EasyChair Preprint

№ 1839

Introducing a New Method for Early Detection of Distributed Denial of Service Attack on Software Defined Networks

Reza Bakhtiari Shohani and Akbar Mostafavi

November 5, 2019

ارائه یک روش جدید برای تشخیص زودهنگام حمله منع سرویس توزیع شده در شبکه‌های نرم‌افزارمحور

رضا بختیاری شوهانی [۱] and سید اکبر مصطفوی [۲]

[۱] کارشناس ارشد، دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، r.bakhtiari92@gmail.com

[۲] استادیار، دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، r.bakhtiari92@gmail.com

چکیده

جداسازی لایه کنترل و لایه داده در شبکه‌های نرم‌افزارمحور، سبب مدیریت و کنترل بهتر شبکه شده است. با این وجود، زمینه حملات منع سرویس جدید را فراهم کرده است. یکی از این حملات منع سرویس، ارسال بسته‌های جعلی مهاجم به چندین مقصد متفاوت در شبکه است. این حمله سبب نرخ بالای خطای جدول [۱] در سوییچ و ارسال بسته‌ها به کنترلر و اشغال شدن آن می‌شود. جذابیت این نوع حمله از آنجا ناشی می‌شود که توسط روش‌های تشخیص حمله مبتنی بر معیار فاصله مانند آنتروپی قابل تشخیص نمی‌باشد. ما در این مقاله یک روش جدید مبتنی بر رگرسیون خطی به منظور تشخیص این نوع حمله را معرفی می‌کنیم. ابتدا نشان می‌دهیم بر اساس رگرسیون خطی می‌توان تقریب قابل قبولی برای پیش‌بینی تعداد خطاهای جدول سوییچ ارائه داد. سپس براساس این تقریب در بازه‌های زمانی ۵۰ ثانیه‌ای به صورت پویا، خط آستانه را برای تعداد خطاهای جدول سوییچ تعریف می‌کنیم. ارزیابی نتایج نشان می‌دهد این حمله که توسط روش‌های مبتنی برآنتروپی قابل تشخیص نیست توسط روش پیشنهادی به خوبی تشخیص داده می‌شود.

کلمات کلیدی

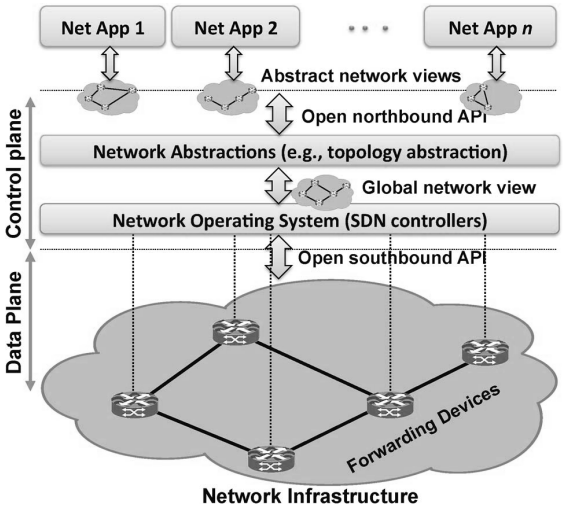
شبکه‌های نرم‌افزارمحور، حمله DDoS، تشخیص حمله DDoS، رگرسیون خطی.

۱ مقدمه

ایده شبکه های نرم افزار [۲] محور که در سال ۲۰۰۷ مطرح شد یک بستر مناسب را برای نوآوری و خلاقیت در دنیای شبکه فراهم کرد. در معماری SDN بخش ارسال داده [۳] و بخش کنترل [۴] از هم جدا شده‌اند. این امر سبب مدیریت و پیکربندی راحت‌تر و مطمئن‌تر در شبکه شده است. در SDN مغز حاکم بر شبکه به صورت منطقی در کنترلر متمرکز شده است. در نتیجه تجهیزات پیچیده شبکه مانند مسیریاب‌ها و سوییچ‌ها که هر کدام دارای سیستم عامل جداگانه و بخش کنترلی مختص به خود بودند، تبدیل به تجهیزات ساده‌ای شده‌اند که فقط دستورات را از کنترلر دریافت و اجرا می‌کنند. همانطور که در شکل ۱ مشخص شده است، معماری SDN از سه لایه اصلی زیر تشکیل شده است [۱]:

۱. لایه برنامه: این لایه مربوط به برنامه‌هایی است که در بالای کنترلر پیاده شده‌اند و از طریق کنترلر به منابع شبکه دسترسی دارند.
۲. لایه کنترل: این لایه مغز حاکم بر شبکه است و به دلیل دید سراسری که بر روی شبکه دارد از توپولوژی شبکه آگاه است و در نتیجه به خوبی می‌تواند برای رفتار شبکه تصمیم گیری کند.
۳. لایه داده: تجهیزات سخت افزاری مانند سوییچ‌ها در لایه داده قرار دارند. این لایه دستورات را از لایه بالاتر دریافت کرده و بر روی بسته‌ها اعمال می‌کند.
این لایه‌ها توسط واسطهای ارتباطی باز که در شکل مشخص شده است با هم دیگر ارتباط برقرار می‌کنند. هر کدام از این لایه‌ها در فصل۲ به صورت جداگانه بررسی می شود.

SDN با جداسازی لایه‌های کنترل و داده مدیریت آسان‌تر شبکه را به ارمغان آورده است. اما این جداسازی مسائل امنیتی جدیدی را ایجاد کرده است. یکی از این مسائل امنیتی، حمله منع‌سرویس توزیع‌شده [۵] است. کنترلر به عنوان مغز حاکم بر شبکه هدف جذابی برای مهاجمین است که می‌توانند با حملات متنوع DDoS کارایی آن را کاهش دهند یا در حملات شدیدتر، آن را از دسترس شبکه خارج کنند در نتیجه تشخیص زود هنگام [۶] این حمله در شبکه‌های نرم‌افزارمحور امری حیاتی است. طی سال‌های اخیر بررسی این نوع حمله در شبکه‌های نرم‌افزارمحور توجه زیادی را به خود جلب کرده است. با توجه به گستردگی تنوع حملات DDoS ، هنوز روش جامعی برای تشخیص و مقابله با آن وجود ندارد. روش‌های مبتنی بر



شکل ۱: معماری شبکه‌های نرم‌افزارمحور [۱]

آنتروپی [۲، ۳، ۴، ۵، ۶، ۷] سعی می‌کنند با مدل‌های آماری مبتنی بر آنتروپی این نوع حمله را به صورت زودهنگام تشخیص دهند. روش‌های مبتنی بر یادگیری ماشین [۸، ۹، ۱۰] نیز قادرند طیف گسترده‌ای از حملات را تشخیص دهند. اما این روش‌ها دارای سربار زمانی و پردازشی زیادی هستند. مهاجمین همیشه سعی کرده‌اند حملات جدیدی را طراحی کنند که توسط روش‌های موجود قابل تشخیص نیستند یا روش‌های موجود کارایی لازم در تشخیص آن‌ها را ندارند. نوآوری‌های این مقاله به صورت زیر است:

- معرفی یک حمله منع سرویس توزیع شده که روش‌های مبتنی بر آنتروپی قادر به تشخیص آن نیستند.
- ارائه یک روش جدید مبتنی بر رگرسیون خطی برای تشخیص حملات منع سرویس توزیع شده.
- تحلیل و ارزیابی روش پیشنهادی و مقایسه با روش‌های مبتنی بر آنتروپی.

ساختار مقاله در ادامه به این صورت است که در بخش دوم کارهای پیشین به همراه معایب آن‌ها آورده شده است. بخش سوم مربوط به توصیف سیستم و توپولوژی و ساختار شبکه مورد آزمایش می‌باشد. روش پیشنهادی و مراحل رسیدن به آن در بخش چهارم مورد بررسی قرار می‌گیرد. ارزیابی روش پیشنهادی و مقایسه آن در بخش پنج آوده شده است. در نهایت نتیجه‌گیری حاصل از این کار پژوهشی در بخش آخر قرار دارد.

۲ مروری بر کارهای پیشین

به طور کلی تحقیقات مربوط به تشخیص حملات DDoS در SDN را می‌توان به سه دسته روش‌های مبتنی بر آنتروپی، روش‌های مبتنی بر یادگیری ماشین و سایر روش‌ها تقسیم کرد.

۲-۱ روش‌های مبتنی بر آنتروپی

روش‌های مبتنی بر آنتروپی میزان تصادفی بودن ترافیک ارسالی را می‌سنجند. در واقع آنتروپی بالا نشان دهنده ارسال بسته به صورت تصادفی در شبکه است. همانطور که در شکل ؟؟ مشخص شده است، در حالت عادی جریان‌ها به صورت تقریبا مساوی بر اساس آدرس مقصد تقسیم‌بندی می‌شوند که نشان دهنده آنتروپی بالاست. اما درحالت حمله با حضور ترافیک‌های ناگهانی، قسمت عمده جریان‌ها به سوی یک مقصد مشخص هدایت می‌شود. این اتفاق منجر به کاهش ناگهانی آنتروپی می‌شود.

به طور کلی در مقاله [۲] برای تشخیص زودهنگام از مفهوم آنتروپی به شکل ساده استفاده شده است. در حالت نرمال شبکه، بسته‌ها به صورت یک توزیع یکنواخت بر اساس آدرس مقصد تقسیم بندی می‌شوند. اما در حالت حمله این توزیع یکنواخت نخواهد بود. این روش قادر نیست حملاتی که به صورت یکنواخت بین تمامی آدرس‌ها ارسال می‌شوند را تشخیص دهد. روش safety [۳] سعی کرده است با استفاده از مفهموم آنتروپی حملات SYN flood را تشخیص دهد. البته علاوه بر ویژگی آدرس مقصد از ویژگی TCP flag نیز بهره برده است.

از دیگر مقالاتی که روش‌های مشابه استفاده کرده‌اند می‌توان به مرجع [۴] و JESS[۵] اشاره کرد. در روش JESS از آنتروپی مشترک[۷] برای تشخیص و کاهش حملات منع سرویس استفاده شده است. در این شیوه از دیگر ویژگی‌های بسته مانند ویژگی‌های لایه TCP نیز برای محاسبه آنتروپی استفاده می‌شود. روش StateSec [۶] نیز از روش آنتروپی با ترکیب ویژگی‌های آدرس مقصد ومبدا به همراه شماره پورت بهره برده است. روش [۷] از مفهوم آنتروپی برای بخش تشخیص حمله منع سرویس بهره برده است.

۲-۲ روش‌های مبتنی بر یادگیری ماشین

روش‌های مبتنی بر یادگیری ماشین به طور کلی شامل استخراج اطلاعات، جمع‌آوری ویژگی‌های بسته‌ها و طبقه‌بندی ترافیک بر اساس یک یا چند الگوریتم مبتنی بر یادیگیری ماشین می‌باشد. این روش‌ها قادر هستند ترافیک ورودی به شبکه را به طور کلی به دو دسته ترافیک بی‌خطر و پرخطر تقسیم بندی کنند. در مقاله [۸] سعی شده است با استفاده از الگوریتم SVM و استخراج ۶ ویژگی از بسته‌ها مانند نرخ تولید بسته‌هایی با آدرس مبدا، مقصد و شماره پورت مشخص، یک روش برای تشخیص حمله منع سرویس ارائه شود

در مقاله [۱۰] یک روش تشخیص حمله منع سرویس مبتنی بر یادگیری عمیق پیشنهاد داده شده است. روش تشخیص در این مقاله داری سه ماژول به شرح زیر می‌باشد:

۱. Traffic Collector and Flow Installer: این ماژول تمامی بسته‌های ورودی را جمع‌آوری و بر اساس نوع پروتکل TCP, UDP, ICMP استخراج می‌کند.
۲. Feature Extractor: این ماژول ویژگی‌هایی مانند نسبت تعداد جریان‌های TCP, UDP, ICMP به کل جریان‌های ورودی یا خروجی، نسبت جریان‌های متقارن به کل جریان‌ها، نسبت جریان‌های نامتقارن به کل جریان‌ها، مقدار آنتروپی بر اساس آدرس IP مبدا برای جریان‌های TCP, UDP, ICMP را در نظر می‌گیرد.
۳. Traffic Classifier: بعد از اینکه ماژول قبل تمامی ویژگی‌های ذکر شده را استخراج کرد، این ماژول فراخوانی می‌شود. این ماژول بر اساس داده‌های آموزشی ترافیک ورودی را به ۸ دسته طبقه‌بندی می‌کند. که ۷ دسته شامل انواع حمله و یک دسته ترافیک نرمال می‌باشد. روش‌های مطرح شده نیاز به پردازش داده‌های آموزش دارند و عموما نمی‌توانند به صورت زودهنگام

حملات منع‌سروسی توزیع‌شده را تشخیص دهند.
در مقاله [۹] مولفه‌های اصلی ترافیک وارد شده به شبکه تحلیل و یک ماتریس از این ترافیک ایجاد می‌شود. از طریق تفاضل ماتریس در حالت حمله و نرمال می‌توان حمله را تشخیص داد. اما این ماتریس در زمان حمله بسیار بزرگ می‌شود و تحلیل آن نیاز به صرف زمان از سوی کنترلر است. در نتیجه روش مناسبی برای تشخیص زودهنگام نیست.

۲-۳ سایر روش‌ها

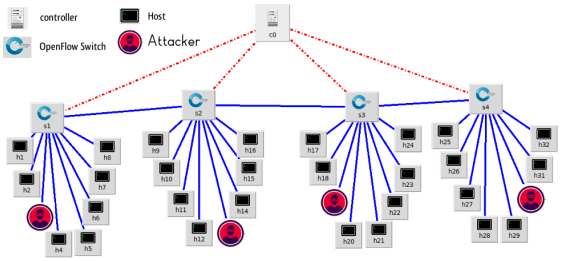
AVANT_GAURD [۱۱] یک روش مبتنی بر SYN cookies است که برای مقابله با حمله‌های SYN flood در شبکه‌های نرم‌افزارمحور طراحی شده است. در این روش لایه داده تبدیل به یک پراکسی بین کنترلر و مهاجم می‌شود. این پراکسی اطمینان حاصل می‌کند بسته‌هایی که دست دهی سه مرحله‌ای را تکمیل نکرده‌اند هیچگاه به سمت کنترلر ارسال نشوند. این روش فقط قادر است حملات مبتنی بر پروتکل TCP را تشخیص دهد و قادر به تشخیص دیگر حملات نیست. در Flowsec [۱۲] هنگامی که حمله بر اساس میزان مصرف منابع و ظرفیت لینک ارتباطی تشخیص داده شود، کنترلر یک پیام کنترلی به منظور کاهش نرخ ارسال داده به سوی کنترلر ارسال می‌کند. به این طریق میزان ازدحام لینک کنترل می‌شود. یکی دیگر از روش‌هایی که برای کاهش اثر حمله که مبتنی بر زمان‌بندی است پیشنهاد شده است، روش MLFQ [۱۳] است. در این روش در کنترلر برای هر سوییچ یک صف در نظر گرفته شده است که درخواست‌های هر سوییچ در آن قرار گرفته می شود. به محض افزایش درخواست سوییچ، صف مربوط به آن به چند زیر صف گسترش داده می‌شود. الگوریتم سرکشی به صف‌ها بر اساس نوبت‌دهی گردشی وزن دار می‌باشد. در این روش بسته‌های معمولی با سرعت بیشتری نسبت به بسته‌های تقلبی پردازش می‌شوند. به این صورت اثر حمله کاهش پیدا می‌کند. در [۱۴] یک روش زمانبندی مطرح شده است که شبیه به روش MLFQ است. در این روش نیز در کنترلر برای هر سوییچ یک صف در نظر گرفته می‌شود. برای هر صف متناسب با میزان حمله به سوییچ مربوطه یک ضریب اختصاص داده می‌شود. در نهایت کنترلر هنگام سرکشی به صف‌ها به نسبت عکس ضرایب به آنها time slot اختصاص می‌دهد. به این صورت هر سوییچ که به احتمال بیشتر مورد حمله قرار گرفته است برش زمانی کمتری برای پردازش درخواست‌های او توسط کنترلر در نظر گرفته می‌شود.

۳ مدل سیستم

در این بخش مدل سیستم برای مدل‌سازی و تشخیص حمله DDoS تشریح می‌شود. ابتدا مدل سیستم و بستری که در آنجا آزمایشات انجام شده است معرفی می‌شود. سپس نوع حمله انجام شده و تاثیراتی که این حمله به سیستم می‌تواند داشته باشد نشان داده می‌شود.

۳-۱ توپولوژی شبکه

توپولوژی استفاده شده در این مقاله مانند شکل ۲ از یک کنترلر متصل به ۴ سوییچ تشکیل شده است. به هر کدام از سوییچ‌ها ۸ میزبان متصل هستند. ترافیک مورد استفاده از نگاشت ترافیک یک شبکه واقعی به توپولوژی شکل ۲ بدست آمده است[۱۵].



شکل ۲: توپولوژی مربوط به شبکه مورد بررسی

۳-۲ سناریوی حمله

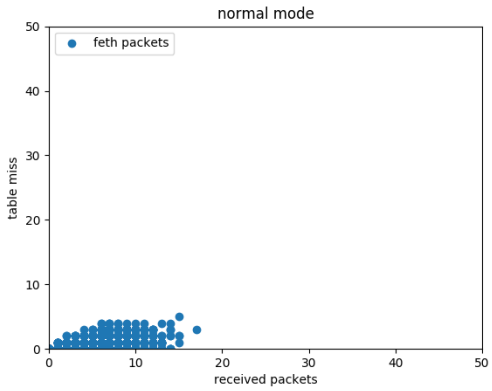
در توپولوژی شکل ۲ میزبان‌های H3, H13, H19, H30 به عنوان درگاه ورود حملات DDoS در نظر گرفته شده‌اند. این درگاه‌ها با شکل مشخص نشان داده شده‌اند. در این نوع حمله DDoS ، فرض می‌شود مهاجمین آدرس‌های میزبان‌های شبکه را در اختیار دارند. مهاجمین سعی می‌کنند بسته‌های جعلی خود را از طریق این درگاه‌هابه مقصد تمامی میزبان‌ها ارسال کنند. این نوع حمله باعث می‌شود نسبت خطای جدول سوییچ‌ها یا بسته‌های PACKET_IN به تعداد کل بسته‌های وارد شده به سوییچ افزایش پیدا کند. این نوع حمله به طور ویژه می‌تواند بخش‌های زیر از شبکه SDN را مورد هدف قرار دهد:

- کنترلر: با ایجاد تعداد خطای زیاد در جدول سوییچ‌ها، بسته‌های PACKET_IN با شدت زیاد روانه کنترلر می‌شوند. این روند می‌تواند بار پردازشی زیادی را به کنترلر تحمیل کند. در نتیجه ممکن است کنترلر برای مدت زمانی از دسترس کاربران بی‌خطر خارج شود.
- سوییچ : این حمله جداول سوییچ را پر می‌کند و مانع از تنظیم قوانین جدید در جدول سوییچ می‌شود. این اتفاق منجر به نادیده گرفتن بسته‌های بی‌خطر می‌شود.
- لینک ارتباطی بین سوییچ و کنترلر: پس از حمله، به دلیل ارسال بسته‌های زیاد PACKET_IN این لینک ارتباطی به سرعت پر می‌شود.

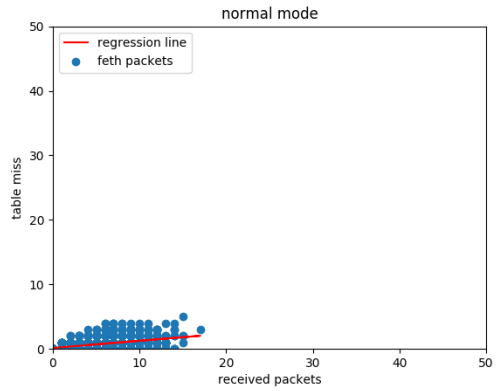
در این نوع حمله، مهاجمین یک سرور خاص را هدف قرار نمی‌دهند، بلکه سعی می‌کنند بسته‌های خود را به تمامی میزبان‌های موجود در شبکه ارسال کنند. درنتیجه آنتروپی تمامی جریان‌های وارد شده به سوییچ‌ها، به صورت یکنواخت بدون تغییر زیاد باقی خواهند ماند و تشخیص جریان مشکوک با استفاده از تکنیک‌های مبتنی بر آنتروپی میسر نخواهد بود. در بخش بعدی روش پیشنهادی برای تشخیص این نوع حمله معرفی خواهد شد.

۴ روش پیشنهادی

در این بخش روش پیشنهادی برای تشخیص حمله DDoS تشریح می‌شود. نشان داده می‌شود که بر اساس آزمایشات انجام شده و تحلیل آن‌ها چگونه یک مدل قابل قبول برای تشخیص حمله DDoS طراحی شده است.



شکل ۳: نمودار تعداد بسته‌های دریافتی ـ خطای جدول



شکل ۴: نمودار تعداد بسته‌های دریافتی ـ خطای جدول وخط رگرسیون

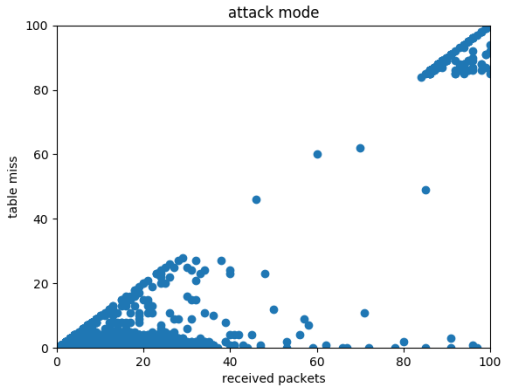
۴-۱ اصول روش

در شبکه‌های SDN به ازای هر بسته ورودی که قانون مربوط به آن در سوییچ تنظیم نشده باشد، یک خطای جدول اتفاق می‌افتد. نتیجه‌ی این خطا ارسال سرآیند بسته به سوی کنترلراست. پاسخ این خطا تنظیم قانون مناسب توسط کنترلر در سوییچ است. هدف، ایجاد یک مدل ریاضی برای شبکه SDN در حالت رفتار نرمال شبکه است. برای این کار نمودارهای مربوط به تعداد بسته‌های دریافتی هر سوییچ و تعداد خطاهای جدول آن‌ها مورد بررسی قرار گرفته است. به منظور ارزیابی و طراحی یک مدل منطقی برای تعداد خطاهای جدول سوییچ، شبکه SDN با سناریو ذکر شده در بخش ۳-۱ پیاده‌سازی شده است. برای این منظور شبکه‌ای با نرخ ارسال میانگین ۷ بسته در ثانیه پیاده‌سازی شده است. نمودار شکل ۳ وضعیت شبکه را در حالت نرمال بر حسب تعداد بسته‌های دریافتی سوییچ و تعداد خطاهای جدول سوییچ نشان می‌دهد. بررسی این نمودار نشان می‌دهد که با افزایش تعداد بسته‌های دریافتی هر سوییچ، تعداد خطاهای جدول آن سوییچ نیز افزایش پیدا کرده است.این آزمایش با تغییر در نرخ‌ارسال، تعداد سوییچ‌های متصل به کنترلر و تعداد میزبان‌های متصل به سوییچ‌ها تکرار شده است. نتایج مانند نمودار شکل ۳ نشان می‌دهد که می‌توان با رگرسیون خطی رابطه بین تعداد بسته‌های وارد شده به شبکه و تعداد خطاهای جدول سوییچ را مانند شکل ۴ توصیف کرد. خط رگرسیون [۸] مربوط به نمودار شکل ۳ با رنگ قرمز مشخص شده است. خطای مطلق میانگین این خط از رابطه (۱) بدست می‌آید:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i) \qquad (۱)$$

در این رابطه Y نشان دهنده تعداد خطای جدول سوییچ و $\hat{Y}$ تخمین این مقدار می‌باشد. خطای مطلق میانگین بدست‌آمده برای نمودار ۳ برابر مقدار ۶۲٫۰ است. اگر چه این خطا نسبتا زیاد است اما نتایج نشان می‌دهد که تقریب قابل قبولی برای تعداد خطاهای جدول سوییچ و در نتیجه تشخیص حمله می‌باشد.

برای مقایسه شبکه در حالت حمله و نرمال یک حمله پیاده سازی شده است. نمودار مربوط به تعداد خطاهای جدول سوییچ در حالت حمله مانند شکل ۵ می‌باشد. همانطور که مشخص است در حالت حمله شیب خط



شکل ۵: رابطه بین تعداد بسته‌های دریافتی و تعداد خطای جدول سوییچ در حالت حمله

رگرسیون به شکل قابل ملاحظه ای افزایش پیدا می‌کند. بیشترین مقدار شیب خط رگرسیون در حالت حمله، ۱ می‌باشد که نشان دهنده وقوع خطای جدول سوییچ برای تمامی بسته‌های وارد شده است. درگاههای مهاجمین در توپولوژی شکل ۲ مشخص شده‌اند. نرخ حمله برای هر درگاه ۱۰۰ بسته بر ثانیه تنظیم شده است و آزمایش حمله در ۶۰۰ ثانیه انجام گرفته است. به نظر می‌رسد که بتوان با تعریف یک حد آستانه برای شیب خط رگرسیون، حملات DDoS را در SDN تشخیص داد.

۴-۲ روش تشخیص

راه حل پیشنهادی برای تشخیص حمله منع سرویس از سه فاز کلی واکشی، تخمین و تشخیص تشکیل شده است:

- واکشی اطلاعات آماری شبکه: کنترلر در هر ثانیه تعداد بسته‌های دریافتی و تعداد خطاهای جدول هر سوییچ را واکشی می‌کند.
- تخمین پارامترهای خط رگرسیون آستانه به صورت پویا: پارامترهای خط آستانه در هر t ثانیه تخمین زده می‌شود. برای این منظور با استفاده از روش EWMA پارامترهای شیب و مقدار عرض از مبدا خط آستانه برای بازه بعدی تخمین زده می‌شوند.
- تشخیص حمله: در این فاز تعداد خطاهای جدول سوییچ با مقدار

تخمینی آستانه مقایسه می‌شود و در صورت تخطی، حمله تشخیص داده می‌شود.

۴-۲-۱ فاز۱: واکشی اطلاعات

در این فاز کنترلر در هر ثانیه اطلاعات تعداد بسته‌های دریافتی و تعداد خطاهای جدول سوییچ را از سوییچ‌ها درخواست می‌کند. این فاز به منظور جمع‌آوری و تحلیل داده‌ها در فازهای بعدی انجام می‌شود. پس از هر t ثانیه اطلاعات آماری جمع‌آوری شده در فازهای بعدی تحلیل می‌شود.

۴-۲-۲ فاز۲: تخمین معادله خطوط آستانه

خط آستانه یک معادله خط به صورت $ax+b$ می‌باشد. به منظور تعیین خط آستانه مطمئن‌تر پارامترهای شیب و عرض از مبدا به صورت پویا در هر ۵۰ ثانیه به استفاده از روش Exponetial Weighted Moving Aver-age برای بازه بعدی تخمین زده می‌شود. روش EWMA به این صورت است که بر اساس مقادیر گرفته شده در بازه فعلی، مقادیر بازه آتی را تخمین می‌زند. الگوریتم ؟؟ مراحل تخمین را نشان می‌دهد.
برای درک بهتر الگوریتم‌های ارائه شده، پارامترهای زیر تعریف شده است:

- $coef_{current}$: شیب خط برازش شده روی نقاط مرزی در بازه زمانی فعلی.
- $coef_{thr}$: شیب خط آستانه در بازه زمانی فعلی.
- $intercept_{current}$: عرض از مبدا خط برازش شده روی نقاط مرزی در بازه زمانی فعلی.
- $intercept_{thr}$: عرض از مبدا خط آستانه در بازه زمانی فعلی.
- $coef_{estimated}$: مقدار تخمینی شیب خط برازش شده روی نقاط مرزی در بازه زمانی بعدی.
- $pre\_coef_{estimated}$: مقدار $coef_{estimated}$ در بازه زمانی قبلی.
- $intercept_{estimated}$: مقدار تخمینی عرض از مبدا خط برازش شده روی نقاط مرزی در بازه زمانی بعدی.
- $pre\_intercept_{estimated}$: مقدار $intercept_{estimated}$ در بازه زمانی قبلی.
- dev_{coef}: حاصل قدرمطلق تفاضل شیب خط برازش شده روی نقاط مرزی در بازه زمانی فعلی و مقدار تخمینی آن در بازه زمانی بعدی.
- pre_dev_{coef}: مقدار dev_{coef} در بازه زمانی قبلی.
- $dev_{intercept}$: حاصل قدرمطلق تفاضل عرض از مبدا خط برازش شده روی نقاط مرزی در بازه زمانی فعلی و مقدار تخمینی آن در بازه زمانی بعدی.
- $pre\_dev_{intercept}$: مقدار $dev_{intercept}$ در بازه زمانی قبلی.

در الگوریتم ۱ تابع Estimate() معرفی شده است که کار تخمین پارامترهای شیب خط آستانه را انجام می‌دهد. ضرایب a, b دارای مقادیری بین صفر و یک می‌باشند و k یک عدد مثبت است. در خطهای ۳ الی ۵ همانند روش تخمین time out در پروتکل TCP، مقادیر مربوط به شیب و عرض از مبدا خط رگرسیون در بازه فعلی به عنوان ورودی داده

الگوریتم ۱ تابع Estimate() برای تخمین پارامترهای خطوط آستانه

ورودی: مقادیر $pre\_coef_{estimated}$ ، $coef_{current}$ ، pre_dev_{coef}
خروجی: $coef_{thr}$

1: Estimate($pre\_coef_{estimated}, coef_{current}, pre\_dev_{coef}$)
2: {
3: $coef_{estimated} \leftarrow (1-a) \times pre\_coef_{estimated} + (a \times coef_{current})$
4: $dev_{coef} \leftarrow (1-b) \times pre\_dev_{coef} + (b \times (|coef_{current} - pre\_coef_{estimated}|))$
5: $coef_{thr} \leftarrow coef_{estimated} + (k \times dev_{coef})$
6: $pre\_coef_{estimated} \leftarrow coef_{estimated}$
7: $pre\_intercept_{estimated} \leftarrow intercept_{estimated}$
8: $pre\_dev_{coef} \leftarrow dev_{coef}$
9: return $coef_{thr}$
10: }

می‌شود. سپس شیب و عرض از مبدا خط رگرسون برای بازه بعدی تخمین زده می‌شود. بر اساس مقادیر تخمین زده شده شیب و عرض از مبدا خط آستانه محاسبه می‌شود. در خطوط ۶ الی ۸ به منظور تخمین پارامترهای خطوط آستانه در بازه‌های بعدی، آماده سازی پارامترها انجام می‌شوند.
الگوریتم ۲ با استفاده از تابع Estimate() معادله خط آستانه را برای بازه بعدی محاسبه می‌کند. این الگوریتم ورودی‌های زیر را دریافت می‌کند و معادله خط آستانه اول را به عنوان خروجی برمی‌گرداند:

- x: تعداد بسته‌های دریافتی در بازه زمانی فعلی
- y: تعداد خطاهای جدول سوییچ مربوط به بازه فعلی

در ابتدای اجرای کنترلر، مدت زمانی برای جمع آوری اطلاعات آماری شبکه در حالت نرمال مورد نیاز است. این مدت زمان در الگوریتم ۲ با حرف T نشان داده شده است. پس از گذشت T ثانیه ابتدایی، داده‌های جمع‌آوری شده در خطوط ۴ الی ۷ با استفاده از روش رگرسیون خطی روی یک خط برازش می‌شوند. سپس در خطوط ۸ الی ۱۲ پارامترهای تخمین مقداردهی اولیه می‌شوند. بعد از انجام این مراحل، در هر t ثانیه پارامترهای خطوط آستانه تخمین زده می‌شوند. این عمل با استفاده از تابع Estimate() انجام می‌شود. در خطوط ۱۴ و ۱۵ شیب و عرض از مبدا خط آستانه اول تخمین زده می‌شود.

۴-۲-۳ فاز۳: تشخیص حمله

همانگونه که در شکل ۵ نشان داده شده است، در حالت حمله خط رگرسون داری شیب بیشتری است. پس می‌توان با مقایسه تعداد خطای جدول سوییچ و حد آستانه بدست آمده در الگوریتم ۲ حمله منع سرویس را تشخیص داد. الگوریتم ۳ روند تشخیص حمله را نشان می‌دهد. برای تشخیص حمله تعداد خطاهای جدول سوییچ با مقدار خط آستانه مقایسه می‌شود. در صورت تخطی مقدار ۱ و در غیر این صورت مقدار صفر برگردانده می‌شود.

الگوریتم ۲ بدست آوردن پارامترهای معادله خط آستانه برای بازه بعدی

ورودی: x و y

خروجی: پارامترهای معادله خط آستانه

Sleep(T)
$firsttime \leftarrow True$
while True do
  $reg \leftarrow LinearRegression()$
  $reg.fit(x, y)$
  $coef_{current} \leftarrow reg.coef\_$
  $intercept_{current} \leftarrow reg.intercept\_$
  if $firsttime = True$ then
    $pre\_coef_{estimated} \leftarrow coef_{current}$
    $pre\_intercept_{estimated} \leftarrow intercept_{current}$
    $pre\_dev_{coef} \leftarrow coef_{current} \times .5$
    $pre\_dev_{intercept} \leftarrow intercept_{current} \times .5$
  end if
  $coef_{thr} \leftarrow$ Estimate($coef_{current}, pre\_coef_{current}, pre\_dev_{coef}$)
  $intercept_{thr} \leftarrow$ Estimate($intercept_{current}, pre\_intercept_{current}, pre\_dev_{intercept}$)
  $firsttime \leftarrow False$
  return $coef_{thr}, intercept_{thr}$
  Sleep(t)
end while

الگوریتم ۳ تشخیص حمله منع سرویس

ورودی: $packet\_in, all\_packets, coef, intercept$

خروجی: تشخیص حمله منع سرویس

$x \leftarrow all\_packets$
$y \leftarrow packet\_in$
$y\_thr \leftarrow coef_{thr} \times x + intercept_{thr}$
if $y\_thr > y$ then
  return 1
else
  return 0
end if

الگوریتم ۴ شبه کد ایجاد حمله منع سرویس توزیع شده

ورودی: لیست آدرس‌هاس IP مربوط به توپولوژی مورد آزمایش

خروجی: انجام حمله منع سرویس توزیع شده

$targetIP \leftarrow$ list of 32 IP in our topology
$srcip$=RandIP()
$dstip$=RandIP($targetIP$)
send($IP(src$ =srcip, dst=dstip,inter=.01,loop=1)

۵ ارزیابی روش پیشنهادی

در این بخش ابتدا محیط آزمایش و ترافیک مورد استفاده برای ارسال بسته‌های نرمال معرفی می‌شوند. همچنین ابزار استفاده شده برای پیاده سازی حمله DDoS به همراه شبه‌کد آن بررسی می‌شوند. در نهایت نتایج حاصل از ارزیابی آورده شده است.

۵-۱ محیط آزمایش

توپولوژی مورد آزمایش و موقعیت مهاجمین در این پایان‌نامه در شکل ؟؟ آورده شده است. برای پیاده سازی این توپولوژی از شبیه‌ساز Mininet [۱۶] استفاده شده است. همانگونه که در شکل ؟؟ مشخص است مهاجمین حمله منع سرویس در هر کدام از سوییچ‌ها به شکل متفاوتی جایگذاری شده‌اند.

برای شبیه‌سازی از کنترلر Ryu[۱۷] و سوییچ OpenVswitch[۱۸] با ظرفیت ۲۵۶ جدول و ۳۲ میزبان متصل به آن‌ها استفاده شده است. مقادیر مربوط به hard_timeout و idle_timeout به ترتیب ۳ و ۱ ثانیه در تمامی سوییچ‌ها تنظیم شده است. پروتکل استفاده شده برای ارتباط بین کنترلر و سوییچ‌ها OpenFlow نسخه ۳ می‌باشد.

۵-۲ مجموعه داده ترافیک مورد استفاده

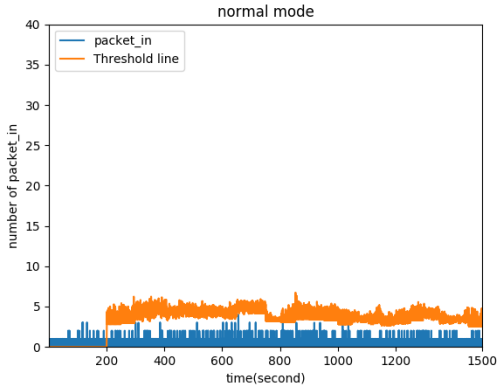
ترافیک نرمال ارسال شده توسط میزبان‌های بی‌خطر از یک شبکه واقعی برداشت شده است که دارای بیش از ۱۰۰۰ آدرس IP متفاوت می‌باشد [۱۵]. تمامی بسته‌های رد و بدل شده به این شبکه همراه با اطلاعات کامل آن‌ها ضبط و ذخیره شده است. از بین تمامی آدرس‌های موجود در شبکه واقعی ۳۲ عدد آدرس IP از پر ترافیک‌ترین آن‌ها انتخاب و به شبکه توپولوژی شکل ؟؟ نگاشت شده است. ترتیب و نرخ ارسال بسته‌های مربوط به آدرس‌های فیلتر شده بر اساس برچسب زمانی هر بسته در شبکه واقعی می‌باشد. به این صورت سعی شده است یک شبکه واقعی را طبق توپولوژی شکل ؟؟ پیاده‌سازی شود.

۵-۳ تولید بسته‌های حمله

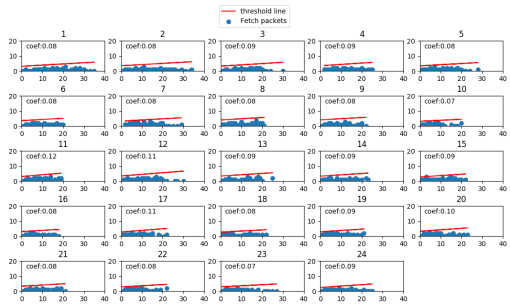
برای ایجاد حمله منع سرویس توزیع شده از ابزار scapy[۱۹] استفاده شده است. شبه کد مربوط به حمله منع سرویس توزیع شده در الگوریتم ۴ آورده شده است. در شبه کد ۴ هدف حملات منع سرویس توزیع شده ۲۸ آدرس IP نشان داده شده در توپولوژی شبکه شکل ؟؟ می‌باشند که در هر بار ایجاد حمله به صورت تصادفی عنوان آدرس مقصد انتخاب می‌شوند. آدرس مبدا بسته‌های حمله نیز به صورت تصادفی توسط تابع RandIP() تولید می‌شوند. به این صورت بسته‌های حمله با آدرس‌های مبدا متفاوت ساخته و به سوی تمامی میزبان‌های موجود در شبکه ارسال می‌شوند. بسته‌های حمله که دارای آدرس مبدا و مقصد هستند در خط ۴ ساخته و ارسال می‌شوند. پارامتر $inter$ فاصله زمانی بین ارسال بسته‌های حمله را نشان می‌دهد. با این پارامتر می‌توان نرخ ارسال بسته‌های حمله را تنظیم کرد. به عنوان مثال با تنظیم $inter = .01$ نرخ ارسال روی ۱۰۰ بسته در ثانیه تنظیم می‌شود. شبه کد ۴ توسط هر ۴ درگاه حمله که در شکل ؟؟ با نماد متفاوت نشان داده شده است، اجرا می‌شود.

۵-۴ نتایج و ارزیابی

در این بخش روش پیشنهادی شبکه طبق تنظیمات مطرح شده پیاده سازی شده است. ابتدا شرایط شبکه و نحوه عملکرد روش پیشنهادی در شرایط نرمال بررسی می‌شود. سپس حمله DDoS مطابق با الگوریتم؟؟ پیاده



شکل ۶: نمودار خط آستانه تخمینی نسبت به تعداد خطاهای سوییچ‌ها



شکل ۷: نمودار خط آستانه تخمینی نسبت به تعداد خطاهای سوییچ‌ها

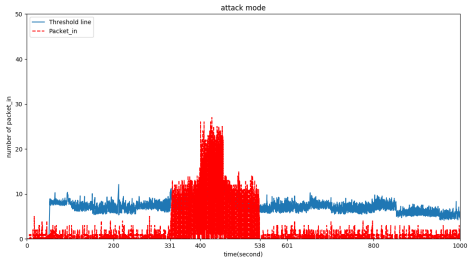
سازی می‌شود.

نمودار ؟؟ ، خط آستانه تولید شده توسط الگوریتم پیشنهادی را در حالت نرمال نشان می‌دهد. این آزمایش در ۱۵۰۰ ثانیه انجام شده است. از لحظه ۲۰۰ ثانیه به بعد خط آستانه تخمین زده می‌شود. همانگونه که در شکل مشخص است، این خط زرد رنگ با فاصله منطقی نسبت به تعداد خطاهای سوییچ تولید شده قرار گرفته است. در واقع طبق الگوریتم پیشنهادی، بر اساس مقادیر شیب و عرض از مبدا خط رگرسیون فعلی و بازه قبلی، مقادیر شیب و عرض از مبدا مربوط به بازه بعدی تخمین زده می‌شود. برای توصیف دقیقتر روش پیشنهادی نمودار ؟؟ موقعیت خط آستانه را نسبت به تعداد خطاهای جدول سوییچ‌ها در هر بازه ۵۰ ثانیه‌ای نشان می‌دهد. مقدار تخمینی شیب خط آستانه در بالای نمودار نوشته شده است.

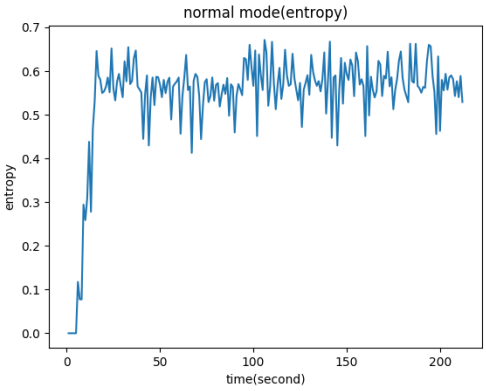
نمودار شکل ۸ حالت حمله شبکه را نشان می‌دهد. حمله در لحظه۳۵۱ الی ۵۳۸ رخ داده است. همانظور که مشخص است، در لحظه حمله خط آستانه توسط تعداد خطاهای جدول سوییچ قطع و درنتیجه تشخیص داده شده است.

۵-۵ مقایسه با روش مبتنی بر آنتروپی

در چند سال اخیر از تکنیک آنتروپی در تشخیص حملات منع سرویس توزیع شده به عنوان یک روش مناسب استفاده شده است و مقالات متعددی [۲، ۳، ۴، ۵، ۶، ۷] سعی در بهبود این روش داشته‌اند. تکنیک آنتروپی به صورت کامل در این مقاله پیاده‌سازی شده است. شکل ۹ نمودار آنتروپی را درحالت نرمال نشان می‌دهد. همانظور که مشخص است مقدار آنتروپی به

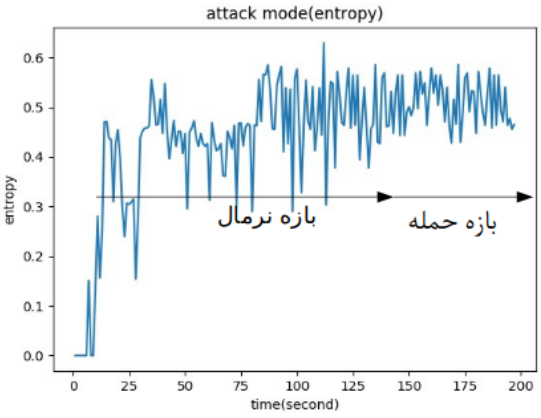


شکل ۸: تشخیص حمله با استفاده از خط آستانه



شکل ۹: حالت نرمال - آنتروپی بالا

صورت تقریبا ثابت قرار دارد. اما در حالت حمله معرفی شده در این مقاله، به دلیل اینکه بسته‌های حمله به صورت یکنواخت به تمامی میزبان‌های شبکه ارسال می‌شود، آنتروپی تغییر زیادی نمی‌کند. درنتیجه این تکنک قادر به تشخیص حمله نیست. شکل ۱۰ این موضوع رابه خوبی نشان می‌دهد.



شکل ۱۰: حالت حمله - آنتروپی بدون کاهش

۶ نتیجه‌گیری

به دلیل حساست بالای کنترلر، تشخیص زودهنگام حملات منع سرویس توزیع شده از اهمیت بالایی برخوردار است. در این مقاله سعی شده است با استفاده از ساختار SDN مدلی مبتنی بر رگرسیون خطی برای توصیف

رفتار شبکه ارئه شود و از طریق این مدل حملات منع سرویس توزیع شده را تشخیص داد. از آنجا که این مدل از نوع پروتکل و بسته‌های وارد شده به شبکه مستقل است و فقط بر اساس تعداد بسته‌های وارد شده و تعداد خطاهای جدول سوییچ شبکه را مدل می‌کند، می‌تواند محدوده بزرگی از حملات منع سرویس توزیع شده در SDN را تشخیص دهد. یکی از حملات جدید منع سرویس که در این مقاله به آن پرداخته شد، ارسال بسته‌های تصادفی به تمامی میزبان‌های شبکه است. از آنجا که این حمله آنتروپی شبکه را تغییر نمی‌دهد، توسط روش‌های مبتنی بر آنتروپی قابل تشخیص نمی‌باشد. این در حالی است که روش پیشنهادی به خوبی قادر به تشخیص این حمله می‌باشد.

مراجع

[1] D. Gao, Z. Liu, Y. Liu, C. H. Foh, T. Zhi, and H.-C. Chao, "Defending against packet-in messages flooding attack under sdn context," Soft Computing, vol.22, no.20, pp.6797–6809, 2018.

[2] S. M. Mousavi and M. St-Hilaire, "Early detection of ddos attacks against software defined network controllers," Journal of Network and Systems Management, vol.26, no.3, pp.573–591, 2018.

[3] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "Safety: Early detection and mitigation of tcp syn flood utilizing entropy in sdn," IEEE Transactions on Network and Service Management, vol.15, no.4, pp.1545–1559, 2018.

[4] N. Sambandam, M. Hussein, N. Siddiqi, and C.-H. Lung, "Network security for iot using sdn: Timely ddos detection," in 2018 IEEE Conference on Dependable and Secure Computing (DSC), pp.1–2, IEEE, 2018.

[5] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, "Jess: Joint entropy-based ddos defense scheme in sdn," IEEE Journal on Selected Areas in Communications, vol.36, no.10, pp.2358–2372, 2018.

[6] F. Rebecchi, J. Boite, P.-A. Nardin, M. Bouet, and V. Conan, "Traffic monitoring and ddos detection using stateful sdn," in 2017 IEEE conference on network softwarization (NetSoft), pp.1–2, IEEE, 2017.

[7] A. Rai, P. D Vyavahare, and A. Jain, "Distributed dos attack detection and mitigation in software defined network (sdn)," Anjana, Distributed DoS Attack Detection and Mitigation in Software Defined Network (SDN)(April 1, 2019), 2019.

[8] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A ddos attack detection method based on svm in software defined network," Security and Communication Networks, vol.2018, 2018.

[9] D. Wu, J. Li, S. K. Das, J. Wu, Y. Ji, and Z. Li, "A novel distributed denial-of-service attack detection scheme for software defined networking environments," in 2018 IEEE International Conference on Communications (ICC), pp.1–6, IEEE, 2018.

[10] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based ddos detection system in software-defined networking (sdn)," arXiv preprint arXiv:1611.07400, 2016.

[11] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp.413–424, ACM, 2013.

[12] M. Kuerban, Y. Tian, Q. Yang, Y. Jia, B. Huebert, and D. Poss, "Flowsec: Dos attack mitigation strategy on sdn controller," in 2016 IEEE International Conference on Networking, Architecture and Storage (NAS), pp.1–2, IEEE, 2016.

[13] P. Zhang, H. Wang, C. Hu, and C. Lin, "On denial of service attacks in software defined networks," IEEE Network, vol.30, no.6, pp.28–33, 2016.

[14] Q. Yan, Q. Gong, and F. R. Yu, "Effective software-defined networking controller scheduling method to mitigate ddos attacks," Electronics Letters, vol.53, no.7, pp.469–471, 2017.

[15] T. project team, "http://tcpreplay.appneta.com/wiki/captures.html," 2019.

[16] mininet project team, "http://mininet.org/," 2019.

[17] ryu project team, "Sdn framework ryu using openflow: https://osrg.github.io/ryu/," 2019.

[18] A. L. F. C. Project, "https://docs.openvswitch.org," 2019.

[19] scapy project team, "https://scapy.net/," 2019.

پانویس‌ها

[۱] Table Miss
[۲] SDN
[۳] Data Plane
[۴] Control Plane
[۵] DDoS
[۶] Early Detection
[۷] joint entropy
[۸] Regression Line