2. *Gonzalez R.C.* Digital Image Processing 3rd edition / R.C. Gonzalez, R.E. Woods.– Pearson Prentice Hall, 2008.– 954 p.
3. *Евдокимов В.Ф.* О построении системы ультразвукового контроля конструкционных материалов объектов энергетики и машиностроения / В.Ф. Евдокимов, А.С. Огир // Электронное моделирование. 2001. – Т.23, №5.– С.85-90.
4. *Merigot A.* Parallel processing for image and video processing: Issues and challenges / A. Merigot and A. Petrosino // Parallel Computing. – 2008. – Vol. 34. – P.694-699.
5. *Micikevicius P.* GPU Performance Analysis and Optimization // GPU Technology Conference 2012, NVIDIA.
6. *M. Rofouei, T. Stathopoulos, S. Ryffel, W. Kaiser, and M. Sarrafzadeh.* Energy-Aware High Performance Computing with Graphic Processing Units. // In Proceedings of the 2008 con-ference on Power aware computing and systems, p. 11-11. USENIX Association, 2008.
7. *Wolfgang E.* ShaderX3: Advanced Rendering with DirectX and OpenGL / Engel Wolfgang.– Charles River Media, 2004.– 654 c.
8. *Горнаков С. Г.* Инструментальные средства программирования и отладки шейдеров в DirectX и OpenGL / С.Г. Горнаков.– С-П.: БХВ-Петербург, 2005.– 256 с.
9. *Боресков А. В.* Разработка и отладка шейдеров / А.В. Боресков.– С-П.: БХВ-Петербург, 2006.– 488 с.
10. *Калашник Д.А.* Медицинские приборы. Разработка и применение / Д.А. Калашник и др. – М.: Стормовъ-Медицина, 2004. – 241 с.

Поступила 12.10.2017р.

УДК 004.056:004.75

М.Р. Шабан, Київ

ВИКОРИСТАННЯ АПАРАТУ РЕГУЛЯРНИХ ВИРАЗІВ ДЛЯ АНАЛІЗУ ФУНКЦІОНАЛЬНОГО ПРОФІЛЮ ЗАХИСТУ

Abstract. A regular expression (sometimes called a rational expression) is, in theoretical computer science and formal language theory, a sequence of characters that define a search pattern. Usually this pattern is then used by string searching algorithms for "find" or "find and replace" operations on strings.

Вступ

Регулярні вирази – засіб пошуку по тексту на основі шаблонів. Шаблон описує закономірність, який повинен підкорятися шуканій послідовності символів в тексті.

Для визначення доцільності застосування функціональних послуг безпеки (ФПБ) [1] в інформаційних системах здійснюється експертиза системи захисту інформації на об'єкті. В процесі експертизи оцінюється рівні інформації, що оброблюються в системі і оцінюються ризики її втрати,

модифікації або розголошення. Для цього будується функціональний профіль захисту (ФПЗ), який включає в себе переліки та рівні ФПБ, які необхідні для забезпечення прийнятного рівня безпеки інформації.

Сучасні методи та засоби аналізу ФПЗ мають деякі проблеми в питаннях оцінки захищеності автоматизованих систем [2], де розглянуті питання проблеми декомпозиції оцінюваємої системи з метою виділення, класифікації і опису ресурсів, які потребують захисту.

Також невирішеними є питання вибору стратегії оцінки безпеки інформації. Іншими словами – як і з якою періодичністю робити оцінку (вибірково, планово, контрольно-оглядово тощо), за якими критеріями, а саме – вітчизняні критерії (більш 110 часткових показників згідно з НД ТЗІ 2.5-004-99), або досить проста універсальна 6-10 бальна шкала безпеки критеріїв TCSEC, ITSEC, чи шкала більш 280 часткових показників найкращих міжнародних критеріїв CCITSE (стандарт ISO/IEC 15408) [5].

Актуальним завданням є наукове забезпечення підтримки прийняття рішень при проведенні державної експертизи, а саме постановка та вирішення завдань, які виникають при розробці та аналізу ФПЗ. Це – визначення ФПЗ, ранжування ФПБ, аналіз їх взаємозв'язку, аналіз повноти та непротиріччя, які вже були розглянуті мною раніше [3].

А зараз розглянемо ФПЗ з точки зору використання апарату регулярних виразів.

ФПЗ як строкова послідовність

Функціональний профіль захисту представляє собою строкову послідовність. Кожний символ строкової послідовності є елементом цієї послідовності.

Строкова послідовність з кінцевим числом елементів, яка містить як найлівіші, так і найправіші елементи, назвемо лінійною строковою послідовністю або лінійною строкою. Строкову послідовність з кінцевим (ненульовим) числом елементів, яка не має ні найлівіших, ні найправіших елементів, назвемо строковою петлею або циклічною строковою послідовністю. Строкова послідовність з безкінечною кількістю елементів, яка має найлівіший елемент, назвемо нескінченною строковою послідовністю; нескінченну строкову послідовність, яка не має ні найлівішого, ні найправішого елемента, назвемо нескінченною петлею.

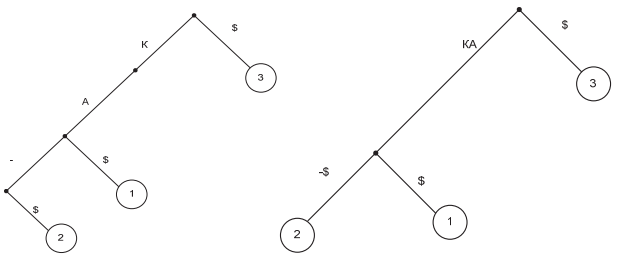
Внутрішні патерни

Прийшов час розглянути патерни. Внутрішні патерни (intrinsic patterns) в певному розумінні є «власними» або «природними» властивостями строкових послідовностей – не кожна строка містить частковий патерн (specific pattern), наприклад «*бб*» або характеристичний патерн (generic pattern), такий як кратні строки. З іншого боку, внутрішні патерни можна знайти у всіх строках, і вони також в певному сенсі характеризують строкові послідовності. Введемо поняття «синтаксичне дерево».

Нехай $X = \{x_1, x_2, \ldots, x_m\}$ – множина попарно різних строк. Тоді синтаксичне дерево на множині X – це дерево пошуку, що має у точності $m+1$ кінцевих вузлів: по одному для кожної строки $x_i (i = 1, 2, \ldots, m)$ плюс один для порожньої строки ε. Ребра синтаксичного дерева помічені буквами, які містяться у строках з множини X, і спеціальним «сигнальним» символом $, який означає кінець строки. Строка x_i розкладається на окремі літери – від кореня дерева до кінцевого вузла, яким закінчується ребро помічене символом $. У загальному випадку з кожним низхідним шляхом від вузла N_1 до вузла N_2, наприклад, асоціюється деяка строка u. Щоб уникнути зайвих складнощів виразів і в той же час можливих термінологічних ускладнень, будемо говорити, що вузол N_2 – це строка u, і якщо N_1 є коренем дерева, тоді строка u складається з літер, якими помічений шлях від вузла N_1 до вузла N_2. Відмітимо, що в цьому випадку строка u є префіксом хоч би одній який-небудь строки $x_i$$. Якщо ж $N_1 = N_2$, тоді $u = \varepsilon$. Зокрема, корінь дерева є порожньою строкою.

Використання символу $ обумовлено тим, що $m+1$ кінцевих вузлів є листям дерева *i*, як показано на рис.1, а для множини $X = \{КА, КА\ \}$, без символа $ неможливо було б виділити вузол 1, відображаючий той факт, що строка «КА» є префіксом строки «КА-». На цьому рисунку можна помітити ще одне характерне явище такої структури даних – загальні префікси (такі, як КА або ε) на синтаксичному дереві відображаються тільки один раз.

На рис.1, б показано компактне синтаксичне дерево, яке отримано з синтаксичного дерева шляхом виключення вузлів 2, які мають батька і тільки одного сина, тобто корінь дерева до числа вузлів, що виключаються, не



а) б)

Рис.1. Синтаксичне дерево для множини $X = \{КА, КА-\}$

входить в компактному синтаксичному дереві буде мати ребра, помічені не окремими буквами, а підстроками, як показано на рис.1, б. Синтаксичне дерево є дуже важливим внутрішнім патерном, який має усі ознаки декомпозиційного підходу [6].

Часткові патерни

Патерни, які будуть обговорювані зараз, відносяться до того типу, які обчислюються «класичними» алгоритмами обробки патернів, тобто часткові патерни: маємо строку (іноді її називають текстом) і потрібно знайти одне або усі входження в неї іншої заданої строки (яка має назву патерн). Запишемо алгоритм обробки патернів.

Пошук усіх входжень строки p у строку x

for $i = 1$ *to* $n - m + 1$ *do*
$j =$ *порівняти*(i, m)
if $j = m + 1$ *then*
output i

Цей алгоритм достатньо простий. Але з іншого боку, ні для кого не складе труднощів написати подібний алгоритм, оскільки він очевидний: для кожної позиції i в строці *x* виконується тривіальна процедура *порівняти (i,m)*, яка порівнює по одній літері зліва направо строку $p\,[1..m]$ з підстрокою $x[i..i+m-1]$. Ця процедура повертає значення $m+1$, якщо $p = x[i..i+m-1]$, або найменше ціле $j \in 1..m$, таке, що $p\,[1..j] \neq x[i..i+j-1]$. Таким чином, якщо $j = m+1$, то це означає, що строка p входить в строку x починаючи з позиції i.

Апроксимуючі патерни

Основна ідея у використанні апроксимуючих (наближених) патернів полягає у визначенні «відстані» між двома строками. У загальному випадку ***відстань*** між строками p_1 і p_2 розраховується як зважену кількість стандартних операцій редагування, необхідних для виконання перетворення p1 → p2. Розглядають наступні операції редагування.

- **Вставка** – наприклад, вставка символу *«,»*, в строку«*КА-2*» сформує строку «*КА-2,»*;
- **Видалення** – наприклад, видалення символу *«,»*, з строки«*КА-2,»*, сформує строку «*КА-2*»;
- **Підстановка** – наприклад, підстановка символу «Д» замість символу «А» з строки «КА-2» сформує строку«*КД-2*» .

Приклад

На основі отриманих знань, взявши за основу POSIX [4] стандарт на мові програмування Python 3.1 [8], побудуємо регулярний вираз на перевірку відповідності заданого функціонального профілю захисту еталону:

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
# імпортуємо модуль «re», який дозволяє опрацьовувати регулярні вирази
import re
#створимо функцію «match» з двома аргументами regexp(регулярний вираз),
#str (ФПЗ) з метою визначення коректності опрацювання шаблону заданим
#ФПЗ.
def  match(regexp,str):
     match = re.search(regexp,str)
     if match
          return match.group()
     return u"Не відповідає шаблону"
profil = u'{ КА-2, КД-2, ЦА-1 , ЦД-1, ДС-1, ДЗ-1 ДВ-1, НР-2, НИ-2, НК-1, НО-
1, НЦ-2, НТ-2}'
shablon=ur'^\{((КД-[1-4],)?(КА-[1-4],)?(КО-1,)?(КК-[1-3],)?(КВ-[1-4],)?(ЦД-[1-
4],)?(ЦА-[1-4],)?(ЦО-[1-2],)?(ЦВ-[1-3],)?(ДР[1-3],)?(ДС-[1-3],)?(ДЗ-[1-3],)
?(ДВ-[1-3],)?(НР-[1-5],)?(НИ-[1-3],)?(НК-[1-2],)?(НО-[1-3],)?(НЦ-[1-
3][,|\}])(НТ-[1-3][,|\}])?(НВ-[1-3][,|\}])?(НА-[1-3][,|\}])? (НП-[1-3] [,|\}])?)$'
print match(shablon, profil),
```

де () – групування; позиція всередині рядка: ^ – початок рядка; $ – кінець рядка; ? – квантифікація від 0 до 1; | – буліве «або».

Мова програмування Python була обрана з метою наочності та зрозумілості синтаксису навіть для початківців [7]. Унікальність кожної ФПБ досягається за рахунок використання кватнифікатора «?».

Правила оформлення ФПЗ

Правильним вважається наступний формат профілю захисту. Опис профілю складається з трьох частин: літеро-числового ідентифікатора, знака рівності і переліку рівнів послуг взятого в фігурні дужки. Ідентифікатор у свою чергу включає: позначення класу АС (1, 2 або 3), літерну частину, що характеризує види загроз, від яких забезпечується захист (К, і/або Ц, і/або Д), номер профілю і необов'язкове літерне позначення версії. Всі частини ідентифікатора відділяються один від одного крапкою. Ранжування ФПБ повинно відбуватись згідно нормативного документу НД ТЗІ 2.5-004-99 «Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу».

Висновок

У статті було проведено аналіз можливостей використання апарату регулярних виразів для ФПЗ з метою побудови на їх основі системи «Програма підтримки прийняття рішень при проведенні державної експертизи». Були розглянуті методи регулярних виразів: порівняння строк; дерево суфіксів; апроксимуючі патерни; часткові патерни. Показано, що технологія апроксимуючих патернів вирішують поставлені завдання аналізу ФПЗ і можуть бути використані для побудови системи. Дані результати підтверджені експериментальним шляхом, за рахунок успішної реалізації модуля " Визначення профілю захисту" в середовищі VisualStudio 2013 на мові програмування C#.

1. *А.Н. Давиденко, М.Р. Шабан* Разработка методики проведения экспертиз комплексных систем защиты информации // Зб. наук. праць ІПМЕ НАН України. – Київ, 2014 – Вип. 73. – С.114-121.

2. *В. Щербина, А.А. Скопа* Проблемы оценки защищенности автоматизированных систем // Науково-технічний журнал «Захист інформації». – Київ, 2008 – № 4. – С.23-28.

3. *М.Р. Шабан* Формалізація правил перевірки повноти та несуперечності функціонального профілю захисту // Зб. наук. праць ІПМЕ НАН України. – Київ, 2016 – Вип. 76. – С.89-94.

4. *Фридл Дж.* Регулярные выражения, 3-е издание // Фридл Дж – СПб.: Символ_Плюс, 2008. – С.188-190.

5. *А. Ільніцький* Безпека інформації в комп'ютерних системах та деякі підходи до її експертної оцінки // Науково-технічний збірник «Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні». – Київ, 2001. – Вип. 3. – С.86-90.

6. *А.Л. Яловец* Представление и обработка знаний с точки зрения математического моделирования. Проблемы и решения // А.Л. Яловец. – К.: Наукова думка, 2011. – С.64-75.

7. *Лутц М.* Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – С.43-44.

8. *Саммерфилд М.* Программирование на Python 3. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – С.607.

Поступила 25.10.2017р.