بهبود و موازی‌سازی سازوکار تشخیص نفوذ شبکهٔ Snort با استفاده از واحد پردازش گرافیکی

مهدی عباسی* و مطهره افشاری حقدوست

گروه مهندسی کامپیوتر، دانشکده مهندسی، دانشگاه بوعلی سینا، همدان، ایران

چکیده

سامانه تشخیص نفوذ شبکه به‌منظور برقراری امنیت کامل در شبکه‌های رایانه‌ای به‌طور گسترده مورداستفاده قرار می‌گیرد. سامانه تشخیص نفوذ شبکه مبتنی بر امضا نسبت به نوع مبتنی بر ناهنجاری به‌دلیل نرخ هشدار اشتباه پایین‌تر، از عمومیت بالاتری برخوردار است. فرآیند تطبیق الگو در چنین دستگاهی نیازمند پردازش محاسباتی بالا است. از سوی دیگر توسعه سریع پهنای باند شبکه و سرعت‌های بالای پیوند که خود موجب ازدست‌رفتن تعداد زیادی از بسته‌های ورودی در سامانه تشخیص نفوذ شبکه می‌شود، به‌عنوان عوامل کلیدی محدودکننده کارایی این نوع سامانه، آن را با چالش‌هایی روبه‌رو کرده است. Snort یک سامانهٔ تشخیص نفوذ شبکه مبتنی بر امضا بوده که به‌دلیل متن‌باز، رایگان و سبک‌بودن بسیار پرکاربرد است. در این مقاله جهت بهبود کارایی سامانه تشخیص نفوذ شبکه snort، از ایده کلیدی فیلترکردن بسته‌های غیرضروری شبکه بر اساس فهرست سیاه نشانی‌ها، به‌عنوان یک سازوکار پیش‌پردازش استفاده شده است. یکی از چالش‌های مهم این سازوکار کاهش سرعت فیلترکردن بسته‌ها، با افزایش حجم ترافیک شبکه است؛ بنابراین به‌عنوان بهبود دوم، جهت تسریع عملکرد این سامانه ارائه‌شده، نسخه موازی آن را روی بستر رمز جهت اجرا روی واحد پردازش گرافیکی ارائه کردیم. الگوریتم پیشنهادی را بر روی مجموعه‌داده DARPA در یک پردازنده گرافیکی آزمایش شد. نتایج ارزیابی نشان می‌دهد که روش پیشنهادی با تسریعی بیش از سی برابر نسبت به نسخه متوالی، باعث بهبود قابل‌توجهی در عملکرد فیلتر بسته مبتنی بر فهرست سیاه می‌شود. همچنین، بهره‌وری روش پیشنهادی در استفاده از منابع پردازنده گرافیکی برای اجرای موازی تشخیص نفوذ نسبت به بهترین روش موجود حدود ۸۱ درصد بیشتر است.

واژگان کلیدی: سامانه تشخیص نفوذ شبکه، فیلتر بسته، فهرست سیاه، تطبیق الگو، واحد پردازش گرافیکی.

Improvement and parallelization of Snort network intrusion detection mechanism using graphics processing unit

Mahdi Abbasi* & Motahhareh Afshari Haghdoost

Department of Computer Engineering, Engineering Faculty, Bu-Ali Sina University, Hamedan, Iran

Abstract

Nowadays, Network Intrusion Detection Systems (NIDS) are widely used to provide full security on computer networks. IDS are categorized into two primary types, including signature-based systems and anomaly-based systems. The former is more commonly used than the latter due to its lower error rate. The core of a signature-based IDS is the pattern matching. This process is inherently a computationally intensive task, and in the worst case, about 80% of the total processing time of an IDS is spent on it. On the other hand, the rapid development of network bandwidth and high link speeds, which in turn leads to a loss of a large number of inbound packets in the network intrusion detection system, has posed challenges as crucial factors limiting the performance of this type of system. Snort is a signature-based NIDS that is highly interested due to being open-source, free, and easy to use. To resolve the challenges mentioned above, we propose an enhanced version of Snort, which is enriched by exploiting two key

* Corresponding author

*نویسندهٔ عهده‌دار مکاتبات

•تاریخ ارسال مقاله: ۱۳۹۷/۱۱/۱۲ •تاریخ پذیرش: ۱۳۹۹/۱۱/۱۱ •تاریخ انتشار: ۱۴۰۰/۰۳/۰۱ •نوع مطالعه: پژوهشی

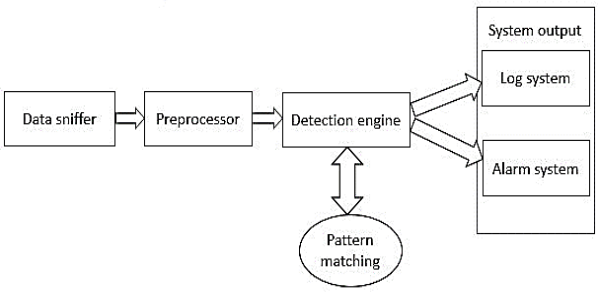
[DOI: 10.52547/jsdp.18.1.150]

ideas. The first idea is the filtering of unnecessary packets based on a blacklist of source IP addresses. This filter is used as a preprocessing mechanism to improve the efficiency of the Snort. However, the packet filtering speed is decreased by increasing the network traffic volumes. Therefore, to accelerate the function of this mechanism, we have proposed a second crucial idea. The data-parallel nature of snort functions lets us parallelize two main computationally intensive functions of it on the graphical processing unit. These functions include the lookup on the blacklist filter in the preprocessing stage and the signature matching of Snort, which completes the intrusion detection process. For parallelizing the preprocessing step of Snort, first, a blacklist is provided from the DARPA dataset. Next, this blacklist is transferred together with the Snort ruleset to the global memory of the GPU. Finally, each thread concurrently matches each packet against the blacklist filters. For parallelizing the signature matching step of Snort, the well-known pattern matching algorithm of Boyer-Moore is parallelized similarly.
Evaluation results show that the proposed method, by up to 30 times faster than the sequential version, significantly improves the blacklist-based filtering performance. Also, the efficiency of the proposed method in using GPU resources for parallel intrusion detection is 81 percent higher than the best state-of-the-art method.

Key words: Network intrusion detection system, packet filter, black list, pattern matching, graphics processing unit.

۱- مقدمه

سامانه‌های تشخیص نفوذ شبکه به‌عنوان یک ابزار امنیتی ضروری جهت حفاظت شبکه‌های مختلف در مقابل نفوذها و فعالیت‌های مخرب به‌کار گرفته می‌شوند [6]. سامانه‌های تشخیص نفوذ شبکه بر اساس نحوه تشخیص نفوذ به دو دسته تقسیم می‌شوند: سامانه تشخیص نفوذ مبتنی بر ناهنجاری و سامانه تشخیص نفوذ مبتنی بر امضا [12-7]. در نوع نخست حملات و تهدیدها توسط مقایسه رویدادهای جاری با وقایع عادی از پیش تعریف‌شده، تشخیص داده می‌شوند؛ درحالی‌که در نوع دوم نفوذها با مقایسه سرآیند و محتوای بسته با قوانین از پیش‌ساخته‌شده موجود در یک پایگاه داده، شناسایی می‌شوند [5-3]. سامانه‌های تشخیص نفوذ شبکه مبتنی بر امضا به‌دلیل نرخ هشدار پایین نسبت به نوع مبتنی بر ناهنجاری از عمومیت بالاتری برخوردار هستند [10-7]. نرم‌افزار snort یک سامانه تشخیص نفوذ مبتنی بر امضا است که به‌دلیل رایگان، متن‌باز و سبک‌بودن، مورداستفاده بسیاری از کاربران قرار می‌گیرد. سامانه تشخیص نفوذ snort همان‌طور که در شکل (۱) نشان داده شده از چهار جزء اصلی تشکیل شده که شامل شنودکننده داده، پیش‌پردازش‌گر، موتور تشخیص و سامانه ثبت و هشدار است [13].



(شکل-۱): معماری سامانهٔ تشخیص نفوذ snort [1]
(Figure-1): The architecture of Snort IDS [1]

در ابتدا بسته‌های ضبط‌شده توسط شنودکننده به قسمت پیش‌پردازش‌گر ارسال می‌شوند؛ سپس بعد از تجزیه‌وتحلیل بسته توسط افزونه‌های موجود در بخش پیش‌پردازش جهت تطبیق با مجموعه قوانین snort به بخش موتور تشخیص فرستاده شده و درنهایت در صورت تطبیق، سامانه خروجی هشداری مبنی بر وجود حمله صادر می‌کند. با توجه به این‌که موتور تشخیص اصلی‌ترین بخش سامانه تشخیص نفوذ است، تعداد و پیچیدگی امضاها و همچنین بررسی تعداد بسته‌های ورودی عمل مؤثر در کارایی سامانه تشخیص نفوذ است [14].

باوجوداینکه در سال‌های اخیر قدرت محاسباتی رایانه‌ها افزایش یافته اما به‌دلیل حجم بالای ترافیک و افزایش تعداد و پیچیدگی امضاها، قابلیت محاسباتی محدود همچنان یک موضوع اساسی برای سامانه‌های تشخیص نفوذ شبکه مبتنی بر امضا است. به‌عنوان نمونه حجم ترافیک بالا در شبکه‌های پرسرعت موجب می‌شود که حافظه در سامانه تشخیص نفوذ snort به‌سرعت پایان یابد و درنتیجه، snort تعداد زیادی از بسته‌های ورودی را از دست می‌دهد. همچنین در ترافیک وب با حجم بالا، snort به‌طورتقریبی هشتاد درصد زمان کل پردازش خود

[DOI: 10.52547/jsdp.18.1.150]

را صرف تطبیق رشته ورودی با امضاهای ذخیره‌شده می‌کند [18-15]. با توجه به آنچه بیان شد، افزایش روزافزون حملات و نفوذهای شبکه و از سوی دیگر با توجه به توسعه سریع فناوری سخت‌افزار و پهنای باند شبکه و سرعت‌های بالای پیوند، مشکلاتی را برای سامانه تشخیص نفوذ شبکه مطرح کرده است. به‌منظور مقابله با مشکلات و بهبود کارایی یک سامانه تشخیص نفوذ شبکه به‌خصوص سامانه تشخیص نفوذ شبکه snort راه‌کارهای متفاوتی که شامل موازی‌سازی الگوریتم تطبیق الگو در جهت تسریع عملکرد موتور تشخیص، دسته‌بندی یا فیلتر بسته‌های شبکه به‌عنوان یک سازوکار پیش‌پردازش و موازی‌سازی بخش پیش‌پردازش است، ارائه شده است.

در این مقاله، جهت افزایش کارایی سامانه تشخیص نفوذ snort در شبکه‌های پرسرعت روش ارائه‌شده در [2] را که روشی کارا جهت فیلترکردن بسته‌های شبکه در بخش پیش‌پردازش، مبنای کار خود قرار داده و با موازی‌سازی فیلتر بیان‌شده توسط واحد پردازش گرافیکی کارایی فیلتر و درنتیجه سامانه تشخیص نفوذ شبکه را ارتقا داده‌ایم. فیلتر یادشده با استفاده از ایده کلیدی فیلترکردن بسته‌های غیرضروری بر اساس فهرست سیاه باعث کاهش بسته‌های ارسالی به سامانه تشخیص نفوذ شبکه و درنتیجه باعث بهبود کارایی سامانه‌های تشخیص نفوذ می‌شود؛ اما، افزایش حجم بسیار زیاد ترافیک و تهدیدات در شبکه‌های پرسرعت، چالش‌هایی را برای این فیلتر نیز به وجود می‌آورد؛ بنابراین، ایده تسریع در عملکرد فیلتر یاده شده و همچنین تطبیق الگوهای نفوذ، به‌عنوان یک راه‌کار پیش‌پردازش می‌تواند موجب ارتقای عملکرد سامانه تشخیص نفوذ شبکه شود.

مقاله ارائه‌شده بدین صورت سازمان‌دهی شده است: در بخش دوم، پس از معرفی واحد پردازش گرافیکی، قوانین و نحوه فرآیند تطبیق الگو در سامانه تشخیص نفوذ snort، به بررسی و مرور برخی از پژوهش‌های مرتبطی که در زمینه بهبود عملکرد سامانه تشخیص نفوذ شبکه انجام‌شده، پرداخته خواهد شد. در بخش سوم، نحوه موازی‌سازی الگوریتم فیلترکردن بسته‌های شبکه و اجرای آن توسط واحد پردازش گرافیکی توضیح داده و در بخش چهارم مقاله، نتایج ارزیابی به‌دست‌آمده از موازی‌سازی روش پیشنهادی بر روی مجموعه داده DARPA از نقطه‌نظر دقت، حافظه، تسریع و گذرداد بررسی می‌شود و درنهایت در بخش پنجم به نتیجه‌گیری و ارائه راه‌کارهایی جهت پیشرفت پژوهش موجود اختصاص می‌یابد.

۲- مفاهیم بنیادی

در این بخش در ابتدا به معرفی قوانین سامانه تشخیص نفوذ شبکه snort و فرآیند تطبیق الگو پرداخته می‌شود؛ سپس ساختار واحد پردازش گرافیکی بررسی شده و درنهایت، برخی از کارهای انجام‌شده را در راستای بهبود عملکرد سامانه تشخیص نفوذ شبکه، مرور می‌کنیم.

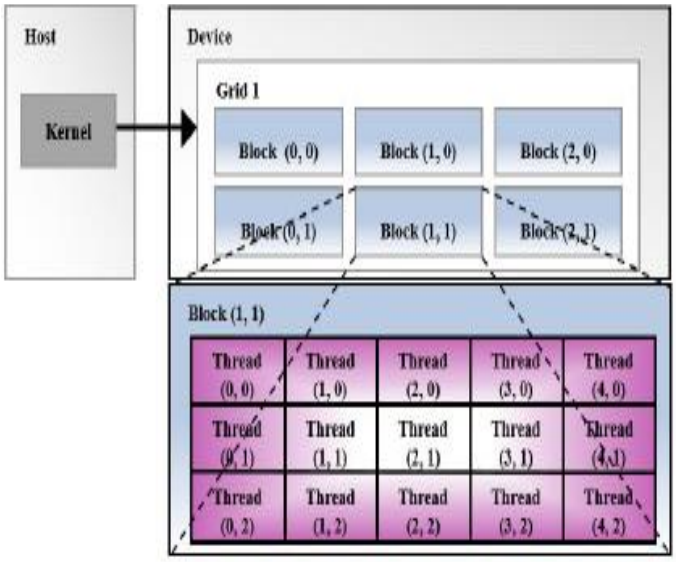
۲-۱- سامانهٔ تشخیص نفوذ شبکه snort

سامانه تشخیص نفوذ شبکه مبتنی بر امضا snort، یک نرم‌افزار امنیتی متن‌باز و رایگان است که به‌طور گسترده مورداستفاده قرار می‌گیرد. در snort، یک قانون شامل دو قسمت سرآیند و بدنه است. قسمت سرآیند شامل فیلد پروتکل، نشانی IP مبدأ و مقصد، شماره پورت مبدأ و مقصد است و قسمت بدنه شامل رشته‌هایی جهت تطبیق الگو، پیام، آفست، عمق، جریان و غیره است. به‌طورکلی، در سامانه تشخیص نفوذ snort قوانین بر اساس شماره پورت مبدأ و مقصد و نشانی IP مبدأ و مقصد دسته‌بندی می‌شوند. در این سامانه، هنگامی‌که سرآیند بسته‌های ورودی با فیلدهای متناظر در سرآیند قانون‌های مربوطه تطبیق داده شود، آن‌گاه محتوای بسته توسط الگوریتم تطبیق الگو با رشته موجود در بدنه قانون‌ها تطبیق داده می‌شود؛ در صورت تطبیق، هشداری از سوی سامانه صادر خواهد شد.

الگوریتم‌های تطبیق الگو در دو دسته، تطبیق الگوی منفرد و تطبیق الگوی چندگانه تقسیم می‌شوند. در الگوریتم‌های تطبیق الگوی منفرد، هر الگو در رشته ورودی جداگانه جستجو می‌شود؛ درحالی‌که در الگوریتم‌های تطبیق الگوی چندگانه، با ساخت ماشین حالت و انتقال آن در حافظه، مجموعه‌ای از الگوها به‌صورت موازی در رشته ورودی جستجو می‌شوند [19]. Boyer-Moore که به‌اختصار BM نامیده می‌شود، یکی از پرکاربردترین الگوریتم تطبیق الگوی منفرد است. پیچیدگی زمان اجرای این الگوریتم، هنگامی‌که پسوند رشته در یک جریان ورودی به‌ندرت ظاهر شود، به‌دلیل وجود حالت پرش، خطی می‌شود [20].

Aho-Corasick که به‌اختصار AC نامیده می‌شود، نیز یکی از پرکاربردترین الگوریتم تطبیق الگوی چندگانه است. AC یک الگوریتم سریع است که با ساخت ماشین خودکار متناهی قطعی، جستجوی چند الگویی را انجام می‌دهد. الگوریتم AC در ابتدا یک ماشین تطبیق الگو حالت متناهی جهت جستجو از مجموعه الگوهای مربوطه می‌سازد؛ سپس رشته ورودی را توسط این ماشین در یک مرحله پردازش می‌کند. به‌طورکلی عملکرد الگوریتم AC به‌وسیلهٔ سه تابع تعیین می‌شود که این توابع شامل تابع goto جهت رفتن از حالت جاری به حالت بعدی، تابع failure جهت رفتن به حالت بعدی هنگامی‌که تطبیقی رخ ندهد و درنهایت تابع output جهت دریافت مجموعه الگوهایی که در حالت جاری تطبیق داده شده، است [21].

[DOI: 10.52547/jsdp.18.1.150]



(شکل-۲): لوک و نخ در واحد پردازش گرافیکی [10]
(Figure-2): Block and Thread in GPU

۲-۲- واحد پردازش گرافیکی

واحد پردازش گرافیکی که جهت نمایش تصاویر گرافیکی به‌کار می‌رود، به‌عنوان یک واحد محاسباتی موازی قدرتمند است و به‌جای واحد پردازش مرکزی، نقش کلیدی را در تسریع پردازش‌های محاسباتی سنگین می‌تواند بر عهده گیرد. دلیل اصلی این تحول بزرگ محاسباتی آن است که درگذشته افت شدیدی در عملکرد پردازنده مرکزی به دلیل پردازش تصویر وجود داشت. درنتیجه، معماری واحد پردازش گرافیکی، به‌خصوص برای انجام محاسبات فشرده و عملیات موازی طراحی‌شده است.

واحد پردازش مرکزی و واحد پردازش گرافیکی ازلحاظ معماری رایانه، تفاوت‌های چشم‌گیری باهم دارند. واحد پردازش گرافیکی برخلاف واحد پردازش مرکزی که دارای چند هسته با تعداد زیادی حافظه نهان است، از صدها هسته تشکیل شده است که می‌تواند به‌طور هم‌زمان هزاران کار را در حالت اجرا داشته باشد. درنتیجه می‌تواند سرعت اجرای برخی از پردازش را صدبرابر واحد پردازش مرکزی شتاب دهد. در این راستا به‌منظور اجرای محاسبات غیر گرافیکی روی پردازنده گرافیکی شرکت انودیا[1]، نرم‌افزار رمز را عرضه کرد که یک مدل برنامه‌نویسی و سکوی پردازش موازی است [22].

معماری یک واحد پردازش گرافیکی در شکل (۲) نشان داده شده است، ازلحاظ سخت‌افزاری هر پردازنده گرافیکی از تعدادی چندپردازنده‌های جریانی[2] یا به‌اختصار SM تشکیل شده‌اند. پردازنده‌های جریانی[3] یا به‌اختصار SP بین این چندپردازنده‌های جریانی تقسیم می‌شوند. در هر SM چندین وارپ[4] تعریف می‌شود و هر وارپ از ۳۲ پردازش نخی تشکیل شده است .هر SM دارای یک حافظه اشتراکی و یک حافظه نهان L1 است. همچنین حافظه نهان L2 در یک پردازنده گرافیکی بین تمام SM ها و حافظه سراسری قرار دارد. هر برنامه‌ای که در کودا نوشته می‌شود، هسته[5] نامیده شده و هر هسته توسط یک گرید[6] که خود از چندین بلوک[7] تشکیل‌شده اجرا می‌شود. هر بلوک نیز از چندین نخ اجرایی تشکیل شده است. به‌طورکلی هر بلوک به‌وسیلهٔ یک SM اجرا می‌شود؛ اما یک SM می‌تواند چندین بلوک را به‌صورت هم‌روند نیز اجرا کند [23].

هر واحد پردازش گرافیکی دارای انواع مختلفی از حافظه با ویژگی‌های متفاوت ازلحاظ دسترسی است. این مجموعه، شامل حافظه سراسری[8]، حافظه ثابت[9]، حافظه بافت[10]، حافظه محلی[11]، حافظه مشترک[12] و ثبات[13] است. حافظه سراسری، بزرگ‌ترین حافظه با قابلیت ذخیره‌سازی صدها مگابایت داده است؛ اما تأخیر دسترسی در این حافظه نسبت به انواع دیگر حافظه‌ها بیشتر است. حافظه ثابت، شبیه حافظه سراسری بوده با این تفاوت که پهنای باند موردنیاز آن بسیار کمتر از حافظه سراسری است؛ بنابراین سرعت انتقال داده‌های آن بسیار بیشتر خواهد بود.

[1] Nvidia
[2] Streaming Multiprocessor
[3] Streaming Processor

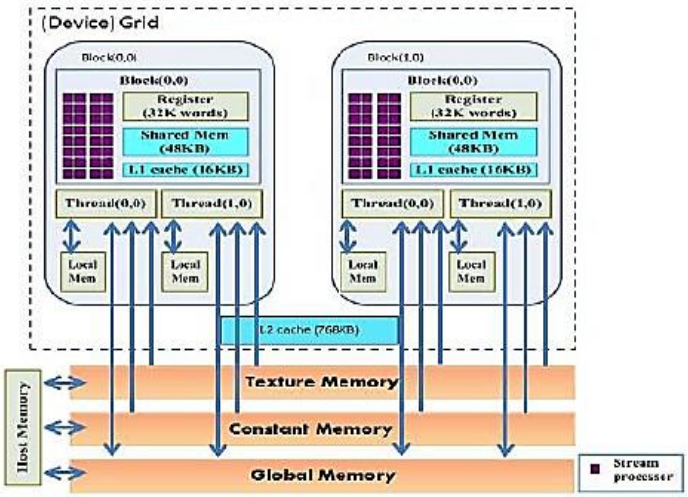
[4] Warp
[5] Kernel
[6] Grid
[7] Block
[8] Global memory
[9] Constant memory
[10] Texture memory
[11] Local memory
[12] Shared memory
[13] Register

[DOI: 10.52547/jsdp.18.1.150]

اندازه حافظه ثابت کوچک است و به‌صورت فقط خواندنی مورد استفاده قرار می‌گیرد. حافظه بافت حد وسط بین حافظه سراسری و حافظه ثابت است. حافظه بافت مانند حافظه ثابت یک حافظه فقط خواندنی است با این تفاوت که به‌دلیل بهره‌گیری از الگوهای دسترسی به حافظه موجب کاهش زمان تأخیر دسترسی به حافظه می‌شود. حافظه محلی، دارای ظرفیت محدود و محل نگهداری متغیرهای محلی پردازش‌های نخی بوده و به‌ازای هر نخ اجرایی قسمتی از این حافظه در اختیار آن قرار می‌گیرد. حافظه مشترک، امکان همکاری بین نخ‌های یک بلوک خاص را فراهم می‌کند. درواقع حافظه محلی درون یک بلوک بوده و به‌گونه‌ای است که هرکدام از نخ‌های بلوک اجازه دسترسی به آن را دارند و دارای ظرفیتی بیش‌تر از حافظه محلی است. ثبات، عنصر ذخیره‌ساز در سطح پردازنده گرافیکی است که سرعت آن نسبت به سایر عناصر ذخیره‌ساز بالاتر بوده و قیمت آن نیز بیشتر است.

به‌ازای هر نخ یک ثبات وجود دارد که ظرفیت آن حد میان حافظه محلی و حافظه مشترک و همچنین یک ذخیره‌ساز سریع و محلی برای نخ‌ها در حین اجرای هسته است. شکل (۳) سلسله‌مراتب حافظه را نشان می‌دهد که از پایین به سمت بالا ظرفیت حافظه‌ها کم؛ ولی سرعت آن‌ها زیاد می‌شود. درنتیجه، هنگامی‌که حجم داده‌های ورودی بسیار بالا باشد باید از حافظه سراسری استفاده شود [23, 24].

به‌دلیل وجود تعداد قابل‌توجهی هسته پردازش موازی در کنار سلسله‌مراتب حافظه با ویژگی‌های متفاوت ازلحاظ پهنای باند و تأخیر دسترسی، در کنار قیمت بسیار کمتر در واحد پردازش گرافیکی نسبت به سامانه‌های چندپردازنده متداول، تاکنون، چندین کار ازجمله [31-25] با استفاده از بستر کودا برای پیاده‌سازی موازی الگوریتم‌های دسته‌بندی بسته‌های شبکه و همچنین موازی‌سازی توابع شبکه از قبیل جستجوی IP در جداول مسیریابی باهدف دست‌یابی به گذرداد بالاتر، پرداخته‌اند.



(شکل-۳): ساختار سلسله مراتبی حافظهGPU [۱۱]

(Figure-3): Hierarchical memory system of GPU

۳- پیشینهٔ پژوهش

مروری بر پژوهش‌های اخیر در زمینه موازی‌سازی بخش‌های مختلف سامانه تشخیص نفوذ شبکه، نشان می‌دهد که نیاز به انجام پژوهش‌های بیشتر درزمینه موازی‌سازی الگوریتم‌های تشخیص نفوذ وجود دارد. هرچند رویکردهای متفاوتی درزمینه تسریع این نوع الگوریتم‌ها وجود دارد، اما رویکردهای موجود موضوع افزایش نرخ ترافیک را در شبکه‌های رایانه‌ای در نظر نگرفته و بهبود قابل‌توجهی در کارایی زمانی الگوریتم‌های تشخیص نفوذ ایجاد نکرده‌اند.

به‌منظور بهبود کارایی سامانه تشخیص نفوذ شبکه راه‌کارهای متفاوتی ارائه شده است که از دیدگاه افزایش سرعت تشخیص نفوذ شبکه به‌طورکلی در دو دسته تقسیم می‌شوند. دسته نخست رویکردهایی هستند که در آن‌ها بهبود الگوریتم تطبیق الگو جهت تسریع عملکرد موتور تشخیص موردتوجه قرارگرفته است. در دسته دوم به دسته‌بندی یا فیلتر بسته‌های شبکه به‌عنوان یک سازوکار پیش‌پردازش جهت تسریع عملکرد موتور تشخیص پرداخته می‌شود.

ابتدا، به بررسی و مرور مهم‌ترین راه‌کارهای نوع نخست می‌پردازیم. به‌عنوان‌مثال، Vasiliadis و همکاران در سال ۲۰۰۸ یک سامانه آزمایشی به نام Gnort را بر مبنای سامانه تشخیص نفوذ شبکه snort ارائه دادند [32]. در این مقاله الگوریتم تطبیق الگو Aho-Corasick در واحد

[DOI: 10.52547/jsdp.18.1.150]

پردازش گرافیکی با دو روش پیکربندی نخ‌ها اجراشده است. در روش نخست هر بسته توسط یک بلوک نخ و در روش دوم هر بسته توسط یک نخ متفاوت پردازش می‌شود که درنهایت هر دو روش منجر به تسریع پردازش با ضریب دو شد. همچنین، در پژوهش بعدی Vasiliadis و همکاران در سال ۲۰۰۹ کارایی سامانه تشخیص نفوذ شبکه به‌وسیلهٔ اجرای الگوریتم تطبیق عبارات منظم بر روی واحد پردازش گرافیکی شصت درصد بهبود پیدا کرد[33] . در ادامه این کار، Vasiliadis و همکاران در سال ۲۰۱۱ یک معماری سامانه تشخیص نفوذ موازی چندگانه برای شبکه‌های پرسرعت معرفی کردند[25]. این معماری از دو قسمت تشکیل‌شده است. قسمت نخست وظیفه دارد که از کارت‌های شبکه چندصفی بسته‌های شبکه را دریافت کرده و توسط هسته‌های پردازنده مرکزی دسته‌بندی کند. قسمت دوم، مسئول اجرای الگوریتم تطبیق الگو AC-compact در واحد پردازش گرافیکی است. در این معماری به‌منظور بهبود کارایی سامانه تشخیص نفوذ برای اجرای الگوریتم تطبیق الگو، از چندپردازنده گرافیکی استفاده شده که درنهایت منجر به دست‌یابی بیشینه توان پردازشی هفتاد گیگا بیت بر ثانیه می‌شود.

Hung و همکاران در سال ۲۰۱۲ یک معماری جدول درهم سازی سلسه‌مراتبی جهت اجرای موازی الگوریتم تطبیق الگو بر روی یک پردازنده گرافیکی توسعه دادند[22]. در این الگوریتم هر بسته به بخش‌های ثابت و مساوی تقسیم شده و هر نخ هر قسمت از بسته را به‌صورت موازی جستجو می‌کند؛ درنتیجه، منجر به تسریع عملکرد تطبیق الگو و افزایش توان پردازشی ترافیک به‌اندازه ۲/۴ گیگا بیت در ثانیه شد.

Jamshed و همکاران در سال ۲۰۱۲ یک نرم‌افزار با مقیاس‌پذیری بالا را بر مبنای سامانه تشخیص نفوذ تحت عنوان Kargus به‌منظور بهره‌برداری کامل از پتانسیل محاسباتی سخت‌افزار معرفی کردند [34]. در آن پژوهش نشان داده شده است که Kargus بسته‌های ورودی را به‌صورت دسته‌ای در کارت شبکه پردازش می‌کند و درنهایت نرخ ورودی تا چهل گیگا بیت در ثانیه، حتی برای بسته‌هایی با کمینه اندازه، افزایش می‌یابد. این نرم‌افزار با برقراری توازن در حجم پردازش الگوریتم تطبیق الگو بین پردازنده‌های چندهسته‌ای و پردازنده‌های گرافیکی موجب ایجاد موازت قابل‌توجهی در پردازش می‌شود. درواقع kargus با تحقق دو اصل کلیدی دسته‌بندی و موازی‌سازی به‌طور چشم‌گیری موجب بهبود کارایی می‌شود.

همان‌طور که درقبل بیان شد، دسته دوم راه‌کارها از ایده دسته‌بندی و فیلتر بسته‌های شبکه به‌عنوان یک سازوکار پیش‌پردازش استفاده می‌کنند. دسته‌بندی موجب کاهش قوانین مورداستفاده در فرآیند تطبیق الگو است. از سوی دیگر، فیلتر بسته‌های شبکه موجب کاهش‌یافتن تعداد بسته‌های هدف و درنتیجه کاهش بار محاسباتی سامانه تشخیص نفوذ شبکه می‌شود. برای نمونه Song و همکاران در سال ۲۰۰۵ یک معماری دسته‌بندی بسته برای سامانه تشخیص نفوذ شبکه مبتنی بر FPGA تحت عنوان BV-TCAM در مقاله خود ارائه دادند [35]. در این کار از حافظه تداعی‌گر سه‌وضعیتی[1] جهت اجرای جستجوی موازی موردنیاز در دسته‌بندی بسته‌ها بر اساس فیلدهای پروتکل و نشانی IP طبق الگوریتم بردار بیتی استفاده کردند. موتور جستجوی پیشنهادی در آن پژوهش قادر به اجرای متوسط ۱۲/۵ جستجو در ثانیه بر روی مجموعه قوانین snort شد.

Meng و همکارش در سال ۲۰۱۴ از یک فیلتر که دارای فهرست سیاه برای فیلترکردن بسته‌های شبکه بر مبنای نشانی IP بود، استفاده کرد [36]. معماری فیلتر بیان‌شده شامل فهرست سیاه و جدول جستجو است. فهرست سیاه شامل نشانی‌های IP منابع مخرب بوده و به‌صورت برخط در یک بازه زمانی مشخص به‌روزرسانی می‌شود. به‌طورکلی هنگامی‌که بسته ارسال می‌شود، نشانی IP آن در فهرست سیاه جستجو شده و در صورت وجود، با امضاهای موجود در جدول جستجو که بر مبنای مجموعه قوانین snort است، تطبیق داده و در انتها، در صورت تطبیق هشداری صادر و بسته فیلتر می‌شود؛ درنتیجه، به سامانه تشخیص نفوذ شبکه ارسال نخواهد شد.

Hung و همکاران در سال ۲۰۱۴ الگوریتم تشخیص نفوذ Aho-Coarsic را برای بررسی بسته‌های با اندازه یکسان بر روی پردازنده گرافیکی Geforce GTX980 به‌صورت موازی پیاده‌سازی کردند. آن‌ها توانستند به و گذردادی در حدود ۲/۵۶ گیگابیت در ثانیه و تسریعی در حدود ده برابر نسبت به پیاده‌سازی متوالی همان الگوریتم بر روی پردازنده مرکزی دست یابند [5].

Hung و همکاران در سال ۲۰۱۵ الگوریتم Bloom Filter را برای بررسی محتوای دادهٔ حمل‌شده به‌وسیلهٔ بسته‌های شبکه با اندازه‌های متفاوت بر روی پردازنده‌های گرافیکی به‌صورت موازی پیاده‌سازی کردند. بیشینه میزان تسریع حاصل از اجرای موازی الگوریتم نسبت به نسخه

[1] Ternary Content Addressable Memory (TCAM)

[DOI: 10.52547/jsdp.18.1.150]

متوالی آن ۵/۴ برابر و بیشینه گذرداد نسخه موازی دو گیگابیت بر ثانیه گزارش شده است [4].

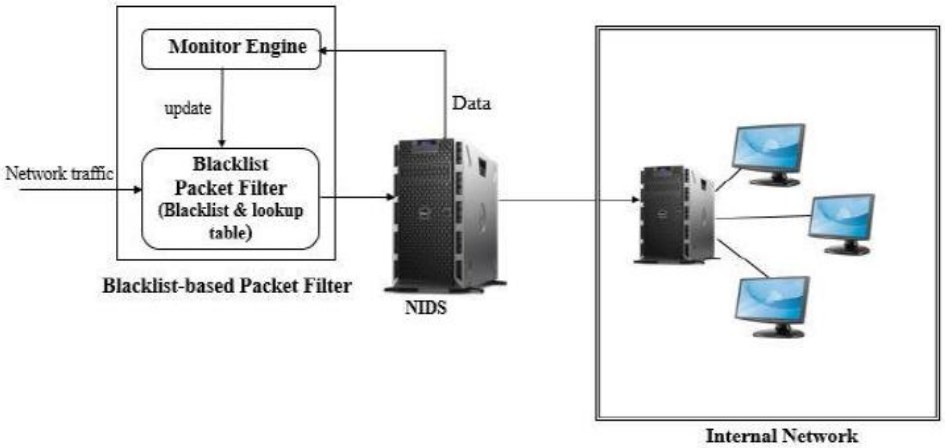
Ho و همکاران در سال ۲۰۱۸ از موازی‌سازی الگوریتم درهم‌سازی Cuckoo-hashing بر روی پردازنده گرافیکی Geforce 660 برای تطبیق هم‌زمان بسته ورودی با چند الگو استفاده کردند [3]. نتایج پژوهش مرتبط مؤید برتری روش Ho و همکاران نسبت به روش Hung و همکاران در موازی‌سازی Bloom-Filter بوده و از منظر سرعت حدود ۶۹ درصد بهبود را نشان می‌دهد.

Ramesh و همکاران در سال ۲۰۱۸ از الگوریتم‌های بررسی سرآیند بسته به‌صورت موازی در سطح پردازش‌های نخی و یا در سطح پردازش‌های بلوکی در کنار نسخه موازی الگوریتم‌های Rabin-Karp، Aho-Corasic و Wu-Manber بر روی پردازنده گرافیکی K80، برای وارسی عمیق بسته‌ای شبکه استفاده کردند [37]. نتایج پژوهش آن‌ها نشان می‌دهد موازی‌سازی در سطح بلوک بسیار کاراتر از موازی‌سازی در سطح نخ‌ها بوده و تا ۱۱۶ برابر تسریع حاصل می‌شود.

با توجه به کارهای مرورشده، روش‌های موجود در کسب سطح قابل قبولی از بهره‌وری در استفاده از منابع پردازشی برای موازی‌سازی تشخیص نفوذ ناتوان بوده‌اند. همچنین، در هیچ‌یک از کارهای اخیر موازی‌سازی بخش پیش‌پردازش سامانه تشخیص نفوذ شبکه به‌عنوان یک راه‌کار جهت تسریع عملکرد کلی سامانه تشخیص نفوذ مورد تجزیه‌وتحلیل قرار نگرفته است؛ بنابراین در این مقاله با ارائه یک روش جهت موازی‌سازی راه‌کار فیلتر بسته‌های شبکه به‌عنوان یک سازوکار پیش‌پردازش، سعی در بهبود عملکرد سامانه‌های تشخیص نفوذ شبکه از نظر بهره‌وری داریم.

۴- روش پیشنهادی

در هر شبکه که شامل سامانه تشخیص نفوذ شبکه باشد، لازم است که در ابتدا بسته‌های شبکه از یک فاز پیش‌پردازش عبور کنند. به‌این‌ترتیب، از ورود حجم بالای ترافیک بسته‌های شبکه به سامانه تشخیص نفوذ شبکه جلوگیری می‌شود. شکل (۴) هم‌بندی اصلی فیلتر یادشده را نشان می‌دهد که توسط Meng و همکارش در سال ۲۰۱۴ به‌منظور تقسیم بار ترافیکی مناسب در حجم بالای ترافیک، پیشنهاد شده است [2]. در این مقاله به‌منظور ارتقای عملکرد سامانه تشخیص نفوذ شبکه از راه‌کار دوم یعنی بهبود بخش پیش‌پردازش استفاده شده است. به این منظور، روش پیشنهادی [2] را مبنای کار خود قرار داده و الگوریتم آن را با هدف تسریع عملکرد آن، بر روی واحد پردازش گرافیکی موازی و اجرا می‌کنیم. همان‌طور که در بخش دوم مقاله بیان شد، واحد پردازش گرافیکی دارای چندین نوع حافظه با ظرفیت‌ها و دسترسی‌های متفاوت است. با توجه به حجم زیاد مجموعه بسته‌ها، فهرست سیاه و همچنین مجموعه قوانین snort، که حدود چندین مگابایت است، از حافظه سراسری واحد پردازش گرافیکی جهت ذخیره‌سازی ساختار داده الگوریتم استفاده می‌کنیم.



(شکل-۴): ساختار فیلتر مبتی بر لیست سیاه [2]

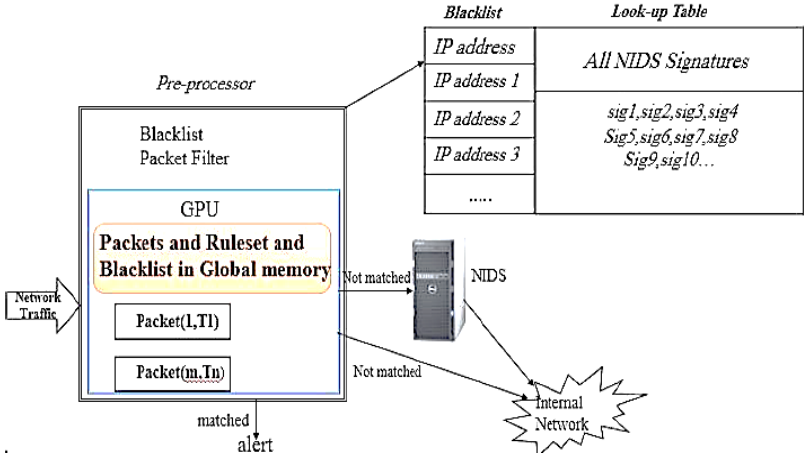
(Figure-4): The structure of a black-list-based filter

معماری موردنظر شامل دو بخش اصلی فیلتر بسته فهرست سیاه و موتور نظارت است. فیلتر بسته فهرست سیاه شامل دو قسمت به نام‌های فهرست سیاه و جدول جستجو است و بسته‌های شبکه را بر اساس نشانی IP مبدأ و قوانین موجود، فیلتر می‌کند. از سوی دیگر، موتور نظارت نیز فهرست سیاه را به‌صورت برخط بر اساس اطلاعات ورودی از سامانه تشخیص نفوذ شبکه، به‌روزرسانی می‌کند.

[DOI: 10.52547/jsdp.18.1.150]

بر اساس اطلاعات موجود در سامانه سیاه سه حالت مختلف پاسخ‌دهی وجود دارد. هنگامی‌که بسته ورودی شبکه ابتدا به فاز پیش‌پردازش ارسال می‌شود، نشانی IP مبدأ آن بررسی می‌شود؛ اگر این نشانی در فهرست سیاه وجود داشته باشد، جهت بررسی با مجموعه امضاهای موجود در جدول جستجو که بر اساس قوانین snort است، تطبیق داده می‌شود. در صورت تطبیق، آن بسته فیلتر و درصورت عدم تطبیق، به شبکه ارسال می‌شود. همچنین، درصورتی‌که نشانی IP مبدأ بسته در فهرست سیاه وجود نداشته باشد، آن بسته به سامانه تشخیص نفوذ شبکه ارسال می‌شود [2].

بیشترین زمان اجرای الگوریتم فیلترکردن بسته‌های شبکه صرف بررسی نشانی IP در فهرست سیاه و سپس تطبیق قوانین موجود در جدول جستجو می‌شود؛ بنابراین، با افزایش حجم بسیار زیاد ترافیک و تهدیدات در شبکه‌های پرسرعت، فیلتر یادشده نیز دچار مشکل شده و سرعت بررسی بسته‌های شبکه در آن کاهش پیدا می‌کند. در روش پیشنهادی به‌منظور مقابله با این مشکل، الگوریتم فیلترکردن بسته‌های شبکه را (همان‌طور که در شکل (۵) نشان داده شده است) بر روی واحد پردازش گرافیکی اجرا کردیم. بدین‌صورت که پس از تهیه فهرست سیاه و بر مبنای ارزیابی تشخیص نفوذ darpa، همراه با بسته‌های شبکه و قوانین snort، آن‌ها به حافظه سراسری پردازنده گرافیکی انتقال دادیم؛ سپس هر بسته توسط یک نخ اجرایی به‌صورت جداگانه به‌صورت موازی بر اساس الگوریتم فیلتر بسته مبتنی بر فهرست سیاه پردازش می‌شود. با توجه به شبه کدی که در شکل (۵) ارائه‌شده است، P بیان‌گر بسته‌ها، B بیان‌گر فهرست سیاه و R مجموعه‌قوانین snort است. هنگامی‌که بسته‌های ورودی در بافرهایی با اندازه‌های مختلف به‌طور کامل دریافت شدند و بافر پر شد، همراه با فهرست سیاه که به‌صورت برون‌خط و بر مبنای ارزیابی تشخیص نفوذ darpa تهیه‌شده (خط ۱) و مجموعه قوانین snort به حافظه سراسری واحد پردازش گرافیکی که نسبت به انواع دیگر حافظه دارای ظرفیت بالایی است، انتقال پیدا می‌کنند. (خط ۲). با توجه به خط چهارم به هر بسته یک نخ جداگانه اختصاص داده شده و هر بسته به‌وسیلهٔ یک نخ متفاوت پردازش می‌شود. هنگامی‌که تعداد بسته‌ها از تعداد نخ‌های بلاک تعریف‌شده بیش‌تر شود، هر نخ مسئول پردازش بسته‌های بیش‌تری خواهد شد (خط ۳).



(شکل-۵): معماری فیلتر مبتنی بر لیست سیاه بر روی واحد پردازش گرافیکی

(Figure-5): The architecture of the blacklist on GPU

عملیات اصلی فیلتر بسته‌ها در خط چهارم انجام می‌شود؛ به‌طوری‌که هر بسته توسط یک نخ جداگانه پردازش می‌شود؛ بنابراین باید اندیس این بسته‌ها با توجه به موقعیت بلوک در گرید و موقعیت نخ در بلوک محاسبه شود. این اندیس توسط متغیر ***tid*** در خط چهارم محاسبه می‌شود. درصورتی‌که اندیس به‌دست‌آمده، از تعداد کل بسته‌ها کمتر باشد، نخ یادشده یک بسته را از حافظه سراسری برداشته و نشانی IP آن را در فهرست سیاه B موجود در حافظه سراسری، جستجو می‌کند (۷) در صورت وجود عملیات اصلی تطبیق قوانین بر اساس مجموعه قوانین snort ***R***، که در حافظه سراسری است، انجام می‌شود (خط ۹).

تطبیق قوانین بدین‌صورت است که در ابتدا نشانی IP مبدأ و مقصد، شماره درگاه مبدأ و مقصد و پروتکل بسته با قسمت سرآیند قوانین تطبیق داده می‌شود. درصورت تطبیق، محتوای بسته با رشته موجود در قوانین مربوطه توسط الگوریتم تطبیق الگو BM تطبیق داده خواهند شد؛ همچنین سایر قسمت‌های اختیاری قوانین به‌صورت دقیق بررسی و درصورت تطبیق، بیت مربوط به آن قانون نگاشت می‌شود. درنتیجه مجموعه قوانین تطبیق

[DOI: 10.52547/jsdp.18.1.150]

داده‌شده متناظر با بسته به‌عنوان خروجی، در آرایه outputArray جهت صدور هشدار مربوط به هر قانون ذخیره می‌شود (خط ۱۱).

درصورت عدم تطبیق بسته با هیچ‌یک از قوانین موجود در جدول جستجو به شبکه ارسال می‌شود (خط ۱۳). همچنین هنگامی‌که نشانی IP بسته در فهرست سیاه موجود نباشد آن بسته جهت تجزیه‌وتحلیل به سامانه تشخیص نفوذ شبکه ارسال (خط ۱۶) و تمامی بسته‌ها

به‌صورت موازی توسط یک نخ جداگانه پردازش می‌شود.

در پیاده‌سازی جدول جستجو فیلتر بسته مبتنی بر فهرست سیاه از الگوریتم BM به‌دلیل مصرف حافظه کم و دقت بسیار بالا جهت تطبیق الگو استفاده می‌شود و همان‌طور که در بخش ۲-۱ بیان شد، این الگوریتم از دو قانون نویسه بد و پسوند خوب جهت تطبیق الگو سریع استفاده می‌کند. اگرچه الگوریتم‌های تطبیق الگو چندگانه سرعت بالاتری در تطبیق الگو نسبت به تطبیق الگو منفرد BM دارند، ولی مصرف حافظه بسیار بالای آن‌ها یکی از چالش‌هایی هست که با آن روبه‌رو هستند و همچنین دقت عملکرد آن‌ها نسبت BM پایین‌تر است [3, 20, 38, 39]. در پیاده‌سازی فیلتر بسته مبتنی بر فهرست سیاه ازآن‌جایی‌که تنها برخی از بسته‌ها که در فهرست سیاه هستند، جهت تطبیق در جدول جستجو بررسی می‌شوند؛ درنتیجه در پیاده‌سازی الگوریتم فیلتر بسته به‌دلیل تطبیق سرآیند و محتوا بسته‌ها، الگوریتم BM ضمن مصرف بهینه حافظه و دقت بالا می‌تواند از کارایی بالایی برخوردار است.

افزایش سرعت فیلتر بسته‌های غیرضروری ضمن کاهش بار محاسباتی سامانه تشخیص نفوذ شبکه مبتنی بر امضا، باعث تسریع و بهبود عملکرد سامانه تشخیص نفوذ شبکه نیز می‌شود.

```
Input : snort rules R , Set of packets  P , Blacklist  B
Output: set of rules that matched In each packet
Pre − processing://performed within host (CPU)
1:  Construct Bblacklist from corresponding DARPA intrusion detection
2:  global memory ← host memory(R, P, B)
3:  begin
        // Declare n threads; one for each packet
4:      for all i∈ [0, |p| / (|Blocks| × 1024)] do
5:          tid ← threadIdx + i × (|Blocks| × 1024) + blockIdx ∗ 1024 ,
6:          bl ← Null, mrules ← Null ,
7:          if tid < |P| then
8:              packet ← ReadPacket(tid)
9:              bl ← Checkblacklist(B, packet)
10:             if bl == true then
11:                 mrules ← Matching(packet, R)
12:                 if mrules ≠ Null then
13:                     outputArray(tid, mrules)
14:                 else
15:                     send packet to network
16:                 end if
17:             else
18:                 send packet to NIDS
19:             end if
20:         end if
21:         i ← i + 1
22:     end for
23: end
```

(شکل-۶): شبه کد پیاده‌سازی فیلتر در واحد پردازش گرافیکی
(Figure-6): The pseudo-code of filter in GPU

۵- نتایج تجربی

در این پژوهش از پردازنده گرافیکی و سیستمی با مشخصات مندرج در جدول (۱) استفاده کردیم. همچنین، در محیط برنامه‌نویسی visual c++ نسخه ۲۰۱۲ از کتابخانه cuda نسخه ۶/۵ و از کتابخانه winpcap استفاده کردیم. در ادامه ابتدا پیچیدگی زمانی و حافظه‌ای روش پیشنهادی را بررسی می‌کنیم. سپس، کارایی موازی‌سازی فیلتر بسته مبتنی بر فهرست سیاه را بر روی واحد پردازش گرافیکی با مجموعه‌داده DARPA [۴۰] از جنبه‌های مختلف کارایی نظیر دقت عملکرد، زمان پردازش بسته، گذرداد و تسریع مورد ارزیابی قرار می‌دهیم.

[DOI: 10.52547/jsdp.18.1.150]

(جدول-۱): مشخصات سیستم

(Table-1): The system specifications

CPU			Intel(R) Core™ i7Q 2670 @ 2.20GHz	
RAM			6 GB	
Operating System			Windows 8 Ultimate, 64-bit (Service Pack 1)	
GPU	**Model**		Nvidia GT 525M	
	Architecture		Fermi	
	Cuda Cores		96	
	Graphics clock		600 MHz	
	Memory Clock (effective)		1.8 GHz	
	Processing Power (GFLOPS)		215.04	
	Memory bandwidth		28.8 GB/s	
	SMs		2	
	Bus width (bit)		128	

۱-۵- صحت و دقت عملکرد فیلتر بسته

در این بخش دقت عملکرد فیلتر بسته فهرست سیاه بر روی مجموعه‌داده Darpa موردبررسی قرار می‌گیرد. در آزمایش‌های مرتبط، ابتدا فهرست سیاه را به‌صورت برون‌خط و بر اساس تجزیه‌وتحلیل نشانی IP های مبدأ مخرب در ارزیابی سامانه تشخیص نفوذ Darpa در هفته چهارم از ماه آوریل (به‌دلیل وجود حجم حملات بیش‌تر در ساعات آغازین) تولید کردیم. مجموعه حملاتی که دقت عملکرد فیلتر مبتنی بر فهرس سیاه بر مبنای آن بررسی شده در جدول (۲) ارائه شده است. در فرآیند تطبیق در جدول جستجوی فیلتر بسته مبتنی بر فهرست سیاه از مجموعه‌قوانین snort استفاده‌شده است. همچنین، نحوه تشخیص نفوذ با نشانی‌های مبدأ موجود در فهرست بر اساس روش snort است.

به‌منظور اجرای موازی الگوریتم توسط نخ‌های همروند، بسته‌های ورودی شبکه همراه با مجموعه قوانین snort و فهرست سیاه به حافظه سراسری واحد پردازش گرافیکی انتقال می‌یابد؛ سپس، فرآیند جستجوی آدرس IP مبدأ در فهرست سیاه و تطبیق قوانین موجود در جدول جستجو جهت بررسی و تشخیص نفوذ snort آزمایش می‌شود. یادآوری این نکته لازم است که علاوه‌بر قوانین snort، کل سرآیند و محتوای بسته‌های ورودی باید به حافظه سراسری پردازنده گرافیکی انتقال یابند. همچنین، فضای حافظه موردنیاز الگوریتم‌های تطبیق الگو چندگانه بسیار زیاد است [21, 22, 28, 29, 41-45]. برای کاستن تعداد بسته‌های ورودی موردبررسی به‌کمک فهرست سیاه، از الگوریتم تطبیق سریع الگو به روش Boyer-Moore در سامانه تشخیص نفوذ snort می‌کنیم. روش پیشنهادی از کارایی بالایی برخوردار است.

(جدول-۲): مشخصات حملات بررسی‌شده

(Table-2): Specifications of the analyzed intrusions

حملات	توضیح
Ntinfoscan	یک اسکنر مبتنی بر netbios است که جهت به‌دست‌آوردن اطلاعاتی مشترک، نام کاربران، خدمات در حال اجرا و سایر اطلاعات قربانی‌های شبکه را اسکن می‌کند.
Ipsweep	به‌عنوان یک پیگیری جهت تعیین میزبان‌های در حال شنود در شبکه بکار گرفته می‌شود. این اطلاعات برای یک مهاجم به‌منظور جستجوی دستگاه‌های آسیب‌پذیر مفید است.
Netbus	یک حمله راه دور است که مهاجم از یک برنامه تروجان جهت نصب و اجرا سرور netbus روی ماشین قربانی، استفاده می‌کند. سپس مهاجم قادر به دسترسی از راه دور توسط کلاینت netbus خواهد بود.
Sshtrojan	مهاجم سرپرست سیستم را جهت نصب یک نسخه تروجان از برنامه ssh فریب داده و سپس می‌تواند از طریق ssh به سیستم قربانی ورود کند.

به‌منظور اجرای موازی الگوریتم توسط نخ‌های همروند، بسته‌های ورودی شبکه همراه با مجموعه قوانین snort و فهرست سیاه به حافظه سراسری واحد پردازش گرافیکی انتقال می‌یابد؛ سپس، فرآیند جستجوی آدرس IP مبدأ در فهرست سیاه و تطبیق قوانین موجود در جدول جستجو جهت بررسی و تشخیص نفوذ snort آزمایش می‌شود. یادآوری این نکته لازم است که علاوه‌بر قوانین snort، کل سرآیند و محتوای بسته‌های ورودی باید به حافظه سراسری پردازنده گرافیکی انتقال یابند. همچنین، فضای حافظه موردنیاز الگوریتم‌های تطبیق الگو چندگانه بسیار زیاد است [21, 22, 28, 29, 41-45]. برای کاستن تعداد بسته‌های ورودی موردبررسی به‌کمک فهرست سیاه، از الگوریتم تطبیق سریع الگو به روش Boyer-Moore در سامانه تشخیص نفوذ snort می‌کنیم. روش پیشنهادی از کارایی بالایی برخوردار است.

شکل (۷) درصد False Positive، False Negative, True Positive و True Negative را در عملکرد فیلتر مبتنی بر فهرس سیاه جهت تشخیص حملات و نفوذهای رخداده برای مجموعه بسته‌های ورودی با اندازه‌های مختلف نشان نمایش می‌دهد. ازآن‌جاکه فرآیند تطبیق الگو در جدول جستجوی فیلتر بر مبنای قوانین و ساختار سامانه تشخیص نفوذ snort است، تعداد و

[DOI: 10.52547/jsdp.18.1.150]

نوع حملات تشخیصی در مقایسه با حملات تشخیص داده‌شده در ارزیابی پایه سامانه تشخیص نفوذ Darpa مورد ارزیابی قرارگرفته است.

با تعریف صحت، حساسیت و ویژگی با روابط (۱) تا (۳) به‌صورت زیر:

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+FN+FP+TN)} \quad (۱)$$

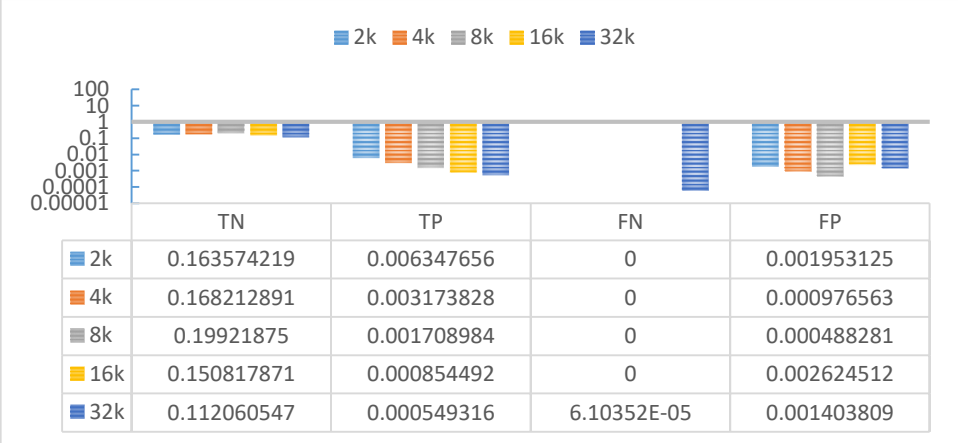
$$Sensitivity = \frac{(TP)}{(FN+P)} \quad (۲)$$

$$\text{Specificity} = \frac{(TN)}{(FP+TN)} \quad (۳)$$

همان‌طور که در شکل (۸) مشاهده می‌شود دقت روش ارائه‌شده در حد روش Thomas و همکاران است [40]. روش پیشنهادی دو حمله را که از حملات netbus و حمله sshtrojan (به‌دلیل رمزنگاری ترافیک) بوده در تعداد ۳۲k بسته ورودی تشخیص نداده ولی سایر حملات در مجموعه بسته‌های ورودی شبکه به‌درستی تشخیص و شناسایی شده است.

با توجه به نتایج به‌دست‌آمده از شکل (۸) به‌ازای اندازه‌های مختلف بسته، عملکرد فیلتر بسته فهرست سیاه به‌نسبه از صحت بالایی برخوردار بوده و بالاترین میزان صحت تشخیص در ۸k بسته به مقدار ۹۹/۷۵ درصد است؛ همچنین، در این شکل نسبت حساسیت و ویژگی عملکرد فیلتر بسته مبتنی بر فهرست سیاه به‌ازای اندازه‌های مختلف بسته بررسی نمایش داده‌شده است. درصد حساسیت که به آن نرخ مثبت درست و یا احتمال تشخیص نیز گفته می‌شود تا ۱۶k بسته صد درصد بوده که این بیان‌گر تشخیص درست تمامی حملات است؛ ولی با توجه به این‌که در ۳۲k بسته دو حمله netbus و sshtrojan تشخیص داده نشده، است این مقدار به نود درصد کاهش پیدا کرده است.

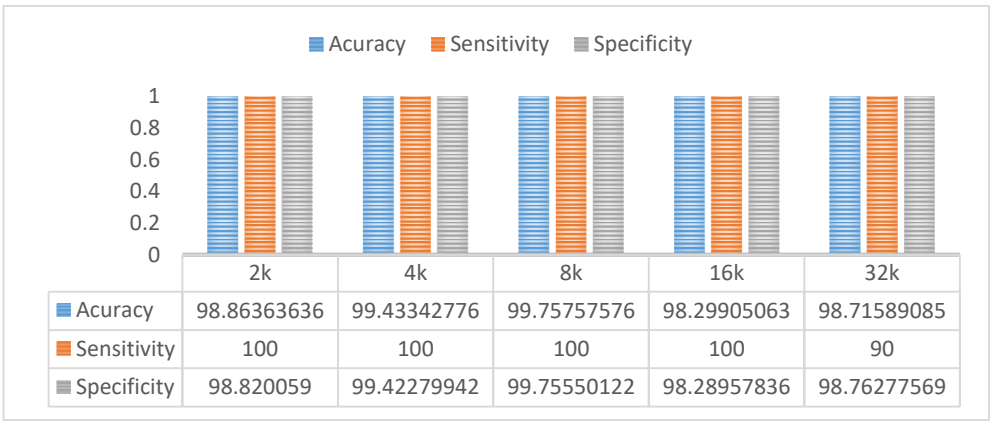
همچنین، در شکل (۸) درصد ویژگی به‌ازای اندازه‌های مختلف بسته مشاهده می‌شود. این درصد بر مبنای نرخ منفی درست و نرخ مثبت اشتباه موردبررسی قرارگرفته و به‌ازای اندازه‌های مختلف بسته‌های ورودی به‌دست‌آمده است. بیشترین درصد ویژگی در مورد ۸k بسته ورودی و مقدار ۹۹/۷۵ درصد است.



	TN	TP	FN	FP
2k	0.163574219	0.006347656	0	0.001953125
4k	0.168212891	0.003173828	0	0.000976563
8k	0.19921875	0.001708984	0	0.000488281
16k	0.150817871	0.000854492	0	0.002624512
32k	0.112060547	0.000549316	6.10352E-05	0.001403809

(شکل-۷): نمودار لگاریتمی نسبت False Negative (FN), False Positive (FP), True Negative (TN) and True Positive (TP)

(Figure-7): The logarithmic rates of False Negative (FN), False Positive (FP), True Negative (TN) and True Positive (TP)



	2k	4k	8k	16k	32k
Acuracy	98.86363636	99.43342776	99.75757576	98.29905063	98.71589085
Sensitivity	100	100	100	100	90
Specificity	98.820059	99.42279942	99.75550122	98.28957836	98.76277569

(شکل-۸): صحت، حساسیت و ویژگی عملکرد فیلتر بسته لیست سیاه

(Figure-8): The accuracy, sensitivity and specificity of blacklist-based packet filter

[DOI: 10.52547/jsdp.18.1.150]

۱-۱-۵- میزان تسریع و گذرداد

در این بخش، کارایی سازوکار پیشنهادی جهت موازی‌سازی روی واحد پردازش گرافیکی از جنبه‌های مختلف کارایی نظیر زمان فیلتر کردن در سامانه مبتنی بر فهرست سیاه، گذرداد و تسریع بررسی می‌شود. زمان فیلتر مبتنی بر فهرست سیاه، مدت‌زمانی است که بسته‌ها به‌وسیلهٔ واحد پردازش گرافیکی در فیلتر مبتنی بر فهرست سیاه بررسی می‌شوند. گذرداد، نشان‌دهنده تعداد بسته‌های ورودی است که در فیلتر مبتنی بر فهرست سیاه در یک ثانیه بررسی می‌شوند. همچنین، نسبت زمان بررسی بسته‌ها در فیلتر مبتنی بر فهرست سیاه در پردازنده گرافیکی به زمان بررسی بسته‌ها در فیلتر مبتنی بر فهرست سیاه در پردازنده مرکزی، تسریع نامیده می‌شود. در جدول (۳)، به‌ازای اندازه‌های مختلف بسته‌های ورودی، زمان فیلتر مبتنی بر فهرست سیاه، گذرداد، تسریع و همچنین زمان انتقال ساختار داده موردنیاز جهت بررسی بسته‌ها در فیلتر مبتنی بر فهرست سیاه بین پردازنده مرکزی و پردازنده گرافیکی اندازه‌گیری شده است. واحد زمان برای محاسبات در این جدول بر اساس میلی‌ثانیه و واحد گذرداد گیگا بیت در ثانیه است.

با توجه به نتایج به‌دست‌آمده که در جدول (۳) نشان داده شده است، میزان تسریع و گذرداد با افزایش اندازه بسته‌های ورودی روند رو به بالایی دارد. طبق نمودار، در پردازش ۳۲k بسته بیشترین تسریع و گذرداد به‌دست آمده است. نتایج حاصل به تعداد بسته در فهرست سیاه و ترافیک شبکه بستگی دارد. بدین‌صورت که افزایش تعداد بسته‌های ورودی که نشانی IP مبدأ آن‌ها در فهرست سیاه وجود داشته باشد؛ موجب به‌کارگیری تعداد بیشتری از پردازش‌های نخی هم‌روند شده و درنتیجه موجب افزایش تسریع می‌شود. در بررسی‌های انجام‌شده بر روی فهرست سیاه به‌دست‌آمده از ترافیک ورودی که نشانی IP مبدأ آن‌ها در فهرست سیاه وجود داشته، مجموعه ۲k بسته دارای بیش‌ترین حملات بوده است. این حملات شامل تعداد زیادی حملات از نوع ntis و ipsweep بوده که به‌عنوان حملات probe جهت شناسایی سامانه‌های آسیب‌پذیر در شبکه استفاده می‌شوند؛ همچنین به‌ازای k2، ۸k و ۱۶k بسته به‌دلیل تعداد زیاد بسته‌های موجود در فهرست سیاه و همچنین ترافیک وب، پردازش بیش‌تری صورت گرفته است.

برای مقایسه نتایج روش پیشنهادی با کارهای اخیر، از معیار درصد بهره‌وری منابع محاسباتی می‌توان استفاده کرد. این معیار با نماد E طبق رابطه (۴) محاسبه می‌شود:

$$E = \frac{Speedup}{number\ of\ cores} \times 100 \qquad (۴)$$

جدول (۴) بهره‌وری روش پیشنهادی را با بهترین بهره‌وری حاصل از موازی‌سازی الگوریتم‌های تطبیق الگو مقایسه می‌کند. همان‌گونه که از نتایج پیداست روش پیشنهادی تسریعی بیش از تسریع روش [5] و کمتر از تسریع روش‌های [3] و[4] دارد. اما محاسبه بهره‌وری نشان می‌دهد که هر سه روش بهره‌وری کمتری نسبت به روش پیشنهادی داشته‌اند و در استفاده بهینه از هسته‌های پردازنده گرافیکی جهت اجرای موازی الگوریتم تطبیق الگو ناموفق بوده‌اند. روش پیشنهادی با ۳۵/۷ درصد بهره‌وری، بهترین روش در موازی‌سازی کارآمد الگوریتم تطبیق رشته و تشخیص نفوذ روی پردازنده گرافیکی بوده است.

(جدول-۳): گذرداد و تسریع سیستم فیلتر بسته‌ها بر اساس فهرست سیاه

(Table-3): The throughput and speedup of the blacklist-based packet filter

اندازه بسته	زمان فیلتر مبتنی بر فهرست سیاه	زمان انتقال	زمان اجرا	تسریع	گذرداد
2K	0.16	0.021	0.18	26.13	12.5
4K	0.20	0.036	0.24	27	20
8K	0.38	0.069	0.45	30.65	21.1
16K	0.53	0.13	0.66	33.11	30.19
32K	0.66	0.27	0.93	34.23	48.48

[DOI: 10.52547/jsdp.18.1.150]

(جدول-۴): بهره‌وری روش پیشنهادی در مقایسه با روش‌های اخیر

(Table-4): The efficiency of the proposed method compared with recent methods

	Hung و همکاران [5]	Hung و همکاران [4]	Ramesh و همکاران [3]	روش پیشنهادی
نوع پردازنده گرافیکی	GeForce GTS 450	GeForce GTX 980	GeForce GTX 660	Geforce GT 525M
بیشینه تسریع	13	345.6	189.14	34.23
تعداد هسته‌های محاسباتی	192	2048	960	96
درصد بهره‌وری	6.8	16.9	19.7	35.7

۶- نتیجه‌گیری و کارهای آینده

با افزایش روزافزون حملات و نفوذهای شبکه که منجر به افزایش تعداد امضاهای مرجوعه قوانین موجود در ابزارهای تشخیص نفوذ شده و همچنین با توسعه سریع فن‌آوری‌های تولید سامانه‌های شبکه‌ای سریع، اهمیت سامانه‌های تشخیص نفوذ شبکه افزایش یافته و درنتیجه، افزایش سرعت پردازش بسته‌های شبکه در سامانه‌های تشخیص نفوذ، از مهم‌ترین چالش‌های مطرح در این حوزه است.

واحد پردازش گرافیکی به‌دلیل داشتن تعداد قابل‌توجهی هسته پردازش موازی در کنار سلسله‌مراتب حافظه با ویژگی‌های متفاوت ازلحاظ پهنای باند و تأخیر دسترسی، در کنار قیمت بسیار کمتر نسبت به سیستم‌های چندپردازنده‌ای متداول، به‌عنوان بستر مناسبی برای موازی‌سازی بخش‌های مختلف سامانه تشخیص نفوذ شبکه جهت تسریع عملکرد آن موردتوجه قرارگرفته است. روش ارائه‌شده در این مقاله با موازی‌کردن روش تشخیص نفوذ بر اساس فهرست سیاه بر روی پردازنده گرافیکی، بهبود قابل توجهی در عملکرد سامانه تشخیص نفوذ شبکه ایجاد می‌کند. نتایج ارزیابی نشان می‌دهد که روش موازی ارائه‌شده علاوه‌بر دست‌یابی به دقت صد درصد در تشخیص نفوذ بسته‌های موجود در فهرست سیاه، به ۳۵/۷ درصد بهره‌وری در استفاده از منابع پردازنده گرافیکی جهت اجرای موازی الگوریتم تشخیص نفوذ می‌رسد که از بهره‌وری بهترین روش‌های موجود جهت تشخیص نفوذ موازی بر روی پردازنده گرافیکی بالاتر است.

درهمین‌اواخر خوشه پردازنده‌های گرافیکی، به بستری نوین برای محاسبات توزیع‌شده با موازات چشم‌گیر تبدیل شده است. در ادامه پژوهش، برای افزایش موازات در اجرای الگوریتم فیلتر بسته‌های شبکه می‌توان از خوشه پردازنده‌های گرافیکی استفاده و در صورت استفاده از چندین پردازنده گرافیکی در قالب خوشه پردازنده گرافیکی، از تعداد SMهای بیشتری برای فیلتر سریع‌تر بسته‌ها می‌توان استفاده کرد.

۷- مراجع 7- References

[1] R. Chi, "Intrusion detection system based on snort," Lecture Notes in Electrical Engineering, vol. 272, pp. 657-664, 2014.

[2] Y. Meng and L.-F. Kwok, "Adaptive blacklist-based packet filter with a statistic-based approach in network intrusion detection," J. Netw. Comput. Appl., vol. 39, pp. 83-92, 2014.

[3] T. Ho, S. Cho, and S. Oh, "Parallel multiple pattern matching schemes based on cuckoo filter for deep packet inspection on graphics processing units," IET Information Security, vol. 12, pp. 381-38.۲۰۱۸ ,۸

[4] C. Hung, P. Wu, H. Wang, and C. Lin, "Efficient Parallel Muti-pattern Matching Using GPGPU Acceleration for Packet Filtering," in 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, 2015, pp. 1843-1847.

[5] C.-L. Hung, C.-Y. Lin, and H.-H. Wang, "An efficient parallel-network packet pattern-matching approach using GPUs," Journal of Systems Architecture, vol. 60, pp. 431-439, 2014/05/01/ 2014.

[6] K. RahimiZadeh, M. Torkamani, and A. Dehghani, "Mapping of McGraw Cycle to RUP Methodology for Secure Software Developing," Signal and Data Processing, vol. 17, pp .33-46 , 2020

[۶] کیوان رحیمی‌زاده، محمدعلی ترکمانی، عباس دهقانی، "نگاشت چرخه McGraw به متدولوژی RUP برای توسعه نرم‌افزار امن"، پردازش علائم و داده‌ها، ۱۳۹۹ ، دوره ۱۷، شماره ۲، صفحات ۳۳-۴۶.

[6] Rahimizadeh K, Torkamani M, Dehghani A. Mapping of McGraw Cycle to RUP

[DOI: 10.52547/jsdp.18.1.150]

Methodology for Secure Software Developing. JSDP. 2021, 17 (2): 33-46.

[7] K. Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems," Electrical Engineering and Computer Science, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, MIT Lincoln, 1999.

[8] P. Innella and O. McMillan, An Introduction to Intrusion Detection Systems, 2001.

[9] P. Bunel, "An introduction to Intrusion Detection Systems," in Sans Security Essentials v1.4c, ed LONDON, 2004.

[10] P. Bunel, "Host- vs. Network-Based Intrusion Detection Systems," in Sans Security Essentials, ed, 2004.

[11] X. Luo, "Model design artificial intelligence and research of adaptive network intrusion detection and defense system using fuzzy logic," Journal of Intelligent & Fuzzy Systems, pp. 1-9.

[12] A. S. Almogren, "Intrusion detection in Edge-of-Things computing," Journal of Parallel and Distributed Computing, vol. 137, pp. 259-265, 2020.

[13] F. Erlacher and F. Dressler, "On high-speed flow-based intrusion detection using snort-compatible signatures," IEEE Transactions on Dependable and Secure Computing, 2020.

[14] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," Procedia Computer Science, vol. 167, pp. 636-645, 2020.

[15] B. Caswell, J. Beale, and A. R Baker, Snort IDS and IPS Toolkit. Syngress Publishing, Inc. Elsevier, Inc.: Williams, Andrew 2007.

[16] R. Chi, Intrusion Detection System Based on Snort. China, 2014.

[17] M. Roesch, C. Green, and S. Team, SNORT Users Manual 2.9.9, 2016.

[18] S. Sharma and M. Dixit, "A Review on Network Intrusion Detection System Using Open Source Snort," vol. 9, pp. 61-70, 2016.

[19] C.-H. Lin, "Accelerating String Matching Algorithms on Multicore Processors".

[20] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," Commun. ACM, vol. 20, pp. 762-772.۱۹۷۷ ,

[21] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," Commun. ACM, vol. 18, pp. 333-340, 1975.

[22] C. L. Hung, C. Y. Lin, H. h. Wang, and C. Y. Chang, "Efficient Packet Pattern Matching for Gigabit Network Intrusion Detection Using GPUs," in High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on, 2012, pp. 1612-1617.

[23] C. S. Kouzinopoulos and K. G. Margaritis, "String Matching on a Multicore GPU Using CUDA," in Informatics, 2009. PCI '09. 13th Panhellenic Conference on, 2009, pp. 14-18.

[24] h. sadeghi and A. Akhavan Bitaghsir, "Signal Detection Based on GPU-Assisted Parallel Processing for Infrastructure-based Acoustical Sensor Networks," Signal and Data Processing, vol. 14, pp. 19-30, 2018.

[۲۴] حامد صادقی، امیر اخوان بی تقصیر، "آشکارسازی سیگنال بر اساس پردازش موازی مبتنی بر جی‌پی‌یو در شبکه‌های حس‌گری صوتی دارای زیرساخت"، پردازش علائم و داده ها، ۱۳۹۶ ، دوره ۱۴، شماره ۴، صفحات ۱۹-۳۰.

[24] Sadeghi H, Akhavan A. Signal Detection Based on GPU-Assisted Parallel Processing for Infrastructure-based Acoustical Sensor Networks. JSDP. 2017, 14 (4): 19-30.

[25] G. Vasiliadis, M. Polychronakis, and S. Ioannidis, "MIDeA: a multi-parallel intrusion detection architecture," presented at the Proceedings of the 18th ACM conference on Computer and communications security, Chicago, Illinois, USA, 2011.

[26] S. Soroushnia, M. Daneshtalab, T. Pahikkala, and J. Plosila, "Parallel Implementation of Fuzzified Pattern Matching Algorithm on GPU," in 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2015, pp. 341-344.

[27] N. P. Tran, M. Lee, S. Hong, and J. Choi, "High Throughput Parallel Implementation of Aho-Corasick Algorithm on a GPU," in Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International, 2013, pp. 1807-1816.

[DOI: 10.52547/jsdp.18.1.150]

[28] G. Vasiliadis, M. Polychronakis, and S. Ioannidis, "Parallelization and characterization of pattern matching using GPUs," in Workload Characterization (IISWC), 2011 IEEE International Symposium on, 2011, pp. 216-225.

[29] C. H. Lin, C. H. Liu, L. S. Chien, and S. C. Chang, "Accelerating Pattern Matching Using a Novel Parallel Algorithm on GPUs," IEEE Transactions on Computers, vol. 62, pp. 1916-06 ,20313.

[30] L. Vokorokos, M. Ennert, M. >Čajkovský, and J. Radušovský, "A Survey of parallel intrusion detection on graphical processors," Central European Journal of Computer Science, vol. 4, pp. 222-230, 2014.

[31] A. P. M.S., "Parallelizing a network intrusion detection system using a GPU," Master of Science, Computer Science and Engineering, Louisville, UK, 2012.

[32] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis, "Gnort: High Performance Network Intrusion Detection Using Graphics Processors," in Recent Advances in Intrusion Detection: 11th International Symposium, RAID 2008, Cambridge, MA, USA, September 15-17, 2008. Proceedings, R. Lippmann, E. Kirda, and A. Trachtenberg, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 116-134.

[33] G. Vasiliadis, M. Polychronakis, S. Antonatos, E. P. Markatos, and S. Ioannidis, "Regular Expression Matching on Graphics Hardware for Intrusion Detection," in Recent Advances in Intrusion Detection: 12th International Symposium, RAID 2009, Saint-Malo, France, September 23-25, 2009. Proceedings, E. Kirda, S. Jha, and D. Balzarotti, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 265-283.

[34] M. A. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S .Lee, et al., "Kargus: a highly-scalable software-based intrusion detection system," presented at the Proceedings of the 2012 ACM conference on Computer and communications security, Raleigh, North Carolina, USA, 2012.

[35] H. Song and J. W. Lockwood, "Efficient packet classification for network intrusion detection using FPGA," presented at the Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays, Monterey, California, USA, 2005.

[36] Y. Meng and L. f. Kwok, "Adaptive context-aware packet filter scheme using statistic-based blacklist generation in network intrusion detection," in Information Assurance and Security (IAS), 2011 7th International Conference on, 2011, pp. 74-79.

[37] M. Ramesh and H. Jeon, "Parallelizing Deep Packet Inspection on GPU," in 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), 2018, pp. 248-253.

[38] S. Hakak, A. Kamsin, P. Shivakumara, G. A. Gilkar, W. Z. Khan, and M. Imran, "Exact String Matching Algorithms: Survey, Issues, and Future Research Directions," IEEE Access, 2019.

[39] C.-L. Hung, T.-H. Hsu, H.-H. Wang, and C.-Y. Lin, A GPU-based Bit-Parallel Multiple Pattern Matching Algorithm, 2018.

[40] C. Thomas, V. Sharma, and N. Balakrishnan, Usefulness of DARPA dataset for intrusion detection system evaluation, 2008.

[41] P. P, M. T, and L. Raj, A Comparative Study on String Matching Algorithm of Biological Sequences, 2014.

[42] N. Tuck, T. Sherwood, B. Calder, and G. Varghese, Deterministic Memory-Efficient String Matching Algorithms for Intrusion Detection vol. 4, 2004.

[43] "Exact Matching: Classical Comparison-Based Methods," in Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, D. Gusfield, Ed., ed Cambridge: Cambridge University Press, 1997, pp. 16-34.

[44] C. H. Lin, "Accelerating String Matching Algorithms on Multicore Processors," vol. 2, pp. 52-59, 6 June- 2016 2016.

[45] J. Yu, Y. Xue, and J. Li, "Memory efficient string matching algorithm for network intrusion management system," Tsinghua Science and Technology, vol. 12, pp. 585-593, 2007.

مهدی عباسی مدرک دکترای خود را در سال ۱۳۹۱در رشته معماری سیستم‌های رایانه‌ای از دانشگاه اصفهان اخذ و در حال حاضر دانشیار گروه مهندسی رایانه دانشگاه بوعلی سینا است. پژوهش‌های ایشان به‌طور عمده برروی اینترنت اشیا،

[DOI: 10.52547/jsdp.18.1.150]

پردازنده‌های شبکه‌ای، بهینه‌سازی و پردازش سیگنال متمرکز است.

نشانی رایانامه ایشان عبارت است از:

abbasi@basu.ac.ir

مطهره افشاری حقدوست مدرک کارشناسی ارشد خود را در رشته فناوری اطلاعات-شبکه‌های رایانه‌ای از دانشگاه بوعلی سینا در سال ۱۳۹۶ اخذ کرده است. پژوهش‌های ایشان به‌طور عمده برروی حوزه‌های زیر متمرکز است:

- طراحی شبکه‌های رایانه‌ای
- امنیت شبکه‌های رایانه‌ای

نشانی رایانامه ایشان عبارت است از:

a.afshari1371@gmail.com

[DOI: 10.52547/jsdp.18.1.150]