بررسی و تحلیل تاثیر مجازی گری بر روی امنیت سیستم های کامپیوتری

مجید تقی‌لو
mtaghiloo@amnafzar.net
پیشگامان امن افزار

محمدعلی عاقلی
agheli@amnafzar.net
پیشگامان امن افزار

چکیده: با همه گیرشدن ماشینهای مجازی، ایجاد، تغییر، و یا توزیع ماشین جدید برای کاربران بسیار راحت خواهد بود. بهره مندی از چنین انعطاف بالایی، مزایایی فراوانی را برای کاربران فراهم خواهد کرد. تاکنون در بسیاری از روشهای ارائه شده، به منظور بالابردن امنیت سیستم موجود، سعی می‌شد در معماری سرویسها و سیستم عاملها تغییر و دستکاری صورت گیرد. اعمال چنین تغییراتی، علاوه بر پیچیدگی فراوان، گاهاً بوجودآورنده چالشهای امنیتی جدیدی نیز خواهد بود. حال آنکه، با بهره-گیری از ساختار ماشین مجازی می‌توان بسیاری از سرویسهای امنیتی را بدون دستکاری و حتی با اعتماد به سیستم عامل و برنامه‌های موجود، در لایه ناظر ماشین مجازی اضافه کرد. در این مقاله، یک معماری امنیتی مبتنی بر ماشینهای مجازی ارائه شده است که بموجب آن، بدون اعمال تغییر در سیستم های موجود، قابلیت اضافه کردن سرویسهای امنیتی فراهم شده است. برای نمایش بهتر مزایای این ساختار، مفید بودن آن را برای سرویس ثبت امن شرح می‌دهیم. همچنین در ادامه مشکلات امنیتی ناشی از استفاده از ماشینهای مجازی نیز بررسی و تحلیل شده است.

واژه های کلیدی: امنیت سیستم‌عامل، ماشینهای مجازی، سرویس‌های امنیتی.

۱- مقدمه

در سیستمهای کامپیوتری و طراحی‌های نرم افزار باید مسائل امنیتی و حفاظتی مورد توجه ویژه قرار گیرند و همچنین حفاظهای مناسب و مکانیزمهائی جهت اعمال سیاستهای امنیتی در آنها نیز باید مد نظر قرارگیرند. پروژه‌های مختلفی در حوزه امنیت هسته لینوکس طراحی شده‌اند که برخی از آنها مبتنی بر مکانیسم کنترل دسترسی الزامی(MAC) می باشند. برای مثال می‌توان از کارهایی چون: Generic Software Wrapper [1]، Linux Port of Janus [2]، Kernel Hypervisors [3]، LIDS[4]، Malcolm Beattie's MAC [5]، Medusa DS9 [6]، Pitbull LX [7]، RSBAC [8]، SAIC DTE [9]، Secure Enhanced Linux [10]، Immunix/Subdomain [11]، VXE[12]، و Willian&Mary DTE[13]، یاد کرد. دلیل عدم وجود استقبال گسترده از این روشها، ممکن است بالا بودن هزینه کلی در استفاده از آنها باشد. در این روشها، هزینه افزودن مکانیزم امنیتی MAC به دلیل ایجاد تغییر در هسته و کامپایل مجدد آن، برابر با ناسازگاری با هسته و نرم‌افزارهای موجود، افزایش سربار مدیریتی، و یا تغییر الگوهای استفاده متعارف ار سیستم می‌باشد. در میان کاربران عادی، هزینه کلی پذیرش قابلیتهای جدیدMAC، پرهزینه تر از مزایای درک آنها می‌باشد و این امر دلیلی بر دلسردی در پذیرش گسترده آن است.

برای برآورده کردن نیازهای امنیتی می توان از قابلیت‌های نهفته در ماشینهای مجازی بهره گرفت. قابلیتهای مهم و کارآمدی که در پس ماشینهای مجازی نهفته است عبارتند از: امکان ایجاد، ذخیره سازی، خواندن و تغییر، به اشتراک گذاری، انتقال و بازگشت به عقب حالت اجرا. چنین قابلیتهای منعطفی از دید کاربران و مدیران از ارزش بالایی برخوردار است بخصوص وقتی که این امکانات به همان راحتی دستکاری یک فایل برآورد می شوند.

از آنجایی که مانیتورهای ماشین مجازی، واسط سخت‌افزارهای موجود را شبیه‌سازی می‌کنند، کاربرها از این مزیت در سیستم عاملها، برنامه‌ها و ابزارهای مدیریتی موجود استفاده می‌کنند. بطوریکه در اغلب موارد سرورها و کامپیوترهای رو میزی به تدریج با محیطهای مجازی جایگزین می‌شوند. متأسفانه این انتقال راحت به گونه‌ای فریبنده نیز خواهد بود.

هدف اصلی روش ارائه شده در این مقاله ایجاد معماریی است که در آن سیستم عامل و برنامه‌هایی که بصورت مستقیم بر روی ماشین حقیقی اجرا می شوند، بگونه ایی بر روی ماشین مجازی انتقال پیدا کنند که تمام مکانیزمهای امنیتی لازم از طریق مانیتور ماشین مجازی اعمال شود.

تنها برنامه ای که بصورت مستقیم برروی ماشین واقعی اجرا می شود، سیستم عامل میزبان است. مانیتور ماشین مجازی، برنامه ای است که مدیریت محلی و سرویسهای اضافی فعال شده با ساختار مبتنی بر ماشین مجازی(مثلا معماری امنیتی ارائه شده در این مقاله)، را تدارک می بیند. بسیاری از سرویسهای شبکه باید در ماشین مجازی اجرا شوند؛ برای مثال در حوزه شبکه، فقط کافی است ماشین حقیقی بسته‌های شبکه را به ماشین مجازی تحویل دهد.

طراحی معماری امنیتی مبتنی بر ماشین مجازی، این امکان را فراهم می-کند تا بسیاری از مکانیزمها و سرویسهای امنیتی را پایین تر از سیستم عامل میهمان و برنامه هایش ایجاد کرد. به طور مثال شبیه سازی سخت‌افزار ماشین حقیقی از این قبیل سرویسها می باشد که بصورت نرم‌افزاری پیاده سازی شده است. فلذا پیاده سازی آنها در مقایسه با پیاده سازی و تهیه سخت افزار بسیار راحتتر و منعطف تر خواهد بود. در این مقاله سعی بر آن است که نگرش نوین مطرح در حوزه تحقیقات "امنیت زیربنایی" را تحلیل و ارزیابی کنیم.

ساختار مقاله در ادامه بدین صورت خواهد بود که در بخش ۲، به مزایای محیط ماشین مجازی پرداخته شده است. در بخش ۳، از موانع موجود در مسیر ارائه سرویس صحبت شده است. بخش ٤ معماری امنیتی را ارائه داده که در آن مکانیزمها و سرویسهای امنیتی می‌توانند اعمال شوند. این بخش

با پیاده سازی سرویس ثبت وقایع امنیتی در لایه امنیت بخش مانیتور ماشین مجازی، کاربرد معماری مزبور را در اعمال سرویسهای امنیتی بطور شفاف شرح می دهد. لازم به ذکر است که معماری ماشینهای مجازی خود ممکن است مشکلات امنیتی ایجاد کنند که در بخش ۵ به بررسی نمونه‌ای از آنها خواهیم پرداخت. در انتها نتایج تحقیق ارائه شده است.

۲- مزایا

ارائه سرویس از طریق دستکاری ماشین مجازی، مشابه با مزایای دستکاری ماشین واقعی می‌باشد. این سرویس‌ها به صورت مستقل از تمام processهای ماشین مجازی اجرا می‌شوند. این استقلال برای امنیت و قابلیت حمل بودن مفید می‌باشد. امنیت بالا می‌رود، زیرا سرویسها مجبور نیستند به سیستم‌عامل میهمان اطمینان کنند. آنها تنها کافی است که به مانیتور ماشین مجازی اطمینان داشته باشند. اندازه مانیتور ماشین مجازی به صورت قابل توجهی هم کوچکتر و هم ساده‌تر از کد سیستم‌عامل است. اطمینان کردن به مانیتور ماشین مجازی همانند اطمینان به پردازنده واقعی می‌باشد؛ هردو واسط محدودی را در اختیار قرار می‌دهند(مجموعه دستورالعملهای معماری). در مقابل، سرویسها در سیستم‌عامل در برابر خطاهای تصادفی و بداندیشانه بسیار آسیب‌پذیر هستند، چرا که سیستم عاملها بسیار بزرگتر و برای حفره‌های امنیتی پایدار بسیار مستعد می‌باشند. همچنین جداسازی سرویسها از سیستم‌عامل میهمان موجب بالارفتن میزان قابل حمل بودن می‌شود. ما می‌توانیم بدون تغییر در سیستم عامل، سرویسها را پیاده‌سازی کنیم. بنابراین، آنها می‌توانند با نسخه‌ها و انواع مختلف سیستم‌عامل‌ها کار کنند.

از آنجایی که ارائه سرویس در ماشین مجازی همان مزیتی را دارد که این سرویس در ماشین واقعی تدارک دیده شود، ماشین مجازی علاوه بر آن دارای مزیتهایی به نسبت ماشین فیزیکی‌ای که آن را شبیه سازی کرده است دارد. اول، ماشین مجازی بسیار راحتتر از ماشین فیزیکی دستکاری می‌شود، زیرا مانیتور ماشین مجازی که ماشین مجازی مجزا را ایجاد می‌کند یک لایه نرم‌افزاری است. دوم، دستکاری حالت ماشین مجازی بسیار ساده‌تر از دستکاری حالت ماشین فیزیکی است. حالت ماشین مجازی قابل ذخیره‌سازی، کپی، رمز، انتقال، یا بازیابی می‌باشد در حالی که هیچ یک از اینها در ماشین فیزیکی به راحتی قابل انجام نیستند. سوم، به دلیل اینکه ماشین مجازی بر روی ماشین میزبان به عنوان یک برنامه کاربردی اجرا می‌شود، دارای ارتباط بسیار سریع با سیستم محاسباتی آن می‌باشد. در مقابل، ماشینهای فیزیکی برای استفاده از توان محاسباتی همدیگر مجبورند از طریق شبکه فیزیکی باهم ارتباط برقرار کنند که در مقایسه با باس حافظه‌ای که ماشین مجازی را به میزبان وصل می‌کند بسیار کندتر است.

۳- چالشها

تهیه سرویسها در لایه ماشین مجازی دو چالش را به همراه دارد. اولی کارایی می‌باشد. اجرای تمام برنامه‌ها بر روی ماشین مجازی به دلیل وجود سربار ناشی از نمایش، به کارایی سیستم آسیب وارد می‌کند. برای مثال، توابع فراخوان سیستم در ماشین مجازی توسط مانیتور ماشین مجازی گرفته‌شده و سپس به طرف سیستم عامل میهمان ارسال می‌شود. عملیات سخت افزاری برآمده از سیستم عامل میهمان، توسط مانیتور ماشین مجازی گرفته‌شده و پس از تفسیرشدن دوباره ارسال می‌شوند. برخی از سربارها در ماشین مجازی اجتناب ناپذیر هستند؛ سرویسهای فعال شده توسط ماشین مجازی باید به این هزینه (کاهش کارایی) بچربند. نمایش تصویر، در ماشین‌های مبتنی بر x86 سربار اضافی را موجب می‌شود زیرا پردازنده‌های x86 برخی از دستورالعملهائی که باید مصورشوند را به تله نمی‌اندازند(مثل خواندن برخی از رجیسترهای سیستم). یکی از راههای پیاده‌سازی ماشین‌مجازی مجهز به این دستورالعملهای غیر مصور، بازنویسی باینریها در زمان اجرا برای مجبور ساختن این دستورالعملها جهت تله افتادن می‌باشد [15]، اما این امر مقدار قابل توجهی سربار ایجاد می‌کند.

چالش دوم برای سرویسهای ماشین مجازی، وجود یک شکاف معنایی بین ماشین مجازی و سرویس است. سرویسهای ماشین مجازی در زیر سیستم‌عامل میهمان و برنامه‌ها عمل می‌کنند. این امر می‌تواند موجب مشکل شدن ارائه سرویس شود. برای مثال، ارائه سرویس بررسی جامعیت سیستم فایل، بدون داشتن اطلاعاتی در رابطه با ساختار دیسک بسیار مشکل می‌باشد. برخی از سرویسها به هیچ اطلاعاتی از سیستم عامل میهمان نیازی ندارند. ثبت امن، یک مثالی از چنین سرویسهایی می‌باشد. برای سرویسهایی که به اطلاعاتی از لایه بالاتر نیاز دارند، این اطلاعات باید به شکلی باز سازی شوند. اطلاعات کامل، نیازمند پیاده سازی مجدد سیستم عامل میهمان در ماشین مجازی ویا زیر آن می‌باشد.

٤- معماری امنیتی

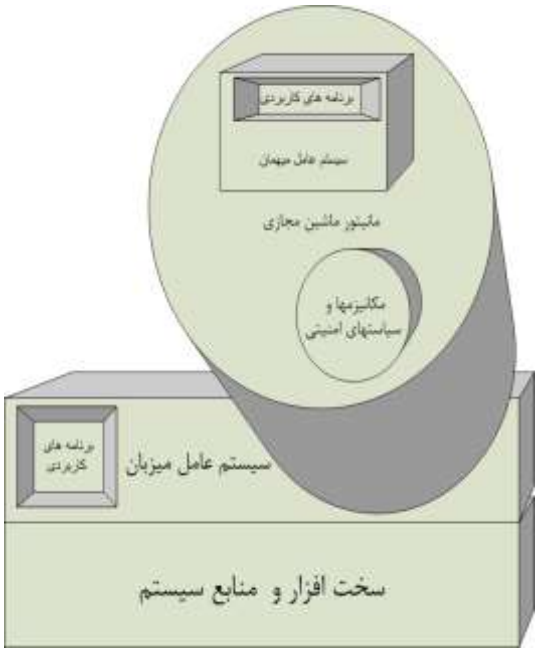
امروزه ماشینهای مجازی از جایگاه مهمی در حوزه امنیت برخوردارند. استفاده از چنین سیستم‌هایی در حوزه امنیتی، مسائل تحقیقاتی جدیدی را پیش روی محققین قرار داده است. همانطور که در شکل ۱ مشاهده می‌شود، ماشینهای مجازی بطور معمول بر روی یک سیستم عامل میزبان اجرا می‌شوند و از آن به عنوان واسط ارتباط با سخت افزار و سایر منابع سیستم استفاده می‌کند.

بخش مانیتور ماشین مجازی که وظیفه مدیریت محلی سیستم را به عهده دارد، امکان اجرای سیستم عاملهای گوناگون را به عنوان سیستم میهمان فراهم می‌کند. در معماری ارائه شده در این مقاله، این بخش به عنوان مکان هدف جهت اعمال سیاستها و مکانیزمهای امنیتی انتخاب شده است.

مزیت این معماری بر این است که، تمام تراکنشهای سیستم از قبیل ارتباط سیستم‌عامل و حتی ارتباط برنامه های کاربردی آن با دنیای خارج بصورت کاملا شفاف در اختیار بخش مانیتور ماشین مجازی قرار می‌گیرد. فلذا، با کنترل امنیت سیستم بسیار راحت و کارآمدتر از روشهای متداول خواهد بود.

ارائه یک سرویس امنیتی نمونه در این معماری می‌تواند در درک بیشتر آن کمک شایانی داشته باشد. از این رو، در ادامه روش ارائه سرویس ثبت

امن به عنوان بخشی از "مکانیزمها و سیاستهای امنیتی" معماری، بطول کامل طراحی و ارزیابی شده است.



شکل۱- نمایش نحوه ارائه سرویسهای امنیتی در ماشینهای مجازی

٤-١- ثبت امنیت

بسیاری از سیستم عاملها به عنوان بخشی از استراتژی امنیت، رخدادهای جالب توجه را ثبت می‌کنند. برای مثال، یک سیستم ممکن است یک رکوردی از تلاشهایی که برای login کردن انجام شده و یا نامه‌های ارسالی و دریافتی را در خود نگهدارد. مدیران شبکه از اطلاعات ثبت شده برای اهداف گوناگونی استفاده می‌کنند. برای نمونه، داده‌های ثبت شده ممکن است به مدیر شبکه کمک کند تا از نحوه دسترسی نفوذگر شبکه به سیستم اطلاع کسب کند، یا از آسیبهایی که نفوذگر پس از دسترسی به سیستم به آن وارد کرده آگاهی پیدا کند.

متاسفانه واقعه نگاری استفاده شده در سیستم‌های موجود دارای دو ضعف مهم از منظر جامعیت هستند: اول، حمله کننده می‌تواند به راحتی فرایند واقعه نگاری را پس از مسلط شدن به سیستم خاموش‌کند؛ بنابراین، محتوای ثبت شده پس از نفوذ به سیستم غیر قابل اعتماد خواهد بود. دوم، پیش بینی نوع اطلاعات مورد نیاز برای تحلیل بعد از حمله بسیار دشوار می‌باشد؛ بنابراین داده‌های ثبت شده ممکن است فاقد اطلاعات مورد نیاز جهت تشخیص نحوه انجام نفوذ و یا کارهای انجام شده پس از نفوذ می‌باشد.

ماشینهای مجازی فرصتی را برای تصحیح سیستم واقعه‌نگاری مرسوم فراهم می‌کنند. برای بالابردن جامعیت واقعه نگاری، ما می‌توانیم سیستم واقعه نگاری را به خارج از سیستم عامل برده و در مانیتور ماشین مجازی قرار دهیم. مانیتور ماشین مجازی بسیار کوچکتر و ساده‌تر از سیستم عامل میهمان است و در برابر حملات از آسیب‌پذیری کمتری برخوردار می‌باشد. از طریق انتقال نرم افزار واقعه‌نگاری به درون مانیتور ماشین مجازی، ما آن را در واقع وارد ناحیه‌ای کرده‌ایم که خارج از کنترل نفوذگر است. حتی در زمانی که نفوذگر حق دسترسی با امتیاز بالا را نیز بدست بیاورد و یا به طور کامل سیستم عامل میهمان را جایگزین کند، نمی‌تواند تاثیری در نرم‌افزار واقعه نگاری و با داده های ثبت شده برجای بگذارد. داده‌های ثبت شده می‌توانند به سرعت در سیستم عامل میزبان نوشته شوند، دلیل این امر به ارتباط سریع مانیتور ماشین مجازی و کامپیوتر میزبان مربوط می‌شود.

برای بالابردن جامعیت واقعه‌نگاری، ما روشی را پیشنهاد می‌کنیم که در آن بتوان با نگهداری داده‌های کافی، اجرای کامل ماشین مجازی را دوباره بتوان انجام داد[۱۷]. اطلاعات مورد نیاز برای انجام عملیات بازگشت به عقب صحیح، به نقاط بازگشتی که در آن نقاط ماشین مجازی مقداردهی اولیه - شده و همچنین رخدادهای غیرقطعی که به اجرای اصلی ماشین‌مجازی در زمان ذخیره‌سازی نقطه بازگشت تاثیر گذاشته است، محدود می‌شود. این رخدادهای غیرقطعی به دو دسته تقسیم می‌شوند: ورودی‌های خارجی و زمان. ورودی‌های خارجی به داده‌هایی مربوط می‌شود که از طریق رخدادهای ثبت نشده ارسال می‌شوند، مثل کاربر انسان و یا کامپیوتر خارجی(سرور وب). زمان به نقطه‌ای از جریان اجرا اشاره می‌کند که در آن یک رخداد اتفاق افتاده است. به عنوان مثال، برای تکرار الگوی جایگذاری بین نخ‌ها، باید آن دستورالعملی که توسط وقفه زمان قبضه شده است را ثبت کنیم [18] (فرض ما بر این است که مانیتور ماشین مجازی بر روی سیستم چندپردازنده اجرا نمی شود). به خاطر داشته باشید که بسیاری از دستورالعملهای اجرا شده در ماشین مجازی نیازی به ثبت شدن ندارند؛ تنها رخدادهای غیرقطعی مرتبط با هم که تعداد آنها کم می‌باشد، نیاز به ثبت شدن دارند.

استفاده از مانیتور ماشین مجازی برای انجام فرایند واقعه نگاری امن یک سری چالشهای تحقیقاتی را به وجود می‌آورد. اولین سوال، مقدار حجم مورد نیاز برای ثبت داده جهت پشتیبانی از تکرار می‌باشد. به نظر می‌رسد که حجم داده‌هایی که نیاز به ثبت دارند، نباید هزینه بالایی داشته باشد. همه رخدادهای غیرقطعی محلی، مثل رخدادهای زمانبندی نخها و ورودی‌های کاربران، حجم کوچکی را اشکال می‌کنند. خواندن داده از دیسک می‌تواند بزرگ باشد، اما اینها فرایند قطعی هستند.

بزرگترین تولید کننده داده ثبت، بسته‌های ورودی شبکه می‌باشند. ما می توانیم حجم داده ثبت شده شبکه را با استفاده از تکنیکهای ثبت پیغام توسعه داده شده در [۱۹]، به صورت چشمگیر کاهش دهیم. برای مثال، هیچ نیازی به ثبت داده‌های دریافتی از کامپیوترهایی که خورشان ثبت وقایع می‌کنند، نیست، زیرا که این کامپیوترها می‌توانند از طریق تکرار دوباره داده پیغامهای ارسالی را دوباره ایجاد کنند. اگر همه کامپیوترهای موجود در یک شبکه محلی مشابه، در طول ثبت و تکرار با هم همکاری داشته‌باشند، تنها پیغامهایی نیاز به ثبت خواهند داشت که از سایت خارجی آمده باشند. برای یک کلاس سرویس مهم (مثل: سرورهای وب) حجم داده‌های دریافتی بصورت قابل توجهی کوچک می‌باشد (درخواست های GET و HTTP POST). گذشته از این حرفها، قیمت دیسک بصورت متوالی نزول پیدا می‌کند، بسیاری از کامپیوترها (بخصوص سرورهایی که وظیفه ثبت وقایع را دارند)

قادرخواهند بود تا چندین گیگابایت را برای نگهداری داده‌های ثبت شده اختصاص دهند [20].

سوال تحقیقاتی دوم طراحی ابزارهای تحلیل رفتار ماشین مجازی در طول اجرای مجدد می‌باشد. نوشتن ابزارهای کارآمد تحلیل، در این حوزه یک چالش می‌باشد زیرا یک شکاف معنایی بین رخدادهای ماشین مجازی و فعالیتهای سیستم عامل مربوطه وجود دارد. ابزار تحلیل ممکن است مجبور به تکرار برخی از قابلیتهای سیستم عامل برای تبدیل داده‌های ثبت شده به اطلاعات مفید باشد. برای مثال، ابزار تحلیل ممکن است نیاز به فهم فرمت سیستم فایل موجود بر روی دیسک جهت ترجمه انتقال دیسک دیده شده توسط مانیتور ماشین مجازی به انتقال سیستم فایل تولید شده در سیستم عامل دارد. ترجمه رخدادهای ماشین مجازی به رخدادهای سیستم عامل یک چالش مهم می‌باشد(و حتی ناممکن) بخصوص اگر نفوذگر، سیستم‌عامل را دستکاری کرده باشد. یک خانواده از ابزارهای تحلیل که ما آرزو داریم ساخته شود ابزاری است که بتواند جریان اطلاعات سیستم را ردیابی کند.

۵- مسائل امنیتی ماشین مجازی

با توجه به ارئه معماری امنیتی و نیز تحلیل و توصیه آن در بخش قبل، لازم به ذکر است که ماشینهای مجازی به دلیل خصوصیات ذاتی خود بستر برخی مسائل امنیتی می‌باشند و در این بخش به بررسی آنها می‌پردازیم.

مانیتور ماشین مجازی، یک لایه‌ای از نرم‌افزار بین سیستم‌عامل و سخت افزار ماشین ایجاد می‌کند. در اصل یک ماشین مجازی کل حالت سیستم عامل میهمان را نگهداری می‌کند. حالت ماشین همانند یک فایل معمولی، می‌تواند کپی و یا در شبکه به اشتراک گذاشته شود و همچنین می‌تواند در شبکه‌ها نمونه سازی شود و همانند ماشین فیزیکی نیاز به پیکربندی و مدیریت دارد. می‌توان حالت ماشین مجازی را مثل ماشین فیزیکی از طریق اجرا و یا دستکاری مستقیم فایل تغییر داد.

برای **رشد و توسعه** ماشینهای فیزیکی محدودیت‌هایی چون زمان و مقدار بودجه موجود در سازمان مربوطه، مشهود می‌باشد. در مقابل، تولید ماشین مجازی جدید به راحتی کپی کردن یک فایل است. هر کاربر می‌تواند برای اهداف مختلف (مثل کارهای تستی یا آزمایشگاهی)، می‌تواند چندین ماشین مجازی ایجاد کند. گسترش سریع محیطهای مجازی می‌تواند بر سیستم‌های امنیتی سازمان تاثیر بگذارد. تمام وظایف مدیریتی نمی‌توانند به صورت تمام خودکار انجام شوند. ارتقاء، مدیریت بسته‌ها، و پیکربندی، از طریق استفاده ترکیبی از ابزارهای خودکار و یک سری عملیات دستی مدیران قابل انجام است. در نتیجه، رشد سریع و غیرقابل پیش بینی ماشینهای مجازی می‌تواند وظایف مدیریتی را سخت‌تر کند و بصورت قابل توجه تاثیر رخدادهای فاجعه‌آمیز را تشدید می‌کند، برای مثال در مقابله با حملات کرمهای کامپیوتری روی تمام ماشینها باید بسته امنیتی نصب شود، از نظر آسیب‌پذیری بررسی شوند، و کدهای مخرب روی آنها پاک شود.

مسئله قابل بحث دیگر **ناپایداری** می‌باشد. در محیطهای کامپیوتری متعارف مثل یک شبکه کاربران دارای یک یا چند ماشین هستند که در بیشتر زمانها روشن می‌باشد. در مقابل، مجموعه‌ای از ماشینهای مجازی خاص که در برابر برخی از پدیده‌ها فعال می‌شوند، این حالت را در شبکه ایجاد می‌کنند که تعداد زیادی از ماشینها به صورت تصادفی در شبکه ظاهر شده و یا از دید آن پنهان شوند. شبکه‌های مرسوم به سرعت می‌توانند به یک حالت خوب شناخته‌شده برسند، با حضور ماشینهای ناپایدار در شبکه، رسیدن به حالت شناخته شده ناممکن خواهد بود. برای مثال، وقتی یک کرم الکترونیکی وارد شبکه می‌شود بطور معمول تمام ماشینهای آسیب‌پذیر به سرعت آلوده خواهند شد. در زمان وقوع این اتفاق، مدیر به راحتی می‌تواند ماشینهای آلوده را شناسایی کند، سپس با پاک کردن کرم و نصب بسته های امنیتی لازم به سرعت شبکه را به حالت ثابت می‌رساند.

در محیطهای مجازی کنترل نشده، به چنین حالت پایداری هرگز نمی‌توانیم دسترسی پیداکنیم. ماشینهای آلوده در مدت کوتاه ظاهر می‌شوند و ماشینهای دیگر را آلوده می‌کنند و سپس قبل از شناسایی‌شان از دید شبکه پنهان می‌شوند.

یکی از اصول ساخت سیستم‌های امنیتی به حداقل رساندن مقدار زمان حفظ داده حساس در سیستم است. مانیتور ماشین مجازی این اصل را تخریب می‌کند. برای مثال، مانیتور ماشین مجازی برای انجام عملیات بازگشت به عقب مجبور است حالت اجرا را ثبت کند. این امر سعی سیستم عامل میهمان را برای نابود کردن داده‌های حساس(مثل: کلیدهای رمزنگاری، مستندات پزشکیو..) بی‌حاصل می‌گذارد. پس با این تفاسیر مفهوم **عمرداده** در ماشینهای مجازی بی‌معنا خواهد بود.

٦- نتیجه

اجرای یک سیستم‌عامل و بسیاری از برنامه‌های کاربردی در ماشین مجازی، این قابلیت را به طراح سیستم می‌دهد که سرویسهای امنیتی را در دربخش "مکانیزمها و سیاستهای امنیت" معماری ارائه شده (شکل ۱) اضافه کنند. این ساختار سرویسها را بدون حصول اطمینان و یا دستکاری سیستم‌عامل یا برنامه‌های کاربردی، فراهم می‌کند. با توجه به خصوصیات یاد شده ماشین مجازی، بسیاری از سرویسهای امنیتی مثل: تشخیص نفوذ و ... را می‌توان در سطح لایه ماشین مجازی پیاده سازی کرد. این انتقال برای تمام سرویسهای امنیتی قابل انجام می‌باشد و به عنوان کارهای تحقیقاتی آتی می‌تواند در حوزه امنیت مطرح شود. در دنیای امروز برای بالابردن سطح امنیت سرورها (مثل وب سرور) می‌توان از ماشینهای مجازی استفاده کرد.

مراجع

[1] T. Fraser, L. Badger, and M. Feldman. Hardening COTS Software with Generic Software Wrappers. In Proceedings of the 1999 IEEE Symposium on Security and Privacy, pages 2–16, Berkeley, California, May 1999.

processes. In Proceedings of 1996 IEEE Symposium on Computer Security and Privacy, 1996.

[23] Gene H. Kim and Eugene H. Spafford. The design and implementation of Tripwire: a file system integrity checker. In Proceedings of 1994 ACM Conference on Computer and Communications Security, November 1994.

[2] I. Goldberg, D. Wagner, R. Thomas, and E. Brewer. A Secure Environment for Untrusted Helper Applications. In Proceedings of the 6th USENIX Security Symposium, pages 1–13, San Jose, California, July 1996.

[3] T. Mitchem, R. Lu, and R. O'Brien. Using Kernel Hypervisors to Secure Applications. In *Proceedings of the 13th Annual Computer Security Applications Conference*, San Diego, California, December 1997.

[4] H. Xie and P. Biondi. LIDS. http://www.lids.org.

[5] M. Beattie. MAC. http://users.ox.ac.uk/ mbeattie/linux.

[6] M. Zelem, M. Pikula, and M. Ockajak. Medusa DS9. http://medusa.fornax.sk.

[7] Argus Systems Inc. Pitbull LX. http://www.argussystems.com/products/white paper/lx/.

[8] A. Ott. Regel-basierte Zugriffskontrolle nach dem Generalized Framework for Access Control-Ansatz am Beispiel Linux. Master's thesis, Universitat Hamburg, Fachbereich Informatik, 1997.

[9] SAIC. SAIC DTE. http://research-cistw.saic.com/cace/dte.html.

[10] P. A. Loscocco and S. D. Smalley. Integrating Flexible Support for Security Policies into the Linux Operating System. In Proceedings of the FREENIX Track: USENIX Annual Technical Conference, June 2001.

[11] C. Cowan, S. Beattie, G. Kroah-Hartman, C. Pu, P. Wagle, and V. Gligor. SubDomain: Parsimonious Server Security. In Proceedings of the 14th USENIX Systems Administration Conference (LISA 2000), New Orleans, LA, December 2000.

[12] U. InteS, Odessa. VXE. http://www.intes.odessa.ua/vxe.

[13] S. Hallyn and P. Kearns. Domain and Type Enforcement for Linux. In *Proceedings of the 4th Annual Linux Showcase* & Conference (ALS 2000), Atlanta, Georgia, October 2000.

[14] Peter M. Chen, Brian D. Noble, "When Virtual Is Better Than Real," *hotos*, p. 0133, Eighth Workshop on Hot Topics in Operating Systems, 2001.

[15] Kevin Lawton. Running multiple operating systems concurrently on an IA32 PC using virtualization techniques, 1999. http://plex86.org/research/paper.txt.

[16] Robert P. Goldberg. Survey of Virtual Machine Research IEEE Computer, pages 34–45, June 1974.

[17] Thomas C. Bressoud and Fred B. Schneider. Hypervisor-Based Fault-Tolerance. In Proceedings of the 1995 Symposium on Operating Systems Principles, pages 1–1, December 1995.

[18] Mark Russinovich and Bryce Cogswell. Replay for concurrent non-deterministic shared-memory applications. In Proceedings of the 1996 Conference on Programming Language Design and Implementation (PLDI), pages 258–266, May 1996.

[19] David B. Johnson and Willy Zwaenepoel. Sender-Based Message Logging. In Proceedings of the 1987 International Symposium on Fault-Tolerant Computing, pages 14–19, July 1987.

[20] John D. Strunk, Garth R. Goodson, Michael L. Scheinholtz Craig A.N. Soules, and Gregory R. Ganger. Self-securing storage: Protecting data in compromised systems. In Proceedings of the 2000 Symposium on Operating Systems Design and Implementation (OSDI), October 2000.

[21] Ian Goldberg, David Wagner, Randi Thomas, and Eric A. Brewer. A Secure Environment for Untrusted Helper Applications. In *Proceedings of the 1996 USENIX* Technical Conference, July 1996.

[22] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for Unix