

комплексному підході планування циклів лабораторних випробувань різного спрямування дозволяють охопити широкий спектр ситуацій, які характеризуються невизначеністю умов.

У роботі [2] для прогнозування надійності пропонується використання прискорених методів випробувань. Представлена у роботі модель надійності підтверджує необхідність застосування технологій прогнозування з метою аналізу впливу початкових некритичних дефектів на випадкові відмови в процесі експлуатації. Досягти достовірності прогнозування надійності можливо завдяки підвищенню точності планування прискорених лабораторних випробувань.

Таким чином, одним з варіантів технології прогнозування надійності друкованих плат при плануванні лабораторних випробувань є використання методів прискорених випробувань в процесі планування на основі Agile методології за рахунок використання спеціалізованих програмних засобів.

#### Список посилань

1. Кравчук В.І., Баранов Г.Л., Комісаренко О. Інформаційна технологія прогнозування та випробування майбутньої аграрної техніки. *Техніко-технологічні аспекти розвитку та випробування нової техніки і технологій для сільського господарства України*. 2018. № 22 (36). С. 27-35.

2. Лусте О.Я. Прискорені методи випробувань для прогнозування надійності. *Термоелектрика*. 2018. № 3. С. 68-72.

УДК 004.056.5:004.056.2

**Висоцька О.О., канд. техн. наук**

Державний університет «Київський авіаційний інститут», lek\_vys@ukr.net

**Давиденко А.М., докт. техн. наук, професор**

Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України,  
davidenkoan@gmail.com

### ДОСЛІДЖЕННЯ ВРАЗЛИВОСТЕЙ АВТОМАТИЗОВАНИХ СИСТЕМ ПОВ'ЯЗАНИХ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Стрімке зростання частоти використання штучного інтелекту (ШІ) в автоматизованих системах (АС) підприємств призводить до зміщення акцентів при виявленні критичних вразливостей таких систем. Всі виявлені вразливості фіксуються в National Vulnerability Database [1]. Але застосування ШІ в АС призводить до того, що деякі вразливості стають більш актуальними, а їх наявність може мати більш критичні наслідки, ніж в системах без ШІ, крім того з'являються нові вразливості. В першу чергу критичними є атаки на моделі машинного навчання (МН) та різноманітні маніпуляції з даними. Даний факт робить актуальними задачі дослідження вразливостей в АС, які використовують ШІ [2], та пошуку оптимальних методів виявлення саме таких вразливостей.

Специфіка використання ШІ, пов'язана з використанням великих об'ємів даних і складних алгоритмів МН, породжує нові типи загроз, які були відсутні у АС без ШІ. Вразливості можуть виникати на різних рівнях, а саме: в навчальних даних; в моделі, яка використовується; під час її інтеграції в загальну архітектуру АС, тощо. Звичайні методи кібербезпеки не завжди можуть забезпечити належний рівень захисту вразливих місць ШІ-систем.

На основі проведеного дослідження були визначені наступні вразливості, котрі є найбільш поширеними та критичними для АС з використанням ШІ:

1. **Exec Code**. Приклад використання: застосування зловмисного коду для зміни моделі ШІ або впровадження зловмисного сценарію.

2. **DoS**. Приклад використання: атака на моделі ШІ з метою виведення їх з ладу під час обробки великих даних.

3. **DoS Overflow**. Приклад використання: введення надмірно великих даних у модель для порушення її функціонування.

4. **Bypass**. Приклад використання: отримання доступу до моделей ШІ без відповідної авторизації.

5. **Overflow**. Приклад використання: атака на моделі з метою викликати збій через переповнення вхідних даних.

6. **Exec Code Overflow**. Приклад використання: комбінована атака, що дозволяє виконувати зловмисний код через переповнення.

7. **Dir. Trav**. Приклад використання: перехоплення критичних даних через маніпуляції з файловою системою моделі ШІ.

8. **XSS**. Приклад використання: застосування зловмисних скриптів для зміни вхідних даних, що подаються в ШІ-модель.

9. **SQL-ін'єкції**. Приклад використання: застосування SQL-ін'єкцій для зміни або видалення навчальних даних моделі ШІ.

10. **Exec Code XSS**. Приклад використання: вставка зловмисного коду у АС для доступу до моделі ШІ та її компрометації.

11. **CSRF**. Приклад використання: атака на ШІ моделі через підроблені запити до сервера для виконання деструктивних дій.

Як вже було сказано, ці вразливості можуть виникати на різних рівнях. Такі вразливості, як Exec Code, DoS, Overflow і Bypass, стосуються базових аспектів безпеки АС, що працюють з великими обсягами даних та складними алгоритмами. Використання ШІ робить ці системи привабливими для атак, оскільки порушення в обробці даних або виконанні кодів призводить до серйозних наслідків, наприклад, до некоректних рішень або компрометації всієї системи. Системи, що автоматично приймають рішення на основі алгоритмів МН, особливо вразливі до таких загроз, тому вивчення і виявлення цих типів вразливостей є необхідним для підвищення надійності систем з використанням ШІ.

Вразливості, пов'язані з SQL-ін'єкціями, XSS та CSRF, виявляють слабкі місця в аспекті взаємодії ШІ з зовнішніми джерелами даних та інтерфейсами. ШІ-системи, які обробляють великі об'єми інформації з баз даних або інтегруються з веб-застосунками, стають особливо вразливими до маніпуляцій з боку зловмисників. Ці загрози дозволяють маніпулювати вхідними даними, що безпосередньо впливає на результативність моделі та призводить до некоректних висновків.

Важливість та необхідність дослідження саме цих вразливостей аргументується їх універсальністю. Вони можуть проявлятися як у звичайних АС, так і в системах з використанням ШІ, але саме в останніх вони набувають специфічних форм та наслідків, що значно посилює їхню небезпеку. У середовищі ШІ моделі здатні адаптуватися до нових даних, що дає змогу зловмисникам, котрі атакують систему, впливати на внутрішню структуру і поведінку системи за допомогою ворожих змін, наприклад, під час атак типу Model Extraction.

У даному випадку такі вразливості, як Exec Code Overflow, можуть бути використані для несанкціонованого модифікування навчальних моделей, а також для витоку конфіденційної інформації, яка використовується в процесі навчання. Це створює загрозу не лише для даних, але й для цілісності архітектури всієї системи. Саме універсальність цих вразливостей у поєднанні з унікальними проявами у середовищі ШІ обґрунтовує необхідність їх дослідження та пошуку оптимальних методів їх виявлення.

Також важливим є той факт, що загрози для безпеки у програмах з ШІ є більш динамічними, оскільки їх робота залежить від якості даних, налаштувань моделі та її навчання, що створює нові потенційні вразливості, яких не існує у застосунках без ШІ. Одна з ключових відмінностей у вразливостях систем з використанням ШІ полягає у залежності від навчальних даних. Якщо дані скомпрометовані, це як правило призводить до того, що

модель навчиться на зловмисних або неправильних даних, що спричинить помилки у прогнозах або навіть дозволить зловмисникам керувати результатами роботи моделі. Крім цього, система з ШІ може бути піддана атакам на рівні моделей або під час їхнього використання. Наприклад, маніпуляції з параметрами моделей можуть призвести до некоректної поведінки, що може бути використано для обходу захисних механізмів. Крім того, вразливості ШІ-застосунків у реальному часі також викликають додаткові складнощі. У системах, що працюють з великими потоками даних і приймають рішення в реальному часі, критично важливо відслідковувати зміни в поведінці моделі та виявляти потенційні загрози або аномалії у прийнятті рішень. Це ставить перед системами ШІ нові вимоги до безпеки, оскільки звичайні підходи до моніторингу часто не здатні вчасно виявити ці загрози.

Дослідження цих вразливостей допоможе не тільки виявити слабкі місця в архітектурі сучасних АС, але й розробити нові методи захисту, що враховують специфіку роботи систем з МН і ШІ. Спеціалізовані системи для аналізу архітектури програм, що використовують ШІ, мають враховувати всі етапи життєвого циклу моделі – від збору даних і навчання до прийняття рішень і взаємодії з користувачем у реальному часі. Врахування всіх зазначених особливостей систем з використанням ШІ та їх вразливостей необхідно для побудови ефективних систем захисту відповідних систем.

#### Список посилань

1. National Vulnerability Database (NVD) [Electronic resource]. – Access mode: <https://www.nist.gov/programs-projects/national-vulnerability-database-nvd>
2. Vysotska O. Modeling the mindfulness people's function based on the recognition of biometric parameters by artificial intelligence elements / O. Vysotska, A. Davydenko, O. Potenko // Radioelectronic and computer systems. 2023. – No 3 – P. 136-149. DOI: <https://doi.org/10.32620/reks.2023.3.11>

УДК 004.932:536.24

**Жульковський О.О., канд. техн. наук, доцент**  
Дніпровський державний технічний університет, [olalzh@ukr.net](mailto:olalzh@ukr.net)  
**Жульковська І.І., канд. техн. наук, доцент**  
Університет митної справи та фінансів, [inivzh@gmail.com](mailto:inivzh@gmail.com)

### **SIMD-ОПТИМІЗАЦІЯ ЧИСЕЛЬНОГО МОДЕЛЮВАННЯ ТЕПЛОВИХ РЕЖИМІВ У ТЕХНОЛОГІЧНИХ СИСТЕМАХ**

Сучасні інженерні завдання у галузі теплофізики все частіше вимагають моделювання нестационарних процесів із високою просторовою та часовою роздільною здатністю. У зв'язку з цим суттєво зростають обчислювальні витрати, необхідні для адекватного опису теплових режимів у складних технологічних об'єктах та системах. Одним із ефективних напрямів підвищення продуктивності таких обчислень є використання векторизації обчислювальних алгоритмів на рівні даних (Data-Level Parallelism), зокрема із застосуванням SIMD-інструкцій сучасних процесорів [1].

У роботі, для прикладу, розглянуто задачу чисельного розв'язання рівняння нестационарної теплопровідності у сталевому стрижні з теплоізоляцією бокової поверхні та теплообміном на торцях. Для просторово-часової дискретизації використано явну різницеву схему, побудовану методом теплового балансу. Оптимізація реалізації алгоритму здійснена через заміну скалярного циклу обчислення температури у внутрішніх вузлах сітки на SIMD-реалізацію з використанням інструкцій AVX2 і intrinsic-функцій Microsoft Visual Studio C++ [1].

Особливістю побудованого рішення є збереження алгоритмічної простоти при впровадженні векторизації лише в тій частині коду, яка не має послідовної залежності між операціями. Для обробки залишкових елементів сітки, які не вкладаються у векторні