پشتیبانی از امنیت در فازهای توسعه نرم افزار برای متدولوژی مبتنی بر عاملِ Prometheus

محمد حسین داورپور[1*]، محمد احمدی‌نیا[2*]

1- مربی، گروه مهندسی کامپیوتر، واحد سمنان، **دانشگاه آزاد اسلامی**، سمنان، ایران
2- مربی، گروه مهندسی کامپیوتر، واحد کرمان، **دانشگاه آزاد اسلامی**، کرمان، ایران
* پست الکترونیک: davarpour@semnaniau.ac.ir، ahmadinia@iauk.ac.ir

چکیده

اگر چه امنیت در توسعه سیستمهای پیچیده نرم افزاری و بخصوص سیستمهای چند عامله موضوع بسیار مهمی به شمار می رود ولی در زمینه افزودن امنیت به متدولوژی های مهندسی نرم افزار عاملگرا تا کنون کار چندانی صورت نگرفته است. این مقاله به متدولوژی Prometheus تقابلیتی را می افزاید تا بتواند موضوع امنیت را در فازهای مختلف توسعه نرم افزارهای عاملگرا مد نظر قرار دهد. بدین منظور ابتدا مفاهیم امنیتی مورد نیاز تعریف و تشریح می گردد و سپس نحوه مجتمع کردن و ادغام مفاهیم امنیتی در فازهای مختلف متدولوژی Prometheus توضیح داده می شود. جهت نشان دادن نحوه عملکرد این روش از سیستم اطلاعات مراقبتی و پزشکی به عنوان مطالعه موردی استفاده شده است.

کلیدواژگان

امنیت، مهندسی نرم افزار عامل‌گرا، متدولوژی Prometheus ، سیستم‌های چند عامله

Supporting Security in Software Development Phases for Prometheus Methodology

Mohammad Hossein Davarpour [1*], Mohammad Ahmadinia [2*]

1- Department of Computer Engineering, Semnan Branch, Islamic Azad University, Semnan, Iran.
2- Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran.
* davarpour@semnan.aiu.ac.ir, ahmadinia@iauk.ac.ir

Abstract

While security is a key feature in development of complex softwares and multi-agent systems, there are few works around this subject, especially in adding security to agent-oriented software methodologies. This paper adds an extention to Prometheus methodology so it can cover security concerns in different phases of agent-oriented software development. To this end, we define the required security concepts and then integrate them in different phases of Prometheus methodology. The functionality of this extention is explained using a HMIS as a case study.

Keywords

Multi-Agent Systems, Security, Prometheus Methodology, Agent-Oriented Software Enginnering

1- مقدمه

کمیته تحقیق agent توسعه و ایجاد یک متدولوژی کامل جهت تحلیل و طراحی سیستم های چند عامله را یک نیاز مهم تلقی کرده است. چنین متدولوژی باید در تمام فازهای توسعه یک سیستم به توسعه دهنده کمک کند و مهمتر از آن بتواند خواصی که در سیستم های چند عامله اهمیت زیادی دارند مثل قابلیت انعطاف، حل مسائل به صورت خودکار و تعاملات زیاد بین عاملهای جداگانه را مدل کند. امنیت در توسعه یک سیستم کامپیوتری پیچیده یک موضوع مهم می باشد و بر طبق [1] "امنیت مربوط به تمام فازهای توسعه نرم افزار می شود از تحلیل نیازمندیها تا طراحی، پیاده سازی، آزمایش و استقرار". به نظر ما اگر بخواهیم مهندسی نرم افزار عامل گرا (AOSE) را به عنوان یک پایه برای توسعه سیستم های کامپیوتری مبتنی بر عامل قرار دهیم باید مقوله امنیت را در مراحل مختلف توسعه مهندسی نرم افزار عامل گرا مدنظر قرار دهیم.

تاکنون برای اضافه کردن توافقات امنیتی به فازهای تحلیل و طراحی یک سیستم مبتنی بر عامل خیلی کم کار شده است. روش عمومی که جهت تضمین امنیت سیستم استفاده میگردد، شناسایی کردن نیازمندیهای امنیتی پس از تعریف سیستم می باشد که این روش باعث ایجاد سیستم های کامپیوتری ناقص با آسیب پذیریهای امنیتی زیاد شده است. از نقطه نظر الگوی امینتی سنتی باید امکان حذف چنین مشکلی با استفاده گسترده تر از روشهای فرمال و مهندسی نرم افزار بهتر وجود داشته باشد. در [2] یک توسعه برای متدولوژی Tropos جهت افزودن امنیت به مراحل اولیه آن یعنی مراحل تعیین نیازمندیهای اولیه و نهایی انجام شده است و همچنین در [9] سعی شده است که توسعه ای برای متدولوژی Tropos جهت افزودن امنیت به تمام فازهای Tropos ایجاد گردد. ما در این مقاله قصد داریم که متدولوژی Prometheus که یک متدولوژی مهندسی نرم افزار عامل گراست را جهت مدل کردن امنیت در فازهای مختلف توسعه دهیم و بدین منظور مفاهیم امنیتی را به مفاهیم موجود اضافه می کنیم مفهوم نمودار امنیت و محدودیتها را شرح میدهیم و دلیل بکار بردن این مفاهیم و اینکه آنها چگونه می توانند به تحلیل امنیت سیستم کمک کنند را تشریح می نماییم.

بخشهای مقاله به صورت زیر سازماندهی شده اند : بخش2 امنیت در مهندسی نرم افزار را معرفی می کند. در بخش3 مروری بر متدولوژی Prometheus انجام می گیرد. در بخش4 یک سیستم اطلاعات مراقبتی و

پزشکی به عنوان نمونه مورد مطالعه معرفی می گردد تا بتوان توسعه های پیشنهاد شده را در غالب این مثال نشان داد. در بخش 5 ما مفاهیمی که در مدل ارائه شده جهت اضافه کردن امنیت بکار می بریم را تشریح می کنیم و در بخش6 توضیح میدهیم که چگونه روش ما می تواند به فازهای مختلف متدولوژی Prometheus اضافه گردد و بالاخره در بخش 7 نتیجه گیری و کارهای آینده ارائه خواهد شد.

2- امنیت در مهندسی نرم افزار

امنیت معمولاً در اصلاح به وجود خواصی از قبیل قابلیت اعتماد، احراز هویت، جامعیت، کنترل دسترسی، انکار نکردن و قابلیت دسترسی و توانایی غلبه بر تهدیدات ممکن اطلاق میگردد. نیازمندیهای امنیتی سیستم بعد از مطالعه سیاست امنیتی سازمان بدست می آیند.

در حال حاضر، تعریف نیازمندیهای امنیتی معمولاً بعد از طراحی سیستم ملاحظه می شوند این معمولاً بدین معناست که مکانیزم اجرای امنیت باید درون یک طراحی از قبل وجود داشته اضافه شود و بنابراین مشکلات طراحی به معمولاً به آسیب پذیری مالی نرم افزار منترل می گردد. تطبیق دادن امنیت در سراسر فرایند توسعه نرم افزار یک راه حل برای کم کردن این مشکل ارائه می دهد و مهندسی نرم افزار به نیازمندیهای امنیتی همانند نیازمندیهای کارآیی و قابلیت اطمینان به عنوان نیازمندیهای غیر وظیفه مندی می نگرد. نیازمندیهای غیر وظیفه مندی خواص کیفیتی را تعریف می کنند آنها همچنین محدودیت هایی که سیستم با آنها مواجه است را نیز ارائه می دهند. طراحان نرم افزار پیش از این به اهمیت مجتمع کردن نیازمندیهای غیر وظیفه¬مندی از قبیل کارآیی و قابلیت اطمینان با فرایند طراحی نرم افزار پی برده اند هرچند هنوز برای نیازمندیهای امنیتی فکری نشده است.

حداقل دو دلیل برای کمبود پشتیبانی از مهندسی امنیت وجود دارد. اولاً، تحلیل کردن و مدل کردن نیازمندیهای امنیتی مشکل است و ثانیاً توسعه دهندگان تخصص کافی جهت توسعه نرم افزار امن ندارند. گذشته از این سیاستهای امنیتی عموماً بر حسب مدلهای امنیتی که با مدلهای مهندس نرم افزار عمومی مجتمع نشده اند مشخص شده اند. در تکنولوژی جدید خصوصیات امنیتی که در فرایند مهندسی نیازمندیها وجود دارند بیشتر به دلایل کیفیتی پشتیبانی می شوند تا دلایل قانونی (فرمال) وجود دلایل فرمال تایید یک پروتکل که از قبل مشخص شده را پشتیبانی می کند در حالی که جهت گیری های کیفیتی یک روش فرایندگرا ارائه میدهند که نیازمندیهای غیر وظیفه¬مندی را به عنوان اهدافی که بالقوه هماهنگ و یا ناهماهنگند و در طول توسعه سیستمهای نرم افزاری استفاده می شوند در نظر می گیرند.

ما عقیده داریم که امنیت باید در طول تمام فرایند توسعه مورد توجه قرار گیرد و باید با تشخیص نیازمندیها تعریف شود. اگر امنیت فقط در مراحل خاص از فرایند توسعه مورد توجه قرار گیرد امکان دارد که نیازهای امنیتی با نیازمندیهای وظیفه مندی سیستم برخورد پیدا کنند و تداخل پیش آید.

الگوی عمل گرا یک روش عملی برای اضافه کردن امنیت به مهندسی نرم افزار ارائه میدهد. همانطور که در [3] اشاره شده است عاملها به نیابت از محاورات افراد و شرکتهایی که بر طبق یک بافت سازمانی انجام وظیفه می کنند عمل می کنند. اضافه کردن امنیت به بافت احتیاج دارد که بقیه زیر سیستم ها هم به نیازمندیهای امنیتی که در سیاست امنیتی مشخص شده، در زمان شناسایی کردن اهداف و محاوراتشان که باعث انتشار نیازمندیهای امنیتی به بقیه زیر سیستمها می گردد توجه نمایند.

بهر حال در حال حاضر هیچ متدولوژی نرم افزاری عامل گرایی به نیازمندیهای امنیتی به عنوان یک قسمت از همه فرایند توسعه نرم افزار توجه نمی کند. ما در بخش بعد متدولوژی Prometheus را توسعه می دهیم بدین منظور که بتوان امنیت را در فازهای مختلف آن مورد توجه قرار داد.

3- مروری برمتدولوژی Prometheus

متدولوژی Prometheus یک متدولوژی مهندسی نرم افزار عامل گرا است که یک فرایند مفصل و کامل (Start-to-end) جهت شناسایی، طراحی و پیاده سازی سیستم های مبتنی بر عاملهای هوشمند می باشد[8]. هدف ایجاد و توسعه Prometheus این بوده است که یک متدولوژی عامل گرا که قابل آموزش به دست اندرکاران صنعت و دانشجویانی که از مفهوم agent پیش زمینه کافی ندارند باشد ارائه شود. این متدولوژی شامل سه فاز جهت توسعه نرم افزار می باشد : مشخصات سیستم، طراحی معماری و طراحی جزئیات. که در ادامه جزئیات و محصولات هر فاز را توضیح می‌دهیم.

3-1- مشخصات سیستم

این فاز با یک ایده کلی از سیستم شروع می گردد که می تواند توصیف کلی سیستم در چند پاراگراف باشد سپس نیازمندیهای سیستم بر حسب موارد زیر مشخص می شود.

1- اهداف سیستم
2- سناریوهای موارد کاربری
3- قابلیت‌های سیستم
4- رابط سیستم با محیط که برحسب اعمال و دریافت‌ها تعریف می گردد.

فرایند درآوردن اهداف سیستم با شناسایی یک مجموعه اولیه از اهداف از توصیف سطح بالای سیستم آغاز می گردد. برای مثال از توصیفی مثل "ما می خواهیم یک سیستم زمانبندی گروهی ایجاد کنیم که در آن کاربران بتوانند جلسات با یکدیگر را ثبت کنند ...." ما می توانیم اهدافی مثل "زمانبندی جلسه"، "زمانبدی دوباره جلسه" و "مدیریت تقویم های کاربران" را استخراج کنیم. با سؤال کردن اینکه هر هدف چگونه محقق می شود، می توان زیر هدفها را مشخص کرد و مجموعه اهداف را توسعه داد [4]. مثلاً می توان پرسید جلسات چطور زمانبندی می گردند و اگر جواب دهیم " باید یک زمان آزاد مشترک پیدا شود" یک زیر هدف جدید برای هدف سطح بالای "زمانبندی یک جلسه" پیدا می شود. اهداف با استفاده از نمودار هدف نشان داده می شوند که در این نمودار هدفها به کمک لوزی نمایش داده می شوند و ارتباط بین یک هدف و زیر هدف به کمک یک بردار از هدف به زیرهدف نمایش داده می شود.

با گروهبندی کردن هدفها ما قابلیتهای سیستم را مشخص می کنیم یک قابلیت شامل اهداف، دریافتهای مربوطه، اعمالی که انجام میدهد و داده هایی است که ازآن استفاده می کند. قابلیتهای سیستم در فازهای بعدی می توانند به تواناییهای عاملها تبدیل گردند. قابلیتها با استفاده از توصیفگرها شرح داده می شوند. که یک سری فرمهای متنی هستند که شامل اطلاعات لازم در مورد قابلیت می باشند. یکی دیگر از محصولات این فاز سناریوهای موارد کاربری است. سناریوی موارد کاربری یک توصیف کامل از ترتیب رویدادهایی است که برای رسیدن به یک هدف خاص و یا در پاسخ به یک رویداد خاص

انجام می گیرد و بالاخره در این فاز محیطی که سیستم مورد نظر باید در آن کار کند تعریف می گردد که به صورت توصیف دریافتهای سیستم و اعمالی که سیستم باید انجام دهد شکل می گیرد.

شکل 1. علائم مورد استفاده در نمودار system overview [8]

شکل 2. علائم مورد استفاده در نمودار agent overview [8]

3-2- طراحی معماری

در این فاز بر روی موارد زیر تکیه می گردد :

شناسایی عاملهای سیستم: که عاملها به وسیله گروه بندی کردن قابلیتهای سیستم براساس روابط بین آنها شناسایی می گردند که به وسیله نمودارهای data coupling و agent acquaintance نمایش داده می شوند زمانی که یک گروه بندی انجام می گیرد توضیحات مربوط به عامل در توصیفگرهای عامل آورده می شوند.

نمایش تعاملات بین عاملها با استفاده از نمودار تعامل و پروتکلهای تعامل که نمودارهای تعامل از سناریوهای موارد کاربری مشتق می شوند و با تجدیدنظر و عمومیت بخشیدن به این نمودارها پروتکلهای تعامل ایجاد می شوند.

طراحی ساختار کلی سیستم:ساختار کلی سیستم توسط نمودار system Overview تعریف می شود و مستند می گردد. این نمودار عاملهای موجود در سیستم، مرزهای سیستم و رابطهای سیستم (برحسب اعمال و دریافتها) را نمایش میدهد. شناسایی عاملهای موجود در سیستم مهمترین عملی است که در این فاز انجام می گیرد و مهمترین محصول این فاز نمودار system Overview می باشد.

بعضی از علائم و نشانه هایی که در این نمودار مورد استفاده قرارمی-گیرند در شکل1 نمایش داده شده است.

3-3- طراحی جزئیات

در این فاز اعمال زیر انجام می گیرد:

1- ساختار داخلی یک عامل بر حسب تواناییهایش توسعه داده می شود. که این عمل به وسیله نمودار agent Overview و توصیفگرهای توانایی انجام می گردد.

2- ایجاد نمودارهای فرایند با توجه به پروتکلهای تعامل تولید شده در فاز قبل.توسعه جزئیات تواناییها بر حسب تواناییهای دیگر و نیز با استفاده از رویدادها طرحها و داده ها به وسیله نمودار Capability Overview انجام می گیرد. تواناییها یک مکانیزم ساختاری وابسته به واحدها می باشند. یک توانایی می تواند شامل نقشه ها، داده ها و رویدادها باشد. همچنین با ساختار سلسله مراتبی می تواند شامل تواناییهای دیگر هم باشد.

3- در این فاز ساختار هر عامل به وسیله نمودار agent Overview نمایش داده می شود. این نمودار شبیه نمودار system Overview است با این تفاوت که در این نمودار نود agent وجود ندارد و نودهای Capability که در نمودار system Overview وجود ندارند در این نمودار می توانند لحاظ شوند. انواع نودهایی که در این نمودار میتوانند نمایش داده شوند در شکل2 نشان داده شده اند.

4- نمونه مورد مطالعه : سیستم اطلاعات مراقبتی و پزشکی

برای اینکه نیاز به توسعه Prometheus و نحوه توسعه دادن آن را نشان دهیم اجازه دهید که یک سیستم شبیه به سیستم eSAP که ابتدا به وسیله Moratiadiseta [5] ارائه شده است را مورد توجه قرار دهیم در این سیستم که سیستم اطلاعات مراقبتی و پزشکی است 4 عامل دخالت دارند.

Patient : بیمار شخصی است که می خواهد سلامتی اش را بازدید و تحت مراقبت پزشکی قرار گیرد.

Professional : پزشک متخصص که اطلاعات بیمار را دریافت می کند و برای سلامتی بیمار اقدام می نماید.

Benefit Agency : یک نمایندگی بیمه که بیمار را از لحاظ مالی پشتیبانی میکند.

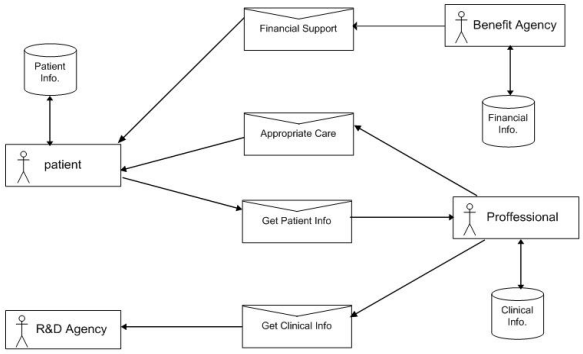
B & D Agency : یک مؤسسه توسعه و تحقیق که برای انجام تحقیقاتی به دریافت اطلاعات در مورد بیماران نیاز دارد.

برای تحلیل این سیستم به کمک Prometheus ما جزئیات را درنظر نگرفته ایم و تمام نمودارهای مورد نیاز در فرایند Prometheus را رسم نمی کنیم چون هدف ما فقط اضافه کردن فرایند امنیت به این متدولوژی است و امنیت در تعاملات عاملها با یکدیگر مطرح میشود که این تعاملات در نمودار system Overview باید نمایش داده شوند. اگر ما نمودار system Overview مربوطه را که ساختار کلی سیستم را نمایش می دهد رسم کنیم نمودار به صورت شکل 3 خواهد بود.

اگرچه ارتباطات بین عاملها به صورت واضح در این نمودار نشان داده شده است، بعضی از محدودیتهای ممکن (در اینجا مربوط به امنیت) که امکان دارد به بعضی عاملها تحمیل شوند نشان داده نشده اند. به عنوان مثال Patient با Benefit Agency جهت دریافت پشتیبانی مالی ارتباط دارد اما ممکن است Patient بخواهد یک محدودیت برای Benefit Agency تعریف کند که مثلاً اطلاعات مالی را به صورت خصوصی و پنهانی نگهداری کند. به عنوان مثالی دیگر B & D Agency با Professional ارتباط دارد تا اطلاعات بالینی را جهت انجام تحقیقات بدست آورد اما ممکن است که Professional از طرف Patient محدود شده باشد که اطلاعات بالینی را فقط به صورت بدون نام در اختیار بگذارد. به علاوه امکان دارد Patient یک محدودیت به Professional تحمیل کند که فقط در صورتی که رضایت Patient کسب شد اطلاعات را به اشتراک بگذارد.

با در نظر گرفتن این محدودیتها می توان ارتباطات موجود بین عاملها را بهتر پالایش کرد و اهداف امنیتی را هم در سیستم لحاظ نمود.

حال در اینجا امکان دارد ابهامی پیش آید که میتوان بدون اعمال محدودیت در ارتباطات بین عاملها و با استفاده از مفاهیم موجود در بعضی موارد ارتباط امن داشت مثلاً در مثال بالا پیغام "رضایت بیمار" از طرف patient به Professional بدون اعمال هیچ محدودیتی به سیستم ارسال شود. این امکان وجود دارد اما این روش یک فرایند کاملاً موردی ارائه میدهد که فقط به توانایی و تجربه طراح بستگی دارد و در بعضی موارد هم امکان پذیر نیست. استفاده از مفهوم محدودیتها به ایجاد یک روش سیستماتیک در پالایش ارتباطات موجود و ایجاد ارتباطات امن مربوط به امنیت سیستم کمک می کند و همچنین دلایلی که این اهداف باید به سیستم لحاظ شوند و اینکه به کدام عامل مربوطند را مشخص می نماید.



شکل 3. نمودار system overview [8]

5- توسعه Prometheus جهت اضافه کردن امنیت به آن

یک در حال حاضر فرایند اضافه کردن نیازمندیهای امنیتی به تمام مراحل توسعه نرم افزار کاملاً موردی است. وجود یک فرایند سیستماتیک که توسعه دهنده را در ت وجه کردن به نیازمندیهای امنیتی (همانند دیگر نیازمندیهای غیر وظیفه مندی) در طول تمام فازهای توسعه راهنمایی کرد و نمادها و مفاهیم یکسانی را در طول فازهای توسعه استفاده نماید، امری ضروری به نظر می رسد.

در چند سال اخیر کار در زمینه مهندسی نرم افزار عامل گرا بر روی تأمین یک متدولوژی کامل که در همه فازهای توسعه یک سیستم مبتنی بر عامل کمک کند و مهمتر از آن برای مدلسازی خواصی مانند انعطاف پذیری، حل مسائل به صورت خودکار و مقاملات بین عاملهای جداگانه متمرکز شده است و مسئله مهمی مانند امنیت اساساً نادیده گرفته شده است و در زمینه پشتیبانی متدولوژی های مهندسی نرم افزار و عامل گرا از امنیت در طول مراحل توسعه کار کمی انجام شده است.

به عبارت دیگر اگرچه کارهایی برای درآوردن نیازهای غیر وظیفه (شامل امنیت) انجام شده است و به جایگزین های طراحی مختلفی توجه شده ، ولی هیچکدام از این روشها با الگوی عامل توسعه داده نشده اند.

در هر صورت نیازهای غیر وظیفه مندی ممکن است مربوط به خواص کیفی سیستم باشند و یا محدودیتهایی را بر سیستم اعمال می کنند. خواص کیفی مشخصاتی از سیستم هستند که برای سهامداران مهمند ولی محدودیتها قوانین و شرایطی هستند که به سیستم تحمیل می شوند. ما در این بخش سعی می کنیم که مفاهیمی که باید به متدولوژی Prometheus اضافه گردند تا بتوان در تمامی فازهای توسعه نرم افزار امنیت را مد نظر قرار داد تشریح می کنیم این مفاهیم عبارتند از : محدودیت (Constraint) ، محدودیت امنیتی (secure Constraint) ، موجودیت امن (Secure Entity) ، ارتباط امن (secure Communication) و نمودار امنیت (Security Diagram).

5-1- محدودیت‌ها (Constraint)

محدودیتها قیودی هستند که به کارهای ویژه ای اجازه انجام شدن نمی دهند و یا اینکه از برآورده شدن اهداف ویژه ای جلوگیری می کنند. بنابراین محدودیتها می توانند یک مجموعه شرط ارائه نمایند. قوانین و قیودی به سیستم تحمیل کنند و سیستم باید به گونه ای عمل کند که این محدودیتها نقض نشوند.

غالباً در متدولوژی ها محدودیتها به ویژگیهای مفاهیم موجود اضافه می شوند و بر حسب توصیفات متنی غیر رسمی بیان شده اند ولی این روش باعث تعریف مبهم یک محدودیت و نقش آن در توسعه سیستم می شود و در ضمن باعث خطاهایی در مراحل توسعه اولیه می گردد که این خطاها به مراحل بعدی متنشر می شوند و باعث مشکلات جدی و زیادی در زمان ترمیم می گردد.

به علاوه اضافه کردن محدودیتها به مفاهیم موجود با ایجاد مشکلاتی در زمان اصلاح سیستم می گردد. بنابراین ما معتقدیم که محدودیتها باید به عنوان یک مفهوم جداگانه در طول همه فرایند توسعه معرفی گردند.

تعریف محدودیتها به عنوان یک مفهوم جداگانه به معنای جدا کردن آنها از بقیه سیستم نیست بلکه محدودیتها به طور کامل مربوط به قسمتهایی از سیستم هستند که توسط آن محدودیت محدود می شوند.

5-2- محدودیتهای امنیتی (Security Constraints)

محدودیتها می تواند براساس نیازمندیهای غیر وظیفه مندی که مربوط به آن هستند گروه بندی شوند. بنابراین محدودیتها می توانند به گروههای قابلیت اطمینان، کارآیی، امنیت و ... تقسیم بندی شوند. در این مقاله ما قصد داریم که محدودیتهای امنیتی را به سیستم اعمال کنیم تا به پیش برد امنیت در سیستم کمک کنند. ما محدودیت امنیتی را به عنوان محدودیتی که مربوط به امنیت سیستم می شود تعریف می کنیم.

محدودیتها به صورت مثبت و یا به صورت منفی در امنیت سیستم تأثیرگذار می گردند. یک مثال از محدودیت امنیتی مثبت می تواند این باشد که فقط به پرسنل اجازه دسترسی به پرونده بیمار داده شود. ولی ارسال پرونده بیمار به صورت رمز نشده یک محدودیت امنیتی منفی تلقی می گردد. به علاوه محدودیت امنیتی ممکن است تأثیر مثبت یا منفی در نیازمندیهای دیگر سیستم داشته باشد.

محدودیتهای امنیتی می تواند از روی نمودار امنیت به سیستم اعمال شود. با اعمال محدودیتهای امنیتی به قسمتهای مختلف سیستم ما می توانیم تداخلات و برخوردهای احتمالی بین امنیت و دیگر نیازمندیهای سیستم را شناسایی کنیم، محدودیتهایی که می تواند سیستم را در خطر قرار دهند بشناسیم و روشهای ممکن جهت یک طراحی که امنیت و مهندسی سیستم را برای توسعه یک سیستم امن تر مجتمع کند پیشنهاد دهیم.

لازم به ذکر است که ما به محدودیتهای امنیتی در یک سطح بالا از تجرید توجه می کنیم. بدین معنی که محدودیت امنیتی شامل مشخص کردن پروتکلهای امنیتی ویژه ای نمی گردد تا توسعه سیستم به یک راه حل امنیتی خاص محدود نشود. یعنی ما پیاده سازی سیستم را به پروتکلهای امنیتی خاص و به تبع آن زبان برنامه نویسی خاص محدود نمی کنیم.

یک محدودیت امنیتی در نمودار بصورت شکل4 نمایش داده می شود.

5-3- موجودیت‌های امن (Secure Entities)

اصلاح موجودیتهای امن به اهداف، اعمال و داده های تواناییها امن سیستم اطلاق می گردد یک موجودیت امن به یک عامل در سیستم در دستیابی به محدودیت امنیتی کمک می کند به عنوان مثال اگر عامل Professional محدودیت امنیتی داشته باشد که "فقط اگر رضایت مشتری کسب شد اطلاعات را به اشتراک بگذارد." می توان یک هدف امن برای سیستم تعریف کرد : "بدست آوردن رضایت Patient " .

شکل 4. نحوه نمایش محدودیت امنیتی [8]

شکل 5. صورت‌های نمایش موجودیت امن [8]

یک هدف امنیتی مشخص نمی کند که محدودیت امنیتی چگونه اعمال شود. چون روشهای متفاوت می توانند بکار روند که یک هدف برآورده شود. ولی در مورد اعمال امن این امکان وجود دارد چون action کار انجام شده و یا روش انجام کار را مشخص می نماید بنابراین یک عمل امن (Secure Action) ، انجام کاری در جهت رسیدن به یک هدف امن را مشخص می نماید. برای مثال برای رسیدن به هدف امن "چک کردن احراز هویت"، ما می توانیم اعمال امنی از قبیل "چک کردن کلمه عبور" و یا "چک کردن امضای دیجیتال" داشته باشیم. یک منبع داده مربوط به یک موجودیت امن، یا یک محدودیت امن یک داده امن (Secure Data) نامیده می شود.

ما از موجودیتهای امن در نمودارهای system Overview و agent Overview استفاده می کنیم. توانایی امن، قابلیتی است که به یک عامل اضافه می شود تا بتواند یک هدف امن را برآورده سازد. موجودیتهای امن به صورت یک S درون پرانتز و به صورت نمایش داده شده در شکل5 نشان داده می شوند.

5-4- ارتباطات امن (Secure Communications)

در یک ارتباط امن یک محدودیت امنیتی از طرف عامل گیرنده یا عامل فرستنده پیشنهاد میگردد. بدین منظور که ارتباط به صورت موفقیت آمیز و به درستی انجام شود. هم گیرنده پیام و هم فرستنده باید بر روی این محدودیت یا محدودیتها توافق داشته باشند تا این ارتباط امن معتبر باشد. این موضوع در طرف گیرنده بدین معنا است که گیرنده از فرستنده انتظار دارد که محدودیتهای امنیتی را برآورده سازد و در طرف فرستنده بدین معنا است که فرستنده تلاش خواهد کرد که پیغام را با رعایت محدودیتهای امنیتی به گیرنده بفرستد سه نوع مختلف از ارتباط امن وجود دارد :

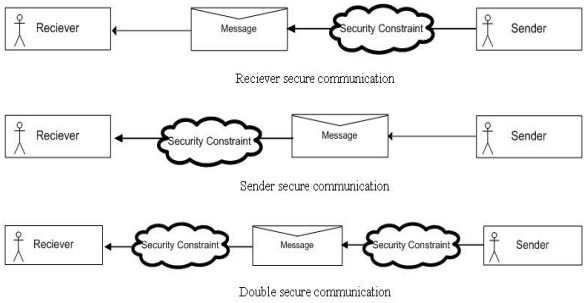
1- ارتباط امن گیرنده: در این نوع ارتباط گیرنده محدودیتهای امنیتی جهت ارتباط را معرفی می کند. در این حالت فرستنده باید محدودیتهای امنیتی تعریف شده از طرف گیرنده را برآورده سازد. در غیر این صورت امنیت ارتباط به خطر خواهد افتاد. این نوع از ارتباط امن با یک محدودیت در طرف فرستنده در شکل 6 نمایش داده شده است.

2- ارتباط امن فرستنده : در این نوع ارتباط امن، فرستنده محدودیتهای امنیتی ارتباط را معرفی می کند و گیرنده باید این محدودیتهای امنیتی را جهت رسیدن به ارتباط امن رعایت نماید. این نوع از ارتباط امن به صورت گرافیکی با محدودیت در طرف گیرنده در شکل 6 نمایش داده شده است.

3- ارتباط امن دوگانه : در این شکل از ارتباط، محدودیتهای امنیتی را هم فرستنده و هم گیرنده تعریف می کنند و برای رسیدن به ارتباط امن هر دو باید محدودیتهای امنیتی تعریف شده را رعایت نمایند. این نوع از ارتباط امن با محدودیتهایی در هر دو طرف فرستنده و گیرنده در شکل 6 نشان داده شده است.

5-5- نمودارامنیت (Security Diagram)

نمودار امنیتی سیستم می تواند پس از تحلیل نیازمندیهای امنیتی سیستم و محیط ساخته شود و شبیه به کاتالوگ امنیتی است که ابتدا به وسیله Yu [6] معرفی شد.



شکل 6. مثالی از ارتباط امن [8]

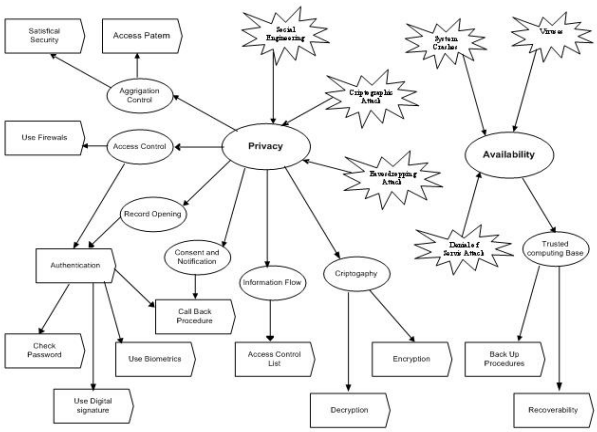
فرآیند تحلیل نیازمندیهای امنیتی سیستم و محیط به صورت یک امر ثابت و واحد نیست و بستگی به مهندسی تحلیلگر دارد. این فرایند معمولاً شامل شناسایی نیازهای امنیتی سیستم و مشکلات مربوط به امنیت سیم (از قبیل تهدیدات و آسیب پذیریهای احتمالی) و راه حلهای ممکن برای حل مشکلات امنیتی می باشد (این راه حلها می تواند بر حسب سیاست امنیتی که سازمان مشخص کرده باشند).

نمودار امنیتی دو قابلیت استفاده دارد اولاً به طراح کمک می کند تا محدودیتهای ممکن که باید در سیستم معرفی شوند را شناسایی نماید ثانیاً در مراحل بعدی توسعه، مکانیزمهای امنیتی که بتوانند محدودیتهای امنیتی را به سیستم اعمال نمایند مشخص می نماید. علاوه بر این نمودار امنیتی فواید عمومی زیر را نیز داراست. یک چارچوب برای نیازها، تهدیدات و راه حلهای ممکن امنیتی با استفاده از مفاهیم شناخته شد و برای مهندس نرم افزار فراهم می آورد.

اکثر سیستم های تحت توسعه شباهت زیادی به سیستمهایی که از قبل وجود داشته¬اند دارند. بنابراین نمودار امنیت می تواند به عنوان یک مرجع در نظر گرفته شود و می تواند براساس نیازهای خاص اصلاح گردد یا توسعه داده شود. تا زمان و زحمت توسعه دهندگان صرفه جویی به عمل آید.

در این مقاله ما قصد داریم که متدولوژی Prometheus را توسعه دهیم بنابراین نمودار در فاز مشخصات سیستم و با توسعه نمودار هدف و با مفاهیم موجود در متدولوژی Prometheus مثل هدف، زیر هدف عمل و دریافت تطبیق می دهیم که باعث می شود مهندس نرم افزار در تمام فرایند توسعه Prometheus از مفاهیم یکسانی استفاده کند.

در فرایند ساخت نمودار امنیت مهندس خصوصیات ایمنی سیستم، اهداف حفاظتی، مکانیزمهای امنیتی و تهدیدات امنیتی سیستم را لحاظ می کند.



شکل 7. نمودار امنیت [8]

نمودار امنیت ارتباط بین خصوصیات امنیتی، تهدیدات، اهداف حفاظتی و مکانیزمهای امنیتی را ارائه میدهد تا اهداف سیستم به درستی برآورده شود و هر خصیصه امنیتی از اهداف حافظتی مختلف تأثیر مثبت می پذیرد و از تهدیدات تأثیر منفی دریافت می کند. تأثیرات مثبت در رسیدن به خصیصه های امنیتی کمک می کنند در حالی که تأثیرات منفی خصیصه های امنیتی را در خطر قرار می دهند. بعلاوه نمودار مکانیزمهای امنیتی ممکن که در رسدن به اهداف حفاظتی تأثیر مثبت یا منفی دارند را نیز نشان می دهد.

خصیصه های امنیتی، خصوصیات مربوط به امنیت که سیستم باید دارای آن باشد را ارائه میدهد و می توان در نمودار امنیت آنها را به صورت اهداف کلی و ریشه در نظر گرفت. مثالهایی از خصیصه های امنیتی عبارتند از خصوصی بودن، ایمنی، جوابگو بودن، در دسترس بودن و جامعیت.

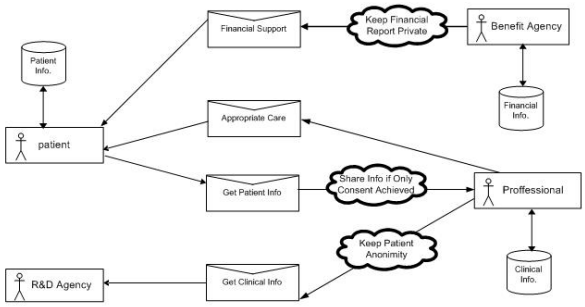
تهدیدها رویدادهایی را ارائه می دهند که به طور بالقوه بتوانند باعث زیان شوند و یا بتوانند خطری برای خصیصه های امنیتی سیستم باشند. ما برای نشان دادن تهدیدها در نمودار امنیت از نماد مربوط به دریافتها (Percept) استفاده می کنیم چون تهدیدها معمولاً رویدادهایی هستند که به سیستم از بیرون تحمیل می شوند.

اهداف حافظت، مجموعه ای از اصول و قوانینی که در رسیدن به خصیصه های امنیتی دخالت می کنند را ارائه می دهند. این اصول راه حلهای ممکن برای مشکلات امنیتی که معمولاً در سیاست امنیتی سازمان دیده می شود را مشخص می نمایند.

ما اهداف حفاظت را با مفهوم هدف ارائه میدهیم این تصمیم از آنجا گرفته شده است که یک هدف یک وضعیت مطلوب سیستم را نشان میدهد و به طور مشابه یک هدف حافظتی هم یک وضعیت ایمنی مطلوب برای سیستم را ارائه می دهد.

مکانیزمهای امنیتی، مکانیزمهای حافظتی ممکن جهت رسیدن به اهداف حافظتی را مشخص مینمایند. به منظور ارائه مکانیزمهای امنیتی ما مفهوم عمل (action) را بکار می بریم. چون یک عمل کاری است که سیستم جهت رسیدن به هدفی انجام میدهد.

یک قسمت از نمودار امنیت سیستم اطلاعات مراقبتی و پزشکی در شکل7 نشان داده شده است. در نمودار امنیت ارائه شده ما دو خصیصه امنیتی مطلوب برای سیستم در نظر گرفته‌ایم : خصوصی بودن و در دسترس بودن. باید توجه داشت که این نمودار به طور کامل و دقیق نمودار امنیت سیستم را در بر نمی گیرد اما به عنوان یک مثال برای فهم بهتر مفاهیم و نمادهای نمودار امنیت استفاده می شود. هم Privacy و هم Availability از تهدیدات مختلف سیستم از قبیل مهندسی اجتماعی، استراق سمع، حملات رمز نگاری، ویروسها، از کار افتادن سیستم و غیره تأثیر منفی می پذیرند. اساساً ما در این مثال اهداف حافظتی را بر اساس سیاست امنیتی که Ross [7] در مورد سیستم اطلاعات پزشکی پیشنهاد کرده در نظر گرفته ایم.



شکل 8. نمودار System Overview با اعمال محدودیت‌های امنیتی [8]

به عنوان مثال یکی از مهمترین اهداف حافظت که به خاصیت Privacy سیستم کمک می کند کنترل دسترسی است. کنترل دسترسی به صورت احراز هویت شخصی که سعی می کند به منابع دسترسی پیدا کند انجام می شود. احراز هویت می تواند به وسیله مکانیزمهای امنیتی مختلفی از قبیل امضای دیجیتالی، زیست سنجی و یا کلمه عبور انجام شود.

6- اضافه کردن امنیت به فازهای مختلف Prometheus

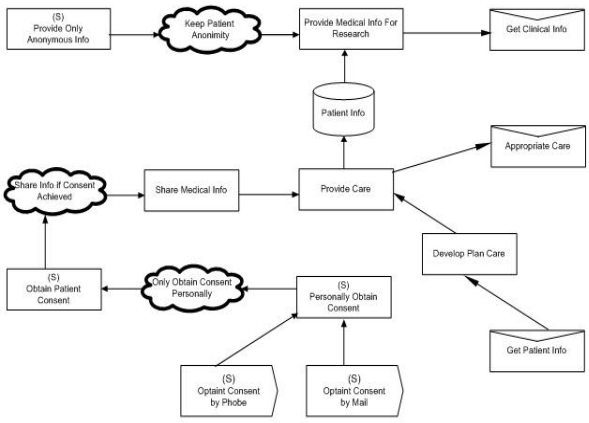
در این بخش فرایند امنیت اضافه شده به مراحل Prometheus شرح داده می شود.

6-1- فاز مشخصات سیستم

در این فاز نیازمندیهای امنیتی که از طرف محیط و یا از طرف سهامداران به سیستم تحمیل می شوند براساس سیاست امنیتی سازمان تحلیل می گردند و با توجه به آن خصیصه های امنیتی مورد نیاز سیستم و سپس اهداف حافظتی و مکانیزمهای برآوردن آنها و همچنین تهدیدات امنیتی که امکان دارد به سیستم وارد گردد شناسایی و در غالب نمودار امنیت نمایش داده می شود در بخش قبلی نحوه ایجاد نموار امنیت و مفاهیم بکار رفته در آن توضیح داده شد و قسمتی از نمودار امنیت مربوط به سیستم اطلاعات مراقبتی و پزشکی که قبلاً تشریح شد رسم گردید.

6-2- فاز مشخصات سیستم

در این فاز ساختار کلی سیستم و عاملهای موجود در آن و ارتباطات بین آنها و دریافتها و اعمال سیستم در غالب نمودار system Overview نمایش داده می شود و از آنجا که موضوع امنیت بیشتر در تعاملات عاملها با یکدیگر مطرح می شود جهت اضافه کردن مقوله امنیت به این فاز باید ارتباطات بین عاملها ارتباط امن باشد و همانگونه که در بخشهای قبلی اشاره شد ارتباط امن ارتباطی است که محدودیتهای امنیتی لازم را لحاظ نماید. بنابراین در این فاز ما محدودیتهای امنیتی را در ارتباطات بین عاملها در نمودار system Overview اعمال می کنیم محدودیتهای امنیتی یا با توجه به نمودار امنیت که در فاز قبل ایجاد شده به سیستم اضافه می شوند و یا از جانب سهامداران سیستم یعنی با توجه به نیازمندیهای سیستم و توصیفگرهای ایجاد شده در فاز قبل. اگر نمودار system Overview شکل3 مربوط به سیستم اطلاعات مراقبتی و پزشکی که محدودیتهای امنیتی در آن لحاظ نشده اند را در نظر بگیرید متوجه کمبودهای امنیتی نمودار می شوید که در بخشهای قبلی توضیح داده شد. ما در اینجا قصد داریم که محدودیتهای امنیتی را به نمودار اضافه کنیم به عنوان مثال Patient با Benefit Agency جهت دریافت پشتیبانی مالی ارتباط دارد اما ممکن است که Patient بخواهید یک محدودیت برای Benefit Agency تعریف کند که مثلاً اطلاعات مالی را به صورت خصوصی نگهدارد. بدین منظور می توانیم در ارتباط بین Benefit Agency و Patient یک محدودیت در سمت Benefit Agency قرار دهیم و یک ارتباط امن داشته باشیم. همچنین می توان محدودیت Professional در به اشتراک گذاشتن اطلاعات بالینی به صورت بدون نام و نیز به اشتراک گذاشتن اطلاعات در صورت کسب رضایت Patient را نیز با اضافه کردن این محدودیتها به نمودار نمایش داد. شکل8 نمودار system Overview مربوط به سیستم اطلاعات مراقبتی و پزشکی پس از اعمال محدودیت‌های امنیتی را نشان می‌دهد.



شکل 9. نمودار Agent Overview با اعمال محدودیتهای امنیتی [8]

3-6- فاز مشخصات سیستم

در این فاز ساختار داخلی عامل توسعه داده می‌شود و تواناییهای عامل، ارتباطات بین تواناییها و نیز دریافتها و اعمال عامل با کمک نمودار agent Overview نشان داده میشود. ما در نمودار agent Overview محدودیتهای امنیتی که وجود دارند را لحاظ می کنیم تواناییهای امن جهت اعمال محدودیتهای امنیتی را به نمودار اضافه می کنیم. در این نمودار می توانیم بین تواناییهای یک عامل ارتباط امن داشته باشیم.

به عنوان یک مثال در شکل 9 نمودار Agent Overview مربوط به عامل Professional مشاهده می شود. در اینجا عامل Professional دو محدودیت امنیتی باید لحاظ نماید اول اینکه اطلاعات را فقط در صورت کسب رضایت به اشتراک بگذارد و دوم اینکه اطلاعات مربوط به بیمار را به صورت بی نام نگهدارد. عامل Professional باید توانایی به اشتراک گذاشتن اطلاعات بالینی را دارا باشد این توانایی به وسیله محدودیت "فقط در صورت کسب رضایت اطلاعات را به اشتراک بگذارد" که به وسیله Patient به Professional تحمیل شده تحت تأثیر قرار می گیرد. بهر حال این توانایی به صورت های مختلفی می تواند به اجرا درآید مثلاً Professional می تواند رضایت بیمار را به صورت شخصی کسب نمایدو یا از طریق یک پرستار اقدام نماید. بنابراین می توان زیر محدودیت "رضایت فقط به صورت شخصی کسب شود" را تعریف کرد و جهت برآورده شدن این محدودیت باید یک توانایی امن دیگر به نمودار اضافه گردد به عنوان "بدست آوردن رضایت به صورت شخصی". که این توانایی به دو صورت قابل انجام است که رضایت بیمار با تلفن و یا با پست الکترونیکی کسب شود. بنابراین می توان دو عمل و یا دو مکانیزم امنیتی "کسب رضایت با کمک پست الکترونیکی" و "کسب رضایت به وسیله تلفن" را به نمودار اضافه کرد. Professional باید توانایی "تأمین اطلاعات پزشکی جهت تحقیق را هم داشته باشد. که محدودیت "نگهداری اطلاعات بیمار به صورت بی نام" این توانایی را محدود می نماید در نتیجه این محدودیت توانایی امن "تأمین اطلاعات فقط به صورت بی نام" تعریف می گردد.

7- نتیجه گیری و کارهای آینده

در این مقاله ما توسعه ای به متدولوژی Prometheus افزودیم تا بتوان مفاهیم امنیتی سیستم را در فازهای توسعه نرم افزار عاملگرا مد نظر قرار داد. هدف ما این بود که فرایندی ارائه نماییم تا به صورت واضح امنبت و نیازمندیهای وظیفه مندی سیستم را در تمام فازهای توسعه نرم افزار مجتمع کنیم . چنین فرایندی باید از مفاهیم و نمادهای یکسانی در طول فازهای توسعه استفاده نماید. همانطور که اشاره شد توسعه در هر سه فاز متدولوژی Prometheus لحاظ شده است بدین صورت که در فاز اول نمودار امنیت مربوط به سیستم ساخته شده و در فاز دوم نمودار System Overview با توجه به محدودیتهای امنیتی که از طرف سهامداران یا سیستم به نرم افزار تحمیل می گردد توسعه داده شد و در فاز سوم Prometheus محدودیتهای امنیتی را به نمودار Agent Overview اضافه کردیم. مفهوم موجودیتهای امن جهت اعمال محدودیتهای امنیتی را نیز توضیح دادیم و در نمودار Agent Overview بکار بردیم.

کار بعدی ما این است که مفاهیم امنیتی را در دیگر محصولات این سه فاز هم مد نظر قرار دهیم به طوری که بتوان یک سیستم با امنیت بالا طراحی نمود. بدین صورت که ما با معرفی و تشریح سناریوهای امن در فاز مشخصات سیستم و تبدیل آن به نمودارهای تعاملی امن در فاز طراحی معماری، روند اجرای مکانیزمهای امنیتی و نیز نحوه و ترتیب افزودن موارد امنیتی به سیستم را مدلسازی کرده و به صورت کامل تشریح می نماییم. ودر فاز طراحی جزئیات از نمودارهای تعاملی امن، نمودار فرایند امنیتی را ایجاد نموده و در ضمن توانایی‌های امن را در قالب نمودار توانایی امن توسعه می دهیم تا بتوان قابلیت‌های امنیتی عامل‌ها در سیستم عاملگرا را به طور کامل مدل نمود.

8- مراجع

[1] P. Devanbu, S. Stubblebine, Software Engineering for Security: a Roadmap, *Proceedings of the conference of The future of Software engineering (ICSE' 00)*, pp. 227-239, 2000
[2] H. Mouratidis, P. Giorgini , A Natural Extension of Tropos Methodology for Modelling Security, University of Trento, Italy, 2003.
[3] N. R. Jennings, M. Wooldridge, Agent-Oriented Software Engineering, *Handbook of Agent Technology (ed. J. Bradshaw)*, AAAI/MIT Press, 2001.
[4] A. van Lamsweerde. Goal-oriented requirements engineering: A guided tour. *In Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE'01)*, pp. 249–263, Toronto,2001.
[5] H. Mouratidis, P. Giorgini, G. Manson, I.Philp, Using Tropos Methodology to Model an Integrated Health Assessment System, *Proceedings of the 4th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002)*, Toronto-Ontario, May 2002.
[6] E. Yu, L. Cysneiros, Designing for Privacy and Other Competing Requirements, *2nd Symposium on Requirements Engineering for Information Security (SREIS'02)*, Raleigh, North Carolina, November, 2002.
[7] R. Anderson, Security Engineering, Wiley Computer Publishing, 2001.
[8] L. Padgham, M. Winikoff, The Prometheus Methodology, Book Chapter in Methodologies and Software Engineering for Agent Systems, Volume 11 of the series Multiagent Systems, Artificial Societies, and Simulated Organizations, pp. 217-234, 2004.
[9] H. Mouratidis ,P. Giorgini, G. Manson, Modelling Secure Multiagent Systems, *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS03*), Pages 859-866 2003.