مجله علمی ـ پژوهشی «پدافند الکترونیکی و سایبری»
سال اول، شماره ۳، پاییز ۱۳۹۲، ص ۵۱-۴۳

طراحی و ساخت پایگاه دانش سیستم خبره برای آزمون امنیت شبکه

محمدعلی جوادزاده[۱*]، محمدرضا کنگاوری[۲]، سید جواد فتحی[۳]
۱- دانشجوی دکتری، دانشگاه علم و صنعت ایران، ۲- دانشیار، دانشگاه علم و صنعت ایران
۳- دانشجوی کارشناسی ارشد، دانشگاه امام حسین (ع)

(دریافت۲۴/ ۸۹/۰۸، پذیرش: ۹۲/۰۴/۰۹)

چکیده

مرحله تولید پایگاه دانش سیستم‌های خبره، تنگنای طراحی سیستم‌های خبره محسوب می‌شود. هزینه انجام این مرحله از ابعاد مختلف از قبیل زمان، سرمایه، نیروی انسانی، دقت و مانند آن به‌حدی است که بخش اعظم هزینه تولید سیستم خبره محسوب می‌شود. موفق‌ترین روش برای برخورد با این تنگنا، توسعه ابزارهای خاص اخذ دانش از انسان خبره است. این ابزارها که تک منظوره هستند، به انسان خبره امکان می‌دهد پایگاه دانش سیستم خبره را با هزینه مناسبی تولید نماید. هدف مقاله حاضر، شرح طراحی زبان مدل‌سازی دانش امنیت شبکه NSKMAL و اشاره‌ای به محیط گرافیکی NSKTOOL است که جهت تولید پایگاه دانش سیستم خبره تحلیل‌گر امنیت شبکه طراحی و تولید شده‌اند. انسان خبره امنیت شبکه قادر است با استفاده از محیط گرافیکی NSKTOOL دانش امنیت را فرموله و به پایگاه دانش منتقل نماید. نتیجه تعامل انسان خبره و NSKTOOL به مجموعه‌ای از دستورات زبان NSKMAL تبدیل شده که متعاقباً توسط مفسر زبان NSKMAL تفسیر و تغییرات لازم در پایگاه دانش اعمال می‌شود.

واژه‌های کلیدی: امنیت شبکه، دانش، سیستم خبره، پایگاه دانش، مدل‌سازی دانش.

۱. مقدمه

آسیب‌پذیری شبکه‌های کامپیوتری به‌عنوان زیرساخت فناوری اطلاعات، یکی از مشکلات مهم این حوزه محسوب می‌شود. بخش عمده این آسیب‌پذیری‌ها به‌علت پیکربندی‌های نادرست در نرم‌افزار و سازمان شبکه است. از این رو متخصصین شبکه، از ابزارهای امنیتی مختلفی استفاده می‌کنند تا از منابع و سرویس‌های ارزشمند در مقابل تهدیدات محافظت به‌عمل آورند. ابزارهای امنیتی به‌تنهایی نمی‌توانند دانش لازم را جهت همبستگی و چگونگی در کنار هم قراردادن این ابزارها در اختیار کاربران آنها قرار دهد. برای رسیدن به امنیت مطلوب، به‌ناچار سازمان‌ها باید از متخصصان حرفه‌ای برای هدف خود استفاده کنند (مانند Red Team). این متخصصان به داده‌های جمع‌آوری شده نظم و ترتیب داده و هر نوع حمله‌ای را تحلیل می‌کنند. به‌عنوان مثال ایشان نموداری از وضعیت آسیب‌پذیری‌های موجود در سیستم‌ها را که می‌توانند منجر به بروز حمله شوند، تهیه می‌کنند. از این رو نیاز مبرمی برای به‌دست آوردن درک عمیق از گزارش‌های امنیتی استخراج شده وجود دارد تا مشخص شود که واقعاً چه چیزی در پشت صحنه اتفاق می‌افتد. برای مثال باز بودن پورت غیرضروری روی یک ماشین خاص می‌تواند منجر به یک حمله ناشناخته شود. بنابراین باید کاوشی عمیق در مورد چگونگی انجام یک حمله انجام داد. حمله‌های امنیتی با اجرای یک یا چند اکسپلویت انجام می‌شود. اکسپلویت برنامه‌ای است که یک یا چند آسیب‌پذیری موجود در نرم‌افزار نصب شده که سبب ایجاد یک رفتار غیرمنتظره در سیستم نهایی می‌شود را آشکار می‌سازد. در گذشته، تلاش‌هایی به‌منظور توصیف مفاهیم حمله صورت گرفته که یکی از آنها توسط Templeton و Levitt انجام شده که اجزای تشکیل دهنده حمله و چگونگی وابستگی آنها به یکدیگر را مدل می‌کند [۶]. در این روش، حمله به اجزای سازنده خود تجزیه می‌شود. با این کار مطالعه نیازمندی‌های اجزای حمله و تأثیر آنها بر محیط اطراف امکان‌پذیر می‌شود. یکی از الزامات اصلی برقراری امنیت در شبکه‌های دارای داده‌های حساس، به‌کارگیری رویکرد دفاع در عمق، در طراحی و پیاده‌سازی این سیستم‌ها و شبکه‌هاست. سیستم‌های تشخیص و جلوگیری از نفوذ، از زیرسیستم‌های اصلی این رویکرد به‌شمار می‌روند. مقابله با حملات، وظیفه اصلی این زیرسیستم‌ها است. در سیستم‌های جدید، تلفیق زیرسیستم‌های تشخیص نفوذ و رویدادنگاری به‌عنوان روشی در جهت بهبود تشخیص نفوذ به‌کار رفته است. این سیستم‌ها در بهترین حالت در زمان رخ دادن حمله به صورت برخط و بلادرنگ، آن را تشخیص داده و در صورت امکان از آن جلوگیری می‌کنند. در راستای تکمیل این لایه دفاعی، بهترین روش، استفاده از دانش مهاجمان برای بررسی سیستم‌ها و شبکه‌های سازمان جهت شناخت نقاط آسیب‌پذیر و راه‌های نفوذ است. این فعالیت تحت عنوان آزمون نفوذ به صورت دوره‌ای توسط گروه‌های متخصص با هزینه‌های بسیار زیاد انجام می‌شود. به‌دلیل هزینه‌های هنگفت این فعالیت‌ها، هم از نظر مالی و هم از نظر توان تخصصی مورد استفاده، اجرای آزمون بیشتر از دوبار در سال توصیه نمی‌شود.

با رشد فناوری‌های دانش، استفاده از سیستم‌های مبتنی بر فناوری سیستم خبره می‌تواند با قرار گرفتن در شبکه، اصلی‌ترین فعالیت‌هایی که یک آزمونگر در راستای کشف آسیب‌پذیری‌ها انجام می‌دهد را به اجرا گذاشته و در پایان نتایج را به‌صورت گزارش‌هایی

*ایمیل نویسنده پاسخگو: javadzadeh@iust.ac.ir

در اختیار مدیران شبکه قرار داده یا هشدارهای لازم را صادر کند. این‌گونه سیستم‌ها اگرچه نیاز به آزمون‌های دوره‌ای توسط تیم‌های تخصصی را از بین نمی‌برد، ولی با اجرای اصلی‌ترین آزمون‌ها در فواصل بررسی تیم‌های تخصصی، ضمن کاهش هزینه‌ها، امکان مناسبی در جهت مدیریت آسیب‌پذیری سیستم‌ها و شبکه‌ها را فراهم می‌کند. هدف از به‌کارگیری این سیستم خبره، صرفه‌جویی در زمان، هزینه، نگهداری دانش تجربیات قبلی و بالا بردن دقت تحلیل است.یکی از قسمت‌های مهم سیستم خبره، پایگاه دانش است.

ایجاد پایگاه دانش یکی از سخت‌ترین مراحل تولید سیستم خبره بوده و تنگنای طراحی محسوب می‌شود [۷]. به‌دلیل سازگاری دانش این حوزه با قوانین، دانش متخصصان مربوط، به‌طور عمده با زبان نمایش دانش به‌شیوه قوانین فرموله می‌شود. اما چالش مهم در استفاده از این سیستم‌های خبره، تغییرات زود هنگام در دانش متخصصان موضوع است. استفاده از سیستم خبره تحلیل‌گر امنیت شبکه، گرچه در خصوص انجام استنتاج و ارائه نتایج تحلیل مناسب است، ولی جهت به‌کارگیری توسط متخصصین امر برای تولید پایگاه دانش و ویرایش آن، از انعطاف لازم برخوردار نیست.

از آنجایی‌که کاربران سیستم خبره تحلیل‌گر امنیت، انسان‌های خبره در زمینه دانش امنیت شبکه بوده و در دانش مهندسی متخصص نیستند و همچنین بیان مفاهیم پیچیده نیاز به یک زبان پیچیده و فنی دارد، بنابراین رابط انسان- ماشین (کامپیوتر) در این ابزارها می‌بایست به‌نحوی طراحی شود که از یک طرف انسان خبره بتواند به‌راحتی دانش خود را به سیستم خبره تحلیل‌گر امنیت منتقل نماید و از طرف دیگر سیستم باید او را در جهت انتقال تمامی دانش خود در زمینه مورد نظر تشویق و یادآوری کند. بنابراین مطالعه و تحقیقات در زمینه طراحی و تولید یک ابزار مؤثر برای اخذ دانش امنیت از خبره امنیت و تولید پایگاه دانش آغاز شده که ضمن تسهیل فرایند تولید پایگاه دانش، تا حد زیادی انسان خبره را در جهت انجام وظایف خود پشتیبانی و در مواردی نیز مساعدت نماید. نتیجه مطالعات و تحقیقات انجام شده در مقاله حاضر، به‌صورت مختصر ارائه شده است. در ادامه ابتدا به معرفی اجمالی سیستم خبره تحلیل‌گر امنیت پرداخته می‌شود و به‌طور خاص توضیحاتی در موضوع پایگاه دانش ارائه می‌شود. سپس مطالبی در ضرورت طراحی یک زبان مدل‌سازی جهت فرموله کردن دانش امنیت بیان می‌گردد. سپس زبان مدل‌سازی دانش امنیت NSKMAL، که توسط تیم نویسندگان این مقاله طراحی و پیاده‌سازی شده است، تشریح و محیط گرافیکی NSKTOOL معرفی می‌شود. در پایان نیز نتیجه‌گیری و پس از آن مراجع ذکر شده است. برای بهره بردن از این مقاله، دانستن اطلاعات تخصصی دانش امنیت ضرورت ندارد، زیرا تأکید این مقاله بیشتر به جنبه‌های تخصصی نرم‌افزار مدل‌سازی دانش معطوف شده است.

۲. کارهای مرتبط

امنیت سیستم‌های کامپیوتری یکی از الزامات به‌کارگیری فناوری اطلاعات محسوب می‌شود. تحقیقات قابل توجهی با استفاده از شیوه‌های مختلف در این زمینه صورت گرفته است. رویکردهایی که به تحقیقات ما وابسته است، به‌همراه نقاط ضعف آنها در ادامه بیان می‌شود.

۲-۱. بررسی‌های آسیب‌پذیری Hard- Coding

در سال ۱۹۸۷ رابرت بدوین[1] مقاله‌ای منتشر کرد که در آن روشی برای تحلیل مبتنی بر قاعده به‌نام Kaung ارائه شده بود [۸]. پس از آن دنیل[2] و یوگن[3] این روش را به‌صورت یک بررسی کننده امنیتی سودمند بهبود دادند [۹]. تا آن زمان این تلاش‌ها، آسیب‌پذیری را تنها در یک میزبان مطرح می‌کرد. پس از آن تحقیقات دیگری انجام گرفت که کار Kaung را روی چند میزبان در شبکه یکسان گسترش می‌داد و Net Kaung نام داشت [۱۰]. متأسفانه روش Kaung بررسی آسیب‌پذیری را به‌صورت Hard-Coded در پیاده‌سازی انجام می‌داد. با وجود اینکه آن روش در زمان خود از کارایی لازم برخوردار بود، اما به دلیل اینکه امروزه با رشد سریع کشف آسیب‌پذیری مواجه هستیم، این روش به دلیل اینکه امروزه هر بررسی کننده امنیتی باید بتواند ویژگی‌های رسمی آسیب‌پذیری‌ها را از منابع مختلف دریافت کند ناکارآمد است. علاوه بر این، مشاهده شده که بیشتر حملاتی که امروزه رخ می‌دهند، ناشی از حملات چند مرحله‌ای روی چند میزبان هستند.

۲-۲. بررسی مدل

بررسی مدل به‌طور اساسی یک سیستم انتقال وضعیت است که به بررسی اینکه آیا سیستم از وضعیت درست به وضعیت دیگری منتقل شده است یا خیر می‌پردازد [۱۱]. به‌کار گرفتن بررسی مدل در امنیت شبکه به‌نحوی است که در آن، حمله به سیستم سبب انتقال از وضعیت فعلی به وضعیت دیگری خواهد شد. متأسفانه آن‌طور که زیمینگ[4] ذکر کرده، اشکال بررسی مدل این است که بیشتر ترتیب‌های انتقال وضعیت سیستم بازرسی می‌شوند و در مقیاس وسیع به انفجار فضای وضعیت منجر می‌شود [۱۲].

۲-۳. تحلیل گراف حمله

روش تحلیل گراف حمله در تحقیقات بسیار مورد توجه قرار می‌گیرد. هدف از این روش، به‌دست آوردن یک گراف مستقل از اکسپلویت است. گراف حمله برای تحلیل فعالیت‌هایی که مهاجم برای دسترسی به هدف انجام می‌دهد مورد استفاده قرار می‌گیرد. متأسفانه چندین مشکل در این زمینه به‌صورت مختصر در مقاله لیپمن[5] [۱۳] آورده شده است.

[1] Robert Badwin
[2] Daniel
[3] Eugene
[4] Xinming
[5] Lippmann

۲-۴. برنامه نویسی منطقی

این روش توسط زیمینگ [۱۴] و سوداکار[1] [۱۵] در چارچوب تحلیل امنیتی مبتنی بر Datalog به‌نام MulVAL ارائه شده است [۱۲]. MulVAL بر اساس وضعیت آزمایشی، اکسپلویت‌هایی را که می‌توانند اجرایی باشند دنبال می‌کند. بر اساس مقاله [۱] MulVAL دارای کاستی‌هایی زیر است:

– MulVAL مبتنی بر Datalog است که تنها می‌تواند تحلیل امنیتی را به‌صورت آفلاین انجام دهد. اگرچه برای هدفی که MulVAL در نظر دارد قابل قبول است اما اعتقاد بر این است که با تحلیل آنلاین می‌تواند بهبود پیدا کرده و اطلاعات امنیتی جدید را تشخیص و به تحلیل‌گر داده شود.

– مدل‌سازی دامنه MulVAL به‌شکل مستندات Datalog است که در مقیاس وسیع می‌تواند به‌صورت غیرقابل نگهداری تولید شود. اطلاعات یک موجودیت واحد، بین مستندات مختلف توزیع شده که درک مدل دامنه را برای به‌دست آوردن و نگهداری مشکل‌تر می‌کند.

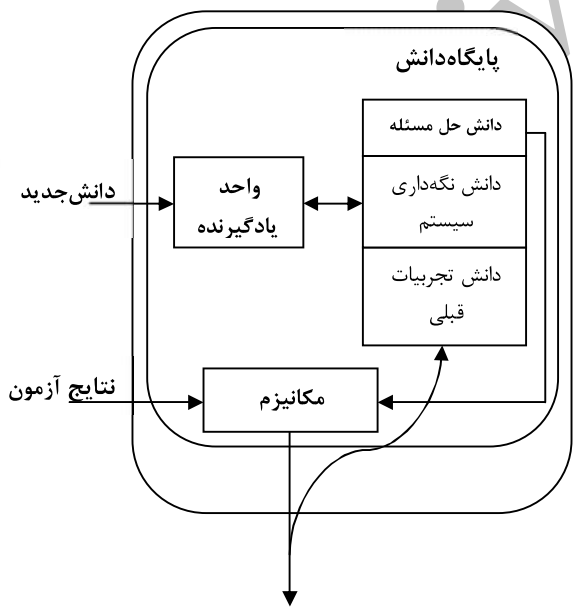
– Datalog بیشتر برای مقاصد آکادمیک مورد استفاده قرار می‌گیرد و به‌منظور استفاده گسترده از چارچوب‌های باز، به‌راحتی قابل سازگاری است و زبان برنامه نویسی استفاده شده نقش مهمی را در آن ایفا می‌کند.

۲-۵. استفاده از سیستم خبره

تسودیک[2] و سامر[3] از مرکز تحقیقات IBM در [۱۶] سیستم خبره‌ای جدید، ممیزی امنیت به‌نام AudES ارائه کرده‌اند. هدف ارائه‌کنندگان این مقاله، کاهش هزینه و زمان لازم برای ممیزی امنیت به‌صورت دستی است. به‌دلیل اینکه برای کسب دانش سیستم پیشنهادی از ISO 27000 استفاده شده، بنابراین سیستم فقط قادر به مدیریت امنیت اطلاعات بوده و در حوزه تشخیص آسیب‌پذیری کاری انجام نشده است. آنات هواو[4] و همکارانش، سیستم خبره‌ای مبتنی بر ISO17799 و استانداردهای NIST پیشنهاد کرده‌اند [۱۷]. در سیستم پیشنهادی، نمایش دانش به‌صورت قوانین بوده و از استنتاج رو به جلو استفاده شده است. به‌دلیل اینکه منابع کسب دانش سیستم مبتنی بر استاندارد است، بنابراین فقط قادر به مدیریت امنیت اطلاعات به‌صورت لیست‌های کنترلی بوده و از ارزیابی آسیب‌پذیری‌های موجود ناتوان است. پانگالوس[5] و همکارانش با استفاده از تکنولوژی سیستم خبره، روشی جدید برای مدیریت پویای کنترل‌های دسترسی ارائه نموده‌اند [۱۸]. هدف تحقیق حاضر، پیشنهاد سیستمی است که با استفاده از فنون‌های سیستم خبره، ورود غیرمجاز را تشخیص داده و از آن جلوگیری نماید. همچنین در این تحقیق نحوه استفاده از سیستم‌های خبره مبتنی بر قانون، برای مدل‌های جدید کنترل‌های دسترسی نیز توصیف شده است. در تمامی تحقیقات انجام شده، کمبود سامانه‌ای که بتواند دانش پویای امنیت را در زبان سطح بالا از خبره امنیت دریافت و سپس آنرا فرموله نموده و در پایگاه دانش درج نماید، احساس می‌شود. از این رو حضور مهندس دانش در به‌روز کردن دانش، بسیار پررنگ است. به‌عبارت دیگر، حضور دائمی مهندس دانش باعث می‌شود که هزینه به‌روزرسانی دانش سیستم خبره بسیار زیاد شود و در نتیجه استفاده از سیستم خبره را مقرون به‌صرفه ننماید. در این مقاله تلاش می‌شود با استفاده از طراحی و پیاده‌سازی یک زبان مدل‌سازی جهت فرموله کردن دانش امنیت (NSKMAL) و محیط گرافیکی ابزار مربوطه (NSKTOOL) این مشکل برطرف شود.

۳. معماری سیستم خبره تحلیل‌گر امنیت

سابقه موضوع این مقاله (از نظر سنخیت دانش و فرموله سازی دانش به زبان قوانین) به تحقیقات دیگری تحت عنوان "طراحی و پیاده‌سازی سیستم خبره تحلیل‌گر نتایج آزمون تونل باد" برمی‌گردد، که نتایج آن در اولین کنفرانس بین‌المللی هوافضا در دانشگاه شریف ارائه و مورد استقبال متخصصین هوافضا واقع شد [۱]. همچنین دو پایان نامه کارشناسی ارشد [۲ و۳] و یک گزارش فنی در این خصوص ارائه شده است [۴]. مقاله‌ای دیگر نیز در دومین همایش روش‌های تحقیق در علوم و فنون مهندسی تحت عنوان "به‌کارگیری تکنیک‌های هوش مصنوعی در استخراج دانش از منابع اطلاعاتی" در همین زمینه ارائه گشده است [۵]. معماری عمومی سیستم خبره تحلیل‌گر امنیت شبکه مطابق شکل **۱** است.



شکل ۱: ساختار عمومی سیستم خبره تحلیل‌گر امنیت شبکه

این سیستم از قسمت‌های پایگاه دانش، واحد یادگیرنده، مکانیزم استنتاج و رابط کاربر تشکیل می‌شود. محور بحث این بخش مربوط به قسمت پایگاه دانش است.

[1] Sudhakar
[2] Tsudik
[3] Summers
[4] Anat Hovav
[5] Pangalos

۳-۱. پایگاه دانش

دانش مورد استفاده در سیستم خبره تحلیل‌گر باید بتواند به‌اندازه‌ای صحیح و دقیق باشدکه نیاز استفاده از انسان را به حداقل ممکن تقلیل دهد. این دانش شامل:

الف) واقعیت‌ها، فرضیه‌ها، احکام و قضایا : این بخش از دانش، قابلیت فرموله شدن را دارند و می‌توان تحت فرمول‌های مختلف در منابع متعدد یافت.

ب) مهارت: استفاده از واقعیت‌ها و فرضیات احکام و قضایا در جهت حل مسئله و تولید پاسخ است. به‌سختی قابلیت فرموله شدن دارد و عمده‌ترین منبع آن نیز انسان خبره است.

به‌طور کلی برای طراحی پایگاه دانش مراحلی وجود دارد که این مراحل را برای ایجاد پایگاه دانش سیستم خبره تحلیل‌گر در نظر گرفته و مرحله به مرحله سعی شده تا به آن شکل داده شود.

در طراحی و تولید سیستم‌های خبره، کار استخراج دانش و جمع‌آوری آن به‌عنوان تنگنای طراحی محسوب می‌شود و از اهمیت فوق‌العاده‌ای برخوردار است. هرگونه خطا در مرحله اخذ دانش منجر به‌وجود خطا در پایگاه دانش و در نهایت در پاسخ تولید شده به‌وسیله سیستم خبره می‌شود. بنابراین در سیستم خبره تحلیل‌گر، نهایت دقت و اهتمام در تولید دانش صحیح و جمع‌آوری آن، جهت بالا بردن اعتبار سیستم خبره تولید شده به‌کار رفته است. دانشی که برای حل این مسئله استخراج می‌شود، شامل دو بخش دانش واقعیات و مهارت است. این دانش از منابع اصلی دانش که شامل منابع انسانی و منابع غیرانسانی است به روش‌های زیر مورد استخراج قرار گرفته است:

الف) منابع غیرانسانی: با استفاده از مستنداتی مانند مستندات نتایج آزمون‌های قبل، کتب، مقالات و همچنین به‌کمک انسان خبره بخشی از دانش مسئله استخراج و جمع‌آوری گردید.

ب) منابع انسانی: برای استخراج دانش از منابع انسانی به‌عنوان کامل‌ترین منبع بخش مهارت دانش، از روش مصاحبه و ارتباط نزدیک استفاده شد. مصاحبه‌ها هم به‌صورت رو در رو و با انسان‌های خبره و هم از طریق پرسش‌نامه‌ها به‌صورت مکتوب انجام گردید. علی‌رغم اینکه انسان خبره مهم‌ترین منبع دانش حل مسئله محسوب می‌شود، لیکن استخراج دانش از این منابع بسیار پیچیده و گلوگاه اصلی فرایند اخذ دانش است. استفاده از ابزارها و زبان‌های سطح بالا که امکان ثبت و ذخیره دانش انسان خبره را فراهم می‌کنند، به‌عنوان محور مهم تحقیقاتی در زمینه اخذ دانش محسوب می‌شود که هدف تحقیقات انجام شده موضوع مقاله است. دانش جمع‌آوری شده، قابل پردازش به‌وسیله ماشین نیست، به‌همین علت باید آن را به زبان ماشین ترجمه کرد. همچنین زبان‌های مورد استفاده و متداول نمایش دانش مورد بررسی قرار گرفتند. زبان‌های رایج برای فرموله کردن دانش شامل زبان قاب، منطق، قوانین، شبکه معنا و رویه است. برای فرموله نمودن دانش جمع‌آوری شده حل مسئله، به دلایل زیر ترکیبی از روش نمایش قوانین و رویه‌ها استفاده شده است: ۱- سادگی روش فرموله کردن دانش، ۲- داشتن مکانیزم استنتاج ساده‌تر نسبت به سایر زبان‌های فرموله‌سازی دانش، ۳- قابلیت سادگی در توسعه دانش، ۴ - امکان نمایش دانش غیر قطعی، ۵- نزدیک بودن دانش موضوع این تحقیق به قوانین و ۶- رویه‌ای بودن بخشی از دانش.

پایگاه دانش سیستم خبره تحلیل‌گر از سه بخش تشکیل شده که شامل: ۱- دانش حل مسئله، ۲- دانش نگهداری سیستم و ۳- دانش تجربیات قبلی.

دانش نگهداری سیستم، امکان به‌روز کردن دانش سیستم را فراهم می‌کند تا سیستم خبره به‌مرور زمان دچار افول و زوال نشود. دانش تجربیات قبلی نیز کمک می‌کند تا قبل از آزمون شبکه، سیستم خبره با اخذ مشخصات شبکه، به‌طور نسبی پیش‌بینی کند که آیا شبکه حاضر با توجه به تجربیات قبلی، در آزمون امنیت موفق خواهد بود یا خیر؟

۳-۲. دانش حل مسئله

در قسمت دانش حل مسئله، دانش مورد استفاده در تحلیل نتایج آزمون امنیت ذخیره شده است. این دانش توسط مکانیزم استنتاج مورد استفاده قرار می‌گیرد. دانش حل مسئله توسط مهندس دانش، جمع‌آوری شده و با تلفیق زبان‌های قوانین و رویه فرموله شده است. این دانش شامل موارد زیر است:

الف– دانش مورد استفاده در تحلیل نتایج آزمون امنیت.

ب– دانش تجربیات حاصل از نتایج انجام آزمون امنیت.

ج– دانش چگونگی برطرف نمودن عیوب تشخیص داده شده است.

د– دانش مورد استفاده در ارائه پیشنهادات.

فعالیت‌های این سیستم خبره هر چهار بخش را شامل می‌شود. دانش تحلیل نتایج در یک بخش مستقل ذخیره می‌شود. این بخش توسط مکانیزم استنتاج جهت تحلیل آزمون‌ها، مورد استفاده قرار می‌گیرد و دانش جدیدی که ممکن است طی حیات سیستم به آن اضافه گردد، به این بخش اضافه می‌شود.

بر اساس نوع خاص دانش جمع آوری شده در سیستم خبره تحلیل‌گر امنیت، ترکیبی از قوانین و رویه‌ها برای فرموله کردن دانش در این مرحله استفاده شده است. پایگاه دانش سیستم خبره تحلیل‌گر امنیت دارای ساختار خاصی است که از یک طرف تحلیل نتایج آزمون امنیت را امکان پذیر می‌نماید و از طرف دیگر نگهداری سیستم خبره را میسر می‌سازد.

از زبان قوانین برای بیان دانش مورد نظر استفاده شده است. اما موارد بسیاری وجود دارد که قوانین به‌تنهایی نمی‌توانند دانش مورد نظر را بیان نمایند و باید قابلیت‌های زبان رویه‌ای را در اختیار داشته باشند تا بتوانند دانش مورد نظر را بیان نمایند. در سیستم خبره تحلیل‌گر، هر دو زبان بیان دانش قوانین و رویه به طور مطلوبی با هم به‌کار گرفته شده است. به‌عنوان مثال می‌توان یک قانون را به شرح زیر تعریف نمود:

str =Aba1:Vias^Rzac^Fdes;
str = قانون شروع کننده
ba1= نرم افزار
ias = آسیب‌پذیر
zac = نرم‌افزار جزء نرم افزارهای آسیب پذیر یا در حال ریسک قرار دارد
des = تمامی نرم افزارهای وابسته را نیز شناسایی کن / تمامی دارایی مورد استفاده توسط نرم‌افزار را در حالت ریسک قرار بده.

به هر تعداد دلخواه و لازم در تعریف یک قانون می توان از رویه‌ها، خصیصه- ارزش و قوانین استفاده نمود. در بین شرایط از کاراکتر "^" به‌عنوان علامت AND استفاده می‌شود. زوج خصیصه و ارزش با کاراکتر ":" از یکدیگر جدا می‌شوند. در انتهای تعریف هر قانون از کاراکتر ";" استفاده می‌شود. در مثال فوق Aba1:Vias یک زوج خصیصه و ارزش است که به‌طور مشخص Aba1 یک خصیصه است زیرا با حرف A آغاز شده و Vias ارزش یا مقدار است زیرا با حرف V آغاز شده است. همچنین Rzac یک قانون و Fdes یک رویه است. قوانین با حرف R و رویه‌ها با حرف F آغاز می‌شوند. رویه مذکور به‌صورت یک برنامه اجرایی بر روی رسانه ذخیره‌سازی، ذخیره می‌شود. رویه مذکور می‌تواند رویه‌ای همانند برنامه ذیل باشد:

۱- شروع،
۲- نرم‌افزار را در متغییر SW قرار بده،
۳- اگر نرم‌افزار آسیب‌پذیر بود، آنگاه برو به ۵ در غیر این‌صورت برو به ۴،
۴- نرم‌افزار آسیب نیست. توقف کن،
۵- تمامی نرم‌افزارهای وابسته را در متغییر dep قرار بده،
۶- متغییر securitystate مربوط به نرم‌افزار را به حالت خطرناک به‌روزرسانی کن،
۷- تمامی دارایی‌های موجود روی سیستم را به‌دست بیاور،
۸- اگر نرم‌افزار آسیب‌پذیر از دارایی‌های همان سیستم یا سیستم‌های همسایه استفاده می‌کرد:
آنگاه تمامی دارایی‌های مورد استفاده توسط نرم‌افزار را در حالت خطرناک قرار بده،
۹- پایان.

توجه: از مرحله ۵ به بعد برای تمامی نرم‌افزارهای وابسته، جداگانه اجرا می‌شود.

رویه Fdes را می‌توان به یکی از زبان‌های رویه‌ای سطح بالا پیاده‌سازی کرد. برای مثال به رویه زیر که یک نمونه از مدل‌سازی الگوهای حمله CAPEC است، می‌توان اشاره نمود:

۱- CAPEC: " دسترسی به ویژگی‌های اصلی توسط ACL به درستی کنترل نمی‌شود"

```
Rule
when
در صورت داشتن یک مهاجم #
$attacker : User(attacker == true)
مهاجم نرم‌افزاری را به‌عنوان هدف انتخاب می‌کند #
$software : Software()
در صورت وجود یکی از عیوب زیر در نرم‌افزار #
exists(
Weakness(software == $software ,
identifier in ("CWE-285", "CWE-732",
"CWE-276", "CWE-693",
"CWE-721", "CWE-434")
)
)
و در صورت وجود یک اکانت فعال بر روی آن، مهاجم اکانت را #
به‌دست آورده
exists(
UserAccount(software == $software , state ==
UserAccountState.ACTIVE
) from $attacker.getAccounts()
)
مهاجم غیرمجاز وارد نرم‌افزار شده و به آن دسترسی پیدا می‌کند #
eval(
$attacker.getHost().canReach(
$software.getHost()
)
)
then
print("[CAPEC-1] Attacker '%s' می‌تواند غیرمجاز به نرم‌افزار
دسترسی پیدا کند '%s'", $attacker.getFullName(),
$software.toString());
end
```

۴. ضرورت طراحی یک زبان مدل‌سازی جهت فرموله کردن دانش آزمون امنیت شبکه

ایجاد یک زبان جهت فرموله نمودن دانش قوانین به زبان سطح بالا کمک می‌کند که انسان خبره بتواند با یادگیری آن، به‌راحتی دانش خود را به زبان سطح بالا، برنامه نویسی نموده و به سیستم ارائه دهد. Syntax این زبان باید به زبان محاوره‌ای انسان خبره نزدیک باشد و تا حد امکان نمادی (symbolic) باشد. زبان مدل‌سازی طراحی شده در سیستم خبره تحلیل‌گر، بسیار مناسب جهت برنامه‌نویسی و بیان دانش قوانین است. محدوده این زبان شامل آن دسته از دستوراتی می‌شود که بتوان با آن دانش قوانین را به سیستم ارائه نمود. با نگاه اجمالی به ساختار قوانین، دیده می‌شود که یک قانون به‌طور معمول از زوج‌های خصیصه، ارزش و قوانین دیگر تشکیل می‌شود و جایی برای بیان دانش رویه‌ای در قانون دیده نشده است [۱۹].

نکته منحصر به‌فرد در سیستم خبره تحلیل‌گر، استفاده تلفیقی از زبان قوانین و زبان رویه‌ای در فرموله کردن دانش است. این تلفیق در نوع خود، برای اولین مرتبه در این سیستم مورد استفاده واقع شده است. از این رو زبان مدل‌سازی مورد نظر باید علاوه بر داشتن دستوراتی همچون تعریف قانون و تعریف زوج خصیصه- ارزش، دستور تعریف رویه را نیز دارا باشد. همچنین دستوراتی برای بیان پایگاه دانش و متعلقات آن، اصلاح دانش، حذف و اضافه نمودن قانون در پایگاه دانش و موارد ضروری دیگر را داشته باشد [۳]. برای بهره بردن از چنین زبانی ابتدا باید گرامر مناسب آن زبان طراحی گردد. در ادامه به طراحی این زبان و گرامر آن پرداخته می‌شود.

۵. زبان مدل‌سازی دانش امنیت شبکه NSKMAL

دانشی که بر مبنای گرامر زبان انتزاعی مدل‌سازی امنیت شبکه NSKMAL پیاده‌سازی یا فرموله و ذخیره شود، خود پایگاه دانش محسوب می‌شود. بنابراین باید تمامی شرایط یک پایگاه دانش را داشته باشد. بدین معنی که باید صحت این پایگاه دانش در شرایط مختلف تامین شود. حالات مختلف پایگاه دانش شامل موارد زیر است:

۱- ساخته شدن پایگاه دانش

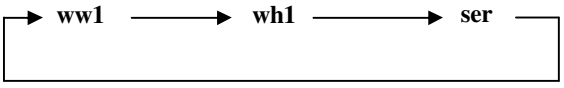
۲- ویرایش پایگاه دانش

۳- نگه‌داری پایگاه دانش.

ساخته شدن پایگاه دانش: زمانی که پایگاه دانش در وضعیت ساخته شدن است، باید امکاناتی در اختیار کاربر سیستم قرار گیرد که وی را در ایجاد پایگاه دانش صحیح و بدون خطا یاری دهد. می‌دانیم که کاربران سیستم خبره تحلیل‌گر، انسان‌های خبره در زمینه دانش امنیت بوده و کاربران مهندسی دانش نیستند و همچنین بیان مفاهیم پیچیده نیاز به یک زبان پیچیده و فنی دارد. به‌طور مثال یکی از مواردی که هم در ایجاد و هم در ویرایش پایگاه دانش باید مورد توجه قرار گیرد، احتمال وقوع حلقه یا چرخه در قوانین پایگاه دانش است. در چنین مواقعی، سیستم باید انسان خبره را از وقوع چرخه آگاه سازد تا وی در صورت صلاح‌دید اجازه ایجاد چرخه و یا عدم ایجاد آن را صادر نماید، زیرا با وقوع چرخه در قوانین، احتمال تزلزل در صحت دانش وجود دارد [۳]. برای مثال قوانین زیر (بدون توجه به مفهوم نام هر قانون)، گویای وقوع چرخه غیرمستقیم است.

ww1 ← Aper:Vmeg^Rwh1 ;
wh1 ← Fper ^ Rser ^ Amor: Vlue;
ser ← Fmai ^ Rww1;

علت احتمال نامعتبر بودن این است که صحت نتیجه قانون ww1 منوط به صحت نتیجه قانون wh1 و صحت نتیجه قانون wh1 منوط به صحت نتیجه قانون ser و صحت نتیجه قانون ser منوط به صحت نتیجه قانون ww1 می‌باشد. بنابراین یک گراف به صورت شکل ۲ ایجاد می‌شود.



شکل ۲: ایجاد چرخه در قوانین

این قوانین در زمان استنتاج، هر کدام منتظر نتیجه‌گیری از قانون دیگر می‌شوند. بنابراین نمی‌تواند هیچ گونه استنتاجی را برای سیستم خبره انجام بدهند، مگر آنکه تدابیر لازم جهت خروج از چرخه، در داخل قوانین قرار گرفته در چرخه، اندیشیده شده باشد. از این رو باید وقوع چرخه به کاربر یادآوری گردد تا رفتار لازم در رابطه با چرخه ایجاد شده به عمل آید.

ویرایش پایگاه دانش: زمانی که نیاز به ویرایش پایگاه دانش به‌وجود می‌آید، لازم است که پایگاه دانش از یک وضعیت پایدار به وضعیت پایدار دیگر برسد. گاهی انسان خبره لازم می‌داند که با حذف و یا اضافه نمودن به برخی از قسمت‌های دانش سیستم، آن را بهینه و اصلاح نماید. این نیاز زمانی شدت بیشتری پیدا می‌کند که تجربیات انسان خبره افزایش یافته و دانش پیشین را برای حل مسئله کافی نداند. از این رو باید در سیستم خبره، امکاناتی برای بهینه کردن دانش پیش‌بینی شود. برای اصلاح دانش سیستم، ابتدا باید دانش پیشین را حذف نمود، سپس دانش جدید را به سیستم آموخت. لازم به ذکر است که در زبان NSKMAL، می‌توان دانش را ویرایش نمود و نیاز به حذف آن جهت ویرایش نیست. به‌طور مثال در صحت دانش هنگام ویرایش قانون می‌توان به حذف قانون اشاره نمود. برای حذف قانون، چند ویژگی جهت بقای صحت قانون اصلی لحاظ شده که با ذکر مثال بیان می‌شود. در حذف قانون، انسان خبره با چند حالت مختلف روبرو می‌شود که در ذیل بررسی می‌گردد.

اگر از قانون مورد نظر فقط یکی موجود باشد: دو وضعیت ممکن است به‌وجود آید:

الف: اگر قانون مورد حذف فقط در سمت چپ قوانین بود، بعد از دادن پیام اخطار به کاربر، اجازه حذف آن را به کاربر می‌دهد.

مثال: فرض کنید پایگاه دانش شامل قوانین ذیل باشد:

vse=Fvx1^Fvmx^Rpfm^Fvz1;
pfm=Fca1^Fcn1^Fcmz1^Rhys;

در قوانین بالا vse فقط یک مورد و آن هم در سمت چپ قوانین قرار دارد. بنابراین سیستم اجازه حذف آن را به کاربر می‌دهد، زیرا این‌گونه عمل حذف هیچ مشکلی برای عملیات استنتاج ایجاد نمی‌کند.

ب: اگر قانون در سمت راست قوانین باشد: در این صورت قانون حذف نشده و پیام " این قانون قابل حذف نیست و باید قانون در سمت چپ آن ابتدا اصلاح شود" به کاربر داده می‌شود.

مثال:

vse=Fvx1^Fvmx^Rpfm^Fvz1;
hys=Fhcn^Fhcy^Fhmx^Rpsj^Fhcm;

در مثال بالا اگر کاربر قصد حذف قانون pfm را داشته باشد، سیستم به کاربر اطلاع می‌دهد که ابتدا باید قانون vse ازجدول حذف شود، زیرا با حذف شدن کد قانون pfm در وسط قانون vse، در زمان استنتاج، صحت نتیجه‌گیری با اشکال مواجه می‌شود. زیرا کد قانون (همانند کد قانون pfm) که در سمت راست قانون دیگری موجود است دچار نقص می‌شود و نتیجه‌گیری واقعی را درسیستم دچار مشکل می‌کند.

- اگر قانون مورد نظر بیش از یک مورد باشد: دو وضعیت ممکن است به‌وجود آید.

- اگرقانون مورد حذف فقط در سمت چپ باشد:

سیستم به کاربر اجازه می‌دهد که فقط یکی از قوانین را حذف نماید. در مثال ذیل سیستم به کاربر اجازه می‌دهد فقط یکی از vseها از سیستم حذف شود.

vse=Fvx1^Fvmx^Rpfm^Fvz1;
vse =Fco1^Rvse;

زیرا ممکن است قانون vse در سمت راست قوانین دیگر به‌کار رفته باشد و تا زمانی که حداقل یک قانون اصلی vse در پایگاه دانش وجود داشته باشد، موتور استنتاج با مشکل مواجه نمی‌شود.

– اگر قانون در سمت راست قوانین دیگر نیز باشد:

سیستم به کاربر اطلاع می‌دهد که برای حذف قانونی مثل pfm ابتدا باید vse حذف و یا ویرایش شود تا pfm در سمت راست آن نباشد. برای راهنمایی بهتر کاربر، تمام قوانینی که در سمت راست آنها قانون مورد نظر وجود دارد، همانند لیست قوانین زیر برای کاربر نمایش داده شود.

```
vse=Fvx1^Fvmx^Rpfm^Fvz1;
pfm=Fca1^Fcn1^Fcmz1^Rhys;
```

نگهداری پایگاه دانش: در مواردی نیاز است که از پایگاه دانش مراقبت‌های ویژه‌ای به‌عمل آید. از آنجایی که این موارد متنوع است، تنها می‌توان به برخی از آنها همانند موارد ذیل اشاره نمود.

- بر اساس افزایش معلومات انسان خبره، لازم است به دانش سیستم اضافه شود تا سیستم، خبرگی خود را حفظ نماید. بنابراین باید مراقبت نمود که دانش جدید با دانش قبلی در تعارض نباشد،
- نیازهای جدید ایجاب می‌کند تا دانش متناسب با آن نیازها به سیستم اضافه شود. بنابراین نباید تداخل دانش کاربردهای مختلف پدید آید و دانش هر کاربرد جداگانه نگهداری شود،
- تجربه سیستم از تحلیل آزمون‌های انجام شده می‌تواند به مرور زمان به‌طور پیوسته به‌نحو خودکار به پایگاه دانش اضافه شود. مراقبت‌های خاص این افزایش دانش باید اندیشیده شود،
- سایر دلایل.

۶. ساختار گرامر زبان NSKMAL

گرامر زبان NSKMAL همانند گرامر سایر زبان‌های برنامه‌سازی یک گرامر مستقل از متن [۲۰] است. این گرامر از ۴۰ قاعده تولید تشکیل شده است. عنصر ابتدایی در تعریف BNF [۲۰] گرامر زبان NSKMAL، متغییر <Program> است. یک برنامه صحیح در NSKMAL از لحاظ نحوی، رشته‌ای است که می‌تواند با یک اشتقاق موفق از عنصر ابتدایی گرامر زبان NSKMAL که با قاعده تولید <Program> شروع می‌شود به‌دست آید. لیست تمامی قواعد تولید در زیر آمده است.

لیست کلیه قواعد تولید موجود در گرامر زبان NSKMAL:

1. <Program>::= <Creat knowledge base> | <Update knowledge base>
2. <Creat knowledge base> ::= creat_knowledge_base<space> <knowledge base name> <space> begin <space> decleration<space><decleration> {<starter rule> <space> <list of rule>} end.
3. <starter rule>::= starter <rule name>
4. <identifier>::=<letter> | <letter> <letter or digit>
5. <knowledge base name>::= <identifier>
6. <rule name>::=<identifier>
7. <procedure name>::=<identifier>
8. <attribute name>::=< identifier>
9. <file name>::=< identifier>.exe | <identifier>.com
10. <space>::= ␣<space> | ␣
11. <letter or digit>::= <letter><letter or digit> | <digit><letter or digit> |_<letter or digit> |<letter> | <digit>
12. <letter>::= a|b|…|z
13. <digit>::=0|1|2|…|9
14. <sign>::=+|-
15. <string>::=<letter><string>|<digit><string>|<letter>|<digit>
16. <value>::=<number>|<sign><number>
17. <number>::=<digit><number>|<digit>
18. <list of rule>::= <rule><space><list of rule>|<rule>
19. <rule>::=<rule name> = <rule body>;
20. <rule body>::=<body part><space> ^ <space><rule body>|<body part>
21. <body part>::=(<attribute name>:<value>)|<rule name>| <procedure name>
22. <starter rule >::= starter <space><rule name>;
23. <decleration>::= <procedure decleration> <decleration>|<rule decleration> <decleration> | <attribute decleration> <decleration> | <procedure decleration> |<rule decleration> | <attribute decleration>
24. <procedure decleration>::= procedure <space> <list of procedure name>
25. <attribute decleration>::= attribute <space> <list of attribute name>
26. <rule decleration>::= rule <space> <list of rule name>
27. <list of rule name>::=<rule name>,<list of rule name>|<rule name>;
28. <list of attribute name>::= <attribute name>,<list of attribute name> | <attribute name>
29. <list of procedure name>::=< procedure name>\\<drive name>\<path>, <list of Procedure name>|<procedure name>\\ <drive name>\<path>;
30. <drive name>::= a: | b: |c: | d: | e: | f: | g: | h: | i: |A: |B: |C: |D: |E: |F: |G: |H: |I:
31. <path>::=<string>\<path>\ |<string>\ <file name> | <file name>
32. <Update knowledge base>::=update <space><knowledge base name><space> begin <space> decleration<space> <decleration>{<update command>}end.
33. <update command>::=<edit rule><update command>|<insert rule><update command>|<insert procedure> <update command>|<delete Procedure><update command>|<delete rule><update command>|<edit rule>| <insert Procedure>|<insert rule>|<delete Procedure>|<delete rule>
34. <insert rule>::=insert_rule<space><rule>
35. <insert Procedure>::=insert_Procedure <space><list of Procedure name>
36. <delete Procedure>::=delete_Procedure <space><set of procedure name>
37. <set of Procedure name>::=<Procedure name>,<set of Procedure name>| <Procedure name>
38. <delete rule>::=delete_rule<space><set of rule name>
39. <set of rule name>::=<rule name>,<set of rule name>|<rule name>
40. <editrule>::=edit_rule<space><rule> <space> to<rule>

قاعده تولید <Program>: قاعده تولید <Program> عنصر ابتدایی برای گرامر زبان NSKMAL محسوب می‌شود. در واقع یک برنامه می‌تواند یا یک پایگاه دانش تولید کند و یا پایگاه دانش موجود را اصلاح نماید. بنابراین قاعده تولید <Program> به‌شرح ذیل بیان می‌شود (قاعده تولید شماره ۱).

<Program>::= <Creat knowledge base> | <Update knowledge base>

قاعده تولید <Creat knowledge base>: این قاعده تولید باعث می‌شود که برنامه‌نویس بتواند یک پایگاه دانش را به دلخواه خود با هر تعداد قانون و رویه و خصیصه و ارزش تولید نماید (قاعده تولید شماره ۲).

قاعده تولید <identifier>: کلیه اسامی که در یک برنامه به‌کار برده می‌شود، <identifier> محسوب می‌شود (قواعد تولید شماره ۹- ۴).

قاعده تولید <space>: در زبان‌های برنامه‌سازی فضای بین کلمات می‌تواند از یک کاراکتر space و یا بیشتر تشکیل شود (قاعده تولید شماره ۱۰). همچنین در کلیه زبان‌های برنامه‌نویسی سطح بالا، از اعداد و رشته‌ها استفاده می‌شود (قواعد تولید شماره ۱۷-۱۱).

قاعده تولید <list of rule>: این قاعده تولید بدنه اصلی پایگاه دانش را تشکیل می‌دهد. از آنجایی که در قسمت استنتاج، از استنتاج رو به عقب استفاده شده است، لازم است قانون شروع کننده برای عملیات استنتاج توسط انسان خبره با استفاده از گرامر قاعده تولید <starter> مشخص شود (قواعد تولید شماره ۲۲- ۱۸).

قاعده تولید <decleration>: این قاعده تولید امکان تعریف identifier برای اسامی، قوانین، خصیصه‌ها و رویه‌ها را در اختیار می‌گذارد (قواعد تولید شماره ۳۱-۲۳).

قاعده تولید <Update knowledge base>: این قاعده تولید جهت بازنگری بر پایگاه دانش و اصلاح، حذف و یا اضافه کردن بر آن طراحی شده است (قواعد تولید شماره ۳۲ و ۳۳).

قاعده تولید <insert rule>: زمانی که نیاز به اضافه نمودن یک قانون به پایگاه دانش وجود داشته باشد، از این قاعده تولید استفاده می‌شود (قاعده تولید شماره ۳۴).

قاعده تولید <insert Procedure>: زمانی که نیاز به اضافه نمودن یک رویه به لیست رویه‌های موجود در پایگاه دانش وجود داشته باشد از این قاعده تولید استفاده می‌شود (قاعده تولید شماره ۳۵).

قاعده تولید <delete Procedure>: جهت حذف نمودن یک رویه از لیست رویه‌های موجود در پایگاه دانش استفاده می‌شود (قواعد تولید شماره ۳۶ و ۳۷).

قاعده تولید <delete rule>: جهت حذف نمودن یک قانون از لیست قوانین موجود در پایگاه دانش، از این قاعده تولید استفاده می‌شود. اصلاح یک قاعده تولید<edit rule>: برای این عمل از گرامر شماره ۴۰ استفاده می‌شود.

۷. NSKTOOL

یکی از راه‌های ایجاد تعامل با سیستم خبره تحلیل‌گر، استفاده از زبان مدل‌سازی NSKMAL است. اما زبان NSKMAL همه نیازمندی‌های تعامل با این سیستم را پاسخ‌گو نیست. جهت تسهیل و تسریع فرآیند اخذ دانش، ابزار NSKTOOL بر اساس NSKMAL طراحی و پیاده‌سازی گردید. با استفاده از این ابزار، انسان خبره به‌راحتی در یک محیط گرافیکی بر اساس فناوری پنجره می‌تواند تمامی عملیات مربوط را انجام دهد و حاصل این تعامل به‌وسیله یک مترجم به زبان NSKMAL ترجمه و در نهایت پایگاه دانش تولید می‌شود و یا تغییرات لازم بر روی پایگاه دانش موجود انجام می‌گیرد. بدین ترتیب انسان خبره از یک رابط کاربر مناسب برخوردار شده و از پیچیدگی‌های زبان NSKMAL به‌دور است. اموری همچون تولید دانش، رویت دانش موجود در سیستم، اخذ انواع گزارشات از سیستم، انجام عملیات استنتاج، رویت نتایج استنتاج، دیدن پیشنهادات سیستم برای بهبود در روند و انجام آزمون شبکه، تعریف سطوح دسترسی و مشخص نمودن مجوز کاربری برای کاربران و مانند آن، همگی توسط این ابزار ممکن شده است. شرح مراحل طراحی و تولید ابزار NSKTOOL نیاز به مقاله‌ای دیگر دارد، بنابراین در این مقاله به آن پرداخته نمی‌شود.

۸. نتیجه‌گیری

در سیستم خبره تحلیل‌گر نتایج آزمون امنیت، به‌دلیل عدم داشتن رابط کاربر مناسب، تعامل بین انسان و ماشین (کامپیوتر) با مشکل همراه بود. در حالی که با طراحی زبان NSKMAL و توسعه محیط گرافیکی NSKTOOL امکان انتقال دانش انسان خبره امنیت شبکه به پایگاه دانش سیستم خبره تحلیل‌گر به شکل مناسبی امکان‌پذیر گردید. از آنجایی که کاربران سیستم خبره تحلیل‌گر، انسان‌های خبره در زمینه دانش امنیت شبکه بوده و متخصص در مهندسی دانش نیستند و همچنین بیان مفاهیم پیچیده نیاز به یک زبان پیچیده و فنی دارد، بنابراین رابط گرافیکی انسان- ماشین در این ابزار به شکلی طراحی گردید که از طرفی انسان خبره بتواند به‌راحتی دانش خود را به سیستم خبره تحلیل‌گر منتقل نماید و از طرف دیگر در جهت انتقال تمامی دانش خود در زمینه مورد نظر مساعدت گردد. برای طراحی زبان مدل‌سازی، از ۴۰ قاعده تولید در گرامر زبان NSKMAL استفاده شده که منجر به پیاده‌سازی مفسر زبان در محیط مبتنی بر تکنولوژی پنجره گردید. این زبان قادر است امکان تولید پایگاه دانش و ویرایش آن را در اختیار انسان خبره قرار دهد. همچنین برای طراحی NSKTOOL، از ۲۵ پنجره اصلی به‌عنوان ابزار رابط گرافیکی استفاده شده است. برای توسعه و بهینه شدن NSKMAL و محیط گرافیکی NSKTOOL مواردی وجود دارد که در ذیل به چند مورد اشاره می‌شود:

۱. ایجاد یک زبان مدل‌سازی دانش رویه‌ای سطح بالا: در حال حاضر از زبان‌های برنامه‌سازی برای مدل‌سازی دانش رویه‌ای استفاده

می‌شود. این امر سبب اعمال محدودیت شدید بر انسان خبره می‌شود. از این رو فرد دیگری که احاطه کامل بر زبان‌های برنامه سازی دارد، باید دانش رویه‌ای را جداگانه فرموله نماید.

۲. تکمیل واحد نگهداری: این کار باید به‌نحوی انجام گیرد که دانش‌های جدیدی که به سیستم اضافه می‌شود، به‌طورکامل کنترل شده و از ایجاد تناقض در پایگاه دانش جلوگیری گردد. این عمل در تحقیق حاضر در حد مناسب به انجام رسیده ولی تا حصول یک سیستم کامل با امکانات مطلوب در این زمینه، هنوز جای کار زیادی وجود دارد. به‌عنوان مثال می‌توان امکان استفاده از تجربیات را به این قسمت اضافه نمود.

۳. امکان نمایش دانش غیر قطعی: NSKMAL در حال حاضر فقط توانایی نمایش دانش قطعی را داراست در حالی که قسمت عمده‌ای از دانش محیط از نوع غیرقطعی می‌باشد.

۹. مراجع

۱- م. ر. حسنی آهنگر، م.ع. جوادزاده، "سیستم خبره تحلیلگر نتایج آزمون تونل باد"، اولین کنفرانس بین‌المللی هوا- فضای ایران. دانشگاه صنعتی شریف، ۱۳۷۹.

۲- م. ر. حسنی آهنگر، "طراحی سیستم خبره تحلیل‌گر نتایج آزمون‌های تونل باد و تولید پایگاه دانش آن"، پایان نامه کارشناسی ارشد. دانشگاه امام حسین علیه‌السلام. سال ۱۳۷۹.

۳- م. ع. جوادزاده، "طراحی یک زبان مدل‌سازی جهت فرموله کردن دانش آیرودینامیک تجربی و پیاده‌سازی مفسر آن"، پایان نامه کارشناسی ارشد. دانشگاه آزاد اسلامی- واحد اراک. سال ۱۳۸۴.

۴- مرکز تحقیقات آیرودینامیک قدر ، "طراحی تفصیلی سیستم خبره WTTAES و ساخت پوسته مورد نیاز تولید آن" گزارش فنی مرکز قدر.۱۳۷۹.

۵- م. ع. جوادزاده، م. ر. حسنی آهنگر، م. ر. کنگاوری، "به‌کارگیری تکنیک‌های هوش مصنوعی در استخراج دانش از منابع اطلاعاتی"، دومین همایش روش‌های تحقیق در علوم و فنون مهندسی، اردیبهشت ۱۳۸۱.

[6] S. J. Templeton, K. Levitt. “A Requires/Provides Model for Computer Attacks”; ACM Press, 2000.

[7] M. Georges, “Knowledge Engineering Trends In Europe Current Developments in Knowledge Aequisitions EKAW 92”; springer – Verlay Germany 1992.

[[8] R. W. Baldwin. “Rule Based Analysis of Computer Security”; MIT, 1987.

[9] D. Farmer, E. H. Spafford. “The COPS Security Checker System”; Purdue, 1994.

[10] D. Zerkle, K. Levitt, NetKuang, “A Multi-Host Configuration Vulnerability Checker”, California, 1996.

[11] R. W. Ritchey, P. Ammann. “Using Model Checking to Analyze Network Vulnerabilities”; IEEE Symposium on Security and Privacy, 2000.

[12] X. Ou, S. Govindavajhala, A.W. Appel. “MulVAL: A Logic-based Network Security Analyzer”; Proceedings of the 14th USENIX Security Symposium, 2005.

[13] R. P. Lippmann, K. W. Ingols. “An Annotated Review of Past Papers on Attack Graphs”; MIT 2005.

[14] X. Ou. “A logic-programming approach to network security analysis”; Princeton University, 2005.

[15] S. Govindavajhala. “A Formal Approach to Practical Network Security Management”; Princeton University, 2006.

[16] Gamal, B. Hasan, etc., “A Security Analysis Framework Powered by an Expert System”; s.l. : International Journal of Computer Science and Security, Issue 6, Vol. 4, 2009.

[17] G. Tsudik, R. Summers., “AudES - an Expert System for Security Auditing”; 2000.

[18] T .N. Kim, A. Hovav., “An Expert System for the Evaluation of Information Security Programs:A Helping Hand for SMEs”; 2005.

[19] J. Durhin “Expert System Design And Development", Macmillan publishing USA, 1994.

[20] P. Linz, “An Introduction to Formal Languages and Automata”; Jones and Bartlett Publishers.1997.