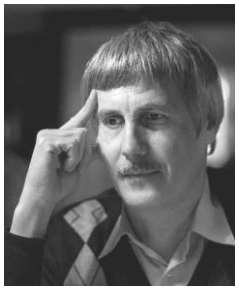
© Hilgurt S.Ya. Constructing Bloom filters by reconfigurable means for solving information security tasks // Ukrainian Scientific Journal of Information Security, 2019, vol. 25, issue 1, pp. 53-58.

DOI: 10.18372/2225-5036.25.13594

ПОБУДОВА ФІЛЬТРІВ БЛУМА РЕКОНФІГУРОВНИМИ ЗАСОБАМИ ДЛЯ ВИРІШЕННЯ ЗАДАЧ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Сергій Гільгурт

Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України



ГІЛЬГУРТ Сергій Якович, к.т.н., с.н.с.

Рік та місце народження: 1964 рік, Каракульський р-н Бухарської обл., Узбекистан.
Освіта: Київський інститут інженерів цивільної авіації (з 2000 року – Національний авіаційний університет), 1986 рік.
Посада: старший науковий співробітник відділу №2 Теорії моделювання з 2004 року.
Наукові інтереси: реконфігуровні обчислення, технічний захист інформації.
Публікації: більше 100 наукових публікацій, серед яких, наукові статті, матеріали та тези доповідей на конференціях, авторські свідоцтва на винаходи.
E-mail: hilgurt@nau.edu.ua.
Orcid ID: 0000-0003-1647-1790.

Анотація. *Системи виявлення вторгнень, антивіруси, додатки протидії мережевим хробакам, та інші засоби технічного захисту, робота яких заснована на використанні сигнатур, мають вирішувати в реальному часі обчислювально складну задачу множинного розпізнавання рядків. Програмні рішення вже не впорюються з цією проблемою через сталий зріст об'єму мережевого трафіку, кількості та складності атак. Тому все більшого поширення набувають апаратні рішення з використанням реконфігуровних пристроїв на базі ПЛІС типу FPGA, які поєднують в собі близьку до апаратної продуктивність із гнучкістю програмного забезпечення. На сьогодні найбільш поширеними є три основні підходи щодо побудови схем множинного розпізнавання на ПЛІС: асоціативна пам'ять, фільтр Блума та цифрові автомати. У даній статті досліджено властивості другого з перелічених підходів – фільтра Блума. Розглянуто базову схему та її модифікації. Проаналізовані особливості (переваги та недоліки) підходу в сенсі покращення вартісних, швидкісних та функціональних показників ефективності, а також показників масштабування. Досліджено специфіку реалізації фільтра Блума на ПЛІС.*

Ключові слова: *захист інформації, апаратне прискорення, ПЛІС, сигнатура, множинне розпізнавання рядків, фільтр Блума.*

Вступ

Протидія загрозам інформаційної безпеки потребує вживання заходів різних типів – організаційних, технічних, криптографічних тощо. Системи виявлення вторгнень [1] та протидії мережевим хробакам, антивіруси, та інші засоби технічного захисту, робота яких заснована на пошуку в інтенсивному потоці даних заздалегідь відомих ознак шкідливої активності (сигнатур), мають вирішувати в реальному часі обчислювально складну задачу множинного розпізнавання рядків [2, 3]. Зростання кількості та складності зовнішніх атак на комп'ютерні системи призводить до необхідності використання апаратного прискорення, тому що програмні рішення все менш відповідають сучасним вимогам щодо швидкодії. В якості платформи апаратного прискорення задач інформаційної безпеки найбільш придатним є такий клас пристроїв як реконфігуровні обчислювачі на базі ПЛІС типу FPGA [4].

Отже вдосконалення реконфігуровних засобів інформаційного захисту є актуальною науково-технічною проблемою. В рамках її вирішення у роботі [5] запропонований метод створення універсальної комбінованої структури модуля розпізнавання, суть якого полягає у поєднанні в одному пристрої переваг різних підходів до побудови схем розпізнавання на ПЛІС. Для його реалізації ключовим чинником є детальне розуміння технічних властивостей кожного з технічних рішень, що комбінуються. Тому актуальним завданням є також глибокий аналіз особливостей відомих підходів до побудови апаратних схем розпізнавання на базі ПЛІС.

На сьогодні найбільш поширеними є три підходи, які базуються на використанні: асоціативної пам'яті, хеш-функцій та цифрових автоматів. Дана робота присвячена всебічному аналізу другого з перелічених підходів, а саме використанню функцій хешування. Найчастіше дану технологію реалізують у вигляді так званого фільтра Блума (ФБ).

Аналіз існуючих досліджень

На відміну від інших підходів до побудови апаратних схем розпізнавання рядків, напрям, що вивчається в даній роботі, має відомого автора та чітку дату з'явлення. Бартон Говард Блум винайшов схему

обробки даних, що пізніше була названа його ім'ям, та оприлюднив її в 1970-му році в публікації [6].

Першою реалізацією фільтра Блума на ПЛІС вважається робота [7]. Після чого подібні дослідження почали проводитися доволі активно. Причому більшість з них була присвячена мережевим застосуванням [8]. Починаючі с 2004 року, фільтр Блума успішно використовується в галузі інформаційної безпеки, зокрема для побудови модуля розпізнавання (pattern matching) мережевих систем виявлення вторгнень (МСВВ), відповідний англомовний термін – Network Intrusion Detection System (NIDS). Відомі чисельні роботи щодо вирішення множинного розпізнавання рядків в системах МСВВ на базі ПЛІС, в тому числі – з використанням фільтра Блума [9-15]. Але найсуттєвіше на ефективність сигнатурних реконфігуровних засобів інформаційної безпеки впливають наступні чинники:

– особливості (переваги та недоліки) в сенсі забезпечення вартісних, швидкісних та функціональних показників ефективності, а також показників масштабування [16];

– специфіка реалізації на ПЛІС;

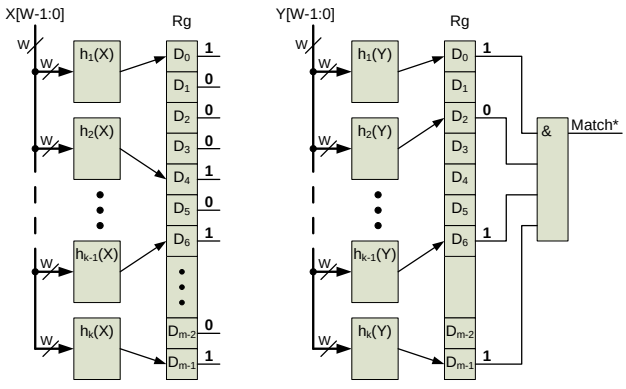
– складнощі та проблеми, що виникають при створенні реконфігуровних засобів та шляхи їх подолання.

Метою даної роботи є аналіз саме цих чинників стосовно підходу до побудови засобів множинного розпізнавання на базі фільтра Блума.

Основні принципи побудови та використання фільтра Блума

Фільтр Блума – це абстрактний пристрій, який дозволяє визначити чи належить певний фрагмент даних до заданого набору. Наприклад, чи співпадає задана послідовність символів с якимось з патернів, що входять до складу словнику сигнатур.

ФБ складається з двох ключових компонентів (рис. 1): комплекту з k хеш-функцій $h_1(x)$, $h_2(x)$, $h_3(x)$, …, $h_k(x)$ та регістру Rg (масиву бітів) довжиною в m розрядів, заповненому нулями в початковому стані.



а) програмування б) розпізнавання

Рис. 1. Принцип дії фільтра Блума

На етапі програмування на входи всіх хеш-функцій послідовно подається кожен з n елементів словнику сигнатур (довжиною W бітів). Для кожного елемента обчислюються всі m хеш-функцій, отримані значення яких інтерпретуються як номер (адреса) комірки у бітовому масиві. У відповідні комірки заносяться одиниці.

В процесі функціонування фільтра Блума на його вхід подається фрагмент вхідної послідовності символів (також довжиною W бітів). Для нього обчислюються значення всіх m хеш-функцій. За отриманими адресами здійснюється звернення до комірок бітового масиву. Якщо у всіх позиціях, на які вкажуть хеш-функції, містяться одиниці, вважається, що вхідна комбінація символів з певною вірогідністю співпадає з одним з патернів, що приймали участь у програмуванні ФБ. Але якщо хоча б одна хеш-функція вкаже на комірку з нульовим значенням, це гарантовано свідчить про відсутність відповідної послідовності символів серед множини патернів, що брали учать у програмуванні.

Отже, ФБ функціонує з деякою вірогідністю виникання помилки розпізнавання другого роду (false positive), проте без помилок першого роду (false negative). Тобто інколи він може сигналізувати про наявність збігу, якого насправді немає, але ніколи не пропустить справжній збіг. Значення вірогідності помилки другого роду для заданих значень кількості патернів n, розміру бітового масиву m, та кількості хеш-функцій k знаходиться згідно виразу [9]:

$$p_{пом.2} = \left(1 - \left[1 - \frac{1}{m}\right]^{kn}\right)^k \approx \left(1 - e^{-kn/m}\right)^k .$$

При цьому значення m покладається досить великим, що на практиці відповідає дійсності.

Як можна бачити, збільшення кількості патернів, що зберігаються в ФБ призводить до збільшення системної помилки, в той час як збільшення розміру бітового масиву – до зменшення помилки. Проте залежність від кількості хеш-функцій k не настільки очевидна. Можна довести, що для певних значень n та m вірогідність помилки мінімальна при

$$k = \frac{m}{n}\ln 2 \approx 0{,}6931\frac{m}{n}.$$

При цьому вірогідність помилки

$$p_{пом.2} = 2^{-k} \approx 0{,}6185^{m/n}.$$

З останнього виразу випливає, що для утримання помилки в певних межах потрібно забезпечувати пропорційність параметру m щодо n.

При використанні наведених співвідношень для вибору параметрів побудови ФБ слід мати на увазі той факт, що оскільки хеш-функції обчислюються цифровими схемами у двійковій системі числення, значення параметру розміру бітового масиву m чисельно буде дорівнювати ступеню двійки.

Аналізуючи наведені вище відомості щодо основ функціонування фільтра Блума можна зробити попередню оцінку його можливостей в якості підходу щодо створення реконфігуровних засобів розпізнавання на базі реконфігуровної логіки.

По-перше, ФБ дозволяє дуже економно використовувати ресурси пам'яті. Розмір словнику сигнатур не впливає напряму на розмір бітового масиву. Додання нових патернів до вже присутніх у цієї структурі даних призводить лише до несуттєвого підвищення вірогідності помилки розпізнавання, але не до збільшення об'єму запам'ятовуючого пристрою. Кількість та складність хеш-функцій, які потрібно обчислювати в

процесі розпізнавання, тобто, апаратні витрати (як наслідок – продуктивність) при даному підході також не залежать від об'єму словника сигнатур. Нарешті, довжина патернів також не впливає напряму ні на кількість ресурсів пам'яті, ані на продуктивність.

Отже можна зробити висновок, що ФБ добре масштабується принаймні за двома напрямками: об'єму словнику та довжині підрядків, що відшукуються. Особливо важливе значення має незалежність витрат фільтра Блума на пам'ять від довжини патерна. Навіть дуже довгі рядки після перетворення хеш-функціями потребують для зберігання ті ж самі k комірок бітового масиву.

Кластеризація

На жаль, фільтр Блума має дуже важливий недолік. Цьому підходу притаманна непомітна на перший погляд особливість усіх рішень на базі хешування: розмір вхідної послідовності символів, тобто довжина патерну має бути фіксованою для обраного набору хеш-функції. Інакше кажучи, кожен окремий ФБ здатен розпізнавати тільки множину патернів однакової довжини.

Єдиною можливістю подолання цього недоліку фільтра Блума – здатності розпізнавати тільки патерни однакової довжини – представляється побудова структури, яка містить кілька одночасно працюючих ФБ, кожен з яких налаштований на розпізнавання патернів однієї з можливих довжин.

На рис. 2 наведено таку схему з чотирьох фільтрів Блума BF_1, … BF_4, які розпізнають патерни довжиною від 2 до 5 символів відповідно [9]. Вхідна послідовність подається на конвеєр, складений з 8-розрядних регістрів RG_i. до кожного ФБ надходить відповідна кількість 8-розрядних шин.

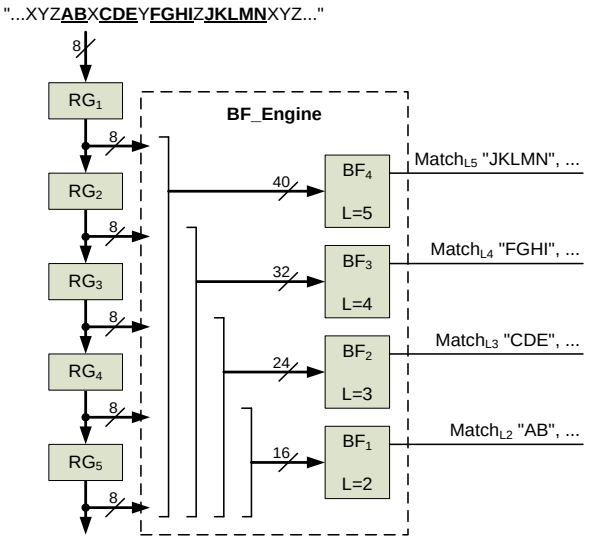


Рис. 2. Спільне підключення різних фільтрів Блума

Слід зауважити, що схема на рис. 2 має дещо ілюстративну форму для більш наочного представлення. На практиці на вхід кожного з ФБ дані надходять послідовно, байт за байтом. Це не призводить до додаткових затримок, оскільки на вхід розпізнавального модуля інформація подається також послідовно. Отже при синтезі ФБ на ПЛІС немає потреби у витратах ресурсів на створення широких багатодротових шин передачі даних.

Для реалізації розглянутої структури необхідно провести кластеризацію бази даних сигнатур за розміром патернів. У загальному випадку модуль розпізнавання, побудований на засадах фільтра Блума, має стільки окремих фільтрів, скільки різних довжин мають патерни, що входять до складу бази даних сигнатур. Це призводить до великих витрат ресурсів, що значно погіршує вартівні показники ефективності ФБ. Але це погіршення не стосується швидкісних показників. Про це свідчить той факт, що певна кількість успішних практичних розробок МСВВ на базі фільтра Блума фактично є системами попередження вторгнень, до яких висуваються значно жорсткіші вимоги щодо продуктивності [9,10, 13, 14].

Особливості реалізації фільтра Блума на ПЛІС

Розглянемо особливості реалізації фільтра Блума на ПЛІС стосовно генерації хеш-функцій та організації роботи з бітовим масивом.

Вибір хеш-функцій має важливе значення для ефективного створення фільтра Блума [8]. Ці функції мають, по-перше, однозначно відображати множину комбінацій вхідних символів на множину вихідних значень, тобто результати обчислень хеш-функції не повинні повторюватися для різних вхідних рядків. По-друге, цифрова схема реалізації функцій $h_1(x)$, $h_2(x)$, $h_3(x)$, …, $h_k(x)$ повинна мати якомога простішу апаратну структуру, щоб забезпечити високу швидкодію при невеликих ресурсних витратах.

Слід також зауважити, що діапазон вихідних значень хеш-функції повинен відповідати розміру бітового масиву m, оскільки має вказувати на довільну комірку регістру Rg. Але параметр m може приймати доволі різні значення з метою забезпечення оптимальності параметрів фільтра Блума. Отже, вихідний діапазон результатів обчислення хеш-функцій також має легко змінюватися в залежності від параметрів задачі.

Крім того, у випадку використання фільтра Блума в складі сигнатурної системи захисту інформації, множина хеш-функцій має легко реалізовуватися на реконфігуровній платформі.

Сформульованим вимогам задовольняють так звані хеш-функції класу H_3 [17, 18], які останніми роками успішно використовуються при створенні фільтрів Блума [9, 14]. Стосовно реалізації на ПЛІС такі хеш-функції можливо обчислювати рекурсивно. Ця властивість призводить до регулярності цифрової схеми та дозволяє знизити потрібні ресурси.

Особливістю базової схеми фільтра Блума (рис. 1) є необхідність одночасного доступу до масиву бітів Rg великої кількості хеш-функцій. У разі його реалізації на єдиному пристрої зберігання даних можуть виникнути колізії. Помірні вимоги до обсягу пристрою з боку схеми фільтра Блума дозволяють реалізовувати бітовий масив на базі внутрішнього ресурсу пам'яті ПЛІС – блоків вбудованої пам'яті BRAM. Бібліотечні компоненти більшості сучасних виробників НВІС реконфігуровної логіки надають можливість без надмірних зусиль синтезувати двопортові пристрої пам'яті. Такі пристрої дозволя-

ють одночасний доступ до комірок бітового масиву двох хеш-функцій. Але оскільки реальна кількість хеш-функцій в практичних додатках може сягати декількох десятків, колізії неминучі.

Рішення полягає у розбитті бітового масиву на декілька пристроїв залежно від кількості хеш-функції та здатності одночасного доступу схеми пам'яті. Для цього потрібно, по-перше, розподілити виходи блоків реалізації хеш-функцій між відповідними оперативним запам'ятовуючими пристроями (ОЗП), по-друге, на функціональність цих блоків накласти додаткові обмеження, щоб їх вихідний діапазон укладався у зменшений об'єм запам'ятовуючого пристрою. В роботі [9] показано, що такі обмеження не впливають суттєво на вірогідність помилок другого роду фільтра Блума.

Розпаралелювання

Підвищити продуктивність рішення з використанням підходу на основі фільтра Блума можливо за рахунок паралельного підключення із зсувом кількох однакових блоків, кожен з яких містить потрібний комплект фільтрів Блума. На рис. 2 такий блок BF_Engine обведений штриховою лінією. На рис. 3 наведено приклад паралельного використання трьох блоків.

Слід зауважити, що вхідна послідовність просувається швидше, аніж в базовій схемі (рис. 2), в даному прикладі – на три символи вперед на кожному циклі обробки інформації блоками BF_Engine_i.

Як можна бачити, трикратне прискорення призводить рівно до трикратного збільшення апаратних ресурсів. Тобто фільтр Блума також має добру масштабованість з швидкодії.

Можливість динамічної реконфігурації

Базова схема фільтра Блума не дозволяє в процесі функціонування вилучати патерни, додані в схему під час його програмування. Якщо ця властивість потрібна, застосовують такий різновид підходу, як фільтр Блума з лічильниками (Counting Bloom Filter). Вперше подібна модифікація була запропонована в роботі [19]. В схему додається *m* лічильників. В процесі програмування фільтра вихідні сигнали з хеш-функцій подаються на лічильні входи відповідних лічильників в режимі додавання. В разі потреби вилучити патерн з роботи активні значення виходів хеш-функцій керують лічильниками в режимі віднімання. Перехід лічильника зі стану "0" в стан "1" призводить до запису "1" у відповідну комірку бітового масиву фільтра Блума, перехід лічильника зі стану "1" в стан "0" – до запису "0". В іншому принцип функціонування схеми не змінюється.

Недоліком техніки фільтра Блума з лічильниками є додаткові витрати ресурсів ПЛІС. Причому точне значення розрядності лічильників складно визначити, тому що кількість "влучень" різних хеш-функцій в одну й ту саму комірку бітового масиву є випадковою величиною. Якщо використати лічильники недостатньої розрядності, їх переповнення призводитиме до виникнення помилок першого роду (false negative), тобто, схема буде розпізнавання не всі патерни, що були запрограмовані.

В роботах [8] та [19] показано, що 4-розрядних лічильників достатньо для більшості практичних застосувань, наведено оцінку вірогідності помилки першого роду.

Зважаючи на те, що потреба в вилучанні патернів зі словнику сигнатур виникає досить рідко, можливий варіант реалізації фільтра Блума з лічильниками програмними засобами [9]. Такий прийом не потребує додаткових апаратних витрат порівняно з базовою схемою, але в цьому випадку необхідно задіяти спеціальний режим конфігурації, при якому програмується тільки зміст вбудованої пам'яті, а інші частини ПЛІС залишаються незмінними.

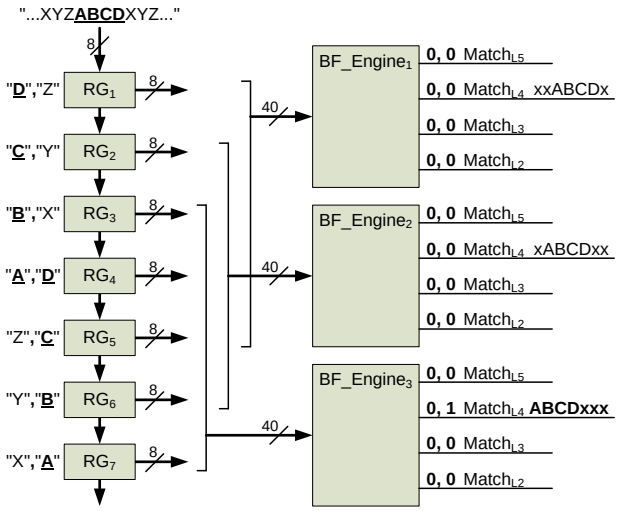


Рис. 3. Паралельне підключення кількох блоків розпізнавання на базі фільтрів Блума

Отже техніка використання лічильників дозволяє динамічно реконфігурувати фільтр Блума в реальному часі без перепрограмування ПЛІС та тимчасового припинення функціонування сигнатурної системи захисту інформації.

Уточнення результатів

Як було вказано вище, Фільтру Блума притаманні системні помилки розпізнавання другого роду. Тому результати розпізнавання потрібно уточнювати, тобто переконатися, чи насправді виявлена послідовність символів співпадає з патерном. Операція уточнення може бути виконана програмно на хост-комп'ютері, апаратними засобами із використанням частки ресурсів програмованої логіки [10] або за допомогою вбудованих в ПЛІС процесорних ядер універсальної архітектурі [14]. У двох останніх випадках цю операцію зазвичай виконують із застосуванням вторинної хеш-таблиці, яка зберігається в бортовому ОЗП реконфігуровного обчислювача, зовнішньому по відношенню до мікросхеми ПЛІС.

Слід зауважити, що операція уточнення виконується значно повільніше аніж процедура розпізнавання. У випадках, коли ознаки зловмисних дій в мереженому трафіку зустрічаються рідко, це не призводить до порушення коректності функціонування. Але якщо зловмисник здійснює атаку на систему виявлення вторгнень шляхом насичення мережевого трафіку пакетами, що імітують напад (тобто співпадають з патернами бази даних сигнатур МСВВ), це

може ускладнити роботу системи аж до втрати працездатності.

Отже необхідність уточнення результатів розпізнавання робить фільтр Блума вразливим до атак на МСВВ. Вирішення проблеми можливо через апаратний захист від переповнення черги запитів на уточнення. У цьому випадку в разі атаки схема уточнення не припиняє функціонування, проте подає на вихід частково неперевірені дані. Як наслідок система виявлення вторгнень в цілому починає робити помилки розпізнавання другого роду. Втім слід пам'ятати, що за рахунок належного вибору параметрів k та m вірогідність хибного спрацьовування фільтра Блума може бути знижена до несуттєвого значення.

Подальше підвищення швидкодії

Як згадувалося вище, фільтру Блума притаманна висока продуктивність. Багато дослідників намагалися підвищити швидкісні характеристики цього підходу. Але крім розпаралелювання інших дієвих технік знайдено не було.

Багато розробок присвячено скороченню надлишковості хеш-функцій та оптимізації часу їх обчислення [11, 12]. Більшість таких рішень зв'язана з каскадуванням. Суть техніки полягає у наступному. Оскільки в фільтрі Блума для отримання позитивного висновку про розпізнавання патерна з бази сигнатур необхідна наявність одиниць в усіх відповідних комірках бітового масиву, достатньо одного нуля, щоб зробити негативний висновок. Тому, якщо обчислювати хеш-функції послідовно, можна позбавитися відпрацьовування більшості з них. Але організувати обчислювальний процес таким чином, щоб послідовна обробка випереджала паралельну, складно. В результаті більшість схем каскадування призводять до суттєвого енергозбереження, але не прискорення.

В роботі [12] вдалося досягнути загального підвищення продуктивності системи МСВВ у цілому, але за рахунок маніпулювань із запитами на обробку вхідних даних, причому на рівні пакетної обробки інформації, не змінюючи власне структуру фільтра Блума.

В роботі [13] запропонована техніка, застосування якої зменшує швидкодію за рахунок одночасного доступу до двох бітів при зверненні до ОЗП. Таку можливість надає особливий режим роботи деяких запам'ятовуючих пристроїв, зокрема механізм burst-type I/O синхронної пам'яті типу SDRAM. За рахунок зменшення кількості звернень до бітового масиву вдалося досягти певного прискорення. Але це призвело до підвищення вірогідності помилок другого типу, хоча й незначного. На жаль дана техніка не може бути використана при реалізації фільтра Блума на ПЛІС.

Висновки

Підсумуємо отримані результати щодо особливостей та показників ефективності розглянуто підходу.

Головні переваги фільтра Блума при побудові реконфігуровних засобів інформаційної безпеки – це *економне споживання ресурсів* та *висока пропускна здатність*.

ФБ *добре масштабується* стосовно *швидкодії*, *об'єму словнику* та *довжині патернів*.

Його регулярна та логічна структура обумовлює *добру здатність до реалізації на ПЛІС*. Єдина складність полягає в одночасному зверненні множини хеш-функцій до багаторозрядного масиву бітів. Але ця проблема вирішується шляхом розбиття запам'ятовуючого пристрою на декілька окремих модулів та незначної корекції обчислювачів хеш-функцій.

Також позитивною рисою підходу є можливість реалізації режиму *динамічної реконфігурації* без припинення функціонування системи розпізнавання. Схема ФБ з лічильниками за рахунок невеликих додаткових витрат дозволяє здійснити таку функцію.

Головний недолік підходу на базі фільтра Блума полягає у *низької гнучкості*, тому що він здатен розпізнавати тільки патерни однакової довжини. Тобто в загальному випадку для розпізнавання множини патернів потрібно стільки окремих ФБ, скільки різних довжин ці патерни мають, що призводить до суттєвих ресурсних витрат, тобто зводить нанівець одну з головних переваг підходу.

Більш відомим, але менш важким за наслідками недоліком ФБ є *наявність системних помилок* розпізнавання другого роду, що призводить до необхідності виконання додаткового обчислювального етапу уточнення результатів роботі схеми. Сама по собі ця властивість не є критичною. Але, оскільки додаткова операція виконується значно повільніше аніж процедура розпізнавання, фільтр Блума стає вразливим до специфічних атак на МСВВ, якщо не вжити запобіжних заходів.

Здобуті в даному дослідженні відомості дозволять розробникам створювати більш ефективні реконфігуровні засоби інформаційної безпеки.

Література

[1]. С. Казмірчук, А. Корченко, Т. Паращук, "Аналіз систем виявлення вторгнень", *Захист інформації*, Т. 20, № 4, С. 259-276, 2018.

[2]. Б. Смит, *Методы и алгоритмы вычислений на строках. Теоретические основы регулярных вычислений. Пер. с англ.* М.: Вильямс, 2006, 496 с.

[3]. С. Гільгурт, "Множинне розпізнавання рядків у системах виявлення вторгнення на базі реконфігуровних обчислювачів", *Сучасні комп'ютерні системи та мережі: розробка та використання: матеріали 5-ої Міжнар. наук.-техн. конф. ACSN-2011, 29 вересня – 01 жовтня 2011, Львів, Україна*. Л.: Вид-во Нац. ун-ту «Львів. політехніка», С. 54–56, 2011.

[4]. С. Гильгурт, "Реконфигурируемые вычислители. Аналитический обзор", *Электронное моделирование*, Т. 35, № 4, С.49-72, 2013.

[5]. С. Гильгурт, "Применение реконфигурируемых вычислителей для аппаратного ускорения сигнатурных систем защиты информации" *// Тез. доп. Міжнар. наук.-техн. конф. «Моделювання-2018». – Київ: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України*, С. 107-110, 2018.

[6]. B. Bloom, "Space/time trade-offs in hash coding with allowable errors", *Communications of the ACM,* Vol. 13, No. 7, pp. 422-426, 1970.

[7]. D. Pryor, M. Thistle, N. Shirazi, "Text searching on Splash 2", *IEEE Workshop on FPGAs for Custom Computing Machines*, pp. 172-177, 1993.

[8]. A. Broder, M. Mitzenmacher, "Network applications of bloom filters: A survey", *Internet Mathematics*. Vol. 1, No. 4, pp. 485-509, 2004.

[9]. S. Dharmapurikar, P. Krishnamurthy, T. Sproull, J. Lockwood, "Deep packet inspection using parallel bloom filters", *IEEE Micro*, Vol. 24, No. 1, pp. 52-61, 2004.

[10]. S. Dharmapurikar, M. Attig, J. Lockwood, "Design and Implementation of a String Matching System for Network Intrusion Detection using FPGA-based Bloom Filters", *Washington University in St. Louis Engineering D. o. C. S.*: Scholarship W. U. O., 2004.

[11]. T. Kocak, I. Kaya, "Low-power bloom filter architecture for deep packet inspection", *IEEE Communications Letters*, Vol. 10, No. 3, pp. 210-212, 2006.

[12]. N. Artan, K. Sinkar, J. Patel, H. Chao, "Aggregated Bloom Filters For Intrusion Detection and Prevention Hardware", *IEEE Global Telecommunications Conference (GlobeCOM),* pp. 349–354, 2007.

[13]. Y. Chen, A. Kumar, J. Xu, "A New Design of Bloom Filter for Packet Inspection Speedup", *IEEE Global Telecommunications Conference (GlobeCOM),* pp. 1-5, 2007.

[14]. J. Harwayne-Gidansky, D. Stefan, I. Dalal, "FPGA-based SoC for real-time network intrusion detection using counting bloom filters", *Conference Proceedings - IEEE Southeastcon-2009, Article number 5174096,* pp. 452-458, 2009.

[15]. S. Geravand, M. Ahmadi, "Bloom filter applications in network security: A state-of-the-art survey", *Computer Networks,* Vol. 57, No. 18, pp. 4047-4064, 2013.

[16]. В. Евдокимов, А. Давиденко, С. Гильгурт, "Централизованный синтез реконфигурируемых аппаратных средств информационной безопасности на высокопроизводительных платформах", *Захист інформації*, Т. 20, № 4, С. 247-258, 2018.

[17]. L. Carter, M. Wegman, "Universal classes of hashing functions Bloom filter applications in network security: A state-of-the-art survey", *Computer Networks*, Vol. 57, No. 18, pp. 4047-4064, 2013.

[18]. M. Ramakrishna, E. Fu, E. Bahcekapili, "Efficient hardware hashing functions for high performance computers", *IEEE Transactions on Computers,* Vol. 46, No. 12, pp. 1378-1381, 1997.

[19]. L. Fan, P. Cao, J. Almeida, A. Broder, "Summary cache: A scalable wide-area Web cache sharing protocol", *IEEE/ACM Transactions on Networking*, Vol. 8, No. 3, pp. 281-293, 2000.

УДК 004.274:004.056

Гильгурт С. Построение фильтров Блума реконфигурируемыми средствами для решения задач информационной безопасности

Аннотация. *Системы обнаружения вторжений, антивирусы, приложения противодействия сетевым червям и другие средства технической защиты, работа которых основана на использовании сигнатур, должны решать в реальном времени вычислительно сложную задачу множественного распознавания. Программные решения уже не справляются с этой проблемой из-за устойчивого роста объема сетевого трафика, количества и сложности атак. Поэтому все большее распространение получают аппаратные решения с использованием реконфигурируемых устройств на базе ПЛИС типа FPGA, которые сочетают в себе близкую к аппаратной производительность с гибкостью программного обеспечения. На сегодня наиболее распространены три основных подхода к построению схем множественного распознавания на ПЛИС: ассоциативная память, фильтр Блума и цифровые автоматы. В данной статье исследованы свойства второго из перечисленных подходов – фильтра Блума. Рассмотрены базовая схема и ее модификации. Проанализированы особенности (преимущества и недостатки) подхода в смысле улучшения стоимостных, скоростных и функциональных показателей эффективности, а также показателей масштабирования. Исследована специфика реализации фильтра Блума на ПЛИС.*

Ключевые слова: *защита информации, аппаратное ускорение, ПЛИС, сигнатура, множественное распознавание строк, фильтр Блума.*

Hilgurt S. Constructing Bloom filters by reconfigurable means for solving information security tasks

Abstract. *Recently, the string matching task became a performance bottleneck in network intrusion detection, antivirus, anti-worms and other signature-based information security systems. Unlike the firewall, NIDS examines not only packet headers, but also the packet bodies, that is, performs the deep packet inspection (DPI). The multi-pattern string matching task is a specific type of string matching functionality performed in DPI systems to search an input stream for a set of patterns rather than a single pattern. Due to rising traffic rates, increasing number and sophistication of attacks and the collapse of Moore's law for sequential processing, traditional software solutions can no longer meet the high requirements of today's security challenges. Therefore, hardware approaches are proposed to accelerate pattern matching. Combining the flexibility of software and the near-ASIC performance, reconfigurable hardware devices based on Field Programmable Gate Arrays (FPGA) have become increasingly popular for this purpose. There are three main approaches to fulfill the computation-intensive multi-pattern string matching task using FPGA. The techniques (and underlying technologies) of these approaches are: content addressable memory (based on digital comparators), Bloom filter (based on hash-functions) and Aho-Corasick Algorithm (based on finite automata). This article is devoted to the investigation of the second approach - Bloom filter. The features (advantages and disadvantages) of this approach in terms of resource costs, speed/throughput parameters, as well as scaling parameters are explored. The basic scheme and its modifications are considered. The performance characteristics, problems and challenges of implementing this approach on reconfigurable accelerators as well as ways to overcome them are analyzed. The obtained results allow improving the technical parameters when designing pattern matching modules, which are speed-critical components of the signature-based information protection systems hardware. The knowledge obtained in this study allows developers to create more effective reconfigurable means for information security.*

Keywords: *information security, reconfigurable accelerator, FPGA, Bloom filter, hash-function, signature, multi-pattern string matching, DPI.*

Отримано 20 *березня 2019 року*, затверджено редколегією 15 *квітня 2019 року*