# مطالعه‌ای روی طراحی سیستم دفاعی برای تشخیص حمله DOS و دفاع از آن در کنترلرهای SDN

**علیرضا محمودی‌فرد[۱] ، علی ملکی[۲]***

[۱] دانشجوی‌کارشناسی‌ارشد ناپیوسته مدیریت صنعتی، دانشکده علوم انسانی، دانشگاه شاهد (و فارغ‌التحصیل کارشناسی ارشد مهندسی برق و مدرس دانشگاه‌ها)، alireza10.m10@gmail.com

[۲] دانشجوی کارشناسی‌ارشد ناپیوسته مهندسی مخابرات دانشگاه تربیت دبیر شهید رجایی، A.malekibme@gmail.com

**اطلاعات مقاله**

*تاریخچه مقاله:*
تاریخ دریافت مقاله: ۱۴۰۲/۰۶/۱۷
تاریخ پذیرش مقاله: ۱۴۰۲/۰۷/۲۰
تاریخ انتشار مقاله: ۱۴۰۲/۰۷/۲۲

*کلمات کلیدی:*
اینترنت
شبکه تعریف شده نرم‌افزار (SDN)، کنترل کننده SDN
حملات DoS
طراحی سیستم دفاعی
تشخیص حمله DoS

**چکیده**

با توسعه اینترنت، مشکلات انعطاف‌پذیری و مدیریت‌پذیری معماری شبکه‌های سنتی به‌طور فزاینده ای برجسته شده است؛ برای حل این مشکل، شبکه تعریف شده نرم‌افزار[۱] در سال‌های اخیر متولد شد؛ ایده اصلی SDN، جدا کردن لایه انتقال داده و لایه کنترل است که SDN را متمرکز، قابل گسترش و برنامه‌ریزی می‌کند. شبکه تعریف شده با نرم‌افزار (SDN)، یک معماری شبکه است که هدف آن ارائه انعطاف‌پذیری بالا از طریق جدا کردن منطق شبکه از توابع ارسال است؛ سهولت برنامه‌ریزی، SDN را به یک پلتفرم عالی برای اجرای طرح‌های مختلف تبدیل می‌کند که شامل استقرار برنامه‌ها، راه‌حل‌های امنیتی و مدیریت شبکه غیرمتمرکز در یک محیط مرکز داده چند مستاجر است؛ اگرچه این می‌تواند کاربردهای زیادی را در حوزه‌های مختلف معرفی کند و منجر به تاثیر زیاد در جنبه‌های مختلف شود، اما امنیت معماری SDN یک سوال باز باقی‌مانده است و باید بر اساس مفهوم جدید SDN مورد بازبینی قرار گیرد؛ مکانیسم‌های تشخیص حمله مبتنی بر SDN فعلی دارای محدودیت‌هایی هستند؛ ایده اصلی SDN جدا کردن لایه انتقال داده و لایه کنترل است که SDN را متمرکز، قابل گسترش و برنامه‌ریزی می‌کند. خدمات اصلی، پیکربندی مهم و سایر عملکردهای SDN بر روی کنترلر SDN مستقر شده‌اند که به تمرکز مدیریت شبکه کمک می‌کند، اما همچنین تهدیداتی را برای امنیت شبکه معرفی می‌کند. اگر به کنترلر SDN حمله شود، بر پوشش کنترلر تاثیر می‌گذارد؛ کل شبکه، در موارد شدید، فلج خواهد شد. مقاله مبنا، بر اساس تجزیه و تحلیل ویژگی‌های معماری شبکه SDN، پروتکل OpenFlow و اصل حملات DoS، دو روش تشخیص و کاهش DoS را پیشنهاد و اجرا کرد؛ اولین مورد از کنترلر Ryu، sFlow و Postman برای تجسم و حذف ترافیک حملات DoS استفاده می‌کند؛ روش دوم یک الگوریتم تشخیص خودکار حمله DoS با کنترلر POX است.

[۱] SDN

## ۱ - مقدمه

• شبکه تعریف شده با نرم‌افزار (SDN)، یک معماری شبکه نوظهور است که برای تغییر محدودیت‌های زیرساخت‌های شبکه فعلی پیشنهاد شده است؛ این منطق، کنترل شبکه و ترافیک شبکه را برای معرفی مفاهیم صفحه کنترل و صفحه داده جدا می‌کند؛ اگرچه SDN چندین مفهوم مشابه با شبکه‌های سنتی دارد، اما ویژگی‌های منحصر به فرد خود را هم دارد. منطق کنترل (کنترل‌کننده) برای مدیریت منطق شبکه و فرآیند تصمیم‌گیری متمرکز می‌شود، در حالی‌که دستگاه‌های صفحه داده (سوئیچ‌های شبکه) به دستگاه‌های حمل و نقل ساده تبدیل می‌شوند. در SDN، کنترل‌کننده تصمیم می‌گیرد که چگونه ترافیک در شبکه باید مدیریت شود و سپس تصمیمات را به‌عنوان قوانین جریان به دستگاه‌های صفحه داده منتقل می‌کند؛ اگرچه این تفکیک منجر به مزایای بسیاری مانند درجه انعطاف‌پذیری بالا می‌شود که یکی از دلایل اصلی پذیرش گسترده SDN حتی در بین شرکت‌های بزرگ است [۱ و ۴۱]، سهولت انطباق سایر تکنیک‌ها، برنامه‌ریزی، مقیاس‌پذیری و غیره، هنگامی‌که اجزای آن، مانند کنترلر SDN، سوئیچ‌های فعال OpenFlow، هدف اصلی حمله باشند، می‌تواند ضعیف‌ترین نقطه باشد، زیرا می‌تواند کل شبکه را تحت‌تاثیر قرار دهد [۲ و ۴۱ و ۴۲]. با شتاب قابل‌توجهی به سمت تطبیق SDN در سیستم‌های شبکه، مکانیسم‌های دفاعی امنیتی بسیاری پیشنهاد شده است؛ با این حال، مکانیسم‌های تشخیص حمله فعلی که مبتنی بر SDN هستند، دارای محدودیت‌هایی هستند؛ به شکلی دیگر می‌توان بیان نمود که SDN از کنترل‌کننده متمرکز منطقی برای حفظ نمای گسترده شبکه استفاده می‌کند و تصمیمات ارسال را برای پشتیبانی از سیاست‌های مدیریت شبکه ریزدانه انجام می‌دهد، که به برنامه‌نویسی، انعطاف‌پذیری کامل می‌دهد و باعث کاهش پیچیدگی پیکربندی و عملیات شبکه می‌شود [۳ و ۴۱ و ۴۲]؛ با این حال، بدیهی است که کنترل‌کننده متمرکز، سربار قابل توجهی دارد و به‌راحتی تبدیل به یک گلوگاه می‌شود؛ این منجر به مشکلاتی در مقیاس‌پذیری و امنیت می‌شود؛ در حالی‌که مطالعات و راه‌حل‌های زیادی در مورد مساله مقیاس‌پذیری وجود دارد، تحقیقات بسیار کمی در مورد مسائل امنیتی حتی چالش برانگیزتر وجود دارد؛ به‌عنوان مثال، مهاجمان ممکن است به سادگی با ارسال بسته‌های بی‌فایده عظیم، حملات اشباع را روی کنترل‌کننده SDN نصب کنند؛ در نتیجه، کنترل‌کننده هر بسته جدید بی‌فایده را برای ایجاد ورودی جریان کنترل می‌کند، که تا حد زیادی منابع محاسباتی را اشغال می‌کند و کنترل‌کننده را تحت‌تاثیر قرار می‌دهد [۴و۵]؛ بنابراین، یک برنامه کاربردی تشخیص حمله DoS برای محافظت از سیستم SDN در برابر کاربران مخرب مورد نیاز است [۴۱ و ۴۲].

## ۲- متن بررسی

### ۲-۱- معماری شبکه‌های تعریف‌شده نرم‌افزاری

• معماری جداسازی لایه‌ای SDN عمدتا از سه لایه تشکیل شده است: ۱) لایه زیرساخت، ۲) لایه کنترل و ۳) لایه برنامه ("شبکه‌سازی تعریف‌شده توسط نرم‌افزار: هنجار جدید برای شبکه‌ها - بنیاد شبکه باز"، ۲۰۲۱). دو لایه بالایی وظیفه کنترل و لایه پایینی وظیفه انتقال اطلاعات را بر عهده دارند [۶ و ۱]؛ ارتباط بین لایه کنترل و لایه زیرساخت، یک رابط به جنوب است که در حال حاضر با استفاده از پروتکل OpenFlow پیاده‌سازی می‌شود و عمدتا مسئولیت مسیریابی، کنترل دسترسی و عملکردهای امنیتی شبکه را بر عهده دارد؛ ارتباط بین لایه کنترل و لایه برنامه، یک رابط شمالی است که دستورالعمل‌ها را برای برنامه‌ها صادر می‌کند و در نتیجه، منطق برنامه را قابل کنترل می‌کند [۴۲].

چنین معماری شبکه‌ای SDN را به روش‌های زیر از شبکه‌های سنتی متمایز می‌کند. ابتدا در شبکه‌های سنتی، استراتژی‌های کنترل و حمل و نقل شبکه در یک دستگاه پیاده‌سازی می‌شوند، در حالی‌که SDN لایه کنترل و لایه ارسال را جدا می‌کند [۷ و ۴۱]؛ بار شبکه را تا حد زیادی کاهش می‌دهد و کارایی ارسال را بهبود می‌بخشد؛ ثانیا، در شبکه‌های سنتی، سازندگان مختلف منجر به تفاوت‌هایی در سخت‌افزار و نرم‌افزار تجهیزات می‌شوند که منجر به ایجاد شبکه ایزوله می‌شود و تنها از استانداردهای یکپارچه ثابت می‌توان برای برقراری ارتباط با دیگران استفاده کرد. SDN به‌طور منطقی لایه کنترل را متمرکز می‌کند، حمل و نقل سخت‌افزار زیرین را کنترل می‌کند، وضعیت کل شبکه را نظارت می‌کند و توانایی کنترل شبکه جهانی را از طریق رابط‌های جنوب و شمال دریافت می‌کند؛ علاوه بر این، شبکه SDN قابل برنامه‌ریزی، با رابط‌های باز و استاندارد است؛ رابط شامل دو جنبه است؛ یکی از آن‌ها، رابط شمالی لایه کنترل و لایه کاربردی است؛ کاربران می‌توانند چندین برنامه را برای دسترسی به شبکه SDN از طریق رابط شمال، برنامه‌ریزی کنند؛ دیگری رابط جنوب بین لایه کنترل و لایه ارسال است؛ استاندارد یکپارچه رابط،

تفاوت‌های تجهیزات را حذف می‌کند و کنترل‌کننده نیازی به نظارت بر هر دستگاه ارسال ندارد [۸ و ۴۱ و ۵].

### ۲-۲- OpenFlow

• اولین مقاله درباره OpenFlow در سال ۲۰۰۸ توسط تیمی به سرپرستی پروفسور نیک مک کئون از استمفورد منتشر شد؛ پیش از این، به‌دلیل وجود دستگاه‌ها و پروتکل‌های بی‌شماری در شبکه‌ها، آزمایش‌ها یا کارکرد پروتکل‌های جدید در یک محیط شبکه واقعی بسیار دشوار بود؛ متقاعد ساختن سازندگان سخت‌افزار مختلف برای یکسان‌سازی مشخصات و استانداردهای تجهیزات تولید بسیار غیر عملی بود [۹]؛ سوئیچ OpenFlow به‌وجود آمد؛ سوئیچ OpenFlow، نه تنها دارای ویژگی‌های کارایی بالا و هزینه کم است، بلکه می‌تواند طیف وسیعی از تحقیقات تجربی را تکمیل کند؛ جدول جریان، یک مفهوم بسیار مهم در OpenFlow است. در معماری شبکه سنتی، کنترل و ارسال شبکه از طریق جدول نقشه‌برداری آدرس مک دو لایه و جدول مسیریابی آدرس IP سه لایه انجام می‌شود؛ OpenFlow نیز از همین مفهوم استفاده می‌کند، با این تفاوت که کلمات کلیدی هر لایه متمرکز هستند. جدول جریان عمدتا از دستورالعمل‌ها و ورودی‌های جدول جریان تشکیل شده است تا به سوئیچ اطلاع دهد که با جریان داده چه کاری انجام دهد [۱۰ و ۱۱ و ۴۱].

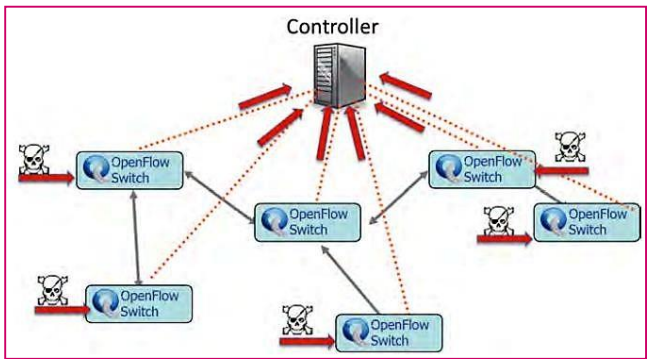
کانال OpenFlow کانالی را برای تبادل پیام بین سوئیچ OpenFlow و کنترل‌کننده OpenFlow فراهم می‌کند و دستورالعمل‌ها و داده‌های صادر شده توسط کنترلر را به سوئیچ منتقل می‌کند؛ پروتکل OpenFlow شامل سه نوع پیام است: کنترل‌کننده به سوئیچ، ناهمزمان و متقارن که هر کدام دارای چندین پیام فرعی است. Controller-to-Switch توسط کنترلر به سوییچ ارسال می‌شود؛ پیام‌های ناهمزمان برای همگام‌سازی سوئیچ و کنترلر استفاده شده و توسط سوئیچ آغاز می‌شوند؛ پیام‌های متقارن می‌توانند توسط هر دو طرف تعامل آغاز شوند. سوئیچ OpenFlow به سوئیچی اطلاق می‌شود که از پروتکل OpenFlow پشتیبانی می‌کند؛ خط لوله سوئیچ شامل جداول جریان چندگانه است؛ هنگامی‌که بسته ارسال شده توسط کاربر توسط سوئیچ OpenFlow دریافت می‌شود، سوئیچ برای تطبیق یک به یک به جدول جریان مراجعه می‌کند؛ هنگام تطبیق، سوئیچ OpenFlow با جدول جریان با بالاترین مقدار وزن شروع می‌شود. اگر یک ورودی جدول جریان منطبق پیدا نشد، بسته داده فعلی به کنترل‌کننده SDN ارسال می‌شود و کنترل‌کننده تصمیم می‌گیرد که آیا بسته داده دور انداخته شود یا به جداول جریان بعدی ارسال شود [۱۲]؛ برای مثال می‌توان گفت که با در نظر گرفتن OpenFlow، سوئیچ SDN بسته‌ها را مطابق با قوانین جدول جریان به جلو می‌فرستد، جایی‌که بسته جدید یا بسته غیرعادی که نمی‌تواند در جدول جریان پردازش شود، به کنترل‌کننده ارسال می‌شود؛ از این نظر، نیازی نیست که مهاجم قبل از شروع حمله، آدرس IP یا مکان کنترلر را از طریق اسکن بگیرد؛ از آنجایی‌که مهاجم بسته‌های حمله خاص و بسته‌های غیرعادی را به شبکه‌های SDN ارسال می‌کند، همه سوئیچ‌ها به‌طور خودکار این بسته‌ها را به کنترل‌کننده خود ارسال می‌کنند [۴۱ و ۴۲].

در مقایسه با حملات DDoS سنتی [۵] که در ابتدا نیاز به سوء استفاده از آدرس‌های IP میزبان قربانی دارند، این نوع DDoS در کنترل‌کننده SDN کور است؛ بنابراین ما آن را به‌عنوان حمله DDoS کور تعریف می‌کنیم؛ فلج شدن کنترلر در نتیجه فوران جریان داده ارسال شده به شبکه SDN، اجرای موفقیت‌آمیز یک حمله DDoS کور را نشان می دهد. شکل ۱ نمای کلی حمله DDoS کور را نمایش می‌دهد [۴۱ و ۸ و ۱۴].

### ۲-۳- حملات انکار سرویس[۲]

• Denial of Service (DoS) حمله‌ای است که از تکنیک‌های هک موجود استفاده می‌کند تا به‌طور خاص نقص‌های پروتکل شبکه را هدف قرار دهد و باعث می‌شود سیستم سرویس هدف را متوقف کند یا با تاخیر مواجه شود. یکی از روش‌های حمله DoS، حملات مبتنی بر سیل است که عمدتا شامل UDP، TCP SYN و ICMP است [۱۵ و ۴۱]. مهاجمان مقادیر انبوهی از بسته‌های داده فریبنده را به میزبان هدف ارسال می‌کنند و باعث می‌شوند میزبان هدف منابع محاسباتی خود مانند CPU، حافظه و پهنای باند شبکه را تمام کند و در نهایت، کل شبکه فلج شود. در پروتکل OpenFlow، اگر سوئیچ OpenFlow یک بسته جدید دریافت کند و هیچ اطلاعات مرتبطی در جدول جریان نداشته باشد، کنترل‌کننده تطبیق را پردازش نمی‌کند بلکه مستقیما مسیر مسیریابی را محاسبه می‌کند؛ اگر مهاجم تظاهر به سوئیچ کند و تعداد زیادی بسته جدید برای کنترلر ارسال کند، این امر باعث می‌شود که کنترل‌کننده مشغول محاسبه مسیر مسیریابی باشد و منابع و پهنای باند زیادی مصرف کند که تاثیر جدی بر روی کاربران عادی خواهد داشت. شکل ۱ چگونگی مختل نمودن سیستم هدف توسط حمله DoS را نمایش می‌دهد [۱۶ و ۴۱ و ۱۴].

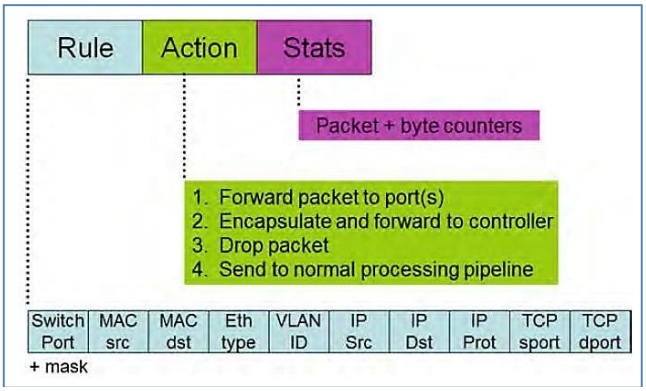
[۲] DoS




**شکل ۱- حمله کور DoS به کنترلر SDN (حملات DDOS نیز مشابه این‌گونه حمله کور است) [۱۲ و ۴۲].**

برای آسیب‌پذیری‌های احتمالی در سیستم SDN، یک مهاجم ممکن است یک حمله DoS یا یک حمله انکار سرویس توزیع شده[۳]را روی کنترل‌کننده انجام دهد؛ به‌عنوان مثال، یک مهاجم ممکن است حجم زیادی از ترافیک جدید را در مدت‌زمان کوتاهی ایجاد کند و از کنترل‌کننده درخواست کند تا به درخواست‌های کاربران عادی پاسخ داده نشود و باعث فلج شدن شبکه شود [۱۷ و ۴۲ و ۴۱].

شکل ۲ نیز ساختار ورودی جریان در سوئیچ SDN را نمایش می‌دهد.




**شکل ۲- ساختار ورودی جریان در سوئیچ SDN [۱۲و۴۲]**

مطابق شکل ۲، هر ورودی جریان در جدول جریان یک سوئیچ شامل سه مورد است، یعنی قانون، عمل و آمار؛ مهاجم می‌تواند یک بسته جدید یا غیرعادی از IP، پورت، MAC و غیره که با دقت انتخاب شده است بسازد و سپس آن را به سوییچ ارسال کند. به‌طور کلی، هیچ قانونی در سوئیچ وجود ندارد که با بسته جدید ارسال شده برای اولین بار مطابقت داشته باشد؛ بسته در کنترل‌کننده آپلود می‌شود و سپس کنترل‌کننده اطلاعات این بسته را برای همه رابط‌های شبکه پخش می‌کند تا مسیر آن را پیدا کند [۱۲ و ۴۱ و ۴۲]. پس از دریافت مسیر، کنترل‌کننده قوانین مربوطه را به جدول جریان سوئیچ صادر می‌کند؛ در غیر این‌صورت، کنترلر برای سوئیچ‌ها قانونی وضع می‌کند تا این بسته‌ها را رها کنند؛ کل این فرآیند پاسخ‌گویی به زمان زیادی نیاز دارد؛ سپس مهاجم گروهی از بسته‌ها را با همان اطلاعات برای بار دوم به سوی سوئیچ ارسال می‌کند، اگر زمان پاسخ بسیار کوتاه‌تر از زمان اول باشد، می‌توان شبکه را به‌عنوان معماری SDN تعیین کرد؛ یک مهاجم ممکن است Blind DDoS را مستقیما روی شبکه‌ای راه‌اندازی کند که ادعا می‌کند معماری شبکه SDN است یا مهاجم قبلا می‌داند که سیستم SDN با اسکن است (شکل ۱)؛ حمله DDoS کور، یک تهدید جدی برای امنیت SDN است؛ از یک طرف، مقدار زیادی از جریان داده حمله ممکن است باعث شود جدول جریان سوئیچ پر از قوانین زباله باشد و در نتیجه باعث کاهش عملکرد یا سرریز شدن ورودی‌های جدول جریان شود؛ از طرف دیگر، حمله Blind DDoS با ایجاد نادرست کارکرد کنترلر باعث فلج شدن شبکه می‌شود؛ روش‌های سنتی امنیت شبکه هیچ دفاع موثری در برابر این نوع حملات ارائه نمی‌دهند؛ بنابراین نیاز به توسعه روش دفاعی جدید برای کاهش تهدید SDN دارد [۱۸ و ۱۹ و ۴۱].

### ۲-۴- روش دفاع از هدف متحرک در برابر DDoS کور

• سیستم‌های دفاعی موجود از جمله فایروال، IDS، IPS، WAF و غیره، همگی از فناوری‌های پدافند غیرعامل ساکن استفاده می‌کنند؛ در نتیجه قادر به ارائه دفاع امنیتی پویا به‌طور موثر در برابر حملات ناشناخته یا آنی در شبکه نیستند. بیشتر سیستم‌های دفاعی به‌دنبال شناسایی کامل و جلوگیری از همه حملات هستند؛ با این حال، واضح است که منطقی نیست زیرا آسیب‌پذیری‌های روز صفر بی‌پایانی وجود دارد؛ بنابراین، محققان امنیت شبکه به‌طور فعال در حال کاوش مدل امنیتی جدید [۱۷-۱۹]، به‌دنبال تعادل پایدار بین هزینه‌های امنیتی و دفاعی هستند؛ پدافند هدف متحرک یکی از این دستاوردها است که با مدل قبلی امنیت شبکه مبتنی بر تشخیص کاملا متفاوت است [۲۰ و ۱۳ و ۴۱].

### ۲-۵- روش شناسی

#### ۲-۵-۱- رویکرد A با استفاده از کنترلر Ryu

• Ryu یک کنترل‌کننده SDN منبع باز است که در پایتون نوشته شده است؛ از پروتکل OpenFlow پشتیبانی می‌کند و استفاده از آن با APIهای به‌خوبی تعریف شده آسان است؛ این رویکرد، از Ryu به‌عنوان کنترل‌کننده SDN، همراه با sFlow به‌عنوان یک روش نظارت بر ترافیک برای درک عملکرد تشخیص حمله DoS استفاده می‌کند، سپس از Postman برای ارسال جدول جریان برای رها کردن بسته‌ها استفاده می‌کند؛ روند آزمایشی دقیق در زیر نشان داده شده است [۲۱ و ۴۱]:

[۳] DDoS

۱. با اجرای اسکریپت پایتون مربوطه، کنترلر Ryu راه‌اندازی می‌شود.
۲. شبکه شبیه‌سازی شده Mininet، با دو میزبان متصل به یک سوئیچ راه‌اندازی می‌شود.
۳. sFlow Agent، راه‌اندازی شده و این سوئیچ تحت نظارت قرار می‌گیرد؛ هنگامی‌که هیچ حمله DoS رخ نمی‌دهد، باید به‌عنوان تصویری نشان داده شود.
۴. با استفاده از دستوری، یک حمله سیل بر روی Mininet ایجاد می‌شود
۵. دوباره sFlow را بررسی کرده و اکنون حمله DoS پیدا می‌شود.
۶. Postman را راه‌اندازی کرده و یک جدول جریان، به سوییچ ارسال می‌شود تا ترافیک DoS (بسته‌های ICMP) حذف شود.
۷. سپس بررسی می‌شود که sFlow و حمله DoS با موفقیت مسدود شده‌اند یا خیر [۱۸].

**۲-۵-۲- رویکرد B با استفاده از کنترلر POX**

• POX یک کنترلر SDN منبع باز است که در پایتون نوشته شده است و از پروتکل OpenFlow 1.0 پشتیبانی می‌کند؛ این رویکرد از POX به‌عنوان کنترل‌کننده SDN، همراه با hping3 به‌عنوان روش حمله DoS برای اجرای عملکرد تشخیص حمله DoS استفاده می‌کند؛ کنترلر IPهای منبعی را که بسته‌های بیشتری از حد آستانه ارسال کرده‌اند، مسدود می‌کند، که در نسخه نمایشی زیر ۵۰ بسته شده است. مراحل آزمایشی دقیق در زیر نشان داده شده است [۲۲ و ۴۱ و ۱۱]:

۱. اجرای اسکریپت topo و باز کردن هر ترمینال میزبان با استفاده از دستور xterm.
۲. با اجرای اسکریپت پایتون تشخیص DoS، کنترلر POX راه‌اندازی می‌شود.
۳. در هاست، hping3 اجرا شده تا با یک src_ip جعلی، ۴۵ بسته به ۱۰،۱۰،۱،۳ ارسال شود؛ از آنجایی‌که تعداد بسته‌ها از آستانه (۵۰) فراتر نرفته است، این ترافیک مسدود نخواهد شد.
۴. وقتی hping3 برای ارسال ۶۵ بسته به ۱۰،۱۰،۱،۳ با یک src_ip جعلی دیگر اجرا شد، ترافیک به‌عنوان مخرب شناسایی و مسدود می‌شود.
۵. به‌طور مشابه، ارسال یک بسته به‌طور مداوم با IP منبع یکسان بیش از ۵۰ بار نیز باعث مسدود شدن آدرس IP می‌شود [۱۵].

**۲-۶- نتیجه و بحث**

سپس در مقاله مبنا، روش‌ها مورد آزمایش قرار گرفتند تا ببینند که آیا الزامات پیشنهادی به‌عنوان هدف، پروژه را با موفقیت پشت سر گذاشته‌اند یا خیر؛ جدول ۱، نتیجه هر آزمون را نشان می‌دهد؛ از هر پنج آزمون، چهار مورد قبول شد و یک شرط وجود دارد که انجام نشد [۲۳ و ۴۱ و ۲۴].

**جدول ۱- نتیجه پروژه [۲۵ و ۴۱]**

| ID | مورد نیاز | شرح | نتیجه |
|---|---|---|---|
| ۱.۰ | توانایی تشخیص حملات DoS | الزامات عملکرد اولیه: سیستم باید بتواند حمله DoS را در زمان وقوع آن تشخیص دهد. | Pass: این روش می تواند به وضوح نشان دهد که چه زمانی یک حمله DoS رخ می دهد. |
| ۲.۰ | توانایی کاهش حملات DoS | الزامات عملکرد اولیه: سیستم باید بتواند حمله DoS را کاهش دهد. | Pass: حمله DoS را می توان با حذف بسته های حمله مسدود کرد. |
| ۳.۰ | شناسایی مهاجم DoS | نیاز زیرعملکردی: سیستم باید بتواند تعیین کند که مهاجم DoS از کدام سوئیچ بسته ها را ارسال می کند. این نیز برای کاهش حملات DoS ضروری است. | Pass: این روش می تواند سوئیچ مورد حمله را با نظارت بر هر سوئیچ شناسایی کند. |
| ۴.۰ | تعیین نوع حمله DoS | نیاز زیر عملکردی: سیستم باید توانایی تشخیص حملات DoS مختلف، از جمله سیل لایه برنامه، DDoS و DoS ناخواسته را داشته باشد ("انواع حملات انکار سرویس - استراتژی های کاهش DOS"، ۲۰۲۱). | ناموفق: روش نمی تواند نوع حمله DoS را تعیین کند. |
| ۵.۰ | استفاده از Mininet | نیاز سطح پایین: Mininet باید به عنوان پلت فرم شبیه سازی برای پروژه استفاده شود. | Pass: Mininet برای راه اندازی شبکه و شبیه سازی حمله Dos استفاده شد. |

## ۳- نتیجه‌گیری

محتوای اصلی پروژه مقاله مبنا، دارای جنبه‌های مختلف است؛ اصل حمله DoS تحلیل می‌شود و معماری شبکه تعریف‌شده نرم‌افزار (SDN) و مکانیسم پروتکل OpenFlow به‌طور خلاصه معرفی می‌شوند؛ دو روش تشخیص حمله DoS پیشنهاد و پیاده‌سازی شده است که یکی از آن‌ها پیکربندی sFlow Agent در ماشین مجازی و سپس مشاهده وضعیت نظارت در sFlow WebUI است؛ مورد دیگر، تشخیص خودکار حمله dos با شمارش اینکه آیا تعداد بسته‌ها از آستانه فراتر رفته است یا خیر؛ سپس، این پروژه دو اقدام متقابل را برای شناسایی حملات DoS پیشنهاد می‌کند؛ یکی این است که به سوییچ دستور دهید تا ترافیک DoS را با افزودن جدول جریان به سوئیچ حذف کند؛ دیگر اینکه سرویس را متوقف کنید. این کار اکثر نیازهای پروژه مربوطه را تکمیل کرده است؛ اما به‌دلیل کمبود دانش و محیط آزمایشی، این پروژه دارای محدودیت‌های زیر است:

۱. اگرچه روش ۲ می‌تواند به‌طور خودکار حملات dos را شناسایی کند، به‌دلیل تنوع وضعیت شبکه و حملات DoS، شمارش ساده بسته‌ها به‌جای استفاده از ویژگی‌های جدول جریان برای تعیین اینکه آیا ترافیک یک حمله DoS است، دارای خطای خاصی است.

۲. به‌سبب محیط آزمایشی و مشکلات نرم‌افزاری، الگوریتم تشخیص DoS با کنترل‌کننده ONOS شبیه‌سازی نشد؛ اگرچه روش‌های تشخیص و دفاع بسیاری برای حملات DoS در SDN وجود دارد، اما با رایج شدن و استفاده از شبکه‌های SDN، مطمئنا انواع جدیدی از حملات DoS ظاهر خواهند شد؛ بنابراین، تحقیق بر روی روش‌های تشخیص حمله و دفاع DoS مبتنی بر SDN بسیار ارزشمند است؛ تنها با انجام تحقیقات عمیق می‌توان محیط شبکه‌ای امن‌تری داشت.

## ۴ - مراجع


[1] Chen, Y., Zhang, P, Zhou, S. (2019). Study on SDN Technology Based on OpenFlow and Its Application Prospect. Itm-conferences.org. Retrieved 29 August 2021, from https://www.itmconferences.org/articles/itmconf/pdf/2019/02/itmconf_icicci2018_01017.pdf.

[2] Haji, S., Zeebaree, S., Saeed, R., Ameen, S., Shukur, H., & Omar, N. (2021). Comparison of Software Defined Networking with Traditional Networking. Asian Journal of Research in Computer Science, 1-18. https://doi.org/10.9734/ajrcos/2021/v9i230216

[3] HumayunKabir, M. (2021). Software-Defined Networking (SDN): A Revolution in Computer Network. Retrieved 29 August 2021, from http://www.iosrjournals.org/iosr-jce/papers/Vol15-issue5/Q0155103106.pdf.

[4] Kandoi, R., & Antikainen, M. (2015). Denial-of-service attacks in OpenFlow SDN networks. Ieeexplore.ieee.org. Retrieved 29 August 2021, from https://ieeexplore.ieee.org/abstract/document/7140489.

[5] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., & Rexford, J. (2008). OpenFlow. ACM SIGCOMM Computer Communication Review, 38(2), 69-74. https://doi.org/10.1145/1355734.1355746

[6] OpenFlow Switch Specification Version 1.3.1 (Wire Protocol 0x04). Opennetworking.org. (2012). Retrieved 29 August 2021, from https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.3.1.pdf.

[7] Software-Defined Networking: The New Norm for Networks - Open Networking Foundation. Open Networking Foundation. (2021). Retrieved 28 August 2021, from https://opennetworking.org/sdnresources/whitepapers/software-defined-networking-the-new-norm-for-networks/.

[8] Types of Denial of Service Attacks - DOS Mitigation Strategies. Developer.okta.com. (2021). Retrieved 29 August 2021, from https://developer.okta.com/books/api-security/dos/what/.

[9] What is SD-WAN? Ibm.com. (2021). Retrieved 29 August 2021, from https://www.ibm.com/services/network/sdn-versus-traditional-networking.

[10] Shin, S., Gu, G.: Attacking software-defined networks: a first feasibility study. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 165–166. ACM (2013).

[11] Lau, F., Rubin, S.H., Smith, M.H., et al.: Distributed denial of service attacks. In: 2000 IEEE International Conference on Systems, Man, and Cybernetics, vol. 3, pp. 2275–2280. IEEE (2000).

[12] Curtis, A.R., Mogul, J.C., Tourrilhes, J., et al.: DevoFlow: scaling flow management for high-performance networks. ACM SIGCOMM Comput. Commun. Rev. 41(4), 254–265 (2011). (ACM)

[13] Tootoonchian, A., Gorbunov, S., Ganjali, Y., et al.: On controller performance in software-defined networks. In: USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE) (2012).

[14] Yu, M., Rexford, J., Freedman, M.J., et al.: Scalable flow-based networking with DIFANE. ACM SIGCOMM Comput. Commun. Rev. 41(4), 351–362 (2011).

[15] Kang, M.S., Lee, S.B., Gligor, V.D.: The crossfire attack. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 127–141. IEEE (2013).

[16] Wang, H., Zhang, D., Shin, K.G.: Detecting SYN flooding attacks. In: Proceedings of the IEEE Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2002, vol. 3, pp. 1530–1539. IEEE (2002).

[17] Siris, V.A., Papagalou, F.: Application of anomaly detection algorithms for detecting SYN flooding attacks. Comput. Commun. 29(9), 1433–1442 (2006).

[18] Braga, R., Mota, E., Passito, A.: Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: 2010 IEEE 35th Conference on Local Computer Networks (LCN), pp. 408–415. IEEE (2010).

[19] Seungwon, S., Yegneswaran, V., Porras, P., Gu, G.: AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. In:

Target Defense. Creating Asymmetric Uncertainty for Cyber Threats, vol. 54, pp. 1–28. Springer, New York (2007).
[30] Song, H.: Protocol-oblivious forwarding: unleash the power of SDN through a future-proof forwarding plane. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 127–132. ACM (2013).
[31] Feng, Y., Guo, R., Wang, D., Zhang, B.: Research on the active DDoS filtering algorithm based on IP flow. In: 2009 Fifth International Conference on Natural Computation, pp. 628–632. IEEE (2009).
[32] Kohonen, T.: The self-organizing map. Proc. IEEE 78(9), 1464–1480 (1990) 28. Broder, A., Mitzenmacher, M.: Network applications of bloom filters: a survey. Internet Math. 1(4), 485–509 (2004).
[33] K. Giotis, G. Androulidakis, V. Maglaris, A scalable anomaly detection and mitigation architecture for legacy networks via an OpenFlow middlebox, Int. J. Appl. Eng. Res. 9 (2015) 1958–1970.
[34] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, V. Maglaris, Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments, Comput. Netw. 62 (2014) 122–136.
[35] P. Dong, X. Du, H. Zhang, T. Xu, A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows, in: 2016 IEEE International Conference on Communications, ICC 2016, 2016, pp. 1–6.
[36] R.M. Thomas, D. James, DDOS detection and denial using third party application in SDN, in: 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017, IEEE, 2018, pp. 3892–3897.
[37] S. Lim, J. Ha, H. Kim, Y. Kim, S. Yang, A SDN-oriented DDoS blocking scheme for botnet-based attacks, in: International Conference on Ubiquitous and Future Networks, ICUFN, 2014, pp. 63–68.
[38] L. von Ahn, M. Blum, N.J. Hopper, J. Langford, CAPTCHA: Using hard AI problems for security, in: International Association for Cryptologic Research, 2003, pp. 294–311.
[39] T. Lukaseder, A. Hunt, C. Stehle, D. Wagner, R. Van Der Heijden, F. Kargl, an extensible host-agnostic framework for SDN-assisted DDoS-mitigation, in: Conference on Local Computer Networks, LCN, 2017, pp. 619–622.
[40] S. Wang, K.G. Chavez, S. Kandeepan, SECO: SDN secure controller algorithm for detecting and defending denial of service attacks, in: 5th International Conference on Information and Communication Technology, ICoIC7 2017, 2017, pp. 1–6.
Proceedings 20th ACM Conference on Computer and Communications Security (CCS 2013), Berlin, Germany, November 2013.
[20] Floodlight. http://floodlight.openflowhub.org. Accessed December 2013 15. Sun, H., Lui, J.C.S., Yau, D.K.Y.: Defending against low-rate TCP attacks: dynamic detection and protection. In: Proceedings of the 12th IEEE International Conference on Network Protocols, 2004, ICNP 2004, pp. 196–205. IEEE (2004).
[21] Kuzmanovic, A., Knightly, E.W.: Low-rate TCP-targeted denial of service attacks and counter strategies. IEEE/ACM Trans. Network (TON) 14(4), 683–696 (2006).
[22] Wang, F., Uppalli, R., Killian, C.: Analysis of techniques for building intrusion tolerant server systems. In: 2003 IEEE Military Communications Conference, 2003, MILCOM 2003, vol. 2, pp. 729–734. IEEE (2003).
[23] Nguyen, Q.L., Sood, A.: A comparison of intrusion-tolerant system architectures. IEEE Secur. Priv. 9(4), 24–31 (2011).
[24] Nguyen, Q.L., Sood, A.: Designing SCIT architecture pattern in a cloud-based environment. In: 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 123–128. IEEE (2011).
[25] Hardman, O., Groat, S., Marchany, R., et al.: Optimizing a network layer moving target defense for specific system architectures. In: Proceedings of the ninth ACM/IEEE Symposium on Architectures for Networking and Communications Systems, 2013, pp. 117–118. IEEE Press (2013).
[26] Jackson, T., Homescu, A., Crane, S., Larsen, P., Brunthaler, S., Franz, M.: Diversifying the software stack using randomized NOP insertion. In: Jajodia, S., Ghosh, A.K., Subrahmanian, V.S., Swarup, V., Wang, C., Sean Wang, X. (eds.) Moving Target Defense II. Application of Game Theory and Adversarial Modeling, vol. 100, pp. 151–173. Springer, New York (2013).
[27] Dixit, A., Hao, F., Mukherjee, S., et al.: Towards an elastic distributed SDN controller. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 7–12. ACM (2013).
[28] Jafarian, J.H., Al-Shaer, E., Duan, Q.: Openflow random host mutation: transparent moving target defense using software defined networking. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, 2012, pp. 127–132. ACM (2012)
[29] Manadhata, P.K., Wing, J.M.: A formal model for a system's attack surface. In: Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Sean Wang, X. (eds.) Moving

[41] Yalan Zhang, Huiyun Ning (2022), DoS Attack Detection and Defense on SDN Controller, Journal of Computer Science and Technology Studies.
[42] Duohe Ma, Zhen Xu, Dongdai Lin (2015), Defending Blind DDoS Attack on SDN Based on Moving Target Defense, Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2015 J. Tian et al. (Eds.): SecureComm 2014, Part I, LNICST 152, pp. 463–480.