Surgical DDoS Filtering With Fast LPM

Jan 2022

IEE Access

No citations

تشخیص (فیلتر) حملات منع‌خدمت توزیع‌شده با استفاده از طولانی‌ترین تطبیق

میخواهیم با استفاده از سخت‌افزارهای مرسوم، مسیرداده‌ای[1] ارایه دهیم که امکان جستجو[2] روی پایگاه‌داده‌های LPM[3] مختلف بزرگ برای هر بسته ۶۴ بایتی با حفظ نرخ‌گذر ۱۰ گیگابیت‌برثانیه و به کمک تنها یک پردازنده را داشته باشیم. روش ما به دلیل معماری غیرقفل[4]، قابلیت مقیاس‌پذیری بالا و همچنین استفاده کم ساختارهای LPM از حافظه (سربار حافظه کم) را دارا می‌باشد.

روشی که جدیداً مرسوم شده است، پیاده‌سازی آنالیز و فیلتر بسته‌ها با سرعت زیاد به صورت نرم‌افزاری هست. اما روش‌های نرم‌افزاری سربار حافظه و تاخیر زیادی دارند، به همین دلیل مکانیزم‌هایی مانند DPDK یا NetMap ارایه شده است که مستقیماً بسته‌ها را برای پردازش به فضای کاربر می‌برند. حتی مکانیزم‌های مانند XDP/eBPF نیز مطرح شده‌است که کدهای آنالیزگر را به صورت در لحظه[5] به کد ماشین تبدیل می‌کنند که منجر به سربار کمت CPU شود و مناسب معماری پردازنده‌های امروزی نیز باشد.

راهکارهایی مانند XDP در سمت نهایی (کاربر موردحمله) پیاده‌سازی می‌شوند.

یک جستجوگر LPM مدرن برای برنامه‌های فیلتر بسته‌ها می‌سازیم که امکان اجرای چندین lpm query با استفاده از کلیدها (آدرسهای آی‌پی) تصادفی برای هر بسته با حفظ سرعت بالا را داشته‌باشد.از رویکرد محبوب فعلی که پیاده‌سازی تابع‌های فیلتر بسته به صورت در لحظه در کرنل با سرعت بالا باشد، استفاده نمی‌کنیم. و نشان می‌دهیم فیلترسازی در سطح کاربر سریعتر صورت می‌گیرد. روشی که ارایه می‌دهیم سازگار با جهان واقعی و شامل یک زبان توصیف فیلتر کامل به همراه یک پارسر سریع و کامپایلر و همچنین رابط کنترل عملکردی با قابلیت پیکربرندی مجدد و جمع‌آوری آمار درلحظه، می‌باشد.

---

[1] Datapath
[2] query
[3] Longest Prefix Matching
[4] Lockless Architecture
[5] Just In Time

طرحی که ارایه می‌دهیم باید به صورتی انعطاف‌پذیر باشد که برخی prefix IP های خاص را پشتیبانی کند و همچنین آدرس دهی زیرشبکه[6] کاملتری داشته باشد تا بتواند الگوهای جدید آی‌پی بسیاری را به سرعت تشخیص دهد.

طرح‌های LPM فعلی:

- POPTrie
- SAIL
- DXR : قدیمی تر از آندو

دو روش اول متداول‌ترین‌ها از نظر میزان استفاده هستند. از DXR به عنوان طرح اصلی LPM برای ساخت مسیرداده فیلترکن خودمان استفاده می‌کنیم، که البته بهبودهایی نیز به آن اضافه کرده‌ایم که در ادامه معرفی می‌شود. دلیل انتخاب ما نرخ گذر بالا و میزان مصرف کم حافظه است. البته در هنگام ارزیابی از دو روش ساختاری دیگر نیز استفاده می‌کنیم.

روش کار DXR :

استفاده از K بیت باارزش[7] برای جستجو[8] در جدول اصلی[9] که ممکن است منجر به برخورد شود(ایندکس جدول بعدی) یا درغیر اینصورت یک محدوده[10] را با استفاده از بیتهای باقی‌مانده جستجو می‌کند. اینکه سرعت جستجو مهم تر است یا میزان استفاده از حافظه، بستگی به انتخاب اندازه جدول با 2^k عنصر دارد. همانطور که در شکل دیده میشود، عملکرد آن را بهبود داده ایم:جستجو یا در مرحله eXtension تمام می شود یا با استفاده از R بیت باقی‌مانده، جستجوی دودویی در جدول آخر انجام می‌دهیم.

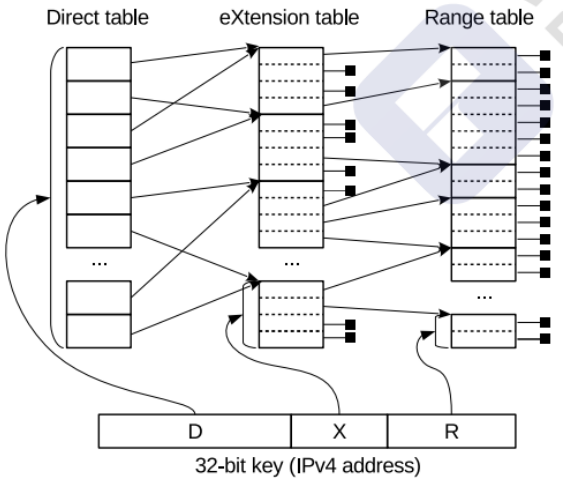

FIGURE 1. Enhanced DXR structures and lookup process.

[6] subnet
[7] Most significant
[8] indexing
[9] Primary table
[10] range

زمانی که برچسبهای[11] خروجی زیادی داریم روش ما به خوبی عمل می کند.

یکی از روشهای جلوگیری از حملات منع خدمت BGP blackholding است. قربانی با ارسال پیغام به ارائه‌دهنده (های) بالادست خود[12] برای جلوگیری از آسیب به بقیه زیرساخت‌ها در زیر بار ترافیک بیش از حد، از مسیریابی تمام ترافیک به میزبان (های) هدف خودداری می‌کند. لیکن قصد داریم به‌جای استفاده از این روش که آدرس قربانی ها را فیلتر میکند، شرایطی فراهم کنیم که ارایه‌دهندگان بتوانند به صورت دقیق نقشه میزبانهای‌آلوده و مشکوک مهاجم که حمله از طرف آنها احتمالن صورت می‌گیرد، را به دست آورند.

مسیرداده[13] پیشنهادی ما: تحت شرایط شدید، مانند مدیریت حملات منع خدمت حجیم، وظیفه یک مسیر داده فیلتر، انتقال بسته ها از یک رابط به رابط دیگر پس از طبقه‌بندی آنها و اعمال اقدامات مناسب در سریع‌ترین زمان ممکن است، در حالی که هنوز آمار عملیاتی اولیه را ارائه می دهد. همچنین ممکن است مقادیر قابل کنترل از نمونه ها را به یک پردازشگر بسته جداگانه برای تجزیه و تحلیل دقیقتر هدایت کند. یک ابزار خارجی، مانند سیستم تشخیص حملات منع‌خدمت، ممکن است از بسته‌های نمونه جمع‌آوری‌شده و آمار ترافیک برای تولید قوانین فیلترینگ در صورت حمله استفاده کند (شکل زیر) . یک دستگاه فیلتر لازم نیست حتمن همانند یک مسیریاب آی‌پی عمل کند. در واقع، اکثر فایروال‌های قدیمی این امکان را دارند که حضور خود را با به‌روزرسانی نکردن فیلد زمان برای زنده بودن (TTL) بسته‌های ارسال‌شده پنهان کنند، یا به سادگی در حالت پل‌سازی شفاف استفاده می‌شوند.

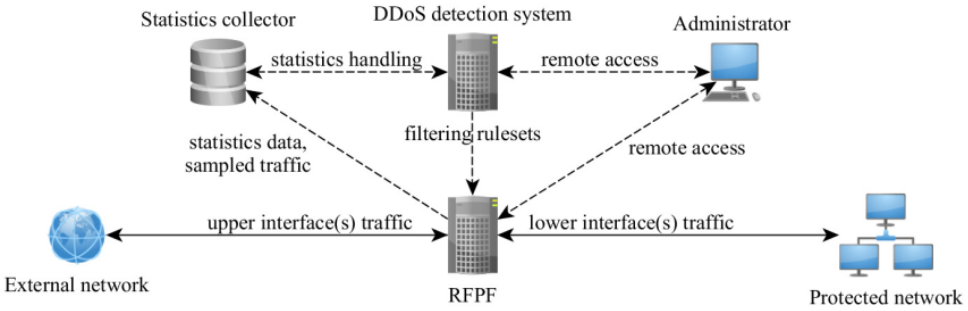

FIGURE 2. A deployment of the filtering system datapath with the working DDoS detection tool.

RFPF: Reduced Feature-Set Packet Filter که به معنای ساده بودن و موثر بودن است. دو مجموعه اینترفیس تعریف می‌کنیم و ترافیکی که از پایین به بالا درجریان هست را upstream و بالعکس را downstream می‌نامیم. هر طرف جریان ترافیک، یک مجموعه قوانین فیلتر مستقل خود را دارد. یک پارسر دو مجموعه قوانین را به دو تابع در زبان C تبدیل می‌کند که در نهایت با استفاده از کامپایلر gcc به یک فایل

[11] label
[12] Upstream Provider
[13] datapath

آبجکت اجرایی تبدیل می‌شود. واحد پردازشی تابع، بسته نیست بلکه صفی کامل از بسته‌هاست، که netmap به اپلیکیشن‌های سطح کاربر در قالب یک بافر ارسال می‌کند.

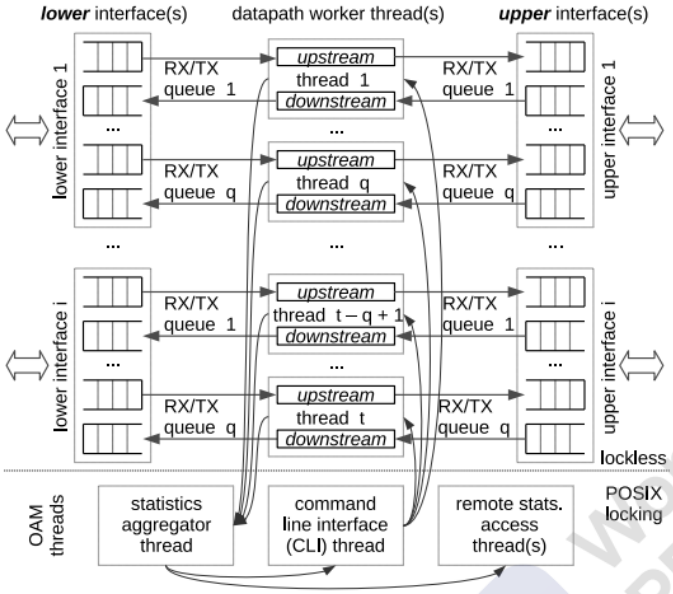

FIGURE 3. **RFPF datapath: interfaces, queues, worker threads.**

ایک بهینه سازی ضمنی مهم برای کامپایل کردن قوانین در کد ماشین، انتشار ثابت هایی[14] مانند شماره پورت L4، محدوده آدرس L3 و غیره در جریان دستورالعمل[15] است که وابستگی های داده[16] و توقف خط لوله CPU را کاهش می‌دهد، که در غیر این صورت بایستی هنگام واکشی داده ها این عملیات رخ دهد. حتی resolve ثابت‌های ظاهراً جزئی در زمان کامپایل، مانند ماسک‌های شاخص بافر دایره‌ای[17]، می‌تواند تأثیر قابل‌توجهی بر توان عملیاتی داشته باشد،با توجه به اینکه که ممکن است تعیین شده باشد که زمان‌بندی پردازش هربسته به از ۱۰۰ تا ۱۵۰ سیکل ساعت تجاوز نکند. این یک محدودیت زمانی متداول برای ارسال بسته با نرخ ۱۰ گیگابیت بر ثانیه روی یک هسته واحد CPU است. تمام اجزای دیگر (پارسرپیکربندی، CLI و غیره) نیز در C پیاده سازی شده و در فضای کاربر اجرا می‌شوند. از آنجایی که هر NIC را می توان به گونه ای پیکربندی کرد که ترافیک ورودی را روی صف‌های دریافتی متعدد (ring) متعادل کند، یک نخ کارگر[18] جداگانه به هر جفت صف دریافت/انتقال بین رابط های "بالا" و "پایین" متناظر اختصاص داده می‌شود. در صورت تمایل، بسته‌ها را می‌توان به یک واسط «منحرف» تغییر مسیر داد یا نمونه‌برداری کرد. دو مجموعه خصوصی از شمارنده های آماری[19] به

[14] constant
[15] Instruction stream
[16] Data dependecy
[17] ring
[18] Workload thread
[19] statics

هر نخ کارگر اختصاص داده شده است. در هر زمان معین، هر نخ کارگر روی یک مجموعه کار می‌کند، در حالی که دیگری توسط یک نخ انباشت‌کننده[۲۰] آمار مجزا قابل دسترسی است، که شمارنده‌های هر صف را در یک فرم یکپارچه قابل ارائه به مدیر سیستم[۲۱] یا ابزارهای خارجی[۲۲] جمع‌آوری می‌کند. برای اینکه همیشه آمارهای به‌روز ارائه شود، نخ انباشت‌کننده بین مجموعه شمارنده‌های «گرم» و سایه دو بار در هر ثانیه سوئیچ می‌کند. از آنجایی که این کار در فرکانس پایین انجام می‌شود، می‌توان آن را با به‌روزرسانی ناهمزمان یک پرچم مشترک سیگنال‌دهنده تغییر به رشته‌های کارگر، با کمترین اختلال در عملکرد فیلترینگ و بدون از دست‌دادن داده ای، انجام داد. رشته‌های کارگر فقط مجاز خواندن داده‌های مشترک، مانند جداول LPM هستند که در طول عمر یک پیکربندی فیلتر تغییر ناپذیر می‌مانند.

هنگامی که پیکربندی فیلتر بسته تغییر می‌کند، مجموعه جدیدی از ساختارهای داده با داشتن ویژگیهای قابل اشتراک و نخ-محلی بودن تخصیص داده می‌شود و رشته های کارگر به طور ناهمزمان به توابع فیلتر جدیداً کامپایل شده که در حالت[۲۳] جدید کار می‌کنند، سوئیچ می‌کنند. این انتقال غیرمسدودکننده است (یعنی هیچ بسته ای از بین نمی رود) اما تضمینی برای اتمیک[۲۴] بودن آن وجود ندارد، زمانی که نخ‌ها حالت را تغییر می دهند حالت قدیمی و جدید به طور همزمان عمل می‌کنند، تا زمانی که آخرین نخ کارگر رفرنس موجود در حالت قدیمی را طبق توافق RCU آزاد کند. هنگامی که همه نخ‌ها گذر را انجام دادند، توابع و داده ساختارهای قدیمی به حالت آماده به کار[۲۵] تبدیل می‌شوند تا به صورت اختیاری درخواست‌های بعدی مجدداً فعال شوند (بازگردانی فوری)، یا زمانی که بهشان نیازی نیست به عنوان زباله[۲۶] جمع‌آوری شوند. این همچنین باعث می‌شود تا چندین پیکربندی فیلتر از قبل آماده شده و در صورت نیاز برای فعال سازی فوری آماده شوند.

بازسازی[۲۷] روتین های فیلترینگ و تمام ساختارهای داده مرتبط از یک فایل پیکربندی نمونه نشان داده شده در مقاله که شامل پارس‌کردن prefix ها با فرمت ASCII، ساخت ساختارهای LPM، کامپایل کد تولید شده به صورت پویا و پیوند آن با برنامه در حال اجرا، است، ۴.۲۵ ثانیه طول می‌کشد. ساخت[۲۰] پیکربندی که شامل لیست سیاه [۲۸]با ۱۱۱۳۳۹۳ آدرس به جای لیست سیاه کوچکتر قبلی باشد، تقریباً سه ثانیه بیشتر زمان می‌برد.

[20] Aggregator thread
[21] Administrator
[22] External tools
[23] state
[24] atomic
[25] Hot-standbu stae
[26] garbage
[27] rebuilding
[28] Black-list

طرح پیشنهادی به هیچگونه همگام‌سازی میان نخ‌های کارگر نیازی ندارد. مشکل این است که عملکردهایی [۲۹]که ذاتاً به دسترسی نوشتن همگام به داده های مشترک نیاز دارند، نمی توانند به راحتی در مسیر داده ما بدون استفاده از مکانیزم قفل اضافی قرار بگیرند.گسترش RFPF با ردیابی دقیق اتصال دو جهته پویا یا ترجمه آدرس شبکه پویا (NAT) مستلزم (هم در upstream و هم downstream) جستجوهای همزمان و اصلاحات متعد در جریان های ذخیره شده در داده های مشترک است. این کار را نمی‌توان بدون مکانیزم قفل انجام داد، که در نتیجه به ناچار جریمه عملکرد سنگینی را به همراه خواهد داشت

برای مقایسه، ما فرانت پارسر خود را گسترش داده‌ایم تا اجازه تولید برنامه‌های eBPF/XDP بر اساس پیکربندی‌های منظم[۳۰] RFPF را بدهد، که در این صورت مسیرداده بسته منحصراً در هسته ایجاد می‌شود. با این حال، چنین مسیرداده‌های XDP تولید شده همچنان به ماژول LPM توکار[۳۱] eBPF، که مبتنی بر LC-trie است، متکی هستند.

ارزیابی: مسیرداده‌مان را در یک محیط اترنت در مجموعه قوانین فیلترینگ و شرایط عملیاتی مختلف تست کرده‌ایم. هدف ما بررسی نحوه میزان تغییر افزایش نرخ گذر زمانی که بار پردازشی روی چندین هسته تقسیم شده می‌باشد ، که در طول آزمایش‌ها با تنظیم تعداد صف‌های سخت‌افزاری که کارتهای شبکه بسته‌های ورودی را به آن‌ها توزیع می‌کنند و با روشن کردن تدریجی منبع بسته ۱۰ گیگابیت بر ثانیه تنظیم کردیم، صورت می‌گیرد.

در ابتدا میزان نرخ‌گذر را با استفاده از مجموعه قوانین فوروارد و دورانداختن بدون شرط اندازه گرفته‌ایم وXDP را با RFPF مقایسه کرده‌ایم. به وضوح روش ما بهتر عمل می‌کند.درتست‌های بعدی نیز از یک مجموعه قوانین فیلتر که شامل سه بررسی LPM هست برای اینکه بسته را دور بیندازد یا قبول کند، استفاده می‌کنیم. تست را در دو مرحله که مرحله اول با ترافیک کاملاً تصادفی و مرحله بعد ۱۰ درصد تصادفی، ۲۰ درصد لیست سفید و ۷۰ درصد از لیست سیاه است، انجام دادیم. تست ها را با لیست های سفید/سیاه بزرگتر و الگوریتم های داده ساختار دیگر (DXR,SAIL,...) نیز بررسی کردیم.

کارهای مربوطه را نخواندم

نتیجه‌گیری: در دیتاسنترها با جلوگیری از مهاجمین با استفاده از لینکهای BGP از گسترش آلودگی به بخش‌های دیگر دیتاسنتر قبل از ورود ترافیک به شبکه با حملات منع خدمت مقابله می‌کنند. اما در این مقاله آلودگی زدایی را با استفاده از دستگاه‌های میانی قبل از اینکه ترافیک به دستگاه‌های انتهایی برسند انجام می‌دهیم.مسیر داده

[29] funciotanility
[30] Regular RFPF config
[31] Built-in

فیلتری برای تسریع هدایت LPM بر روی یک دستگاه با پردازنده ۸ هسته‌ای و نرخ گذر ۶۰ مگابیت بر ثانیه که تمامی بسته‌ها روی چندین LPM پایگاه داده که شامل چندین هزار prefix و آدرس هستند، ارایه می‌کنیم. ارزیابی تجربی ما نشان داد که انتخاب طرح LPM باعث می‌شود عملکرد چنین مسیر داده فیلتری را خراب کند، و اینکه برخی از طرح‌های محبوب LPM ممکن است برای فهرست سیاه برنامه‌هایی با مجموعه داده‌های آدرس بزرگ به دلیل محدودیت‌های ساختاری ذاتی اصلاً مناسب نباشند (حافظه ناکافی برای برچسب زدن hop بعدی یا برای پیشوندهای خاص تر، که منجر به ناتوانی در مدیریت مجموعه داده های بزرگتر می شود). ما نشان داده‌ایم که حتی تغییرات جزئی و ساده در ساختارهای داده‌های LPM و الگوریتم‌های جستجو ممکن است باعث افزایش توان عملیاتی در دنیای واقعی نزدیک به ۱۰ Mpps شود.