Archive of SID

نشریه علمی "پدافند الکترونیکی و سایبری"

سال هشتم، شماره ۴، زمستان ۱۳۹۹، ص ۳۹-۳۱

علمی– پژوهشی

ارائه روشی تسریع یافته جهت تشخیص ناهماهنگی قوانین در شبکه‌های مبتنی بر نرم‌افزار

رسول کیانی[۱]، علی بهلولی[۲*]

۱- کارشناسی ارشد ، ۲- استادیار، گروه معماری کامپیوتر، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان

(دریافت: ۱۳۹۸/۰۷/۰۴، پذیرش: ۱۳۹۹/۰۵/۱۵)

چکیده

شبکه‌های مبتنی بر نرم‌افزار به سبب سهولت و بهینه‌سازی فرآیند تنظیم شبکه‌ها مورد توجه قرار گرفته‌اند. این سهولت در مدیریت به‌دلیل تفکیک لایه‌های کنترل و داده از یکدیگر صورت گرفته است. در این شبکه‌ها تنظیم همبندی شبکه با استفاده از جدول‌های جریانی صورت می‌گیرد که دارای قوانین مخصوصی هستند و سرویس‌های شبکه‌ای همانند کیفیت سرویس، امنیت و .. تحت عنوان برنامه‌هایی در شبکه فعالیت می‌کنند. جدول‌های جریان می‌توانند توسط هر کدام از این سرویس‌ها در شبکه مورد تغییر قرار گیرند. دسترسی برنامه‌های مختلف شبکه به جدول‌های جریان در کنار اینکه تنظیم شبکه را آسان‌تر کرده است، می‌تواند منجر به ایجاد ناهماهنگی بین قوانین ثبت‌شده توسط چندین برنامه شود. وجود این ناهماهنگی‌ها علاوه بر مصرف بیش از اندازه حافظه در سوئیچ‌ها می‌تواند مشکلات امنیتی و کاربردی نیز برای شبکه ایجاد کند. با وجود مطالعاتی که تاکنون بر روی تشخیص ناهماهنگی در جدول‌های جریان انجام شده است، اما این روش‌ها نه تنها زمان پردازش زیادی را بر کنترل‌کننده تحمیل می‌کنند، بلکه در برخی از موارد به تنها برطرف کردن تضاد بسنده کرده‌اند. در این مقاله سعی در بهبود سرعت الگوریتم‌های تشخیص ناهماهنگی در جدول‌های جریان شبکه‌های مبتنی بر نرم‌افزار شده است. به همین منظور با استفاده از شمارنده قوانین موجود در جدول‌های جریان محدوده قوانین مورد بررسی کاهش پیدا کرده و سپس بر روی طیف محدودی از قوانین، الگوریتم تشخیص ناهماهنگی اجرا شده است. در ابتدا روش کلی چگونگی تسریع الگوریتم تشخیص گفته‌شده و پس از آن به بهبود این روش با استفاده از متغیرهای گوناگون پرداخته شده است.

کلیدواژه‌ها: شبکه‌های مبتنی بر نرم‌افزار، تشخیص ناهماهنگی، جدول‌های جریان، قانون جریان.

۱- مقدمه

شبکه مبتنی بر نرم‌افزار به‌واسطه جداسازی لایه‌های کنترل و داده توانسته انعطاف‌پذیری بالا و مدیریت آسانی را فراهم کند که به همین سبب مورد توجه بسیاری از پژوهش‌گران قرار گرفته و این شبکه‌ها را از شبکه‌های سنتی متمایز کرده است [۱]. وظایف لایه کنترل در واحدی به نام کنترل‌کننده انجام می‌شود و لایه داده به‌وسیله جدول‌هایی در سوئیچ‌های مبتنی بر نرم‌افزار پیاده‌سازی شده است. این جدول‌ها که به جدول‌های جریان معروف هستند با استفاده از قوانین ترافیکی خاصی سعی در مدیریت بسته‌های دریافتی می‌کنند [۲-۱].

جدول‌های جریان در شبکه‌های مبتنی بر نرم‌افزار شامل چندین متغیر اساسی از جمله الگوی تطبیق و عملیات مورد نظر است:

• الگوی تطبیق متغیری است که بیان می‌کند کدام یک از بسته‌های واردشده به سوئیچ باید مطابق با قانون مورد نظر عمل کند که خود می‌تواند شامل چندین پارامتر از جمله آدرس مبدأ و آدرس مقصد باشد [۳].

• عملیات قانون متغیری است که عملیات پیش‌بینی‌شده برای بسته مورد نظر را نگه‌داری می‌کند و چنانچه ترافیکی با مشخصات ذکرشده در الگوی تطبیق مطابقت داشته باشد، باید عملیات پیش‌بینی‌شده در این متغیر بر روی آن ترافیک لحاظ گردد [۳].

در مواقعی ممکن است ترافیکی به شبکه برسد که تاکنون سوئیچ مورد نظر مشابه آن را تجربه نکرده است، در این حالت سوئیچ از کنترل‌کننده درباره عکس‌العمل شبکه نسبت به آن بسته خاص سؤال می‌کند، کنترل‌کننده که با انواع واحدهای شبکه از جمله کنترل کیفیت و امنیت در ارتباط است، بسته مورد نظر را دریافت کرده و با استفاده از الگوریتم‌هایی که در هر واحد نوشته شده است، تصمیم نهایی و بهینه‌ترین تصمیم را برای سرنوشت آن بسته خاص گرفته و آن را نه تنها به سوئیچ مورد نظر، بلکه به هر کدام از اجزا شبکه که به‌نوعی با تصمیم مورد نظر ارتباط دارند، اطلاع می‌دهد [۴]. به این ترتیب برای ترافیک

* رایانامه نویسنده مسئول: bohlooli@eng.ui.ac.ir

www.SID.ir

Archive of SID

مشابه بعدی که به آن سوئیچ می‌رسد عملیات قبلاً در نظر گرفته شده و بار پردازشی و ترافیکی شبکه بسیار کم شده است.

جداسازی لایه‌ها موجب شکل‌گیری آسان‌تر و تفکیک‌پذیری تنظیمات مورد نیاز هر واحد در شبکه شده است. به‌طوری‌که هر واحد به فراخور نیاز خود سعی در ایجاد قانون در جدول‌های جریان می‌کند. به این ترتیب عمل فکر کردن و تصمیم‌گیری برای هر بسته از حوزه سوئیچ‌ها که نمی‌توانستند تصمیمات بهینه اتخاذ کنند فراتر رفته و بر عهده کنترل‌کننده‌ای گذاشته شده است که از تمامی مشخصات شبکه اطلاع داشته و همانند دانای کل شبکه می‌تواند تصمیماتی اتخاذ کند که در آن لحظه بهینه‌ترین تصمیم است[۵]. جدا از اینکه این تفکیک‌پذیری کیفیت و دقت تنظیم شبکه را بهبود بخشیده است، موجب بروز ناهماهنگی‌هایی در جدول‌های جریان هر سوئیچ شده که این ناهماهنگی‌ها در کنار مصرف بیش از حد نیاز منابع شبکه، در مواقعی ممکن است به کارایی شبکه نیز لطمه وارد سازد [۵].

در شبکه‌های مبتنی بر نرم‌افزار جدول‌های جریان هر سوئیچ می‌تواند توسط چندین واحد در شبکه مورد تغییر قرار گیرد. به‌عنوان مثال واحد تنظیم ترافیک شبکه و واحد دیوار آتش هر کدام به فراخور نیاز خود می‌توانند این جدول‌ها را ویرایش کنند[۶]. این در حالی است که هیچ‌کدام از این واحدها از عملیات صورت گرفته اطلاعی ندارند، به همین سبب می‌تواند شرایطی رخ دهد تا این قوانین با یکدیگر ناهماهنگ بوده و یا حتی متضاد یکدیگر باشند [۷].

برای مثال در جدول (۱) فرض کنید میزان ترافیک مبادله شده بین آدرس‌های۱۹۲,۱۶۸,۱,۱ و ۱۹۲,۱۶۸,۱,۲ به حدی زیاد است که واحد تنظیم ترافیک می‌خواهد لینک ارتباطی آن دو را عوض کرده و ترافیک را روی یک مسیر سریع‌تر و یا سبک‌تر هدایت کند و از سوی دیگر واحد امنیت این ترافیک را خطرناک تشخیص داده و برای جلوگیری از بروز هرگونه حمله سعی در نابود کردن بسته‌ها دارد، در این حالت مشابه بالا قوانین شماره ۱ و ۵ با یکدیگر در تضاد هستند و این در حالی است که هر کدام از این واحدها در لحظه خود بهترین عکس‌العمل را نشان داده‌اند. در سال‌های اخیر توجه به این موضوع در بین پژوهش‌گران حوزه شبکه، افزایش یافته به‌طوری‌که کارهای قابل توجهی در این زمینه به‌دست آمده است، اما نکته‌ای که این کارها به آن کمتر توجه داشته‌اند لزوم ثبت لحظه‌ای قوانین در جدول‌های جریان شبکه‌های مبتنی بر نرم‌افزار است به‌طوری‌که گاهی بر اثر تأخیر به وجود آمده از پردازش زیاد این روش‌ها تأثیر قوانین در شبکه را کم کرده و مشکلات امنیتی و یا کارایی برای شبکه به همراه داشته است[۸]. این مسئله در حالی رخ می‌دهد که بسیاری از این قوانین هیچ ناهماهنگی خاصی نسبت به یکدیگر ندارند، این پژوهش سعی دارد با به‌کارگیری ظرفیت جدول‌های جریان و اجازه دادن به قوانین برای اعمال در سوئیچ‌ها قبل از پردازش، مدت زمان این بررسی را کم کرده و کارایی آن را بهبود بخشد. در ادامه و در بخش ۲ به معرفی کارهای قبلی صورت‌گرفته در این زمینه پرداخته شده و سپس در بخش ۳ و ۴ روش پیشنهادی و ارزیابی آن مورد بررسی قرار خواهد گرفت.

جدول (۱): نمونه‌ای از جدول‌های جریان شبکه‌های مبتنی بر نرم‌افزار

شماره قانون	مبدأ	مقصد	عملیات
1	192.168.1.1	192.168.1.2	Drop
2	192.168.1.2	*.*.*.*	Forward:8
3	192.168.*.2	192.168.1.3	Drop
4	192.168.1.4	192.168.1.3	Forward:8
5	192.168.1.1	192.168.1.*	Forward:6

۲- کارهای مرتبط

خوشبختانه در سال‌های اخیر بحث پیرامون تشخیص ناهماهنگی در شبکه‌های مبتنی بر نرم‌افزار مورد توجه قرار گرفته است، به‌طوری‌که در چندین مقاله در حوزه تشخیص تضاد فعالیت می‌کنند.

FortNOX بر پایه ناکس، (اولین کنترل‌کننده مبتنی بر پروتکل OpenFlow) یک روش برای احراز اصالت بخش‌های مختلف در شبکه‌های مبتنی بر نرم‌افزار ارائه می‌دهد و هم‌زمان با این احراز اصالت وظیفه اولویت‌بندی بخش‌ها را نیز بر عهده خواهد گرفت [۹]. می‌توان گفت راه‌حل ارائه‌شده در FortNOX یک راه‌حل متمرکز است به‌طوری‌که قوانین موجود در FortNOX حفاظت‌شده هستند به این ترتیب زمانی که یک تضاد در بین قوانین موجودی که مدیر شبکه در سامانه FortNOX نوشته است و قانون جدید با استفاده از الگوریتمی که آن را Alias Set Rule Reduction می‌نامد، تشخیص داده شود، اگر قانون ناهماهنگ از بیرون از FortNOX بیاید رد شده و اگر درون سامانه FortNOX باشد با مشاهده اولویت بخش ایجادکننده قانون فعلی و اولویت مالک قانون جدید رأی خود را صادر می‌کند که در آن قانونی ایجاد خواهد شد که از اولویت بالاتری نسبت به دیگری برخوردار باشد[۹].

در VeriFlow با استفاده از فضای ایجادشده در بین کنترل‌کننده و سوئیچ‌های شبکه و مدل‌کردن شبکه به‌صورت گراف قبل از ایجاد قوانین در سوئیچ‌ها آنها را گرفته و قوانینی متناسب با سیاست‌های شبکه از لحاظ امنیتی و محرمانگی ایجادمی‌کند[۱۰]. هدف اصلی در این مقاله پردازش بلادرنگ

www.SID.ir

Archive of SID

قوانین از لحاظ امنیتی است و دیگر موارد ناهماهنگی در این تحقیق در نظر گرفته نشده است.

آویانگ و همکاران بر تشخیص تضاد بین قوانین موجود در شبکه‌های مبتنی بر نرم‌افزار روشی ارائه داده‌اند، در این روش با استفاده از رأی‌گیری بین عناصر مختلف ایجادکننده قانون، رأی به ایجاد و یا حذف قانون می‌دهند[۱۱]. نکته مهم در این باره وجود پارامترهایی برای تأثیر رأی هر بخش در سرنوشت قانون است که در این مقاله دو متغیر مهم و حیاتی وجود دارند: همسانی و دقت[۱۱]. با استفاده از این دو متغیر که برای هر عنصر در هر بازه زمانی و به نسبت هر بسته می‌تواند متفاوت باشد یک رأی‌گیری بر روی قانون انجام گرفته و در صورت به حدنصاب رسیدن، رأی به تصویب آن می‌دهد، نکته قابل توجه اینکه در این روش نوع خاص ناهماهنگی به نام تضاد برطرف خواهد شد[۱۱].

FlowChecker یک سازوکار برای تشخیص انسجام بین سوئیچ‌ها و کنترل‌کننده‌های شبکه، تعیین درستی قوانین با توجه به پروتکل اعمال شده و رفع مشکلات امنیتی شبکه‌های مبتنی بر نرم‌افزار ارائه می‌دهد [۱۲]. در این سیستم دو بخش Flow Visorو Flow Checker استفاده شده است که Flow Visor ، وظیفه گرفتن و نظارت بر دستورات برای ارسال به Flow Checker را بر عهده دارد تا این بخش بر اساس سازوکارهایی که در آن پیاده‌سازی شده است بتواند منسجم بودن و یا نبودن قوانین در لایه داده و پروتکل‌های موجود در کنترل‌کننده را تشخیص دهد[۱۲].

در بیشتر مسائلی که اولویت در آن مهم است یکی از مدل‌های رایج درخت است. این مدل با استفاده از شکل سلسله مراتبی خود به خوبی می‌تواند اولویت‌های شبکه را پیاده کند[۱۳]. HFT روشی مبتنی بر مدل‌کردن اجزا شبکه در قالب یک درخت است که هر چه به برگ‌های آن نزدیک می‌شویم ارتباط بین قوانینی که هر بخش ایجادمی‌کند بیشتر از پیش می‌شود [۱۳]. زمانی که الگوریتم وجود تضاد در بین قوانین دو زیر شاخه برادر شود، راه‌کاری را که مدیر شبکه برای هر قسمت ارائه‌کرده، اجرا می‌کند. در ادامه این تحقیق PANE با ارائه روشی که در آن منجر به اشتراک‌گذاری درخت در کل شبکه می‌شود، وجود ناهماهنگی در درخت را نیز برطرف می‌کند[۱۳].

وانگ و همکاران سعی کرده‌اند، جدا از بحث‌های امنیتی و تشخیص و تصحیح تضاد بین قوانین، به مدل کردن انواع دیگری از ناهماهنگی‌ها نظیر تکرار و پنهان‌شدن پرداخته‌اند. سپس با استفاده از این مدل به تشخیص این ناهماهنگی‌ها و تصحیح آنها اقدام نموده‌اند[۷].

در[۱۴] ژانگ و همکاران با استفاده از روابط بین قوانین موجود در جدول‌های جریان اقدام به ساخت یک درخت رابطه بین قوانین کرده و در صورت اضافه‌شدن، حذف‌شدن و یا تغییر یک قانون، درخت مذکور را برای کشف ناهماهنگی احتمالی جستجو می‌کند. به این ترتیب این روش علاوه بر صرف کردن زمان برای تمامی قوانین موجود در شبکه در زمان ثبت قوانین، مقداری از حافظه را نیز به‌منظور ساخت درخت رابطه مصرف می‌کند که این مسئله نیز علاوه بر زمان مصرف‌شده می‌تواند یکی دیگر از نکات مورد بحث باشد.

همان‌طور که گفته شد با توجه به ظرفیت و امکانات زیادی که جدول‌های جریان در شبکه‌های مبتنی بر نرم‌افزار در اختیار می‌گذارند تنها تعداد معدودی از پارامترها توسط این روش‌ها مورد استفاده قرار گرفته‌اند، پارامترهایی که با استفاده از آنها می‌توان دقت و سرعت روش‌های قبلی را به‌صورت قابل توجهی افزایش داد، در این طرح سعی شده با استفاده از ظرفیت حداکثری جدول‌های جریان در شبکه‌های مبتنی بر نرم‌افزار به سرعت و دقت خوبی در جهت بهبود روش‌های قبلی دست یافت.

۳- مروری بر مفاهیم پایه

در این بخش به معرفی انواع ناهماهنگی‌ها و مدل‌سازی ریاضی آنها پرداخته شده است، لازم به ذکر است که در روابط زیر R_X به معنی قانون شماره X و M به معنی متغیر الگوی تطبیق قوانین و A بیانگر عملیات پیش‌بینی‌شده در قوانین هستند. به‌طور کلی ناهماهنگی‌ها به‌صورت ۴ دسته تقسیم‌بندی می‌شوند[۱۵]:

۳-۱- ناهماهنگی سایه

در این نوع از ناهماهنگی‌ها قانونی که در حال نصب در شبکه است ترافیکی را پوشش خواهد داد که همگی آنها زیرمجموعه قانونی هستند که در جدول جریان وجود دارد[۷]، شکل ریاضی آن به‌صورت زیر به‌دست می‌آید:

$$\forall i,j : R_X(M_i) \subset R_y(M_j), R_X(A_i) \neq R_y(A_j) \qquad (۱)$$

۳-۲- ناهماهنگی تکرار

در این نوع از ناهماهنگی الگوی تطبیق در یک قانون زیرمجموعه قانون دیگر است و عملیات پیش‌بینی شده در هر دو قانون یکسان است[۷]:

$$\forall i,j : R_x(M_i) \subset R_y(M_j), R_x(A_i) = R_y(A_j) \qquad (۲)$$

www.SID.ir

Archive of SID

۳-۳- ناهماهنگی مرتبط‌بودن

در این نوع از ناهماهنگی الگوی تطبیق در دو قانون با یکدیگر هم‌پوشانی داشته اما عملیات پیش‌بینی شده در قوانین با یکدیگر مغایرت دارد[۷]:

$$\exists i, j : R_x\left(M_i\right) \subset R_y\left(M_j\right), R_x\left(A_i\right) \neq R_y\left(A_j\right) \quad (۳)$$

۳-۴- ناهماهنگی جامع‌بودن

در این نوع از ناهماهنگی الگوی تطبیق قانونی که قصد ثبت در جدول جریان را دارد صورت کلی‌تر قانونی است که در جدول حضور دارد در حالی که عملیات پیش‌بینی این دو قانون با یکدیگر مغایر است [۷]:

$$\forall i, j : R_y\left(M_i\right) \subset R_x\left(M_j\right), R_y\left(A_i\right) \neq R_x\left(A_j\right) \quad (۴)$$

۴- روش پیشنهادی جهت تسریع الگوریتم تشخیص ناهماهنگی

در بخش‌های قبلی گفته شد روش‌های فعلی که در حوزه تشخیص ناهماهنگی‌ها در جدول‌های جریان مورد استفاده قرار می‌گیرند، نه تنها زمان زیاد و پردازش‌های زیادی را بر کنترل‌کننده تحمیل می‌کنند بلکه در برخی از موارد تنها به برطرف‌کردن تضاد بسنده کرده‌اند.

جدول‌های جریان در شبکه‌های مبتنی بر نرم‌افزار جدا از متغیرهایی مثل الگوی تطبیق و عملیات پیش‌بینی شده دارای مواردی دیگر از جمله شمارنده هر قانون و همچنین اولویت هر قانون هستند[۱۴و۱۶]. شمارنده هر قانون به معنی شمارش تعداد دفعات استفاده از هر قانون به‌منظور حفظ تعادل در لینک‌های شبکه استفاده می‌شود[۱۴و۱۶]. به این ترتیب هر زمان بسته‌ای به سوئیچ مورد نظر رسید که با استفاده از یک قانون امکان مدیریت آن فراهم بود عملیات آن قانون روی این بسته خاص انجام شده و به مقدار شمارنده یک واحد افزوده می‌شود. شمارنده در شبکه‌های مبتنی بر نرم‌افزار می‌تواند برای هر متغیر خاص مانند قوانین موجود در جدول و یا پورت‌هایی خاص درون سوئیچ در نظر گرفته شود، هم‌چنین می‌توان به‌صورت مدت زمان استفاده نیز این متغیر را تنظیم کرده که در این صورت با دقت یک ثانیه گزارش‌گیری انجام خواهد شد. بنابر استانداردهای قرار داده شده برای این متغیر، سرریز در مقدار آن وجود نخواهد داشت و چنانچه این متغیر از مقدار بیشینه خود بیشتر شود، مقدار بیشینه در آن قرار گرفته و ثابت خواهد ماند، به همین دلیل مشکلی از لحاظ مقدارهای سرریز شده وجود ندارد. به این ترتیب به کمک این متغیر تعداد دفعات استفاده از هر قانون ثبت‌شده و در گزارش‌گیری‌های آماری از شبکه این مقدار نیز همراه متغیرهای دیگر گزارش می‌شود.

دومین متغیری که در جدول‌های جریان شبکه‌های مبتنی بر نرم‌افزار باید محاسبه شود، اولویت هر قانون است[۱۴]. این متغیر بسته به نوع واحدی که قانون را درون جدول جریان سوئیچ قرار می‌دهد می‌تواند متفاوت باشد. برای مثال واحد امنیت باید قوانینی با اولویت بالا ثبت کند و به همین ترتیب باقی واحدها نیز بسته به اهمیت قوانین هر واحد اولویت‌های متفاوتی را ثبت خواهند کرد. با استفاده از این متغیر در جدول‌های جریان می‌توان تأثیر یک قانون در شبکه را کمتر از بقیه قوانین کرده و از خرابی‌های احتمالی آن جلوگیری کرد. به این ترتیب مادامی که حضور یک قانون و مفیدبودن آن مشکوک باشد، می‌توان با کمتر کردن اولویت آن و افزایش تدریجی این اولویت اعتماد لازم را درون شبکه برای این قانون خاص به‌دست آورد.

سومین متغیری که در جدول‌های جریان شبکه‌های مبتنی بر نرم‌افزار قرار دارد، مدت زمان معتبربودن یک قانون است[۱۴]، در این متغیر مدت زمانی که واحد مورد نظر برای این قانون در نظر گرفته که در سوئیچ عملیات خاصی را انجام دهد، ذخیره شده است.

درجدول‌های جریان موجود در شبکه‌های مبتنی بر نرم‌افزار چنانچه چند قانون نسبت به یکدیگر دارای ناهماهنگی باشند، یکی از قوانین بر روی ترافیک مورد نظر اجرا شده و باقی آنها اصلاً آن ترافیک را مشاهده نمی‌کنند، به همین دلیل در بعضی از ناهماهنگی‌ها از این قوانین استفاده نمی‌شود و هیچ ترافیکی را نمی‌توان با این قوانین مدیریت کرد[۱۴]. به بیان دیگر بعضی از متغیرهای آماری هم‌چون شمارنده درجدول‌های جریان نیز عدد صفر و یا نزدیک به صفر را خواهند داشت چون هیچ بسته‌ای به این قوانین نرسیده است تا با اعمال شدن این قوانین روی آن متغیرشمارنده آن افزایش پیدا کند. در این مقاله با استفاده از این خاصیت قوانین ناهماهنگ، قوانین مشکوک شناسایی و الگوریتم تشخیص ناهماهنگی تنها بر روی آن‌ها اجرا خواهد شد.

در روش گفته‌شده در این مقاله به هر قانونی که از سوی واحدهای مختلف برای اعمال و ایجاد به سمت کنترل‌کننده فرستاده می‌شود، اجازه ثبت خواهیم داد و برخلاف روش‌های

www.SID.ir

Archive of SID

قبلی، در موقع نصب قوانین روابط هر قانون را با قوانین دیگر بررسی نخواهد شد. به این ترتیب قوانین ناهماهنگ و هماهنگ به‌صورت کلی وارد شبکه می‌شوند و قوانینی که به هر دلیلی با باقی قوانین موجود هماهنگ نبوده در مدیریت ترافیک شبکه موفق نبوده و مقدار شمارنده آنها عددی نامفهوم (برای مثال صفر) خواهدداشت و در گزارش‌گیری‌های دوره‌ای که در شبکه انجام می‌گیرد شناسایی و الگوریتم تشخیص ناهماهنگی روی آن‌ها اجرا خواهد شد. در ادامه بررسی قانون مورد نظر الگوریتم به ۳ حالت مختلف خواهدرسید:

- قانون مورد نظر هیچ ناهماهنگی خاصی با دیگر قوانین جدول ندارد، این حالت به‌دلیل عدم وجود ترافیک مورد نظر در آن برهه خاص در شبکه شده است و ناهماهنگی خاصی را ایجاد نکرده است.
- قانون مورد نظر، خود موجب ناهماهنگی شده است، این حالت وقتی رخ خواهد داد که قانون مورد نظر دارای اولویت پایین‌تر از قانون کلی‌تری است که در جدول وجود دارد و عدم وجود آن صدمه خاصی را به شبکه وارد نکرده و می‌توان آن را از بین قوانین موجود در جدول حذف کرد. این اولویت توسط واحد ثبت‌کننده قانون در هنگام ثبت قوانین مشخص گردیده و برای هر قانون عدد منحصر بفردی است.
- دیگر قوانین موجب ناکارآمدی این قانون شده‌اند، در این حالت قانون مورد نظر در حالت بهینه خود قرار دارد اما دیگر قوانین موجود به‌واسطه اولویت بالاتر خود و الگوی تطبیقی که مشابه با قانون مورد نظر است ترافیک این قانون را نیز به سمت خود کشیده و موجب ناهماهنگی شده‌اند. در این حالت با اعمال الگوریتم تشخیص قوانین ناهماهنگ به‌راحتی تشخیص داده خواهند شد.

همان‌طور که گفته شد برای به‌دست آوردن اینکه قوانین موجود در جدول‌های جریان با دیگر قوانین ناهماهنگ هستند یا خیر باید شرایط بوجود آمده در متغیرشمارنده را بررسی شده و بر وجود یا عدم وجود ناهماهنگی تصمیم گرفت. اما اینکه اعداد ثبت‌شده در متغیر شمارنده چه زمانی غیرعادی محسوب خواهند شد، مسئله‌ای است که باید در نظر گرفته شود. اگر احتمال وجود قوانین ناهماهنگ به‌صورت زیر نشان داده شود:

p_{Sh} احتمال وجود ناهماهنگی سایه

p_{Co} احتمال وجود ناهماهنگی مرتبط بودن

p_{Re} احتمال وجود ناهماهنگی تکرار

p_{Ge} احتمال وجود ناهماهنگی جامع بودن

p_T احتمال عدم وجود ناهماهنگی

که روابط زیر بین این احتمالات حاکم است:

$$p_T = 1-\left(p_{Sh}+p_{Co}+p_{\mathrm{Re}}+p_{Ge}\right) \quad (۵)$$

$$p_T > p_{Sh}, p_{Co}, p_{\mathrm{Re}}, p_{Ge} \quad (۶)$$

رابطه (۵) از ارتباط بین قوانین در یک شبکه مبتنی بر نرم‌افزار نتیجه شده است که یکی از پنج حالت فوق است پس احتمالات مکمل یکدیگر محسوب می‌شوند.

رابطه (۶) بیان می‌کند که احتمال اینکه قوانین موجود در شبکه نسبت به یکدیگر دارای ناهماهنگی باشند کمتر از احتمال عدم وجود ناهماهنگی بین قوانین است.

به این ترتیب باید شمارنده‌ای که انتخاب می‌شود تابع صحیحی از این احتمالات باشد که بسته به نوع شبکه متفاوت خواهد بود.

رابطه زیر بین احتمالات گفته‌شده و حد آستانه شمارنده در نظر گرفته خواهد شد:

$$Th = \left(K(1-p)/N\right)\sum_{\forall i} C_i \quad (۷)$$

که در این رابطه Th بیانگر حد آستانه برای شمارنده، K ضریب تأثیر، C_i بیانگر مقدار شمارنده برای قانون شماره i و p بیانگر متوسط احتمال وجود ناهماهنگی بین قوانین و N تعداد کل قوانین است.

همچنین مقدار متغیر احتمالی p نیز به‌صورت زیر محاسبه گردیده است:

$$P = \frac{1}{n}\sum_{\forall i=0}^{n} p_i \quad (۸)$$

در این رابطه مقدار n برابر با تعداد انواع ناهماهنگی‌هایی است که می‌توان در سیستم تعریف کرد. به این ترتیب مقدار p_i برابر با احتمال رخداد هر یک از آن‌هاست.

در رابطه بالا با استفاده از مجموع شمارنده‌ها برای کل قوانین موجود در جدول جریان، تعداد بسته‌های رسیده تا آن لحظه به سوئیچ مورد نظر دریافت شده و سپس با استفاده از متوسط احتمالات موجود می‌توان تقریب خوبی برای مقدار حد آستانه به‌دست آورد. مقدار محاسبه شده برای متوسط احتمال را می‌توان میانگین احتمالات تمامی ناهماهنگی‌های در نظر گرفته شده

www.SID.ir

Archive of SID

دانست در حالی که این مقدار می‌تواند با استفاده از روش های دیگری نیز به‌صورت دقیق تر محاسبه گردد که در ادامه این موضوع را بحث خواهیم کرد. مقدار K میزان دقت احتمالات موجود در شبکه که توسط مدیر شبکه مشخص می‌گردد را بیان می‌کند. این میزان که عددی از یک بیشتر است مشخص‌کننده تأثیر احتمالات بیان‌شده در مقدار حد آستانه خواهد بود.

به‌منظور پوشش دادن طیفی از شمارنده‌های با مقادیر درست در الگوریتم از اختلاف بین میانه و میانگین مجموعه شمارنده‌ها استفاده خواهیم کرد. میانه مجموعه می‌تواند حد وسط مجموعه شمارنده‌ها را مشخص کرده و قدرمطلق اختلاف بین این مشخصه با مقدار میانگین شمارنده‌ها معیار مناسبی برای حدود مقادیر شمارنده‌های قوانین مختلف است به عبارت دیگر قوانینی که شمارنده آن‌ها در بازه زیر قرار داشته باشد درست و خارج از بازه زیر مشکوک تلقی خواهند شد. در رابطه زیر Th حد آستانه محاسبه‌شده از رابطه شماره ۷، N تعداد کل قوانین، C_i شمارنده هر قانون و $Mean$ مقدار میانه مجموعه کل شمارنده‌ها می‌باشد.

$$TS = \left(Th - \left(\left|\left(1/N\right)\sum_{\forall i} C_i - Mean\left(C\right)\right|\right), Th + \left(\left|\left(1/N\right)\sum_{\forall i} C_i - Mean\left(C\right)\right|\right)\right) \quad (۹)$$

پس از مشخص‌شدن قانون مشکوک مطابق با حد آستانه به‌دست‌آمده الگوریتم تشخیص ناهماهنگی برای این قانون اجرا خواهد شد، به این ترتیب می‌توان از گستردگی طیف قوانین نیازمند بررسی کاسته و نیاز پردازشی و هم‌چنین زمان اجرای الگوریتم را کاهش داد.

در شبه کد جدول (۲)، روش کلی الگوریتم تشخیص ناهماهنگی نشان داده شده است، این شبه کد شامل ۳ بخش کلی است:

- بخش اول به‌منظور تفکیک قوانین مشکوک از قوانین سالم قرار داده شده است که با استفاده از روابطی مقدار حد آستانه را به‌دست آورده و سپس شمارنده هر قانون را با آن مقایسه خواهد کرد.
- بخش دوم شامل تفکیک روابط میان قوانین مشکوک است، در این بخش مشخص می‌شود که آیا قوانین مشکوک واقعا ناهماهنگ هستند یا خیر؟
- بخش سوم شامل تشخیص ناهماهنگی بین قوانین است، با استفاده از دو بخش قبلی مجموعه قوانین ناهماهنگ به‌دست‌آمده و در اینجا نوع ناهماهنگی مشخص خواهد شد.

جدول (۲): شبه کد الگوریتم تشخیص ناهماهنگی تسریع یافته

Detection Algorithm (C , P , M , A , K)

```
1.relation = unknown
2.anomaly = false
3.if  C_x ∉ TS  then
4.   for all variables k in M then
5.      if  M_x[k]  ⊇  M_y[k]  then
6.         if                                  relation
∈ {superset, correlated} then
7.            relation = correlated
8.         else
9.            relation = subset
10.      else if  M_x[k]  ⊂  M_y[k]  then
11.         if                                  relation
∈ {subset, correlated} then
12.            relation = correlated
13.         else
14.            relation = superset
15.      else
16.         return Disjoint
17.   end for
18.   if  A_x = A_y  then
19.      return Redundancy
20.   else
21.      switch (relation)
22.         case superset
23.            return  General
24.         case subset
25.            return  Shadowing
26.         case correlated
27.            return  Correlation
28.         end switch
29.   end if
30.end if
```

تا به این‌جا با ایده اصلی روش گفته‌شده در این پژوهش و نمونه ساده آن آشنا شدیم، هم‌چنین بیان شد که برای تفکیک قوانین ناهماهنگ از قوانین صحیح از معیار شمارنده قوانین استفاده کردیم که به‌وسیله یک حد آستانه مقدار آن کنترل‌شده و چنان‌چه مقدار شمارنده از حدود بازه محاسبه‌شده خارج باشد، قانون مورد نظر ناهماهنگ تلقی خواهد شد.

مهم‌ترین چیزی که باعث دقت الگوریتم گفته‌شده در این مقاله خواهد شد مقدار احتمال‌هایی است که برای هرکدام یک از ناهماهنگی‌ها در نظر گرفته شده است. در ابتدا فرض شد که این احتمال‌ها به‌صورت ثابت بوده و با کارکردن شبکه تغییر نخواهند کرد، اما به‌عنوان گامی رو به جلو در این بخش الگوریتم به سمت هوشمندی خاصی سوق داده و این احتمالات با توجه به آثار و نتایج به‌دست آمده از الگوریتم تغییر داده می‌شود.

نکته دیگر چگونگی به‌دست‌آمدن این حد آستانه است، به‌طوری‌که در بخش قبلی از متوسط‌گیری احتمال‌های هر کدام

www.SID.ir

Archive of SID

از ناهماهنگی‌ها به‌دست آمد. به‌عنوان دو ویژگی شاخص می‌توان موارد زیر را به الگوریتم افزود:

احتمال وجود هر ناهماهنگی با استفاده از تاریخچه شبکه نیز می‌تواند به‌دست آید، چنانچه قوانین ناهماهنگی در جدول‌های جریان یافت شود می‌توان بر احتمال وجود آن ناهماهنگی خاص افزود و چنانچه در دفعات پی‌درپی الگوریتم ناهماهنگی خاصی پیدا نشد، می‌توان احتمال وجود آن ناهماهنگی خاص را کاهش داد. به این ترتیب حد آستانه با کارکردن شبکه مقدار جدیدتری دریافت خواهد کرد.

متوسط‌گیری در احتمال‌های انواع ناهماهنگی‌ها را می‌توان وابسته به شرایط هر ناهماهنگی دانست برای مثال ناهماهنگی مرتبط بودن تأثیرات امنیتی شدیدی ایجاد خواهد کرد، در حالی که ناهماهنگی تکراری‌بودن در عمل تنها منابع بیش از حد را مصرف خواهد کرد، به این ترتیب با متوسط‌گیری وزن‌دار بین احتمال‌های هر کدام از ناهماهنگی‌ها می‌توان تأثیر آن را در تشخیص الگوریتم تقویت کرد.

لازم به‌ذکر است در الگوریتم پیشنهادی به سبب اجرای در لحظه قوانینی که سرویس‌های مختلف شبکه پیشنهاد داده‌اند میزان خرابی شبکه و یا تأخیر به وجود آمده در سایر الگوریتم‌ها کاهش پیدا کرده و در نتیجه از میزان packet loss احتمالی شبکه نیز کاسته خواهد شد.

۵- ارزیابی

همان‌طور که در بخش‌های قبلی مشاهده شد تحقیقات پیشین که بر روی تشخیص ناهماهنگی‌ها در شبکه‌های مبتنی بر نرم‌افزار صورت گرفته است، همگی در زمان نصب این قوانین بر روی سوئیچ‌ها کارکرده و الگوریتم‌ها پیش از زمان نصب انجام شده‌اند. به این ترتیب این الگوریتم‌ها بدون در نظرگرفتن اینکه آیا قوانین در شبکه با یکدیگر ناهماهنگ هستند یا خیر، بر روی تمامی قوانین اجراشده و به همین دلیل زمان زیادی از شبکه صرف تشخیص ناهماهنگی می‌شود، که در عمده زمان‌ها اصلاً وجود نخواهد داشت. در این مقاله الگوریتم پیشنهادی تنها به ازای وجود قوانین مشکوک صرف‌شده و در غیر این‌صورت الگوریتم اجرا نخواهد شد. به‌منظور روشن‌شدن کارایی این الگوریتم و در صورت در نظر گرفتن مدت زمان اجرای الگوریتم در کارهای مشابه با نماد T ، مدت زمان تقریبی اجرای روش پیشنهادی ما در بهینه‌ترین حالت ممکن به‌صورت زیر پیش‌بینی می‌شود:

$$T_{tot} = T\left(1-p\right) \qquad (۱۰)$$

که در این رابطه p احتمال وجود قوانین هماهنگ و درست در جدول‌های جریان و T_{tot} مدت زمان تقریبی اجرای الگوریتم در بهترین حالت است.

به‌منظور ارزیابی بهتر، الگوریتم پیشنهادی با استفاده از شبیه‌ساز mininet و بر روی سیستم‌عامل لینوکس، توزیع fedora با مقدار حافظه اصلی ۲ گیگابایت و بر پایه کنترل‌کننده ریو تست شده است.

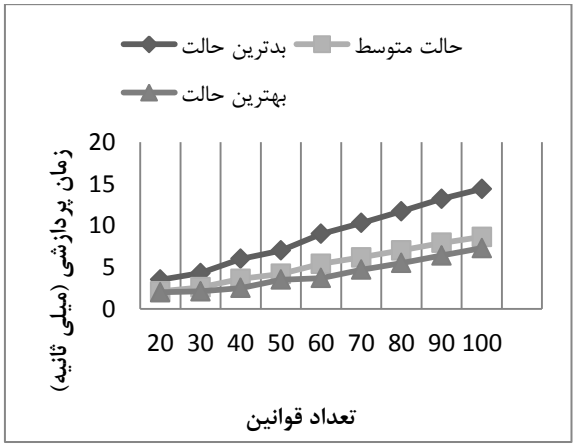
همچنین مطابق روش‌های متداول برای ارزیابی الگوریتم‌ها سعی‌شده سه دسته قوانین به‌عنوان ورودی در الگوریتم اجرا شوند تا بتوان زمان پردازشی الگوریتم برای سه وضعیت بدترین، متوسط و بهترین حالت ورودی بررسی کرد. این سه مجموعه با توجه به نوع نقشی که در ارزیابی شبکه دارند، دارای درصدی از ناهماهنگی‌های ذکر شده در بالا هستند که می‌توان به‌صورت زیر بیان کرد.

جدول (۳): دسته‌های مختلف توزیع ناهماهنگی و درصد رخداد آن‌ها

نام دسته	سایه	تکرار	مرتبط بودن	جامع بودن
بهترین حالت	٪۰	٪۵	٪۲	٪۰
حالت متوسط	٪۲	٪۹	٪۳	٪۲
بدترین حالت	٪۵	٪۱۳	٪۷	٪۵

همان‌طور که مشخص است این دسته‌ها به‌گونه‌ای انتخاب شده‌اند تا بتوان الگوریتم پیشنهادی را با سه حالت مختلف و نزدیک به واقعیت بررسی کرد.

زمان پردازشی الگوریتم برای این سه دسته از قوانین که مطابق با مجموعه‌های گفته‌شده در [۱۵] بر الگوریتم اعمال شده است، در شکل (۱) قابل مشاهده است.

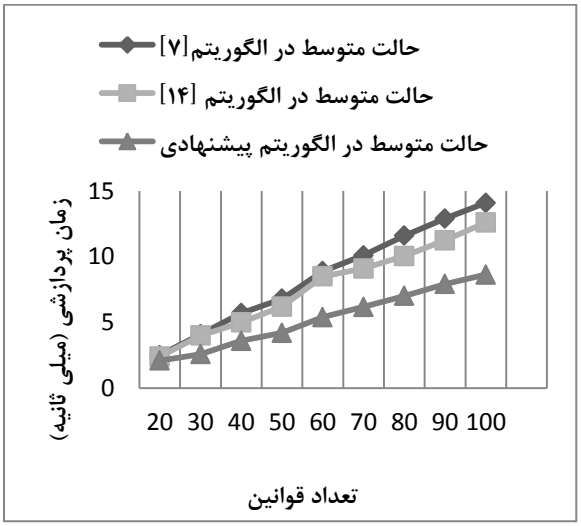


شکل (۱): نمودار به‌دست‌آمده از مدت زمان اجرای الگوریتم بر حسب تعداد قوانین موجود در جدول جریان

همچنین برای مقایسه هرچه بهتر الگوریتم پیشنهادی با نمونه‌های مشابه نمودار حالت متوسط این الگوریتم با روش‌های پیشنهادی در [۷] و [۱۴] نیز در شکل (۲) آورده شده است. همان‌طور که در شکل (۲) مشاهده می‌شود در حالت یکسان بین دو روش، الگوریتم پیشنهادی زمان اجرای کمتری نسبت به دو الگوریتم دیگر دارد.

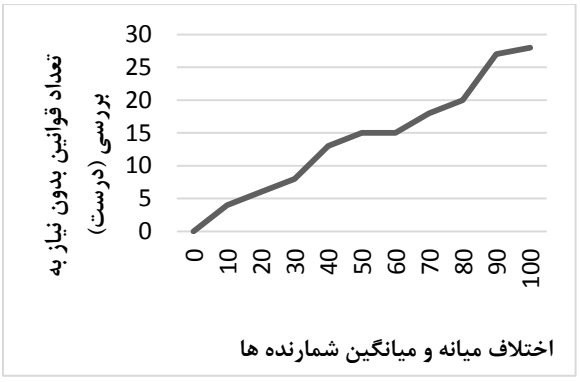
www.SID.ir

Archive of SID



شکل (۲): مقایسه زمان پردازشی الگوریتم پیشنهادی با الگوریتم‌های ارائه‌شده در مراجع [۷] و [۱۴]

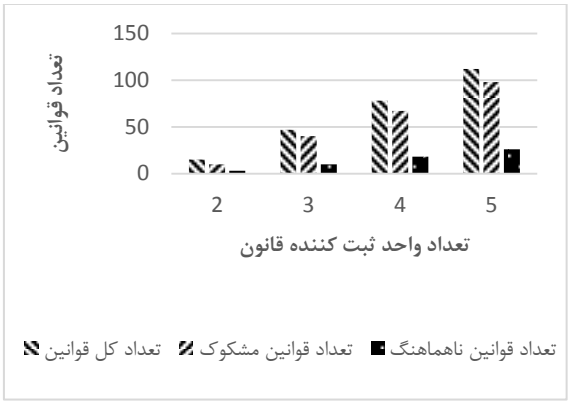
یکی از مقادیری که در مقایسه قوانین مشکوک و درست در الگوریتم گفته‌شده تعیین‌کننده است، مقدار اختلاف میانگین و میانه شمارنده‌های قوانین خواهد بود زیرا اختلاف این دو متغیر منجر به کاهش و یا افزایش بازه قوانین مشکوک به ناهماهنگی در الگوریتم پیشنهادی می‌شود. در شکل (۳) سعی شده است در حالت‌های مختلف اختلاف میانه و میانگین شمارنده‌های قوانین را زیاد کرده تا بتوان تأثیر آن را بر عملکرد الگوریتم مشاهده کرد.



شکل (۳): نمودار تعداد قوانین درست برحسب اختلاف میانه و میانگین مقادیر

همان‌طور که مشخص است با افزایش اختلاف این دو متغیر بازه تعداد شمارنده درست افزایش پیدا کرده و الگوریتم تشخیص، تعداد قانون بیشتری را در ابتدای امر درست تشخیص می‌دهد به این ترتیب الگوریتم برای تعداد کمتری از قوانین اجرا خواهد شد تا زمان اجرای کل فرآیند تشخیص را کاهش‌دهد.

گفته‌شد که یکی از دلایل ایجاد ناهماهنگی در جدول‌های جریان وجود تعداد واحدهای زیاد برای ایجاد قانون در این جدول‌ها است. واحدهایی که در اکثر مواقع از وضعیت یکدیگر بی‌خبر بوده و همین امر بر شدت ناهماهنگی‌ها تأثیرگذار خواهد بود. در شکل (۴) نمودار تعداد قوانین ناهماهنگ بر حسب تعداد واحدهای تأثیرگذار در شبکه نشان داده شده است. هم‌چنین با توجه به چگونگی نوع الگوریتم تشخیص گفته‌شده در این مقاله، به ترتیب ستون‌ها از چپ به راست کل قوانین موجود در جدول‌های جریان در ستون اول، قوانین مشکوک به ناهماهنگی، در ستون دوم و قوانینی که پس از اجرای الگوریتم، ناهماهنگ تشخیص داده‌شده، در ستون سوم نشان داده شده است.



شکل (۴): نمودار تعداد قوانین مشکوک و ناهماهنگ بر حسب تعداد واحدهای به‌کار رفته در شبکه

۶- نتیجه‌گیری و کارهای آینده

شبکه‌های مبتنی بر نرم‌افزار به‌واسطه امکانات نرم‌افزاری که در جهت تنظیم شبکه در اختیار کاربران قرار داده‌اند به‌طور چشم‌گیری مورد توجه قرارگرفته‌اند اما این قابلیت که در این شبکه‌ها توسط جدول‌های جریان به‌دست می‌آید، می‌تواند علت بروز خرابی‌هایی در شبکه شود که به مراتب اهمیت بیشتری نسبت به شبکه‌های سنتی دارد. وجود ناهماهنگی در بین قوانین موجود در جدول‌های جریان یکی از این خرابی‌هاست که در این مقاله سعی شد با توجه به روش‌های پیشین سرعت الگوریتم تشخیص آن را بهبود بخشیده شود.

به‌عنوان اهداف این تحقیق در آینده باید به دنبال راهی به‌منظور بهبود هر چه دقیق‌تر مقدار حد آستانه بوده تا بتوان در مدت زمان کمتری قوانین ناهماهنگ را درون جدول‌های جریان شبکه‌های مبتنی بر نرم‌افزار تشخیص داد.

۷- مراجع

[1] Y. Wang and I. Matta, "Sdn management layer: Design requirements and future direction," in Network Protocols (ICNP), IEEE 22nd International Conference on, 2014.

[2] M. Ghasabi, M. Deypir, and E. Mahdipour, "A New Algorithm Based on Hellinger Distance for Mitigation of DDoS Attacks in Software Defined

www.SID.ir

Archive of SID

Networks," Journal of Electronical & Cyber Defence, vol. 5, pp. 29-41, 2017. (In Persian)

[3] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of

[4] software-defined networking: Past, present, and future of programmable networks," IEEE Communications Surveys & Tutorials, vol. 16, pp. 1617-1634, 2014.

[5] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," IEEE Communications Surveys & Tutorials, vol. 16, pp. 1955-1980, 2014.

[6] H. Li, F. Wei, and H. Hu, "Enabling Dynamic Network Access Control with Anomaly-based IDS and SDN," In Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2019.

[7] M. H. Khairi, S. H. Ariffin, N. A. Latiff, A. S. Abdullah, and M. K. Hassan, "A Review of Anomaly Detection Techniques and Distributed Denial of Service (DDoS) on Software Defined Network (SDN)," Engineering, Technology & Applied Science Research, vol. 8, pp. 2724-2730, 2018.

[8] P. Wang, L. Huang, H. Xu, B. Leng, and H. Guo, "Rule Anomalies Detecting and Resolving for Software Defined Networks," in IEEE Global Communications Conference (GLOBECOM), 2015.

[9] P. Zhang, S. Xu, Z. Yang, H., Li, H. Wang, and C. Hu, "FOCES: Detecting Forwarding Anomalies in Software Defined Networks," In 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 2018.

[10] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in Proceedings of the first workshop on Hot topics in software defined networks, 2012.

[11] A. Khurshid, W. Zhou, M. Caesar, and P. Godfrey, "Veriflow: Verifying network-wide invariants in real time," ACM Sigcomm Computer Communication Review, vol. 42, pp. 467-472, 2012.

[12] A. AuYoung et al., "Democratic resolution of resource conflicts between sdn control programs," in Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, 2014.

[13] E. Al-Shaer and S. Al-Haj, "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures," in Proceedings of the 3rd ACM workshop on Assurable and usable security configuration, 2010.

[14] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Hierarchical policies for software defined networks," in Proceedings of the first workshop on Hot topics in software defined networks, 2012.

[15] G. Zhang, S. Cheng, X. Song, and F. Jiang, "Detecting and Resolving Flow Entries Collisions in Software Defined Networks," In Proceedings of the 2019 3rd International Conference on Computer Science and Artificial Intelligence, 2019.

[16] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan, "Conflict classification and analysis of distributed firewall policies," IEEE journal on selected areas in communications, vol. 23, pp. 2069-2084, 2005.

[17] Q. Shafi, A. Basit, S. Qaisar, A. Koay, and I. Welch, "Fog-Assisted SDN Controlled Framework for Enduring Anomaly Detection in an IoT Network," IEEE Access, vol. 6, pp. 73713-73723, 2018.

www.SID.ir

Archive of SID

Journal of Electronical & Cyber Defence
Vol. 8, No. 4, 2021, Serial No. 32

An Accelerated Method for Rules Anomaly Detection in Software Defined Networks

R. Kiani, A. Bohlooli*

*Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.

(Received: 26/09/2019, Accepted: 05/08/2020)

ABSTRACT

Software defined networks have attracted enormous attention because they simplify the process of setting up the network. They have been able to leave behind in a short time, most of the technologies that were used in traditional networks by industrialists and researchers. The ease and efficiency are due to the separation of control and data planes from each other such that the control plane is a logically centralized controller and the data plane switches in as the flow table has been implemented. In these networks, network topology adjustment is done using a flow table that has special flow rules and network services, such as Qos, security and etc., which operate as programs on the network. Flow tables can be directly or indirectly changed by any of these services in the network. Although access to the table of current units simplifies network configurations, it could lead to anomalies between flow rules for separate modules. In addition to consuming too much switch memory, these anomalies can cause problems for network security and applications. Fortunately, so far, some researches on the detection of anomalies in flow tables of software defined networks has been done but this method is not only imposing a great deal of time and processing on the controller, in some cases only conflict resolution has been performed. In this paper we have shown how to speed up the detection algorithm and then tried to improve the speed of anomaly detection algorithm in the flow table of switches in the software defined networks using different variables.

Keywords: Software Defined Network, Anomaly Detection, Flow Tables, Flow Rules

* Corresponding Author Email: bohlooli@eng.ui.ac.ir

www.SID.ir