УДК 004.422.632.2(045)

Дворська Л.О.

АНАЛІЗ СТІЙКОГО ШВИДКОГО АЛГОРИТМУ ВІДПОВІДНОСТІ РЯДКІВ ДЛЯ ЗАХИСТУ БЕЗПЕКИ МЕРЕЖІ

Інститут комп'ютерних технологій
Національного авіаційного університету

Розглянуто проблему відповідності рядка у сфері мережевої безпеки, особливо проблему боротьби з алгоритмічними ефективними атаками. Запропоновано новий стійкий і швидкий алгоритм відповідності рядків, що базується на основі алгоритму Бойєра-Мура –найбільш ефективного алгоритму відповідності рядка

Введення

Відповідність рядка є одним із основних алгоритмів у області мережевої безпеки. Швидкий алгоритм відповідності рядка може зробити значний внесок у розвиток багатьох інших наукових сфер. Запропоновано новий стійкий і швидкий алгоритм відповідності рядків (СШР), що базується на основі алгоритму Бойєра-Мура (БМ) – найбільш ефективного алгоритму відповідності рядка у звичайних додатках. СШР використовує вдосконалену евристику поганих символів, щоб досягти більшої величини області зсуву, а також вдосконалену евристику хороших суфіксів, щоб значно поліпшити продуктивність у найгіршому варіанті виконання алгоритму. Обидві евристики в поєднанні з новою визначальною умовою перемикання між ними дозволять СШР досягнути більш високої продуктивності, ніж БМ, як у звичайних, так і в найгірших умовах. Експериментальні результати свідчать про те, що СШР виявляється набагато ефективнішим алгоритмом, ніж БМ у гірших умовах, і чим довший зразок, тим більше показників покращення. Робочі показники СШР на 7,57 ~ 36,34% вищі, ніж показники БМ, при пошуку англійського тексту, на 16,26 ~ 26,18% вищі, ніж показники БМ, при пошуку рівномірно випадкового тексту, та на 9,77% вищі, ніж показники БМ, при реальному пошуку встановленого зразка *Snort*.

Snort є однією із систем розпізнання атак *IDS* (*Intrusion Detection Systems*), які дозволяють виявити можливі способи вторгнення ще до того, як вони відбудуться. *Snort* працює з попередньо заданими шаблонами шкідливого трафіку, які називаються правилами (*rules*) і дозволяють визначити, який трафік в мережі є шкідливим, а який – ні. Це схоже на антивірусні програми, оскільки правила потрібно періодично обновлювати. *Snort* може виявляти лише відомі атаки. Але можна займатись створенням правил самостійно, що допоможе глибше інтегрувати *Snort* в мережу.

Рядок відповідності є одним із основних і важливих предметів дослідження в області комп'ютерної науки. Рядок відповідності полягає в пошуку одного, або більш в загальному, всіх випадків входження шуканого рядка, який називається зразком, у вхідний рядок. Якщо більш ніж один шуканий рядок зіставляється навпроти вхідного рядка одночасно, це – множинне зіставлення зі зразком. В іншому випадку він називається єдиним зіставленням зі зразком. Розглянемо алгоритм єдиного зіставлення зі зразком.

Єдине зіставлення зі зразком широко використовується в середовищах мережевої безпеки. У сфері мережевої безпеки, зразок є рядком, який вказує на мережеві вторгнення, атаки, віруси, спам або брудну мережеву інформацію та ін. Наприклад, в *Snort*, відомих відкритих джерелах легких *NIDS* [1, 2], до алгоритму Бойєра-Мура (БМ) [3, 15, 16] звертаються, коли кількість зразків, які повинні бути порівняні, менше п'яти. Алгоритм єдиного зіставлення зі зразком також є основою для побудови основаного на виключенні алгоритму зіставлення зі зразком та механізму гібридного зіставлення зі зразком для обробки вели-

чезних структур, що визначають мережеву безпеку.

Алгоритм зіставлення зі зразком, оснований на виключенні, використовує евристику для виявлення зразків, які могли не входити у вхідний рядок спочатку, а потім використовує алгоритм єдиного зіставлення зі зразком, щоб підібрати зразки, які можуть не бути виключені. Алгоритми $E{\times}B$[4] та $E^2{\times}B$[5] є типовими алгоритмами на основі виключення. Механізм змішаного зіставлення зі зразком запускає різні алгоритми, зазвичай поєднує алгоритми єдиного та множинного зіставлення зі зразком в залежності від різних середовищ та умов застосування, таких як кількість зразків і розмір вхідного рядка. З огляду на той факт, що ні один алгоритм не виконує цих функцій найкраще у всіх випадках, механізм змішаного зіставлення зі зразком видається найкращим методом у додатках мережевої безпеки [6, 7].

У процесі розвитку технологій мережевих атак навіть устаткування мережевої безпеки стає об'єктом атак. Не є виключенням і алгоритм відповідності рядка. Ефективним методом атаки на алгоритм відповідності рядка є «алгоритмічна ефективна атака»: зловмисник навмисно забезпечує введення, які перевантажують або призводять до виконання найгіршого сценарію алгоритму [6]. Це може сповільнити пошук і стати причиною втрати пакетів, які зловмисник може перехопити. Наприклад, час обробки *Snort* може бути сповільнений у 25 разів під впливом такого нападу [8]. Вдосконалення ефективності виконання алгоритму відповідності рядка одночасно за умов середнього та найгіршого виконання сценарію фактично захистить алгоритм від алгоритмічного ефективного нападу. Алгоритм БМ можна віднести до найбільш ефективних алгоритмів відповідності рядка у звичайних додатках, а також вважати, що він забезпечує найкраще виконання середнього сценарію виконання будь-яких відомих алгоритмів [6, 13].

Зосереджено увагу на поліпшенні виконання БМ одночасно за умов середнього та найгіршого виконання сценарію. Запропоновано новий стійкий швидкий алгоритм відповідності рядка (СШР).

Аналіз алгоритму Бойєра-Мура

Розглянемо алгоритм БМ. Відповідність рядка має справу з рядком n $T= t_1t_2 \ldots t_n$ (вхідний рядок), розмір якого дорівнює n, алгоритм досліджує його на всі випадки входження іншого, коротшого рядка $P= p_1p_2 \ldots p$ (шуканий рядок), розмір якого дорівнює m ($n > m$). Обидва рядки формують кінцевий набір символів, який називається алфавітом і позначається символом Σ.

Алгоритми відповідності рядка сканують рядок T за допомогою механізму «висувного віконця». Розмір віконця зазвичай дорівнює m. По відношенню до контрольної точки $j(1\leq j \leq n)$, вікно знаходиться на сегменті t_jt_{j+1} t_{j+m-1} . На кожному сегменті перевірки символи віконця порівнюються з символами в рядку пошуку. Після повної відповідності або невідповідності, вікно зміщується вздовж рядка T в залежності від евристики кожного алгоритму.

БМ використовує дві евристики, поганий символ і хороший суфікс з метою скорочення кількості зіставлень (у порівнянні з алгоритмом послідовного (прямого) пошуку відповідності рядка. Пошук здійснюється за допомогою переміщення (зсуву) вікна зліва направо, при цьому точка t_1 є першою контрольною точкою. В межах кожного сегменту перевірки символи віконця і рядка P порівнюються справа наліво. Обидві евристики запускаються при невідповідності. Припустимо, порівняння здійснюється між рядком t_{i+1} t_{i+2} $\ldots t_{i+m}$, $0\leq i\leq n\text{-}m$ i рядком $p_1p_2 \ldots p_m$, при

$P_{j+1}+p_{j+2}+\ldots p_m=t_{i+j+1}+t_{i+j+2}+t_{i+m}$,

$1\leq j\leq m\text{-}1$, невідповідність відбудеться при P_j, $P_j\neq t_{i+j}$. Евристика поганого символу діє таким чином: у разі невідповідності віконце зміщується вправо, тобто символ невідповідності t_{i+j} рядка T, суміщається з сегментом P_{kl}, де $k1=\max\{k2|p_{k2}=t_{i+j}, 1\leq k2\leq j\}$. Якщо символ t_{i+1} не з'являється в рядку$P_1p_2\ldots p_{j-1}$ вікно зсувається на позицію назад, таким чином, що сегмент p_1 накладається на сегмент t_{i+j}.

Евристика хорошого суфікса діє інакше: коли відбувається невідповідність, то позначається не порожній суфікс. Вікно зсувається до наступного входження суфіксу $p_1p_2\ldots p_{m-1}$. Якщо входжень суфікса нема, вікно зсувається на позицію, так що сегмент p_{k3} суміщається з сегментом t_{i+m}, де

$$k3 = \max\{k4 \mid p_1p_2\ldots p_{k4} = t_{i+m-k4+1}\ldots t_{i+m}\},$$

при цьому $1 \le k4 \le m - j - 1$. Якщо символ $k3$ відсутній, вікно зсувається на одну позицію, так що сегмент p_1 накладається на сегмент t_{i+m}.

БМ здійснює багато зсувів, що зумовлюються двома евристиками.

Аналіз стійкого швидкого пошуку відповідності рядка

Алгоритм СШР використовує вдосконалену евристику поганого символу і евристику хорошого суфікса з метою покращення ефективності виконання БМ одночасно в середньому й гіршому випадках. Вдосконалена евристика поганого символу алгоритму СШР показана на рис.1. Передостанній символ справа поточного «віконця» завжди використовується в якості поганого символу. Наприклад, (рис.1) в поточній контрольній точці, позначеній штриховою лінією у «віконці», використовується символ «*s*» із рядка *T* як «поганий символ», для того, щоб визначити величину зсуву. У порівнянні з евристикою поганого символу в БМ, це забезпечує збільшення значення області зсуву та легко виконується [14].

Основний недолік БМ полягає в тому, що після зсуву він забуває інформацію про символи, які вже зіставлені. Таким чином, у ситуації найгіршого випадку, він поводить себе так само, як алгоритм послідовного (прямого) пошуку.

Пропонується нова вдосконалена евристика хорошого суфікса, яка може за потребою запам'ятовувати символи, що вже зіставлені.

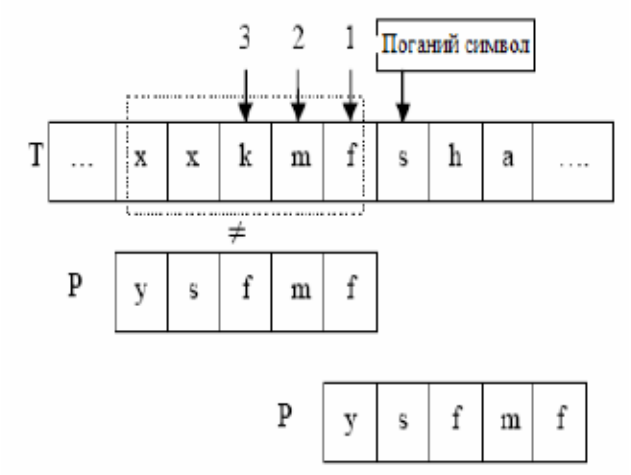

Рис.1. Вдосконалена евристика поганого символу алгоритму СШР

Вдосконалена евристика хорошого суфікса полягає в наступному: коли потрібно запам'ятати інформацію символів у визначеній контрольній точці, які вже зіставлені, у цій контрольній точці використовується лише евристика хорошого суфіксу. У порівнянні з політикою БМ (в кожній контрольній точці завжди підраховується значення зсуву хорошого суфіксу і поганого символу і обирається більше значення), ми можемо легко запам'ятати символи, вже зіставлені, за допомогою змінних і не порівнювати їх знову в наступній контрольній точці. Якщо значення зсуву (*shift*) (рис. 2) розраховується евристикою хорошого суфіксу в поточний момент перевірки, то

$$p_0p_1\cdots p_{m-shift-1} = p_{shift}p_{shift+1}\cdots p_{m-1}.$$

Використаємо змінну *offset = m - shift* для запам'ятовування початкової позиції в наступній контрольній точці. Далі в наступній контрольній точці потрібно порівняти лише p_{offset} …p_{m-1}.

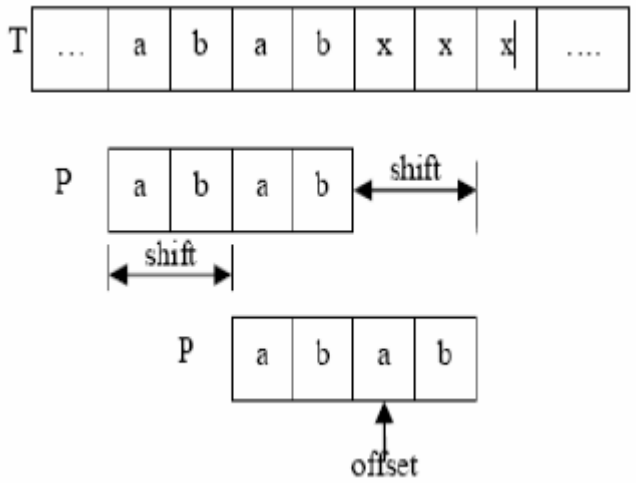

Рис.2. Вдосконалена евристика хорошого суфікса СШР Для збереження простоти реалізації та досягнення більшого значення зсуву пропонується проста визначальна умова для визначення, чи потрібно використовувати вдосконалену евристику хорошого суфіксу з метою запам'ятовування інформації символів, які вже зіставлені.

За умови найгіршого випадку, поведінка БМ така ж, як і в алгоритму послідовного (прямого) пошуку (*the Brute Force*). Це означає, що значення зсуву завжди дорівнює одиниці. Тому, якщо при реалізації СШР значення зсуву, визначене за допомогою вдосконаленої евристики поганого символу, дорівнює одиниці в поточній контрольній точці, то використовується вдосконалена евристика хорошого суфіксу. Це має дві переваги. По-перше, можна отримати довший зсув, оскільки значення зсуву, що розраховується за допомогою евристики хорошого суфіксу, не може бути менше одиниці. По-друге, можна зменшити кількість символів, яким необхідна перевірка в наступній контрольній точці, тому що вдосконалена евристика хорошого суфіксу запам'ятовує вже зіставлені символи.

В даній роботі розглядається часова характеристика складності алгоритму. Часову складність будемо підраховувати у виконуваних командах: кількість арифметичних операцій, кількість порівнянь, пересилань (в залежності від алгоритму). [17].

Процедура пошуку запропонованого алгоритму СШР показана далі. Скористаємося двома таблицями для зберігання значення зсуву, розрахованого двома евристиками: СШРПс[] (*RQSBc[]*) для поганого символу та СШРХс[] (*RQSGs*) для хорошого суфіксу. Звернімо увагу, що *P* – це шуканий рядок, *m* –довжина шуканого рядку, *T* – вхідний рядок, *n* –довжина вхідного рядку.

Процедура пошуку СШР

```
If (n<m) do exit;
j = 0; // j представляє контрольну
точку
offset = 0;
while (j <= n - m) do begin
i = m-1;
if (RQSBc[T[j + m]] > 1) do // вико-
нистання евристики поганого символу
СШР
while(P[i] = T[j+i] and i>=0) do begin
i = i-1;
end while
j = j+RQSBc [T[j + m]];
else // using RQS good suffix heuristic
while (P[i] = T[i + j] && i >= offset)
do begin
i = i-1;
end while
if (i < offset) do // виявлено відповідність
while
if (i < offset) do // виявлено відповідність
j = j+RQSGs[0];
offset = m-shift; //запам'ятовується
символ
already matched
else do
j = j+RQSGs[i];
offset = 0;
end while
```

Експерименти та результати досліджень

Виконано алгоритм СШР і здійснено порівняння його ефективності з алгоритмом БМ. Коди БМ є високоякісним впровадженням і використані в [13].

Експериментальне середовище

У всіх експериментах довжина вхідного рядка *T* є 16Кб. Проведено експерименти з трьома видами вхідного рядка *T*.

Перший складається із того ж символу '*A*'

для випробування тесту найгіршого випадку. Другий – це частина англійського тексту, взятого із інструкції використання програмного забезпечення. І третій рядок складається з рівномірно випадкових символів з 256-символьного алфавіту.

В експериментах використовувався персональний комп'ютер з 2,4 ГГц процесором на базі *Intel Pentium ® 4*, з кешпам'яттю 8KB *L*1, 512 КБ *L*2 і 512MB оперативної пам'яті. Операційна система – *Microsoft ® Windows XP Professional*.

Використовується компілятор *Microsoft ®Visual C* + + 6.0.

Випробування найгіршого випадку виконання алгоритму

Вхідний рядок і рядок пошуку складаються з однакового символьного ряду, значення зсуву завжди дорівнює одиниці, а число відповідностей – максимальне. Це забезпечує варіант виконання найгіршого випадку для алгоритму відповідності рядка. На практиці використовуються наступні ключові слова, наприклад, *blowdetection rule Snort.*

alert tcp any any -> any any (ack:0; flags:SFU12;

content:"AAAAAAAAAAAAAAAA"; depth: 16;).

Алгоритм намагається знайти ключове слово '*AAAAAAAAAAAAAAAA*' у всьому потоці протоколу *TCP*.

T = AA ... A і *C = AA ... A* були використані для перевірки ефективності алгоритмів СШР і БМ. Результати досліджень показані на рис.3.

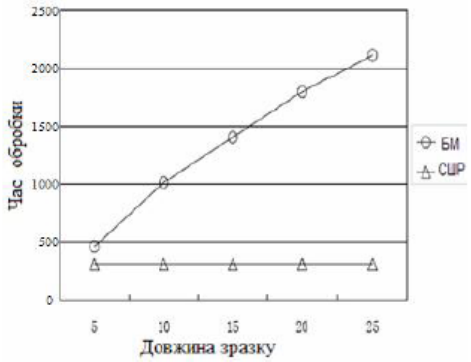

Рис. 3. Залежність часу обробки від довжини зразку при виконання алгоритму за умови, що *P=AA…A і T=AA…A*

З цієї ситуації можна побачити, що ефективність стійкого швидкого алгоритму відповідності рядків стабільна і набагато вища, ніж ефективність БМ. Оскільки запам'ятовується інформація символів, які вже є зіставленими, кількість символів, що порівнюються, різко скоротилося.

Дійсно, оскільки значення зсуву завжди дорівнює одиниці в рамках цієї ситуації, вдосконалена евристика хорошого суфіксу використовується в кожній контрольній точці. Таким чином, кількість порівнюваних символів, дорівнює довжині вхідного рядка і це відбувається незалежно від довжини ключового слова.

Експерименти з англійським текстом

Текст взятий із інструкції використання програмного забезпечення. Зразки випадковим чином вибираються з тексту.

Для кожної довжини зразка було відібрано 250 різних зразків, і 250 зразків порівнюються з текстом один до одного.

Результати досліджень показані на рис.4. Час обробки – це середній час пошуку 250 зразків.

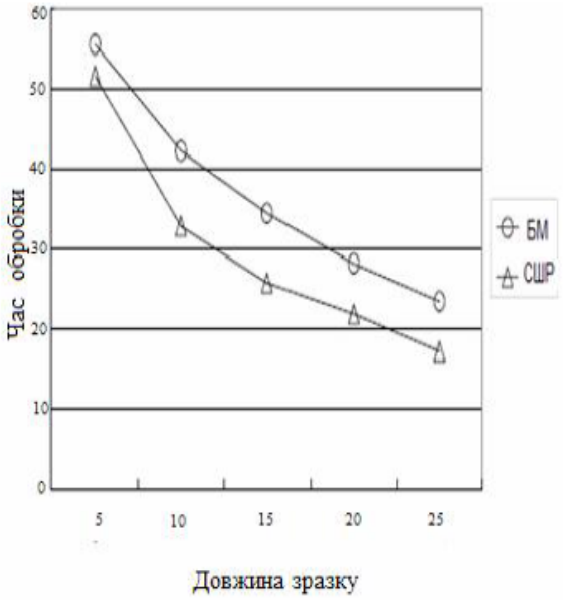

Рис.4. Залежність часу обробки від довжини зразку при пошуку англійського тексту

Можна побачити, що ефективність стійкого швидкого алгоритму відповідності рядків на 7.57~36.34% вища, ніж ефективність алгоритму Бойєра-Мура при пошуку англійського тексту.

Експерименти з рівномірно випадковим текстом

Оброблено текст, який складається з рівномірно випадкових символів 256-символьного алфавіту. Зразки випадковим чином вибираються з тексту. Для кожної довжини зразка відібрано 250 різних моделей, і 250 зразків порівнюються з текстом один до одного. Результати досліджень проілюстровані на рис.6. Час обробки – це середній час пошуку 250 зразків.

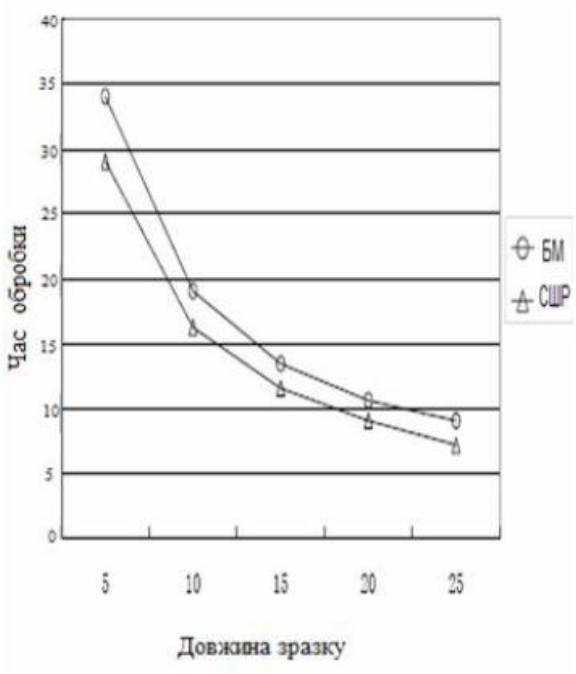

Рис.5. Залежність часу обробки від довжини зразку при пошуку рівномірно випадкового тексту

Результати досліджень свідчать про те, що ефективність СШР на 16.26~26.18% вища, ніж ефективність БМ при пошуку рівномірно випадкового тексту.

Експерименти з набором шаблонів Snort

Винайдено набір шаблонів реального світу за правилами *Snort.* Використовуються останні правила *Snort,* що були розроблені 15 червня 2006 року. Існує 2410 абсолютно різних моделей. Довжина зразків варіює від 1 до 122. Використаний текст – це рівномірно випадковий текст з 256-символьного алфавіту. 2410 зразків порівнюються з текстом один до одного.

Результати досліджень показані на рис.6. Час обробки – це середній час пошуку 2410 зразків.

Проілюстрований результат дослідження показує, що ефективність СШР на 9.77% вища, ніж ефективність БМ при пошуку набору зразків *Snort* в реальному світі.

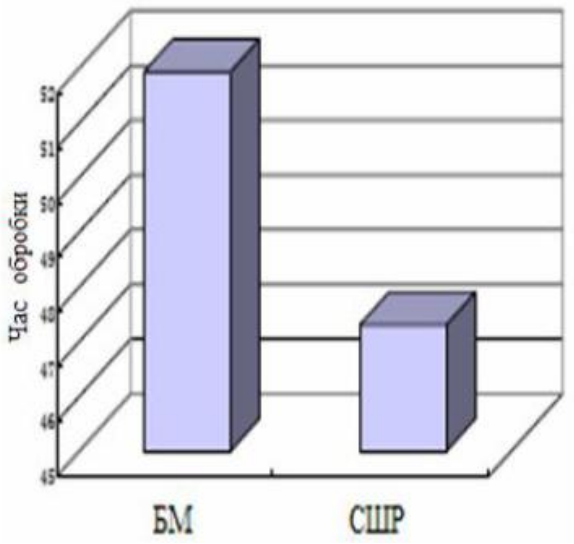

Рис.6. Ефективність при пошуку набору шаблонів *Snort*

Висновки

Розглянуто проблему відповідності рядка у сфері мережевої безпеки. Особливу увагу звернуто на проблему боротьби з алгоритмічними ефективними атаками, а також представлено розробку більш досконалого алгоритму Бойєра-Мура, стійкого швидкого алгоритму відповідності рядка. Алгоритм СШР використовує вдосконалену евристику поганого символу, щоб досягти більшого значення області зсуву, а також вдосконалену евристику хороших суфіксів, щоб за потребою запам'ятати інформацію символів, вже зіставлених. Пропонується нова визначальна умова для визначення того, чи потрібно звертатися до вдосконаленої евристики хорошого суфікса. Таким чином ефективність при найгіршому випадку виконання алгоритму СШР значно покращилася у порівнянні з алгоритмом БМ і в той самий час покращилися нормальні показники ефективності також.

Проведено оцінку і порівняно алгоритми СШР і БМ за допомогою використання різних текстів та зразків. Експериментальні результати свідчать про те, що алгоритм СШР є набагато ефективнішим, ніж алгоритм БМ у гіршому випадку виконання. Слід зазначити, що його ефективність покращується зі збільшенням довжини рядку і, відповідно, збільшенням довжини зразка. Алгоритм СШР є також більш ефективним, ніж алгоритм Бойєра-Мура, при нормальній ситуації. Ефективність СШР на 7,57 ~ 36,34% більша, ніж у БМ при пошуку англійського тексту, на 16,26 ~ 26,18% більша, ніж у БМ при пошуку рівномірно випадкового тексту, на 9,77% вища, ніж БМ при пошуку набору зразків *Snort*.

Список літератури

1. *M. Norton, and D. Roelker*, "The New Snort", Computer security journal, vol.19, no. 3, 2003. – P. 37–47.

2. *M. Roesch*, "Snort: lightweight intrusion detection for networks", Proc. 13th System Administration Conference and Exhibition (LISA'1999), 1999. – P. 229–238.

3. *R. Boyer, and J. Moore*, "A fast string searching algorithm", Communications of the ACM, vol. 20, no.10, 1977. – P. 762–772.

4. *E. P. Markatos, S. Antonatos, M. Polychronakis, and K. G. Anagnostakis,* "EXB: Exclusion-based signature matching for intrusion detection", Proc. The CCN'02, 2002.

5. *K. G. Anagnostakis, E. P. Markatos, S. Antonatos, and M. Polychronakis*, "E2XB: A domain-specific string matching algorithm for intrusion detection", Proc. 18th IFIP International Information Security Conference (SEC2003), 2003.

6. *M. Fisk, and G. Varghese*, "Fast content-based packet handling for intrusion detection", UCSD Technical Report CS2001-0670, May 2001.

7. *S. Antonatos, K. G. Anagnostakis, E. P. Markatos, and M. Polychronakis*, "Performance analysis of content matching intrusion detection system", Proc. 2004 International Symposium on Applications and the Internet (SAINT'04), 2004.

8. *S. Antonatos, K. G. Anagnostakis, and E. P. Markatos*, "Generating realistic workloads for network intrusion detection systems", Software engineering notes, vol.29, no. 1, 2004. – P. 207–215.

9. *R. N. Horspool*, "Practical fast searching in strings", Software practice and experience, vol. 10, no. 6, 1980. – P. 501– 506.

10. *R. M. Karp, and M. O. Rabin*, "Efficient randomized pattern-matching algorithms", IBM J. Res. Dev., vol. 31, no.2, 1987. – P. 249–260.

11. *D. Knuth, J. Morris, and V. Pratt*, "Fast pattern matching in strings", SIAM journal on computing, vol. 6, no. 2, 1977. – P. 323–350.

12. *M. Crochemore, M. C. Hancart*, Pattern Matching in Algorithms and Theory of Computation Handbook. CRC Press Inc., Boca aton, FL, 1999.

13. *C. Charras, and T. Lecroq*, "Exact string matching algorithms", http://wwwigm.univ-mlv.fr/~lecroq/string/, 1997.

14. *D. M. Sunday*, "A very fast substring search algorithm", Communications of the ACM, vol. 33, no. 8, 1990. – P. 132–142.

15. *Кормен, Т.* Алгоритмы: построение и анализ/ Т. Кормен, Ч. Лейзерсон, Р. Ривест – М.: МЦНМО, 2002.

16. *Ахо, Альфред* Структура данных и алгоритмы. – М.: Издательский дом «Вильямс», 2000. – 384 с.

17. *Матрос Д.* Элементы абстрактной и компьютерной алгебры: Учеб. пособие для студ. педвузов [Текст]. – М.:Издательский центр «Академия», 2004. –240 с.