УДК 004.056:004.4

СИГНАТУРНИЙ АНАЛІЗ НА ОСНОВІ РЕГУЛЯРНИХ ВИРАЗІВ ПРИ ПОТОКОВІЙ ОБРОБЦІ ФАЙЛІВ

Пилипенко В.І., старший викладач
Київський національний університет технологій та дизайну
Гордієнко В.В., студент
Київський національний університет технологій та дизайну
Олійник І.Л., студент
Київський національний університет технологій та дизайну

Ключові слова: сигнатурний аналіз, регулярні вирази, потокова обробка даних, кібербезпека, аналіз мережевого трафіку, тестування, регулярні вирази, програмні решення, Python.

Сигнатурний аналіз на основі регулярних виразів у поєднанні з потоковою обробкою даних є ефективним підходом до виявлення шаблонів (сигнатур) у великих обсягах інформації без необхідності повного завантаження файлів у оперативну пам'ять [1]. У межах цього підходу обробка здійснюється інкрементально: вхідні дані аналізуються частинами, а на кожному кроці перевіряється їх відповідність заздалегідь визначеним регулярним виразам. Така модель обчислень забезпечує можливість роботи в умовах обмежених ресурсів і є придатною для аналізу потокових або високошвидкісних даних[2].

Зазначений підхід широко застосовується в задачах кібербезпеки, зокрема у міжмережевих екранах, системах виявлення та запобігання вторгненням (IDS/IPS), аналізі мережевого трафіку, обробці журналів подій та системах контент-фільтрації. Його ефективність обумовлена поєднанням виразної потужності регулярних виразів, які дозволяють формалізувати складні шаблони, та потокової обробки, що мінімізує витрати пам'яті й забезпечує виконання аналізу в реальному часі. Це є критично важливим для систем моніторингу та реагування на загрози. Водночас ключовим обмеженням сигнатурного підходу є його залежність від якості та повноти бази сигнатур.

Дослідження показують, що проблема якості регулярних виразів є системною. Зокрема, у роботі [3] автори виявили, що 83,2% місць у коді, де використовуються регулярні вирази, взагалі ніяк не перевіряються тестами. Це означає, що розробники масово впроваджують складні шаблони пошуку, сподіваючись на їх безпомилковість, що є вкрай ризикованим для стабільності системи.Навіть якщо розробник пише тести, вони зазвичай однобокі. Майже 73% виразів перевіряються або тільки на правильних даних, або тільки на неправильних. Це залишає "сліпі зони", де можуть залишатися помилки, які проявляться лише в специфічних умовах.

В таблиці 1 показано детальні результати тестування в програмному коді.

Таблиця 1

Результати тестування в програмному коді

Показник тестування	Значення	Опис проблеми
Відсутність тестів	83,2%	Місця в коді, де регулярні вирази взагалі не охоплені тестуванням
Однобоке тестування	73,0%	Вирази, що перевіряються лише на позитивних (accepted) або лише на негативних (rejected) даних
Мінімальне покриття	42,0%	Для перевірки складного шаблону використовується лише один рядок даних
Логічне покриття	Низьке	Більшість тестів проходять лише найпростішим маршрутом, залишаючи «сліпі зони»

У роботіЕріка Ларсона [4] дослідження інструменту ACRE продемонструвало, що використання 11 спеціалізованих перевіряючих модулів дозволяє автоматично виявити 283 дефекти у базі з 826 регулярних виразів. Це становить 75% від загальної кількості помилок, ідентифікованих у синергії з методом тест-генерації EGRET.

В таблиці 2 порівнюється ефективність інструментів валідації ACRE та EGRET. Незважаючи на те, що більшість регулярних виразів у реальних проєктах не тестуються (83,2%), вони містять значну кількість дефектів. Хоча інструмент ACRE довів, що 3/4 усіх помилок (75%) можна виявити автоматично без складного ручного тестування.

На відміну від класичного сигнатурного аналізу, який обмежується перевіркою відповідності регулярному виразу, пропонується розширений підхід, що поєднує потокову перевірку з оцінюванням покриття шаблону. Ідея полягає у відстеженні активованих фрагментів регулярного виразу під час обробки потоку даних, що дозволяє виявляти неактивні або недостатньо протестовані частини сигнатури.

Таблиця 2

Порівняльна ефективність інструментів

Параметр	Інструмент ACRE	Інструмент EGRET
Кількість помилок	283	94 (додатково до ACRE)
Частка від загальної кількості	75%	25% (специфічні помилки)
Метод перевірки	11 автоматизованих модулів	Генерація тестових рядків
Переваги для розробника	Автоматична діагностика та поради щодо виправлення	Виявлення помилок, що потребують контексту
Трудомісткість	Низька (автоматизовано)	Висока (потребує ручного аналізу рядків)

Практична реалізація запропонованого підходу може бути представлена у вигляді потокового аналізатора, який обробляє дані блоками фіксованого розміру та застосовує регулярні вирази до кожного блоку з урахуванням перекриття. Приклад спрощеної реалізації наведено в програмному лістингу 1.

*Програмний лістинг 1:*Потоковий аналізатор обробки даних блоками фіксованого розміру.

```
import re

def stream_regex_analysis(file_path, pattern, chunk_size=1024):
regex = re.compile(pattern)
matches = 0
processed = 0


with open(file_path, 'r', encoding='utf-8', errors='ignore') as f:
buffer = ''

while True:
chunk = f.read(chunk_size)
```

```
if not chunk:
break

data = buffer + chunk
found = regex.findall(data)

matches += len(found)
processed += len(data)

# збереження перекриття для коректної обробки шаблонів
buffer = data[-100:]

return matches, processed
```

Запропонований алгоритм забезпечує потокову обробку даних із мінімальним використанням пам'яті та дозволяє застосовувати регулярні вирази до довільно великих файлів. Ключовою особливістю є використання буфера перекриття (buffer), який забезпечує коректне виявлення шаблонів, що перетинають межі блоків.Новизна підходу полягає в доповненні цієї базової схеми механізмом оцінювання покриття регулярного виразу. Зокрема, кожен збіг може бути зіставлений із підвиразами або групами шаблону, що дозволяє оцінити, які частини регулярного виразу фактично активуються під час обробки реальних даних. Це відкриває можливість автоматизованого виявлення «сліпих зон» фрагментів сигнатур, які не перевіряються на практиці.Таким чином, запропонована модифікація сигнатурного аналізу поєднує переваги потокової обробки з можливістю кількісної оцінки якості регулярних виразів, що є важливим для підвищення надійності систем кібербезпеки.

Список використаних джерел

1.Wang, Y., Xiang, Y., Zhou, W., &Yu, S. (2012). Generating regular expression signatures for network traffic classification in trusted network management. Journal of Network and Computer Applications, 35(3), 992-1000.

2. Kapoor, M., Fuchs, G., & Quance, J. (2021, November). Rexactor: Automatic regular expression signature generation for stateless packet inspection. In 2021 IEEE 20th international symposium on network computing and applications (NCA) (pp. 1-9). IEEE.

3.Wang, P., & Stolee, K. T. (2018, October). How well are regular expressions tested in the wild?. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 668-678).

4. Larson, E. (2018, September). Automatic checking of regular expressions. In 2018 IEEE 18th international working conference on source code analysis and manipulation (SCAM) (pp. 225-234). IEEE Computer Society.