Optimasi Pencocokan Pola dalam Sistem Deteksi Intrusi Snort Menggunakan GPU

Afrizal Fikri

September 16, 2018

13513004

Overview

Permasalahan

Definisi

Intrusi

Serangkaian percobaan yang tidak berhak baik bertujuan untuk merusak maupun tidak terhadap sumber daya

Sistem deteksi intrusi

Mekanisme yang mengotomasi proses deteksi dan pencegahan intrusi terhadap satu atau beberapa sumber daya

Signature

Pola tertentu yang terdapat pada paket seperti *byte payload* atau servis tertentu

Sistem Deteksi Intrusi

Beberapa metode deteksi intrusi:

1. *Signature-based detection*
   Pencocokan dilakukan berdasarkan *signature* yang telah disiapkan
2. *Anomaly-based detection*
   Pencocokan dilakukan dengan membandingkan profil dengan *threshold* tertentu
3. *Stateful protocol analysis*
   Pencocokan dilakukan secara dalam dengan melihat rangkaian protokol dalam *traffic*

Latar Belakang

- Peningkatan kapasitas *traffic* jaringan dan jenis serangan
- Diperlukan sumber daya besar untuk analisis *traffic* jaringan
- Porsi analisis terbesar ada pada bagian pencocokan string
- Perkembangan GPU yang lebih cepat dan murah sebagai alternatif penggunaan CPU
- Kebutuhan akan implementasi pada GPU yang meminimalkan latensi

Kondisi Saat Ini

-
-

Rumusan Masalah

1. Apa saja rancangan yang akan digunakan dalam pencocokan *signature dengan* GPU?
2. Bagaimana arsitektur yang memadai terkait pencocokan pola dengan GPU?
3. Bagaimana kinerja sistem deteksi intrusi jaringan akibat dari penggunaan GPU?

Tujuan

1. Mencari rancangan-rancangan yang akan digunakan untuk pencocokan signature dengan GPU
2. Mengembangkan sebuah arsitektur yang memadai untuk pencocokan pola dengan GPU
3. Melakukan perbandingan kinerja sistem deteksi intrusi jaringan sebelum dan setelah menggunakan GPU

Analisis

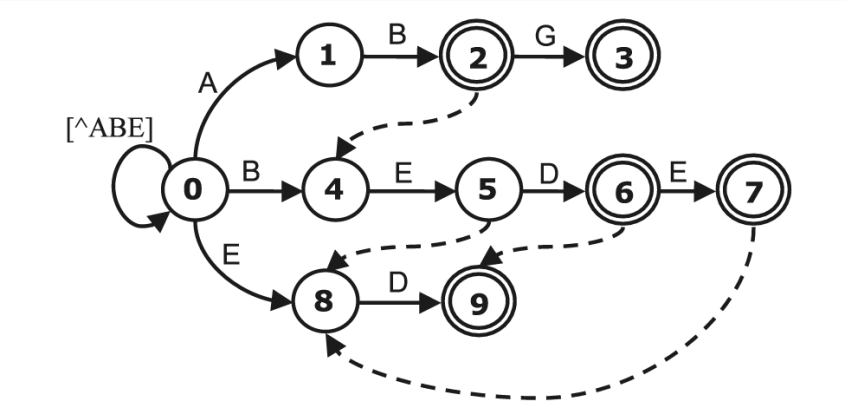
Analisis

- Algoritma pencocokan *signature*
- Struktur penyimpanan kamus
- Alokasi *thread*
- Optimasi latensi pada GPU

Algoritma Pencocokan String

- Pencocokan string menurut sumbernya ada 2 macam: *single pattern* dan *multi pattern*
- Pada *multi pattern string matching*, pola akan dikumpulkan dalam sebuah kamus
- Pencocokan dilakukan dengan pembacaan dari kamus sekaligus

Algoritma Pencocokan Signature



Algoritma akan dikembangkan dengan berbasis algoritma Aho-Corasick karena:

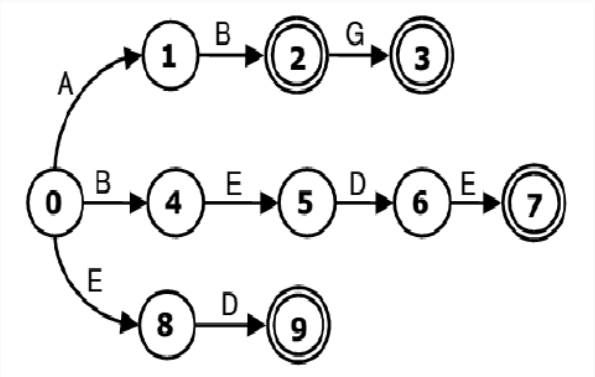
- mampu melakukan pencocokan kamus dalam sekali penelusuran
- stabil terhadap jumlah
- memiliki kinerja paling baik pada IDS Snort

Algoritma Pencocokan Signature

Adaptasi Aho-Corasick untuk memaksimalkan pencocokan secara *multithreading*:

Data Parallel AC
Stream input di partisi menjadi beberapa bagian, kemudian masing-masing akan dilakukan pencocokan dengan satu *thread*

Parallel Failureless AC
Masing-masing *thread* akan memulai pencocokan dari satu karakter dari *stream input* untuk mencari satu pola



Algoritma Pencocokan Signature

DPAC	PFAC
Porsi tiap *thread seimbang*	*Thread* memiliki umur yang pendek
Butuh ekstensi sepanjang pola terpanjang minus satu	Tiap *thread* hanya bertugas mencari satu pola
	Banyak terjadi *overlap* dalam akses *stream input*

Struktur Penyimpanan Kamus

- Pola akan disusun dalam struktur tertentu, seperti *tabel* atau *linked trie*
- Struktur data menentukan operasi pencocokan
- *Spatial locality* dapat berpengaruh pada keseluruhan latensi sistem

Struktur Penyimpanan Kamus

Pendekatan representasi *state machine*:

Linked trie

- Status terenkapsulasi
- Mudah dimodifikasi

Tabel transisi

- Mudah diimplementasi
- Cenderung statik

Alokasi Thread

- *Thread* PFAC akan banyak yang berhenti sangat awal
- Kinerja *warp* menjadi tidak seimbang
- Dapat diatasi dengan skema *assignment* berulang

Optimasi Latensi pada GPU

- Mengurangi transaksi ke memori global
  - Memuat masukan dari memori global ke *shared memory* dengan memanfaatkan *coalescing*
  - Membaca masukan yang *overlap* cukup dari *shared memory*
- Mengurangi latensi ke tabel transisi
  - Mengikat tabel transisi ke memori tekstur
  - Memuat baris pertama ke *shared memory*
- Mengurangi *swappiness* pada *input buffer* dengan *pinned memory*

Rancangan dan Implementasi

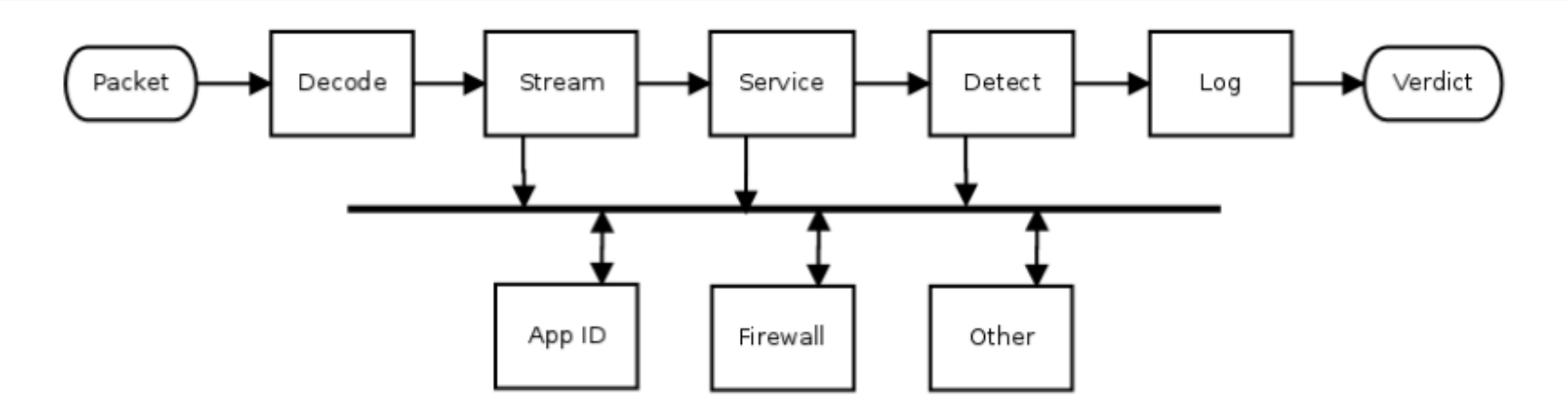
Rancangan Solusi

Pencocokan akan menggunakan implementasi PFAC dengan optimasi dengan memori tekstur dan *pinned memory*

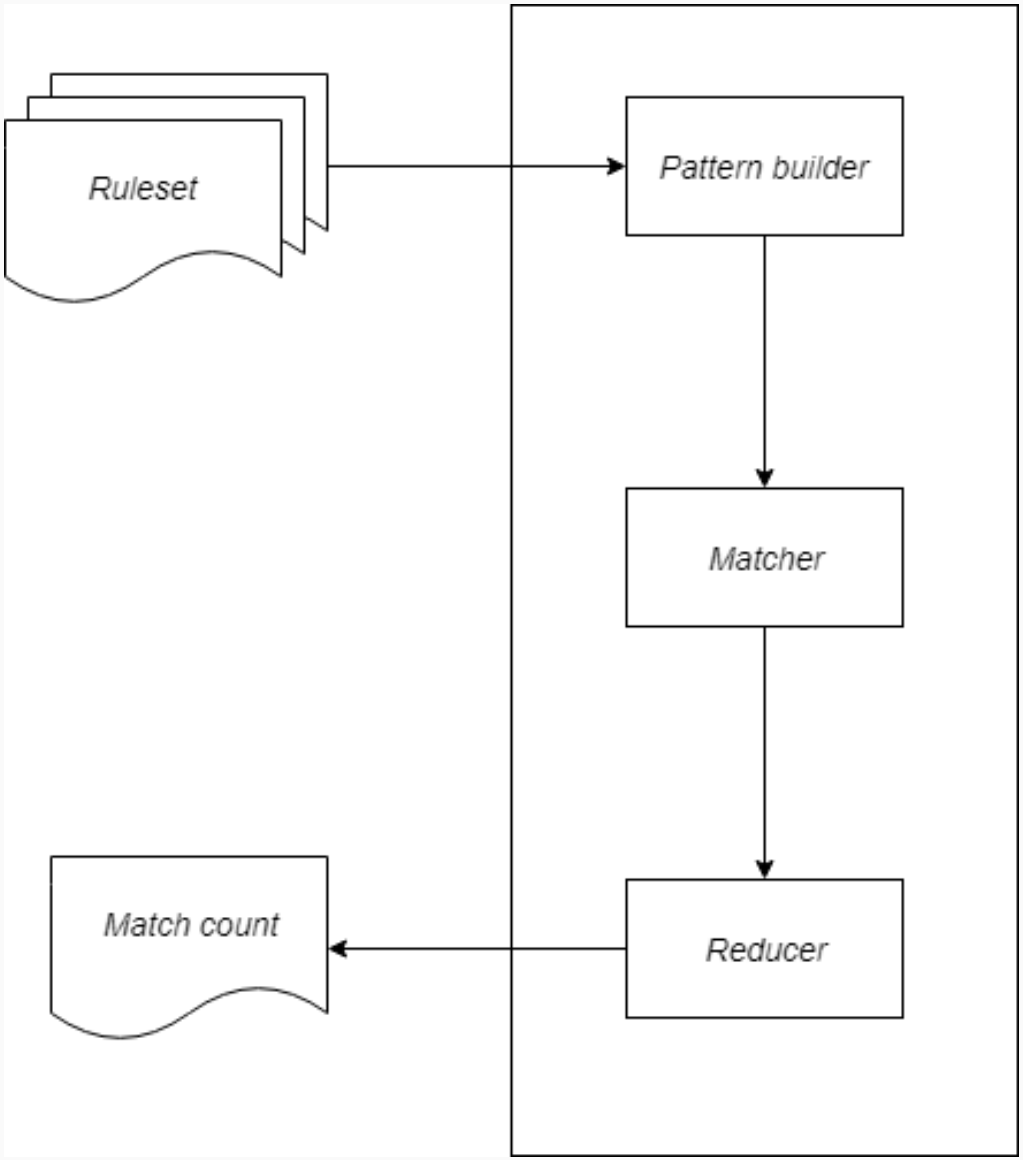
Secara umum, ada 3 tahap yang akan dilakukan oleh modul yang dikembangkan:

1. Pembentukan tabel transisi
2. Pemuatan tabel transisi dalam *device memory*
3. Penelusuran otomata

Arsitektur Snort



Arsitektur Modul



Spesifikasi Komponen

No	Komponen	Spesifikasi
1.	*Handle*	Struktur data yang membungkus fungsionalitas pencocokan untuk memanggil komponen terkait
2.	*State machine*	Kamus yang digunakan dalam pencocokan
3.	*Kernel wrapper*	*Wrapper* yang menyiapkan *payload* sebelum kode *kernel* dijalankan
4.	*Matcher kernel*	Kode GPU yang melakukan penelusuran *state machine*
5.	*Reducer*	Komponen yang menggabungkan hasil pencocokan untuk dikembalikan ke Snort

Demo

Kesimpulan

Summary

Get the source of this theme and the demo presentation from

`github.com/matze/mtheme`

Terima kasih