**УДК 004.056(045)**

# СИСТЕМНИЙ АНАЛІЗ ПРОБЛЕМИ БАГАТОКРИТЕРІАЛЬНОГО ВИБОРУ ВАРІАНТУ УДОСКОНАЛЕННЯ СИСТЕМИ ЗАХИСТУ ІНФОРМАЦІЇ

***Юлія Коваленко***

*У статті представлено рішення системного аналізу проблеми багатокритеріального вибору та застосування організаційних і аналітичних методів розробки, обґрунтування та прийняття рішень для вирішення проблеми вибору засобу фаззінгу з використанням генетичних алгоритмів. Застосовано математичні методи аналізу експертних оцінок, метод аналізу ієрархій та методу парний порівнянь. Було здійснено системний аналіз системи фаззінгу з використанням генетичних алгоритмів, пошук джерел про відповідний тип системи, складено порівняльну таблицю прототипу з відомими системами такого типу, здійснено системний аналіз проблеми багатокритеріального вибору, експертна оцінка переваг та побудова матриць переваг за допомогою методу парних порівнянь, розраховано вектор глобальних пріоритетів та прийнято рішення на основі результатів аналізу.*

***Ключові слова:*** *системний аналіз, багатокритеріальний вибір, альтернативні варіанти оптимізації, фаззер, система захисту.*

**Вступ і постановка задачі.** Актуальною є проблема вдосконалення даної системи так, щоб вона краще вирішувала поставлену задачу.

У результаті дослідження спроектована СЗІ має бути універсальною, комплексною, простою у використанні, мати наочність та функціонувати в умовах високої невизначеності вихідної інформації.

Нами було поставлено такі цілі: розкрити поняття про Leveraging Control Flow for Evolutionary Input Crafting; сформулювати критерії вибору рішення; провести аналіз альтернативних варіантів вирішення проблеми; дослідити експертну оцінку переваг і побудова матриць переваг; розрахувати вектора глобальних пріоритетів.

Зазначимо, оптимізацію СЗІ можна провести декількома шляхами. Наприклад, шляхом модифікації структури через введення нових підсистем або елементів або шляхом заміни одних підсистем або елементів на інші (що потребує спеціальної програмної розробки чи покупки, що впливає на загальну вартість системи); за допомогою оптимізації існуючої структури чи удосконалення певних підсистем або елементів.

Leveraging Control Flow for Evolutionary Input Crafting – один із проектованих засобів фаззінгу з використанням генетичного алгоритму, що базується на представленні шляхів виконання програми у вигляді динамічної Марковській моделі. Першим кроком у виконанні є підготовка графу потоку управління, який у даній методології використовується як підграф усього графу контрольного потоку програми. Він складається з базових блоків на шляху між вхідним блоком та цільовим (рис.1). Граф потоку управління для виконання з двійкового коду може бути отриманий за допомогою дизассемблера IDA Pro.

Після того, як інформація графу потоку управління експортується з IDA Pro, прототип визначає всі цікаві шляхи виконання. Шлях є цікавим, якщо вузол на шляху потоку управління являє собою виклик уразливої функції С. Приклад виклику вразливою функції С є функція бібліотеки, яка не забезпечує довжини скопійованих даних, які є меншими або дорівнюють довжині буфера призначення.

Такі функції, як strcpy() реалізують це визначення. Крім того, будь-яка функція може бути позначена тестером як уразлива. Функція, strcpy(), тільки один з таких прикладів.

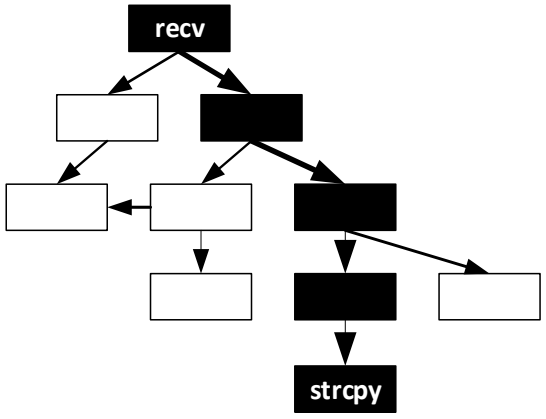


Рис. 1. Ідеалізована діаграма аналізу вхідної проблеми (тобто вхідні дані, які будуть змушувати програму дослідити логіку потоку управління на шляху від функції recv до потенційно вразливої strcpy())

Даний прототип відстежує виконання шляхом розміщення контрольних точок за допомогою відладчика на всіх пунктах входу в підграфи, витягнуті з IDA Pro. Для зменшення часу виконання, графи розбиті на дві групи. Перший на-

бір включає в себе вузли провідних шляхів, які можуть привести до виконання небезпечного виклику API. Другий набір є доповненням першого набору. Якщо програма входить у вузол в першому наборі, відторгаючи набір, то виконання припиняється. В іншому випадку, виконання триває до виклику уразливої функції. Контрольні точки в підграфах служать як інформаційні точки для контролю відповідності розрахунків і моніторингу контрольних точок для зупинки виконання. Фаззер відправляє розрахунки придатності назад до генерації тестування, використовуючи типові оператори генетичного алгоритму.

Leveraging Control Flow for Evolutionary Input Crafting – система, метою якої є пошук і виявлення уразливих функцій та можливих багів, що базується на використанні генетичного алгоритму.

Під вхідним блоком розуміється початкова функція, з якої починається шлях виконання програми, а під цільовим – кінцева функція, яка зупиняє виконання програми.

Враховуючи аналіз, виконаний вище для вирішення проблеми вибору, виділимо такі показники:

– показники функціонування, що характеризують корисний ефект від використання СЗІ за призначенням;

– показники доступності, що характеризують зовнішні властивості системи;

– економічні показники, що характеризують затрати на розробку, виготовлення, експлуатацію системи.

На наступному кроці декомпозиції встановлюємо безпосередньо показники кожної групи або розбиваємо кожну групу показників ще на ряд підгруп. Результат декомпозиції такий.

Параметри функціонування;

– вхідні параметри – вхідні дані, які подаються фаззеру на початку експерименту;

– вихідні параметри – дані, які видає фаззер наприкінці експерименту;

– комплексність типу генетичного алгоритму – метод, за яким виконуються мутації, і його комплексність, тобто можливість фаззера за даним алгоритмом знаходити уразливості різного характеру;

– генерація (породження) некоректних даних;

– моніторинг виключень;

– відсоток помилкових спрацьовувань (значна частина видаваних попереджень повинна бути істинною);

– параметри вхідних даних (введення з інтерфейсних пристроїв, параметри програмного середовища, параметри командного рядка, комунікаційні протоколи, файли, дані в оперативній пам'яті;

– підтримка різних мов реалізації та платформ;

– можливість відтворення виняткової ситуації – незмінність експериментальних даних при повторному експерименті;

– покриття – можливість проаналізувати кожний шлях програми;

– автоматизація роботи – контроль за станом досліджуваного процесу (з його перезапуском у разі помилки) і автоматична реєстрація результатів роботи СЗІ;

– універсальність – можливість використання СЗІ для оцінки безпеки файлів, програм та систем різного типу чи формату;

– масштабованість аналізу – можливість проведення аналізу об'ємом декількох мільйонів рядків коду і сотень тисяч функцій;

Параметри доступності:

– наочність роботи системи;

– простота взаємодії користувача та СЗІ (зрозумілість);

– розширюваність інфраструктури аналізу (доповнення алгоритму ідентифікації різними новими класами уразливостей (дефектів і критичних помилок);

– доступність використання - якісна оцінка зручності використання СЗІ.

Економічні параметри:

– вартість розробки;

– затрати на експлуатацію системи;

– економічна ефективність експлуатації.

Застосуємо метод рядкових сум для обрання чотирьох основних критеріїв вибору рішення про удосконалення системи. У результаті використання методу рядкових сум було отримано 4 ранги основних критеріїв, кількість яких в межах цих рангів – 4. Отже, головними критеріями, за якими система буде удосконалена, є:

– універсальність – можливість використання СЗІ для оцінки безпеки файлів, програм та систем різного типу чи формату (до якого включимо критерій підтримки різних мов програмування);

– покриття – можливість проаналізувати кожний шлях виконання програми;

– масштабованість аналізу – можливість проведення аналізу об'ємом декількох мільйонів рядків коду і сотень тисяч функцій;

– доступність використання – якісна оцінка зручності використання СЗІ.

Для даної системи було знайдено декілька альтернативних варіантів оптимізації. Кожна із виявлених альтернатив проаналізована на сумісність з іншими елементами (існуючими чи альтернативними). Вибрані альтернативи оформлені у морфологічну табл. 1.

Таблиця 1

Альтернативні варіанти оптимізації

| **Класифікаційні ознаки** | **Варіанти реалізації класифікаційних ознак** | | | | | |
|---|---|---|---|---|---|---|
| Інструмент статичного аналізу | IDA Pro | | | IDA Pro/Hex-Rays | | |
| Інструмент динамічного аналізу | відсутній | TrEx | Valgrid | DynamoRio | GNU DeBugger | PIN |
| Засіб проміжного представлення аналізу бінарного коду | LLVM | | Valgrid | | Fuzzgrind | |
| Засіб трасування (симулятор) | AMD SimNow | | | Virtutech Simics | | |
| Зовнішній засіб логування | Log4 | | | | | |
| Fuzzing framework | PAIMEI | Dfuz | SPIKE | Sulley | | Autodafé |
| Засіб серіалізації даних (тільки для PAIMEI) | YAML | | | | | |
| Тип реалізації генетичного алгоритму | CHC-GA | SGA | SGA-MM | | BGA | PSGA |
| Засіб кодування рядків | ANTLR | Coco/R | | GNU bison | | LR-анализатор |

Система, аналіз якої проводиться, базується лише на використанні середовища статичного аналізу, при якому код досліджується без виконання самої програми. Безумовним лідером даної категорії є інструмент IDA Pro – інтерактивний дизассемблер і відладчик, який дозволяє перетворити бінарний код програми в асемблерний текст, що може бути застосований для аналізу роботи програми. Іншою альтернативою, що приведена у морфологічній таблиці, є плагін Hex-Rays – декомпілятор на основі IDA Pro, який конвертує бінарний код в читабельний СІ подібний псевдокод. Hex-Rays дозволяє проводити більш наочний і швидкий аналіз програмного коду.

Другий рядок альтернатив представляє набір інструментів динамічного аналізу. Альтернативна система, що проектується, може не обмежуватися лише статичним аналізом бінарного коду, бо у випадках, коли укладачі програми прийняли заходи для захисту своєї програми від аналізу (наприклад, використовували запакований код, який розпаковується під час виконання програми), статичний аналіз може не дати результатів. В таких випадках може бути використаний динамічний аналіз (аналіз трас програми) на додаток до статичного. Середовищами такого аналізу були відібрали TrEx, спеціально розроблене для інтеграції з середовищем IDA Pro. Менш зручними в плані можливої інтеграції з IDA Pro, але відомими і гнучкими інструментами динамічного аналізу є Valgrid, DynamoRio, GNU DeBugger, PIN.

Фреймворк, існуючий у системі, що аналізується, не є універсальним через підтримку лише одної мови програмування. В якості альтернативи були підібрані інші фреймворки та інструмент YAML, здатний адаптувати PAIMEI до взаємодії з багатьма мовами програмування.

Вибрані типи реалізації генетичного алгоритму відповідають меті розробки системи фаззінгу, зосереджені на пошуку уразливих функцій та можливих багів на шляху виконання програми і взаємодіють між собою.

Із сформованої множини альтернатив методом морфологічного синтезу виберемо 5 для проведення подальших досліджень. Кожен з синтезованих варіантів має включати усі необхідні елементи функціонування системи.

**Варіант 1.** Включає в себе наявний у системі інструмент статичного аналізу IDA Pro (який у даний момент є безумовним лідером даної сфери не має рівних альтернатив), середовище динамічного аналізу TrEx, інструмент проміжного представлення аналізу бінарного коду, присутній в LLVM, засіб трасування Virtutech Simics, зовнішній засіб логування з підтримкою багатьох мов програмування Log4, генетичні алгоритми CHCGeneticAlgorithm, Simple Genetic Algorithm, Simple Genetic Algorithm Mangle Mutator та засіб кодування рядків на основі контекстуальновільної граматики ANother Tool for Language Recognition (ANTLR).

**Варіант 2.** Включає в себе наявний у системі інструмент статичного аналізу IDA Pro, при цьому СЗІ буде базуватися лише на використання статичного аналізу, зовнішній засіб логування з підтримкою багатьох мов програмування Log4, фреймворк Sulley (використовує деякі інструменти фреймворку SPIKE, підсистему PyDBg фреймворку PAIMEI та монітор відладки засобу Peach), генетичні алгоритми CHCGenetic Algorithm, Simple Genetic Algorithm, Simple Genetic Algorithm Mangle Mutator, Problem Space Genetic Algorithm, LR-аналізатор для кодування рядків.

**Варіант 3.** Включає в себе наявний у системі інструмент статичного аналізу IDA Pro із включеним плагіном Hex-Rays Decompiler (який вносить ряд переваг до бінарного аналізу коду та значно спрощує взаємодію користувача і дизасемблера), зовнішній засіб логування з підтримкою багатьох мов програмування Log4, фреймворк PAIMEI, до якого включений засіб серіалізації даних YAML для можливості підтримки різних мов програмування, генетичні алгоритми Simple Genetic Algorithm, Simple Genetic Algorithm Mangle Mutator, Byte Genetic Algorithm, Problem Space Genetic Algorithm, LR-анализатор для кодування рядків.

**Варіант 4.** Включає в себе наявний у системі інструмент статичного аналізу IDA Pro із включеним плагіном Hex-Rays Decompiler (який вносить ряд переваг до бінарного аналізу коду та значно спрощує взаємодію користувача і дизасемблера), середовище динамічного аналізу Valgrid, який включає інструменти проміжного представлення бінарного коду для полегшення процесу взаємодії, засоби трасування, присутні у AMD SimNow (за необхідності зовнішнього засобу трасування), зовнішній засіб логування з підтримкою багатьох мов програмування Log4, генетичні алгоритми CHCGeneticAlgorithm, Simple Genetic Algorithm, Simple Genetic Algorithm Mangle Mutator, Byte Genetic Algorithm, Problem Space Genetic Algorithm та засіб кодування рядків на основі контекстуально-вільної граматики ANother Tool for Language Recognition (ANTLR).

**Варіант 5.** Включає в себе наявний у системі інструмент статичного аналізу IDA Pro із включеним плагіном Hex-Rays Decompiler (який вносить ряд переваг до бінарного аналізу коду та значно спрощує взаємодію користувача і дизасемблера), середовище динамічного аналізу DynamoRio, інструменти проміжного представлення аналізу бінарного коду, присутні в Fuzzgrind, засоби трасування, присутні у Virtutech Simics (за необхідності використання зовнішнього засобу трасування), логування забезпечують внутрішні інструменти, вбудовані у кожну з перелічених підсистем, генетичні алгоритми CHCGeneticAlgorithm, Problem Space Genetic Algorithm та засіб кодування рядків Coco/R.

Кожна з представлених альтернатив значно розширює можливості системи та змушує її відповідати усім відібраним критеріям, а саме робить її більш інтерактивною та універсальною, забезпечує максимальне покриття коду та масштабованість системи. На основі розрахунку вектору глобальних пріорітетів перевагу було віддано варіантам 1 та 3.

На основі вибраних альтернатив та проведених розрахунків сформовано матрицю локальних пріоритетів Dij на нижньому рівні декомпозиції. Елементами матриці Dij є вектори — колонки пріоритетів нижнього рівня ієрархії. У нашому випадку вектори подані в табл. 2, а матриця пріоритетів нижнього Dij наведена нижче.

Таблиця 2

Вектори пріоритетів нижнього рівня задачі удосконалення СЗІ

| | Універсальність | Покриття | Масштабованість | Доступність |
|---|---|---|---|---|
| Варіант 1 | 0,262 | 0,212 | 0,162 | 0,223 |
| Варіант 2 | 0,179 | 0,192 | 0,245 | 0,178 |
| Варіант 3 | 0,253 | 0,197 | 0,198 | 0,247 |
| Варіант 4 | 0,238 | 0,209 | 0,201 | 0,148 |
| Варіант 5 | 0,119 | 0,185 | 0,18 | 0,214 |

Матриця пріоритетів нижнього рівня:

$$D_{ij} = \begin{vmatrix} 0,262 & 0,212 & 0,162 & 0,223 \\ 0,179 & 0,192 & 0,245 & 0,178 \\ 0,253 & 0,197 & 0,198 & 0,247 \\ 0,238 & 0,209 & 0,201 & 0,148 \\ 0,119 & 0,185 & 0,18 & 0,214 \end{vmatrix}.$$

Глобальний вектор пріоритетів $A_i$ відшукують як добуток матриці локальних пріоритетів нижнього рівня на вектор пріоритетів верхнього рівня, а саме:

$$A_i = D_{ij} \times P_j .$$

Вектор глобальних пріоритетів дає загальну оцінку переваги кожної альтернативи. Розраховані пріоритети подані в табл. 3.

Таблиця 3

Розраховані за методом МАІ значення глобальних пріоритетів для альтернатив удосконалення СЗІ

| Варіант альтернативи | Значення пріоритету |
|---|---|
| Варіант 1 | 0,214 |
| Варіант 2 | 0,198 |
| Варіант 3 | 0,213 |
| Варіант 4 | 0,210 |
| Варіант 5 | 0,171 |

Отже, вектор глобальних пріоритетів є основою вибору рішення. На основі його розрахунку перевагу було віддано варіантам 1 та 3, які демонструють кращу універсальність та доступність, ніж інші, а також мають індивідуальні характеристики, що перевершують показники останніх (наприклад, варіант 1 забезпечує найбільше покриття, варіант 3 є найбільш доступним).

Зазначимо, що вектор глобальних пріоритетів є основою вибору рішення. Отже, перевагу надає-

мо варіанту 4 з найбільшим значенням пріоритету 0,22 і вибираємо альтернативу, що включає в себе наявний у системі інструмент статичного аналізу IDA Pro із включеним плагіном Hex-Rays Decompiler (який вносить ряд переваг до бінарного аналізу коду та значно спрощує взаємодію користувача і дизасемблера), середовище динамічного аналізу Valgrid, який включає інструменти проміжного представлення бінарного коду для полегшення процесу взаємодії, засоби трасування, присутні у Virtutech Simics, зовнішній засіб логування з підтримкою багатьох мов програмування Log4, фреймворк PAIMEI, до якого включений засіб серіалізації даних YAML для можливості підтримки різних мов програмування, генетичні алгоритми CHCGeneticAlgorithm, Simple Genetic Algorithm, Simple Genetic Algorithm Mangle Mutator, Byte Genetic Algorithm, Problem Space Genetic Algorithm та засіб кодування рядків на основі контекстуально-вільної граматики ANother Tool for Language Recognition (ANTLR).

## ЛІТЕРАТУРА


[1]. Sherri Sparks, Shawn Embleton, Ryan Cunningham, Cliff Zou. Automated Vulnerability Analysis:Leveraging Control Flow for Evolutionary Input Crafting // Conference: Computer Security Applications Conference, 2007.

[2]. Стародуб Ю.І. Фаззери формату файлу / Стародуб Ю.І. // Політ. Сучасні проблеми науки. – К.: НАУ. – 2013.– С. 157.

[3]. IDA Pro Disassembler. http://www.idapro.ru/

[4]. Тихонов А.Ю., Аветисян А.И. Комбинированный (статический и динамический) анализ бинарного кода. – Режим доступа: http://www.ispras. ru/ ru/ proceedings /docs/2012/22/isp_22_2012_131.pdf

[5]. Richard McNally, Ken Yiu, Duncan Grove, Damien Gerhardy. Fuzzing: The State of the Art / Defence Science and Technology Organisation, 2012, p.p. 34-35

[6]. Intel, Developer Zone. Pin - A Dynamic Binary Instrumentation Tool. https://software.intel.com /en-us/articles/pintool

[7]. DynamoRIO. http://dynamorio.org/

[8]. GNU DeBugger. http://en.wikipedia.org/ wiki/ GNU_Debugger

[9]. LLVM Language Reference Manual. // Llvm.org/ docs/LangRef.html

[10]. AMD SimNow Simulator, http://developer.amd. com/cpu/simnow/Pages/default.aspx

[11]. Roger Lee Seagle Jr. A Framework for File Format Fuzzing with Genetic Algorithms, 2012, p.p. 53-75

[12]. YAML. – Режим доступу:https://ru.wikipedia.org /wiki/YAML

[13]. Coco/R. – Режим доступу:https://ru.wikipedia.org / wiki/Coco/R

[14]. ANTLR. – Режим доступу: https://ru.wikipedia.org /wiki/ANTLR

[15]. LR-анализатор. – Режим доступу: https://ru. wikipedia. org /wiki/LR-анализатор


## REFERENCES


[1]. Sherri Sparks, Shawn Embleton, Ryan Cunningham, Cliff Zou. Automated Vulnerability Analysis:Leveraging Control Flow for Evolutionary Input Crafting // Conference: Computer Security Applications Conference, 2007.

[2]. Starodub Yu.I. File Format Fuzzing/ Starodub Yu.I. // Polit. Modern problems of science. – K.: NAU. – 2013.– p. 157.

[3]. IDA Pro Disassembler. http://www.idapro.ru/

[4]. Tikhonov A. YU., Avetisyan A.I. Combined (static and dynamic) analysis of binary code, http://www.ispras.ru/ru/proceedings/docs/2012/22/isp_22_2012_131.pdf

[5]. Richard McNally, Ken Yiu, Duncan Grove, Damien Gerhardy. Fuzzing: The State of the Art / Defence Science and Technology Organisation, 2012, p.p. 34-35

[6]. Intel, Developer Zone. Pin - A Dynamic Binary Instrumentation Tool. https://software.intel.com/en-us/articles/pintool

[7]. DynamoRIO. http://dynamorio.org/

[8]. GNU DeBugger. http://en.wikipedia.org/ wiki/ GNU_Debugger

[9]. LLVM Language Reference Manual. // Llvm.org/ docs/LangRef.html

[10]. AMD SimNow Simulator, http://developer. amd. com/cpu/simnow/Pages/default.aspx

[11]. Roger Lee Seagle Jr. A Framework for File Format Fuzzing with Genetic Algorithms, 2012, p.p. 53-75

[12]. YAML. https://ru.wikipedia.org/wiki/YAML

[13]. Coco/R. https://ru.wikipedia.org/wiki/Coco/R

[14]. ANTLR. https://ru.wikipedia.org/wiki/ANTLR

[15]. LR-анализатор. https://ru.wikipedia.org/wiki/LR-анализатор


## СИСТЕМНЫЙ АНАЛИЗ ПРОБЛЕМЫ МНОГОКРИТЕРІАЛЬНОГО ВЫБОРА ВАРИАНТА УСОВЕРШЕНСТВОВАНИЯ СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ


В статье представлено решение системного анализа проблемы многокритериального выбора и применения организационных и аналитических методов разработки, обоснования и принятия решений для решения проблемы выбора средства фаззинга с использованием генетических алгоритмов. Применены математические методы анализа экспертных оценок, метод анализа иерархий и метод парных сравнений. Был осуществлен системный анализ системы фаззинга с использованием генетических алгоритмов, поиск источников о соответствующем типе системы, составлена сравнительная таблица прототипа с известными системами такого типа, осуществлен системный анализ проблемы многокритериального выбора, экс-

пертная оценка преимуществ и построение матриц преимуществ с помощью метода парных сравнений, рассчитан вектор глобальных приоритетов и принято решение на основе результатов анализа.
**Ключевые слова:** системный анализ, многокритериальный выбор, альтернативные варианты оптимизации, фаззер, система защиты.

## SYSTEM ANALYSIS OF MULTICRITERION OPTIMIZATION PROBLEM FOR INFORMATION SYSTEM PROTECTION (HACKING)

The article discusses the solution of the system analysis multicriterion optimization problem and application of organizational and analytical methods development, justification and decision-making for solving the problem of choosing the means of fuzzing using genetic algorithms. Mathematical methods of analysis of expert assessments, method of analysis of hierarchies and the method of paired comparisons are applied. A systematic analysis of the fuzzing system using genetic algorithms, the search for the sources of the corresponding type system, compiling a comparative table of the prototype with known systems of this type, a systematic analysis of the problems of multicriteria selection, expert evaluation of the advantages and construction of the matrices of the advantages of using the method of pairwise comparisons, the calculated vector of global priorities and decision on the basis of the results of the analysis are represented in the article.
**Index terms**: system analysis, multicriterion optimization problem, alternative optimization options, fazzer, protection system.

**Коваленко Юлия Борисовна**, кандидат педагогических наук, доцент кафедры безопасности информационных технологий Национального авиационного университета.
E-mail: yleejulee22@gmail.com.
**Коваленко Юлія Борисівна**, кандидат педагогічних наук, доцент кафедри безпеки інформаційних технологій Національного авіаційного університету.
**Yulia Kovalenko**, Ph.D.Academic Department of IT-Security of National Aviation University.

**УДК 621.391:519.7**

# ШВИДКІ АЛГОРИТМИ ПОБУДОВИ $k$ -ВИМІРНИХ НАБЛИЖЕНЬ БУЛЕВИХ ФУНКЦІЙ

***Антон Олексійчук, Сергій Конюшок, Артем Сторожук***

*Знаходження наближень булевих функцій у певних класах функцій, що мають більш просту будову, є традиційною задачею симетричної криптографії. Зокрема, при побудові кореляційних атак на потокові шифри потрібно знаходити наближення булевих функцій від $n$ змінних $k$ -вимірними функціями, тобто такими, що є афінно еквівалентними функціям від $k<n$ змінних. Основним результатом статті є алгоритм побудови списку всіх $k$ -вимірних функцій степеня не вище $d$ , які знаходяться на відносній відстані не більше $2^{-d}(1-\varepsilon)$ від булевої функції $n$ змінних, що задасться вектором її значень, $1\le d\le k<n$, $\varepsilon\in(0,1)$. Запропонований алгоритм є більш ефективним у порівнянні з найкращим раніше відомим (у певних випадках – в 1000 та більше разів) і може бути застосований на практиці при дослідженні кореляційних властивостей функцій ускладнення потокових шифрів.*
**Ключові слова:** *потоковий шифр, нелінійний криптоаналіз, кореляційна атака, $k$ -вимірна булева функція, швидкий алгоритм, знаходження наближень булевих функцій.*

**Вступ.** Як відомо, стійкість сучасних потокових шифрів відносно кореляційних атак визначається наявністю або відсутністю наближень функцій ускладнення, що використовуються в їх конструкціях, більш просто збудованими функціями. Найвідомішими прикладами булевих функцій, які мають просту аналітичну будову, є афінні функції, а також функції, що залежать від малої кількості змінних. Більш широкий клас утворюють $k$ -вимірні функції, тобто булеві функції від довільної кількості $n$ змінних, що є лінійно еквівалентними функціям від фіксованого числа $k<n$ змінних. Дослідженню властивостей таких функцій, зокрема, як можливих наближень довільних булевих функцій, присвячено роботи [1, 3, 4, 8, 10, 14 – 16]. Відомі також різноманітні атаки на генератори гами потокових шифрів, функції ускладнення яких є $k$ -вимірними або близькими до таких [2, 5, 12, 13].

Для побудови ефективних атак на потокові шифри необхідно знаходити $k$ -вимірні функції, що є близькими до заданої функції $f$ від $n$