Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 10, Number 3, Autumn 2021, pp. 99-110
Received: 29/06/2020, Revised: 24/04/2021, Accepted: 02/07/2021

A Real-time Method to Detect Compromised Devices in Software Defined Networks

Mojtaba Bagha[1] and Yousef Darmani[2*]
1- Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, IRAN.
2*- Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, IRAN.
[1] mojtababagha@email.kntu.ac.ir and [2*]darmani@kntu.ac.ir

Corresponding author's address: Yousef Darmani, Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, IRAN

Abstract- Software-defined networking technology is an approach in computer networks that enables network administrators to easily manage the network at a lower cost. This is done by separating the control layer from the data layer. This process makes new vulnerabilities in switches, controllers and communication protocols between them that did not exist in traditional networks. This article provides a simple and inexpensive way to identify endangered network equipment, especially switches and controllers, even when both are compromised. This method is based on periodic equipment validation with the help of a backup controller. To validate the switches, the incorrect forwarding attack model is used and the routing module is checked in the controller. The simulation results show that this method can detect endangered equipment with very low time and processing overhead.

Keywords- Software-Defined Network, Compromised SDN Devices, Security of SDN.

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۱۰، شماره ۳، پاییز ۱۴۰۰، صفحه ۹۹-۱۱۰
دریافت: ۱۳۹۹/۰۴/۰۹، بازنگری: ۱۴۰۰/۰۲/۰۴، پذیرش: ۱۴۰۰/۰۴/۱۱

روشی برای شناسایی بی‌درنگ تجهیزات به خطر افتاده در شبکه‌های نرم افزار محور

مجتبی بقا[۱]، یوسف درمانی*[۲]
۱- دانشکده مهندسی کامپیوتر، *دانشگاه صنعتی خواجه نصیرالدین طوسی*، تهران، ایران.
*۲- دانشکده مهندسی کامپیوتر، *دانشگاه صنعتی خواجه نصیرالدین طوسی*، تهران، ایران.
[1]mojtababagha@email.kntu.ac.ir, [2*]darmani@kntu.ac.ir,

* نشانی نویسنده مسئول: یوسف درمانی، تهران، سیدخندان، دانشگاه صنعتی خواجه نصیرالدین طوسی، دانشکده مهندسی کامپیوتر.

چکیده- شبکه‌های نرم افزار محور یا SDN رویکردی در شبکه‌های کامپیوتری است که مدیران شبکه را قادر به مدیریت آسان شبکه با صرف زمان و هزینه کمتری می‌نماید. این امر از طریق جداسازی لایه کنترل از لایه داده انجام می‌شود. این روند موجب بروز آسیب پذیری‌های جدیدی در سوئیچ‌ها، کنترلرها و پروتکل‌های ارتباطی بین آنها می‌شود که پیش از این در شبکه‌های سنتی شاهد آن نبودیم. در این مقاله روشی ساده و کم هزینه برای شناسایی تجهیزات به خطر افتاده شبکه خصوصا سوئیچ و کنترلر، حتی هنگامیکه هر دو همزمان به خطر افتاده اند ارائه شده است. این روش بر مبنای صحت سنجی تجهیزات به صورت دوره ای و به کمک کنترلر پشتیبان انجام می‌گیرد. برای صحت سنجی سوئیچ‌ها از مدل حمله هدایت نادرست استفاده شده و در کنترلر ماژول مسیریابی مورد صحت سنجی قرار می‌گیرد. نتایج حاصل از شبیه سازی نشان می‌دهد این روش می‌تواند با سربار زمانی و پردازشی بسیار کم، تجهیزات به خطر افتاده را شناسایی کند.

واژه‌های کلیدی: شبکه‌های نرم افزار محور – شناسایی تجهیزات به خطر افتاده- امنیت شبکه‌های SDN

۱- مقدمه

در شبکه‌های سنتی هریک از تجهیزات شبکه شامل سوئیچ‌ها و مسیریاب‌ها دارای نرم افزار تصمیم گیری مستقلی هستند که شامل لایه کنترل[۱] و داده[۲] بوده و به کمک آن در خصوص هدایت[۳] ترافیک تصمیم می‌گیرند. در این نوع شبکه هر دو لایه کنترل و داده در دستگاه‌های متصل شده در سراسر شبکه یکپارچه است و در نتیجه هر گونه تغییر در پیکربندی شبکه و استراتژی مسیریابی نیاز به اصلاح در دستگاه‌های شبکه بوده و مستلزم صرف هزینه و زمان بالا می‌باشد[۱].

بر خلاف شبکه سنتی، SDN[4] یک معماری چابک و انعطاف پذیر شبکه است که پاسخی سریع و موثر به هر تغییر توسط مدیر شبکه را ممکن می‌سازد. روش SDN لایه کنترل شبکه را از لایه داده جدا می‌کند و در آن کنترل‌کننده تصمیم می‌گیرد که ترافیک چگونه و کجا ارسال شود و لایه داده، که به عنوان سطح روانه سازی[۵] نیز شناخته می‌شود، مسئول هدایت ترافیک به گام بعدی می‌باشد. بدین ترتیب لایه کنترل نمایی کلی از شبکه دارد و مدیریتی موثر و سریع برای هر رویداد یا تغییر در شبکه ایجاد می‌نماید. با این حال، قابلیت اطمینان، در دسترس بودن و امنیت شبکه همچنان مسائل مهمی هستند که باید در نظر گرفته شوند. با عنایت به اینکه پروتکل OpenFlow واسطی برای ارتباط لایه داده با لایه کنترل است، اکثر شبکه‌های SDN با این پروتکل کار می‌کنند.

یکی از بحث‌های مهم SDN که به عنوان یک چالش بزرگ آن مطرح است، امنیت می‌باشد. بهبود هوشمندی نرم‌افزار کنترلر ممکن است باعث افزایش آسیب پذیری کنترلر شود و

حمله‌های هکرها را در پی داشته باشد. خطرات امنیتی شبکه‌های SDN شامل آسیب پذیری‌های کنترلر، سوئیچ‌ها، برنامه‌های کاربردی و غیره می‌باشد. یکی از رایج‌ترین حمله‌هایی که چه در شبکه‌های سنتی و چه در شبکه‌های SDN وجود دارد، حمله منع سرویس[۶] می‌باشد. راه حل‌های مختلفی که در این زمینه برای دفاع و جلوگیری از مختل شدن کنترلر SDN ارائه شده است که می‌توان به مقاله‌های [۲,۳,۴] اشاره کرد.

کنترلر SDN به طور معمول هدف اصلی مهاجمین است زیرا یک نقطه مرکزی برای تصمیم گیری در شبکه است و می‌تواند تک نقطه اصلی شکست تلقی شود. تا زمانی که در شبکه از یک کنترلر واحد در لایه کنترلر استفاده می‌شود همیشه خطر از کار افتادن کل شبکه به دلایل مختلف امنیتی و عملکردی وجود دارد. برای برطرف نمودن این مشکل معماری کنترلر توزیعی معرفی شده است که به وسیله این معماری بسیاری از مسائل و مشکل‌های ناشی از استفاده از یک کنترلر واحد در لایه کنترل حل می‌شود که نمونه‌هایی از آن در مقاله‌های [۵,۶,۷,۸] شرح داده شده است. استفاده از معماری کنترلر توزیعی نیز مسائل امنیتی خاص خود را به همراه دارد و سرویس‌های امنیتی پیش‌فرض نمی‌توانند به تنهایی ضامن امنیت شبکه و تجهیزات شبکه شوند. انواع مختلفی از سرویس‌های امنیتی با استفاده از برنامه‌های امنیتی مختلف در لایه کاربرد می‌تواند ارائه شود که در برخی موارد به خاطر سادگی همه آنها در یک پلت فرم واحد قرار داده می‌شوند یا به عنوان یک کنترلر مجزای امنیتی در کنار کنترلر اصلی وظیفه خود را انجام می‌دهد. برخی از این سرویس‌های امنیتی در مقاله‌های [۹,۱۰,۱۱] معرفی شده‌اند.

چنانچه هکرها بتوانند از لایه‌های امنیتی عبور کنند و به کنترلر دسترسی پیدا کنند، قادر خواهند بود از جنبه‌های مختلفی به شبکه آسیب برسانند و حتی ممکن است کل شبکه را از کار بیندازند. همچنین اگر سوئیچ‌ها با توجه به قوانین برنامه ریزی شده خود عمل نکنند، مشکلات حادی برای شبکه به وجود می‌آورند. به عنوان مثال مدیر شبکه یک سوئیچ را طوری برنامه ریزی کرده است که برخی از بسته‌هایی که از پورت ۱ وارد سوئیچ می‌شوند به پورت خروجی ۲ هدایت شوند. اگر مهاجم بتواند به سوئیچ دسترسی پیدا کند یا اینکه سوئیچ دچار خطای نرم افزاری یا سخت افزاری شده باشد، آنگاه ممکن است آن بسته به جای این که به پورت ۲ هدایت شود، به پورت دیگری فرستاده شود یا ممکن است آن بسته را به هیچ کدام از پورت‌های خروجی هدایت نشود. برای جلوگیری از چنین حمله‌هایی نیاز است تا مکانیزم و روشی وجود داشته باشد که به کمک آن بتوان چنین ناهنجاری‌هایی را در سطح شبکه SDN تشخیص داد و تجهیزات شبکه (کنترل و سوئیچ) به خطر افتاده را شناسایی کرد.

۱-۱- کارهای پیشین

در زمینه شناسایی تجهیزات به خطر افتاده و ناهنجاری‌های شبکه چندین مقاله تا بحال ارائه شده است. در مقاله [۱۲] برای شناسایی ناهنجاری‌های شبکه به صورت بلادرنگ و با تاخیر جزئی، روشی به نام Veriflow معرفی شده است که به عنوان یک لایه مجزا بین کنترلر و دستگاه‌های شبکه قرار می‌گیرد و هرگونه قانون هدایت را به صورت پویا بررسی می‌کند. این مکانیزم برای اضافه یا حذف کردن هر قانون، تاثیر آن را بر روی کل شبکه با سرعت بالایی بررسی و اعتبار سنجی می‌کند. هر یک از سوئیچ‌های OpenFlow باید بگونه ای پیکر بندی شوند تا VeriFlow را به عنوان کنترلر خود بشناسند. در مقاله [۱۳] روشی به نام Sphinx پیشنهاد شده است که می‌تواند حمله‌های شناخته شده یا ناشناخته را در لایه داده و تجهیزات شبکه شناسایی کند. این روش از مفهوم گراف جریان[۷] استفاده می‌کند تا بتواند به صورت افزایشی عملیات به روز رسانی شبکه را به صوت بلادرنگ اعتبار سنجی کند. Sphinx به صورت پویا رفتارهای جدید شبکه را یاد میگیرد و اگر تغییرات مشکوکی در لایه کنترل مشاهده کند پیغام هشدار صادر می‌کند. این روش چهار نوع از پیام‌های کنترلی پروتکل OpenFlow یعنی FEATURES_REPLY, PACKET_IN, FLOW_MOD, STATS_REPLY را آنالیز می‌کند تا هم بتواند رفتار جدید شبکه را فراگیرد و هم به صورت افزایشی گراف جریان را تهیه کند.

یک سوئیچ به خطر افتاده یا ابزاری برای حمله به کنترلر می‌شود یا با رفتارهای مخرب خود باعث اخلال در لایه داده شبکه می‌شود. تزریق بسته‌های جعلی درون شبکه که به وسیله سوئیچ‌های به خطر افتاد انجام می‌گیرد، باعث خراب شدن دید کنترلر نسبت به شبکه، تحمیل بار کاری اضافه به کنترلر و نصب قوانین بیهوده بر روی سوئیچ‌ها می‌شود. در این نوع از حملات، سوئیچ ابزاری جهت حمله به کنترلر می‌شود. در مقاله [14] روشی برای مقابله با این حمله با استفاده از یک قطعه سخت‌افزاری به نام Inspector ارائه شده است. بسته های packet-in قبل از اینکه به کنترلر برسند ابتدا به Inspector می‌روند و در آنجا فیلد های سرایند بسته بررسی می‌شود تا مشخص شود که این بسته از طرف سوئیچی که قبلا

اعتبارسنجی شده است آمده است یا خیر. اگر بسته جعلی باشد دور ریخته میشود در غیر اینصورت بسته به سمت کنترلر روانه می‌شود.

در مقاله [15] مدل حمله، رفتارهای مخرب سوئیچ مثل دورانداختن بسته، دست‌کاری بسته یا تاخیر در پردازش بسته می‌باشد. روش ارائه شده برای مقابله با این حمله بر اساس تحلیل بی نظمی ترافیک شبکه بر اساس پارامتر تعداد بسته‌های ورودی به سوئیچ در زمان‌های مختلف می‌باشد. در نهایت مشخص می‌شود که چند درصد از ترافیک شبکه نرمال و چند درصد غیر نرمال می‌باشد. در این روش شناسایی ترافیک مخرب از ترافیک انفجاری مساله مهمی است که باعث می‌شود درصد خطای این الگوریتم بالا باشد. به همین منظور نویسندگان این مقاله در مقاله بعدی خود [16] در راستای شناسایی ترافیک مخرب، با بهره گیری از شبکه‌های عصبی و یادگیری عمیقعملکرد الگوریتم را بهبود بخشیدند. با این وجود در هر دو مقاله فرض بر این است که کنترلر یک موجودیت قابل اعتماد است که می‌تواند ترافیک مخرب را شناسایی کند. در حالی که ممکن است کنترلر خود نیز به خطر افتاده باشد.

در مقاله [17] برخی از مدل‌های حمله از طریق سوییچ‌های به خطر افتاده بیان شده اند و روشی برای شناسایی این دستگاه‌های به خطر افتاده طراحی شده است. روش شناسایی به این صورت است که برخی از قواعد جریان سوئیچ‌هایی که به طور تصادفی انتخاب می‌شوند، زیر نظر گرفته می‌شود و بسته‌هایی ساخته می‌شود که با این قوانین مطابقت داشته باشند. سپس مسیر حرکت بسته زیر نظر گرفته می‌شود که آیا مطابق برنامه ریزی حرکت می‌کند یا خیر. سوئیچ‌ها و قواعد جریان تحت آزمایش به صورت تصادفی انتخاب می‌شوند و همین موضوع باعث می‌شود تا امکان وجود سوئیچ به خطر افتاده ای که شناسایی نشده باشد وجود داشته باشد.

در مقاله [18] یک روش جامع و کامل برای شناسایی تجهیزات به خطر افتاده شبکه به صورت بلادرنگ ارائه شده است که تفاوت اصلی این مقاله با سایرین در این است که در مقاله‌های پیشین همیشه فرض بر این بود که کنترلر قابل اعتماد است و مورد حمله قرار نمی‌گیرد یا دچار عیب و نقص نمی‌شود، اما در این مقاله روش ارائه شده بر این فرض مبتنی است که سوئیچ و کنترلر می‌توانند همزمان دچار مشکل شوند و این مشکل ممکن است به گونه‌ای باشد که به راحتی نتوان آن را تشخیص داد. این روش به صورت بلادرنگ رخدادی که در شبکه اجرا و باعث به روزرسانی شبکه می‌شود را پیگیری می‌کند و از نتیجه اجرای آن به کمک کنترلرهای پشتیبان می‌توان تشخیص داد که کدام تجهیز در شبکه به درستی عمل می‌کند و کدام به درستی عمل نمی‌کند و رفتاری غیر طبیعی را بروز می‌دهد. این روش نسبت به مقاله‌های دیگر کامل‌تر است ولی پیچیدگی الگوریتم و تعداد پیام‌های جا به جا شده در آن زیاد است.

در مقاله حاضر روش ساده و کم هزینه ای ارائه شده که به کمک آن می‌توان تجهیزات به خطر افتاده شبکه، اعم از سوئیچ و کنترلر را حتی زمانی که هر دوی آنها به صورت همزمان به خطر افتاده اند را شناسایی نمود. این روش بر مبنای صحت سنجی تجهیزات به صورت دوره ای و به کمک کنترلر پشتیبان انجام می‌گیرد. مزیت این روش سادگی و سربار بسیار اندک آن است که با کمترین تعداد جا بجایی بسته می‌تواند تجهیزات به خطر افتاده را شناسایی کند.

۲- شناسایی تجهیز به خطر افتاده

در این بخش ابتدا مدل حمله و تهدیدهای امنیتی بالقوه ناشی از کنترلرها و سوئیچ‌های SDN به خطر افتاده و مسأله شناسایی این دستگاه‌ها و همچنین روش پیشنهادی مورد بحث قرار خواهد گرفت.

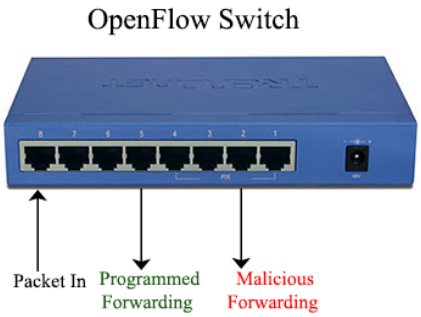
۲-۱- تجهیز به خطر افتاده

مطابق معماری SDN تمامی سوئیچ‌هایی که تحت فرمان کنترلر هستند باید از دستورات کنترلر و مدیر شبکه پیروی کنند. یعنی هنگام ورود بسته، سوئیچ باید مطابق جدول جریانی که از قبل توسط کنترلر ارائه شده عمل کند. زمانی که یک سوئیچ از دستورات برنامه ریزی شده پیروی نکند، به خطر افتاده تلقی میشود. همچنین اگر کنترلر وظایف خود را به درستی انجام ندهد نیز به خطر افتاده قلمداد میشود و باید مورد بازبینی قرار گیرد.

۲-۲- مدل حمله در سوئیچ

فرض کنید مهاجم به یک سوئیچ حمله کرده و کنترل آن را به دست گرفته است. در اینصورت او می‌تواند جدول جریان سوئیچ را مطابق خواسته خود دستکاری کند. جدا از این حالت، سوئیچ به دلایلی دیگر از جمله خطای سخت افزاری ممکن است نتواند بسته ورودی را به پورت خروجی هدایت کند یا با تاخیر زیاد این کار را انجام دهد. حتی ممکن است اشتباها بسته را به پورت دیگری هدایت کند. در این حالت یک بسته با آدرس مبدا

و مقصد مشخص هنگامی که به سوئیچ به خطر افتاده می‌رسد، یا دیگر نمی‌تواند به مسیر خود ادامه دهد یا اینکه به مسیر نادرست هدایت می‌شود و ممکن است هرگز به مقصد نرسد. شکل ۱ نشانگر عمل هدایت نادرست[۸] در سوئیچ SDN است.



شکل ۱- هدایت نادرست در سوئیچ SDN

۲-۳- مدل حمله در کنترلر

فرض می‌کنیم مهاجم کنترلر را هدف حمله خود قرار داده است. آنگاه مهاجم قادر خواهد بود تا کنترل تمام شبکه را در دست گیرد و خسارت زیادی را به شبکه تحمیل کند. مشکلاتی که مهاجم می‌تواند برای شبکه به وجود آورد یا از نوع Active (فعال)، مانند پاک کردن تمام قواعد جداول جریان سوئیچ‌ها یا خاموش کردن سوئیچ‌ها خواهند بود و یا از نوع Passive (غیر فعال)، مانند شنود شبکه و جمع آوری اطلاعات خواهند بود. شناسایی حمله‌های فعال کار دشواری نیست و می‌توان آنها را بسادگی تشخیص داد، اما شناسایی حمله‌های غیر فعال بسیار دشوار خواهد بود و حتی ممکن است مدیر شبکه هیچگاه از وقوع چنین حمله‌هایی مطلع نشود، زیرا در این حالت کنترلر وظایف خود را انجام می‌دهد و سرویس دهی به شبکه همچنان ادامه دارد. در این مقاله فرض بر این است که مهاجم ماژول مسیریابی کنترلر را به گونه‌ای دستکاری کرده است که کنترلر مسیر بهینه را محاسبه نمی‌کند اما بسته را به مقصد می‌رساند. این حمله از نوع غیر فعال محسوب می‌شود و شناسایی آن به راحتی ممکن نیست. بدیهی است که حملات مختلف دیگری ممکن است علیه کنترلر اجرا شود اما در این مقاله فقط ماژول مسیریابی کنترلر مورد صحت سنجی قرار می‌گیرد و فرض بر این است که بقیه ماژول‌های کنترلر به درستی کار می‌کنند.

۲-۴- مکانیزم تشخیص

روش ارائه شده به این صورت است که شبکه ایجاد شده متشکل از سوئیچ‌هایی است که هر کدام به یک کنترلر اصلی و یک یا چند کنترلر پشتیبان متصل هستند. هدف، ارزیابی صحت عملکرد برخی از سوئیچ‌های شبکه در عمل هدایت و صحت عملکرد کنترلر در مسیریابی می‌باشد. ابتدا با توجه به توپولوژی شبکه دو host انتخاب می‌شود تا بسته ای از یک host به دیگری ارسال شود. بسته در host مبدا ساخته می‌شود و سپس به اولین سوئیچ متصل به خود می‌رسد. سوئیچ در جدول جریان خود به دنبال قاعده جریانی می‌گردد که منطبق بر بسته باشد. این اتفاق صورت نمی‌گیرد و سوئیچ بسته را به کنترلرهای متصل به خود یعنی کنترلر اصلی و کنترلرهای پشتیبان ارسال می‌کند. کنترلر اصلی کوتاه‌ترین مسیر از مبدا به مقصد را محاسبه می‌کند و منتظر می‌ماند تا بقیه کنترلرهای پشتیبان نیز مسیری که محاسبه کرده اند را برایش ارسال کنند. از آنجا که احتمال حمله موفق همزمان بر روی اکثر کنترل کننده‌ها نسبتا کم است، مسیرهای دریافتی یکسان از اکثریت کنترل کننده‌ها به عنوان مسیر امن در نظر گرفته می‌شود. اگر تعداد مسیرهای یکسان دریافتی از $\left\lceil \frac{N+1}{2} \right\rceil$ کمتر باشد امکان به دست آوردن مسیر امن وجود ندارد و اکثریت کنترلرهای پشتیبان به خطر افتاده اند و پیغام هشدار صادر می‌شود. در غیر این صورت کنترلر اصلی مسیر خود را با مسیر امن مقایسه می‌کند. اگر مسیر محاسبه شده توسط کنترلر اصلی با مسیر امن یکسان بود، یعنی کنترلر اصلی درست کار می‌کند در غیر این صورت کنترلر اصلی مشکوک است و پیغام هشداری برای مدیر شبکه ارسال می‌شود. بنابراین این الگوریتم در صورتی می‌تواند مورد حمله موفق قرار گیرد که $\left\lceil \frac{N+1}{2} \right\rceil$ کنترلر مورد حمله همزمان قرار بگیرند که در آن N تعداد کل کنترلرهای پشتیبان است.

سپس کنترلر یک قاعده جریان در سوئیچی که درخواست مسیر داده بود نصب می‌کند و با توجه به مسیر محاسبه شده، در یک بازه زمانی مشخص منتظر درخواست سوئیچ بعدی می‌ماند. اگر سوئیچ بعدی درخواستش را طی مهلت زمانی مشخص ارسال نکرد، پیغامی به مدیر شبکه مبنی بر تاخیر عملکرد در سوئیچ مربوطه داده می‌شود. اگر سوئیچ قبل از آستانه زمانی درخواستش را به کنترلر ارسال کرد، کنترلر بررسی می‌کند که آیا درخواست از همان سوئیچ مورد انتظار فرستاده شده است یا خیر. اگر جواب منفی بود پیام هشداری برای مدیر شبکه مبنی بر عدم عملکرد صحیح سوئیچ قبلی در هدایت بسته به پورت

خروجی صحیح ارسال می‌شود. در غیر این صورت کنترلر یک قاعده جریان در سوئیچ درخواست دهنده نصب می‌کند و گام بعدی را برای سوئیچ مشخص می‌کند. سپس منتظر درخواست بعدی از سوئیچ بعدی می‌ماند و این چرخه تا زمانی که بسته تمامی سوئیچ‌های مسیر را طی کند ادامه می‌یابد. هرگاه درخواستی از طرف سوئیچی بیاید که طبق مسیر محاسبه شده و به ترتیب صحیح نباشد، یعنی سوئیچ قبل از آن به درستی عمل نمی‌کند و به خطر افتاده است.در ابتدای الگوریتم تایمر مربوط به دریافت بسته از طرف سوئیچ بعدی مورد انتظار، عددی بزرگ قرار میگیرد تا الگوریتم در اجرای دور اول و بررسی تایمر، خطای منقضی شدن تایمر را ندهد. سپس این تایمر به زمان مورد انتظار برای دریافت بسته بعدی تنظیم می‌شود.

به طور خلاصه الگوریتم پیشنهادی به این صورت است که بسته ای تولید می‌شود که هیچ کدام از سوئیچ‌ها قاعده ای منطبق بر آن در جداول جریان خود ندارند و باید برای هدایت بسته از کنترلر درخواست مسیریابی کنند. درخواست‌های مورد انتظار، تایید و پاسخ داده می‌شوند اما درخواست‌های نامعتبر به اطلاع مدیر شبکه می‌رسند. بدین ترتیب سوئیچ به خطر افتاده شناسایی می‌شود. همچنین با مقایسه مسیر محاسبه شده با مسیرهایی که کنترلرهای پشتیبان ارسال کرده اند می‌توان

الگوریتم ۱ روش شناسایی تجهیزات به خطر افتاده

```
1: p=[]
2: SentStatus=0
3: Timer=time.time()+31556926 #one yearlater
4: if Request is  type of Verification then
5:   if len(p)==0 then
6:       Timer = time.time()+31556926
7:       P = CalculatePath()
8:       SentPath=GetPathFromOtherControllers()
9:       if SentPath equal P then
10:         Alarm: Master Contrller work Correctly
11:      else
12:          Alarm:The Master Controller is Suspicious
13:      end if
14:  end if
15: if time.Now < Timer then
16:    if switchId equal P[0] then
17:      InstallPath for this switchId
18:      Delete p[0]
19:      Timer = Now+1000
20:    else
21:      Alarm:switch p[0] must come first
22:    end if
23: else
24:    Alarm: switch p[0] Not responding
25: end if
26: end if
```

صحت عملکرد کنترلر را نیز بررسی کرد. فلوچارت روش پیشنهادی در شکل ۲ و الگوریتم آن در زیر ارائه شده است.

۲-۵- پیاده سازی سیستمی

در این مقاله برای پیاه سازی روش پیشنهادی، از شبیه ساز Mininet ورژن ۲.۳ و پروتکل OpenFlow ورژن ۱.۳ استفاده شده است. کنترلر اصلی به همراه توپولوژی شبکه بر روی یک ماشین مجازی با تعداد چهار هسته و ۳.۵ GB حافظه RAM و کنترلر پشتیبان بر روی ماشین مجازی دیگر با دو هسته و ۲.۵ GB حافظه RAM پیاده سازی شده اند. هر دوی این ماشین‌های مجازی بر روی یک سیستم با CPU به مشخصات Intel Core i7 2.2 GHz و ظرفیت 8 GB حافظه RAM نصب شده اند. توپولوژی مورد استفاده در این آزمایش همانند شکل ۳ می‌باشد. با این تفسیر که هر سوئیچ به دو کنترلر اصلی و پشتیبان متصل است. برای جلوگیری از پیچیده شدن شکل، کنترلرها در شکل رسم نشده اند.

۲-۵-۱- کنترلر ریو

کنترلر Ryu با ارائه ویژگی‌های مناسب، انتخاب خوبی برای کارهای کوچک و برنامه‌های تحقیقاتی است. این کنترلر با زبان Python نوشته شده است و امکاناتی را برای توسعه برنامه‌ها و ماژول‌ها به این زبان ارائه می‌دهد. با این حال، عدم ماژولاریتی بالا و چند سکویی[۹] نبودن آن، استفاده گسترده از آن در بازار واقعی را محدود می‌کند [۱۶]. Ryu به طور کامل از پروتکل OpenFlow نسخه‌های ۱.۰، ۱.۲، ۱.۳، ۱.۴، ۱.۵ پشتیبانی می‌کند. در این مقاله الگوریتم شناسایی در قالب دو برنامه بر روی پلتفرم Ryu توسعه داده شده است.

۲-۵-۲- نصب قوانین در سوئیچ‌ها

در تمامی کنترلر های اس-دی-ان در هنگام راه اندازی شبکه یک قاعده جریان پیش‌فرض با کمترین اولویت بر روی همه سوئیچ‌ها (در زمان معرفی سوئیچ به کنترلر) نصب می‌شود که به سوئیچ دستور می‌دهد تمامی بسته‌های دریافتی را به سمت کنترل‌کننده ارسال نماید. بدین ترتیب زمانی که بسته ای به سوئیچ می‌رسد، اگر با هیچ یک از قوانین جداول جریان مطابقت نداشته باشد، به سمت کنترلر ارسال می‌شود و کنترلر قاعده جریان متناسب با درخواست آن را بر روی سوئیچ نصب می‌کند.

در برنامه این مقاله، از آنجا که قوانینی که کنترلر اصلی بر روی سوئیچ‌های درخواست دهنده نصب می‌کنند، صرفا جنبه ارزیابی

عملکرد سوئیچ‌ها را دارد، نباید برای مدت طولانی بر روی سوئیچ‌ها باقی بمانند، زیرا ممکن است در اجرای بعدی از برنامه دقیقا همان بسته به همان سوئیچ برسد و سوئیچ درخواستی را به کنترلر ارسال نکند چرا که از قبل قاعده ای برای هدایت این جریان در جدول جریان خود دارد. برای این که اطمینان حاصل شود که در هر بار اجرای برنامه، الگوریتم به درستی اجرا می‌شود، باید فیلد Hard timeout در قوانین جریان برابر با مدت زمانی معین مثلا مدت زمان اجرای دوباره برنامه یا کمتر از آن تعیین شود و فیلد Idle timeout برابر صفر قرار گیرد. در این صورت قوانین جریان پس از سپری کردن زمان Hard timeout به صورت خودکار از جداول جریان پاک می‌شوند.

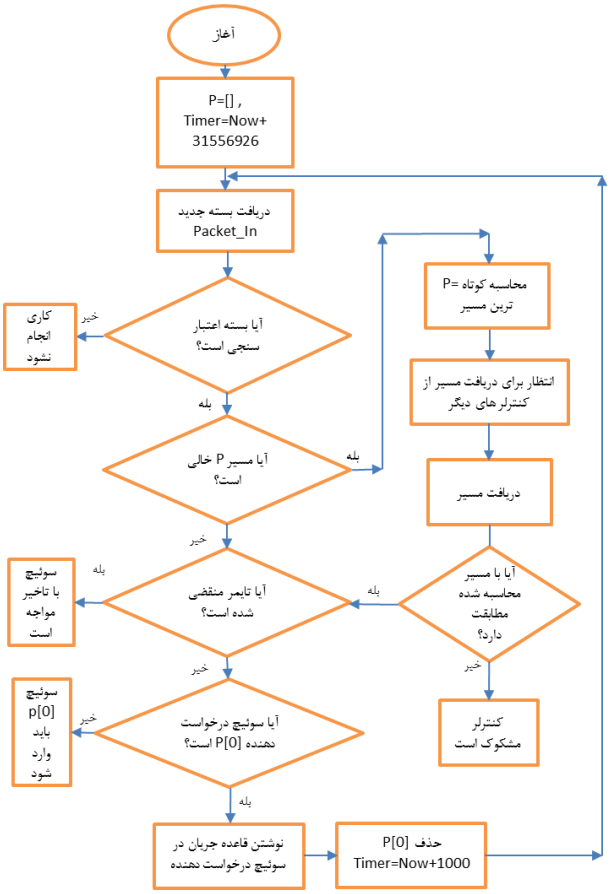
۲-۵-۳- ساخت بسته ابتدایی

قبل از شروع الگوریتم باید بسته ای تولید شود که دارای حداقل فیلدهای آدرس MAC مبدا و مقصد باشد و نیز باید از بین بقیه فیلدهای اختیاری آن، مقادیری انتخاب شوند که در نهایت بسته ای تولید شود که در کل شبکه قاعده جریانی با آن مطابقت نداشته باشد. ساخت چنین بسته ای در شبکه منعطف SDN کار دشواری نخواهد بود، زیرا کنترلر تمام اطلاعات شبکه را در اختیار دارد و به راحتی می‌تواند یک فیلد استفاده نشده را انتخاب و بسته منحصر به فرد را تولید کند. به عنوان مثال با قرار دادن فیلد VLAN_VID=2 می‌توان به این امر دست یافت [۱۴].

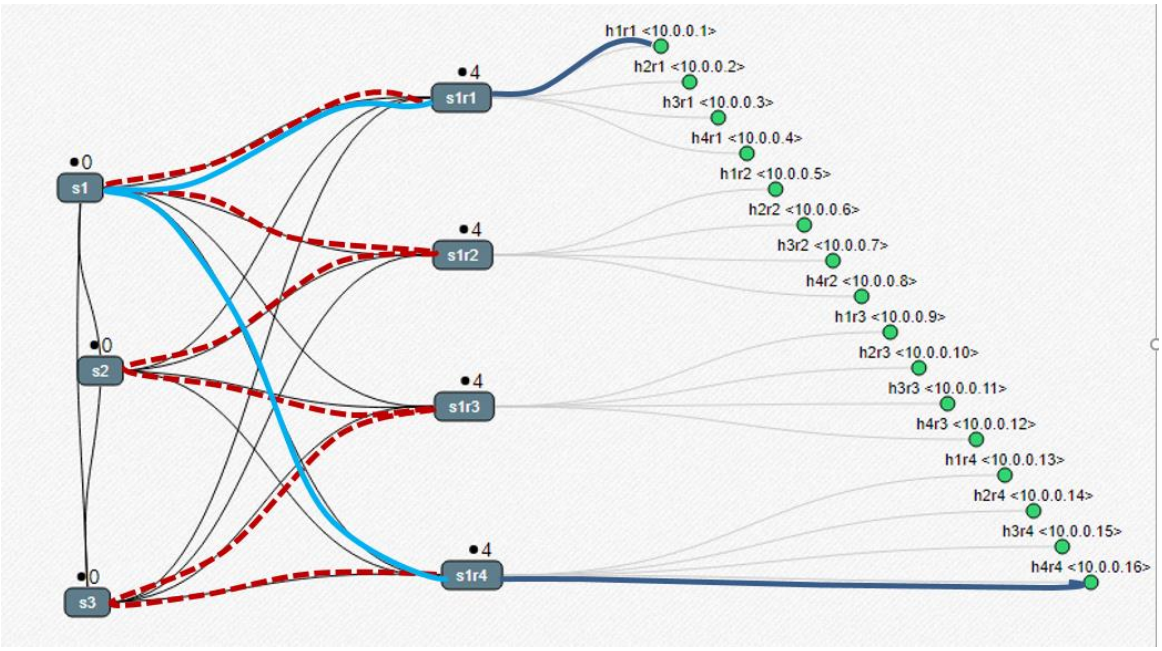
۲-۵-۴- توپولوژی شبکه

در برخی از توپولوژی‌های شبکه معمولا چندین سوئیچ، زیر مجموعه یک سوئیچ بالاسری[۱۰] (ریشه) قرار می‌گیرند. در یک مرکز داده[۱۱] واقعی معمولا بیش از یک سوئیچ ریشه وجود دارد که این سوئیچ‌های ریشه در یک الگوی حلقه[۱۲] به یکدیگر متصل هستند. این کار باعث می‌شود تا هنگام خرابی یکی از سوییچ‌های ریشه، کل شبکه مختل نشود. با این حال، این توپولوژی مشکل حلقه ایجاد می‌کند. اگر کنترلر توانایی مدیریت حلقه‌ها را نداشته باشد، سیلی از بسته‌ها به سمت کنترلر روانه می‌شود و باعث مختل شدن شبکه می‌شود. کنترلر Ryu برای مدیریت حلقه‌ها کتابخانه ای به نام stplib را معرفی کرده است که بر اساس پروتکل درخت پوشا، توپولوژی را مدیریت می‌کند [20]. در پیاده سازی روش ارائه شده در این مقاله نیز از این کتابخانه استفاده شده است که قابلیت سازگاری با انواع توپولوژی‌های شبکه از جمله (Bus, Ring, Star, Mesh, …) را دارا می‌باشد. شکل ۳ توپولوژی مورد استفاده در این مقاله را نشان می‌دهد.

در این توپولوژی تعداد ۴ عدد rack وجود دارد که در هر rack یک سوئیچ وجود دارد (سوئیچ‌های s1r1.s1r2,s1r3,s1r4) هر یک از این ۴ سوئیچ به ۴ عدد host و ۳ سوئیچ ریشه متصل می‌باشند. مثلا سوئیچ s1r1 از یک طرف به h1r1,h2r1,h3r1,h4r1 و از طرف دیگر به S1,S2,S3 متصل می‌باشد. این سوئیچ‌های ریشه به صورت حلقوی به یکدیگر متصل می‌باشند. این توپولوژی نمایانگر توپولوژی یک مرکز داده ساده می‌باشد.



شکل ۲- فلوچارت شناسایی تجهیزات به خطر افتاده در کنترلر اصلی



شکل ۳- توپولوژی مورد استفاده در پیاده سازی

۲-۵-۵- ماژول مسیریابی

ماژول مسیریابی در این کنترلر با استفاده از درخت پوشای ایجاد شده در هنگام پیکربندی اولیه شبکه و داشتن اطلاعات تمامی سوئیچها، لینک‌ها و آدرس MAC تمامی host ها، مسیریابی را از host مبدا تا مقصد انجام می‌دهد و در نهایت لیستی از سوئیچهای مسیر به همراه پورت خروجی هر سوئیچ برگردانده می‌شود. در کنترلرهای دیگر ممکن است مسیریابی توسط ماژول دیگری انجام شود. نکته ی قابل توجه این است که کنترلرهای پشتیبان و اصلی باید از یک ماژول مسیریابی استفاده کنند. اگر دو ماژول مسیریابی متفاوت باشند اما همواره کوتاه‌ترین مسیر را برگردانند، ممکن است به دلیل وجود چند کوتاه‌ترین مسیر، دو مسیر متفاوت به دست آید.

۳- ارزیابی و کارایی

برای ارزیابی عملکرد و کارایی الگوریتم، ابتدا الگوریتم از نظر سرعت تشخیص کنترلر و سوئیچ به خطر افتاده و میزان دقت آن در تشخیص، تحت آزمایش قرار گرفته است. سپس سربار الگوریتم از نظر میزان استفاده از منابع سیستم، یعنی حافظه RAM و پردازنده، مورد آزمایش و ارزیابی قرار گرفته است. تعداد ۱۰۰۰ رکورد از مجموعه داده شامل زمان شناسایی کنترلر، زمان شناسایی سوئیچ، میزان حافظه RAM مصرفی و میزان استفاده از پردازنده، با انجام این آزمایش‌ها جمع آوری شده و ارزیابی‌های لازم روی آن‌ها انجام شده است. بر اساس نتایج به دست آمده از آزمایش‌های عملی، این الگوریتم توانست تمامی رفتارهای نادرست در کنترلر یا سوئیچ را به درستی تشخیص دهد. همچنین در تمامی مواردی که حمله ای رخ نداده بود الگوریتم به درستی بر اساس روند طبیعی خود رفتار میکرد و تشخیص اشتباهی نداشت.

۳-۱- روش ارزیابی

در آزمایش اول ماژول مسیریابی کنترلر اصلی به گونه ای مورد دستکاری قرار گرفت که مسیر انتخاب شده مسیری طولانی‌تر و غیر بهینه باشد (بسته به مقصد می‌رسد اما با مسیر غیر بهینه). این رفتار نوعی از ناهنجاری‌هایی است که تشخیص آنها بسیار سخت است، زیرا کنترلر هدف اصلی که رساندن بسته به مقصد است را برآورده می‌کند اما در حقیقت دچار حمله شده است. مقاله‌های گذشته در این زمینه ([16],[15], [13] , [12]) نمی‌توانند کمک کنند. در این آزمایش از یک کنترلر پشتیبان مورد اعتماد استفاده شده است.

در آزمایش دوم برخی از سوئیچها به گونه ای مورد دستکاری قرار گرفتند تا جداول جریان آنها طبق آن چیزی باشد که

مهاجم می‌خواهد، نه آن چه که کنترلر اصلی به آن می‌گوید. در این آزمایش پس از آن که مسیریابی بسته ابتدایی توسط کنترلر اصلی انجام شود، کنترلر می‌داند که سوئیچ بعدی که درخواست مسیر می‌کند کدام است. بدین ترتیب اگر سوئیچ به درستی عمل کند و بسته را به پورت درست و برنامه ریزی شده هدایت کند، درخواست سوئیچ بعدی مورد انتظار به کنترلر می‌رسد در غیر این صورت اگر سوئیچ بسته را به پورت دیگری هدایت کند، سوئیچی از کنترلر درخواست مسیر می‌کند که مطابق انتظار کنترلر نیست و کنترلر این رفتار را ناهنجاری تشخیص می‌دهد و سوئیچ قبلی را مشکوک اعلام می‌کند.

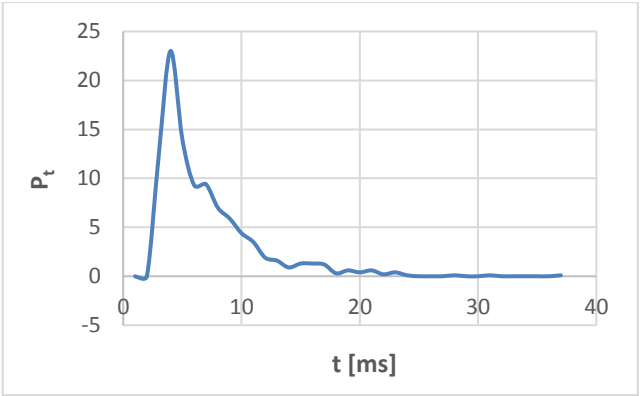
۳-۲- اثربخشی و ارزیابی عملکرد

در آزمایش اول زمانی که اولین بسته با مبدا و مقصد مشخص در قالب پیام packet-in به دست کنترلر می‌رسد، کنترلر مسیریابی را انجام می‌دهد و مسیر خود را با مسیر دریافتی از کنترلر پشتیبان مقایسه می‌کند. در صورتی که هر دوی مسیرها با هم یکسان باشند، پیام "کنترلر اصلی به درستی کار می‌کند" برای کاربر به نمایش در می‌آید. اما اگر این دو مسیر با هم یکسان نباشند پیام "کنترلر اصلی مشکوک است" به نمایش درخواهد آمد. در آزمایش انجام شده این روند یعنی از ابتدای دریافت بسته packet-in تا انتهای نمایش پیام به کاربر، به طور متوسط ۶.۹۲ میلی ثانیه به طول می‌انجامد که زمان بسیار اندک و قابل چشم پوشی می‌باشد .این زمان حاصل مجموع زمان سپری شده برای دو عمل محاسبه کوتاه‌ترین مسیر در کنترلرها و عمل مقایسه کوتاه‌ترین مسیرها که در کنترلر اصلی انجام می‌شود، می‌باشد. به دلیل آنکه عمل محاسبه کوتاه‌ترین مسیر به صورت موازی در هر کنترلر اجرا می‌شود و ماژول مسیریابی کنترلرها یکسان است، پیچیدگی زمانی این عمل O(1) می‌باشد که تابعی از تعداد سوئیچ‌های تحت کنترل کنترلر می‌باشد. همچنین این سربار در کل الگوریتم فقط یک بار و آن هم هنگام مسیریابی اتفاق می‌افتد.

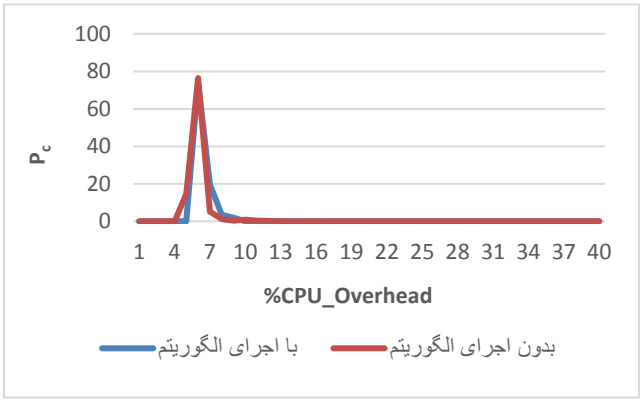
در آزمایش دوم سوئیچ S1 در شکل ۳ مورد هدف قرار گرفت و الگوریتم ۱۰۰۰ بار مورد آزمایش قرار گرفت. حداقل زمان شناسایی سوئیچ، ۱۲۲ میلی ثانیه و حداکثر زمان شناسایی ۱۱۴۲ میلی ثانیه اندازه‌گیری شد و به طور متوسط ۱۰۰۷.۴۲ میلی ثانیه زمان صرف شناسایی سوئیچ مخرب S1 شد. شکل‌های ۴ و ۵ نمودار توزیع احتمال[۱۳] زمان شناسایی کنترلر و سوئیچ به خطر افتاده را نشان می‌دهد. در این نمودارها محور افقی نشانگر زمان شناسایی t و محور عمودی بیانگر درصد شناسایی در آن زمان P_t می‌باشد.

۳-۳- ارزیابی سربار

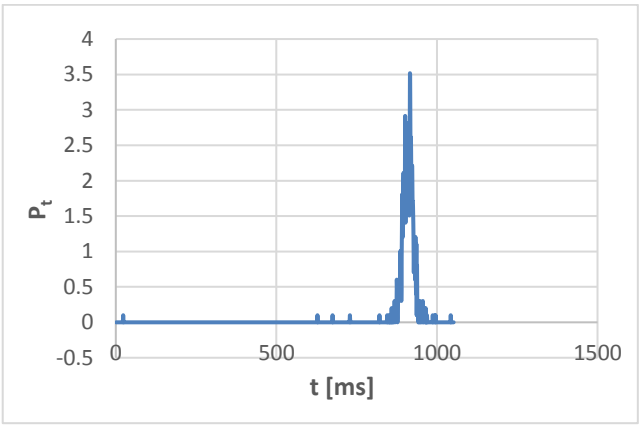
در این قسمت تاثیر اجرای الگوریتم بر روی حافظه RAM و پردازنده، مورد بررسی قرار می‌گیرد. هنگامی که شبکه و کنترلر در سرور اول (اصلی) در حال اجرا هستند، سربار پردازشی (CPU) سیستم به طور میانگین برابر با ۷.۴۲ درصد می‌باشد و در هنگام اجرای الگوریتم این عدد به ۷.۸۸ درصد می‌رسد. همچنین در کنترلر پشتیبان نیز به طور متوسط در اجرای عادی شبکه و کنترلر، سربار پردازنده برابر ۵.۶۲ درصد و در هنگام اجرای الگوریتم به طور میانگین برابر ۵.۸۲ درصد می‌باشد. به طور متوسط در کنترلر اصلی ۰.۴۶٪ و در کنترلر پشتیبان ۰.۲۰٪ سربار پردازشی حاصل از اجرای الگوریتم می‌باشد. همچنین میزان سربار حافظه RAM در هنگام اجرای الگوریتم در کنترلر اصلی و پشتیان در شکل‌های ۸ و ۹ نشان داده شده است. به طور میانگین در کنترلر اصلی سربار حافظه در حالت عادی اجرای شبکه و کنترلر ۵۳.۳ درصد و در هنگام اجرای الگوریتم ۵۲ درصد می‌باشد. در کنترلر پشتیبان نیز در حالت عادی میزان استفاده از حافظه RAM به طور میانگین برابر ۴۷.۹۱ درصد و در هنگام اجرای الگوریتم برابر ۴۷.۰۹ می‌باشد. این کاهش سربار حافظه بعد از اجرای الگوریتم را می‌توان به رفتار سیستم عامل مرتبط دانست که در لحظه های مختلف، منابع سیستم را به منظور کارهای دیگر مشغول یا آزاد می‌کند. اما مشخصا این الگوریتم سربار حافظه رم ندارد. در نمودار شکل‌های ۶ و ۷ محور افقی نشانگر سربار پردازنده سیستم می‌باشد و محور عمودی بیانگر در صد رخداد آن میزان از سربار پردازنده P_c در طول اجرای الگوریتم و بدون اجرای الگوریتم می‌باشد. همچنین در شکل‌های ۸ و ۹ محور افقی سربار حافظه RAM سیستم و محور عمودی در صد رخداد این میزان از حافظه P_m را در کنترلر اصلی و پشتیبان در حین اجرای الگوریتم و بدون اجرای الگوریتم نشان می‌دهد.



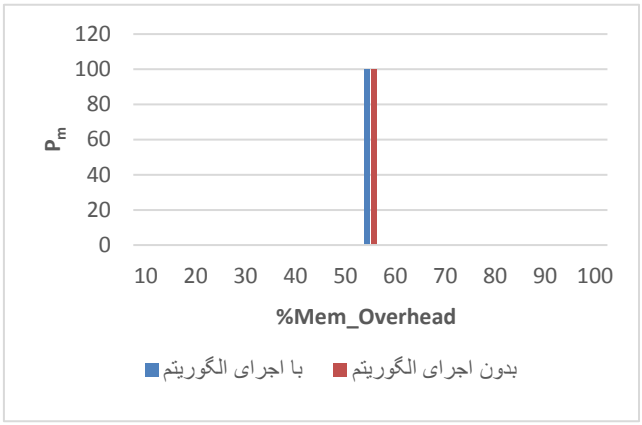
شکل ۴- توزیع احتمال زمان تشخیص کنترلر به خطر افتاده



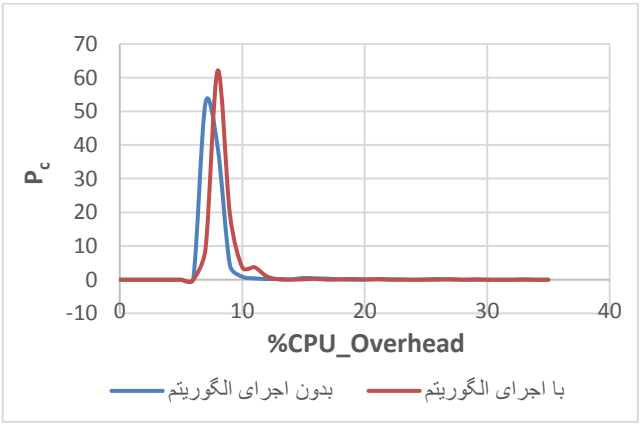
شکل ۷- توزیع احتمال مصرف پردازنده در کنترلر پشتیبان



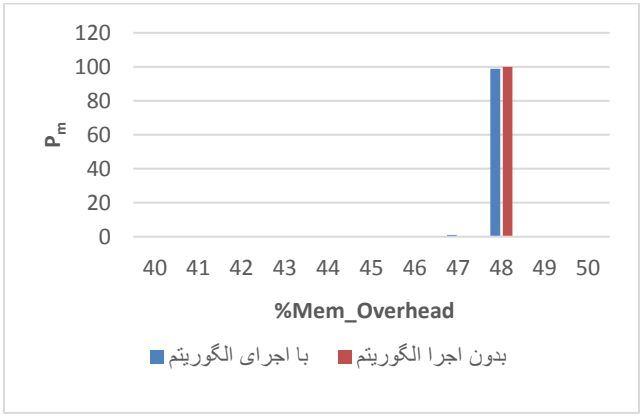
شکل ۵- توزیع احتمال زمان شناسایی سوئیچ به خطر افتاده



شکل ۸- توزیع احتمال مصرف حافظه RAM در کنترلر اصلی



شکل ۶- توزیع احتمال مصرف پردازنده در کنترلر اصلی



شکل ۹- توزیع احتمال مصرف حافظه RAM در کنترلر پشتیبان

۳-۴- جمع‌بندی ارزیابی

بر اساس آزمایش‌های انجام شده، این الگوریتم توانایی شناسایی سوئیچ‌ها و کنترلرهای به خطر افتاده را با قابلیت اعتماد بالا دارد. زمان شناسایی کنترلر به خطر افتاده بسیار اندک و قابل قبول می‌باشد. در قسمت دیگر الگوریتم که مربوط به صحت سنجی عملکرد سوئیچ‌ها است تاخیر به وجود آمده که شامل درخواست سوئیچ‌ها از کنترلر برای هدایت بسته، نصب قانون جریان توسط کنترلر و هدایت بسته توسط سوئیچ می‌باشد به ازای هر سوئیچ تقریبا ۱ ثانیه به طول می‌انجامد. در روش معمول مسیریابی، زمانی که اولین سوئیچ از کنترلر درخواست مسیریابی می‌کند، کنترلر مسیر را محاسبه می‌کند و قوانین جریان را برای تمامی سوئیچ‌های مسیر نصب می‌کند که این کار باعث افزایش سرعت انتقال بسته می‌شود. اما در این روش برای صحت سنجی عملکرد سوئیچ‌ها، هر سوئیچ موظف است به طور مستقل از کنترلر سوال بپرسد و قاعده جریان مربوط به خود را از کنترلر دریافت کند. این زمان تا حدودی زیاد است و در پژوهش‌های بعدی برای کمتر شدن زمان شناسایی سوئیچ تلاش خواهد شد.

مقاله‌هایی که در زمینه صحت سنجی رفتار شبکه یا شناسایی تجهیزات به خطر افتاده که در برگیرنده روش بررسی بسته‌های به‌روزرسانی شبکه باشند، در بخش ۱-۱ آورده شده است. در تمامی روش‌های گذشته فرض بر این بوده است که کنترلر یک موجودیت قابل اعتماد است. بنابراین زمانی که کنترلر به خطر بیفتد، کارایی آن روش قابل تضمین نمی‌باشد. نزدیک ترین و به‌روزترین روش در رابطه با موضوع این مقاله، مقاله های [17] و [18] می‌باشند.

مهمترین تفاوت این مقاله با مقاله [17] در فرض مورد اعتماد بودن کنترلر می‌باشد که در این مقاله این فرض در نظر گرفته نشده است. بنابراین این روش حتی زمانی که کنترلر و سوئیچ به صورت همزمان به خطر افتاده باشند نیز کارا خواهد بود. همچنین در مقاله [17] برای صحت سنجی رفتار سوئیچ ها، چندین سوئیچ به صورت تصادفی انتخاب می‌شود و یک قاعده جریان از جدول جریان آن‌ها انتخاب و عملکرد سوئیچ تحت نظر، مطابق با این قانون مورد ارزیابی قرار می‌گیرد و این روند به صورت تناوبی اجرا می‌شود. همین تصادفی بودن انتخاب سوئیچ ها و قوانین جریان باعث می‌شود تا شناسایی سوئیچ به خطر افتاده با روشی قابل اطمینان و بهینه در زمان صورت نگیرد. . اما در این مقاله با توجه به استفاده از الگوریتم درخت پوشا می‌توان با یک بار اجرای الگوریتم، تمامی سوئیچ‌های شبکه را در مدت زمان مطلوبی مورد بررسی قرار داد.

اما تفاوت اصلی این مقاله با مقاله [18] در روش ارائه شده می‌باشد که این مقاله با استفاده از یک روش ساده و کم هزینه‌تر نسبت به مقاله [18] می‌تواند با سربار پردازشی و حافظه‌ای کمتر، سوئیچ و کنترلر را حتی در صورت به خطر افتادن همزمان، شناسایی کند. اما زمان شناسایی تجهیزات به خطر افتاده در مقاله [18] نسبت به این مقاله کمتر می‌باشد. جدول زیر جزئیات بیشتری در رابطه با مقایسه دو روش را نشان می‌دهد.

جدول ۱ – مقایسه سربار روش فعلی با روش SDN-RDCD

روش	متوسط درصد افزایش پردازنده	متوسط درصد افزایش حافظه	زمان شناسایی کنترلر (میلی ثانیه)	زمان شناسایی سوئیچ (میلی ثانیه)
SDN-RDCD [18]	8.86%	0.87%	۱.۹۲	۷.۴۸
مقاله فعلی	0.46%	0	۶.۹۲	۱۰۰۷.۴۲

۴- نتیجه گیری

در این مقاله ابتدا تهدیدهای بالقوه شبکه‌های SDN بیان شد و سپس روش پیشنهادی معرفی گردید که توانایی شناسایی تجهیزات به خطر افتاده شبکه بر اساس مدل‌های حمله ذکر شده را داراست. از آنجا که روش تشخیص پیشنهادی از پروتکل OpenFlow موجود بدون تغییر استفاده می‌کند، الگوریتم پیشنهادی به عنوان دو برنامه OpenFlow پیاده سازی شده است که می‌توان بدون متوقف کردن شبکه آن‌ها را اجرا کرد. بنابراین این مکانیزم آنلاین بوده و قابلیت اجرا بر روی شبکه‌های OpenFlow را دارد. وجه تمایز این کار با سایر مقاله‌های دیگر که در زمینه شناسایی تجهیزات به خطر افتاده منتشر شده اند در این است که این روش بدون جابجایی حجم زیادی از بسته‌ها و با سربار پردازشی و حافظه ای بسیار کم، سوئیچ و کنترلر را، حتی هنگامی که هر دو بصورت همزمان به خطر افتاده اند در زمانی قابل قبول شناسایی کند.

۱-۴- کارهای آینده

- در این مقاله سعی شد دو حمله و خطری که تجهیزات شبکه SDN را تهدید می‌کنند مدل سازی و

ایمن سازی شوند. ممکن است مدل‌های حمله دیگری نیز وجود داشته باشد که بتوان برای آن‌ها مکانیزم‌های تشخیص طراحی کرد و جلوی این گونه حمله‌ها را گرفت.

- همچنین در این روش برای اینکه بتوان تمامی سوئیچ‌های شبکه را مورد صحت سنجی قرار داد، باید چندین مسیر انتخاب شود که در مجموع تمامی سوئیچ‌ها را مورد پوشش قرار دهد. می‌توان روشی ارائه کرد که با یک بار اجرای الگوریتم، تمامی سوئیچ‌های درخت پوشا مورد بررسی قرار گیرند.
- سرعت شناسایی سوئیچ به خطر افتاده متناسب با فاصله سوئیچ از host مبدا، به صورت خطی افزایش می‌یابد و زمان نسبتا زیادی برای تشخیص سوئیچ مخرب صرف می‌شود. در کارهای آینده تمرکز اصلی بر روی کاهش این زمان قرار خواهد گرفت.

مراجع

[1] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, 2015.

[2] S. Wang, K. G. Chavez and S. Kandeepan, "SECO: SDN sEcure COntroller algorithm for detecting and defending denial of service attacks," 2017 5th International Conference on Information and Communication Technology (ICoIC7), Malacca City, pp. 1-6, 2017.

[3] M. Conti, A. Gangwal and M. S. Gaur, "A comprehensive and effective mechanism for DDoS detection in SDN," 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Rome, pp. 1-8, 2017

[4] Jantila, Saksit, and Kornchawal Chaipah. "A security analysis of a hybrid mechanism to defend DDoS attacks in SDN." Procedia Computer Science 86, pp.437-440, 2016

[5] Chao Qi et al., "An intensive security architecture with multi-controller for SDN," 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), San Francisco, CA, 2016, pp. 401-402, 2016.

[6] W. Etaiwi, M. Biltawi and S. Almajali, "Securing Distributed SDN Controllers against DoS Attacks," 2017 International Conference on New Trends in Computing Sciences (ICTCS), Amman, pp. 203-206, 2017

[7] S. B. Hashemi Natanzi and M. R. Majma, "Secure distributed controllers in SDN based on ECC public key infrastructure," 2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, pp. 1-5, 2017.

[8] Jun-Huy Lam, Sang-Gon Lee, Hoon-Jae Lee and Y. E. Oktian, "Securing distributed SDN with IBC," 2015 Seventh International Conference on Ubiquitous and Future Networks, Sapporo, 2015.

[9] S. Tatlicioglu, S. Civanlar, B. Gorkemli, E. Lokman, A. M. Balci and C. B. Eliacik, "A security services platform for Software Defined Networks," 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, pp. 39-43, 2016

[10] J. Jeong, J. Seo, G. Cho, H. Kim and J. Park, "A Framework for Security Services Based on Software-Defined Networking," 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, Gwangiu, pp. 150-153, 2015.

[11] Tri-Hai Nguyen and Myungsik Yoo, "Analysis of link discovery service attacks in SDN controller," 2017 International Conference on Information Networking (ICOIN), Da Nang, pp. 259-261, 2017.

[12] Khurshid, A., Zou, X., Zhou, W., Caesar, M., & Godfrey, P. B. (2013). Veriflow: Verifying network-wide invariants in real time. In Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13), pp.15-27, 2013.

[13] Dhawan, M., Poddar, R., Mahajan, K., & Mann, V. SPHINX: Detecting Security Attacks in Software-Defined Networks. In Ndss ,Vol. 15, pp. 8-11, 2015.

[14] Alshra'a, Abdullah Soliman, and Jochen Seitz. "Using inspector device to stop packet injection attack in SDN." IEEE Communications Letters 23.7 (2019)

[15] Dinh, Phuc Trinh, et al. "Abnormal SDN switches detection based on chaotic analysis of network traffic." 2019 25th Asia-Pacific Conference on Communications (APCC). IEEE, 2019.

[16] Dinh, Phuc Trinh, and Minho Park. "ECSD: Enhanced Compromised Switch Detection in an SDN-Based Cloud Through Multivariate Time-Series Analysis." IEEE Access 8 (2020.

[17] Po-Wen Chi, Chien-Ting Kuo, Jing-Wei Guo and Chin-Laung Lei, "How to detect a compromised SDN switch," Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), London, pp. 1-6, 2015.

[18] H. Zhou et al., "SDN-RDCD: A Real-Time and Reliable Method for Detecting Compromised SDN Devices," in IEEE/ACM Transactions on Networking, vol. 26, no. 5, pp. 2048-2061, 2018.

[19] O. Salman, I. H. Elhajj, A. Kayssi and A. Chehab, "SDN controllers: A comparative study," 2016 18th Mediterranean Electrotechnical Conference (MELECON), Lemesos, , pp. 1-6, 2016.

[20] "stplib, " [Online]. Available: https://osrg.github.io/ryubook/en/html/spanning_tree.html

پاورقی‌ها:

[1] Control Plane
[2] Data Plane
[3] Forward
[4] Software Defined Networks (SDN)
[5] Forwarding plane
[6] Denial of Service (DoS)
[7] FlowGraph
[8] Incorrect Forwarding
[9] Cross Platform
[10] ToR (Top of Rack)
[11] Data Center
[12] Ring
[13] Probability Distribution