مجله علمی - پژوهشی "پدافند الکترونیکی و سایبری"
سال چهارم، شماره ۴، زمستان ۱۳۹۵، ص ۷۸-۶۳

محافظت در مقابل حملات سیل‌آسا در شبکه‌های سنتی با همکاری ناهمگن سرویس‌دهنده و کنترل‌کننده مبتنی بر نرم افزار(SDN)

رضا محمدی فر[۱]، عباسعلی رضایی[۲*]

۱- دانشجوی کارشناسی ارشد، دانشگاه پیام نور مرکز بین الملل قشم
۲- استادیار، گروه مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه پیام نور، تهران
۳- (دریافت: ۹۴/۰۹/۲۷، پذیرش: ۹۵/۰۲/۱۴)

چکیده

حملات سیل‌آسا از مخرب‌ترین حملات تکذیب سرویس توزیعی (DDos) هستند. در این نوع حملات، مهاجمان با ارسال سیل بسته‌های نامعتبر مقدار زیادی از منابع را اشغال کرده و ارائه خدمات را با چالش جدی مواجه می‌کنند. معماری سلسله مراتبی و ضعف‌های متعدد در ساختار قراردادهای ارتباطی در شبکه‌های سنتی موجب شده تا دیواره‌های آتش مکانیسم موثر و منسجمی را برای مقابله ارائه نکنند. با ظهور شبکه‌های مبتنی بر نرم‌افزار (SDN) امیدهای تازه‌ای برای حل مشکلات ساختاری و امنیتی در شبکه‌های سنتی مطرح شده است. در این مقاله روشی ناهمگن بر اساس همکاری سرویس‌دهنده سنتی و کنترل‌کننده مبتنی بر نرم‌افزار (SDN) را برای مقابله با حملات سیل‌آسا پیشنهاد می‌کنیم. در این روش ماژول تشخیص حملات در سرویس‌دهنده و ماژول پاسخ در کنترل‌کننده (SDN) واقع است. برای شبیه‌سازی روش ناهمگن از امولاتور MiniNet به همراه کنترل‌کننده Pox استفاده می‌کنیم. سپس مدل شبیه‌سازی‌شده را ارزیابی کرده و آشکار می‌کنیم که روش پیشنهادی علاوه بر دفع حملات در شبکه‌های سنتی برتری‌هایی را از نظر میزان بار رایانشی و مدت زمان پاسخ‌گویی نسبت به سایر روش‌های مبتنی بر نرم‌افزار ارائه می‌کند.

واژه‌های کلیدی: شبکه‌های مبتنی بر نرم‌افزار، قرارداد OpenFlow، حملات سیل‌آسا

۱- مقدمه

یکی از شاخصه‌های مهم امنیت اطلاعات[1] دردسترس‌بودن منابع است. یعنی باید از کارکرد صحیح دستگاه‌های پردازش اطلاعات و کانال‌های ارتباطی اطمینان حاصل کرد. به‌عبارت‌دیگر افراد مجاز باید بتوانند در هر زمان که لازم باشد به منابع دسترسی داشته باشند. حملات تکذیب سرویس از جمله خطراتی است که دسترسی به اطلاعات را با چالش جدی مواجه می‌کند. مهاجمان با انجام این حملات سعی می‌کنند با اشغال میزان قابل‌توجهی از منابع در دسترس، باعث اختلال در سرویس‌ها شوند[۸]. حملات سیل‌آسا[2] ازجمله خطرناک‌ترین حملات تکذیب سرویس هستند که به دلیل ضعف در قراردادهای ارتباطی، شبکه را آماج حملات قرار می‌دهند. به‌عنوان نمونه حملات SynFlood که در لایه انتقال رخ می‌دهد بیش از ۹۰ درصد حملات Dos را به خود اختصاص داده است [۳].

ساختار سلسله مراتبی، ضعف در قراردادهای ارتباطی و عدم وجود فرماندهٔ متمرکز در شبکه‌های امروزی مقابله با این حملات را به چالش جدی مبدل کرده است [۲- ۱].

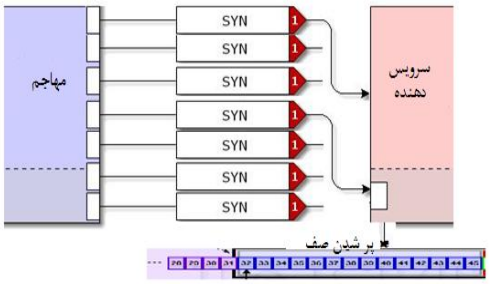
ضعف در قراردادهای ارتباطی و عدم وجود فرماندهٔ متمرکز در شبکه‌های امروزی مقابله با این حملات را به چالش جدی مبدل کرده است [۲- ۱].

به‌عنوان مثال یکی از مهم‌ترین مشکلات در قرارداد[3] کنترل انتقال(TCP)، ضعف در عملکرد دست دهی سه مرحله‌ای است. این مشکل ناشی از عدم تایید هویت آغازکننده ارتباط، توسط سرویس‌دهنده است. در حالت عادی، سرویس‌دهنده پس از دریافت بسته SYN مقداری حافظه برای دنبال‌کردن جلسه ارتباطی تخصیص می‌دهد و منتظر می‌ماند تا ارتباط آغاز شود. مهاجمان با استفاده از این ضعف اقدام به ارسال سیل بسته‌های نامعتبر با علامت SYN=1 می‌کنند و حافظه سرویس‌دهنده را مملو از درخواست‌های نامعتبر می‌کنند (شکل ۱). با پرشدن

* رایانامه نویسنده مسئول: A_rezaee@pnu.ac.ir

1- Information Security
2- Flood Attack
3- Protocol

حافظه، سرویس‌دهنده مجبور به رهاکردن درخواست‌های جدید می‌شود [۵- ۴] و از مدار سرویس‌دهی خارج می‌شود.



شکل (۱). پرشدن حافظه سرویس‌دهنده

ضعف بعد در ساختار قرارداد داده‌نگار کاربر (UDP) است. این قرارداد یک ارتباط بدون اتصال را به وجود می‌آورد؛ یعنی ارتباط به‌صورت یک‌طرفه و در یک مسیر از مبدأ به مقصد و بدون درنظرگرفتن وضعیت گیرنده برقرار می‌شود [۶]. در حملات UDP بسته‌ها از پورت‌های تصادفی به سمت ماشین قربانی ارسال می‌شوند و باعث اشباع پهنای باند ماشین هدف شده و از ارائه سرویس به کاربران مجاز جلوگیری می‌کند. دیواره‌های آتش قادر به توقف این‌گونه حملات نیستند [۹-۷].

سیر تکاملی اینترنت، پدیدآمدن سرویس‌های ابری، مجازی‌سازی و تغییرات در الگوهای مصرفِ داده، موجب آشکارشدن ضعف‌های موجود در شبکه‌های سنتی شده است. با ظهور شبکه‌های مبتنی بر نرم‌افزار امیدهای تازه‌ای برای حل مشکلات ساختاری در شبکه‌های سنتی به وجود آمد [۱۰]. در این مقاله ضمن معرفی شبکه‌های مبتنی بر نرم‌افزار، ایده‌های مطرح‌شده برای مقابله با حملات سیل‌آسا را بررسی می‌کنیم؛ همچنین با تحلیل نقاط ضعف و قوت و به پشتوانه پژوهش‌های قبلی، روشی مبتنی بر همکاری سرویس‌دهنده و کنترل‌کننده SDN[1] برای مقابله با حملات سیل‌آسا در شبکه‌های سنتی ارائه می‌دهیم.

۲- پیش‌زمینه

در این بخش خلاصه‌ای از اطلاعات مورد نیاز را جهت درک بهتر ادامه مقاله ارائه می‌کنیم. در بخش ۲-۱ ساختار شبکه‌های سنتی و محدودیت‌ها برای مقابله با حملات سیل‌آسا را بررسی می‌کنیم. سپس در بخش ۲-۲ شبکه‌های مبتنی بر نرم‌افزار و در بخش ۲-۳ قرارداد OpenFlow را معرفی می‌کنیم.

۲-۱- ساختار شبکه‌های سنتی و محدودیت‌ها در مقابله با حملات سیل‌آسا

در ساختار سنتی، دستگاه‌های موجود در شبکه به‌صورت درونی کنترل می‌شوند. هر دستگاه شامل نرم‌افزار مختص به خود است. این نرم‌افزار وظیفه کنترل و تصمیم‌گیری در دستگاه را به عهده دارد. بنابراین هر دستگاه دارای عملکردی مستقل است. ازاین‌رو این نوع معماری را عمودی (سلسله‌مراتبی) می‌نامند. دفاع در مقابل حملات سیل‌آسا در این شبکه‌ها برای چندین دهه مورد مطالعه قرار گرفته است. در مقالات شماره [۲۱-۱۱]، بسیاری از روش‌های مقابله با حملات به روش سنتی بررسی شده‌اند. از جمله ایده‌های مطرح برای مقابله در برابر حملات سیل‌آسا در این شبکه‌ها، استفاده از دیواره‌های آتش یا دستگاه‌های فیلتر حملات است. بر طبق تحقیقات صورت‌گرفته، در حملاتی که ضعف در قراردادها را مبنای حمله قرار می‌دهند، دیواره‌های آتش و دستگاه فیلتر حملات، راه‌کارهای مؤثری برای مقابله با حملات ارائه نمی‌کنند [۱۳]. البته روش‌هایی نیز برای جبران کاستی‌های موجود در قراردادهای ارتباطی پیشنهاد شده است. به‌عنوان مثال روش SynCookie که برای پوشش ضعف‌های موجود در قرارداد کنترل انتقال در مقابله با حملات سیل Syn پیشنهاد شده است [۱۴]. در این روش به‌جای تخصیص حافظه، یک بسته برای سرویس‌گیرنده ارسال می‌شود. درصورتی‌که سرویس‌گیرنده به بسته پاسخ دهد، حافظه‌ای جهت ارتباط تخصیص داده می‌شود. این روش امروزه در هسته سیستم عامل لینوکس به کار می‌رود. با وجود استفاده از روش SynCookie درصورتی‌که مهاجم سیل زیادی از بسته‌ها به را به سمت سرویس‌دهنده ارسال کند در حمله موفق خواهد بود [۱۵]. همچنین هر پروسه تائید هویت به لحاظ بار محاسباتی می‌تواند یک حفره عدم سرویس‌دهی به شمار آید. معماری شبکه‌های سنتی به دلیل پیچیدگی مدیریت و ضعف در قراردادهای ارتباطی نمی‌توانند راه‌کارهای مؤثر و منسجمی را برای مقابله با حملات سیل‌آسا ارائه دهد.

۲-۲- شبکه مبتنی بر نرم‌افزار (SDN)

در معماری شبکه‌های مبتنی بر نرم‌افزار، سطوح داده[2] و کنترل[3] از یکدیگر جدا شده‌اند. ساختار معماری شبکه افقی، هوشمندتر و کنترل‌پذیرتر شده است. یکی از ایده‌های بزرگ شبکه‌های مبتنی بر نرم‌افزار این است که دستگاهی به نام کنترل‌گر با همه دستگاه‌های موجود در یک دامنه[4] ارتباط مستقیم داشته باشد، از معماری شبکه آگاه باشد و شبکه را از یک نقطه مرکزی برنامه‌ریزی کند. یک کنترل‌کننده SDN مدل برنامه‌ریزی شبکه را از حالت توزیع‌شده به حالت متمرکز تبدیل می‌کند [۱۶]. برنامه‌ریزی متمرکز شبکه، ویژگی ارزشمندی است که می‌توان به‌وسیله آن انواع سیاست‌گذاری‌های امنیتی را

1- Software Defined Network

2- Data Plane
3- Control Plane
4- Domain

به‌صورت کلان انجام داد.

با استفاده از شبکه‌های مبتنی بر نرم‌افزار، دیگر نیازی نیست مدیران فنی هر یک از دستگاه‌های موجود در شبکه را از طریق پایانه، کنترل و مدیریت کنند. کنترل‌گر شبیه لایه‌ای از میان‌افزار عمل خواهد کرد که اجزای لایه فیزیکی شبکه نظیر مسیریاب‌ها، سوئیچ‌ها، دیواره آتش و تعدیل‌کننده بار[1] را از بخش نرم‌افزاری تفکیک می‌کند. در حقیقت کنترل‌کننده می‌تواند به نحوی شایسته شبکه را به‌صورت مرکزی رصد کرده و با استفاده از اطلاعات دریافت شده و فرمان‌های انعطاف‌پذیر، آن را مدیریت کند. در شکل (۲) ساختار شبکه‌های مبتنی بر نرم‌افزار را مشاهده می‌کنیم. بنیان شبکه‌های مبتنی بر نرم‌افزار بر پایه قرارداد OpenFlow شکل‌گرفته است [۱۷].

۲-۳- قرارداد OpenFlow

OpenFlow نخستین واسط ارتباطی استانداردی است که در معماری شبکه‌های مبتنی بر نرم‌افزار، بین لایه کنترل و ارسال داده تعریف می‌شود [۱]. به‌صورت کلی شبکه‌های مبتنی بر نرم‌افزار با استفاده از قرارداد OpenFlow به مهندسان شبکه امکان می‌دهد از زبان‌های برنامه‌نویسی سطح بالا برای تهیه دستورالعمل‌های سطح پایین و انتقال به مسیریاب‌ها و سوئیچ‌ها استفاده کنند. در شبکه‌های مبتنی بر نرم‌افزار، کنترل‌گر مرکزی تمامی قوانین شبکه را نگهداری و برحسب نیاز دستورات را از طریق قرارداد OpenFlow صادر می‌کند [۱۸]. این قرارداد در ابتدا کنترل‌کننده[2] مرکزی را تعریف و سپس بررسی می‌کند. یعنی این کنترل‌کننده، چگونه می‌تواند به‌صورت امن به دستگاه‌های شبکه متصل شود و آن‌ها را کنترل کند؛ در شکل (۳) نمایی از این قرارداد مشاهده می‌شود. پس نخستین گام به‌سوی شبکه‌های مبتنی بر نرم‌افزار تعریف یک قرارداد پیام‌رسانی، میان کنترل‌کننده مرکزی و سوئیچ سخت‌افزاری است که وظیفه هدایت اطلاعات را بر عهده دارد. نسخه اول قرارداد OpenFlow در دسامبر سال ۲۰۰۹ تعریف و منتشر شد که از سه جزء اصلی کنترل‌کننده، کانال‌های امن و جدول جریان تشکیل‌شده است. در سال ۲۰۱۱ نسخه جدید نیز عرضه شد که علاوه بر اجزای قبلی شامل دو جزء جدول گروهی و پایپ لاین است [۱۰، ۱۸، ۱۹ و ۲۰]. تاکنون ۴ نسخه قرارداد OpenFlow عرضه شده که هر نسخه شامل قابلیت‌های بیشتر نسبت به نسخه قبل است؛ در جدول (۱) ویژگی‌های هر نسخه قابل مشاهده است [۲۱].

امروزه در میان مراکز داده با مقیاس بزرگ، علاقه شدیدی به OpenFlow وجود دارد. شرکت‌های بزرگ سوئیچ مبتنی بر این قرارداد را طراحی می‌کنند و بسیاری از این قرارداد اعلام حمایت کرده‌اند. شرکت گوگل این قرارداد را در مراکز داده خود به کار بسته است [۲۶-۲۲].

۳- کارهای مرتبط

مطالعاتی پیرامون روش‌هایی مقابله با حملات تکذیب سرویس، توسط شبکه‌های مبتنی بر نرم‌افزار صورت پذیرفته است. در [۲۷] طرح جهش سرویس‌دهنده به‌صورت تصادفی با استفاده از OpenFlow در شبکه‌های مبتنی بر نرم‌افزار ارائه شده است. در این طرح استفاده از آی‌پی‌های ایستا برای سرویس‌دهنده‌ها در شبکه به‌عنوان چالش مطرح‌شده است؛ بدین‌صورت که استفاده از آی‌پی‌های ایستای فعال در یک دامنه یا شبکه باعث سرازیرشدن انواع حملات پویش[3] و تکذیب سرویس می‌شود. برای رفع این مشکل آن‌ها روشی پیشنهاد دادند. در این روش آی‌پی‌های واقعی با آی‌پی‌های مجازی جایگزین می‌شود و از طریق سامانه، نام دامنه قابل‌دست‌یابی است. آی‌پی‌های واقعی فقط توسط افراد مجاز قابل رؤیت است. این روش از شناسایی منابع فعال در شبکه جلوگیری می‌کند.

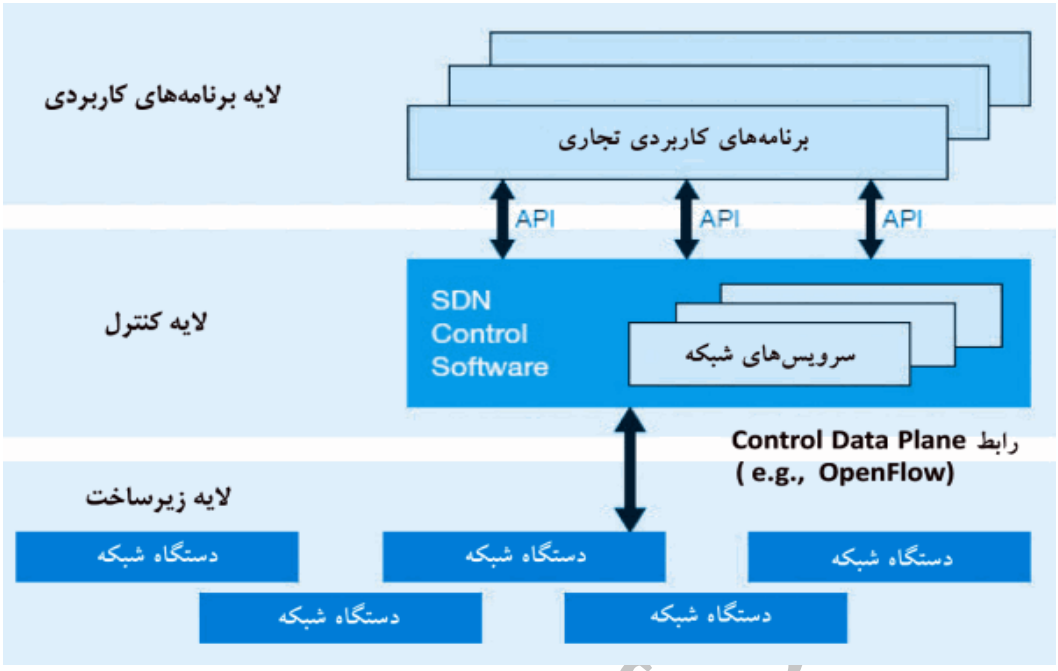
در [۲۸] راهکاری برای امنیت در هسته کنترل‌کننده جهت تشخیص مشکلات امنیتی جریان‌ها در سوئیچ پیشنهاد شده است. این راه‌کار FortNox نام دارد و بر پایه کنترل‌کننده Nox [۲] گسترش‌یافته است. در این روش، هسته امنیتی به کنترل‌کننده اضافه شده است که جریان‌های عبوری را کنترل می‌کنند و تناقضات و مشکلات در جریان‌ها را بررسی می‌کند . همچنین جهت احراز هویت برنامه‌ها از امضای دیجیتال استفاده شده و دارای سربار ۷ میلی‌ثانیه‌ای جهت چک‌کردن جریان‌ها نسبت به کنترل‌کننده Nox است.

در [۲۹] روشی برای اعتبارسنجی آدرس مبدأ بر مبنای کنترل‌کننده Nox پیشنهاد شده است. در این روش هر بسته‌ای از خارج از شبکه قصد ورود به شبکه را داشته باشد به سمت این کنترل‌کننده هدایت‌شده و عملیات اعتبارسنجی منبع ارسال بسته، توسط کنترل‌کننده انجام می‌شود. اگر منبع قابل اطمینان بود اجازه عبور ترافیک صادر می‌شود. در غیر این صورت از عبور بسته جلوگیری می‌شود. این روش می‌تواند از عملیات جعل آی پی در حملات سیل‌آسا جلوگیری کند.

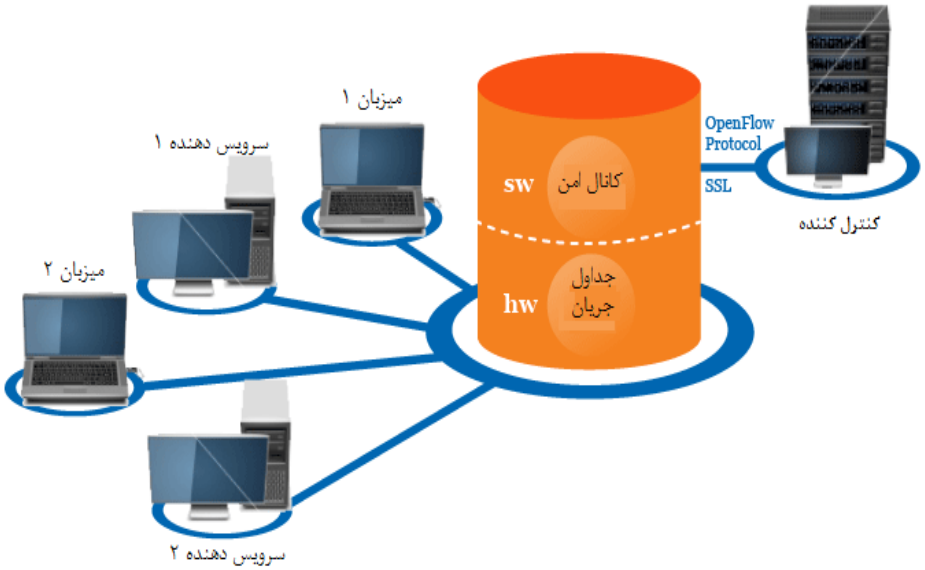
1- LoadBlancer

2- Controller

3- Scaning



شکل (۲). ساختار یک شبکه مبتنی بر نرم‌افزار [۱۷]



شکل (۳). نمایی از قرارداد OpenFlow [۲۰]

جدول(۱). مشخصه‌های OpenFlow [۲۱]

نسخه	1.0.0	1.1.0	1.2	1.3.0
استقرار وسیع	بله	خیر	خیر	خیر
جدول جریان	تک جدول جریان	چند جدول جریان	چند جدول جریان	چند جدول جریان
پشتیبانی از MPLS	خیر	بله	بله	بله
پشتیبانی از IPV6	خیر	خیر	بله	بله
جدول گروهی	خیر	بله	بله	بله (جداول منعطف)
ارتباط با چند کنترلر	خیر	خیر	بله	بله (ارتباط همزمان)

در [۳۰] روشی بر اساس کنترل‌کننده Nox برای تشخیص حملات DDos پیشنهاد شده است. در این روش الگوریتمی جهت کشف حملات Dos پیشنهاد شده است که با استفاده از شبکه‌های عصبی بدون ناظر، بر اساس ویژگی ترافیک ورودی حملات Dos را تشخیص می‌دهد. در این طرح روش‌های پاسخ به حملات بررسی نشده‌اند.

در [۱۳] روش محافظت وب سرورها از حملات SynFlood بر مبنای هسته کنترل‌کننده Pox پیشنهاد شده است. در این روش کنترل‌کننده‌ای به نام OPERETTA معرفی شده که هر بسته Syn را اعتبارسنجی نموده و بعد از اعتبارسنجی، اجازه نصب جریان و آغاز ارتباط را می‌دهد. این روش بسیار شبیه به SynCookie در شبکه‌های سنتی است. در این روش دو سناریو مقابله با حملات سیل‌آسا به‌صورت مرکزی و محلی مورد بررسی قرارگرفته است.

در [۳۱] روشی برای مقابله با حملات ترافیکی توزیع‌شده در محیط‌های ابری بر مبنای کنترل‌کننده Floodlight پیشنهاد شده است. این کنترل‌کننده از دو ماژول تشخیص و دفاع تشکیل‌شده است. ماژول تشخیص حملات را در ابر شناسایی و به ماژول مقابله اطلاع می‌دهد. سپس اقدامات لازم صورت می‌پذیرد. این روش نخستین پیشنهاد مقابله با حملات در محیط‌های ابری به کمک شبکه‌های مبتنی بر نرم‌افزار است.

همان‌گونه که بررسی شد روش‌های متعددی برای شناسایی و مقابله با حملات توسط شبکه‌های مبتنی بر نرم‌افزار پیشنهاد شده است. بر طبق مطالعات صورت‌گرفته مقالات [۱۳، ۳۰ و ۳۱] به ارائه پیشنهادیی برای تشخیص و مقابله با حملات تکذیب سرویس پرداخته است. در این سه طرح، ماژول تشخیص و دفاع به صورت متمرکز در کنترل‌کننده قرار دارد. یکی از اشکالات مهمی که به این طرح‌ها وارد است مقیاس‌پذیری[1] و افزایش بار[2] کنترل‌کننده در زمان حملات سیل‌آسا است. زمانی که حملات آغاز می‌شوند، تعداد زیادی از بسته‌ها به سمت سرویس‌دهنده سرازیر می‌شوند. کنترل‌کننده باید تمام بسته‌ها را تحلیل و اقدامات لازم را صورت دهد. در چنین حالتی بار رایانشی زیادی بر روی کنترل‌کننده تحمیل می‌شود. ممکن است به دلیل بار تحمیل‌شده، کنترل‌کننده در مدیریت و کنترل درخواست‌های شبکه با شکست مواجه شود [۳۳].

دومین مشکل در روش‌های فوق تأخیر در ارسال جریان‌های عادی شبکه به کنترل‌کننده است . احراز صحت بسته‌ها توسط کنترل‌کننده می‌تواند باعث ایجاد تأخیر در پاسخگویی به جریان‌های مجاز شود و چون اکثر نظریه شبکه‌های مبتنی بر نرم‌افزار جریان محور هستند تأخیر اضافی به وجود آمده از نگرانی‌ها مهم به شمار می‌رود [۲۱, ۳۲] .

۴- راه‌کار پیشنهادی دفاع ناهمگن مبتنی بر نرم‌افزار

در روش‌های تشخیص و مقابله متمرکز، نگرانی‌هایی در رابطه با میزان بار رایانشی و تأخیر در پاسخگویی به جریان‌های مجاز ارسالی به کنترل‌کننده مطرح است. ما برای دفع حملات سیل‌آسا و رفع نگرانی‌های فوق روشی به نام دفاع ناهمگن را پیشنهاد می‌کنیم. در این روش برای کاهش بار رایانشی و پیشگیری از تأخیر[3] در جریان‌های مجاز ارسالی به کنترل‌کننده، اقدام به جداسازی بخش تشخیص و مقابله با حملات نموده‌ایم.

در حقیقت روش دفاع ناهمگن روشی بر اساس همکاری سرویس‌دهنده شبکه و کنترل‌کننده مبتنی بر نرم‌افزار با هدف مقابله با حملات سیل‌آسا است. این روش دارای دو مزیت نسبت به روش‌های متمرکز است:

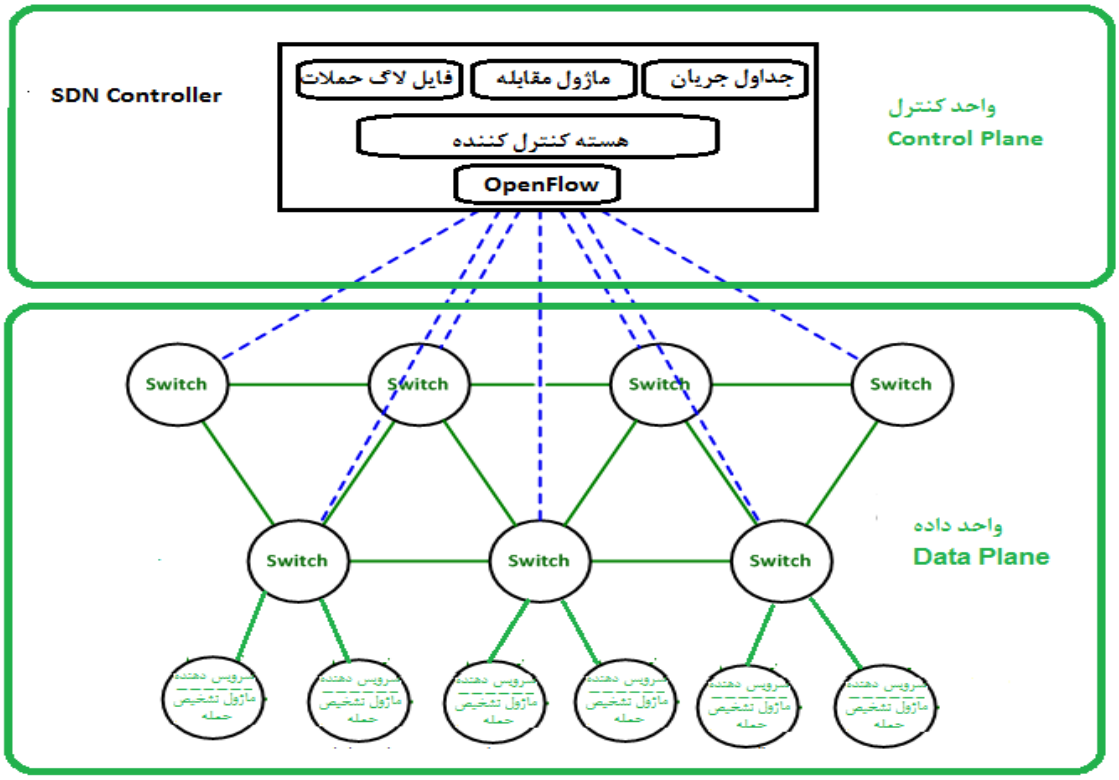
۱- به دلیل واقع‌بودن سیستم تشخیص حملات در سرویس‌دهنده، هیچ بسته‌ای برای تشخیص حمله در کنترل‌کننده معطل نمی‌شود. این مزیت باعث کاهش تأخیر در پاسخ‌گویی به درخواست‌های مجاز می‌شود.

۲- بار رایانشی اضافی ناشی از تحلیل بسته‌ها جهت تشخیص حملات بر کنترل‌کننده تحمیل نمی‌شود.

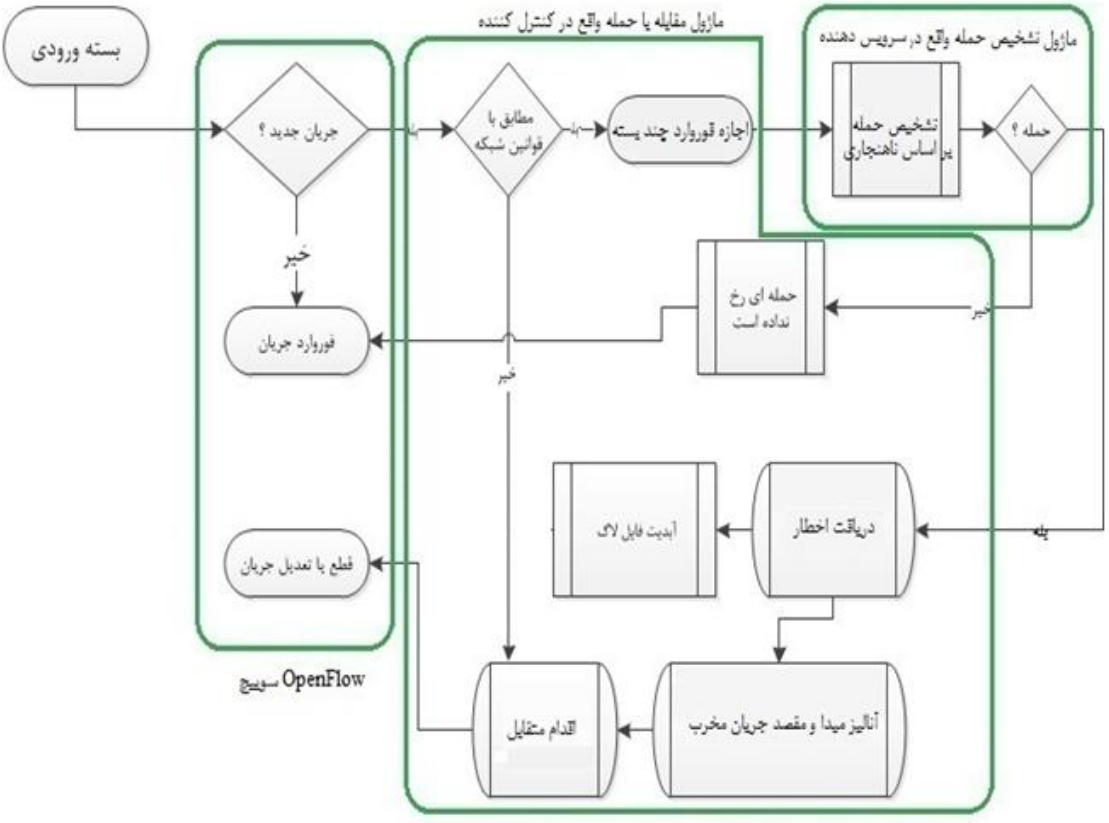
در شکل (۴) نمایی کلی از معماری روش دفاع ناهمگن مقابله با حملات سیل‌آسا آورده شده است.

روش دفاع ناهمگن از دو ماژول تشخیص و مقابله با حملات تشکیل شده است. ماژول تشخیص حملات در سرویس‌دهنده شبکه قرار دارد و برای تشخیص حملات از روش‌های مبتنی بر ناهنجاری استفاده می‌کند. ماژول مقابله با حملات در کنترل‌کننده واقع‌شده است و بر روی هسته کنترل‌کننده Pox [۱۰، ۱۸ و ۳۶] اجرا می‌شود. در شکل (۵) فرایند حرکت بسته در روش دفاع ناهمگن ارائه شده است. در این فرایند هر بسته برای برقراری ارتباط و احراز صحت، ۳ فاز را طی می‌کند. فرآیند دفاع ناهمگن در سامانه پیشنهادی به سه فاز تقسیم می شود.

1- Scalability
2- Load
3- Delay



شکل (۴). نمای کلی از معماری روش دفاع ناهمگن



شکل (۵). فرآیند حرکت بسته در روش دفاع ناهمگن

این فازها عبارتند از:

۱- فاز درخواست ارتباط؛

۲- فاز تحلیل و گزارش رخداد؛

۳- فاز تصمیم و اقدام.

۴-۱- فاز درخواست ارتباط

اولین فاز در الگوریتم دفاع ناهمگن، درخواست ارتباط است. در این فاز، بسته به سوئیچ OpenFlow وارد می‌شود. پس از ورود، سوئیچ، جداول جریان موجود در خود را چک می‌کند. اگر بسته ورودی در جداول جریان موجود باشد، اجازه عبور جریان به سمت مقصد صادر می‌شود. در غیر این صورت (اگر بسته ورودی مربوط به جریان جدیدی باشد) برای کسب تکلیف به سمت کنترل‌کننده فرستاده می‌شود (شکل (۶)).

بعد از ورود بسته به کنترل‌کننده، بررسی‌های ابتدایی مانند تشخیص آدرس مقصد، نوع قرارداد و... انجام می‌شود. درصورتی‌که بسته در چارچوب قوانین جاری شبکه تشخیص داده شود، کنترل‌کننده اجازه عبور چند بسته از جریان مربوطه را به سمت سرویس‌دهنده صادر می‌کند (شکل (۷)).

معمولاً سامانه‌های تشخیص حملات نمی‌توانند انحراف را از روی یک بسته تشخیص دهند. به همین دلیل کنترل‌کننده درخواست ورود چند بسته از جریان را به سمت سیستم تشخیص حمله در سرویس‌دهنده صادر می‌کند.

۴-۲- فاز تحلیل و گزارش رخداد

بعد از فاز درخواست ارتباط، نوبت به فاز تحلیل و گزارش رخداد می‌رسد. در این فاز بسته‌ها به‌وسیله ماژول تشخیص واقع در سرویس‌دهنده حملات، مورد ارزیابی قرار می‌گیرند. روش‌های متعددی برای تشخیص حملات سیل‌آسا ارائه شده است. در این پژوهش قصد نداریم تا شیوه نوینی برای تشخیص ارائه کنیم، بلکه مطالعاتی انجام داده‌ایم تا بهترین شیوه را بر طبق دانش خود برای استفاده در ماژول تشخیص حملات انتخاب کنیم.

طبق تحقیقات انجام‌شده، روش‌های مبتنی بر ناهنجاری دقیق‌تر از روش‌های مبتنی بر امضا در تشخیص حملات عمل می‌کنند [۳۱]. از سوی دیگر، در تشخیص به شیوه ناهنجار، وجود چند بسته برای شناسایی حملات کافی است؛ اما در روش مبتنی بر امضا نیاز به تمام بسته‌های مربوط به آن جریان است که این امر امکان‌پذیر نیست [۳۱]. بنابراین ما از روش تشخیص مبتنی بر ناهنجاری استفاده می‌کنیم. در [۳۵] چندین پارامتر برای تشخیص حملات پیشنهاد شده است. انحراف قابل‌توجه از این پارامترها می‌تواند نشانه‌ای از حمله باشد. انحراف معیار در ترافیک را می‌توان با استفاده از مفهوم آنتروپی اندازه گرفت. ابزارهای حملات سیل‌آسا معمولاً از بسته‌های خاصی جهت حملات استفاده می‌کنند. بالارفتن میزان بسته‌های یکسان هم می‌تواند نشانه‌ای از حمله باشد. ماژول تشخیص حملات با توجه به ورود چند بسته می‌تواند انحراف در روند عادی ترافیک را اندازه‌گیری کنند [۳۶].

بعد از تحلیل بسته‌ها توسط ماژول تشخیص حملات درصورتی‌که انحرافی تشخیص داده نشود. گزارش عادی عدم حمله به کنترل‌کننده ارسال می‌شود. سپس کنترل‌کننده با دستور به سوئیچ، اجازه عبور جریان مربوطه را به سوئیچ واگذار می‌کند. اما درصورتی‌که انحرافی در بسته تشخیص داده شود. ماژول تشخیص حملات، گزارش رخداد را به کنترل‌کننده ارسال می‌کند و کنترل‌کننده وارد فاز تصمیم و اقدام می‌شود.

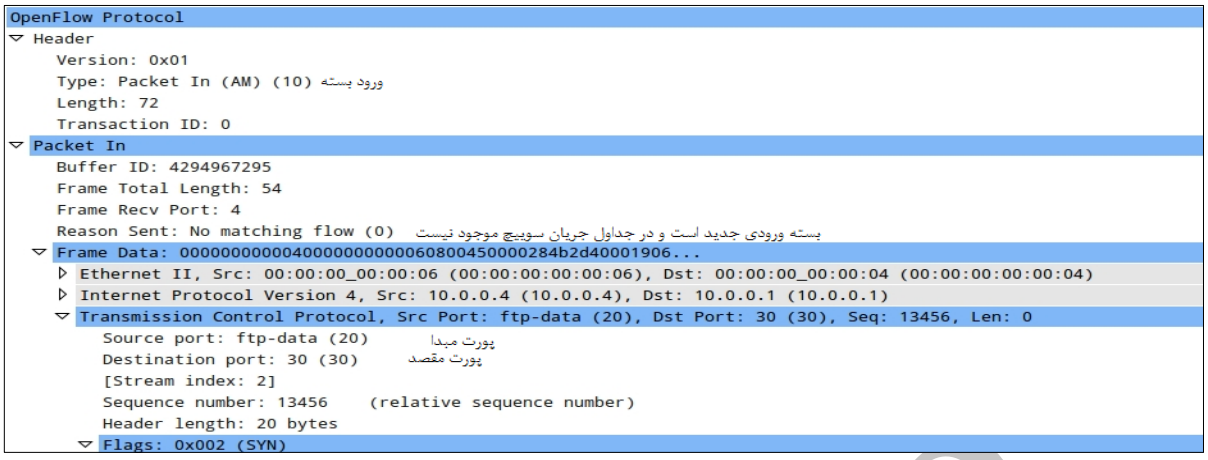
۴-۳- فاز تصمیم و اقدام

پس از دریافت گزارش رخداد[1] توسط کنترل‌کننده، ماژول مقابله با حملات وارد مدار می‌شود و با به‌روزکردن فایل لاگ و تحلیل مبدأ و مقصد جریان، دستوراتی را برای مقابله با حمله به بخش اقدام متقابل ارسال می‌کند.

در این ماژول برای کدنویسی از کتابخانه Openflow_01 استفاده شده است که حامل قواعد ارتباطات سیمی است. یک روش ساده برای متوقف‌کردن حملات سیل‌آسا، جلوگیری از ایجاد جداول جریان برای آغازگر حمله است. یعنی پس از دریافت گزارش حملات باید رخداد Packet_In مربوط به آن جریان خاص را متوقف[2] و از ایجاد جدول جریان برای آن رخداد در مبدأ حمله جلوگیری کنیم. در شکل (۸) قطعه کد مربوط به توقف رخداد حمله قابل مشاهده است. باید به این نکته توجه داشت تنها راه مقابله با حملات جلوگیری از ایجاد جداول جریان نیست. به‌عنوان مثال می‌توان با ایجاد تغییرات در فیلد انطباق، سیاست خاصی را برای یک جریان اعمال کرد.

1- Event

2- Halt



شکل (۶). نمونه از بسته ارسالی به سمت کنترل‌کننده

```
                //ایجاد رویداد گوش فرا دادن به بسته ورودی
def launch ():
    core.openflow.addListenerByName("PacketIn", _handle_PacketIn)

                //تجزیه بسته
def _handle_PacketIn (event):
    packet = event.parsed
    dst_port = table.get(packet.dst)

                //اجازه عبور چند بسته از جریان
    msg = of.ofp_get_n_packet_flow_mod()
    msg.match.dl_src = packet.src
    msg.match.dl_dst = packet.dst
    msg.actions.append(of.ofp_action_output(port = dst_port))
    event.connection.send(msg)
```

شکل (۷). بررسی ابتدایی بسته و اجازه عبور چند بسته از جریان

```
from pox.core import core

def block_handler (event):

        packet = event.parsed                    // تشخیص قانون ارتباطی
        tcpp = event.parsed.find('tcp')
        udpp = event.parsed.find('udp')

        if tcpp:
        if tcpp.srcport in block_ports or tcpp.dstport in block_ports:
            core.getLogger ("SYN Receive")
            core.getLogger("DDos Blocker Start...").info("Blocked TCP-Attack Sourece
            event.halt = True                    //توقف رخداد مربوط به قرارداد TCP
        elif udpp:
                if udpp.srcport in block_ports or udpp.dstport in block_ports:
            core.getLogger("DDos Blocker Start...").info("Blocked UDP-Attack Sourece
            event.halt = True                    //توقف رخداد مربوط به قرارداد UDP
```

شکل (۸). متوقف‌کردن جریان حمله

۴-۴-شبه کد روش دفاع ناهمگن و خلاصه فازهای طی شده

در شکل (۹) شبه کد مربوط به روش دفاع ناهمگن آمده است، همان‌گونه که در این شبه کد مشخص است ابتدا بسته به سوئیچ وارد می‌شود، سپس چک می‌شود که ورودی مربوط به جریان داده جدیدی است یا جدول جریان مربوط به آن از قبل وجود داشته است.

```
// Input Packets in Switchs
For all Packets p input OF-Switchs do
    if p in FlowTable then
        forward p to Destination
    else
        forward p to Controller
        //in controller
            if p According to network rules then
                install Temporary flow table in Src OF-Switchs with action:
                    Flowtable timeout= t
                    foward few packet to Service provider
                    //in Service provider
                        if attack Detction
                            Send Event to controller
                            //in controller
                                Update Log Flie
                                install flowtables in OF-Switchs with action:
                                Dorp or Limit bw flow
                        else
                            Send msg no attack Detction
                            // in controller
                                install Permanent in OF-Switchs with action:
                                forward all packet in flow to Destination
            else
                install flowtables in OF-Switchs with action:
                Dorp or Limit bw flow
end for
```

شکل (۹). شبه کد روش دفاع ناهمگن

اگر بسته مربوط به جریان داده جدیدی باشد، سوئیچ بسته را جهت کسب تکلیف به کنترل‌کننده ارسال می‌کند. بسته به کنترل‌کننده وارد می‌شود. درصورتی‌که در چارچوب قوانین جاری شبکه باشد، اجازه عبور چند بسته از جریان مربوطه را به سمت سرویس‌دهنده صادر می‌شود. پس از ورود بسته‌ها به سرویس‌دهنده، ماژول تشخیص حملات، بسته‌ها را شنود[1] و تحلیل می‌کند. درصورتی‌که حمله‌ای تشخیص داده نشود گزارش وضعیت عادی را به کنترل‌کننده اطلاع می‌دهد. سپس کنترل‌کننده با ارسال دستور به سوئیچ اجازه عبور[2] برای آن جریان خاص را به سوئیچ واگذار می‌کند؛ اما درصورتی‌که ماژول تشخیص، انحرافی را دربسته‌ها کشف کند. گزارش رخداد به کنترل‌کننده ارسال می‌کند. کنترل‌کننده پس از دریافت گزارش رخداد، لاگ مربوط به حملات را بروز می‌کند، با تحلیل بسته مشکوک مبدأ و مقصد جریان مربوط به آن بسته را پیدا کرده و سپس دستوراتی را برای مقابله با حمله و انسداد آن جریان خاص به سوئیچ مربوط صادر می‌کند. در شکل شماره (۱۰) روند اتصال کاربر معمولی و مهاجم در سیستم دفاع ناهمگن ترسیم شده است.

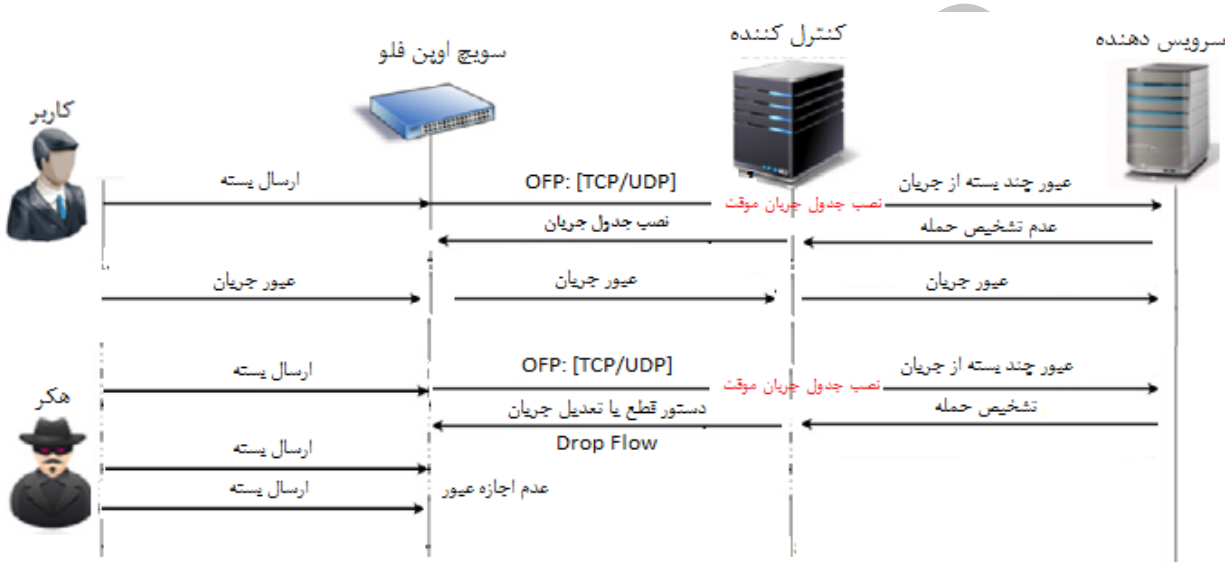
1- Sniff
2- Forward

۵- ارزیابی کارایی روش دفاع ناهمگن

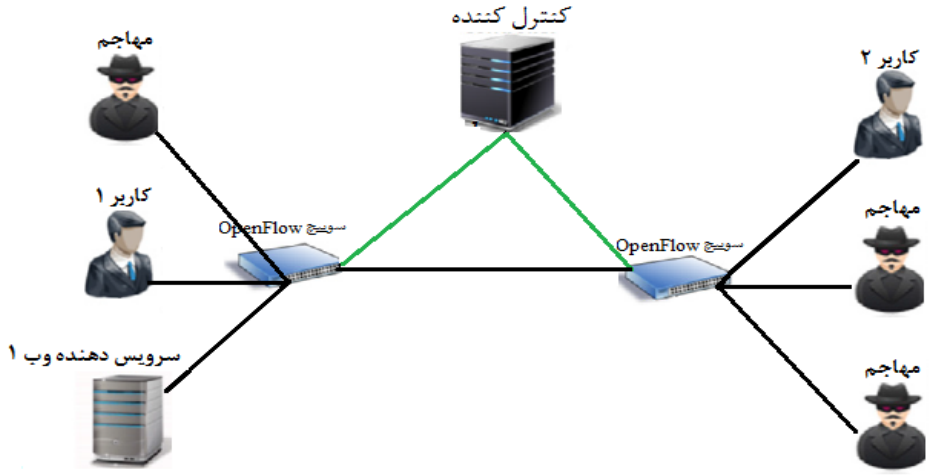
ما طرح پیشنهادی دفاع ناهمگن را در شبیه‌ساز Mininet که از معروف‌ترین شبیه‌سازهای مبتنی بر نرم‌افزار است [۲۰]، با استفاده از کنترل‌کننده POX [۱۰, ۱۸, ۳۶] در بستر سیستم‌عامل لینوکس مورد تحلیل و ارزیابی قرار می‌دهیم.

برای راه‌اندازی شبیه‌ساز Mininet از سیستم‌عامل لینوکس Ubuntu 12.10 32bit با پردازنده Quad Processor Q6600 به همراه ۴ گیگابایت حافظه و برای اجرای کنترل‌کننده Pox از یک لپ‌تاپ با پردازنده Intel 2core i686-1.4 Ghz استفاده می‌کنیم.

جهت انجام حملات سیل‌آسا از برنامه سوکت به زبان پایتون استفاده می‌کنیم. این برنامه با تعیین پورت مبدأ و مقصد سیل بسته‌های حمله را به سمت سرویس‌دهنده ارسال می‌کند. برای تحلیل و اندازه‌گیری نرخ بسته‌های عبوری از ابزار WireShark، برای اندازه‌گیری منابع مصرفی پروسه‌ها[1] از ابزار HTop [۳۷] و برای ایجاد درخواست‌های قانونی دریافت اطلاعات از iPerf [۳۸] استفاده می‌کنیم. معماری شبکه برای آزمون و ارزیابی از ۶ سیستم لینوکس تشکیل‌شده که به دو سوئیچ OpenFlow متصل است. در این معماری سرویس‌دهنده شماره ۱ مشغول خدمات‌رسانی سرویس وب[2] در شبکه است و ۲ سرویس‌گیرنده در حال دریافت خدمات از سرور شماره ۱ هستند. در همین شبکه ۳ سیستم آلوده وجود دارند که درصدد ایجاد حملات سیل‌آسا و ایجاد اختلال در روند عادی خدمات‌رسانی در شبکه هستند. شکل (۱۱)



شکل (۱۰). روند اتصال کاربر معمولی و مهاجم در سیستم دفاع ناهمگن



شکل (۱۱). معماری شبکه و کنترل‌کننده جهت ارزیابی روش دفاع ناهمگن

1- Proccess

2- Web Service

روش دفاع ناهمگن روشی بر اساس همکاری سرویس‌دهنده و شبکه‌های نوین مبتنی بر نرم‌افزار است. این سیستم باید قادر به دفع حملات سیل‌آسا در شبکه‌های سنتی باشد. همچنین مزیت‌هایی را نسبت به روش دفاع متمرکز مبتنی بر نرم‌افزار ارائه دهد. بنابراین برای ارزیابی کارآمدی روش دفاع ناهمگن دو مقایسه را انجام می‌دهیم.

در مقایسه اول که در بخش ۵-۱ ارائه می‌شود، مزیت روش دفاع ناهمگن را نسبت به روش‌های سنتی مورد ارزیابی قرار می‌دهیم. سپس در بخش ۵-۲ به مقایسه سیستم دفاع ناهمگن با روش‌های مقابله متمرکز در کنترل‌کننده می‌پردازیم.

۵-۱- مقایسه روش پیشنهادی دفاع ناهمگن با روش‌های سنتی در دفع حملات سیل‌آسا

در این بخش به مقایسه روش دفاع ناهمگن با روش‌های سنتی مقابله با حملات مانند SynCookies و دیواره آتش موجود در سرویس‌دهنده می‌پردازیم و برای ارزیابی از معیارهایی نظیر زمان پاسخ‌گویی به کاربران مجاز و میزان بار پردازشی تحمیل‌شده به سرویس‌دهنده استفاده می‌کنیم.

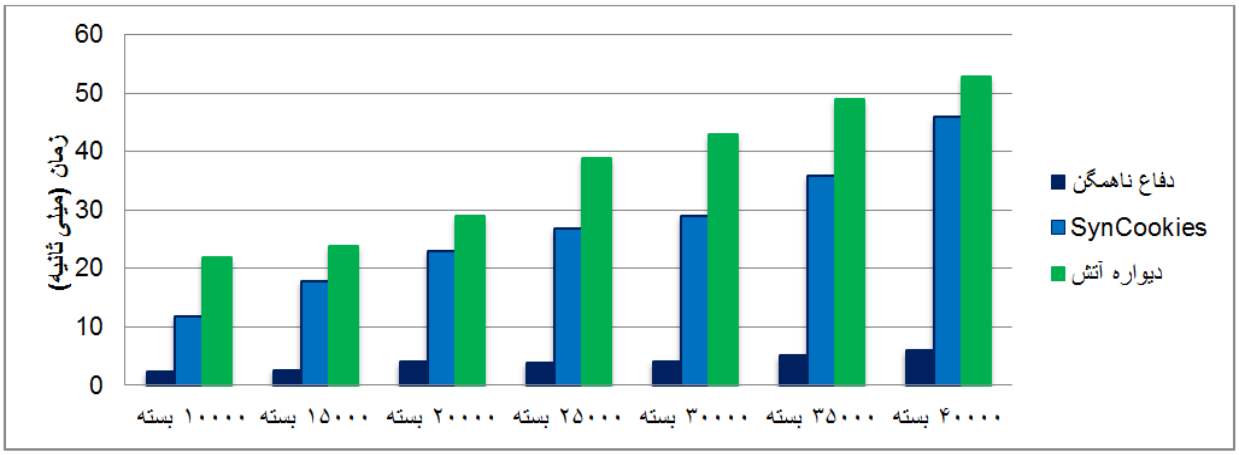
۵-۱-۱- زمان پاسخگویی سرویس‌دهنده

برای اندازه‌گیری زمان پاسخ‌گویی سرویس‌دهنده به کاربران مجاز در شیوه مقابله سنتی و ناهمگن، آزمایشی با چند نرخ تقریبی متفاوت متشکل از بسته‌های Syn را بر روی سرویس‌دهنده وب انجام دادیم. در گام نخست پس از حمله با استفاده از روش‌های سنتی مانند SynCookies و دیواره آتش موجود در سرویس‌دهنده، به مقابله با حملات اقدام و نتایج را ثبت کردیم. سپس با فعال‌سازی سیستم دفاع ناهمگن به مقابله با حملات اقدام نموده و نتایج حاصل از این روش را نیز اندازه‌گیری کردیم. همان‌گونه که نتایج آزمایش در شکل (۱۲) نشان می‌دهد. در روش مقابله سنتی با افزایش نرخ حملات، مقدار قابل‌توجهی از زمان صرف پاسخ‌گویی به درخواست‌های غیرمجاز می‌شود و زمان پاسخ‌گویی به درخواست کاربران مجاز افزایش می‌یابد. اما درروش دفاع ناهمگن سرویس‌دهنده پس از کشف حمله، کنترل‌کننده را از وضعیت مطلع می‌کند؛ سپس کنترل‌کننده با تشخیص مبدأ و مقصد حملات، تصمیماتی را در جهت قطع جریان به سوئیچ متصل به مهاجم ارسال کرده، حملات متوقف شده و زمان پاسخ‌گویی احیا می‌شود. در این روش کنترل‌کننده، اجازه ورود جریان حمله به سمت لینک‌های ارتباطی و منابع سرویس‌دهنده را از سمت سوئیچ مهاجم نمی‌دهد.

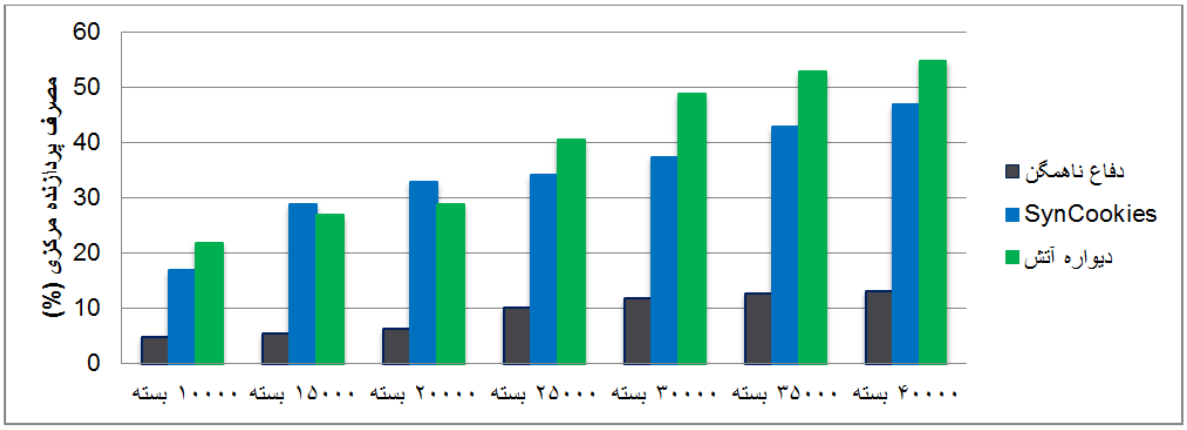
۵-۱-۲- میزان مصرف پردازنده مرکزی

در شکل (۱۳) میزان مصرف پردازنده مرکزی در زمان مقابله با حملات به روش سنتی و ناهمگن بررسی شده است. همان‌گونه که مشاهده می‌شود، روش‌های سنتی میزان بار پردازشی زیادی را در زمان مقابله با حملات، به پردازنده مرکزی سرویس‌دهنده تحمیل می‌کنند. با افزایش نرخ حملات میزان مصرف پردازنده نیز افزایش می‌یابد. درصورتی‌که در روش ناهمگن به دلیل جدابودن سیستم مقابله با حملات، تنها بار تحمیلی بر روی سرویس‌دهنده، ناشی از محاسبات ماژول تشخیص حملات است که در شکل (۱۳) قابل‌مشاهده است.

همان‌گونه که نتایج آزمایش‌های متعدد نشان می‌دهد، روش ناهمگن برخلاف روش سنتی، حفاظت از منابع و دسترسی را برای کاربران مشروع فراهم می‌آورد و مانع از اختلال در منابع شبکه می‌شود.



شکل (۱۲). متوسط زمان پاسخگویی سرویس‌دهنده به درخواست مجاز در روش دفاع سنتی و دفاع ناهمگن



شکل (۱۳). متوسط میزان بار پردازشی سرویس‌دهنده درروش دفاع سنتی و ناهمگن

۵-۲- مقایسه روش دفاع ناهمگن با روش‌های متمرکز در کنترل‌کننده در دفع حملات سیل‌آسا

در این بخش به مقایسه روش دفاع ناهمگن ، با سایر روش‌های دفاع متمرکزمبتنی بر نرم‌افزار می‌پردازیم و برای ارزیابی از معیارهایی نظیر میزان بار رایانشی کنترل‌کننده و تأخیر در پاسخ‌گویی به جریان‌های مجاز استفاده می‌کنیم.

همان‌گونه که در بخش ۳ مطرح شد مقالات [۱۳، ۳۰ و ۳۱] به ارائه روش‌هایی جهت تشخیص و مقابله با حملات تکذیب سرویس بر اساس شبکه‌های مبتنی بر نرم‌افزار پرداخته است.

بر طبق دانش ما و بررسی‌های صورت‌گرفته، طرح پیشنهادی دفاع ناهمگن با تمام طرح‌های فوق تفاوت دارد. در تمام طرح‌های ارائه‌شده، ماژول تشخیص و دفاع به‌صورت متمرکز در کنترل‌کننده قرار دارد. درصورتی‌که در طرح دفاع ناهمگن ماژول تشخیص در سرویس‌دهنده و ماژول مقابله در کنترل‌کننده واقع است. راهکار ارائه‌شده در مقاله [۳۱] مخصوص محیط‌های ابری است. در مقاله [۳۰] فقط روش‌های تشخیص حملات بررسی شده‌اند که در این میان فقط مقاله [۱۳] به ارائه روشی برای مقابله با حملات در شبکه‌های سنتی پرداخته است.

در این طرح کنترل‌کننده‌ای به نام OPERETTA پیشنهاد شده است. در حقیقت OPERETTA نسخه تکامل‌یافته کنترل‌کننده Pox جهت مقابله با حملات سیل‌آسای Syn در شبکه‌های سنتی است. در این طرح هر بسته TCP که قصد رسیدن به سرویس‌دهنده را داشته باشد توسط کنترل‌کننده اعتبار سنجی شده و در صورت تأیید اعتبار، اجازه عبور جریان به سمت سرویس‌دهنده صادر می‌شود. این روش تا حدودی شبیه به روش SynCookies در شبکه‌های سنتی است. از جمله مشکلات مهم در کنترل‌کننده OPERETTA تأخیر زیاد در رسیدن بسته‌های معتبر به سرویس‌دهنده است [۱۳].

در ادامه به ارزیابی طرح پیشنهادی دفاع ناهمگن را با روش متمرکز تشخیص و دفاع مبتنی بر Pox و کنترل‌کننده OPERETTA می‌پردازیم.

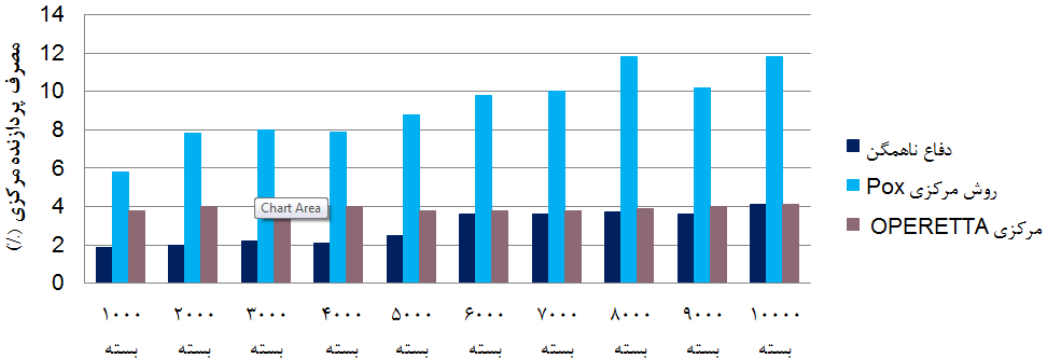
در جدول (۲) مشخصات محیط آزمایش هر کنترل‌کننده نشان داده شده است.

جدول(۲). مشخصات محیط آزمایش

روش تشخیص و مقابله	تشخیص و دفاع متمرکز در کنترل‌کننده Pox	تشخیص و دفاع متمرکز در کنترل‌کننده OPERETTA	روش پیشنهادی ناهمگن
سیستم تشخیص و دفاع	یکپارچه	یکپارچه	مبتنی بر همکاری
شبیه‌ساز	MiniNet	MiniNet	MiniNet
نوع هسته کنترل‌کننده	Pox	OPERETTA	Pox
سخت‌افزار کنترل‌کننده	Intel i3-2365m- 1.4Ghz- Ram 4GB	Intel i3-2365m- 1.4Ghz- Ram 4GB	Intel 2core i686 1.4 Ghz Ram 4GB
سیستم‌عامل آزمایش	K ubuntu 13.10 64bit	Kubuntu 13.10 64bit	Ubuntu 12.10 32bit
قابلیت دفع حملات	SynFlood	SynFlood	SynFlood-UdpFlood

۵-۲-۱- ارزیابی میزان بار پردازشی در کنترل‌کننده

برای ارزیابی میزان بار پردازشی کنترل‌کننده در روش دفاع ناهمگن، حمله‌ای سیل‌آسا متشکل از بسته‌های Syn با چند نرخ متفاوت بر روی سرویس‌دهنده وب (شکل (۱۱)) انجام دادیم. سپس با استفاده از ابزار Htop [۳۷] نتایج میزان مصرف پردازنده مرکزی را بر روی پروسه کنترل‌کننده اندازه‌گیری و نتایج به‌دست‌آمده را با کنترل‌کننده مرکزی OPERETTA و روش استاندارد متمرکز در Pox که از مقاله [۱۳] استخراج‌شده مقایسه کردیم؛ نتایج این مقایسه در شکل (۱۴) قابل مشاهده است.

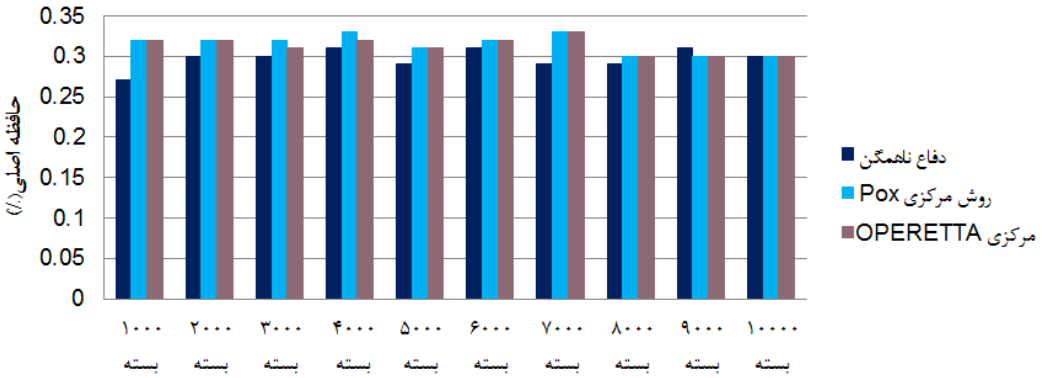


شکل (۱۴). مقایسه میانگین مصرف پردازنده مرکزی درروش ناهمگن با روش متمرکز Pox و کنترل‌کننده OPERETTA

همان‌گونه که نتایج نشان می‌دهد، طرح دفاع ناهمگن در نرخ‌های کمتر از ۱۰۰۰۰ بسته، نرخ پردازش پایین‌تری را نسبت به روش‌های دیگر ارائه داده و در نرخ ۱۰۰۰۰ بسته تقریباً با کنترل‌کننده OPERETTA برابری می‌کند. البته همان‌گونه که در جدول (۲) آمده است، پردازشگر کنترل‌کننده ناهمگن مشخصات پردازشی ضعیف‌تری نسبت به پردازنده کنترل‌کننده PERETTA و Pox دارد.

۵-۲-۲- ارزیابی میزان مصرف حافظه در کنترل‌کننده

در شکل (۱۵) نتایج مربوط به مصرف حافظه در نرخ‌های متفاوت حملات نشان داده شده است. همان‌گونه که از نتایج آزمایش مشخص است، روش ناهمگن در حملات با نرخ پایین مقادیر مصرف حافظه کمتری را نمایش می‌دهد. اما به‌صورت کلی با افزایش نرخ حملات میزان مصرف حافظه به کنترل‌کننده OPERETTA نزدیک می‌شود.



شکل (۱۵). مقایسه میانگین مصرف حافظه اصلی درروش ناهمگن با روش متمرکز Pox و کنترل‌کننده OPERETTA

۵-۲-۳- زمان پاسخگویی به درخواست‌های مجاز در کنترل‌کننده

یکی از چالش‌برانگیزترین مشکلات درروش دفاع متمرکز، تأخیر اضافی به‌وجودآمده ناشی از تائید صحت بسته‌ها است. چون اکثر درخواست‌ها در شبکه‌های مبتنی بر نرم‌افزار، جریان‌محور هستند، تأخیر در پاسخگویی کنترل‌کننده به جریان‌های مجاز از نگرانی‌ها بسیار مهم به شمار می‌رود [۲۱ و ۳۲].

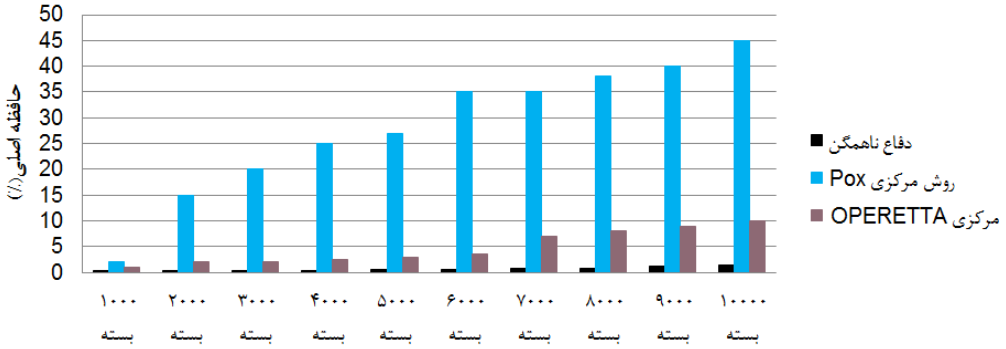
همان‌گونه که در بخش ۴ مطرح شد یکی از مهم‌ترین دلایل برتری روش دفاع ناهمگن، معطل نکردن بسته‌ها در کنترل‌کننده برای تشخیص حمله است. در این روش، تمامی بسته‌ها به سمت سرویس‌دهنده هدایت شده و عملیات احراز صحت در آنجا انجام می‌شود. این امر باعث می‌شود که کنترل‌کننده مشغول بررسی یا پاسخ به این درخواست‌ها نشده و زمان پاسخگویی به

درخواست‌های مجاز کاهش یابد. برای اثبات این ادعا آزمایشی را برای ارزیابی زمان پاسخگویی روش ناهمگن ارائه می‌دهیم.در این آزمایش، حملات سیل‌آسا را با چندین نرخ مختلف در شبکه (شکل (۱۱)) انجام داده، سپس درخواست‌هایی را برای دریافت خدمات وب به سمت کنترل‌کننده ارسال کرده و مدت‌زمان طی شده برای نصب جریان یک جلسه کامل HTTP را اندازه‌گیری می‌کنیم. سپس این نتایج با خروجی کنترل‌کننده OPERETTA و روش متمرکز مبتنی بر Pox را در شکل (۱۶) قیاس می‌کنیم.

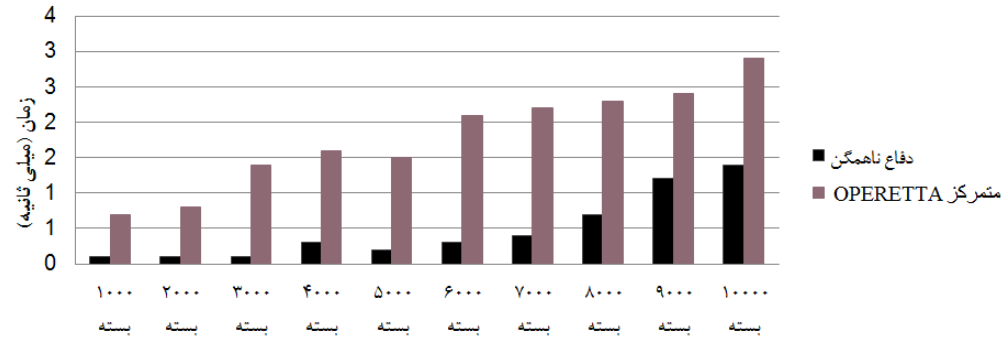
همان‌گونه که نتایج نشان می‌دهد در روش دفاع ناهمگن، میزان تأخیر پاسخ‌دهی بسیار پایین‌تری نسبت به کنترل‌کننده مرکزی OPERETTA و روش متمرکز مبتنی بر Pox ارائه می‌دهد. در حملات با نرخ پایین کنترل‌کننده OPERETTA میزان تأخیر ۲ یا ۳ ثانیه‌ای را ارائه داده، اما روش ناهمگن زمان‌های پاسخگویی زیر ۱ ثانیه را ارائه داده است. با افزایش نرخ حملات، روش ناهمگن مزیت‌های بیشتری را از نظر زمان پاسخ‌گویی نسبت به سایر روش‌ها ارائه می‌دهد.

۵-۲-۴- زمان موردنیاز برای دفع حملات توسط کنترل کننده

در این بخش به ارزیابی زمان مورد نیاز برای مقابله در حالات مختلف می‌پردازیم. برای انجام این مقایسه سیل بسته‌های syn را به سمت سرویس‌دهنده جاری می‌کنیم . و زمان مورد نیاز برای دفع حملات توسط کنترل‌کننده را مورد بررسی قرار می‌دهیم. نتایج حاصل از ارزیابی نشان می‌دهد که روش ناهمگن مدت زمان کمتری را برای دفع حملات نسبت به روش‌های متمرکز ارائه می‌دهد. چرا که در رویکرد متمرکز کنترل‌کننده مجبور است که بار ناشی از تشخیص و پاسخ به حملات را علاوه بر کنترل شبکه به تنهایی انجام دهد. نتایج حاصل از ارزیابی با نرخ‌های مختلف در شکل (۱۷) آورده شده است.



شکل (۱۶). مقایسه میانگین زمان پاسخگویی به درخواست‌های مجاز توسط روش ناهمگن با روش متمرکز Pox و کنترل‌کننده OPERETTA



شکل (۱۷). متوسط زمان دفع حملات درروش دفاع متمرکز و دفاع ناهمگن

۶- نتیجه‌گیری

در این مقاله به بررسی شبکه‌های سنتی و شبکه‌های مبتنی بر نرم‌افزار و راه‌حل‌های ارائه شده برای مقابله با حملات سیل‌آسا پرداختیم. سپس روشی ناهمگن را بر اساس همکاری سرویس‌دهنده و کنترل‌کننده مبتنی بر نرم‌افزار (SDN) پیشنهاد کردیم. همچنین نشان دادیم که روش پیشنهادی برتری‌هایی از نظر مصرف منابع رایانشی و مدت زمان پاسخگویی نسبت به سایر روش‌ها (سنتی و روش‌های تشخیص و دفاع متمرکز در کنترل‌کننده) ارائه داده است. در حالت طبیعی و بر طبق معیارهای NOX (اولین کنترل‌کننده شبکه‌های مبتنی بر نرم‌افزار و پدربزرگ کنترل‌کننده Pox)، این کنترل‌کننده می‌تواند در بهترین حالت کارایی در هر واحد زمانی ۳۰۰۰۰ جریان را اداره کند [۳۳]؛ چون کنترل‌کننده روش دفاع ناهمگن، بر اساس هسته کنترل کننده Pox طراحی شده در حملات با نرخ بالا کنترل‌کننده با مشکلات مقیاس‌پذیری روبه‌رو می‌شود. مقالات و راه‌کارهای متعددی برای حل مشکلات مقیاس‌پذیری شبکه‌های مبتنی بر نرم‌افزار منتشر شده است [۳۲] که نشان می‌دهد که

مشکلات مقیاس‌پذیری مربوط به ذات شبکه‌های مبتنی بر نرم‌افزار نیست [۳۲]. موازی‌سازی دستگاه‌های کنترل‌کننده و توزیع پردازش بر روی چند کنترل‌کننده، روشی مناسب برای حل این مشکلات است. مطالعه بعدی ما این است که یک مکانیسم مبتنی بر رأی‌گیری جهت تشخیص حملات به‌صورت ناهمگن ارائه کنیم. در این روش هر سرویس‌دهنده دارای یک رأی بوده و کنترل‌کننده با توجه به آرا، اقدامات حفاظتی را در شبکه پیاده‌سازی می‌کند.

۷- مراجع

[1] P. Goransson and C. Black, "Software Defined Networks: A Comprehensive Approach," Elsevier, 2014.

[2] M. McCauley, "About pox," URL: http://www.noxrepo.org/pox/about-pox/. Online, 2013.

[3] D. Moore, G. M. Voelker, and S. Savage, "Inferring internet denial-of-service attack," In Proceedings of USENIX Security Symposium, 2001.

[4] C. Meadows, "A formal framework and evaluation method for network denial of service," In Computer Security Foundations Workshop, 1999. Proceedings of the 12th IEEE, pp. 4-13, 1999.

[5] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP," In Security and Privacy, 1997 Proceedings, 1997 IEEE Symposium on, pp. 208-223, 1997.

[6] U. D. Protocol, "RFC 768 J. Postel ISI 28 August 1980," Isi, 1980.

[7] " UDP flood attack" https://en.wikipedia.org/wiki/UDP_flood_attack.

[8] "Alert (TA14-017A) UDP-based Amplification Attacks," US-CERT, 2014.

[9] P. Froutan, "How to defend against DDoS attacks," URL : http://www. computerworld. com/s/article/94014/How_to_defend_against_DDoS_attacks, 2004.

[10] N. McKeown et al., "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, pp. 69-74, 2008.

[11] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," ACM Computing Surveys (CSUR), vol. 39, p. 3, 2007.

[12] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," Communications Surveys & Tutorials, IEEE, vol. 15, pp. 2046-2069, 2013.

[13] S. Fichera, L. Galluccio, S. C. Grancagnolo, G. Morabito, and S. Palazzo, "OPERETTA: An OPEnflow-based REmedy to mitigate TCP SYNFLOOD Attacks against web servers," Computer Networks, vol. 92, pp. 89-100, 2015.

[14] B. Hang and R. Hu, "A novel SYN Cookie method for TCP layer DDoS attack," In BioMedical Information Engineering, 2009. FBIE 2009. International Conference on Future, pp. 445-448, 2009.

[15] J. Lemon, "Resisting SYN Flood DoS Attacks with a SYN Cache," In BSDCon, pp. 89-97, 2002.

[16] E. Borcoci, "Software Defined Networking and Architectures," In Fifth International Conference on Advances in Future Internet (AFIN 2013), 2013.

[17] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: challenges and research opportunities for future internet," Computer Networks, vol. 75, pp. 453-471, 2014.

[18] T. N. Subedi, K. K. Nguyen, and M. Cheriet, "OpenFlow-based in-network Layer-2 adaptive multipath aggregation in data centers," Computer Communications, vol. 61, pp. 58-69, 2015.

[19] M. Kobayashi et al., "Maturing of OpenFlow and Software-defined Networking through deployments," Computer Networks, vol. 61, pp. 151-175, 2014.

[20] "Improved hardware limitations," http://www.xinguard.com/en/content.aspx?id=70

[21] A. Tavakoli, "Exploring a centralized/distributed hybrid routing protocol for low power wireless networks and large-scale datacenters," University of California, Berkeley, 2009.

[22] "OpenFlow," http://wiki.mikrotik.com/wiki/Manual:OpenFlow.

[23] "Google shared details on its production use of OpenFlow in its SDN network at this spring's Open Networking Summit," http://www.networkcomputing.com/networking/inside-googles-software-defined-network/a/d-id/1234201.

[24] "Software Defined Networking at Scale," Research at Google. https://research.google.com/pubs/archive/42948.pdf.

[25] H. Solomon, "Alcatel Now Supports OpenFlow," OpenStack on Switches, 2013.

[26] S. Sondur, "|Software Defined Networking for Beginners,"

[27] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "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, pp. 127-132, 2012.

[28] 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, pp. 121-126, 2012.

[29] G. Yao, J. Bi, and P. Xiao, "Source address validation solution with OpenFlow/NOX architecture," In Network Protocols (ICNP), 2011 19th IEEE International Conference on, pp. 7-12, 2011.

[30] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in Local Computer Networks (LCN), 2010 IEEE 35th Conference on, pp. 408-415, 2010.

[31] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and Software-Defined Networking," Computer Networks, vol. 81, pp. 308-319, 2015.

[32] S. H .Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," Communications Magazine, IEEE, vol. 51, pp. 136-141, 2013.

[33] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "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.

[34] G. Bojadziev and M. Bojadziev, "Fuzzy logic for business," finance, and management: World Scientific Publishing Co., Inc., 2007.

[35] K. Lee, J. Kim, K. H. Kwon, Y. Han, and S. Kim, "DDoS attack detection method using cluster analysis," Expert Systems with Applications, vol. 34, pp. 1659-1665, 2008.

[36] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," In DARPA Information Survivability Conference and Exposition, 2003. Proceedings, pp. 303-314, 2003.

[37] "Htop Documentation," http://hisham.hm/htop.

[38] "Iperf Documentation," http//:iperf.fr.

[39] J. Li, Y. Liu, and L. Gu, "DDoS attack detection based on neural network," In Aware Computing (ISAC), 2010 2nd International Symposium on, pp. 196-199, 2010.

Protection Against Flooding Attacks In Traditional Networks in Heterogeneous Partnership With Service Provider And Software Define Network (SDN) Controller

R. Mohammadifar, A. A. Rezaei*

Payame Noor University

(Received: 18/12/2015, Accepted: 03/05/2016)

ABSTRACT

Flood attacks are among the most devastating distributed denial of service (DDoS) attacks. In these attacks, attackers try to use up resources by sending massive floods of packets in order to seriously challenge the provision of services. Hierarchical architecture and numerous weaknesses in the structure of communication protocols in conventional networks lead to the fact that firewalls cannot provide an integrated and effective mechanism against these attacks. With the emergence of Software Defined networking (SDN), there are new prospects for solving structural and security problems in Traditional networks. In this paper, a heterogeneous method proposed based on cooperation between traditional service provider and software-based controller (SDN) to deal with flooding attacks. In this method, the attack detection module is located in servers and the call module is located in controller(SDN). In order to simulate the heterogeneous method the MiniNet emulator is used in combination with the Pox controller. Next, the simulated model is evaluated and it is finally concluded that besides protecting against attacks in conventional networks, the proposed method provides other benefits including the extent of computational load and the response time compared to other Software Defined methods.

Keywords: Flood Attack, Software Defined Network, OpenFlow Protocol

* Corresponding Author Email: A_rezaee@pnu.ac.ir