ISSN 2519-2310

CS&CS Journal

KARAZIN UNIVERSITY
CLASSICS AHEAD OF TIME

2(22) 2022

COMPUTER SCIENCE AND CYBERSECURITY

КОМП'ЮТЕРНІ НАУКИ ТА КІБЕРБЕЗПЕКА

ISSN 2519-2310

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені В.Н.КАРАЗІНА
V.N. KARAZIN KHARKIV NATIONAL UNIVERSITY

КОМП'ЮТЕРНІ НАУКИ ТА КІБЕРБЕЗПЕКА
COMPUTER SCIENCE AND CYBERSECURITY
(CS&CS)

Issue 2(22) 2022

Заснований 2015 року

Міжнародний електронний науково-теоретичний журнал
International electronic scientific journal

The journal publishes research articles on theoretical, scientific and technical problems of effective facilities development for computer information-communication systems and information security question based, on advanced mathematical methods, information technologies and technical means.

The journal is published every six months.
Approved for placement on the Internet by Academic Council of the Karazin Kharkiv National University (December 26, 2022, Protocol No.19).

The journal has Digital Object Identifier: **10.26565/2519-2310** (Online).

Editorial office:
V.N. Karazin Kharkiv National University
Svobody Sq., 6, office 315a, Kharkiv, 61022, Ukraine *(North building of University, 3th floor)*

E-mail: *cscsjournal@karazin.ua*
Web-page: *http://periodicals.karazin.ua/cscs (Open Journal System)*

Published articles have been internally and externally peer reviewed

В журналі публікуються наукові статті з теоретичних і науково-технічних проблем, що пов'язані зі створенням ефективних засобів комп'ютерних інформаційно-комунікаційних систем та питань захисту інформації, на основі передових математичних методів, інформаційних технологій і технічних засобів.

Журнал виходить кожні півроку.
Схвалено до розміщення в мережі Інтернет Вченою радою Харківського національного університету імені В.Н. Каразіна (26.12.2022 р., Протокол № 19).

ISSN (Онлайн): **10.26565/2519-2310**.

Редакція:
Харківський національний університет імені В.Н. Каразіна
пл. Свободи, 6, офіс 315а, Харків, 61022, Україна *(Північний корпус університету, 3 поверх)*

Електронна пошта: *cscsjournal@karazin.ua*
Веб-сторінка: *http://periodicals.karazin.ua/cscs (Open Journal System)*

Опубліковані статті пройшли внутрішнє та зовнішнє рецензування.

ЗМІСТ
TABLE OF CONTENTS

УДК 681.04 DOI: 10.26565/2519-2310-2022-2-01

АНАЛІЗ МЕТОДІВ ПОШУКУ ДАНИХ У КРИПТОГРАФІЧНО ЗАХИЩЕНІЙ БАЗІ ДАНИХ

Теймур Махмудов, Віталій Єсін

Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, 61022, Україна
timkin552364964@gmail.com, v.i.yesin@karazin.ua

Надійшла: серпень 2022. Прийнята: жовтень 2022.

Анотація: *Питання забезпечення безпеки даних, а саме їх конфіденційності та цілісності, як правило, у теперішній час вирішується за рахунок використання відповідних криптографічних примітивів з урахуванням розвитку обчислювальних потужностей. Але, у зв'язку із специфічним способом зберігання (у хмарі), виникає питання ефективності пошуку необхідної інформації. Проблема, яка розглядається у цій роботі, полягає у тому, що шифрування унеможливлює доступ до даних без ключів для зловмисника, позбавляючи при цьому, власника даних, можливості проведення пошуку за цією інформацією. В роботі розглянуто декілька методів шифрування з можливістю пошуку. Для кожного з них приведені алгоритми, приклади використання цих методів і надані пояснювальні рисунки та таблиці. Розглянуті методи є симетричними та динамічними, завдяки чому вони ефективні та мають відносно високий рівень безпеки, але низьку виразність запитів, через що знаходять найбільше використання у NoSQL базах даних. Проведено аналіз для виявлення складності та рівня безпеки методів, а також розглянута продуктивність практичних реалізацій. Зроблено висновки про доцільність використання того чи іншого методу шифрування з можливістю пошуку на практиці. Запропоновані рекомендації щодо комбінації описаних методів для отримання очікуваного результату.*

Ключові слова: *шифрування з можливістю пошуку, ключове слово, індекс, динамічне симетричне шифрування з можливістю пошуку, токен пошуку, лазівка.*

1. Вступ

За останні роки хмарні послуги стали важливою частиною життя більшості людей, замінивши та полегшивши виконання їхньої щоденної рутини. Тепер необов'язково мати на своєму персональному комп'ютері набір інструментів від Office, так як у будь-який момент можна скористатися безкоштовними аналогами Google Docs, Google Sheets або Office 365. Так само на ринку з'явилося безліч рішень для зберігання даних у хмарі: Google, Microsoft, Dropbox, Mega та багато інших, але у всіх з них стоїть питання безпеки даних, що зберігаються. Адже дані не лише передаються через Інтернет, але ще й зберігаються у третіх осіб, тому, не дивлячись на запевнення осіб, які надають послуги, необхідно пам'ятати про безліч новин про витоки даних.

Для здійснення пошуку можна завантажити всю базу даних (БД) і виконати розшифрування. Проте для більшості застосунків цій підхід є недоцільним. Якщо ж покласти на сервер розшифрування даних, щоб він міг виконувати запити і надсилати користувачеві лише результати, то знижується рівень безпеки, оскільки дані, що захищені шифруванням, розкриваються серверу. Тому слід мати можливість проведення пошуку у повному обсязі на стороні сервера без розшифрування даних, що зменшує ризик втрати конфіденційності даних.

У роботі запропоновано огляд і аналіз предметної області, що розглядається, саме, шифрування з можливістю пошуку (ШМП), проводиться його класифікація та наводяться моделі безпеки, які характеризують рівень захищеності методів ШМП. Крім того проведено стислий розгляд конструкції і аналіз відповідних методів за часовою, комунікаційною та просторовою складностями, а також за рівнем забезпечуваної ними безпеки.

Метою роботи є аналіз методів пошуку даних у криптографічно захищених БД для визначення більш ефективних та безпечних реалізацій, а також формулювання рекомендацій стосовно вибору певного методу в залежності від потреб проекту.

2. Основні поняття та тематично пов'язані роботи

Шифрування з можливістю пошуку (ШМП, *англ. searchable encryption*) – це вид шифрування, при якому мається можливість робити пошук за зашифрованими даними з мінімальним витоком інформації.

Ця особливість робить його дуже зручним при використанні хмарних сховищ, які зазвичай сприймаються як чесний-але-допитливий (honest-but-curious) супротивник – це законний учасник комунікаційного протоколу, який не буде відхилятися від визначеного протоколу, але намагатиметься дізнатися всю можливу інформацію з законно отриманих повідомлень [1].

Можливість пошуку у ШМП досягається декількома способами:

1) вироблення таким чином шифртексту, щоб за ним можна було виконувати пошук;

2) створення індексів, за якими виконуватиметься пошук.

Між цим, самі індекси можуть бути двох видів: прямі – кожному документу зіставляється множина ключових слів, та інвертовані – кожному ключовому слову зіставляються документи. Приклад цих індексів наведено на рис. 1. Інвертовані індекси працюють швидше за прямі, оскільки в БД кількість записів значно більше кількості ключових слів.

документ	ключові слова
d1	w2, w5, w7
d2	w1, w2
...	...
dn	w6, w9, wk

Прямий індекс

ключові слова	документ
w1	d2, d5, d11
w2	d1, d2
...	...
wk	d6, d12, dn

Інвертований індекс

Рис. 1 – Види індексів

Для шифрів з можливістю пошуку можна провести класичну для шифрів дихотомію на симетричне на асиметричне ШМП. Проте для таких систем є певна особливість: для симетричної системи є тільки один ключ, який є секретним, відповідно є тільки один користувач – власник ключа, що може здійснювати запис та зчитувати дані, але, якщо йде мова про асиметричну систему, то в ній наявний відкритий ключ, за допомогою якого багато користувачів можуть записувати дані на сервер, а здійснювати пошук може тільки власник секретного ключа. Більш того, якщо використовувати розподілений секретний ключ, то це дозволить створити систему, в якій буде більш однієї людини, що може робити пошук за шифртекстом. Отже, по типу «письменники/читачі» системи можуть бути 1/1, */1, 1/* та */*.

Схеми ШМП також поділяються на статичні та динамічні. Динамічні схеми ШМП – це такі, що дозволяють проводити оновлення структури даних. Тобто, якщо є деяка множина документів, то є можливість додавати нові елементи та видаляти вже присутні. При цьому реалізація цього може бути різною, деякі схеми підтримують додавання, але не підтримують видалення. Для того, щоб такі операції проводити, як правило, генерується відповідний токен оновлення (подібно до токена пошуку), в якому зазначаються ідентифікатор файлу та ключові слова, пов'язані з ним, що треба додати чи видалити. Але зрозуміло, що ці операції також можуть бути небезпечними та уможливлювати витік інформації. З цього приводу у [2] було визначено декілька понять безпеки в контексті ШМП:

1. Пряма секретність або *forward privacy* (*FP*) – гарантує, що сервер не дізнається, чи містять додані документи ключові слова, пошук за якими проводився раніше.
2. Зворотна секретність або *backward privacy* (*BP*) – гарантує, що сервер не може застосовувати запити к видаленим документам.

Більш того, у роботі [3] автори виділили три рівня зворотної секретності: перший є найпотужнішим, на цьому рівні схема при додаванні нового документу розкриває лише документи, що містять ключове слово w, що наявне у запиті, момент часу, коли вони були додані, а також загальну кількість додавань за ключовим словом w; другий рівень розкриває інформацію першого рівня, а також момент часу, коли було виконано будь-яке оновлення за ключовим словом w; третій, найслабкіший, рівень розкриває всю інформацію другого рівня, а також яке саме видалення скасувало певне попереднє додавання ключового слова.

Ще одним аспектом безпеки ШМП є шаблони пошуку та доступу. Шаблон пошуку – це інформація про те, чи створено будь-які два запити за одним ключовим словом чи ні. Шаблон доступу – це інформація про те, які документи містять ключове слово для кожного запиту користувача. Як зазначено в [4], дуже багато ШМП «страждають» від того, що з зазначених вище шаблонів витікає інформація. У симетричних схемах ця проблема ґрунтується на тому, що для генерування лазівки зазвичай використовуються детерміновані алгоритми, тобто для певного ключового слова завжди буде генеруватись однаковий запит.

Т.ч. очевидно, що супротивник може без зусиль встановити, чи мають 2 різні запити одне й те ж саме ключове слово. З іншого боку, є шаблон доступу, який, в свою чергу, може розкрити шаблон пошуку. Якщо шаблон доступу однаковий, то, ймовірно, що два пошукових запити містять однакове ключове слово. В роботі [5] зазначається, що майже у всіх симетричних ШМП витікає шаблон доступу.

Для визначення того, чи є метод ШМП безпечним, використовуються моделі безпеки. У роботі [6] були визначені моделі *IND1-CKA* та *IND2-CKA* (*від англ. indistinguishability against chosen keyword attack*), зазначивши, що врахування лазівок є обов'язковим, так як вони нерозривно пов'язані з безпекою індексів. Перша модель передбачає нерозрізненість проти атак з неадаптивно підібраними ключовими словами, друга – з адаптивно. Обидві моделі гарантують, що ні індекси, ні лазівки не розкривають жодну інформацію про вміст документу та ключові слова, асоційовані з документом (*за виключенням тієї, яку можна вивести з шаблонів пошуку та доступу*).

Проте все зазначене вище стосується тільки симетричних схем шифрування з можливістю пошуку. Що стосується асиметричних ШМП (АШМП), в таких схемах з'являється ще один ключ – відкритий, за допомогою якого генеруються лазівки, отже, безпека лазівок не враховується. Першою роботою стосовно моделі безпеки АШМП стала стаття Дена Боне та інших [7], у якій вони запропонували визначати безпеку асиметричних схем нерозрізненістю двох зашифрованих ключових слів, доки супротивник не має лазівки до цих слів, тобто цю модель також можна назвати *IND-CKA* або *PK-CKA* (від англ. *public key*). Аналогічно 1 версія буде для не адаптивно підібраних ключових слів, а друга – для адаптивно.

3. Конструкція методів шифрування з можливістю пошуку

Розглянемо побудову методів ШМП та криптопримітиви, на яких базуються ці методи. Всі зазначені методи відносяться до симетричного шифрування з можливістю пошуку, а також всі, окрім одного, є динамічними. Такі методи мають перевагу у ефективності та безпеці схем, проте поступаються виразністю запитів, через що найчастіше знаходять використання у NoSQL базах даних. При цьому увага не приділяється методам, що вже були розглянуті та проаналізовані у інших роботах, в яких доведена їх неспроможність чи то з погляду безпеки, чи то з боку ефективності. Наприклад, у роботі [8] було проведено серйозний аналіз великої кількості методів, однак більшість з них виявилися недостатньо ефективними для їх практичного використання. Деякі з зазначених схем є досить повільними при прийнятному рівні безпеки, інші ж працюють досить швидко, проте мають неприйнятний витік інформації.

3.1 Паралельне та динамічне шифрування з можливістю пошуку (PDSSE)

Цей метод (позначимо його як абревіатуру від англійської назви *Parallel and Dynamic Symmetric Seacrable Encryption*, тобто *PDSSE*) розглянуто в роботі [9]. Він заснований на червоно-чорних деревах (ЧЧД). Проте в методі використовуються деяка модифікація ЧЧД: листя дерева зберігають покажчики на документи, а інші вузли – ідентифікатори документів. Також кожен вузол u зберігає бітовий вектор $data_u$ довжиною m біт, де i-ий біт відображає чи наявне ключове слово w_i у його нащадках. Тут встає питання того, що вузли мають по два нащадки (припустимо, що лівий нащадок – це v, а правий – це z), отже, бітовий вектор кожного вузла u визначається як «побітове або» векторів його нащадків, тобто за формулою:

$$data_u = data_v \,\&\, data_z\,.$$

Тоді якщо i-ий біт дорівнює одиниці, то є хоча б один шлях, який приводить до документу d_j, що містить ключове слово w_i. А для знаходження всіх документів, які містять ключове слово, просто необхідно перевіряти вектори нащадків, доки не зустрінемо 0 в i-ії позиції вектора, або не доберемося до листа дерева.

Для проведення оновлення визначається ідентифікатор документа, та операція (додавання чи видалення), яку треба виконати. Після цього визначається частина дерева, яку це оновлення торкнеться, та проводиться перерахунок бітових векторів у вузлах, згідно доданого чи видаленого документу.

На рис. 2 зображено реалізацію методу приведену авторами, в якій є 8 документів та 5 ключових слів. Виходячи з кількості ключових слів, кожен вузол зберігає по два 5-ти бітних вектори, один з яких містить справжню інформацію о ключових словах у своїх нащадках, проте у зашифрованому вигляді. Червоними стрілками позначений пошук 5-го ключового слова, в результаті отримуємо, що 5 ключове слово знаходиться у 3, 6 та 7 документах.

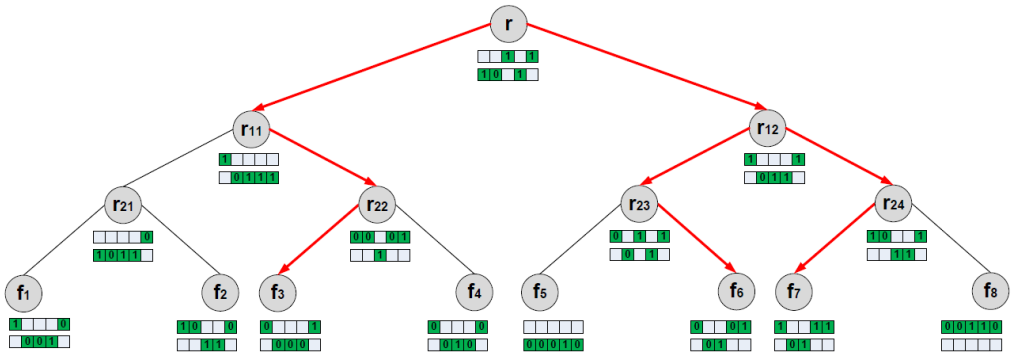


Рис. 2 – Приклад реалізації методу *PDSSE*

3.2 Метод Dyn2Lev

Цей метод визначено у [10]. Він взяв свою назву від принципу роботи: - він є динамічним, а головна ідея полягає у створенні 2 рівнів блоків покажчиків. Базис методу заснований на словнику, ключем у ньому виступають ключові слова, а значенням буде деякий блок довжиною B, що буде містити ідентифікатори записів. Тоді припустимо, що для ключового слова w та бази даних записів DB, існує $|DB(w)|$ записів, що потрібно витягнути з БД, отже, таких блоків буде $\frac{|DB(w)|}{B}$. А останній блок доповнюється до розміру B, якщо це потрібно, щоб їх було неможливо відрізнити.

Таке групування дозволяє значно скоротити кількість необхідних витягнень при пошуку. Але постає питання того, що для різних ключових слів кількість записів, в яких вони при-

сутні, може значно відрізнятись. Тобто для деяких слів, котрим відповідають значні обсяги записів, витягнення все ще може бути неефективним. З іншого боку, якщо B обирається надто великим, то будуть страждати ключові слова, котрим відповідає маленька кількість записів, тому як доповнення до розміру блоку B буде створювати багато надлишкової інформації при доповненні.

Спочатку було запропоновано ввести блок додавати у словник блок не ідентифікаторів, а покажчиків, що посилаються на блоки ідентифікаторів, які зберігаються у зовнішньому масиві «A». Але ця модифікація все ще не вирішує питання щодо наявності як ключових слів з великою кількістю відповідних записів, так і з незначною. З цього приводу планується розрізняти множини на маленькі, середні та великі. Визначається все через розміри блоків: блоки розміром b у словнику та блоки розміром «B» у деякому зовнішньому масиві. Отже, маленькі множини мають кількість записів менше або дорівнює b, середні множини мають кількість записів, що лежить у межах $b < |DB(w)| \leq Bb$, великі – $Bb < |DB(w)| \leq B^2b$. Верхня границя для великих множин дорівнює B^2b, з чого випливає, що необхідно обирати «B» та «b» таким чином, щоб не існувало множини $DB(w)$ потужністю більше ніж B^2b.

Для маленьких множин ідентифікатори будуть зберігатися напряму у словнику, таким чином, не буде потреби у використанні покажчиків. Для середніх будуть створюватися блоки покажчиків, що вказують на відповідні блоки ідентифікаторів. Для великих множин є два рівні (*2 lev*): - у словнику зберігаються покажчики, що вказують на блоки покажчиків, які, в свою чергу, вказують на блоки ідентифікаторів.

Отже, є три способи, розглянемо приклад (*див. табл. 1*), щоб зрозуміти як змінюється кількість витягнень в залежності від підходу для підвищення ефективності. Візьмемо $B = 500$ та $b = 100$.

Таблиця 1 – Кількість витягнень з словника в залежності від підходу

$\|D(w)\|$	Блоки ідентифікаторів	Блоки покажчиків	Два рівня блоків покажчиків
100	1	1	1
200	2	1	1
1100	11	1	1
5100	51	1	1
10100	101	1	1
50100	501	2	1
500100	5001	11	1
5000100	10001	101	1

Таким чином, при застосуванні комбінованого підходу з трьома видами множин, нам завжди необхідно зробити тільки 1 витягнення. Для пошуку клієнт генерує спеціальну позначку на базі ключового слова. За ключем з словника витягується блок, якщо цей блок містить ідентифікатори, то вони відправляються клієнту, якщо ж покажчики, то він використовує покажчики для отримання блоків з масиву A. Якщо ці блоки містять ідентифікатори, то він їх відправляє клієнту, інакше знову використовує позначки, але тепер вже точно отримує блоки ідентифікаторів, які відправляє клієнту.

Для можливості видалення генерується окрема структура даних – множина S_{rev}, що зберігає так звані ідентифікатори відкликання, значення яких отримується від псевдовипадкової функції (ПВФ) з спеціальним ключем ключового слова. Тобто буквального видалення з словника не проводиться. Для можливості додавання генерується ще один словник D^+, що має саму базову конструкцію методу. Якщо треба додати запис, то клієнт відправляє запит

до серверу, що містить ідентифікатор запису і множину ключових слів. Для кожної пари «ідентифікатор/ключове» слово, сервер перевіряє, чи наявний відповідний ідентифікатор відкликання у множині S_{rev}, і якщо наявний, то він його видаляє, а якщо ні – то додає пару до словника D^+.

У зв'язку з можливістю додавання та видалення пошук також дещо змінюється: після перевірки основного словника, проводиться перевірка у словнику для додавання записів, а також перевіряється чи є у множині відповідний ідентифікатору запису ідентифікатор відкликання. Якщо так, то знайдений ідентифікатор прибирається. Отриманий список ідентифікаторів при пошуку відправляється клієнту.

3.3 Логічне симетричне шифрування з можливістю пошуку

Наступний метод визначено у роботі [11], в якій зосереджується увага на виразності методу. Два попередньо розглянуті методи дозволяють по одному ключовому слову отримати певний перелік ідентифікаторів документів або записів БД, що містять це ключове слово. Проте іноді зручним є пошук тих документів чи записів, що містять в собі деяку множину ключових слів. Перед тим, як перейти до безпосереднього розгляду методу, визначимо декілька понять, важливих для розуміння реалізації методу.

Мультипвідображення (МВ) або *multimap* (*MM*) – це відображення, яке для кожного ключа зіставляє більше одного значення. Зокрема, у схемі використовується кортеж значень.

Фільтр Блума – це ймовірнісна структура даних, яку запропонував Бартон Блум у 1970 році [12]. Ця структура даних має дві операції: додавання елемента до множини та перевірка наявності елементу у множині. Імовірність структури полягає у тому, що можливі відповіді на питання наявності елементу – це «можливо» або «ні». Тобто фільтр Блума припускає хибно позитивні відповіді, але хибно негативні неможливі. Ідея полягає у тому, що існує бітовий масив розміру m та k геш-функцій, що видають значення від 0 до $m-1$. Якщо треба додати деякий елемент, то від нього розраховуються всі геш-функції, та відповідні розрахованим значенням біти масиву встановлюються в одиницю. А при перевірці також розраховуються значення геш-функцій та витягуються відповідні значення з масиву, і, якщо всі значення дорівнюють одиниці, то елемент може бути присутнім, а якщо маємо хоча б один нуль, то елементу точно немає у структурі даних.

Онлайн шифр – це блочний шифр, визначений у [13]. Його особливістю є те, що шифрування може проводитись в онлайн або у потоковому режимі. Такі шифри мають наступну вимогу: якщо є деяке повідомлення $M = M_1||M_2|| \dots ||M_l$ та відповідний йому шифртекст $C = C_1||C_2|| \dots ||C_l$, то для розрахування деякого блоку шифртексту C_i достатньо знати блоки відкритого тексту з 1 до i. Таким чином, блок шифртексту C_i не залежить від блоків відкритого тексту $M_{i+1}, \dots, M_l$.

Inclusion-exclusion principle (*IEP*), тобто принцип включень-виключень. Це техніка, що дозволяє визначати потужність об'єднань кінцевого числа кінцевих множин, завдяки ж цьому можна отримати множину унікальних елементів цього об'єднання. Принцип, як випливає з назви, полягає у почерговому застосуванні включення та виключення: спочатку включаються всі множини; після цього виключаються попарні перетини; далі включаються потрійні перетини; після виключаються четверні перетини і так далі до n-го порядку, де n – кількість множин.

Розглянемо приклад *IEP* для випадку перетину множин A, B і C, що наведено на рис. 3. При підсумовуванні трьох множин, двічі враховуються елементи, що наявні в двох множинах, та тричі враховуються елементи, що наявні в усіх трьох множинах. Тому, після підсумовування всіх множин, необхідно виключити подвійні перетини. Таких перетини три, і після

їх віднімання елементи, що наявні рівно у двох множинах, будуть представлені у єдиному екземплярі.

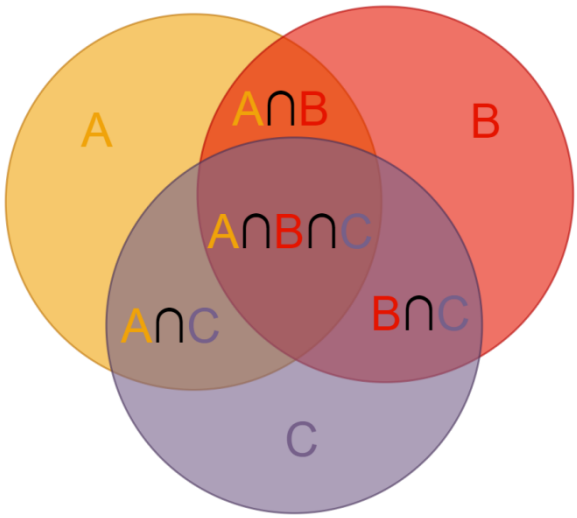


Рис. 3 – Перетин трьох множин

Але, як зазначалось вище, елементи, що наявні у всіх трьох множинах, були враховані тричі, проте на минулому кроці їх тричі відняли, тобто тепер їх не залишилось. Отже, необхідно знову включити їх до загального результату. Таким чином, було продемонстровано застосування *IEP* для отримання унікальних елементів об'єднання множин A, B і C.

Перейдемо до визначення головної конструкції методу – *IEX* (*inclusion-exclusion encryption*), назва якої відсилає на принцип «включень-виключень» (*IEP*). У методі визначаються теги, котрі обчислюються, як шифрування пари ідентифікатор та ПВФ від конкатенації ідентифікатору і ключового слова *(див. (1))*. Т.ч., кожен тег має унікальне значення, навіть якщо його згенеровано для одного ключового слова, або для одного ідентифікатору запису (1):

$$tag_{id} = SKE.Enc_{K_1}\left(id; F_{K_2}(id||w)\right). \quad (1)$$

Як базис методу використовуються три структури даних: два мультивідображення та словник. Перше мультивідображення називається глобальним, в якості ключа приймає ключове слово, а в якості значення – теги тих записів, що містять це ключове слово.

Розглянемо приклад такого мультивідображення на прикладі рис. 4.

MMg

w_1	tag_1, tag_4, tag_{10}
w_2	tag_3, tag_5
w_3	tag_1, tag_9
w_4	tag_6
w_5	tag_4, tag_5, tag_7
w_6	tag_1, tag_4, tag_8

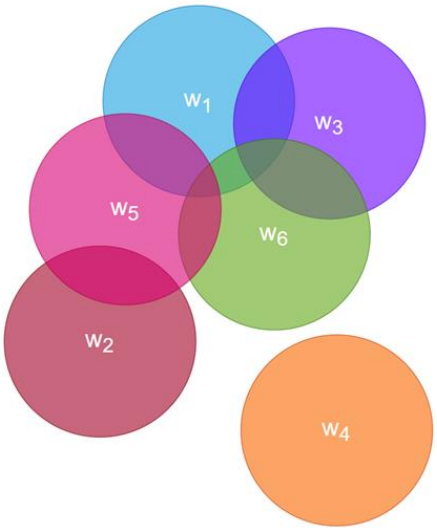


Рис. 4 – Приклад побудови глобального мультивідображення

Також для кожного ключового слова генерується локальне мультивідображення, як ключ приймаються ключові слова $v \in co(w)$. В множині $co(w)$ зберігаються ключові слова, що наявні у записі/документі разом з ключовим словом w. Значення у цьому мультивідображенні – «теги записів», в яких наявні обидва ключових слова. При цьому ці локальні МВ записуються у третю структуру даних – «словник», ключем виступає ключове слово w. Приклади «словника» та локального МВ наведені нижче, на рис. 5.

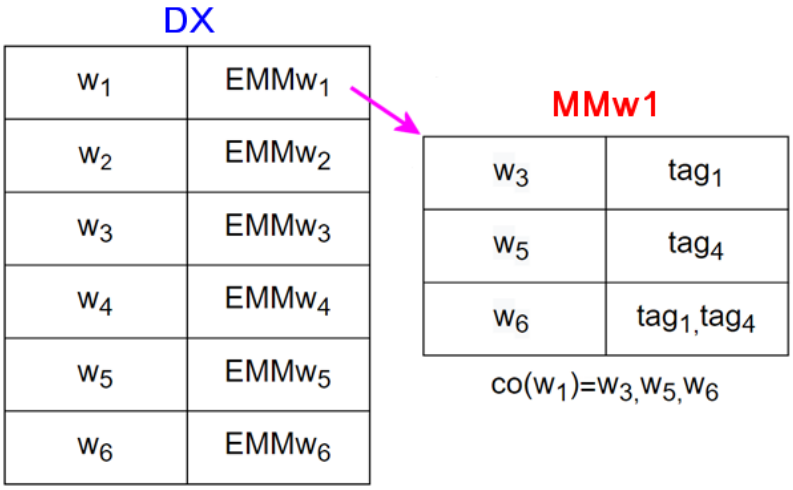


Рис. 5 – Приклад побудови словника та локального мультивідображення

Розберемо як проходить пошук, спочатку для диз'юнктивного виразу. Нехай маємо запит вигляду $w_1 \vee \ldots \vee w_q$. Результат такого запиту отримуємо за принципом включень-виключень: для кожного ключового слова (окрім останнього) спочатку звертаємось до глобального МВ, отримуємо всі теги та додаємо до загальної множини. Далі перебираємо локальні мультивідображення з тими ключовими словами, що йдуть у запиті після поточного, та прибираємо з множини всі теги, що наявні у значеннях цих локальних МВ. Для останнього у запиті ключового слова додаються всі теги. Таким чином, ми отримуємо відповідь на запит без дублювання тегів.

Якщо на вхід подається довільний логічний запит, то його завжди можна представити у кон'юнктивній нормальній формі, тобто $\Delta_1 \wedge \ldots \wedge \Delta_l, \Delta_i = w_{i,1} \vee \ldots \vee w_{i,q}$. А набір кон'юнкцій – це знаходження перетину деякої кількості множин, кожна з котрих може бути представлена деяким набором диз'юнкцій Δ_i. Причому треба відмітити, що перетин цих множин завжди буде підмножиною Δ_1. По суті, на початку маємо множину елементів Δ_1, після цього видаляємо всі елементи, котрих немає у Δ_2, далі всі елементи, котрих нема у Δ_3 і так до Δ_l.

У *динамічному* варіанті відмінність полягає у тому, що реалізації глобального мультивідображення та словника повинні підтримувати оновлення, локальне МВ залишається статичним. Сам процес оновлень теж майже повністю співпадає з цими реалізаціями, окрім генерації відповідних лазівок.

Слід відзначити, що описана схема є деякою надбудовою над іншими схемами ШМП: у якості реалізації МВ використовуються саме симетричні методи шифрування з можливістю пошуку. Хоча *IEX* і дозволяє виконувати логічні запити, його використання значно збільшує займаний простір. Тому автори запропонували компактний метод ШМП – *ZMF*, назва якого присвячена конструкціям на яких він базується - метод *Z-IDX* та *Matryoshka filter*.

Ця схема використовується як реалізація для локальних МВ. Вона має лінійну складність пошуку, проте значно меншу просторову складність, тому що базується на фільтрах Блума. Точніше, для методу розроблена нова структура даних – фільтр-матрьошка, який базується на фільтрах Блума. Визначимо коротко сутність: мається декілька множин з різною кількістю елементів (*наприклад, кортежи тегів у локальному МВ*). Серед цих множин ми визначаємо найбільшу та генеруємо для неї геш-функції та фільтр Блума. Для всіх інших множин фільтри Блума будуть виводитись з цього максимального фільтру (*тобто вони вкладені в максимальний фільтр*), так само як і геш-функції, в залежності від потужності кожної множини. Для безпеки даних, фільтр додається за модулем 2 з маскою. Значення цієї

маски повинні залежати від конкретної бітової позиції фільтру *(інакше це порушить коректність при перевірці наявності елементу*), а також від конкретного фільтра Блума. Безпосередньо маска генерується наступним чином: від елемента розраховується ПВФ, від отриманого значення обчислюється геш-функція. Кожен біт геш-значення доповнюється нулями до блоку розміром B, отриманий рядок шифрується за допомогою онлайн шифру, після чого усікається в залежності від розміру множини. Отримане значення комбінується з ідентифікатором множини (*фільтру Блума*) та подається до випадкового оракулу. Вихідне значення і є маскою для деякого біту фільтру.

3.4 Метод Σοφος

Σοφος – це симетрична схема шифрування з можливістю пошуку, що відповідає поняттю прямої секретності, запропонована у [14]. Як базис схеми використовуються відображення. Перед початком розгляду схеми визначимо основний її криптопримітив – перестановку лазівки або *trapdoor permutation (TDP).*

TDP – це така перестановка π над деякою множиною D, що за допомогою відкритого ключа PK, π може бути обчислена без будь-яких проблем за прийнятний час, але інверсія π^{-1} може бути ефективно обчислена лише за допомогою секретного ключа SK.

Розглянемо основну ідею методу: маємо деяку множину ключових слів W, для кожного $w \in W$ існує відповідний список індексів документів, що містять це ключове слово – $(ind_0, \dots, ind_{n_w-1})$, де $n_w = |DB(w)| - d_w$ – це кількість документів, що містять ключове слово w, за винятком видалених. Кожний елемент цього списку шифрується та записується у логічному місці, яке виводиться від ключового слова w та номеру ідентифікатора документа – c. Це логічне місце позначається як $UT_c(w)$ та, по суті, є токеном оновлення, відповідно, якщо клієнт захоче додати новий документ до БД, який містить ключове слово w, то йому треба зашифрувати індекс та розрахувати логічну позицію для нього, тобто токен оновлення.

Для пошуку клієнт повинен згенерувати токен пошуку $ST(w)$, який дозволяє серверу згенерувати відповідний токен оновлення за допомогою геш-функції, цей токен дозволяє витягнути зашифрований індекс документу. При цьому, мається n_w індексів $(c = n_w - 1)$, що відповідають ключовому слову w, які необхідно отримати при пошуку, але таким чином, щоб сервер не зміг отримати індекси оновлення $UT_i, i > c$, тобто логічні місця, де будуть зберігатися додані у майбутньому ідентифікатори.

Тут вступає у гру саме *TDP*, що зазначалась вище: за допомогою токену пошуку $ST_i(w)$ сервер зможе розрахувати $ST_{i-1}(w)$, використовуючи відкритий ключ PK, але для розрахунку $ST_{i+1}(w)$ знадобиться секретний ключ.

Для додавання можливості видалення елементів пропонується створити ще один екземпляр описаної схеми. А пошук буде різницею алгоритмів пошуку цих двох екземплярів.

3.5 Метод Fides

Розглянемо підхід та схему на його основі, що були запропоновані у роботі [15]. Підхід описує, як з звичайної схеми симетричного ШМП Σ (англ. *Symmetric Searchable Encryption* або *SSE*) отримати схему, що відповідає поняттю зворотної секретності. Ця схема є деякою «надбудовою» над іншими схемами, вона пропонує зберігання не індексів, а комбінації індексу та операції *(див. (2))*, після чого ця комбінація шифрується ключем відповідного ключового слова w.

$$E_{K_w}(ind, op) \tag{2}$$

В іншому, базова схема Σ працює як зазвичай. Проте є декілька важливих моментів щодо зазначеної схеми: - при пошуку сервер не зможе ідентифікувати необхідні записи за ключовим словом, оскільки він не бачить індексів (тому що вони зашифровані). З цього приводу

вводиться ще один раунд протоколу: після того, як клієнт отримає (2), він розшифровує цю структуру та отримує індекси, які відсилає серверу, після чого вже проводиться безпосередній витяг документів.

Тут рішення досить просте: отримуючи відповідь, клієнт прибирає всі індекси, що були видалені, та відправляє решту серверу у відкритому вигляді, але, поряд з цим, він відправляє ці ж самі індекси, зашифровані новим ключем. Таким чином, сервер зможе видалити всі старі індекси, а на заміну додасть нові, утримуючу базу даних в актуальному стані.

Застосовуючи описані вище операції до різних *SSE* схем, можна отримати різні екземпляри *BP*- надійної схеми. *Fides* базується на схемі *Σοφος*.

3.6 Метод Diana

Метод отримується на базі фреймворку *FS-RCPRF* (*forward secure scheme based on the range-constrained pseudorandom function*), то спочатку визначимо його. В цьому фреймворку використовується обмежена псевдовипадкова функція (ОПВФ), яку було паралельно розроблено та представлено у роботах [16] та [17]. Ця ПВФ особлива тим, що асоціюється з сімейством логічних схем $\mathcal{C}$. Для кожної з схем цього сімейства $C \in \mathcal{C}$ мається обмежений ключ K_C, який можна обчислити за допомогою майстер-ключа ПВФ. Цей обмежений ключ дозволяє проводити обчислення тільки за такими значеннями x, для яких $C(x) = 1$.

Ідея полягає у тому, щоб генерувати токени оновлення для ключового слова w за допомогою ОПВФ у режимі лічильника, який збільшується після кожного додавання ідентифікатора, що містить задане ключове слово. Для пошуку клієнту потрібно відправити серверу ключ для ОПВФ, який дозволяє проводити обчислення значень від 0 до $n_w - 1$.

Для створення *Diana* використовується ОПВФ, яка утворюється з псевдовипадкової функції *GGM*, що базується на бінарному дереві [18]. Ця ОПВФ була побудована у [17] та отримала назву найкраще покриття діапазону або *Best Range Cover (BRC)*. Розглянемо принцип її роботи. Нехай мається деякий генератор псевдовипадкових чисел $G: \{0,1\}^{\lambda} \to \{0,1\}^{2\lambda}$, тоді позначимо результат від деякого випадкового початкового значення k як $G(k)$, а першу та другу його частину як $G_0(k)$ та $G_1(k)$ відповідно. Тоді ПВФ GGM від деякого цілого числа x довжиною n біт буде визначатись як:

$$F_K(x) = G_{x_0}\left(\dots\left(G_{x_n-1}(K)\right)\right).$$

Таким чином, листя дерева у *GGM* – це результат функції *F*, вона може бути зіставлена з вихідним значенням x, а також частковим обчисленням цієї функції (*тобто, з* $l < n$ *біт числа* x). Наведемо простий приклад дерева ПВФ з [17].

З рисунку 6 зрозуміло, яким чином отримуються значення у листках та взагалі у вузлах: записуються значення від кореня до вузла. Якщо потрібно пройти до листа з позначкою 3, то обчислюємо $G_1\left(G_1\left(G_0(G_0(k))\right)\right)$. Також можемо проводити асоціювання з частковими значеннями ПВФ: $G_1\left(G_1(f_k(00))\right)$ або $G_1(f_k(001))$. При цьому для деякого діапазону, наприклад, [2-7], як на рисунку 6, існує множина таких часткових значень, які являють собою піддерева. Для обраного діапазону, це $f_k(001)$ з глибиною 1, та відповідно $f_k(01)$, з глибиною 2 (*глибина до листя*). Задача *BRC* як раз і полягає в тому, щоб знайти мінімальну кількість таких часткових значень.

У методі *Diana* треба обмежити ПВФ на проміжку $[0, c]$, для цього будуть генеруватися такі вузли, що покривають цей діапазон, причому не містять шляху до листя, що в заданий діапазон не входять. В іншому, все йде згідно алгоритму *FS-RCPRF*.

В описаному вище методі можливе лише додавання. Уможливити видалення можна подібно до *Σοφος*, тобто створити два екземпляри SE-схеми з підтримкою прямої секретності, одна з яких буде для додавання, а інша – для видалення. Результат отримується як різниця результатів двох схем, але, якщо ці операції буде проводити сервер, то він буде отримувати інформацію про видалені записи. Пропонується інший підхід: при додаванні додаються записи виду (w, ind) до першого екземпляру. Поряд з цим передається пара значень $(F'(K_w, (w, ind)), Enc_{K'}(c))$, де F' це деяка ПВФ, а Enc – деяка CPA-надійна схема шифрування. Сервер приймає цю пару та заносить до спеціального відображення. При видаленні додається до другого екземпляру пара значень $(w, F'(K_w, (w, ind)))$.

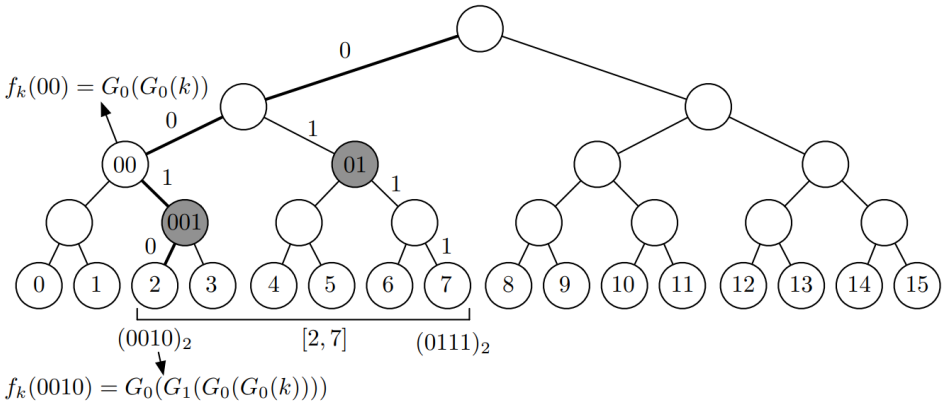


Рис. 6 – Приклад деревовидної ПВФ *GGM*

При пошуку даних спочатку робиться пошук у другому екземплярі, сервер витягує відповідні ключовому слову теги $F'(K_w, (w, ind))$, що відповідають видаленим записам. Далі він застосовує ці теги для витягнення $Enc_{K'}(c)$ з відображення, та відправляє ці зашифровані ідентифікатори клієнту. Клієнт розшифровує ідентифікатори та дізнається про те, які записи були видалені. Маючи цю інформацію, клієнт з початкового діапазону $[0, c]$ генерує декілька діапазонів, які не включають видалені записи. Якщо маємо видалені ідентифікатори $x_1, \dots, x_n$, то діапазони мають вигляд $[0, x_1 - 1], [x_1 + 1, x_2 - 1], \dots, [x_n + 1, c]$. Варто зауважити, що цей метод припускає, що один і той же індекс не додається двічі. Якщо його було видалено, то його не можна додати знов.

4. Аналіз методів шифрування з можливістю пошуку

Для зручності сприйняття проведеного аналізу зведемо всі дані про методи до табл. 2 за показниками часової (*обчислювальної*), комунікаційної та просторової складностей, а загальну характеристику схем зведемо до табл. 3. Позначки таблиць мають наступні значення:

- m – загальна кількість ключових слів;
- n – загальна кількість записів/документів;
- p – кількість процесорів;
- $DB(w)$ – кількість записів, що містять ключове слово w;
- d_w – кількість видалених записів, що містять ключове слово w;
- q – це кількість ключових слів у логічному запиті;
- $co(w)$ – множина ключових слів, що наявні у документах разом з w;
- $query$ – логічний запит;
- l – кількість диз'юнкцій у запиті;
- $M = max_{i \in [q]} |DB(w_i)|$ – це потужність максимальної множини документів для деякого ключового слова w_i;

Таблиця 2 – Порівняння розглянутих методів шифрування з можливістю пошуку за складністю

НАЗВА	ЧАСОВА СКЛАДНІСТЬ		КОМУНІКАЦІЙНА СКЛАДНІСТЬ		ПРОСТОРОВА СКЛАДНІСТЬ	
	Пошук	***Оновлення***	***Пошук***	***Оновлення***	***Клієнт***	***Сервер***
PDSSE	$O(\frac{\lvert DB(w)\rvert}{p}\log n)$	$O(\frac{m}{p}\log n)$	$O(\lvert DB(w)\rvert)$	$O(\lvert W_{id}\rvert + m\log n)$	$O(1)$	$O(m*n)$
Dyn2Lev	$O(\frac{\lvert DB(w)\rvert + d_w}{p})$	$O(1)$	$O(\lvert DB(w)\rvert)$	$O(\lvert W_{id}\rvert + m\log n)$	$O(1)$	$O(n)$
BIEX-ZMF	$O\left(q^2(M + l\lvert DB(\Delta_1)\rvert)\right)$	–	$O(\lvert DB(query)\rvert + lq)$	–	$O(1)$	$O\left(\sum_w(\lvert DB(w)\rvert + int_1)\right)$
BIEX-2Lev	$O\left(q^2(M + l\lvert DB(\Delta_1)\rvert)\right)$	–	$O(\lvert DB(query)\rvert + lq)$	–	$O(1)$	$O\left(\sum_w(\lvert DB(w)\rvert + int_2)\right)$
Σοφος	$O(\lvert DB(w)\rvert)$	$O(1)$	$O(\lvert DB(w)\rvert)$	$O(1)$	$O(m*\log n)$	$O(\sum_w(\lvert DB(w)\rvert + d_w))$
Fides	$O(\lvert DB(w)\rvert)$	$O(1)$	$O(\lvert DB(w)\rvert + d_w)$	$O(1)$	$O(m*\log n)$	$O(\sum_w(\lvert DB(w)\rvert))$
Diana$_{del}$	$O(\lvert DB(w)\rvert)$	$O(\log\lvert DB(w)\rvert)$	$O(\lvert DB(w)\rvert + d_w\log\lvert DB(w)\rvert)$	$O(1)$	$O(m*\log n)$	$O(\sum_w(\lvert DB(w)\rvert + d_w))$

Таблиця 3 – Порівняння розглянутих методів шифрування з можливістю пошуку

НАЗВА	Базис	Криптопримітиви	Безпека	FP	BP	Витік
PDSSE	Червоно-чорне дерево	ПВФ, геш-таблиця, геш-функція, CPA-надійна схема шифрування	IND2-CKA^{RO}	–	–	Шаблон пошуку, шаблон доступу, кількість ключових слів, розмір БД, ідентифікатори документу, розмір документів
Dyn2Lev	Словник, кортеж	ПВФ, CPA-надійна схема шифрування	IND2-CKA^{RO}	–	–	Шаблон пошуку, шаблон доступу, шаблон оновлення, розмір БД, розмір зовнішнього масиву, розмір блоків
BIEX-ZMF	Базується на *ZMF*, фільтр-матрьошка, мультивідображення, словник	ПВФ, ПВП, фільтр Блума, онлайн-шифр, CPA-надійна схема шифрування	IND2-CKA^{RO}	–	–	Шаблон пошуку, шаблон доступу, розмір БД, загальний обсяг всіх ЛМВ у словнику, перетини множин
BIEX-2Lev	Базується на *2Lev*, мультивідображення, словник, кортеж	ПВФ, ПВП, CPA-надійна схема шифрування	IND2-CKA^{RO}	–	–	Шаблон пошуку, шаблон доступу, розмір БД, загальний обсяг всіх ЛМВ у словнику, перетини множин + виток з *2Lev*
Σοφος	Відображення	ПВФ, RSA TDP, геш-функція, CPA-надійна схема шифрування	IND2-CKA^{RO}	+	–	Шаблон пошуку, вся історія взаємодій з ключовим словом: тип операції, час та ідентифікатор документу
Fides	Базується на *Σοφος*, відображення	ПВФ, RSA TDP, геш-функція, CPA-надійна схема шифрування	IND2-CKA^{RO}	+	II	Шаблон пошуку, кількість оновлень за ключовим словом та час, коли вони були виконані
Diana$_{del}$	Відображення	ПВФ, ОПВФ BRC, геш-функція, CPA-надійна схема шифрування	IND2-CKA^{RO}	+	III	Шаблон пошуку, кількість оновлень, їх час, ідентифікатор документу, історія оновлень цього документу, яке саме видалення скасувало додавання запису

- W_{id} – множина ключових слів, що передається при оновленні у методах *PDSSE* та *Dyn2Lev*;
- $int_1 = \max_{v \in co(w)} |DB(w) \cap DB(v)|$ – потужність найбільшої множини у локальному МВ;
- $int_2 = \sum_{v \in co(w)} |DB(w) \cap DB(v)|$ – загальна кількість локальних МВ.

5. Висновки

В роботі було розглянуто процес шифрування з можливістю пошуку, проведено його класифікацію та виділено характерні особливості. Здійснено розгляд та аналіз методів шифрування з можливістю пошуку.

На підставі проведеного аналізу методу *PDSSE* була визначена його відносно висока складність серед більш сучасних методів ШМП, яка підвищує складність пошуку для великих баз даних, навіть при незмінній кількості записів, що містять шукане ключове слово. Проте цей недолік можна компенсувати можливістю паралельного обчислення. Іншим, більш серйозним недоліком є його просторова складність: хоча клієнт зберігає тільки секретні ключи, сервер на кожний елемент дерева зберігає *m*-бітний вектор, тобто по біту на кожне ключове слово. Навіть середні за розміром бази даних можуть мати досить багато ключових слів, через що *PDSSE* займе дуже багато місця на диску. Цих 2-х недоліків достатньо, щоб говорити про неефективність практичного застосування цього методу.

Метод *Dyn2Lev*, завдяки використанню словника, здатен проводити операції оновлення за постійний час. Складність пошуку та займаний простір теж мають досить ефективні значення. Серед недоліків схеми можна відмітити зростаючу кількість видалених записів, які не видаляються насправді, з чого випливає необхідність в періодичному перешифруванні всієї БД. При цьому, метод використовує досить багато покажчиків, тому для отримання ідентифікаторів також витрачається зайвий час, тому реальна швидкість методу, хоча й краща за *PDSSE* та є непоганою на практиці, все ще поступається більш сучасним методам.

Схема *IEX*, як і її логічна та динамічна модифікації, відрізняється від всіх інших розглянутих схем, оскільки являє собою надбудову над іншими схемами для досягнення більшої виразності запитів. Її ефективність та безпека здебільшого базуються на підлеглій схемі, що використовується у якості мультивідображення, тобто нічого не заважає реалізувати вказані схеми над якоюсь більш ефективнішою та безпечнішою схемою.

Серед запропонованих авторами схем *BIEX-ZMF* та *BIEX-2Lev* вибір йде або убік компактності отриманої зашифрованої бази даних (*ZMF*), або убік швидкості роботи та комунікаційної складності (*2Lev*). Отже, рішення про використання *IEX* залежить від необхідності виконувати логічні запити до БД у конкретному проекті, враховуючи на збільшення часу пошуку, комунікаційної та просторової складності.

Метод *Σοφος*, порівняно з раніше розглянутими методами, характеризується кращою часовою та комунікаційною складністю. Серед його недоліків можна виділити необхідність клієнта зберігати значення лічильників для кожного ключового слова. При цьому, незважаючи на часову складність, реальний час пошуку схеми «страждає» через використання перестановки лазівки *RSA*, але він все одно значно кращий чим той же *Dyn2Lev*.

За допомогою фреймворка *FS-RCPRF* на базі *Σοφος* було визначено метод *Fides*, який відповідає не тільки поняттю прямої секретності, але й зворотної другого рівня, при цьому має майже однакові показники складності. Тому використання *Σοφος* не є доцільним, тому що *Fides* має більш кращі показники безпеки.

Модифікація схеми *Diana*, що дозволяє проводити видалення, базується повністю на симетричних криптопримітивах, що у поєднанні з оптимальною часовою складністю робить її найшвидшою з розглянутих схем. При цьому вона відповідає поняттям прямої секретності

та зворотної секретності третього рівня, а що головне – працює значно швидше всіх інших розглянутих методів. Хоч вона і розкриває більше інформації, ніж *Fides*, але швидкість пошуку розрізняється в десятки разів.

Отже, серед розглянутих в межах цієї роботи, методів ШМП, кращим та більш ефективнішим є метод *Diana*. Якщо безпека для проекту є надто важливою, то слід звернути увагу на *Fides*, поступившись швидкістю роботи.

Слід враховувати що, розглянуті методи перш за все підходять для *NoSQL* баз даних, оскільки мають дуже «слабку» виразність запитів. Виразність запитів можна покращити реалізацією *IEX* або *BIEX* «поверх» основної схеми для створювання логічних запитів, але це погіршить всі інші характеристики використовуваного методу.

Список літератури

[1] Paverd, A., Martin, A., & Brown, I. (2014). Modelling and automatically analysing privacy properties for honest-but-curious adversaries. *Tech. Rep*.

[2] Stefanov, E., Papamanthou, C., & Shi, E. (2013). Practical dynamic searchable encryption with small leakage. *Cryptology ePrint Archive*.

[3] Bost, R., Minaud, B., & Ohrimenko, O. (2017, October). Forward and backward private searchable encryption from constrained cryptographic primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1465-1482).

[4] Liu, C., Zhu, L., Wang, M., & Tan, Y. A. (2014). Search pattern leakage in searchable encryption: Attacks and new construction. *Information Sciences*, 265, 176-188.

[5] Islam, M. S., Kuzu, M., & Kantarcioglu, M. (2012, February). Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In *Ndss* (Vol. 20, p. 12).

[6] Curtmola, R., Garay, J., Kamara, S., & Ostrovsky, R. (2006, October). Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security* (pp. 79-88).

[7] Boneh, D., Crescenzo, G. D., Ostrovsky, R., & Persiano, G. (2004, May). Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques* (pp. 506-522). Springer, Berlin, Heidelberg.

[8] Bösch, C., Hartel, P., Jonker, W., & Peter, A. (2014). A survey of provably secure searchable encryption. *ACM Computing Surveys (CSUR), 47*(2), 1-51.

[9] Kamara, S., & Papamanthou, C. (2013, April). Parallel and dynamic searchable symmetric encryption. In International *conference on financial cryptography and data security* (pp. 258-274). Springer, Berlin, Heidelberg.

[10] Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M. C., & Steiner, M. (2014). Dynamic searchable encryption in very-large databases: Data structures and implementation. *Cryptology ePrint Archive*.

[11] Kamara, S., & Moataz, T. (2017, April). Boolean searchable symmetric encryption with worst-case sub-linear complexity. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 94-124). Springer, Cham.

[12] Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, *13*(7), 422-426.

[13] Bellare, M., Boldyreva, A., Knudsen, L., & Namprempre, C. (2001, August). Online ciphers and the hash-CBC construction. In *Annual International Cryptology Conference* (pp. 292-309). Springer, Berlin, Heidelberg.

[14] Bost, R. (2016, October). ∑οφος: Forward secure searchable encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1143-1154).

[15] Bost, R., Minaud, B., & Ohrimenko, O. (2017, October). Forward and backward private searchable encryption from constrained cryptographic primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1465-1482).

[16] Boneh, D., & Waters, B. (2013, December). Constrained pseudorandom functions and their applications. In *International conference on the theory and application of cryptology and information security* (pp. 280-300). Springer, Berlin, Heidelberg.

[17] Kiayias, A., Papadopoulos, S., Triandopoulos, N., & Zacharias, T. (2013, November). Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 669-684).

[18] Goldreich, O., Goldwasser, S., & Micali, S. (1986). How to construct random functions. *Journal of the ACM (JACM), 33*(4), 792-807.

Received: August 2022. Accepted: October 2022.

Authors:
Teimur Makhmudov, CSD Student, V.N. Karazin Kharkiv National University, Ukraine.
E-mail: timkin552364964@gmail.com
Vitalii Yesin, Doctor of Engineering Sciences, Professor, Department of Security of Information Systems and Technologies, V.N. Karazin National University, Kharkiv, Ukraine.
ORCID ID https://orcid.org/0000-0003-1977-7269

E-mail: v.i.yesin@karazin.ua

Analysis of data search methods in cryptographically protected databases.

Abstract. The issue of compliance with data security, namely their confidentiality and integrity, generally being resolved through the use of appropriate cryptographic primitives, taking into account the development of computing power (and/or computational complexity of algorithms). But, in connection with the specific method of storage (in the cloud), the question arises of the effectiveness of the search for the necessary information. The problem considered in this paper is that encryption makes it impossible for an attacker to access data without access key, but deprives the legal owner of the data, of the ability to search for this information.

The article reviewe several encryption methods with searchable. For each of them algorithms given, examples of the use of these methods, explanatory figures and tables were provided. The considered methods are symmetric and dynamic, due to which they are effective and have a relatively high level of security, but low query expressiveness, which is why they are most used in NoSQL databases. The analysis was conducted to evaluate the complexity and level of security of the methods, and the performance of practical implementations was also considered. It was concluded that, conclusions were made about the feasibility of using one or another searchable encryption method in practice, and recommendations were made regarding the combination of the described methods to obtain expected results.

Keywords*:* searchable encryption, keyword, index, dynamic symmetric searchable encryption, search token, trapdoor.

UDC 004.67(075.8) DOI: 10.26565/2519-2310-2022-2-02

MODELING STEGANOCONTENT EXTRACTION ATTEMPTS WITH DIFFERENT LENGTHS STACK SAMPLING SERIES OF IMAGES BLOCKS

Honcharov Mykyta, Pavlova Larysa, Lesnaya Yulia

V.N. Karazin National University, Kharkiv, Ukraine
worldxdark@gmail.com, l.v.pavlova@karazin.ua, xa12284109@student.karazin.ua

Received: August 2022. Accepted: September 2022

Abstract: *The results obtained by using different lengths of sample stacks of runs in simulating the attempts of unauthorized extraction (attack) of steganocontent "protected" by implementing the mechanism of inter-block multiplexing of the parameters of the run lengths of image blocks have been considered in the article. The relationship between the parameters of processing the content (namely, halftone images) and the number of series, as well as the combinatorics of the component elements of the obtained pairs of series parameters, which are the objects of inter-block multiplexing has been demonstrated. It is concluded that the simultaneous use of 2-level data multiplexing significantly extends the capabilities to withstand content attack attempts. It has been found that the use of blocks of higher dimensionality, significantly reduces the role current parameters of the series reliance (base) blocks, in breaking the structure of the original images (those. original content). It is noted that use of two levels of multiplexing of output data at once significantly increases the resistance of the content to attempts at its unauthorized extraction, leading to large distortions in the attacked image, in case of incorrect selection of the active processing parameters.*

Keywords*: run-lengths encoding; steganography; content; hacking; stack.*

1. Introduction

This paper presents the results of modeling the procedures of adapting the method of run-length encoding to implement inter-block multiplexing of steganocontent data, as the main method of preventing the illegitimate extraction of data (*in this case image-content*) from a steganocontainer. These experimental results have been obtained as the part of the research aimed at developing the general concept of a low-resource hybrid steganographic algorithm [1-2]. It is important to emphasize that at this stage of modelling preliminary smoothing of the original images has not performed, which slightly increases the total amount of series in the original (base) image-content array. However, in the current prototype of the algorithm various methods of smoothing the original images are used at the stage of data preprocessing, which allows us to obtain the required result from the number of blocks of identical content when the certain criteria for visual detecting distortions is given. Halftone images of three different types, where the main difference is the characteristic values of the probability of brightness gradient between adjacent image elements, have been used as test data samples [3].

2. Main part

In order to analyze the obtained effects, a simplified version of the inter-block multiplexing of data which is limited by the combinatorics of two elements of the composite key of the data extractor (*the number of blocks and the run lengths*) has been used. Furthermore, to simulate the countering of hacking attempts a simplified version of the masks of the inter-block multiplexing of data for two stacks of different size has been implemented. In other words, during the simulation the attacker is assumed to determine the method of scanning the series and the effective parameter of stack length (*runs sampling base*) correctly, but to be mistaken in determining the current parameters of the displacement of base block (***BB***) and the lengths series BB. The consequences of such an attack are well identified by the vertical "*tracks*" of blocks of different brightness in the background areas of the test image in Fig. 1-2. Thus, if that effect is noticeable when a demo stack of a small length is used, it will be even more noticeable for a wider base.

In the first case, four series were used as a stack, therefore a base of mutual permutations of runs parameters was small [4]. In the second case, the length of the sampling stack was equal to the total number of formed series BB, and the mutual multiplexing of series parameters was carried out between its two halves (*half-stacks*), as well as within each of them. In both cases, the *"destruction"* of the initial pairs of the parameters was carried out: BB – the BB series length [5].

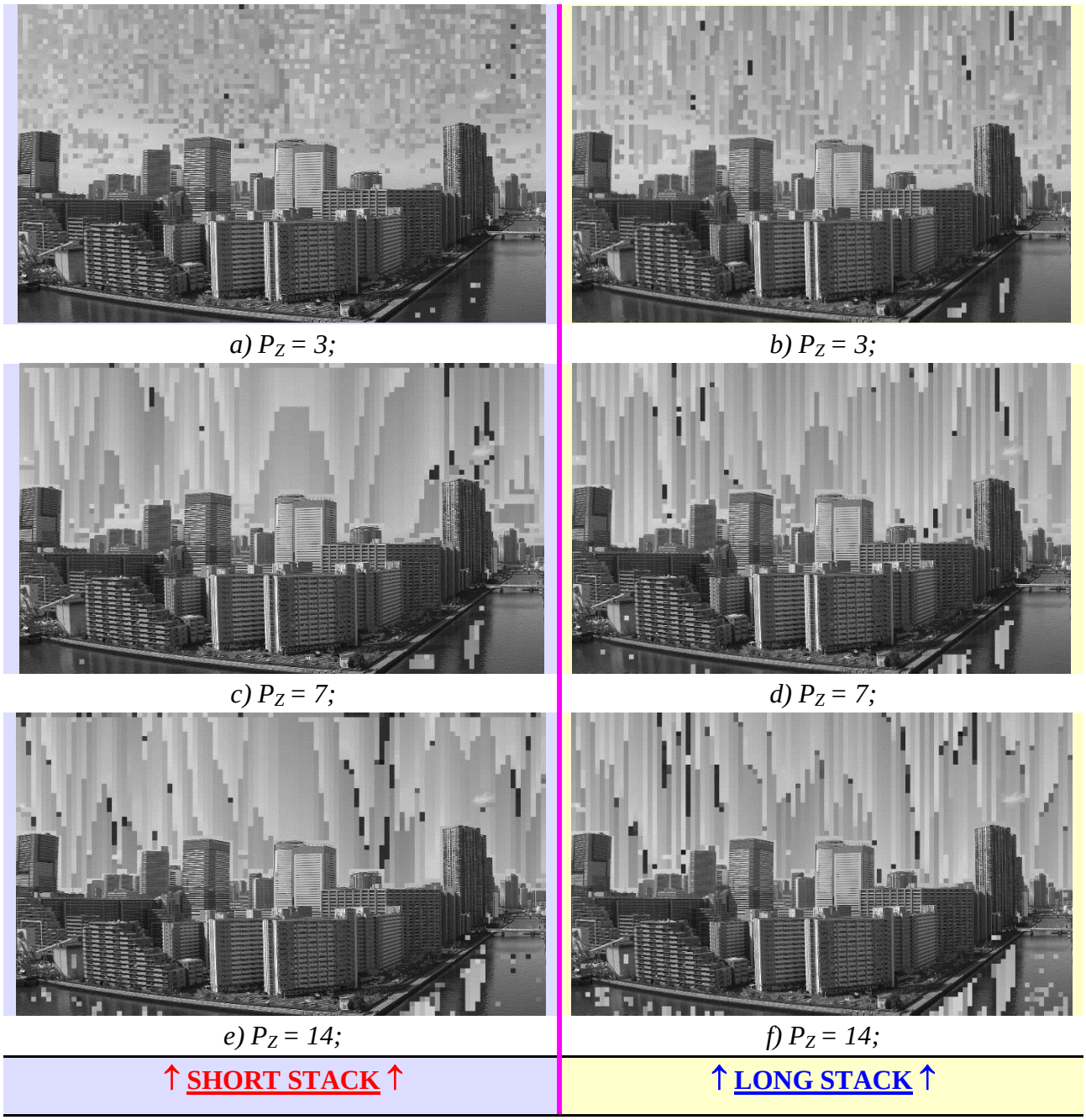


Fig. 1 - The results of attack of the *"landscape"* type test image for stacks of different size and different P_Z *(for BB 8×8 el. scan by "columns")* [7]

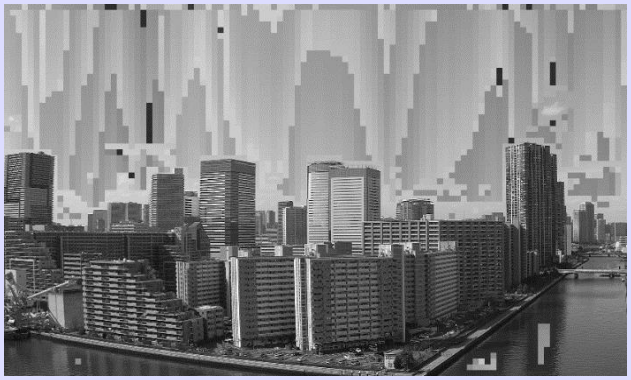
It should be noted that the results of a failed attack (*i.e., an unauthorized extraction of a test image*) presented below were obtained when the source massif had been scanned column by column [6]. The term "*scanning*" in this context should be understood as a method of traversing and consequent extracting the current parameters of the BB runs from the base massif of image – content series. During the simulation, the function of intra-block multiplexing of data [1] was turned off, which is clearly visible in the practically undistorted highly-detailed areas of the test image in Fig. 1-3 (a *part of the image with urban buildings*).

a) BB 4× 4 el., "columns";

b) BB 4 × 4 el., "columns";

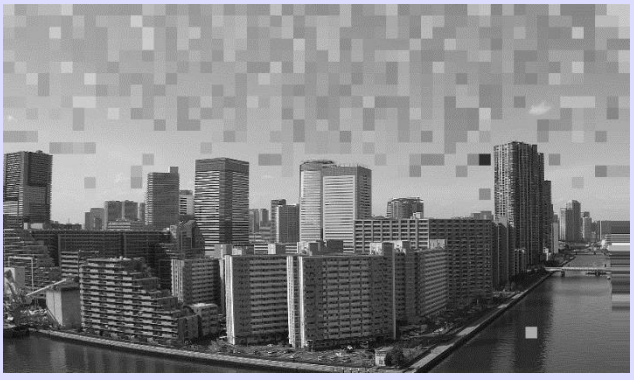
a') BB 4× 4 el., "line by line";

b') BB 4 × 4 el., "line by line";



c) BB 8×8 el.;

d) BB 8×8 el.;



e) BB 16 × 16 el.;

f) BB 16 × 16 el.;

↑ **SHORT STACK** ↑ ↑ **LONG STACK** ↑

Fig. 2 - Attack results of the test image for stacks of different dimensions
(samples a-f, scan by "columns" at $P_Z = 5$; samples a'-b', scan by "line by line" for $P_Z = 7$)

In other words, mutual obfuscation of significant elements for different BBs was not carried out. In addition, to present the total number of series subjected to the inter-block multiplexing visually, all series BB were marked with white *(see Fig. 3 (b, d, f))*. It is important to emphasize that in the case of Fig. 3, a sample stack of a small size was used (*4 series*), which did not affect the sizes of the areas of the test images for which the procedure of inter-block multiplexing of the effective parameters of the runs under the specified limits of the value of P_Z (*where* $P_Z \leq 14$) was implemented.

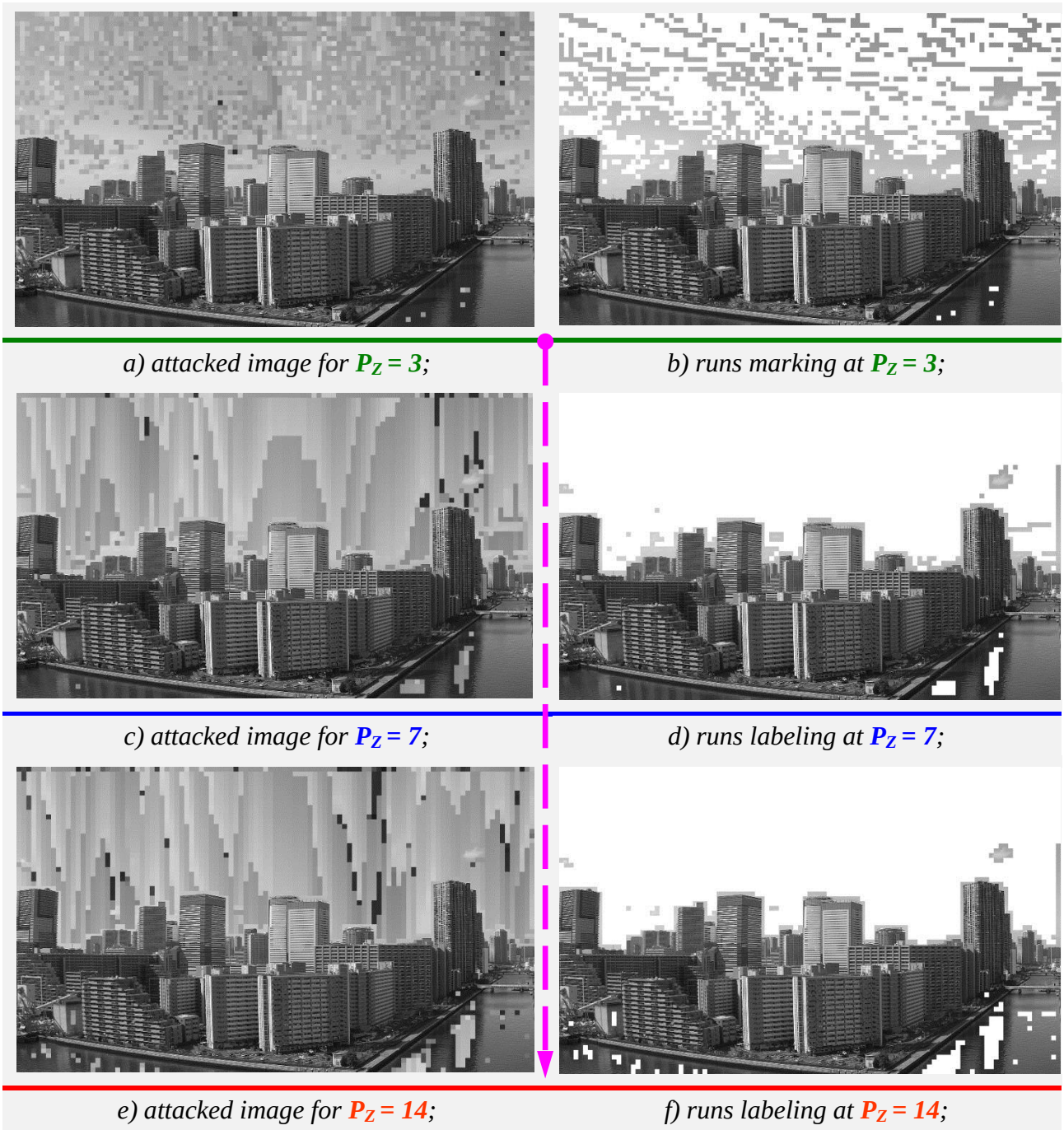


Fig. 3 - The result of image attack *(a, c, e)* and labeling of all series BB *(b, d, f)* for different values P_Z *(for BB 8×8 el., scan by "columns")* [7]

3. Conclusions

1. Increasing the length of the sample stack of runs expands the combinatorics of inter-block multiplexing for both parameters of the formed runs, which significantly destroys the correlations

between the elements of the original array of series of the content. This effect is clearly confirmed by a significant increase in the density of placement of series of different shades in the background areas of the attacked test image, which uses a wide base of permutations *(see comparison of Fig. 1(c) and Fig. 1(d))*.

2. Regardless the size of a sample stack, an increase in the value of the coarsening threshold P_Z results in an increase in the length of the formed BB series *(see the comparison of image columns (a, c, e) and (b, d, f) in Fig. 1)*. In addition, $P_Z \leq 7$ should be considered acceptable for the vast majority of images. In doing so $P_Z = 14$ (*for 256 brightness levels*) should be considered critical (limiting) for most realistic images (*such as portrait and landscape*) [3]. The use of large values leads P_Z to serious degradation of the original data (*red zon in Fig. 1(e)* [8]).

3. Regardless the size of a sample stack, an increase in the size of the BB results in a simultaneous decrease in the number of the series, and in their average length *(clearly visible in Fig. 2, samples (b) and (f))*. This leads to a reduction in the combinatorics of permutations for the current parameters of the series within the limits of the adopted multiplexing masks.

4. With equal values of P_Z, using a longer stack undeniably increases the possibility of resisting attempts of illegitimate extraction (*selection of current multiplexing parameters*) of content (*see of samples comparison (a-b), (c-d) and (e-f), in Fig. 2*).

5. With an increase the threshold value of the acceptable difference in the brightness of the elements of the adjacent blocks of the image (i.e., the P_Z) increasing, the total number of series decreases, and their length increases *(see samples a, c, e, in Fig. 3)*.

6. As the value of P_Z increases by more than 7 gradations of brightness *(when quantizing of the elements is 8 bit/el.)* in the background areas of the test images, there is an erroneous "dropping" of blocks with uncharacteristic brightness of constituent elements *(see chains of black runs in Fig. 3(c) and Fig. 3(e))*. Such an effect does not conform to acceptable level of content distortions, especially for the less informative background areas.

7. It's obvious that the use of two levels of multiplexing [1] of output data at once significantly increases the resistance of the content to attempts at its unauthorized extraction, leading to large distortions in the attacked image, in case of incorrect (erroneous) selection of the current processing algorithm parameters.

8. The dimensionality of the BB and the method of organizing the sweep of the blocks series *(see samples a-b and a′-b′, in Fig. 2)*, are elements of the composite key of the data extractor [1], which determine the current procedure for the implementation of interblock data processing procedures *(1st level of protection)*, as a tools for legitimizing access to content data.

References

[1] Лесная, Ю., Гончаров, Н., & Малахов, С. (2021). ОТРАБОТКА КОНЦЕПТА МНОГОУРОВНЕВОГО МУЛЬТИПЛЕКСА ДАННЫХ ГИБРИДНОГО СТЕГАНОАЛГОРИТМА. Збірник наукових праць SCIENTIA. Вилучено із https://ojs.ukrlogos.in.ua/index.php/scientia/article/view/17666

[2] Гончаров, М., Лєсная, Ю., & Малахов, С. (2021). Дослідження властивостей прототипу гібридного стеганоалгоритму. Комп'ютерні науки та кібербезпека, (2), 45-56. https://doi.org/10.26565/2519-2310-2021-2-05

[3] Прэтт У. (1985). Цифровая обработка изображений (Д. С. Лебедева, пер. с англ.). т. 1,2. Москва: Мир.

[4] Гончаров, Н., & Малахов, С. (2022). Использование параметра длин серий, как элемента межблочного мультиплекса данных стеганоалгоритма. Збірник наукових праць ΛΌГОΣ, 180-187. https://doi.org/10.36074/logos-08.07.2022.050

[5] Бутаков Е. А., Островский В. И., & Фадеев И. Л. (1987). Обработка изображений на ЭВМ. Москва: Радио и связь.

[6] Гончаров, Н., Лесная, Ю., & Малахов, С. (2022). Адаптация принципа кодирования длин серий для противодействия попыткам неавторизованной экстракции стеганоконтента. Grail of Science, (17), 241-247. https://doi.org/10.36074/grail-of-science.22.07.2022.042

[7] Гончаров, М., Лєсная, Ю., & Малахов, С. (2022). МОДЕЛЮВАННЯ СПРОБ ЕКСТРАКЦІЇ СТЕГАНОКОНТЕНТА ПРИ РІЗНІЙ ДОВЖИНІ СТЕКУ ВИБІРКИ ПАРАМЕТРІВ СЕРІЙ. *Grail of Science*, (18-19), 173–177. https://doi.org/10.36074/grail-of-science.26.08.2022.31

[8] Гончаров Н., Лєсная Ю., Семёнов А., Малахов С. Моделирование атаки стеганоконтента на коротком стеке выборки параметров серий при грубых оценках подобия исходных данных. // The main prospects for the development of science in modern life. Proceedings of the XXXVI International Scientific and Practical Conference. Warsaw, Poland. 2022. Pp. 344-347 URL: https://isg-konf.com/the-main-prospects-for-the-development-of-science-in-modern-life/

Надійшла: серпень 2022. Прийнята: вересень 2022.

Автори:

Микита Гончаров, студент факультету комп'ютерних наук (магістрат), Харківський національний університет імені В.Н. Каразіна, Україна.

ORCID ID https://orcid.org/0000-0002-9790-7260

E-mail: worldxdark@gmail.com

Лариса Павлова, ст. викладач кафедри іноземних мов професійного спрямування факультету іноземних мов, Харківський національний університет імені В.Н. Каразіна, Україна.

ORCID ID https://orcid.org/0000-0002-5854-4209

E-mail: l.v.pavlova@karazin.ua

Юлія Лєсная, студентка факультету комп'ютерних наук (магістрат), Харківський національний університет імені В.Н. Каразіна, Україна.

E-**mail**: xa12284109@student.karazin.ua

Моделювання спроб вилучення стеганоконтенту з різною довжиною стеку вибірки серій, блоків зображень.

Анотація. Розглянуто результати, які отримані при використанні стеків вибірки серій різної довжини, при моделюванні спроб несанкціонованого вилучення (атаки) стеганографічного контенту, що «захищається» за допомогою реалізації механізму міжблокового мультиплексування діючих параметрів серій опорних блоків зображення. Підтверджується взаємозв'язок між параметрами обробки контенту (напівтонових зображень) та кількістю серій, а також комбінаторикою складових елементів діючих пар параметрів серій, що є об'єктами міжблокового мультиплексування. Зроблено висновок, що одночасне використання 2-рівневого мультиплексування даних, значно розширює можливості протистояти спробам атаки контенту. Встановлено, що використання блоків більшої розмірності істотно знижує роль поточних параметрів серій опорних (базових) блоків, стосовно порушення структури вихідних зображень (вихідного контенту). Відзначено, що використання одразу двох рівнів мультиплексу вихідних даних, в значної мірі посилює стійкість контенту до спроб його неавторизованого вилучення, обумовлюючи великі спотворення в атакованому зображенні, в разі хибного підбору діючих параметрів обробки.

Ключові слова: кодування довжин серій; стеганографія; контент; атака; стек.

УДК 681.04 DOI: 10.26565/2519-2310-2022-2-03

СУЧАСНІ ЗАГРОЗИ ТА СПОСОБИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ

Кирило Яремчук, Денис Воскобойников, Ольга Мелкозьорова

Харківський національний університет імені В.Н. Каразіна, Харків, 61022, Україна
kir.yaremchuk@gmail.com, denisvoskoboinykov@gmail.com, olha.melkozerova@karazin.ua

Надійшла: вересень 2022. Прийнята: жовтень 2022.

Анотація: *Складність розроблюваних веб-застосунків зростає з кожним роком, що, в свою чергу, робить важкоздійсненним забезпечення їхньої безпеки. Саме тому, доцільно приділяти особливу увагу критичним проблемам захисту програмного забезпечення. Вміння оцінювати ризики та запобігати вразливостям ще на етапі проектування продукту є вкрай важливою задачею, котра знижує потенційні складності при експлуатації застосунку. За останні роки кількість випадків витоку даних у всіх галузях ринку зменшилася, але, їх руйнівність стала значнішою. Серед усіх атак, атаки на веб-застосунки становлять більш ніж 50 відсотків. Згідно зі списком вразливостей OWASP Top Ten, в роботі розглянуто актуальні категорії вразливостей та напрямки атак на існуючі веб-застосунки. Було розглянуто ефективні способи їх запобіганню. Наведені рекомендації щодо реалізації та підтримки захищеності додатків, розроблених з використанням бібліотеки ReactJS. Було виділено найпоширеніші загрози безпеки продуктів на базі React на протязі життєвого циклу додатку. Розглянуті основні способи оптимізації ReactJS.*

Ключові слова: *вразливість; веб-застосунки; загрози веб-застосунків; методи безпеки ReactJS*

1. Вступ

Завдяки тривалому часу свого розвитку, веб-застосунки почали являти собою дещо куди значніше, аніж просто сайти з контентом. На просторах мережі Інтернет с кожним днем з'являються все більш складні веб-застосунки за своїми цілями й можливостями, котрі пропонують нові рішення задля задоволення вимог споживачів в усіх галузях ринку. Веб-застосунки являють собою найбільш зручний та ефективний засіб для представлення інформації й надання послуг у мережі. Компанії з різнобічних галузей ринку продовжують створювати веб-застосунки для просування своїх товарів та послуг у Інтернеті, займаючи свої ніші у цифровому світі. Мобільні інструменти та веб-технології захопили вершину списку на довгі роки. Такі цифрові сервіси часто бувають критично значущими й потребують захисту задля забезпечення безпеки різного роду конфіденційної інформації. Галузь веб-технологій розвивається невпинними кроками, однак несе з собою і певні ризики безпеки: - забезпечення захисту значної кількості різнотипної інформації досягти досить складно. Недотримання вимог безпеки може загрожувати втратою ресурсів, а інколи й проблемами з законом [1]. На рис. 1 представлені тенденції злому систем безпеки у 2014-2021 роках [2].

Відповідно звіту *Risk Based Security* про тенденції порушення даних за 2021 рік, у 2020 та 2021 роках було втрачено 27,81 і 18,88 млрд записів, тоді як у 2019 році усього 4,681 млрд. записів [2]. Попри те, що кількість таких випадків зменшилася у 2020 та 2021 роках, втрати даних стали більш масштабними. В цілому, можна спостерігати відсутність кореляції між кількістю випадків втрати даних та кількістю втрачених даних, що говорить про те, що навіть одне малозначне порушення безпеки може спричинити серйозні наслідки. Щоб на практиці знизити можливі ризики безпеки, потрібно ретельно контролювати усі потенційні загрози та уважно дотримуватися існуючих стандартів інформаційної безпеки (ІБ). На рис. 2 представлено огляд основних тенденцій атак (зломів) по галузям економіки у 2021 році [2].

Так, найбільший вплив загроз ІБ у 2021 році зазнали сфери охорони здоров'я, фінансів, страхування, ІТ- індустрія й наукова галузь, промисловість та галузь державного управління, котрі являють собою значну долю від усіх атак.

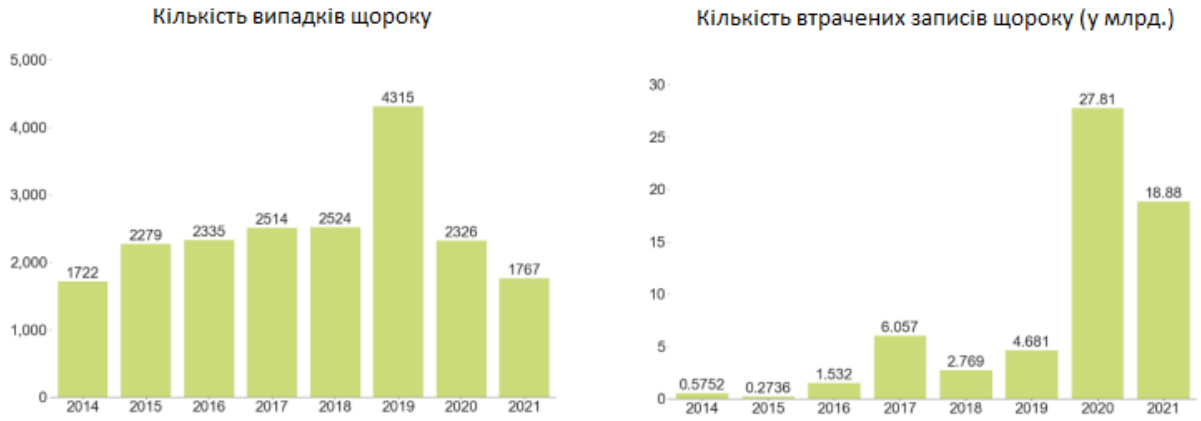


Рис. 1 – Тенденції атак систем ІБ у 2014-2021 роках

Попри те, що у даному дослідженні розкривається лише проблематика загроз та вразливостей веб-застосунків, джерела даних, стосовно випадків витоку даних, можуть бути абсолютно різними.

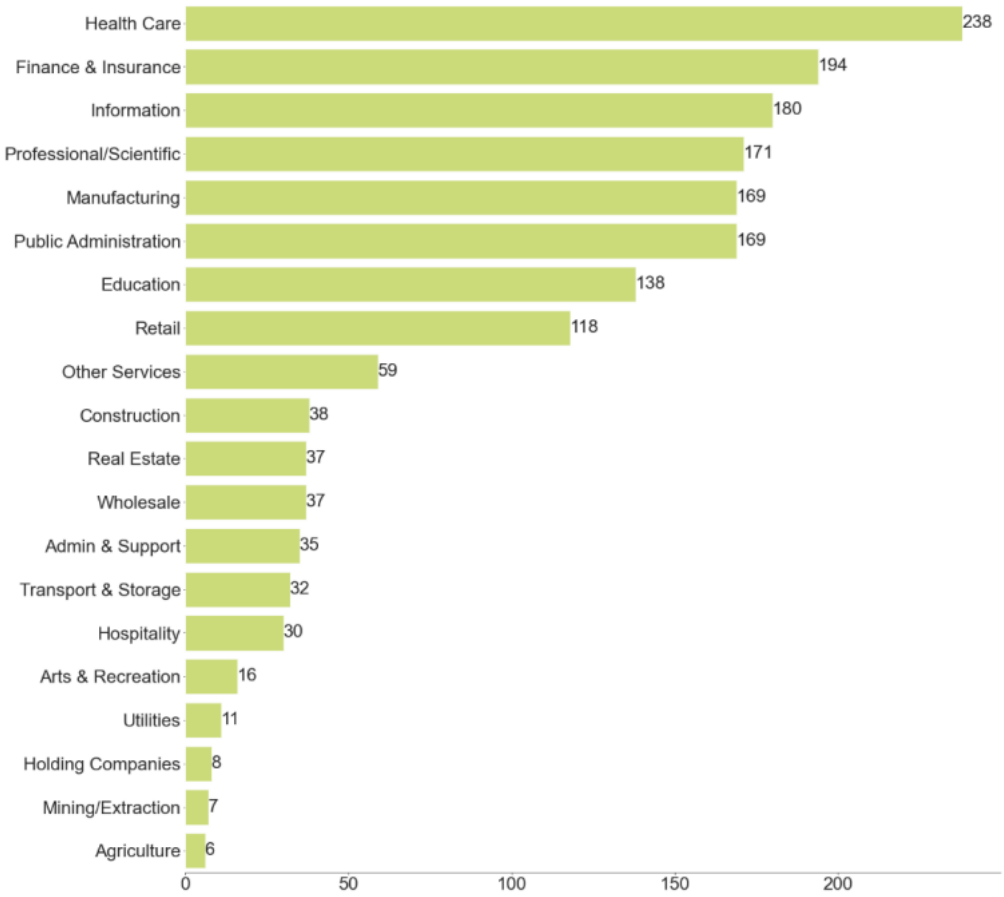


Рис. 2 – Тенденції типових атак за галузями у 2021 році

2. Загальний стан в сфері безпеки веб-застосунків

Атаки на веб-застосунки являють собою найнебезпечнішу загрозу ІБ багатьох сучасних організацій. Атаки, що спрямовані на веб-застосунки становили 40% від усіх зламів у 2015 році й вже понад 50% у 2021 році [3]. Згідно з аналізу *WhiteHat Security* на момент 2021 року, так чи інакше, принаймні 50 відсотків веб-застосунків у таких сферах, як: освіта, промисло-

вість, роздрібна торгівля, комунальні служби та охорона здоров'я, мають принаймні одну вразливість, котра можливо використати для зламу відповідних додатків. В ході дослідження було виявлено, що через те, що з кожним роком все більша кількість галузей фокусуються на розвиток у мережі, ризики витоку їх корпоративних даних залишаються досить високими, що позначається на збільшенні вразливостей [4]. Також було виявлено, що, наприклад, лише деякі з найпоширеніших ризиків від *WhiteHat* можна побачити в списку OWASP (*Open Web Application Security Project*), що регулярно оновлює найбільш розповсюджені вразливості.

При цьому, знаходження проблеми ІБ не вирішує проблему безпеки саме по собі. Для усунення виявлених вразливостей потрібен деякий час. Так, час на виправлення критичних вразливостей застосунків, в середньому, сягає 194 діб. Середній час викриття критичних вразливостей становить 300 діб, а вразливостей с високим рівнем ризику понад 500 діб [4]. Чим більший термін виявлення, тим більші шанси злодіїв на «успіх» від вразливості, що експлуатується. Також, великий відсоток веб-застосунків залишається вразливим завжди, що каже про те, що розробники не в змозі впоратися з усіма вразливостями своїх продуктів.

Усунення проблем безпеки та виявлення вразливостей ще на етапі проектування й розробки веб-застосунку потребує значно менших ресурсів, аніж їх виправлення у вже існуючому продукті. У існуючому застосунку процес виправлення вразливостей здатний призвести до втрати працездатності веб-застосунку, що в першу чергу несе втрату репутації, коштів й ресурсів на вирішення проблеми. Вдосконалення вже існуючого веб-застосунку у більш захищене рішення може коштувати значно більше сил і коштів, аніж його початкова реалізація у вигляді захищеного застосунку.

На сьогоднішній день, розробка веб-застосунків, беручи до уваги стандарти безпеки, є неодмінним обов'язком. Список загроз та вразливостей постійно розширюється, не стоячи на місці, тому розробники веб-застосунків мають мати відповідні компетенції у сфері ІБ, щоб залишати свій продукт безпечним. На жаль, впоратися з цією задачею буває не часто, тому у якості допомоги розробники можуть звертатися до стандартів безпеки або користуватися іншими інструментами, котрі здатні допомогти у забезпеченні безпеки їх програмних рішень.

3. Стислий огляд поширених загроз безпеки веб-застосунків та способів захисту на базі бібліотеки ReactJS

OWASP являє собою головний ресурс з інформацією про стандарти, відомі методи захисту та інструменти безпеки. Через велику базу користувачів та відкриту спільноту, OWASP роками залишається своєрідною бібліотекою, в котрій містяться різноманітні документи, що допомагають вирішити проблеми безпеки при розробці продуктів.

Головною метою проекту OWASP Top Ten є формулювання основних загроз ІБ, визначення найважливіших та найбільш критичних недоліків захищеності веб-застосунків. Цей список створено за сприяння об'єднаної спільноти проекту. На початку існування ціллю проекту було покращити знання розробників, щодо безпеки, але згодом проект став справжнім стандартом безпеки веб-застосунків. Наразі актуальною є версія 2021 року, котра вийшла відносно недавно й включає в себе найбільш актуальний список загроз та вразливостей.

Останню версію OWASP складено на базі опитувань у відповідній галузі та більш ніж 40 відгуків від компаній, що займаються забезпечення безпеки. Також, була зібрана інформація з більш ніж ста тисяч застосунків та API (*Application Programming Interface*), що дозволило створити переконливу інформаційну базу задля подальшого аналізу даних.

Останнім часом, веб-інструменти та технології зазнали суттєвих змін. Так, мову програмування JavaScript та її фреймворки *AngularJS* та *ReactJS* [5], можливо впевнено назвати своєрідним фундаментом або основою мережі. Завдяки цим фреймворкам деяка «відповіда-

льність» серверів була перенесена на сторону браузера. В той же час величезну популярність набрала серверна технологія розробки Node.js, котра працює на базу подій [6]. На рис. 3 можна бачити зміни у списку OWASP Top Ten [7].

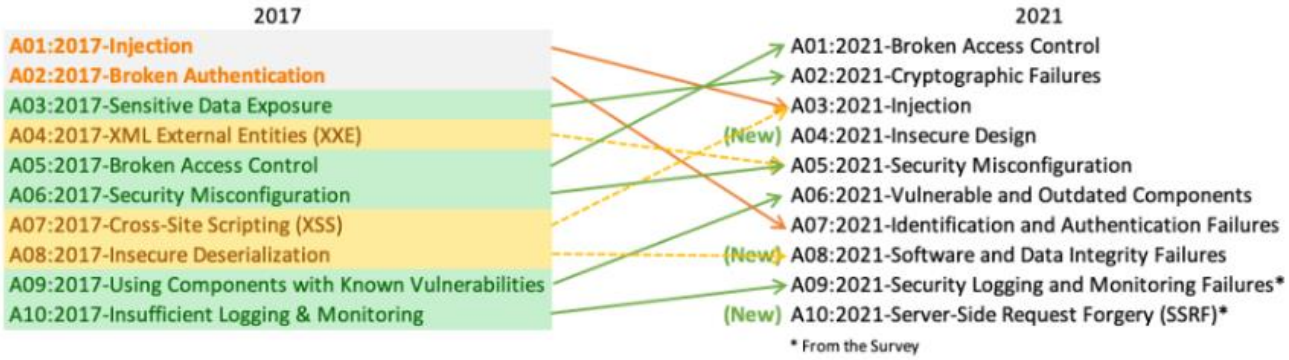


Рис. 3 – Рейтінг OWASP Top Ten за 2017 та 2021 рр..

Як можна бачити, за цей термін, перелік OWASP зазнав великих змін [7]. Так, *Broken Access Control* - піднялася в списку й опинилася на 1-й позиції. В реалізованих веб-застосунках має бути реалізовано доступ до інформації згідно з привілегіями користувачів. У випадку ігнорування безпеки доступу до інформації, це може дозволити користувачам отримати доступ до чутливої інформації без наданих на це повноважень, що може призвести до незворотних наслідків та/або втрати критично важливих даних.

Cryptographic Failures (раніше *Sensitive Data Exposure*) - піднялася на 2-ге місце. Ця вразливість пов'язана з втратою конфіденційних даних чи зломом системи, якщо використовувані криптографічні методи не здатні забезпечити достатній рівень ІБ: наприклад, неактуальні криптографічні шифри та протоколи.

*Injection (*до цієї категорії було додано і XSS *(Cross Site Scripting))* - міжсайтове виконання сценаріїв наразі стало невід'ємною частиною цієї категорії (опустилася на 3 місце). Дана вразливість надає злочинцям змогу впроваджувати у веб-застосунок потрібні їм недекларовані дані та/або інструкції, що забезпечує отримання несанкціонованого доступу до цільових даних. Також, ін'єкції здатні власноруч змінювати веб-застосунок.

Insecure Design - поєднує у собі ризики безпеки, котрі пов'язані з недоліками дизайну (тобто загальної структури і окремих алгоритмічних конструкції додатків). Тому, дизайн має охоплювати шаблони проектування й безпечну еталонну архітектуру програмного забезпечення (ПЗ). Являє нову категорію у релізі загроз 2021 року (4-те місце).

Security Misconfiguration – ця категорія піднялася вгору (на 5-ту позицію), через розширення кількості використань ПЗ з широкими можливостями їх налаштування, що обумовлює потенційну можливість нелегітимного доступу до інформації та маніпулювання даними, у томи числі, модифікацію і зміну даних тощо.

Vulnerable and Outdated Components (раніше, *Using Components with Known Vulnerabilities)* – є єдиною категорією, яка не має жодних CVE (*Common Vulnerabilities and Exposures*), зіставлених із включеними CWE (*Common Weakness Enumeration*), піднялася на 6 місце [7]. Охоплює всі застарілі складові ПЗ, експлуатація складає критичні ризики ІБ системи. Потребує регулярне сканування системи, що дозволяє завчасно виявляти наявні проблеми ІБ (*намічі безпеки, використання експлойт-паків*).

Identification and Authentication Failures (раніше, *Broken Authentication*) – ця категорія, відтепер, включає CWE, котрі більшою мірою відносяться до помилок ідентифікації, опустилася на 7 місце. Категорія охоплює вразливості, що включають в себе фальсифікацію обліко-

вих даних та "brute force" атаки. Вразливості цієї категорії є похідними недоліків застосованих механізмів автентифікації та перевірки сеансу користувача.

Software and Data Integrity Failures - нова категорія (8 місце рейтингу), яка фокусується на ідеях оновлень ПЗ, критичних даних і конвеєрів CI/CD (*Continuous Integration/Continuous Delivery*) без перевірки на цілісність, що забезпечує злочинців можливостями використовувати дані, котрі надходять до серверу, для забезпечення можливостей проведення атак [7].

Security Logging and Monitoring Failures (раніше, *Insufficient Logging & Monitoring*) – яка займає 9 місце, була розширена задля включення більшій кількості типів вразливостей, що пов'язані з незадовільним поточним контролем і фіксуванням подій ІБ. Наявність цих вразливостей обумовлює труднощі зі своєчасністю реагування на збої у роботі застосунків.

Server-Side Request Forgery - нова категорія, яка була додана з опитування галузевих фахівців. Її наявність надає зловмиснику можливість реалізувати недекларованe надсилання (сервером додатку) запитів до обраного зловмисником домену. Ця вразливість може виникнути у разі, якщо веб-застосунок отримує дані без перевірки адреси, яку надає користувач.

Бібліотека *ReactJS* (*або React*) завдяки своїй гнучкості є найбільш поширеним рішенням для створення клієнтської частини веб-застосунків (завдяки JSX (*JavaScript XML*) синтаксису)) [8]. У зв'язку зі зростанням складності розроблюваних застосунків і збільшенням обсягів даних, проблеми безпеки таких продуктів невпинно зростають. Так, серед найпоширеніших загроз для веб-застосунків, що створені із застосуванням *ReactJS* слід виділити [8]:

- Вразливість *Zip Slip*, є критично важливою і дозволяє маніпулювати застосунком за рахунок вилучення архіву та інтегрування будь-яких файлів у систему;
- *XSS* (або *Cross-Site Scripting*), що передбачає можливість виконання шкідливого програмного коду, котрий приймається за справжній і виконується за стосунком;
- *XXE* (*XML External Entity*) атаки на XML парсери у випадках, коли ті некоректно опрацьовують посилання на деякі зовнішні сутності;
- *Arbitrary Code Execution*, яка працює за принципом впровадження зловмисником експлойту (та/або експлуатації вразливості 0-го дню), що забезпечує віддалене ініціювання виконання нештатного програмний коду та/або маніпулювання діями скомпрометованого додатку та/або апаратного засобу;
- Вразливість *Broken Authentication* здійснюється завдяки існування передумов компрометації діючих процедур авторизації (тобто недосконаліть автентифікації);
- Атаки типу *SQL Injection*, де зловмисник може відправляти шкідливі *SQL*-запити до бази даних та маніпулювати даними бази даних задля отримання своїх цілей.

Серед способів покращення *ReactJS*, що обумовлюють показники безпеки веб-застосунків, можна виділити наступні [8]:

- Захист від XSS, за допомогою використання технології прив'язки даних;
- Поточний моніторинг URL посилань та контроль впровадження стороннього шкідливого ПЗ, котре може здійснюватися через URL;
- Своєчасне оновлення використовуваних бібліотек. Актуальні версії React позбавлені від багатьох вразливостей минулих релізів;
- При розміщенні коду до вузлів об'єктної моделі документа (DOM - *Document Object Model*), слід не допускати прямого відкритого доступу;.

- Забезпечити протидію уразливостям, що діють за принципом впровадження даних *JSON* (*JavaScript Object Notation*). Зазвичай ці дані відображаються на боці сервера і передаються разом зі сторінками *React* у форматі обміну даними *JSON*;
- При розгортанні HTML коду, його слід розміщувати відразу у *DOM* вузли, що потребує ретельної перевірки всіх процедур;
- Забезпечити регулярність перевірок поточного стану вразливостей (інструмент *OWASP Dependency-Check*), що надає змогу виявляти відомі різновиди вразливостей, які розташовуються у залежностях проекту, перш ніж додавати їх до проекту;
- Виключити застосування сумнівного коду із бібліотек та використовувати ПЗ, яке оптимізує створюваний код;
- Впроваджувати корисні функції *React* для серверного відображення даних на клієнті, що надасть змогу екранувати дані при їх візуалізації на клієнтській частині застосунку у автоматичному режимі.

4. Висновки

1. Використання методів та практик щодо написання безпечного коду, так само як і проектування веб-застосунків з використанням захищеного дизайну являють собою досить складну задачу.

2. Постійний аналіз стану проблематики, щодо поточного стану відомих вразливостей веб-застосунків та появи сучасних методів їх парирування, є принциповою умовою роботи для фахівців, які професійно пов’язані з розробкою та тестуванням веб-застосунків.

3. Питанню постійного моніторингу вразливостей має приділятися певна увага протягом усього життєвого циклу роботи створеного застосунку. Для контролю за вразливостями слід використовувати діючи рейтинги ІБ, та відповідні засоби відстеження актуальних вразливостей та загроз ІБ.

4. Веб-застосунки, що реалізовані на базі використання бібліотеки *ReactJS*, дозволяють значно зменшити зусилля, які необхідно докласти розробникам програмних додатків для досягнення певного рівня ІБ. Ця бібліотека сприяє дотримання сучасних стандартів безпеки веб-застосунків, та дозволяє уникати типових помилок при розробці відповідних додатків.

Список літератури

[1] Inamdar, D. M., & Gupta, S. (2020). A Survey on Web Application Security. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, (6), 223-228.

[2] RiskBased Security, 2021 Mid Year Data Breach QuickView Report. (2021). Вилучено з https://pages.riskbasedsecurity.com/download-the-2021-mid-year-data-breach-quickview-report-today

[3] Help Net Security. Web app attacks are skyrocketing, it’s time to protect APIs. (2021). Вилучено з https://www.helpnetsecurity.com/2021/12/27/web-app-attacks-increased/

[4] Dark Reading, WhiteHat Security: 50% of Apps Are Vulnerable. (2021). Вилучено з https://www.darkreading.com/application-security/whitehat-security-50-of-apps-are-vulnerable

[5] Imaginary Cloud, Angular vs React: a comparison of both frameworks. (2020). Вилучено з https://www.imaginarycloud.com/blog/angular-vs-react/

[6] AltexSoft, The Good and the Bad of Node.js Web App Programming. (2022). Вилучено з https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/

[7] OWASP Top Ten – 2021. (2021). Вилучено з https://owasp.org/www-project-top-ten/

[8] React.js security best practices. (2020). Вилучено з https://upplabs.medium.com/react-js-security-best-practices-62b9a281cc42

Received: September 2022. Accepted: October 2022.

Authors:
Kyrylo Yaremchuk, student (magistracy), Faculty of Computer Science, V. N. Karazin Kharkiv National University, Kharkov, Ukraine.

E-mail: kir.yaremchuk@gmail.com

Denys Voskoboinykov, student (magistracy), Faculty of Computer Science, V. N. Karazin Kharkiv National University, Kharkov, Ukraine.

E-mail: kir.yaremchuk@gmail.com

Olha Melkozerova, Ph.D., Associate Professor Department of Security of Information Systems and Technologies, V.N. Karazin Kharkiv National University, Ukraine.

ORCID ID https://orcid.org/0000-0002-1134-2925

E-mail: olha.melkozerova@karazin.ua

Modern threats and ways to secure web applications.

Abstract. The complexity of the developed web applications is growing every year, which, in turn, makes it difficult to ensure their security. That is why it is advisable to pay special attention to the critical problems of software protection. The ability to assess risks and prevent vulnerabilities at the product design stage is an extremely important task, which reduces the potential difficulties in the operation of the application. In recent years, the number of data breaches in all market sectors has decreased, but their consequences have become more dangerous. Among all attacks, attacks on web applications account for more than 50 percent. According to the OWASP Top Ten list of the vulnerabilities, the relevant categories of vulnerabilities and directions of attacks on existing web applications were worked out in the work. Effective ways of their prevention are considered. Recommendations for implementing and maintaining the security of applications developed using the ReactJS library are provided. The most common security threats to React-based products throughout the application life cycle have been identified. Modern way of ReactJS optimization are considered.

Keywords: vulnerability; web applications; web application threats; ReactJS security methods.

УДК 681.04: 004.415.53 DOI: 10.26565/2519-2310-2022-2-04

ОГЛЯД ПОТОЧНОГО СТАНУ ЗАГРОЗ, ЩО ОБУМОВЛЕНІ ВПЛИВОМ ЕКСПЛОЙТІВ

Єлизавета Богданова, Тетяна Чорна, Сергій Малахов

Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, Харків, 61022, Україна
xa12850323@student.karazin.ua, tatyanachernaya2002@gmail.com, malakhov@ karazin.ua

Надійшла: жовтень 2022. Прийнята: листопад 2022.

Анотація: *Розглянуто проблематику експлуатації вразливостей програмного забезпечення. Звернено увагу на існування двох іпостасей практичного застосування експлойтів: - як інструменту атаки та, як засобу тестування інформаційних систем, що потребують захисту. Підкреслено, що частіше всього експлойти поділяють за типом вразливості безпеки, що експлуатується. Аналіз відомих інцидентів, пов'язаних з використанням експлойтів, дозволяє стверджувати про існування зв'язку між ступенем популярності програмного продукту або пристрою та ймовірністю створення відповідних експлойтів. Звернено увагу на те, що N–day експлойти складають суттєву частину існуючих загроз безпеки для вразливих пристроїв (систем). Головною причиною подібного становища є, несвоєчасне оновлення використовуваного програмного забезпечення та ігнорування оновлень патчів безпеки. Підкреслено надзвичайну важливість своєчасного випуску патчів безпеки, як ефективного засобу парірування виявлених уразливостей програмного забезпечення. Звернено увагу на те, що процес випуску патчів безпеки є базовою складовою у спектрі можливих захисних реакцій, при вирішенні подібних проблем. Зроблено акцент на тому, що за результатами аналізу відомих випадків протиправного використання експлойтів (за останні 3 роки) вони, в своєї переважній більшості, спрямовані на 3-х векторах атак: - відмова в обслуговуванні; - неправомірне розширення (підвищення) існуючих повноважень управління; - віддалене виконання зловмисного коду.*

Ключові слова*: експлойт; програмна вразливість; патч безпеки; інформаційна безпека.*

1. Вступ

Відомо, що вразливості можуть бути присутні в будь-якій складовій сучасних інформаційних систем: - операційна система (ОС), програмне забезпечення (ПЗ), апаратне забезпечення, Web-сервісі [1]. В будь-якому випадку потенційні кібер зловмисники будуть постійно намагатися сканувати програмно апаратні ресурси обраної жертви та вести мережеву розвідку щодо параметрів використовуваних Web-сервісів [2], щоб знайти вразливість в периметрі безпеки цільової системи, а потім, за допомогою відповідного експлойту забезпечити необхідні умови для подальшої реалізації потрібних несанкціонованих дій [3]. Крім того, самі користувачі (персонал) потенційної жертви атаки, може ненавмисно (несвідомо) сприяти появі додаткових вразливостей інформаційної безпеки (ІБ), наприклад, використовуючи слабкі параметри настроювань конфіденційності у своїх соціальних мережах та/або облікових записах електронної пошти, та/або ставши жертвою інтегрованої багатоходової атаки, з використанням прийомів соціального інжинірингу (*т.з., SE-атак*) [4-5].

Кіберзлочинці можуть використовувати експлойти для різних цілей: від маніпуляцій з окремим локальним атакованим пристроєм до масштабних комп'ютерних злочинів з залученням до атаки заздалегідь скомпрометованих ними мережевих ресурсів, різних за масштабами ІТ структур. Фізично, завдяки експлойтам зловмисники можуть заблокувати доступ до скомпрометованого ними пристрою та/або системі, контролювати роботу використовуваних Web-сервісів та/або процесів, отримати доступ до конфіденційній службовій або персоніфікованій інформації (доксинг) [6], вимагати гроші у жертви та проводити акції з кібербулінгу [4-5]. Однак, поряд з цім, за допомогою програм експлойтів, фахівцями з ІБ можуть завчасно тестуватися на проникнення їх ІТ інфраструктуру, що дозволяє імітувати різні вектори атак на корпоративні ресурси та служби.

Оскільки у пошуку вразливостей зацікавлені обидві сторони, то випущених експлойтів стає все більше. Очевидно, що для упорядкування цього напрямку діяльності, необхідна кла-

сифікація, як вже існуючих, так і майбутніх експлойтів, що полегшить запобігати діям потенційних зловмисників і впорядкувати роботу фахівців щодо визначення вразливостей ІБ.

2. Основна частина

Станом на сьогоднішній день, існують два основних класи експлойтів, відомих серед профільних фахівців як, *N-day* експлойти та експлойти нульового дня (т.з. *zero-day, зеродеї*).

Атаки нульового дня, як правило, привертають до себе увагу, коли йдеться мова про загрози кібербезпеці, але , нажаль, найчастіше, саме відома вразливість *N-day* є набагато більш серйознішою проблемою для багатьох сучасних організацій. *N-day* експлойт - це вразливість, що вже відома, та для якої вже доступне відповідне виправлення [7]. Тобто після публікації патчу безпеки і CVE (*з англ. Common Vulnerabilities and Exposures*) вразливості, експлойт нульового дня стає *N-day* експлойтом (Рис. 1).

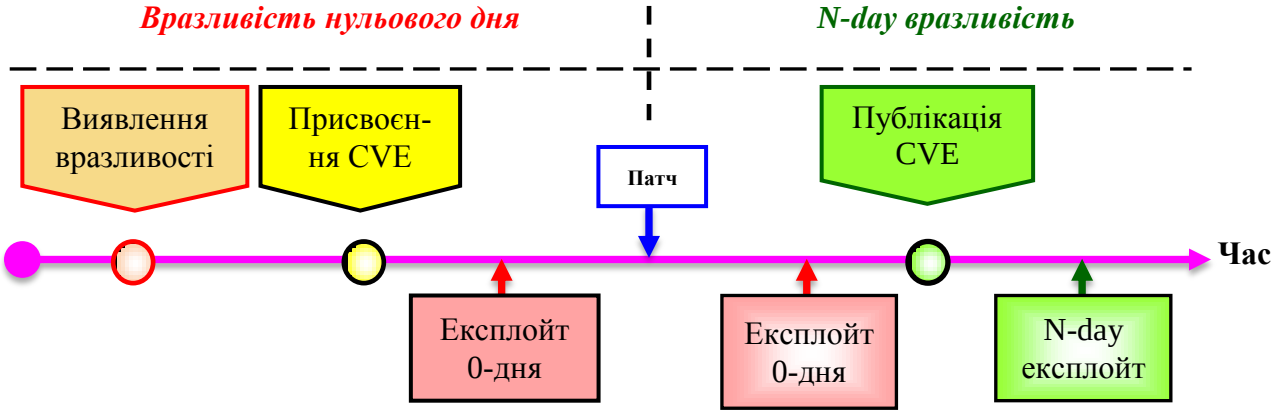


Рис. 1 – Алгоритм реагування на вразливості та публікації експлойтів

Теоретично, «застарілі» експлойти не повинні були б загрожувати безпеці пристроїв та ПЗ, проте вони, нажаль, продовжують використовуватися для атак і продовжують набирати популярності. Певною мірою це зумовлено тим, що зловмисники можуть дізнаватися про подробиці існуючих вразливість все, що їм потрібно, шукаючи відповідні виправлення (т.з. *«бінарне порівняння»)*, або переглядаючи загальнодоступні документи на наявність активних експлойтів. Також є можливість, що постачальники ПЗ та/або телекомунікаційного обладнання, не надають оновлення безпеки для своїх попередніх «продуктів», навіть якщо вони випускають відповідне виправлення ПЗ, або ж самі користувачі не встановлюють своєчасно патчі безпеки. Прикладом може бути відомий *N-day* експлойт *EternalBlue*, який був досить активним (*як інструмент для здійснення атак*), навіть через три роки (з 2017-2020 рр.) після випуску відповідного патчу. Серед іншого, він був використаний програмою-вимагачем *WannaCry*, а за даними *Shodan*, понад 650 000 підключених до Інтернету пристроїв все ще залишаються вразливими для нього[8]. Залежно від способу отримання доступу до вразливого ПЗ, існуючи експлойти умовно поділяються на ***віддалені*** (англ. *remote*), ***локальні*** (англ. *local*) та експлойти ***підставного серверу*** [9].

Віддалений експлойт, використовуючи вразливість без попереднього доступу до вразливої системи (пристрою), дозволяє зловмиснику отримати доступ до неї через мережу [10]. В залежності від того, який сервіс експлуатується, атакуючий отримує права користувача або *root* на сервері, що атакується. Після того, як зловмисник сканує сервер на наявність будь-яких відомих локальних експлойтів, якщо такі знаходяться, він використовує їх для отримання *root*-доступу на сервері. В разі, коли зловмисник отримує *root*-доступ, то він може

встановити руткіти (та/або бекдори) [11], які дозволять йому несанкціоновано увійти в скомпрометовану їм систему, та почати «працювати» на цільовому сервері без загрози його відстеження з боку адміністратора та/або інших користувачів. Найбільш поширеними різновидами *віддалених* експлойтів є переповнення буферу (*використання вразливості CVE-2015-0311, тобто переповнення буфера Adobe Flash Player, яка дозволяє зловмисникам віддалено виконувати довільний код через невідомі вектори атак*)[12] та інші атаки з неперевіреним введенням. Вони виконуються або для загальнодоступних служб (таких, як HTTP та FTP), або під час входу до захищених служб (таких, як POP та IMAP).

Локальний експлойт активується безпосередньо у вразливій системі (пристрої), вимагаючи попереднього доступу до неї, що дозволяє звичайному користувачеві отримати *root*-права, виконавши певну послідовність дій. Як правило, ці експлойти виникають в тому випадку, коли будь-яка привілейована програма містить помилку, яка не виконує достатніх перевірок користувача перед виконанням команди з правами суперкористувача (суперюзера).

Експлойти ***підставного серверу*** - це вразливість по відношенню до існуючих клієнтських додатків, що зазвичай включають змінені сервери, які розсилають експлойти при отриманні доступу до клієнтського додатку. Ця вразливість з «успіхом» працює в парі із SE–атаками, наприклад: фішинг і ланч-фішинг (для розповсюдження рекламного ПЗ) [9].

Очевидно, що без існування вразливостей не буде експлойта, оскільки саме вони і є умовною «точкою входу» для експлойтів в систему (апаратний пристрій), що атакується. Відповідно, логічно класифікувати експлойти за типом уразливості, яку вони використовують, наприклад [13]: - переповнення буферу; міжсайтовий скриптинг; підробка міжсайтових запитів; SQL-ін'єкція; атака повернення до бібліотеки та інші.

Існують загальна система оцінки вразливостей CVSS 2.0 (*Common Vulnerability Scoring System*) та рівень загрози експлойту, за допомогою яких можна дізнатися ступінь небезпеки для пристрою, що атакується. На рис. 2-3 приведені відомості CVSS та рівень загроз експлойту, станом на 2019–2022 р. Як можна помітити 34% відомих експлойтів мають високий рівень загрози (CVSS ≥ 7), а 54% експлойтів мають середній рівень (CVSS < 7 та CVSS ≥ 4). Тобто, 88% загальнодоступних експлойтів розроблені для вразливостей середнього чи високого ступеня небезпеки [13, 14].

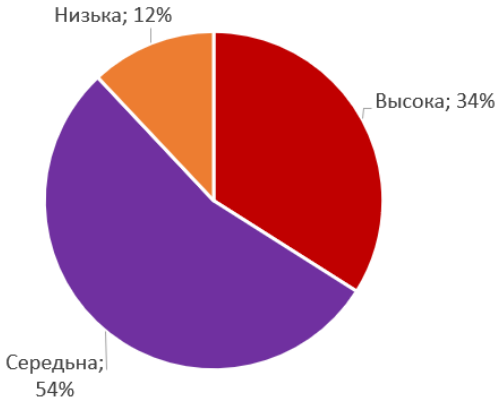


Рис. 2 – Розподіл експлойтів за рівнем загрози вразливості

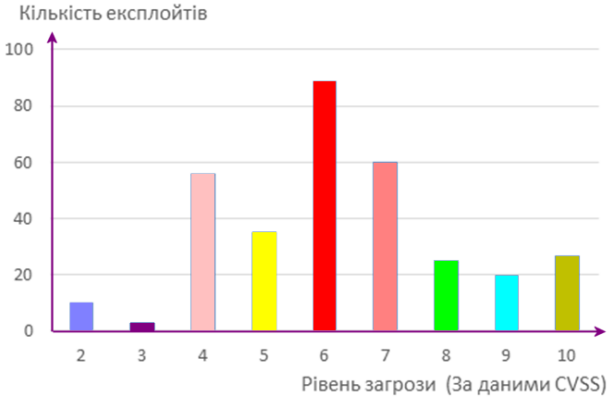


Рис. 3 - Експлойти, що класифіковані за рівнем загрози вразливості CVSS *(2019-2022 рр)*

Відома, також, і інша класифікація експлойтів по здійснюваним діям щодо вразливої цільової системи: - несанкціонований доступ до даних (*наприклад, копіювання, видалення або модифікація*); - віддалене виконання коду (*RCE - Remote Code Execution*); - відмова в обслуговуванні; - підвищення привілеїв користувачів; - обхід (блокування) окремих функції

безпеки; - довільне читання файлів; - пошкодження пам'яті; - несанкціонована зміна налаштувань мережевих засобів (*наприклад, модифікація таблиці комутації адресів роутерів*) та інші [9, 12]. Так наприклад, на рис. 4 наведено діаграми, які демонструють розподіл експлойтів по роках (*за 2019-2022 рр.*), що найчастіше застосовували різні вразливості атакованих систем [12, 15]. Як видно з діаграми найактуальнішими, отже затребуваними, є експлойти для віддаленого виконання коду, відмови в обслуговуванні та підвищення привілеїв.

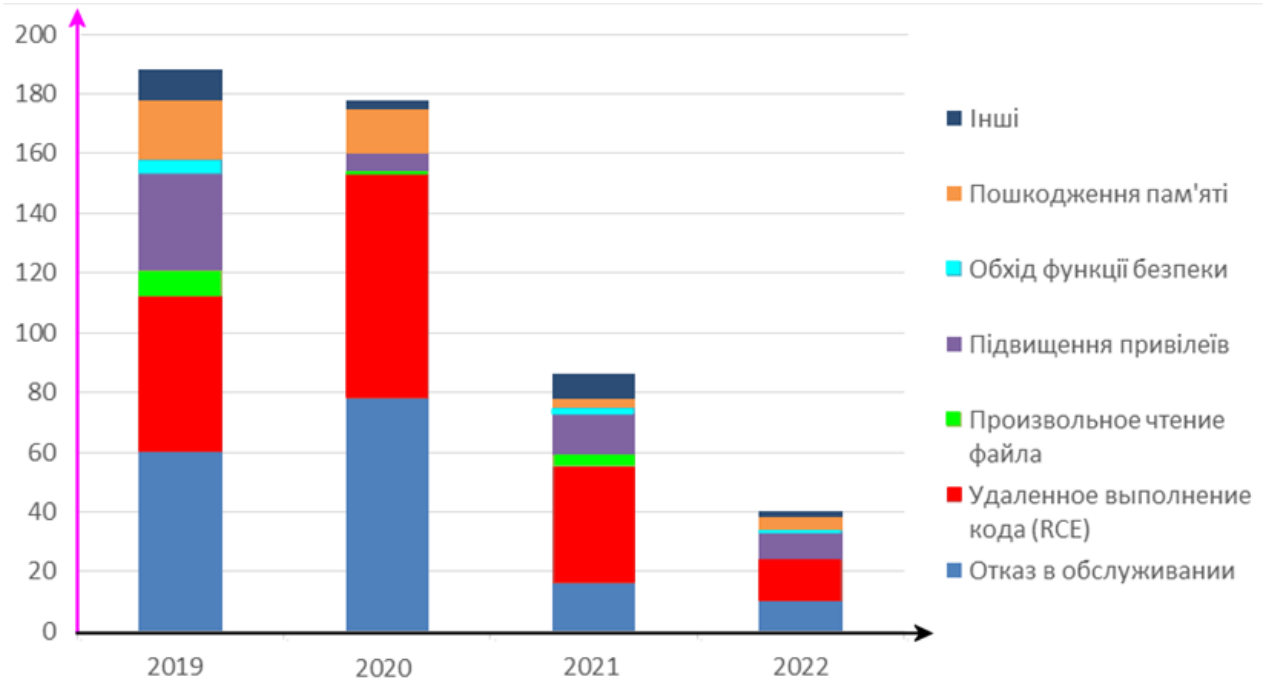


Рис. 4 – Використання та розподіл експлойтів за частотою використання вразливостей, що експлуатується *(за даними CVE)*

Оскільки експлойти мають своєю головною ціллю забезпечення виконання, потрібних для атакуючого, несанкціонованих дій на вразливій системі та/або пристрої, то вони можуть бути класифіковані по кінцевих об'єктах їх застосування [9, 12], наступним чином:

- Експлойти ОС. Вразливості ОС є точками входу для експлойтів, що можуть пошкодити роботу функціоналу пам'яті або призвести до некоректної роботи пристрою.
- Експлойти офісних програм. Вразливості офісного ПЗ найчастіше використовуються зловмисниками для того, щоб в наступному, скомпрометувати всю його систему, та поширити шкідливе ПЗ (*Cobalt Strike*) [16]. Використання каналу проникнення через вразливості офісних програм обумовлено значною поширеністю пакетів офісного ПЗ, практично для всіх типів існуючих платформ (ПК, хмарні додатки, гаджети тощо). При цьому не вчасно виконане оновлення такого ПЗ чи патчів безпеки, створює потрібні передумови для наступної атаки з використанням даної категорії експлойтів.
- Експлойти для спеціального ПЗ.
- Експлойти для Web-серверів, зокрема для найбільш поширених Інтернет браузерів (*Chrome, Firefox тощо*) або Web-сайтів (*facebook.com, livejournal.com*). Такого виду експлойти часто можуть не завдавати шкоди даних, що зберігаються на пристрої.

- Експлойти для апаратних прошивок. «Вдале» використання таких вразливостей виключає необхідність обходити будь-які засоби захисту ОС. При цьому, адаптація відповідного експлойта для різних ОС не складе труднощів, що обумовлено тим, що низькорівневі компоненти ядра для роботи з апаратним забезпеченням на багатьох ОС, практично однакові [17].

3. Висновки

Експлойти нульового дня, еволюційно породжують *N-day* експлойти (рис. 1), які складають суттєву частину загроз для вразливих пристроїв, саме через несвоєчасне оновлення ПЗ та/або ігнорування випусків (оновлення) відповідних патчів безпеки.

Віддалені експлойти «працюють» через мережу, використовуючи наявну вразливість у захисті без попереднього доступу до цільової вразливої системи. Локальні експлойти, навпаки, запускаються у вразливому пристрої (системі), однак потребують попереднього доступу до неї. Ці обидва різновиду експлойтів позиціонуються, переважно як засіб для отримання повноважень суперюзера.

Для ефективної експлуатації серверної вразливості, експлойту потрібно сформувати та надіслати на сервер відповідний запит, що містить шкідливий код, проте потрібно якось примусити (переконати) потенційну жертву підключитися до підробленого серверу. В цьому разі ефективним спойлером експлойту є використання одного із різновидів SE–атак.

Частіше всього експлойти поділяють за типом вразливості, що експлуатується при цьому, за сукупністю підтверджених випадків їх застосування, більшість експлойтів, які використовуються зловмисниками для здійснення потрібних їм несанкціонованих дій у вразливих системах, в своєї переважній більшості спрямовані на: - відмову в обслуговуванні, підвищення діючих повноважень (привілеїв) та віддалене виконання зловмисного коду.

Список літератури

[1] Богданова, Е., & Малахов, С. (2022). Обобщение специфики применения эксплойтов. Збірник наукових праць SCIENTIA, (Vol.2), 28-32. https://ojs.ukrlogos.in.ua/index.php/scientia/issue/view/24.06.2022/759 Available at: DOI 10.36074/scientia-24.06.2022

[2] Кохановська, Т., Нарежний, О., & Дьяченко, О. (2020). Дослідження можливостей технології Honeypot. Комп'ютерні науки та кібербезпека, 1(1), 33-42. https://doi.org/10.26565/2519-2310-2020-1-03

[3] Мелкозьорова, О., Лєсная, Ю., & Малахов, С. (2022). Особливості забезпечення захисту від НСД в сучасних інформаційних системах. InterConf, (97), 506-511. Вилучено із https://ojs.ukrlogos.in.ua/index.php/interconf/article/view/18428

[4] Погоріла, К., Лєсная, Ю., Богданова, Є., & Малахов, С. (2022). Соціальний інжиніринг, як фактор реалізації інсайдерських загроз. InterConf, (111), 494-501. Вилучено із https://interconf.top/documents/2022.06.6-8.pdf

[5] Гайкова, В., & Малахов, С. (2021). Аналіз факторів і умов реалізації кібербулінгу з урахуванням можливостей сучасних інформаційних систем. Комп'ютерні науки та кібербезпека, (1), 50-59. https://doi.org/10.26565/2519-2310-2021-1-04

[6] Чорна Т., Богданова Є., Погоріла К. Проблематика доксингу: - міжнародний досвід щодо забезпечення захисту персональних даних // Study of world opinion regarding the development of science. Proceedings of the IX International Scientific and Practical Conference. Prague, Czech Republic. 2022. Pp. 720-723 URL: https://isg-konf.com/study-of-world-opinion-regarding-the-development-of-science/ Available at: DOI: 10.46299/ISG.2022.2.9

[7] (2021). N-Day Exploit Protection Strategies. Вилучено із: http://surl.li/dsecc

[8] (2022). Предупреждение функции «Анализ сети» Avast: «Уязвимость для атаки WannaCry/DoublePulsar» Вилучено із: http://surl.li/dsecd

[9] (2017). Эксплойты, (Exploits). Вилучено із: http://surl.li/delac

[10] REMOTE SERVICES EXPLOITATION - DEFINITION, EXAMPLES, & PREVENTION – EXTRAHOP. Вилучено із: http://surl.li/dseci

[11] Ализар А. (2013). Каталог эксплойтов АНБ. Хакер, (280). Вилучено із: https://xakep.ru/2013/12/31/61833/

[12] Exploit Database. Вилучено із: http://surl.li/dseck

[13] Common Vulnerability Scoring System version 3.1: Specification Document. Вилучено із: https://www.first.org/cvss/examples

[14] 2022 CWE Top 25 Most Dangerous Software Weaknesses. Вилучено із: http://surl.li/

[15] (2022). Список 25 самых используемых эксплойтов. Вилучено із: http://surl.li/dseco

[16] Никитина Т. (2022). Фейковый PoC устанавливает на машину ИБ-экспертов Cobalt Strike Beacon. Вилучено із: http://surl.li/dsecy

[17] Лянин Чжао, Дэвид Ли. (2021). Аппаратные и программные решения: что опаснее? Вилучено із: https://www.osp.ru/os/2021/01/13055832

Received: October 2022. Accepted: November 2022.

Authors:
Yelyzaveta Bohdanova, 4rd year student, Faculty of Computer Science, Kharkiv National University named after V.N. Karazin, Kharkiv, Ukraine.
E-mail: xa12850323@student.karazin.ua
Chorna Tetiana, 4rd year student, Faculty of Computer Science, Kharkiv National University named after V.N. Karazina, Kharkiv, Ukraine.
E-mail: tatyanachernaya2002@gmail.com
Serhii Malakhov, Ph.D., Senior Researcher, Computer Science Department, V.N. Karazin Kharkiv National University, Ukraine.
ORCID ID https://orcid.org/0000-0001-8826-1616
E-mail: malakhov@karazin.ua

Overview of the current state of threats caused by the influence of exploits.

Annotation. The issue of exploiting the software vulnerabilities is considered in the article. Particular attention has been paid to the two aspects of the practical usage of exploits, as an attack tool and as a means of testing protected information systems. It is emphasized that most often exploits are divided by the type of security vulnerability exploited. Analysis of the known incidents related to the use of exploits, al-lows us to assert the existence of a relationship between the degree of popularity of a software product or device, and the probability of the exploits being created. Attention is drawn to the fact that N-day exploits constitute a significant part of existing security threats for vulnerable devices (systems). The main reason for this situation is untimely updating of the used software and ignoring updates of security patches. The extreme importance of the timely release of security patches as an effective means of preventing the usage of identified software vulnerabilities is emphasized. Releasing security patches is a basic element of possible defensive reactions when dealing with such issues. Attention is drawn to the fact that, according to the results of the analysis of known cases of illegal use of exploits (the last 3 years), they, in their vast majority, are aimed at 3 attack vectors: - denial of service; - illegitimate widening the current powers of managemention; - remote execution of malicious code.

Keywords: exploits; software vulnerabilities; security patches; information security.

УДК 681.04 DOI: 10.26565/2519-2310-2022-2-05

ПОРІВНЯННЯ КОМЕРЦІЙНИХ СКАНЕРІВ ВРАЗЛИВОСТЕЙ ВЕБ-ДОДАТКІВ ТА СКАНЕРІВ З ВІДКРИТИМ КОДОМ

Лахтін Іван, Дмитро Михайленко, Олексій Нарєжній

Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, Харків, 61022, Україна
lakhtin.ivan@gmail.com, xa12850318@student.karazin.ua, o.nariezhnii@karazin.ua

Надійшла: жовтень 2022. Прийнята: листопад 2022.

Анотація: *У роботі порівнюється вісім сканерів вразливостей на основі двох, навмисно вразливих додатків. Порівняння виконується за допомогою п'яти критеріїв: точність, відкликання, розрахунок індекса Юдена, веб-бенчмарк від WASSEC та OWASP. У якості додатків, що тестувалися обрані: OWASP WebGoat та Damn Vulnerable Web Application (DVWA). Серед досліджуваних сканерів є три комерційних сканера: Acunetix, HP WebInspect, AppScan, та п'ять сканерів з відкритим кодом такі, як: Arachni, IronWASP, Skipfish, OWASP ZAP, Vega. За результатами тестування зроблено висновок, що комерційні сканери є більш ефективними по ряду показників (в т.ч. переліку загроз). Деякі сканери з відкритим кодом (наприклад, ZAP та Skipfish) можна визначити, як початково таргетовані на певних видах загроз. Підкреслено, що не існує єдиного сканеру безпеки, який забезпечував би стабільно високі показники виявлення для всіх типів вразливостей. За результатами проведеного огляду стверджується, що існуючи відмінності в частоті хибнопозитивних вразливостей (для обох груп сканерів), обумовлені тим, що більшість комерційних рішень, мають автоматизовані сканери, які виявляються більш ефективнішими, ніж ручне налаштування з боку тестувальника. Вочевидь, що результати ручних налаштувань мають прямий зв'язок з фактичним рівнем компетенцій тестувальника, та в значній мірі обумовлюють кінцеві результати.*

Ключові слова: *веб-додаток; вразливість; атака, сканер; безпека; тестування.*

1. Вступ

Економічна важливість веб-додатків у багатьох сферах, включаючи банківську діяльність, транспорт, промисловість, бізнес та освіту, збільшила потребу в механізмах контролю та підвищення їх якості. Широке використання веб-додатків в усіх галузях сучасної економіки, нажаль, призвело до настільки ж різкого збільшення числа атак. Ці атаки, як правило, спрямовані на «слабкі місця» програмного коду додатків, їх недоліки та помилки, що обумовлює наявність передумов виникнення вразливостей в механізмах забезпечення конфіденційності, цілісності та доступності програмних рішень, що пропонуються [1]. За оцінками спеціалістів з питань інформаційної безпеки (ІБ), щодня атакується більше мільйона веб-сайтів, причому 75% з цих веб-сайтів містять не усунені вразливості [2]. Коли атаки досягають своєї мети, вони можуть призвести до компрометації критичних даних та/або програмно-апаратних ресурсів жертви, та мати інші серйозні наслідки, які можуть поширювати далеко за межи корпоративної інфраструктури компанії (наприклад, створення бот-мережі).

Розробники програмного забезпечення (ПЗ) використовують сканери вразливостей для автоматизації процесу перевірки стану ІБ «своїх» веб-додатків та/або проведення масштабних тестувань поточного стану безпеки багатьох інших відомих веб-додатків [3]. Широке поширення сканерів вразливостей і існуючи відмінності в їх функціональності, щодо виявлення вразливостей, підвищують інтерес відносно оцінки (*тестування*) ефективності їх роботи. Основними цілями подібних тестувань, є оцінка та порівняння продуктивності комерційних сканерів та рішень з відкритим кодом, що дозволяє дослідити реальні показники їх можливостей, стосовно виявлення відомих загроз ІБ. У цій роботі представлені результати порівняльної оцінки деяких функцій безпеки та продуктивності для восьми існуючих інструментів виявлення вразливостей.

На даний час існує достатньо проектів, які спрямовані на вивчення можливостей сканерів вразливостей веб-додатків, це можуть бути інтернет-блоги в сфері IT, або офіційні сайти якогось з інструментів безпеки. Але в цих випадках порівняння сканерів зводиться до пере-

ліку переваг і їх недоліків, короткого опису сканерів та порівняння ціни. Крім того існують більш серйозні - детальні тематичні дослідження, наприклад робота Alsaleh, M. та ін. *«Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners»* [3], в межах якої автори досліджують 4 сканери безпеки з відкритим кодом, аналізуючи відповідні показники виявлення загроз ІБ. У проекті Chen, S. *«WAVSEP 2017/2018 - Evaluating DAST against PT/SDL Challenges. Security Tools Benchmarking»* [4], розглядається близько двадцяти сканерів вразливостей, серед яких є як комерційні, так сканери з відкритим кодом. Вивчення результатів цих досліджень показує, що більшість з них зосереджені лише на впровадженні SQL та міжсайтовому сценарії атак.

2. Основна частина

2.1 Сканери в дослідженні

Архітектура сканерів вразливостей зазвичай поєднує чотири модулі: механізм сканування, базу даних (БД) сканування, модуль звітів та інтерфейс користувача. Механізм сканування виявляє вразливості безпеки щодо встановлених плагінів і порівнює результат із відомими сигнатурами. У БД сканування зберігається детальна інформація про відомі вразливості. Модуль звіту надає результати сканування з рекомендованими рішеннями для розробників та адміністраторів ІБ. Інтерфейс користувача забезпечує візуалізацію взаємодії користувачів зі сканерами. В межах проведеного оглядового дослідження сканерів розглянуто комбінацію з 8 програмних рішень (*як комерційні, так і з відкритим кодом*), що мають графічний інтерфейс користувача і працюють під управлінням ОС Windows:

1. *Acunetix* – комерційне рішення безпеки, що сканує веб-програми на наявність міжсайтових сценаріїв, ін'єкцій SQL та інших типів вразливостей. Використовує мультіпотоковий скан для безперервного обходу ряду веб-сторінок та створює різні форми відповідності вимогам і технічних звітів [5];
2. *WebInspect* – це комерційний інструмент тестування, який виявляє відомі і невідомі вразливості, включаючи міжсайтові сценарії і обхід каталогів у веб-додатках [6];
3. *AppScan* – це комерційна мережа, яка детектує та виправляє відомі вразливості [7];
4. *ZAP* – сканер з відкритим кодом та зручним інтерфейсом користувача, яке використовується для тестування на проникнення [8];
5. *Skipfish* – засіб з відкритим кодом, що надає інтерактивну мапу сайту для цільового сайту, виконуючи рекурсивний обхід та пошук словника. Створена мапа доповнюється результатами наступних тестувань. Підсумковий звіт, може слугувати основою для професійної оцінки стану безпеки веб-додатків [9];
6. *Arachni* – зручний сканер з відкритим кодом, забезпечує швидке сканування та пропонує різні варіанти інтерфейсу користувача, включаючи вихідні дані які представлені у HTML [10];
7. *Iron WASP* (*Iron Web Application Advanced Security Testing Platform*) – платформа тестування безпеки веб-додатків з відкритим кодом, яка постачається в різних комплектаціях «зовнішніх бібліотек», *IronPython, IronRuby* тощо [11];
8. *Vega* – автоматизований сканер з відкритим кодом для виявлення SQL ін’єкції та інших різновидів вразливостей безпеки [12].

В межах проведеного оглядового дослідження розглянути найбільш поширені сканери з відкритим кодом та комерційні сканери, відповідно до критеріїв консорціуму безпеки вебдодатків (*WASC*). Всі сканери перевірені на 2-х свідомо вразливих проектах *WebGoat* та

DVWA. Результати виявлення та продуктивність порівнювалися з використанням цільових показників (*точність, відкликання, Індекс Юдена, WBE і WASSEC*).

2.2 Показники продуктивності

2.2.1 Точність. OWASP визначає точність [13], як відсоток правильно виявлених вразливостей, як часку всіх зареєстрованих вразливостей (включаючи ті, що невірно позначені). Вираз для цієї метрики наведено нижче (1):

$$P = \frac{TP}{TP+FP} \quad , \tag{1}$$

де: - *TP* (*True Positive*), вірно виявлена вразливість; - *FP* (*False Positive*), хибно класифіковані вразливості.

Значення високої точності вказують на високу точність виявлення реальних вразливостей.

2.2.2 Відкликання. Відкликання – це кількість правильно детектованих вразливостей (R у виразі (2)), представлених, як частка всіх відомих вразливостей (*включаючи ті, що не були виявлені*),

$$R = \frac{TP}{TP+FN} \quad , \tag{2}$$

де: - *FN* (*False Negative*), це невизначена вразливість, яка насправді присутня, але сканер її не «помітив» (не виявив).

2.2.3 Індекс Юдена. Використовується для оцінки ефективності діагностичних тестів (3). Відображає значення в діапазоні [-1, 1], де: - значення «1» (*краще виявлення*) вказує на виявлення всіх вразливостей без помилкових спрацьовувань; - значення «-1» вказує лише на помилкові спрацьовування і відсутність справжніх спрацьовувань (*тобто, фактичні вразливості не виявлені*); - а індекс «0», означає, що отримано однаковий результат для веб-додатку з вразливостями, та для додатку без вразливостей (*тобто, неприпустимий результат*) [15].

$$J = \frac{TP}{TP+FN} + \frac{TN}{TN+FP} - 1 \quad , \tag{3}$$

де: - *TN* (*True Negative*), немає вразливостей, сканер підтверджує, що не виявив жодних.

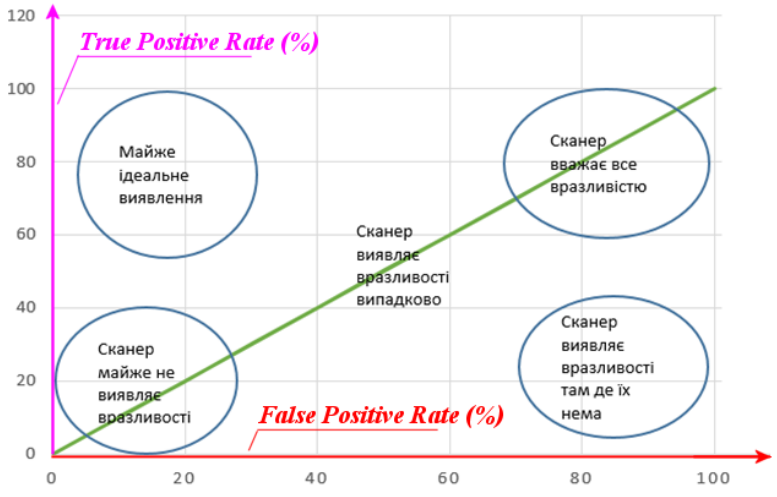


Рис. 1 – Приклад інфографіки представлення звіту WBE

2.2.4 Web Benchmark Evaluation (WBE). Проект *OWASP benchmark* запропонував систему оцінки ефективності інструментів статичного аналізу під назвою OWASP WBE, що являє собою візуальне уявлення ефективності виявлення на основі хибно-позитивних результатів та частоти відкликання.

Як слід із рис. 1, лінія, яка проходить через точки (0%, 0%) та (100%, 100%), є *«лінією вгадування»,* тобто продуктивність на цій лінії вказує на ту ж продуктивність, що і при випадковому виборі. Рис. 1 є прикладом того, наскільки коректно сканер, що тестується, визнає існуючи вразливості додатку [14].

2.2.5 Критерії оцінки сканера безпеки веб-додатків (WASSEC). Являють собою набір принципів для оцінки сканерів веб-додатків на предмет їх здатності ефективно тестувати веб-застосунки та виявляти вразливості безпеки. Основна мета WASSEC, це створення неза-

лежного від постачальників звіту, який допоможе фахівцям із ІБ, орієнтуватися під час оцінки параметрів сканерів веб-додатків. У цьому документі представлений повний перелік функцій, які слід враховувати при проведенні оцінки сканера безпеки веб-додатків. WASSEC поєднує такі можливості, як синтаксичний аналіз, обробку сеансів, тестування, звітність тощо [16], та складається з шести критеріїв оцінки (*див. Табл. 1*), які сприяють визначенню можливостей виявлення сканерів. В цілому, WASSEC надає користувачам можливості адаптивно (*вибірково*) використовувати наявний перелік функцій сканеру, та сфокусувати його на найбільш важливих для кожного користувача функціях, призначивши відповідну вагу для кожній функції [16].

Таблиця 1- Метрика WASSEC

КРИТЕРІЇ	ВРАЗЛИВОСТІ
1. Підтримка протоколу	*Get, Post, Cookie, Header, Secret, Pname, Custom, Proxy, Gzip, Eflate, Ssl.*
2. Автентифікація	*Basic, Digest, Ntlm, Ntlmv2, Kerberos, Form, Cert, Captcha*
3. Управління сеансами	*Custom Cookie, Custom, Header, Logout, Detection, Exclude, Log-Out, Exclude, Url, Exclude, Param*
4. Crawling	*Manual Crawl, Html Crawler, Ajax Crawler, Flash Crawler, Applet Crawler, Silverlight Crawler, Wsdl Crawler, Rest Crawler, Field Autofill, Smart Autofill, Anti Csrf Support, Viewstate Support*
5. Parsing	*Xml, Xmlatt, Xmltag, Json, Netenc, Amf, Javaser, Netser, Wcf, Wcf-Bin, Websock, Dwr, Url File*
6. Тестування	*Sqli, Bsqli, Ssjsi, Rxss, Pxss, Dxss, Jsonh, Lfi, Rfi, Cmdexec, Upload, Redirect, Crlfi, Ldapi, Xpaphi, Mxi, Ssi, Formati, Codei, Xmli, Eli, Buffero, Integero, Codedisc, Backupf, Padding, Authb, Prive, Xxe, Session, Fixation, Csrf, Ados.*

2.3 Результати перевірки вразливих додатків

На рис. 2 представлені *TP* (*True Positive*) оцінки сканерів для 7 вразливостей у DVWA та відповідно, 9 вразливостей у WebGoat. Для випадку DVWA, хоча всі сканери і виявили вразливості CmdExec, XSS, XSS та SQLi, однак їх пошукова продуктивність значно відрізнялася. При цьому, відмінності в показниках виявлення DVWA можна пояснити тим, що окремі сканери розроблялися для пошуку конкретних типів вразливостей, причому правила ліцензування, опосередковано, впливають на кінцевий результат: наприклад, безкоштовна версія *Acunetix*, виявляє тільки XSS-уразливості. Крім того, можливості виявлення сканерів

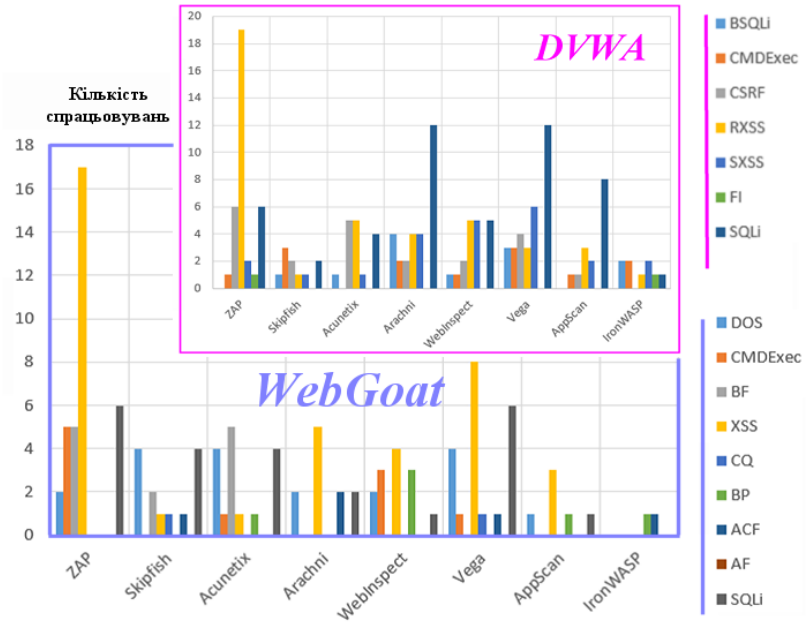


Рис. 2 – *TP*- співвідношення спрацьовувань для DVWA та WebGoat

помітно різняться в залежності від веб-додатків.

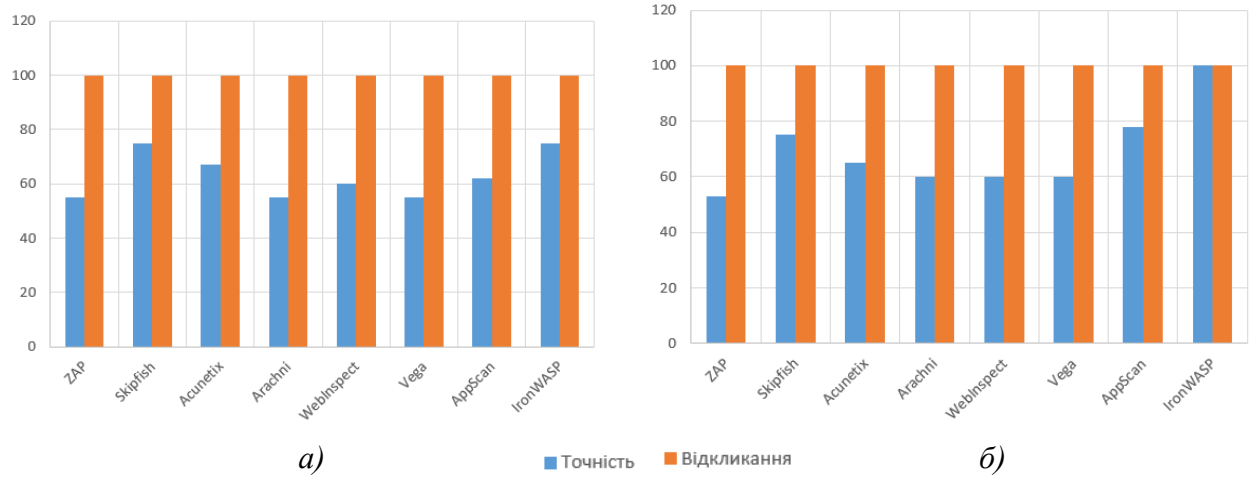
Для випадку WebGoat, всі рішення, за винятком IronWASP (*знайшов лише 2 вразливості*), виявили по кілька вразливостей, причому найбільш вдалого детекту зазнали XSS та SQLi. І хоча жоден сканер не зміг детектувати весь набор WebGoat, їх відмінності є чітким свідченням того, що ці рішення впроваджують різни стратегії їх побудови. В цілому, отримані результати дають наочне уявлення про загальні збіги та взаємодоповнюючі риси сканерів безпеки, що тестувались.

2.3.1 Час сканування. Ефективність сканерів оцінювалась, в т.ч., за параметром часу (в секундах), що потрібен для завершення виявлення вразливостей: - для DVWA, від 30 до 360 сек; - для WebGoat, від 30 до 900 сек (*див. табл. 2*). Відмінності в часі виявлення окремих сканерів, також, можуть бути пов'язані з внутрішніми компонентами безпеки додатків. Так наприклад, якщо для сканеру ZAP, у WebGoat, знадобилося лише 60 сек, то в випадку DVWA, він працював вже 360 сек.

Таблиця 2 – Час сканування вразливих додатків

Тип Сканеру	**Час сканування** *(сек)*	
	DVWA	**Web Goat**
ZAP	*360*	*60*
Skipfish	*120*	*120*
Acunetix	*122*	*120*
Arachni	*60*	*900*
WebInspect	*180*	*181*
Vega	*60*	*60*
AppScan	*62*	*70*
Iron WASP	*60*	*30*
	CPU Intel Core i5-6200U; 2.3 ГГц; 8 Gb RAM; OS Win 10 Home	

2.3.2 Аналіз точності та «відкликання» сканерів. Ці параметри оцінювалися в діапазоні 0÷100%, де: - ефективний інструмент (*без хибно-негативних або хибно-позитивних результатів*), має значення 100%, як для точності, так і для оцінки його «відклику». На рис. 3 наведені значення (в %) для DVWA та WebGoat відповідно.



а) *б)*

Рис. 3 – Точність та відклик для DVWA *(а)* та WebGoat *(б)*

Хоча всі сканери і досягли 100%-го показника відкликання (*що підтверджує їх здатність виявляти реальні вразливості*), однак існують значні відмінності в їх показниках точності, що скоріш за все, можна пояснити «унікальністю» конструктивної реалізації кожного сканеру. Таким чином, показники, котрі менше 100% відображають той факт, що сканери

безпеки маркували «*як уразливості*» певні проблеми, які насправді не були такими (*тобто, це приклад помилкових спрацьовувань*).

2.3.3 OWASP WBE. Довідник з тлумачення результатів OWASP WBE забезпечує графічну візуалізацію ефективності інструментів тестування, зіставляючи його *TP* результат із частотою хибно-позитивних (*FP*) результатів (*див. рис. 4*). В ході тестування, у відповідності з виразами (4) і (5), визначається загальна частота TP_t та FP_t результатів, як загальне число, для DVWA та WebGoat:

$$TP_t = TP_D + TP_W\,, \tag{4}$$

$$FP_t = FP_D + FP_W\,, \tag{5}$$

де: - TP_t та FP_t, це загальні показники істинних і хибно-позитивних результатів;

- TP_D та FP_D, це показники істинних і хибно-позитивних результатів для випадку DVWA;
- TP_W і FP_W, це показники істинних і хибно-позитивних результатів для випадку WebGoat.

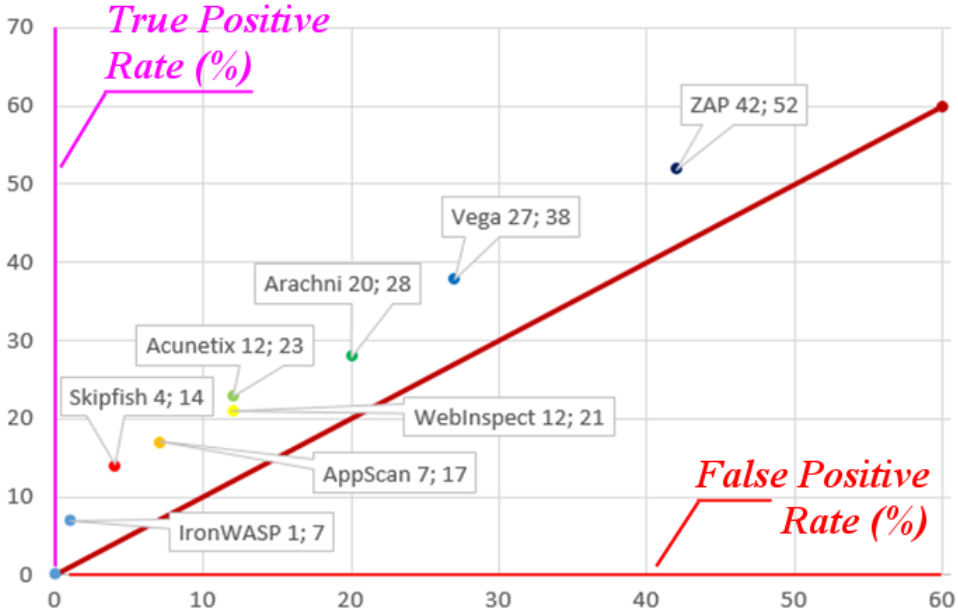


Рис. 4 – Інтерпретації WBE для досліджуваних сканерів

Як тлумачиться в посібнику WBE (розділ 4.4) [14], ефективність детектування вразливостей для того чи іншого сканеру, визначається його положенням на даної інфографіки. Так, наприклад, позиція IronWasp відповідає категорії «ніщо не вразливе» (*де істинні і хибно-позитивні показники невеликі*). Продуктивність цього сканеру можна обґрунтувати тим, що це рішення було розроблено для певного типу виявлення вразливостей.

Щодо позиції ZAP, у верхньому правому куті, то цей сканер виявляє та повідомляє, що *«все вразливе»* (справжні та хибно-позитивні показники, високі). Решта сканерів, згідно із пропонованою інтерпретацією результатів, потрапили до категорії *«інструмент повідомляє, що ніщо не вразливе»*, за винятком Arachni - (20;28), що є надто близький до категорії *«інструмент повідомляє про вразливість випадковим чином»*.

2.3.4 Індекс Юдена. На рис. 5 представлений Індекс Юдена для досліджуваних сканерів безпеки. Згідно з представленими даними, IronWASP має найвищий індекс (0,83), що вказує на його високу ефективність у виявленні відомих вразливостей, з невеликою кількістю помилкових спрацьовувань або взагалі без них.

Наступними сканерами з найкращими індексами є, Skipfish, Appscan, Webinspect і Acunetix (0,45, 0,31, 0.23 і 0,21 бали). Крім того, результати свідчать, що кілька сканерів з відкритим кодом, можуть функціонувати так само ефективно, як і деякі комерційні рішення.

Тобто, сам факт ліцензування не є аргументованим показником для оцінки ефективності інструментів тестування вразливостей.

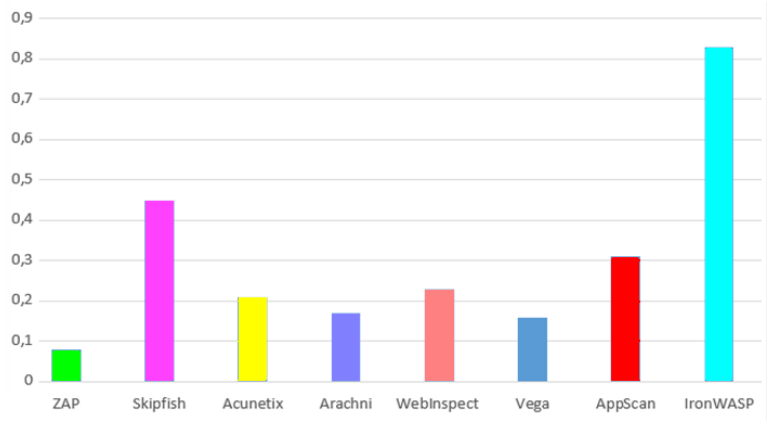


Рис. 5 – Індекс Юдена

Наступними сканерами з найкращими індексами є, Skipfish, Appscan, Webinspect і Acunetix (*0,45, 0,31, 0.23 і 0,21 бали*). Крім того, результати свідчать, що кілька сканерів з відкритим кодом, можуть функціонувати так само ефективно, як і деякі комерційні рішення. Тобто, сам факт ліцензування не є аргументованим показником для оцінки ефективності інструментів тестування вразливостей.

2.3.5 Критерії оцінки сканера безпеки веб-додатків (WASSEC). У таблиці 3 наведені результати виявлень за критеріями WASSEC [16] для тестованих сканерів. Згідно з ними сканер Acunetix має кращу підтримку протоколу, потім Appscan та Skipfish. Незважаючи на відмінності в продуктивності сканерів, є подібності в області обходу, автентифікації та тестування. На рис. 6 наведені середні результати WASSEC, що визначають пару лідерів з кращою продуктивністю, це Acunetix та AppScan (*0,81 та 0,65*). При цьому, сканери з відкритим кодом Skipfish та ZAP (*третє і четверте місце, з хорошими показниками та оцінками 0,43 і 0,40*), підтверджують високе реноме, як для не ліцензованих рішень.

Таблиця 3 – Результати виявлень WASSEC

Сканер	Підтримка протоколу	Управління сеансами	Тестування	Parsing	Автентифікація	Crawling
ZAP	*7*	*5*	*12*	*2*	*3*	*4*
Skipfish	*8*	*5*	*13*	*3*	*3*	*4*
Acunetix	*10*	*6*	*28*	*7*	*7*	*9*
Arachni	*6*	*5*	*12*	*2*	*3*	*3*
WebInspect	*7*	*6*	*4*	*0*	*7*	*1*
Vega	*6*	*5*	*10*	*2*	*3*	*3*
AppScan	*8*	*6*	*22*	*4*	*6*	*8*
IronWASP	*6*	*5*	*9*	*3*	*3*	*2*

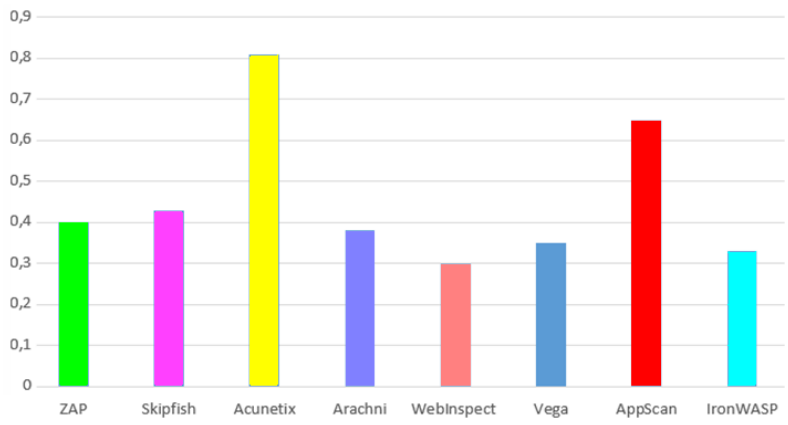


Рис. 6 – Середні значення WASSEC

7. Висновки

1. Сканери з відкритим кодом, як і їх комерційні рішення, є ефективним інструментом виявлення вразливостей у веб-додатках. Основні відмінності між цими двома групами сканерів полягають в різних показниках їх точності (*помилкових виявленнях*), що свідчить про необхідність оцінки ефективності використовуваних інструментів, на основі найменшої кількості хибно-позитивних результатів. При цьому слід враховувати, що різні сканери по різному виявляють різні вразливості (*тобто схильні для певного типу вразливостей*), наприклад: – Acunetix, проявив гарні здібності у виявленні XSS вразливостей, а OWASP ZAP, навпаки, у виявленні вразливостей типу CmdExec.

2. Результати проведеного огляду дають підстави стверджувати, що комерційні сканери Acunetix та Appscan, є більш ефективними при виявленні розглянутого складу вразливостей, але сканери з відкритим кодом (*наприклад, ZAP та Skipfish*), були однаково ефективними при виявленні деяких різновидів вразливостей, що розглядалися (тобто, є початково таргетовані на цих загрозах).

3. Не існує єдиного сканеру, який забезпечував би стабільно високі показники виявлення для всіх типів вразливостей. На прикладі проведеного огляду результатів тестування, можна стверджувати, що існуючи відмінності в частоті хибно-позитивних вразливостей (*для обох груп сканерів*), обумовлені тим, що більшість комерційних рішень, мають автоматизовані сканери і обхідники, котрі виявляються більш ефективнішими, ніж ручне налаштування та втручання тестувальника, яке необхідне у випадку використання сканерів з відкритим кодом і, що потребує певних додаткових компетенцій від персоналу.

Список літератури

[1] Amankwah, R., Chen, J., Kudjo, P. K., & Towey, D. (2020). An empirical comparison of commercial and open-source web vulnerability scanners. Software: Practice and Experience, 50(9), 1842–1857. https://doi.org/10.1002/spe.2870

[2] El, M., McMahon, E., Samtani, S., Patton, M., & Chen, H. (2017). Benchmarking vulnerability scanners: An experiment on SCADA devices and scientific instruments. 2017 IEEE International Conference on Intelligence and Security Informatics (ISI). https://doi.org/10.1109/isi.2017.8004879

[3] Alsaleh, M., Alomar, N., Alshreef, M., Alarifi, A., & Al-Salman, A. (2017). Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners. Security and Communication Networks, 2017, 1–14. https://doi.org/10.1155/2017/6158107

[4] Chen, S. (2017, November 10). WAVSEP 2017/2018 - Evaluating DAST against PT/SDL Challenges. Security Tools Benchmarking. https://sectooladdict.blogspot.com/.

[5] Acunetix | Web Application Security Scanner. (2022). Acunetix. https://www.acunetix.com/

[6] The leader in Web application security assessment. (2011). Retrieved November 4, 2022, from http://www.hp.com/hpinfo/newsroom/press_kits/2011/risk2011/HP_WebInspect_data_sheet.pdf

[7] HCL Software. (2021). Hcltechsw.com. https://www.hcltechsw.com/appscan

[8] OWASP ZAP Tutorial: Comprehensive Review Of OWASP ZAP Tool. (2022). Www.softwaretestinghelp.com. https://www.softwaretestinghelp.com/owasp-zap-tutorial/

[9] skipfish. (2017). Kali.tools. Retrieved November 4, 2022, from https://kali.tools/all/?tool=1256

[10] Arachni - Web Application Security Scanner Framework. (n.d.). Arachni - Web Application Security Scanner Framework. https://www.arachni-scanner.com/

[11] IronWASP - Инструменты Kali Linux. (n.d.). Retrieved November 4, 2022, from https://kali.tools/?p=1786

[12] Vega Vulnerability Scanner. (n.d.). Subgraph.com. https://subgraph.com/vega/

[13] OWASP. (n.d.). OWASP foundation, the open source foundation for application security. Owasp.org. https://owasp.org/

[14] OWASP Benchmark. (n.d.). Owasp.org. https://owasp.org/www-project-benchmark/

[15] Youden, W. J. (1950). Index for rating diagnostic tests. Cancer, 3(1), 32–35. https://doi.org/3.0.co;2-3">10.1002/1097-0142(1950)3:1<32::aid-cncr2820030106>3.0.co;2-3

[16] The Web Application Security Consortium / Web Application Security Scanner Evaluation Criteria. (2009). Projects.webappsec.org. http://surl.li/dtuti

Received: October 2022. Accepted: November 2022.

Authors:

Ivan Lakhtin, student Computer Science Department (magistrate), V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.

E-mail: lakhtin.ivan@gmail.com

Dmytro Mykhailenko, CSD Student (bachelor degree), V. N. Karazin Kharkiv National University, Kharkov, Ukraine.
E-mail: xa12850318@student.karazin.ua

Oleksii Nariezhnii, Ph.D., Associate Professor, V. N. Karazin Kharkiv National University, Kharkov, Ukraine.
ORCID ID https://orcid.org/0000-0003-4321-0510
E-mail: o.nariezhnii@karazin.ua

Comparison of commercial web application vulnerability scanners and open source scanners.

Abstract. The paper compares eight vulnerability scanners based on two intentionally vulnerable applications. The comparison is performed using five criteria: accuracy, recall, Juden index calculation, web benchmark from WASSEC and OWASP. OWASP WebGoat and Damn Vulnerable Web Application (DVWA) are selected as the tested applications. Among the tested scanners there are three commercial scanners: Acunetix, HP WebInspect, AppScan, and five open source scanners such as: Arachni, IronWASP, Skipfish, OWASP ZAP, Vega. According to the results, it was concluded that commercial scanners are more effective in a number of criteria (including the list of threats). Some open source scanners (such as ZAP and Skipfish) can be characterized as originally targeted at certain types of threats. It is emphasized that there is no single security scanner that provides consistently high detection rates for all types of vulnerabilities. Based on the results of the review, it is claimed that the existing differences in the frequency of false-positive vulnerabilities (for both groups of scanners) are due to the fact that most commercial solutions have automated scanners, which are more effective than manual settings by the tester. It is obvious that the results of manual settings have a direct relationship with the actual level of the tester's competence, and largely determine the final results.

Keywords: web application, vulnerability, attack, scanner, security, testing.

ISSN 2519-2310 DOI: 10.26565/2519-2310-2022-2

Journal Requisites

Publishing, cover design: V.N. Karazin Kharkiv National University, 2022

Наукове видання

КОМП’ЮТЕРНІ НАУКИ ТА КІБЕРБЕЗПЕКА

Випуск 2(22) 2022

Міжнародний електронний науково-теоретичний журнал

Англійською, українською, *та ін. мовами*

Комп’ютерне верстання – Федоренко В.В., Єсіна М.В.

61022, Харків, майдан Свободи, 6
Харківський національний університет імені В.Н. Каразіна

V. N. Karazin Kharkiv National University Publishing

CS&CS

2022