E., D'elia D. C., Demetrescu C., Finocchi I. A Survey of Symbolic Execution Techniques. *ACM Computing Surveys*. 2018. Vol. 51, Iss. 3. P. 1–39.

3. Bertrand Van Ouytsel Ch.-H. Analysis and classification of malware based on symbolic execution and machine learning : Doctoral dissertation. Munich, 2024.

4. Прищепа О., Доценко О. Огляд статичних методів аналізу зловмисного програмного забезпечення. *Комп'ютерні науки та кібербезпека.* 2020. № 2. С. 15–24.

5. Білоус І. В., Нагорний П. В. Статичний аналіз коду C# з використанням Roslyn API. *Інформаційні моделюючі технології, системи та комплекси* : зб. матеріалів доп. учасників III Міжнар. наук.-практ. конф. / Черка. нац. унт. ім. Б. Хмельницького. Черкаси, 2021. С. 54–57.

6. Вступ до SpotBugs: інструмент для статичного аналізу коду. URL: https://cutt.ly/Hw8jwhPj

7. Sapozhnikov A., Olsthoorn M., Panichella A., Kovalenko V., Derakhshanfar, P. TestSpark: IntelliJ IDEA's Ultimate Test Generation Companion. *ACM/IEEE 46th International Conference on Software Engineering: Companion Proceedings.* 2024. P. 30–34.

Ключові слова: taint-аналіз коду, вразливості, неперевірені зовнішні данні, символьне виконання, статичний аналіз, UnitTestBot, алгоритми обробки правил.

Keywords: taint checking, vulnerabilities, unverified external data, symbolic execution, static analysis, UnitTestBot, rule processing algorithms.