الگوریتم‌هایی برای تشخیص و رفع سریع تضاد در سیاست‌های امنیتی مبتنی بر IPSec در شبکه‌های خصوصی مجازی

سلمان نیک‌صفت
دانشجوی دکتری، گرایش امنیت اطلاعات
دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر
niksefat@aut.ac.ir

مسعود صبایی
استادیار
دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر
sabaei@aut.ac.ir

چکیده: پیچیدگی و تنوع سیاست‌های امنیتی مبتنی بر IPSec در شبکه‌های خصوصی مجازی، می‌تواند منجر به ترکیبی از سیاست‌ها شود که نه تنها سرویس‌های امنیتی مورد نیاز را فراهم نمی‌کند، بلکه امنیت ارتباط را نیز به مخاطره می‌اندازند. در این پژوهش یک روش فورمال با استفاده از عبارات بولی جهت تشخیص و رفع تضادها در سیاست‌های IPSec ارائه شده است. وقوع هر تضاد در این روش با یک عبارت بولی بیان می‌شود. در این روش سیاست‌های ورودی پردازش شده و بر اساس نیاز کاربر مجموعه‌ای از سیاست‌های جدید که بدون تضاد و مشکلات امنیتی است، ایجاد می‌شود. پیاده‌سازی و ارزیابی این روش با روش‌های قبلی نشان داد که این روش علاوه بر این‌که تضاد را با کارایی بهتری نسبت به روش‌های موجود تشخیص می‌دهد، قادر به ارائه مجموعه‌ای از سیاست‌های جدید و ایمن نیز هست.

واژه های کلیدی: شبکه خصوصی مجازی، IPSec، سیاست‌ها در IPSec، تضاد در سیاست‌ها، تشخیص و رفع تضاد.

۱- مقدمه

شبکه‌های خصوصی مجازی مبتنی بر IPSec امروزه بطور گسترده‌ای برای ایجاد ارتباط فیزیکی مجازی و ایمن بین بخش‌های مختلف سازمان‌ها و یا بین کاربران راه دور با شبکه آن‌ها مورد استفاده قرار می‌گیرد.

نحوه عملکرد دروازه‌ها و گره‌های شبکه برای استفاده از IPSec با استفاده از سیاست‌های IPSec مشخص می‌شود. در حال حاضر سیاست‌ها در IPSec به‌صورت دستی اعمال می‌شوند. این کار برای شبکه‌های بزرگ توزیعی ناکارآمد بوده و احتمال بروز خطا در آن وجود دارد. لذا استفاده از یک سیستم سیاستگذار که بصورت سیستماتیک پیکربندی و مدیریت سیاست‌ها در IPSec را کنترل می‌کند ضروری به‌نظر می‌رسد.

با توجه به مشکلات و کاستی‌های رویکردهای فعلی، روشی که برای تحلیل و تشخیص تضادها در سیاست‌های IPSec، علاوه بر تشخیص به رفع تضادها نیز بپردازد و در شرایط پویا، یعنی زمانی که سیاست‌ها مدام در حال اضافه یا کم شدن هستند، کارایی مناسبی داشته باشد، یکی از زمینه‌های تحقیقاتی مناسب است که در این مقاله به آن پرداخته شده است.

در این مقاله، در بخش دوم پس از معرفی اجمالی سیاست‌ها در IPSec، به بررسی مساله تضاد در فیلترهای بسته و سیاست‌های IPSec می‌پردازیم. در بخش سوم کارهای مشابه و مزایا و معایب هریک از روش‌های ارائه شده بررسی می‌شود. در بخش چهارم روش پیشنهادی و الگوریتم‌های مربوط به آن ارائه می‌گردد. در بخش پنجم الگوریتم‌های ارائه شده مورد تحلیل و ارزیابی قرار گرفته‌اند.

۲- انواع تضاد در سیاست‌های IPSec

از آنجایی‌که در IPSec سیاست‌ها غالباً به‌صورت دستی توسط مدیر سیستم مشخص می‌شود، امکان بروز خطا و اعمال سیاست‌های غلط در سیستم وجود دارد.

۲-۱ انواع تضاد در سیاست‌های IPSec

تضاد در سیاست‌ها به تضاد در شرایط لیست قواعد و تضاد در پارامترهای رمزنگاری و تونل، تقسیم می‌شود. در این بخش یک دسته‌بندی از انواع تضادها مطرح شده است که نسبت به دسته‌بندی‌های پیشین([1]، [2] و [3]) جامع‌تر بوده و الگوریتم‌های ارائه شده در روش پیشنهادی بر اساس آن عمل می‌کنند.

۲-۱-۱- تضاد در شرایط لیست قواعد

تضادهای زیر در شرایط لیست قواعد ممکن است رخ دهد:

- افزونگی : این حالت وقتی رخ می‌دهد که شرط یک قاعده زیر مجموعه یک یا چند قاعده پیش از آن است و لذا سبب می‌شود این قاعده هیچ‌گاه مورد رجوع قرار نگیرد. در صورتی‌که عمل این قاعده با عمل قواعد پیشین(که زیر مجموعه آنهاست)، یکی باشد این افزونگی را افزونگی یکسان و در غیر این صورت افزونگی متضاد گوییم.
- هم‌پوشانی جزیی: این حالت وقتی رخ می‌دهد که شرط یک قاعده با بخشی از شرایط قواعد پیش از آن همپوشانی(اشتراک) دارد. در این حالت، عملی که روی بخش‌های دارای همپوشانی انجام می‌شود، عمل قاعده‌ای است که در مکان بالاتری از لیست قرار

دارد. اما مدیر سیستم ممکن است بخواهد به عمل قاعده پایین‌تر اولویت بدهد.

۲-۱-۲- تضاد در پارامترهای رمزنگاری و تونل‌ها

در صورتی‌که عمل مورد نظر در یک قاعده حفاظت باشد، پارامترهای حفاظتی می‌بایست مشخص شود. در صورتی که از حالت کاری تونل استفاده شود، مقصد تونل جزو این پارامترهاست. در IPSec اعمال چندین تابع امنیتی و همچنین چندین تونل تودرتو مجاز است. برخی از این حالات ممکن است ترکیب‌های ناامن و نادرستی ایجاد نمایند:

- هم‌پوشانی تونل‌ها : ترافیک خروجی از دروازه IPSec می‌تواند بر روی چندین تونل به‌طور همزمان قرار گیرد. به این مورد تونل‌های تودرتو گفته می‌شود. در این حالت اگر تونل بیرونی به مقصدی دورتر از تونل درونی برود با مشکل امنیتی مواجه خواهیم شد.
- ترکیب نادرست توابع رمزنگاری: در RFC2401، به برخی از ترکیبات توابع امنیتی اشاره شده که یا حالات افزونه ایجاد می‌کند و یا از نظر امنیتی ضعیف است. همسایگی انتقال به اعمال بیش از یک پروتکل امنیتی بر روی یک بسته IP، بدون استفاده از تونل اطلاق می‌شود. این رویکرد برای ترکیب AH و ESP تنها یک لایه از ترکیب را اجازه می‌دهد، و لایه‌های بیشتر باعث امنیت بیشتر نشده و تنها کارایی را کاهش می‌دهد(RFC2401 [4]، صفحه ۱۱). همچنین اگر AH بخواهد در حالت انتقال به‌صورت ترکیبی با ESP مورد استفاده قرار بگیرد، سرآیند AH می‌بایست درست بعد از سرآیند IP و پیش از سرآیند ESP قرار بگیرد. در این حالت AH بر روی خروجی رمزنگاری شده ESP اعمال می‌شود. در مقابل برای SA حالت Tunnel، ترتیب های مختلفی از AH و ESP می‌تواند مورد استفاده قرار بگیرد (RFC 2401 [4]، صفحه ۱۲)

۳- تحقیقات و کارهای مرتبط

در زمینه تشخیص و رفع تضادها در سیاست‌های IPSec دو کار عمده تحقیقی صورت گرفته که هر دو دارای کاستی‌هایی هستند. روش Wu [5] نیازمندی‌های امنیتی را مطرح کرده که استفاده از آن در بسیاری از حالات عملی بنظر نمی‌رسد و همچنین پیچیدگی بالایی نیز دارد. روش Shaer [1] نیز هرچند چارچوب مناسبی برای تحقیقات جدید است اما محدود به تشخیص تضادها است، نه رفع آنها. همچنین الگوریتم مورد استفاده در این روش برای تشخیص تضادها عملکرد بهینه‌ای ندارد. لذا با توجه به تحقیقات موجود ارائه روشی برای تحلیل سیاست‌های IPSec که علاوه بر تشخیص به رفع تضادها نیز بپردازد و در شرایط مختلف کارایی مناسبی داشته باشد، ضروری به‌نظر می‌رسد.

۴- روش پیشنهادی

در این بخش به بررسی روش پیشنهادی می‌پردازیم. در روش پیشنهادی با افزودن توابع و پارامترهایی به کل لیست قواعد و همچنین نمایش بخش شرط هر قاعده توسط یک تابع بولی، تضادها را با یک الگوریتم کارا تشخیص داده و سپس رفع می‌کنیم. الگوریتم‌های ارائه شده ابتدا تضادها را تشخیص می‌دهند و سپس با توجه به نیاز کاربر لیستی از سیاست‌های بدون تضاد ایجاد می‌کنند. الگوریتم‌های ارائه شده در هنگام حذف و یا اضافه کردن یک قاعده جدید نیز کارایی مناسبی دارند.

۱-۴ نحوه نمایش سیاست‌ها به‌صورت فورمال

نحوه نمایش سیاست‌ها در استاندارد IPSec بسیار انعطاف‌پذیر در نظر گرفته شده است و لذا برای حل مساله نیاز است تا نحوه نمایش سیاست‌ها بصورت دقیق بیان شود. مدل ارائه شده در این‌جا با نیازمندی‌هایی که در استاندارد تدوین شده هماهنگی دارد و لذا قابل استفاده در پیاده‌سازی‌های مختلف است.

تعریف اول:

سیاست در IPSec بصورت لیستی از قواعد(R_i) نمایش داده می‌شود. این لیست را P می‌نامیم(فرمول ۱).

$$P = R_1, R_2, \ldots, R_n \quad (۱)$$

تعریف دوم:

یک قاعده R_i شامل مجموعه‌ای از قیود بر روی فیلدهای فیلتر F، به‌همراه یک عمل از مجموعه عمل‌ها Act_i ، و پارامترهای مورد نیاز حفاظتی است(فرمول ۲).

$$R_i := C_i \rightarrow Act_i \quad (۲)$$
$$Act_i = \{bypass, deny, protect/parameters\}$$

که در رابطه بالا C_i شرطی است که می‌بایست برقرار باشد تا Act_i رخ دهد. C_i می‌تواند به‌صورت یک عبارت بولی بر اساس مقادیر فیلتر نمایش داده شود(فرمول ۳).

$$C_i = fv_i^1 \wedge fv_i^2 \wedge \ldots \wedge fv_i^k \quad (۳)$$

fv مشخصه‌ای از بسته IP است که فیلترینگ می‌تواند بر اساس آن صورت گیرد. به‌عنوان مثال fv می‌تواندیک IP خاص، یک مجموعه از آدرس‌های IP، یک پورت خاص و غیره باشد. هر fv در عبارت بالا می‌تواند به‌راحتی به‌صورت یک عبارت بولی نمایش داده شود[6]. Act_i نیز می‌تواند *حفاظت*، *عبور* و یا *ممانعت* باشد. در حالتی که عمل حفاظت است، پارامترهای اعمال حفاظت مانند نوع الگوریتم‌های رمزنگاری، آدرس انتهایی تونل و مود کاری IPSec (Tunnel یا Transport) می‌بایست مشخص شود.

تعریف سوم:

برای هر لیست قواعد سیاست‌های IPSec، سه تابع بولی برای هریک از عمل‌های موجود(حفاظت، ممانعت و عبور) در نظر می‌گیریم

که بیانگر برآیند شرایطی هستند که منجر به رخ دادن آن عمل خاص می‌شود(فرمول ۴، ۵ و۶).

$$S_{protect} = f_1(fv_1, fv_2, fv_3, ..., fv_n) \quad (۴)$$

$$S_{bypass} = f_2(fv_1, fv_2, fv_3, ..., fv_n) \quad (۵)$$

$$S_{deny} = f_3(fv_1, fv_2, fv_3, ..., fv_n) \quad (۶)$$

این توابع نمی‌بایست به‌طور همزمان true شوند، چون در غیر این صورت عملی که روی یک بسته ورودی می‌بایست انجام شود با ابهام مواجه می‌شود(فرمول ۷).

$$(S_{protect} \wedge S_{deny}) \vee (S_{protect} \wedge S_{bypass}) \vee (S_{bypass} \wedge S_{deny}) == false \quad (۷)$$

در (۷) و سایر فرمول‌ها همانند زبان C، عملگر == برای برابری و عملگر = برای ازاگذاری استفاده شده است. سایر عملگرها، عملگرهای منطقی هستند ($\wedge$ برای عمل AND و $\vee$ برای عمل OR).

تابع بولی S_{all} بصورت برآیند تمامی توابع فوق یا برآیند تمامی شرایط موجود تعریف می‌شود(فرمول ۸).

$$S_{all} = S_{protect} \vee S_{deny} \vee S_{bypass} \quad (۸)$$

۲-۴ الگوریتم پیشنهادی برای تشخیص و رفع تضادها در سیاست‌های IPSec

تشخیص تضاد در سیاست‌ها بر اساس توابع برآیند S صورت می‌گیرد. در هنگام افزودن یک قاعده جدید با کنترل یک یا چند رابطه بولی به رابطه بین شرط قاعده جدید با قواعد موجود پی می‌بریم. به‌عنوان مثال اگر شرط جدید زیر مجموعه شرایط فعلی است، پس قاعده جدید یک قاعده افزونه است. ایده اصلی که برای تشخیص تضاد استفاده شده، محاسبه توابع برآیند جدید از روی شرط قاعده جدید و توابع برآیند موجود است. در روش [1]، محاسبه تابع S پس از هر بار بروزرسانی سیاست، از اول و بر اساس تمامی شروط موجود صورت می‌گیرد، لذا کارایی خوبی ندارد. برای حل این مشکل بدین ترتیب عمل می‌کنیم: برای این‌که لیستی از سیاست‌های بدون تضاد داشته باشیم، سیاست‌ها را از لیست سیاست‌های ورودی برداشته و در لیست جدید($P_{no\text{-}conflict}$) قرار می‌دهیم. پیش از افزودن یک سیاست رابطه آن را با سیاست‌های جدید محاسبه می‌کنیم. در صورتی که افزودن این رابطه باعث ایجاد تضاد در لیست جدید شود، آن را به کاربر گزارش داده و تضاد را رفع می‌کنیم. در غیر این صورت این سیاست را به لیست سیاست جدید اضافه می‌کنیم. پس از افزودن هر سیاست به لیست سیاست‌های جدید، سه تابع S_{bypass}، S_{deny} و $S_{protect}$ را بروزرسانی می‌کنیم، تا وضعیت فعلی سیاست‌ها را نمایش دهند. قاعده جدید را به شکل زیر نمایش می‌دهیم(فرمول ۹).

$$R_{new}: C_{new} \rightarrow Act_{new}$$
$$Act_{new} = \{bypass, deny, protect/parameters\} \quad (۹)$$

در هر لحظه برای این‌که مجموعه‌ای از شرایط را که در لیست سیاست جدید عمل آن‌ها مشخص شده، بدست آوریم، از Sall استفاده می‌کنیم که در (۸) نشان داده شد.

همچنین لیست سیاست‌های فعلی را با P_{old} و لیست سیاست‌های جدید را با $P_{no\text{-}conflict}$ نمایش می‌دهیم. رابطه یک قاعده جدید با قواعد سیاست جدید، می‌تواند شامل حالات زیر باشد.

۱-۲-۴ افزودن یک قاعده افزونه به سیاست‌ها

در این حالت شرط قاعده جدید زیرمجموعه قواعد قبلی است و این قاعده اصلا" اجرا نمی‌شود چون یک یا چند قاعده قبل از آن وجود دارد، که کل شرط آن را به‌صورت کامل پوشش می‌دهند و این قاعده هیچگاه اجرا نمی‌شود. در صورتی که یکی از شروط (۱۲) و یا (۱۳) برقرار باشد، قاعده جدید افزونه است.

$$(C_{new} \Rightarrow S_{All}) == true \quad (۱۰)$$

$$(C_{new} \wedge S_{All}) == C_{new} \quad (۱۱)$$

یعنی شرط جدید زیرمجموعه شرایط قبلی است.

- نحوه تشخیص نوع افزونگی

برای تشخیص نوع افزونگی بدین ترتیب عمل می‌شود. ابتدا شرط (۱۴) بررسی می‌شود.

$$((Act_{new} == deny) and ((C_{new} \Rightarrow S_{deny}) == true)) or$$
$$((Act_{new} == bypass) and ((C_{new} \Rightarrow S_{bypass}) == true)) \quad (۱۲)$$

در صورتی برقرار شرط فوق، قاعده جدید یک قاعده افزونه یکسان است. در غیر این صورت شرط (۱۵) بررسی می‌شود:

$$(Act_{new} == protect)$$
$$and ((C_{new} \Rightarrow S_{protect}) == true)$$
$$and (\forall R_i \in P \mid (Act_i == protect) \quad (۱۳)$$
$$and (C_i \wedge C_{new} \neq false)) \Rightarrow$$
$$(Parameters_i == Parameters_{new}))$$

یعنی پارامترهای رمزنگاری شروط قبلی که در برگیرنده شرط جدید هستند، با پارامترهای شرط جدید یکی باشند. در این صورت نیز قاعده جدید یک قاعده افزونه یکسان است. در سایر حالات قاعده جدید یک قاعده افزونه متضاد می‌باشد.

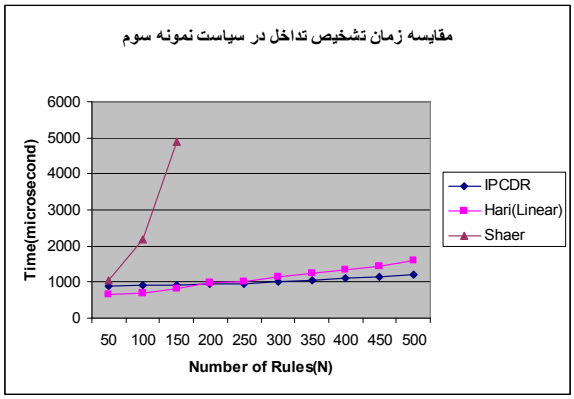
۵- تحلیل و ارزیابی

الگوریتم‌های پیشنهادی برای تشخیص و رفع تضاد در سیاست‌های IPSec در محیط لینوکس با استفاده از زبان C++ پیاده‌سازی شد. برنامه حاصل را "ابزار تشخیص و رفع تضاد در سیاست‌های IPSec" یا IPCDR نامگذاری کردیم. برای کار با توابع بولی از کتابخانه Buddy [7] برای C و C++ که ساختمان داده دیاگرام تصمیم دودویی را پیاده‌سازی کرده، استفاده گردید. ارزیابی کارایی بصورت عملی، در یک

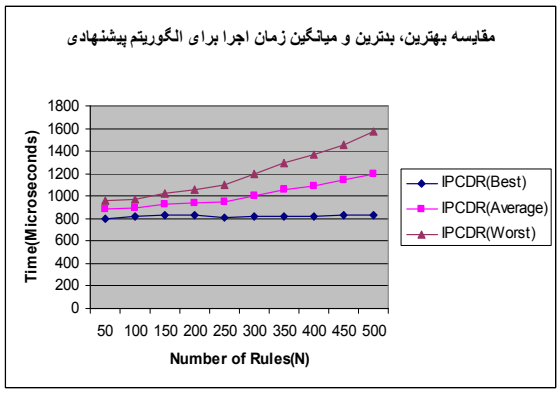
سیستم با پردازنده Intel Mobile 1.4Ghz با 512MB حافظه اصلی انجام شد.

شکل ۱ مقایسه زمان اجرا برای الگوریتم پیشنهادی(IPCDR) و در الگوریتم مشابه را نشان می‌دهد. همانطور که از این شکل پیداست با افزایش تعداد قواعد نمودار زمان مورد نیاز برای تشخیص تضاد در روش پیشنهادی با شیب بسیار کمتری نسبت به روش Hari[3] و Shaer[1] رشد می‌کند و کارایی بهتری دارد.

شکل ۲ زمان مورد نیاز برای تشخیص تضاد در هریک از سیاست‌های مورد نظر نشان می‌دهد. در این شکل بهترین حالت مربوط به سیاست-های کاملاً مستقل است، بدترین حالت مربوط به سیاست‌های افزونه با عمل حفاظت و حالت میانگین ترکیب دو سیاست مستقل و افزونه است.

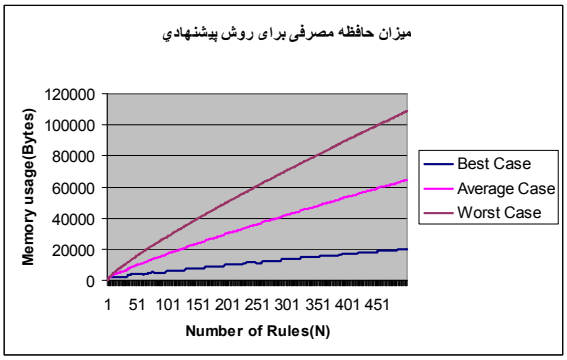


شکل ۱ – مقایسه زمان تشخیص در روش پیشنهادی(IPCDR) و ۲ الگوریتم دیگر برای سیاست نمونه سوم



شکل ۲ – مقایسه بهترین، بدترین و میانگین زمان اجرا برای الگوریتم پیشنهادی

شکل ۳ میزان حافظه مصرفی در الگوریتم پیشنهادی را برای سه سیاست نمونه نشان می‌دهد. با توجه به این شکل میزان حافظه مصرفی در هر سه حالت تقریباً بصورت خطی رشد می‌کند. لیست سیاست اول حافظه مصرفی بیشتری جهت نگه‌داری توابع برآیند می‌خواهد چرا که ساختار BDD هر شرط حجیم‌تر است.



شکل ۳ –میزان حافظه مصرفی در روش پیشنهادی برای تشخیص تضاد

در لیست سیاست دوم از آن‌جاکه بسیاری از متغیرها بصورت Don't Care هستند، حجم حافظه مصرفی برای نمایش توابع بولی مربوط به شرایط و توابع برآیند کمتر است. لیست سیاست سوم نیز ترکیبی از لیست‌های اول و دوم است.

۶- نتیجه‌گیری

در این مقاله روشی برای تشخیص و رفع تضاد در سیاست‌های امنیتی IPSec مطرح شد که نسبت به روش‌های موجود سریع‌تر عمل کرده و مجموعه‌ای از سیاست‌های بدون تضاد را برای مدیر سیستم ایجاد می‌کند. الگوریتم ارائه شده قادر به تشخیص انواع حالات تضاد در شرایط قواعد و همچنین پارامترهای رمزنگاری است. همچنین تضادها را در حالت میانگین با کارایی بهتری نسبت به روش‌های موجود می‌یابد. این مساله برای شرایط پویا، یعنی زمانی که سیاست را مرتب به‌روز می‌کنیم، اهمیت زیادی دارد.

۷- مراجع

[1] Al-Shaer, E. and H. Hamed, *Firewall policy advisor for anomaly detection and rule editing.* IEEE/IFIP Integrated Management IM, 2003.

[2] Yang, Y., et al., *IPsec/VPN security policy correctness and assurance.* Journal of High Speed Networks, 2006. 15(3): p. 275-289.

[3] Hari, A., S. Suri, and G. Parulkar, *Detecting and resolving packet filter conflicts.* INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 2000.

[4] Kent, S. and R. Atkinson, *RFC 2401: Security architecture for the Internet Protocol, November 1998.* Proposed Standard.

[5] Fu, Z. and S.F. Wu, *Automatic Generation of IPSec/VPN Security Policies In an Intra-Domain Environment.* Proceedings of the 12th International Workshop on Distributed System Operation & Management (DSOM 2001), 2001: p. 279-290.

[6] Hazelhurst, S., *Algorithms for Analyzing Firewall and Router Access Lists.* Arxiv preprint cs.NI/0008006, 2000.

[7] Lind-Nielsen, J., *BuDDy-a binary decision diagram package*. 1999, Department of Information Technology, Technical University of Denmark.