ارائه یک راهکار امنیتی در لایه زیرساخت رایانش ابری با استفاده از فناوری محاسبات مطمئن

زینب آل‌بویه[۱]، محمدکاظم اکبری[۲] و مرتضی سرگلزایی جوان[۳]
[۱] دانشگاه آزاد اسلامی واحد علوم و تحقیقات تهران – دانشکده فنی و مهندسی - z.alebouyeh@gmail.com
[۲]دانشگاه صنعتی امیرکبیر- دانشکده مهندسی کامپیوتر و فناوری اطلاعات - akbarif@aut.ac.ir
[۳]دانشگاه صنعتی امیرکبیر- دانشکده مهندسی کامپیوتر و فناوری اطلاعات - msjavan@aut.ac.ir

چکیده - امنیت ماشین‌های مجازی که در لایه زیرساخت محاسبات ابری قرار دارند اهمیت بسیار زیادی دارد. تا کنون راهکارهای نرم‌افزاری زیادی برای توسعه امنیت در ماشین‌های مجازی ارائه شده است. اما استفاده از نرم‌افزار به تنهایی برای حل مشکلات امنیتی کافی نیست. در این مقاله ساختاری برای اجرای کاملا ایزوله کدهای حساس به امنیت با استفاده از سخت‌افزارهای محاسبات مطمئن، در محیط‌های مجازی ارائه شده است. همچنین این ساختار امکان تصدیق کد اجرا شده را به همراه ورودی‌ها و خروجی‌ها، برای سیستم راه دور فراهم می‌کند. ما این ویژگی‌های امنیتی را حتی در شرایطی که بایوس، سیستم‌عامل و حتی فوق‌ناظر آلوده باشند، فراهم می‌کنیم. برای رسیدن به این اهداف، از پشتیبانی‌های سخت‌افزاری که توسط پردازنده‌های جدید *Intel* و *AMD* و همچنین چیپ امنیتی *TPM* ارائه شده است استفاده خواهیم کرد. استفاده از این فناوری‌ها در نهایت یک ریشه اعتماد پویا بوجود آورده و موجب کاهش *TCB* برای کدهای حساس به امنیت می‌شود.

کلید واژه‌ها- امنیت، ماشین مجازی، محاسبات ابری، محاسبات مطمئن، ریشه اعتماد پویا.

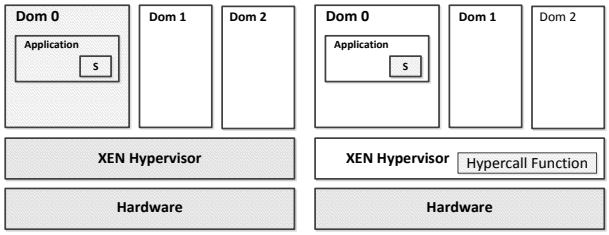
۱. مقدمه

امروزه حجم زیاد کد و پیچیدگی سیستم‌عامل‌های رایج باعث آسیب‌پذیری آنها نسبت به حملات نرم‌افزاری شده است. این سیستم‌ها حجم زیادی از کد را در ممتازترین حلقه‌ی پردازنده، یعنی حلقه‌ی صفر اجرا می‌کنند. از آنجایی که آسیب‌پذیری کدها غیرقابل اجتناب است این حجم کد امکان آلوده‌کردن سیستم را افزایش می‌دهد و چون این کدها در حلقه‌های با اولویت بالا در پردازنده اجرا می‌شوند اگر نفوذی از طریق این کدها صورت گیرد، نفوذگر نیز همین اولویت بالا، یعنی اولویت سیستم‌عامل و یا فوق‌ناظر را در سیستم خواهد داشت.

قلب یک سیستم کامپیوتری مطمئن پایه محاسباتی مطمئن (TCB)Trusted Computing Base است که شامل تمام بخش‌هایی از سیستم است که مسئول پشتیبانی از سیاست امنیتی و پشتیبانی از انزوای اشیا است. امنیت یک سیستم بستگی به امنیت بخش‌هایی از آن دارد که در داخل این ناحیه قرارگرفته‌اند. در سیستم‌های کامپیوتری جامعیت و امنیت هر برنامه‌ای در معرض خطر است زیرا تضمین امنیت یک برنامه فراتر از صرف تضمین امنیت کد آن برنامه است و TCB هر برنامه و به تبع آن امنیت آن، بستگی به لایه‌های نرم‌افزاری مختلفی از جمله، بایوس، سیستم‌عامل، فوق‌ناظر و کد خود برنامه است. با این توصیف‌ها یک راه مؤثر برای بهبود امنیت در برنامه‌ها کاهش اندازه TCB سیستم مربوطه است.

در این مقاله راهکاری ارائه شده است که TCB کدهای حساس به امنیت در ماشین‌های مجازی را به طور چشمگیری کاهش خواهد داد. در روش ارائه شده معماری سیستم به گونه‌ای است که برای تضمین امنیت قطعه کد حساس به امنیت، فقط باید امنیت یک تابع مخصوص در فوق‌ناظر و قطعه کد فراهم شود. در واقع، فوق‌ناظر، دامنه‌ی صفر و برنامه‌ی کاربردی از ناحیه TCB حذف شده‌اند.

شکل ۱ نمای کلی TCB سیستم را در هر دو حالت نشان می‌دهد. در شکل، ناحیه هاشور خورده مربوط به قسمت‌های قرارگرفته در TCB سیستم است. شکل سمت چپ مربوط به یک سیستم مجازی‌سازی معمولی و شکل سمت راست مربوط به راهکار ارائه شده در این تحقیق است.



شکل ۱: ناحیه TCB، سمت چپ سیستم مجازی معمولی، سمت راست سیستم ارائه شده

در ادامه ابتدا پیش زمینه‌ای در مورد فناوری‌های به کار رفته در این معماری ارائه داده و سپس راهکارهای امنیتی مرتبط را بررسی خواهیم کرد. در ادامه اهداف و معماری ارائه شده را توضیح داده و یک کاربرد مناسب برای معماری پیشنهاد می‌دهیم.

۲. پیش‌زمینه

در این قسمت به معرفی فوق‌ناظر Xen و سخت‌افزارهای محاسبات مطمئن می‌پردازیم.

۲,۱. فوق‌ناظ Xen

Xen یک فناوری مجازی‌سازی متن باز است که امکان اجرای موازی چندین سیستم‌عامل را روی یک منبع سخت‌افزاری فیزیکی فراهم می‌کند. یک سیستم Xen از چندین لایه تشکیل شده است که پایین‌ترین لایه آن که بیشترین اولویت را نیز دارد خود فوق‌ناظر Xen است که مستقیما روی سخت‌افزار اجرا می‌شود. Xen به هر کدام از ماشین‌های مجازی مهمان که روی آن اجرا می‌شوند، دامنه می‌گوید.

به طور کلی Xen دو نوع دامنه مهمان دارد:۱. دامنه ممتاز یا دامنه صفر و ۲. دامنه‌های مهمان غیرممتاز یا معمولی. دامنه صفر نه تنها حق دسترسی به منابع ورودی/خروجی فیزیکی را دارد بلکه می‌تواند با دیگر ماشین‌های مجازی که روی سیستم اجرا می‌شوند، تعامل کند. مهمان‌های دامنه غیرممتاز هیچ دسترسی مستقیمی به سخت‌افزار فیزیکی ماشین ندارند و به همین دلیل به آنها دامنه‌های غیرممتاز می‌گویند.[۱۰][۲۷]

۲,۲. فناوری محاسبات مطمئن

محاسبات مطمئن فناوری است که توسط گروه محاسبات مطمئن (TCG) ارائه شده است. گروه محاسبات مطمئن ابتکاری است که توسط شرکت‌های AMD، Intel، HP، IBM و Microsoft برای پیاده‌سازی محاسبات مطمئن آغاز شد. هدف محاسبات مطمئن محافظت از حساس‌ترین اطلاعات در مقابل ربوده شدن و یا استفاده شدن توسط کدهای آلوده است. محاسبات مطمئن فرض می‌کند که امکان آلوده شدن نرم‌افزار در طی دوران کار آن وجود دارد، پس شرایطی را فراهم می‌کند که کلیدهای حساس حتی در حالت آلوده شدن، محفوظ باشند. محاسبات مطمئن برای رسیدن به اهداف امنیتی خود از یکسری امکانات سخت‌افزاری استفاده می‌کند. این امکانات سخت‌افزاری شامل ماژول TPM (Trusted Platform Module) و پردازنده‌های جدید است که در قسمت‌های بعدی مختصرا آنها را شرح خواهیم داد.

۲ ۲ ۱- ماژول پلت‌فرم مطمئن (TPM)

اکثر برنامه‌های ارائه شده در زمینه‌ی محاسبات مطمئن مبتنی بر چیپ کوچکی به نام TPM هستند. این چیپ توسط گروه محاسبات مطمئن به منظور مقاوم‌تر کردن پلت‌فرم در برابر حملات نرم‌افزاری طراحی شده است.

TPM در بردارنده‌ی یک شاه کلید محرمانه است که می‌تواند محافظت در برابر سایر اطلاعات ذخیره شده در سیستم را ممکن سازد. از آنجایی که اسناد می‌توانند در سخت افزار TPM ذخیره شوند، تجاوز به آن‌ها کار بسیار مشکلی است. یک کلید خصوصی در داخل این چیپ قرارگرفته و یک کلید عمومی از آن صادر می‌شود و این چیپ می‌تواند عملیات رمزنگاری را در داخل خود انجام دهد. این چیپ دارای تعدادی ثبات پیکربندی پلت‌فرم است که مقادیر این ثبات‌ها توسط عملیات توسعه تغییر می‌کنند. عملیات توسعه (Extend) به صورت زیر صورت می‌گیرد:

$$\text{Extend}(\text{Data}, PCR_n) \rightarrow PCR_n = \text{SHA1}(PCR_n \parallel \text{Data})$$

در عملیات فوق علامت || نشان‌دهنده‌ی عملیات الحاق است. عبارت فوق بدان معنا است که وقتی می‌خواهیم مقدار جدیدی را در یک ثبات ذخیره کنیم مقدار جدید به محتوای قبلی ثبات الحاق شده و از کل آنها Hash گرفته می‌شود.

این ماژول برای رسیدن به اهداف امنیتی محاسبات مطمئن دو کار اصلی برای ما انجام می‌دهد[۳۵][۳۹]: تصدیق از راه دور و صحه گذاشتن.

۱. تصدیق از راه دور (Remote Attestation): این قسمت کار اصلی محاسبات مطمئن است. برای انجام این کار محیط سخت‌افزار، دنباله بوت، و پیکربندی سیستم عامل میزبان ثبت شده و برای شخص سومی ارسال می‌شود تا تصدیق کند که حالت سخت افزار و نرم‌افزار امن است. این اطلاعات توسط

TPM رمزنگاری شده و به همراه کلید عمومی ارسال می‌شوند تا صحت اطلاعات تائید گردد.

۲. ذخیره‌سازی قفل شده (Sealed Storage): این ویژگی برای رمزنگاری داده‌ها به صورتی است که فقط تحت همان محیط چه از نظر نرم‌افزار و چه از نظر سخت‌افزار، قابل رمزگشایی است. برای دستیابی به این هدف داده با کلیدی که توسط TPM ایجاد شده رمزنگاری می‌شود. این کلید با اطلاعات پیکربندی پلت‌فرم ترکیب شده است.

چیپ TPM v1.2 دارای ۲۴ ثبات پیکربندی پلت‌فرم PCR (Platform Configuration Register) است که هر کدام از این ثبات‌ها قابلیت ذخیره‌سازی یک hash ۱۶۰ بیتی را دارند. ثبات‌های شماره صفر تا ۱۶ ثبات‌های ایستا و ثبات‌های شماره ۱۷ تا ۲۳ ثبات‌های پویا هستند. . مقادیر ثبات‌های ایستا با ریست شدن کل سیستم و چیپ TPM صفر می‌شود اما مقادیر ثبات‌های پویا با ریست سخت‌افزاری ۱- می‌شود و تنها دستور امنیتی پردازنده‌های جدید Intel و AMD مقادیر این ثبات‌ها را صفر خواهد کرد. این ویژگی این امکان را به ما می‌دهد که بدون نیاز به ریست کل سیستم یک ریشه اعتماد پویا برای اجرای کدهای حساس به امنیت فراهم کنیم.

۲ ۴ ۴ - ویژگی شروع با تاخیر /ریشه اعتماد پویا

یکی از اهداف گروه محاسبات مطمئن اطمینان از این است که سیستم‌عامل در زمان راه اندازی مجدد سیستم آلوده نباشد. این گروه دو راه را برای ساخت این اعتماد در طی فرآیند بوت تعریف کرده است: ریشه اعتماد ایستا و ریشه اعتماد پویا. ریشه اعتماد ایستا با استفاده از چیپ TPM و ریشه اعتماد پویا با استفاده از دستور امنیتی پردازنده‌ها و چیپ TPM ساخته می‌شود.

ریشه اعتماد ایستا یک TCB نرمال شامل بایوس، BootLoader و کل سیستم‌عامل است که بسیار بزرگ است پس کاهش سایز آن یکی از مسائل مهم در محاسبات مطمئن به شمار می‌رود[۳۵].

پردازنده‌های جدید Intel و AMD دستورالعمل‌هایی دارند که با کمک آنها می‌توان چندین لایه پایین نرم‌افزاری را از TCB سیستم حذف کرد. این قابلیت شروع با تاخیر در پردازنده‌های AMD به عنوان بخشی از فناوری (Secure Virtual Machine) SVM [۶][۳] و در Intel بخشی از (Trusted Execution TXT Technology) [۱۳] است.

این تکنولوژی‌های جدید این امکان را فراهم می‌کنند که قطعه کدی در یک محیط کاملا ایزوله بتواند اجرا شود. پس می‌توانیم کدهای حساس به امنیت را در این محیط‌های ایزوله اجرا کنیم که در این حالت دیگر نیاز نیست به TCB نرمال اعتماد کنیم؛ زیرا تحت این شرایط TCB کدی می‌شود که در محیط ایزوله در حال اجرا است و معمولا به صورت پویا بارگذاری می‌شود. به همین جهت ریشه اعتماد پویا نام گرفته است.

دستورالعمل‌های SKINIT در AMD و SENTER در Intel، دستوراتی هستند که امکان شروع با تاخیر ناظر ماشین مجازی، هسته امن و یا هر قطعه کد حساس به امنیت دیگری را در هر زمانی و با حفاظت در مقابل حملات نرم‌افزاری، فراهم می‌کنند. هنگام اجرا حفاظت‌های حافظه برای ناحیه‌ای که این کدها در آن واقع شده‌اند فعال می‌گردد. سپس پردازنده صحت کد را با استفاده از TPM بررسی می‌کند و با انجام فعالیت‌هایی از قبیل غیرفعال کردن وقفه‌ها، اجرای امن این کدها را تضمین می‌کند[۳][۴][۱۳].

TPM v1.1 ریشه‌ی اعتماد ایستا را در اختیار ما قرار می‌دهد. اندازه‌گیری‌های پیاپی در ریشه‌ی اعتماد ایستا منجر به ایجاد زنجیره‌ای از اعتماد می‌شود که توسط یک سیستم راه دور می‌تواند مورد ارزیابی قرار گرفته و مشخص گردد که به یک سیستم مطمئن دسترسی دارد یا خیر.

بدلیل مشکلاتی که ریشه اعتماد ایستا داشت، روش جدیدی به نام ریشه‌ی اعتماد پویا توسط گروه محاسبات مطمئن ایجاد شد. فناوری‌های AMD SVM و Intel TXT عملیات شروع با تأخیر را برای خودراه‌اندازی یک ریشه‌ی اعتماد پویا فراهم می‌کنند. اولین تفاوت این است که ریشه‌ی اعتماد پویا می‌تواند بدون نیاز به شروع مجدد کل سیستم ساخته شود.

شروع با تأخیر، پردازنده را به یک حالت امن شناخته شده می‌برد بدون اینکه نیاز به شروع مجدد کل سیستم باشد. این عملیات شامل پیکربندی کنترل‌کننده‌ی حافظه سیستم برای ممانعت دسترسی به کد محافظت شده در مقابل دستگاه‌های با قابلیت دسترسی مستقیم به حافظه است. سپس یکی از ثبات‌های پویا که مقدار آن صفر شده به طور خودکار با مقدار Hash کدی که می‌خواهیم به صورت امن اجرا شود توسعه پیدا می‌کند.

۳. کارهای مرتبط

در طی سال‌های گذشته محققان سیستم‌های امنیتی متعددی را توسعه داده‌اند که این سیستم‌ها بر ویژگی‌های ماشین‌های مجازی برای فراهم‌کردن یک سطح امنتی بالاتر تکیه کرده‌اند.

در سیستم‌های تشخیص نفوذ معمولی به دلیل اینکه سیستم نظارتی در همان ماشینی اجرا می‌شود که می‌خواهیم آن را نظارت کنیم، سیستم نظارتی می‌تواند تحت حمله قرار گیرد. اما در سیستم‌های مطرح شده در [۳۷]، [۲۱]، [۲۳] و [۷] به دلیل اینکه بخش نظارتی سیستم به قسمت فوق‌ناظر، سیستم‌عامل میزبان و یا به یک ماشین مجازی امن منتقل شده است، دیگر امکان از کار انداختن سیستم نظارتی وجود ندارد.

روش‌های [۱۴]، [۲۵]،[۲۴] و [۵]، سیستم‌های نظارت بر جامعیت فایل هستند که با اعمال تغییرات در هسته سیستم‌عامل مهمان، امکان نظارت بر عملیات فایل را در یک ماشین مجازی ممتاز فراهم می‌کنند. مقاله [۱۶] روشی ارائه داده است که حتی نیاز به هیچگونه تغییری در هسته سیستم مهمان نیز نیست.

در [۴۲]، [۳۰] و [۲] نیز سیستم‌های ممانعت از نفوذی ارائه شده است که برای جلوگیری از غیرفعال شدن توسط نفوذگر، سیستم نظارتی را در سیستم‌عامل میزبان، فوق‌ناظر و یا یک ماشین مجازی ممتاز قرار داده‌اند.

با وجود اینکه در اکثر روش‌های ذکر شده بخش نظارتی به دامنه ممتاز منتقل شده و نظارت از نظر ماشین مجازی مهمان شفاف است ولی هنوز این سیستم‌ها از نظر امنیت مشکلاتی دارند. دامنه ممتاز نیز یک هسته لینکوس تغییر یافته است و اگرچه نفوذ به آن سخت است ولی غیرممکن نیست. پس در صورتی که به این دامنه نفوذ شود، نمی‌توان به نتایج حاصل از سیستم‌های امنیتی اطمینان کرد.

در سال‌های اخیر راهکارهای امنیتی زیادی ارائه شده است که برای تضمین امنیت به سمت سخت افزار رفته و با استفاده از قابلیت‌های امنیتی پردازنده‌های جدید و ماژول TPM، امنیت را در سیستم‌های مختلفی از قبیل سیستم‌های معمولی، سیستم‌های مجازی و بسترهای محاسباتی جدید از قبیل رایانش ابری توسعه داده‌اند. در این بخش به بررسی نمونه‌هایی از سیستم‌های طراحی شده در این زمینه می‌پردازیم.

در [۲۰]، [۱۸] و [۳۱] روش‌هایی ارائه شده که با استفاده از دستورات پردازنده‌ها و چیپ TPM امکان اجرای امن کدهای حساس به امنیت را در سیستم‌های معمولی فراهم می‌کنند. در [۶] نیز روشی مطرح شده است که با استفاده از دستور امنیتی پردازنده و ماژول TPM یک ریشه‌یاعتماد پویا برای سیستم فراهم می‌کند. روش مطرح شده در [۳۳] ماژول TPM را مجازی‌سازی می‌کند تا با استفاده از یک ماژول بتوانیم امنیت را برای شمار نامحدودی ماشین مجازی فراهم کنیم. در [۳۴] راهکاری برای سنجش جامعیت ارائه شده که محدودیت‌های TPM را با استفاده از نرم‌افزار بر طرف می‌کند.

[۱۷] راهکاری برای توسعه‌ی امنیت در محیط‌های توزیع‌شده با استفاده از TPM فراهم کرده است. در [۳۶] یک معماری انعطاف‌پذیر برای محاسبات مطمئن ارائه شده که اجازه می‌دهد برنامه‌های با نیازمندی‌های امنیتی مختلف به طور همزمان روی سخت‌افزار معمولی اجرا شوند.

روش ارائه شده در[۳۲] یک سیستم سنجش جامعیت برای لینوکس است که با تغییر هسته‌ی لینوکس و با استفاده از TPM جامعیت سیستم و برنامه‌ها را تضمین می‌کند. در[۲۶]راهکاری برای برطرف کردن محدودیت‌های ابر عمومی با کمک محاسبات مطمئن ارائه شده است، تا بتوان از آن برای کاربردهای حساس به امنیت هم استفاده کرد. در [۹] با ترکیب محاسبات ابری با محاسبات مطمئن، پلت‌فرم ابری مطمئنی برای لایه زیرساخت فراهم شده است.

با قدری تامل در روش‌های ذکر شده متوجه می‌شویم نرم‌افزار به تنهایی نمی‌تواند سیستم را در مقابل حملات نرم‌افزاری حفاظت کند؛ پس باید ترکیبی از سخت‌افزار و نرم‌افزار را برای مقاوم کردن سیستم در مقابل حملات به کار ببریم. در راهکاری که ما در این مقاله ارائه داده‌ایم علاوه بر اینکه سیستم نظارتی در دامنه ممتاز مستقر شده و از نظر ماشین مجازی مهمان شفاف است، برای حفظ جامعیت کد و جامعیت اجرای سیستم امنیتی نیز از سخت‌افزارهای محاسبات مطمئن کمک گرفته‌ایم.

۴. معماری ارائه شده

روش ارائه شده بدین صورت است که برای اجرای قطعه کد حساس به امنیت در ماشین‌های مجازی، از دستور SKINIT که یک دستور امنیتی مربوط به پردازنده‌های جدید AMD است، استفاده می‌کند. این دستور راهکارهایی دارد که حتی در شرایط آلوده بودن سیستم تضمین می‌کند که]جامعیت کد و حافظه مربوط به آن در زمان اجرا حفظ خواهد شد. ز آنجایی که این دستور یک دستور ممتاز است و باید در حلقه‌ی صفر فراخوانی شود و برنامه‌ی کاربردی در حلقه‌ی ۳ اجرا می‌شود، برای حل

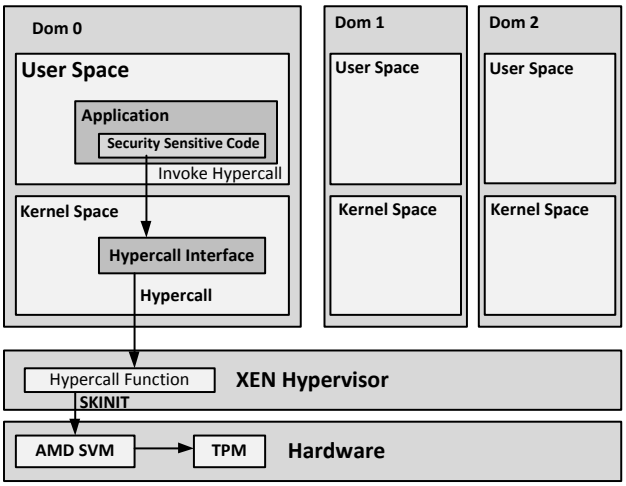
این مشکل از فراخوان فوق‌ناظر استفاده می‌کنیم. در واقع، برنامه‌ی کاربردی با صدا زدن فراخوان فوق‌ناظر از فوق‌ناظر یک سرویس ممتاز را در خواست می‌کند که در اینجا این سرویس ممتاز دستور SKINIT است.

زمانی که دستور SKINIT صدا زده می‌شود، پردازنده به حالت حفاظت شده و مسطح ۳۲ بیتی با صفحه‌بندی غیرفعال می‌رود و در این حالت حفاظت‌های سخت‌افزاری را فعال کرده و اجرای قطعه کد را آغاز می‌کند. این حفاظت‌های سخت‌افزاری شامل غیرفعال کردن وقفه‌ها و دسترسی‌های DMA به حافظه‌ی مربوط به برنامه و همچنین غیرفعال کردن خطایاب‌ها حتی خطایاب‌های سخت‌افزاری می‌شود. سپس دستور SKINIT قطعه کد را برای سنجش جامعیت به چیپ سخت‌افزاری TPM ارسال می‌کند. بدین ترتیب با استفاده از این دو قطعه‌ی سخت‌افزاری جامعیت قطعه کد و جامعیت در زمان اجرا تضمین می‌گردد.

در مجازی‌سازی، از دید سیستم‌عامل بزرگترین تفاوت این است که به جای حلقه‌ی صفر باید در حلقه‌ی ۱ و روی سیستم Xen اجرا شود. این بدان معنا است که دیگر هیچ دستورالعمل ممتازی را نمی‌تواند انجام دهد. به منظور فراهم‌کردن کارکردی مشابه، فوق‌ناظر مجموعه‌ای از فراخوانی‌های فوق‌ناظر را ارائه داده است که متناظر با این دستورالعمل‌های ممتاز هستند. در این حالت هر فراخوانی که قبلاً برای عمل کردن به حلقه صفر نیاز داشته، با فراخوانی‌هایی به Xen جایگزین شده است[۱۰].

در نتیجه، برای دسترسی به حلقه‌ی صفر باید یک فراخوان فوق‌ناظر به کد فوق‌ناظر Xen اضافه کرده و کد آن را کامپایل نمائیم. در این حالت کد مربوط به اجرای دستور SKINIT در تابع مربوط به فراخوان فوق‌ناظر و در حلقه‌ی صفر اجرا می‌شود. جزئیات معماری ارائه شده روی سیستم مجازی‌سازی Xen با یک دامنه‌ی ممتاز به نام دامنه‌ی صفر و دو دامنه‌ی غیر ممتاز به نام دامنه‌های ۱ و ۲ در شکل ۲ آمده است.

همان‌طور که در شکل ۲ مشاهده می‌کنید قطعه‌ی کد حساس به امنیت در دامنه‌ی صفر فراخوان را صدا می‌زند. سپس تابع مربوط به فراخوان که در کد Xen اضافه شده است تابع امنیتی SKINIT را صدا می‌زند. به این ترتیب، قطعه کد حساس به امنیت به صورت امن و بدون نیاز به اطمینان به امنیت فوق ناظر، سیستم‌عامل دامنه صفر و برنامه‌ی کاربردی اجرا می‌گردد. دستور SKINIT قبل از اجرای قطعه کد مورد نظر کد را برای تضمین جامعیت به چیپ TPM ارسال می‌کند. این چیپ بررسی می‌کند که کد نسبت به وضعیت اولیه خود تغییری نکرده باشد. بدین ترتیب، اگر نفوذی به سیستم صورت گرفته باشد و نفوذگر کد امنیتی ما را طبق خواست خود تغییر داده باشد چیپ TPM این تغییر را تشخیص داده و اطلاع می‌دهد.



شکل ۲: معماری امنیتی ارائه شده

۵. ارائه یک کاربرد مناسب

یکی از روش‌ها برای توسعه سیستم‌های امنیتی مثل Tripwire انتقال بخش نظارتی به دامنه‌ی صفر Xen و نظارت جامعیت فایل‌های ماشین‌های مجازی از طریق دامنه‌ی صفر است. به این ترتیب، این سیستم نظارتی از نظر ماشین‌های مجازی شفاف بوده و اگر نفوذی به ماشین‌های مجازی مهمان صورت گیرد، نفوذگر متوجه حضور سیستم نظارتی نشده و نمی‌تواند آنرا غیرفعال کند. اما از آنجایی که خود دامنه‌ی صفر هم یک هسته‌ی لینوکس است پس امکان وجود باگ در آن غیر ممکن نیست و اگر نفوذی به این دامنه صورت گیرد، دیگر نمی‌توان به نتایج به دست آمده از Tripwire اعتماد کرد چون ممکن است نفوذگر پایگاه داده و یا فایل سیاست‌های Tripwire را تغییر داده باشد

در چنین شرایطی می‌توان از معماری مطرح شده برای تضمین جامعیت خود Tripwire، پایگاه داده و فایل‌های سیاست آن استفاده کرد. برای اولین بار و در شرایطی که سیستم امن است می‌توان برنامه را با استفاده از دستور SKINIT اجرا کرده و مقدار Hash کد، پایگاه داده و فایل سیاست‌گذاری را که توسط چیپ TPM تولید شده و در PCR 17 (یکی از ثبات‌های پویای چیپ TPMکه برای ذخیره‌سازی مقادیر Hash به کار می‌رود) ذخیره می‌شود، در محل مشخصی از سیستم ذخیره کرد. در فراخوانی‌های بعدی برنامه که برای سنجش جامعیت دامنه‌ی صفر و یا ماشین‌های مجازی به کار می‌رود، زمانی که دستور SKINIT اجرا می‌شود Hash برنامه، پایگاه داده و فایل

سیاست‌گذاری مجدداً حساب شده و در PCR 17 ذخیره می‌گردد. حال می‌توان مقدار PCR 17 را با مقدار ذخیره شده‌ی قبلی مقایسه کرد تا بتوان تصمیم گرفت که آیا می‌توان به نتایج حاصل شده از کار Tripwire اعتماد کرد یا نه.

Tripwire یک سیستم نظارت بر جامعیت فایل دوره‌ای است و هر زمان که مدیر سیستم احساس نیاز کند و یا کاربر ماشین مجازی این درخواست را داشته باشد می‌توان این برنامه را اجرا کرده و جامعیت فایل‌های ماشین‌های مجازی را سنجید. از آنجایی که کنترل جامعیت در Tripwire به صورت دوره‌ای است به نظر می‌رسد سربار ناشی از اجرای این برنامه با معماری مطرح شده در قبال تضمین امنیت توجیه‌پذیر بوده و بتوان از این معماری برای اجرای امن Tripwire و یا هر سیستم نظارت بر جامعیت فایل که به صورت دوره‌ای عمل می‌کند، استفاده کرد.

۶. مزایای روش پیشنهادی

استفاده از دستور امنیتی SKINIT و چیپ TPM برای اجرای کدهای حساس به امنیت این اطمینان را به ما می‌دهد که حتی در شرایط آلوده شدن سیستم، قطعه کد ما به صورت امن اجرا می‌شود. در واقع، این دستور جامعیت کد و جامعیت در زمان اجرا را تضمین می‌کند. با در نظر گرفتن شرایطی، از این روش می‌توان در دامنه‌های غیرممتاز و دامنه ممتاز استفاده کرد.

از روش ذکر شده برای توسعه‌ی امنیت کلیه سیستم‌های امنیتی می‌توان بهره برد. همان‌طور که در بخش کارهای مرتبط ذکر کردیم برای توسعه‌ی سیستم‌های امنیتی با استفاده از مجازی‌سازی، بخش نظارتی سیستم را به دامنه ممتاز منتقل می‌کنند تا برای سیستم تحت نظارت شفاف بوده و نفوذگر متوجه حضور سیستم نظارتی نشود و نتواند آن را غیرفعال کند. با وجود اینکه دامنه‌ی ممتاز یک دامنه امن بوده و نفوذ به آن بسیار مشکل است اما این کار غیرممکن نیست زیرا دامنه‌ی ممتاز نیز یک هسته‌ی لینوکس است و ممکن است باگ‌هایی داشته باشد که نفوذگر از طریق آنها بتواند وارد سیستم شود. راهکار ارائه شده حتی در این شرایط هم می‌تواند قطعه کد حساس به امنیت را بدون دخالت نفوذگر اجرا کند.

با استفاده از این روش حتی اگر نخواهیم بخش نظارتی را به دامنه‌ی ممتاز منتقل کنیم، می‌توان داخل همان سیستم تحت نظارت بخش امنیتی را با نظارت بالایی اجرا کرد. با وجود اینکه چنین سیستمی شفافیت ندارد و نفوذگر متوجه وجود سیستم امنیتی می‌شود، اما نمی‌تواند در اجرای آن اختلال ایجاد کند.

۷. نتیجه گیری و کارهای آتی

از آنجایی که ماشین‌های مجازی در لایه‌ی زیرساخت محاسبات ابری قرار گرفته‌اند تضمین امنیت در این ماشین‌ها می‌تواند گام مؤثری در فراهم کردن اعتماد مشتریان باشد. هدف ما در این تحقیق استفاده از سخت‌افزارهای فناوری محاسبات مطمئن برای ارائه یک روش امنیتی در فوق‌ناظر Xen بود. در این روش با استفاده از دستور امنیتی پردازنده‌های AMD و چیپ TPM جامعیت کد و جامعیت در زمان اجرا را برای قطعه کد حساس به امنیت فراهم کردیم. با استفاده از این معماری کاربر می‌تواند اطمینان حاصل کند که کد او در سیستم هیچ‌گونه تغییر ناخواسته‌ای نکرده و در زمان اجرا نیز هیچ خللی در کار آن ایجاد نمی‌شود و همان نتیجه‌ای را از اجرای کد خود خواهد گرفت که انتظار آن را داشته است.

این معماری صرفاً یک روش امینتی با کارکردی یگانه نیست بلکه روشی است که با استفاده از آن می‌توان کلیه سیستم‌های امنیتی موجود را به گونه‌ای امن کرد که حتی در صورت وجود باگ در فوق ناظر و سیستم‌عامل و نفوذ به آنها، نفوذگر نمی‌تواند کد سیستم امنیتی را به خطر بیاندازد در نتیجه جامعیت کد حفظ می‌شود. به علاوه در زمان اجرای کد نیز حافظه‌ی مربوط به کد از دسترسی‌های غیرمجاز حفظ می‌گردد و همان نتیجه‌ای را از اجرای سیستم می‌گیریم که انتظار آن را داشته‌ایم. طراحی این روش به گونه‌ای است که اگر از آن برای اجرای سیستم‌های امنیتی و نظارتی در دامنه ممتاز استفاده کنیم، نیازی به تغییر در سیستم تحت نظارت نیست و ابزار امنیتی از این نظر شفاف بوده و نفوذگر حتی متوجه حضور سیستم امنیتی نمی‌شود.

علاوه بر طراحی سیستم‌های امنیتی از این روش می‌توان برای اجرای کدهای حساس به امنیت در هر کدام از ماشین‌های مجازی نیز استفاده کرد. در این صورت می‌توانیم به مشتریان این امکان را بدهیم که جامعیت کد و حتی پلت‌فرم ارائه شده را تائید کنند و با خیال راحت کدهای امنیتی خود را در محیط محاسباتی ما اجرا کنند و دیگر نیازی نیست برای امنیت کدها و برنامه‌های خود حتی به امنیت سیستم‌عامل خود تکیه کنند.

از آنجایی که روش ارائه شده تنها یک معماری امنیتی است، می‌توان در آینده از این روش برای پیاده‌سازی یک سیستم تشخیص نفوذ و یا یک سیستم ثبت استفاده کرد. با توجه به ویژگی‌های معماری ذکر شده این سیستم نسبت به حملات نرم‌افزاری مقاوم‌تر خواهد شد.

به دلیل اینکه ریزپردازنده‌ی موجود در TPM فرکانس پایینی دارد اگر حجم عملیاتی که توسط این چیپ می‌خواهیم انجام دهیم زیاد باشد، کارایی سیستم پایین می‌آید که اگر این عملیات را که در داخل چیپ TPM انجام می‌شود روی پردازنده اصلی سیستم که فرکانس آن بسیار بالاتر از ریزپردازنده داخل TPM است انجام دهیم، کارایی بالاتر خواهد رفت.

۸. منابع

[1] A Practical Guide to Trusted Computing,IBM Press CT312, March 16, 2011

[2] A. Joshi, S. T. King, G. W. Dunlap, and P. M. Chen. "Detecting past and present intrusions through vulnerability-specific predicates". In *SOSP '05: Proceedings of the twentiethACM symposium on Operating systems principles*, pages 91–104. ACM Press, New York, NY, USA, 2005.

[3] AMD. Secure Virtual Machine Architecture Reference Manual, May 2005.

[4] AMD64 Architecture Programmer's Manual Volume 2: System Programming, May 2011.

[5] B. D Payne, M. Carbone, M. Sharif, and W. Lee, "Lares: An Architecture for Secure Active Monitoring Using Virtualization," in *IEEE Symposium on Security and Privacy,. SP 2008* (presented at the IEEE Symposium on Security and Privacy, 2008. SP 2008, IEEE, 2008), 233-247, 2008.

[6] B. Kauer. OSLO: Improving the security of Trusted Computing). In the 16th USENIX Security Symposium, pages 6–10, August 2007.

[7] C, Maiero and M. Miculan, "Unobservable Intrusion Detection Based On Call Traces In Paravirtualized Systems," *Architecture* 2007.

[8] C. A. Waldspurger. Memory resource management in vmware esx server. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*,pages 181–194. ACM Press, New York, NY, USA, 2002.

[9] D. Wallom, M. Turilli, A. Martin, A. Raun, G. Taylor, N. Hargreaves and A. McMoran, "myTrustedCloud: Trusted Cloud Infrastructure forSecurity-critical Computation and Data Management", Oxford e-Research Centre, 2011.

[10] D. Williams, and J.Garcia, *Virtualization with Xen*, United States of America: Syngress Publishing, Elsevier, pp. 1-40, 2007.

[11] D.Grawrock. The Intel Safer Computing Initiative. Intel Press, January 2006.
doc_id=117908.

[12] G. W.Dunlap and S. T .King and S. Cinar and M.Basrai and P. M. Chen. "ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay". In Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI), 2002.

[13] Intel Trusted Execution Technology (Intel TXT), Software Development Guide, March 2011.

[14] J. Hai, G. Xiang, D. Zou, F. Zhao, M. Li, and C.Yu, "A guest-transparent file integrity monitoring method in virtualization environment," *Comput. Math. Appl.* 60, no. 2,pp. 256–266, 2010.

[15] J. Hoopes, *Virtualization for Security*, United States of America: Singress Publishing, Elsevier, pp. 1-30. 2009.

[16] J. Kim, I. Kim, and Y. I. Eom, "NOPFIT: File System Integrity Tool for Virtual Machine Using Multi-byte NOP Injection," in *2010 International Conference on Computational Science and Its Applications (ICCSA)* (presented at the 2010 International Conference on Computational Science and Its Applications (ICCSA), IEEE, pp. 335-338, 2010.

[17] J. M. McCune, T. Jaeger, S. Berger, R. C´aceres, and R.Sailer, "Shamon: A System for Distributed Mandatory Access Control," in Computer Security Applications Conference, 2006. ACSAC *'06. 22nd Annual* (presented at the Computer Security Applications Conference, 2006. 22nd Annual, IEEE), 2006.

[18] J. M.Mccune, N. Qu, Y. Li, A. Datta, V. D. Gligor, and A. Perrig, "TrustVisor: Efficient TCB reduction and attestation," In Proceedings of the IEEE Symposium on Security And Privacy , 2010.

[19] J. Sugerman, G. Venkitachalam, and B. Lim. Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor. In *Proceedings of the General Track:2002 USENIX Annual Technical Conference*, pages 1–14. USENIX Association, Berkeley, CA, USA, 2001.

[20] J.M. McCune, B. Parno, A. Perrig, M.K. Reiter and H. Isozaki "An execution infrastructure for TCB minimization" 2007.

[21] K. Borders, X. Zhao, and A. Prakash. "Sting: Detecting evasive malware". In *IEEE Symposium on Security and Privacy*. 2005.

[22] K. J. Higgins (2007). Vm's create potential risks. Technical

[23] M. Laureano,C. Maziero, and E. Jamhour, "Protecting host-based intrusion detectors through virtual machines," *Computer Networks* 51, no. 5, 2007.

[24] N. A. Quynh and Y. Takefuji, "A novel approach for a file-system integrity monitor tool of Xen virtual machine," in *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, ASIACCS '07 (New York, NY, USA: ACM,), 194–202, 2007.

[25] N. A. Quynh and Y. Takefuji, "A Real-time Integrity Monitor for Xen Virtual Machine," in *International conference on Networking and Services,. ICNS '06* (presented at the International conference on Networking and Services, 2006. ICNS '06, IEEE, 90-90, 2006.

[26] N. Santos, K. P Gummadi and R. Rodrigues, "Towards Trusted Cloud Computing," HotCloud'09 Proceedings of the 2009 conference on Hot topics in cloud computing, 2009.

[27] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177. ACM Press,New York, NY, USA, 2003.

[28] P. M. Chen, and Noble, B. D., "When virtual is better than real", In Proceedings of the 2001 Workshop on Hot Topics in Operating Systems (HotOS), 2001.

[29] R. Goldberg, "Survey of virtual machine research". *IEEE Computer Magazine*, 1974.

[30] R. Meushaw and D. Simard. "Nettop: Commercial technology in high assurance applications". *Tech Trend Notes: Preview of Tomorrow's Information Technologies*, 9(4), 2000.

[31] R. Sahita, U. Warrier, and P. Dewan, "Dynamic software application protection," Intel Technology Journal, 2009.

[32] R. Sailer, X. Zhang, T.Jaeger and L.Doorn, "Design and implementation of a TCG-based integrity measurement architecture," IEEE Security Privacy Magazine 8, no. 3, 2004.
report, darkREADING,[online]. Available: http: //www.darkreading.com/document.asp?

[33] S. Berger, R. C´aceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: Virtualizing the trusted platform module. In Proceedings of the 15th USENIX Security Symposium, pages 21–21, Berkeley, CA, USA, 2006.

[34] S. Cabuk, L. Chen, D. Plaquin and M. Yung, "Trusted Integrity Measurement and Reporting for Virtualized Platforms," in Trusted Systems, ed. vol. 6163 Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[35] S.Kinney, Trusted platform module basics: using TPM in embedded systems (Newnes, 2006).

[36] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: a virtual machine-based platform for trusted computing" pp. 193—206, 2003.

[37] T. Garfinkel and M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection," *IN PROC. NETWORK AND DISTRIBUTED SYSTEMS SECURITY SYMPOSIUM*, 2003.

[38] TCG - Trusted Computing Group: trustedcomputinggroup.org

[39] Trusted Platform Module Specification. URL: https://www.trustedcomputinggroup.org/specs.

[40] V. Basili and B. T. Perricone. Software errors and complexity:

[41] X. Zhao, K. Borders, and A. Prakash, "Virtual Machine Security Systems" in *Advances in Computer Science and Engineering*, pp. 339-365, 2006.

[42] X. Zhao, K. Borders, and Atul Prakash, "Towards protecting sensitive files in a compromised system" 2005.