Journal of Electronic & Cyber Defence
Vol 2, No. 3, 2014

Implementing a Novel Malware Detection System in Virtual Machines

H. Shirazi[1], S. M. R. Farshchi[2*]

1- PhD, Malek Ashtar University of Technology
2- Malek Ashtar University of Technology

(Reccive: 2014/04/20, Accept: 2014/09/27)

Abstract

Today, virtual machines play an important role in efficient and effective management of resources. Virtualization is the concept of creating multiple virtual machine guests on a single hardware that allows the system to provide optimal use of resources. Common behavior of malwares in a virtual machines is wide. Sometimes these malwares change the system objects in the first step, and next, influence the host operating system of the virtual machine at the time of completion of the work, and maybe in a final step they do some malicious task. In this paper we provide a secure method for identification, classification and elimination of malwares in a virtual machine. The proposed method which is called, SSM, will firstly attempt to identify high-risk behaviors using behavioral profiles and evaluating changes . The proposed method is then extracted from pre-treatment to categorize malicious groups. Experimental results show that the sample rate of false negatives has sharply declined. The proposed mechanism is based on the actual samples virtualization Xen, with the Linux implementation. Through detailed analysis, and comparison SSM with current commercial anti-malware, SSM has a good performance in the detection and removal of malware, as well as reducing the rate of false- negative samples were found in a virtual machine.

Keywords:

Securing the Virtual Machine, Classification of Malware, Security in Linux, Behavioral Model

*** Corresponding Author Email:** Smr.farshchi@stu-mail.um.ac.ir

مجله علمی- پژوهشی «پدافند الکترونیکی و سایبری»
سال دوم، شماره ۳، پاییز ۱۳۹۳؛ ص ۳۳-۲۳

ارائه یک‌روش‌جدید برای شناسایی بدافزارها در سطح مجازی‌ساز در ماشین‌های‌مجازی

حسین شیرازی[۱]، سید محمدرضا فرشچی[۲*]

۱- مجتمع فناوری اطلاعات و ارتباطات، دانشگاه صنعتی مالک اشتر
۲- دانشکده فرماندهی و کنترل، دانشگاه صنعتی مالک اشتر
(دریافت: ۹۳/۰۱/۳۱، پذیرش: ۹۳/۰۷/۰۵)

چکیده

امروزه ماشین‌های‌مجازی در مدیریت بهینه و اثربخش منابع سیستمی نقش مهمی ایفا می‌کنند. مجازی‌سازی، به مفهوم ایجاد چند ماشین‌مجازی بر روی یک سخت‌افزار مهمان است که امکان استفاده بهینه از منابع‌سیستمی را فراهم می‌نماید. امروزه، با گسترش بدافزارها در ماشین‌های‌مجازی، ضرورت توجه به آسیب‌پذیری‌ها در این حوزه از سیستم‌های میزبان گسترش یافته است. رفتار یک بدافزار در ماشین‌مجازی، تغییر اشیاء سیستمی در گام اول، و در گام دوم، نفوذ به سیستم‌عامل‌میزبان ماشین‌مجازی در زمان اتمام‌کار، انجام فرآیندهای دلخواه به‌عنوان گام نهایی است. این مقاله برای اولین‌بار به ارائه یک روش امن، برای شناسایی، دسته‌بندی و امحاء بدافزارها در ماشین‌مجازی پرداخته است. روش پیشنهادی به‌نام، *SSM*، در مرحله اول با استفاده از پروفایل رفتاری و بررسی تغییرات، اقدام به شناسایی رفتارهای پرخطر می‌نماید. روش پیشنهادی در مرحله بعد، به طبقه‌بندی گروه‌های رفتاری مستخرج از مرحله قبل اقدام می‌نماید. در مرحله آخر، پروفایل دسته‌های سالم شناسایی‌شده و به‌ماشین میزبان منتقل می‌شود. استفاده از جریان‌های اطلاعاتی فرآیندها در ماشین‌مجازی، دقت‌بسیارمطلوبی را برای مکانیزم‌پیشنهادی فراهم کرده‌است. در روش‌پیشنهادی، اولاً برخلاف‌روش‌های کنونی، تنها قسمتی از اطلاعات سیستمی مورد پردازش قرار می‌گیرد. ثانیا، برخلاف کلیه ضد بدافزارهای موجود، بجای بررسی تک‌به‌تک اشیاء سیستمی، گروه‌های تشکیل‌شده توسط طبقه‌بند را بررسی می‌کنیم. بنابراین، سربار بسیار کمی به لایه مجازی‌ساز اعمال می‌شود. نتایج تجربی نشان می‌دهد، با استفاده از مدل رفتاری مضاعف، نرخ نمونه غلط منفی، به‌شدت کاهش پیدا کرده است. در این تحقیق نمونه واقعی مکانیزم پیشنهادی بر روی مجازی‌ساز *Xen*، در لینوکس پیاده‌سازی شده است. با انجام بررسی‌های دقیق، و مقایسه *SSM* با ضد بدافزارهای تجاری‌کنونی، عملکرد بسیار مناسب در تشخیص و حذف بدافزارها و همچنین کاهش نرخ نمونه‌های‌غلط منفی به‌خوبی محرز شده است.

واژه‌های کلیدی: امنیت‌مجازی‌ساز، طبقه‌بندی بدافزار، امنیت در لینوکس، مدل‌رفتاری

۱. مقدمه

یک ماشین‌مجازی در سطح سیستم‌عامل از مزیت‌های مختلفی، ازجمله، هزینه راه‌اندازی/ غیرفعال‌سازی‌اندک [۱]، نیاز به حداقل‌منابع [۲]، و قابلیت تعمیم‌سازی گسترده از دید اشتراک محیط اجرای سیستم‌عامل‌میزبان، و محدودیت تغییر حالات در ماشین مجازی، برخوردار است. علاوه‌بر این، تحمل‌پذیری نسبت به اجرای دستورات‌غیرمجاز یا بروز خطا در ماشین میزبان، از ویژگی‌های عمومی استفاده از ماشین‌مجازی است. بنابراین امکان نسبی نصب نرم‌افزار، بدون نگرانی از آلودگی به بدافزار و تغییرات‌سیستمی برای کاربر فراهم می‌شود. به‌عبارت دیگر، یک ماشین‌مجازی آلوده به بدافزار را به‌راحتی می‌توان تعویض کرد. اگرچه این راه‌حل بدیهی به‌نظر می‌رسد اما نیازمند صرف هزینه مضاعف برای برگرداندن ماشین جدید به وضعیت‌کاری کاربر در ماشین مجازی قبلی خواهد بود. زمانی که کاربر با لیست پروسس‌های‌عادی در حال کار است، تغییرات مختلفی در سطح سیستم‌عامل رخ می‌دهد. از جمله تغییر در سیستم‌فایل، تغییر در رجیستری (این مفهوم در لینوکس با فایل‌های‌تنظیمات بیان‌می‌شود- تا انتهای مقاله و برای فهم بهتر، این دو واژه مترادف فرض می‌شود) و یا تغییر در سرویس‌های سیستمی. یک روش امن ترکیبی (منظور از ترکیب با ماشین‌مجازی)، در لایه مجازی‌ساز، باید قادر به تفکیک فعالیت‌های غیرمجاز بدافزارها در سطح سیستم‌عامل بوده و بتواند کلیه فعالیت‌های‌صحیح کاربر را بازشناسی‌کرده و به سطح سیستم‌عامل میزبان منعکس کند. ارائه یک روش امن‌ترکیبی باید دارای ویژگی‌های متفاوتی باشد که در ادامه به‌عنوان نمونه به برخی از آنها اشاره شده است:

۱- روش امن ترکیبی باید داده‌های مختلف از نرم‌افزارهای کاربردی را پس از ادغام مناسب و با توجه به فیلترکردن پروسس‌های‌سالم به سطح سیستم‌عامل برای ذخیره‌سازی منعکس‌کند (به‌عنوان مثال، وب‌سرورها، پایگاه‌داده‌ها، یا فایل‌های تنظیمات نرم‌افزارهای کاربردی.)
۲- بیشتر برنامه‌های زمان واقعی بر روی ماشین‌مجازی، دو نمونه از برنامه خود را اجرا می‌کنند. یک نسخه از پروسس مربوط به

* رایانامه نویسنده مسئول: Smr.farshchi@stu-mail.um.ac.ir

اعمال تغییرات در سطح سیستم‌عامل میزبان و نسخه دوم جهت اجرای برنامه‌های کاربردی می‌باشد. یک تعادل ارتباطی بین این دو نمونه برقرار بوده و تغییرات در بازه‌های چندین میلی‌ثانیه‌ای به سطح سیستم‌عامل‌میزبان منتقل می‌شود. این همگام‌سازی حالت، توسط بدافزارهای امروزی پشتیبانی نمی‌شوند. بنابراین مکانیزم امن ترکیبی پیشنهادی، باید قابلیت تطبیق سریع عملکرد صحیح از بدافزار را داشته و بتواند به سرعت عمل همگام‌سازی دستورات صحیح اجرا شده را انجام دهد.

۳- یک روش امن ترکیبی، باید صحت و جامعیت داده‌های حاصل از فعالیت‌های کاربر را برقرار کند. به‌عبارت بهتر تمامی فایل‌های ذخیره‌شده، و فایل‌های موقت مورد نیاز به سیستم‌عامل میزبان به‌گونه‌ای بازگشت داده شود که قابلیت استفاده بعدی را داشته باشد.

۴- کاربر ممکن است از برنامه‌هایی که فعالیتی شبیه بدافزار داشته یا امکان آلودگی سیستم‌عامل را فراهم می‌آورد، استفاده کرده‌باشد. روش پیشنهادی امن ترکیبی، برای بازگرداندن فایل‌های‌موقتی این‌گونه فعالیت‌ها باید به‌درستی عمل‌کرده و فایل‌ها را به سطح سیستم‌عامل میزبان جهت ادغام بازگشت دهد.

با توجه به موارد فوق و همچنین با توجه به معماری‌های کنونی ماشین‌مجازی، تشخیص عملکرد صحیح از مخرب در سطح ماشین‌مجازی از اهمیت خاصی برخوردار است. تکنولوژی‌های کنونی ماشین‌مجازی در سطح سیستم‌عامل، FreeBSD در [۳]، Linux-VServer در [۴]، Solaris Zones در [۵]، Open VZ در [۶]، Virtuozzo در [۷]، عملیاتی برای تجمیع امن فعالیت‌ها فراهم نکرده‌اند. Bhyve یا BSD Hypervisor یک hypervisor (مدیر ماشین‌های مجازی) رایگان است که بصورت فعال برای سیستم عامل FreeBSD و استفاده از ویژگی‌های CPUهای پرقدرت امروزی نظیر EPT (Extended Page Table) توسعه یافته است. bhyve در حال حاضر از پردازنده‌های Intel همراه با EPT پشتیبانی می‌کند و پشتیبانی از پردازنده‌های AMD نیز در حال گذراندن مراحل تست است. نوع دیگری از تکنولوژی مجازی‌سازی بر پایه یک کرنل وجود دارد، که در این حالت سیستم‌عامل‌های مجازی کرنل Host را به اشتراک می‌گذارند، و فایل سیستم، حافظه و پروسس‌ها را برای ماشین مجازی‌ها فراهم می‌کنند و همه درخواست‌ها از طریق این کرنل انجام می‌شود. مزیت این نوع سیستم‌مجازی‌سازی سربار نسبتاً کمتر، و فراهم کردن دسترسی مستقیم به root سیستم مجازی است. در این نوع تکنولوژی‌ها به‌دلیل دشواری در ایزوله کردن guestها بعضی از سرویس‌ها به‌صورت کامل در دسترس نخواهد بود. عموماً تکنولوژی‌های مذکور در منابع [۳] تا [۶] در این دسته قرار می‌گیرند.

طبق دانش نویسندگان تا لحظه نگارش این مقاله اولاً، تشخیص فعالیت‌های مجاز و غیرمجاز (بدافزار)، توسط هسته ماشین مجازی فراهم‌نشده و ثانیا تغییرات ادغامی‌حالت‌ها به سیستم‌عامل‌میزبان بدون توجه به فاکتورهای امنیتی ذکر شده فوق انجام می‌گیرد.

برای فراهم‌کردن یک مکانیزم امن ادغام در چارچوب ماشین مجازی در سطح سیستم‌عامل دو مشکل عمده وجود دارد. اول، سربار یک مکانیزم امن ادغام حالت و تأثیر آن بر روی سیستم‌عامل میزبان حائزاهمیت می‌باشد. بنابراین مدل ارائه شده باید کمترین سربار را بر روی ماشین‌مجازی داشته باشد. از طرف دیگر ماشین‌مجازی خود دارای سرباری بر روی سیستم‌عامل میزبان‌است. بنابراین در صورت داشتن پیچیدگی زمانی یا مکانی، مشکل تنزل مطبوع [۸] (افت‌کلی عملکرد سیستم) نیز ممکن است رخ دهد. مشکل دوم اینکه مکانیزم پیشنهادی باید تفکیک صحیحی میان تمامی فعالیت‌های مشکوک یا تغییرحالت‌های ناشی از رفتار فرآیندها (یا بدافزارها) انجام دهد. بدیهی است، شناسایی‌صحیح رفتار کلیه بدافزارها براحتی امکان‌پذیر نخواهد بود. روش‌های کنونی از قبیل بررسی رخدادها، تشخیص نفوذ مبتنی بر میزبان، یا ضدبدافزارها هر دو مشکل فوق را برطرف نمی‌کنند. برخی روش‌ها در حل مشکل دوم نوآوری دارند، ولی سربار زیادی به ماشین‌مجازی تحمیل می‌کنند. مطالعه مروری خوبی برای کلیه روش‌ها در [۹] انجام شده است.

این تحقیق به ارائه یک روش جدید با سربار اندک به نام SSM (ادغام حالت امن، Secure State Merge) پرداخته شده است. این پیاده‌سازی برای لایه مجازی‌ساز در سطح سیستم‌عامل خواهد بود. مدل پیشنهادی با طبقه‌بندی صحیح رفتارها و تغییرحالت‌هایی که توسط بدافزارها انجام شده است، از ادغام آنها با سیستم‌عامل میزبان ممانعت به‌عمل می‌آورد. نتایج آزمایشات در این تحقیق نشان می‌دهد مکانیزم پیشنهادی سطح وسیعی از امنیت را برای فعالیت مناسب ماشین‌مجازی فراهم می‌کند. اگرچه‌روش‌های مبتنی بر ایجاد چندفرآیندی برای غلبه بر مشکل جامعیت منابع مشترک در [۱۰و۱۱] ارائه شده‌است، اما در اینجا از پرداختن به این مشکل‌خودداری شده است.

روش پیشنهادی منطبق بر سه گام خواهد بود. در گام اول، اشیاء سیستمی ماشین‌مجازی به تعدادی گروه دسته‌بندی می‌شوند. این دسته‌بندی بر اساس جریان اطلاعاتی فرآیندها در سطح سیستم‌عامل و برچسب‌زدن آنها خواهد بود. در گام دوم، گروه‌های دسته‌بندی‌شده با رفتار نسبتاً پرخطر براساس یک هسته تشخیص بدافزار برخط، مورد بررسی قرار می‌گیرد. در گام آخر نیز تغییرات دسته‌های سالم با حالت‌های‌سیستم‌عامل میزبان ادغام خواهد شد.

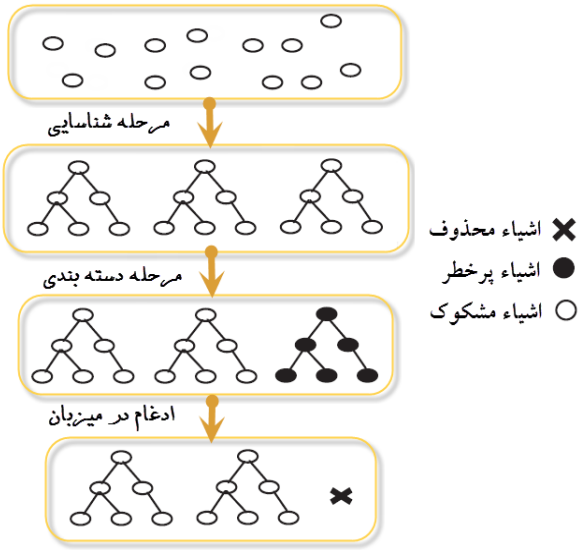
در روش پیشنهادی، SSM، از سه ویژگی‌جدید استفاده شده است. اولین ویژگی عدم توجه به فایل‌های ثبت وقایع (فایل‌های log) به‌دلیل حجیم بودن و سربار زمانی آن‌هاست. در حقیقت از جریان‌اطلاعاتی در سطح سیستم‌عامل و منطق بدافزارها برای دسته‌بندی استفاده شده است. نتایج بررسی‌ها در بخش‌های بعد، سربار اندک روش پیشنهادی نسبت به روش‌های مبتنی بر آنالیز فایل‌های‌ثبت وقایع که عملکرد نسبتا ضیعفی دارند را نشان می‌دهد. ثانیاً، برعکس روش‌های معمول تشخیص نفوذ و بازیابی سیستم [۱۲، ۲۲، ۲۳، ۲۵] که اقدام به بررسی تک‌به‌تک اشیاء سیستمی

می‌کنند، روش پیشنهادی از بررسی دسته به دسته استفاده می‌کند که از لحاظ زمانی کارآیی مضاعفی ایجاد خواهد کرد. و ویژگی آخر اینکه علیرغم توجه روش‌های تشخیص بدافزار مبتنی بر رفتار به مشخصات رفتاری هر بدافزار، در این روش از مدل متفاوتی استفاده شده است. در این الگوریتم از یک زوج رفتار برای دسته‌بندی فرآیندی که سبب تولید این رفتار شده است، استفاده می‌گردد (بدافزار الحاقی یا نرم‌افزاری که سبب اجرای بدافزار شده است).

با ارائه این روش که اولین روش امن ترکیبی برای مقابله با بدافزارهای ماشین‌مجازی در سطح سیستم‌عامل که به‌منظور انتقال فرآیندها به میزبان است چهار مزیت فراهم شده است. اولا استفاده از سه ویژگی‌بالا، سبب ایجاد یک روش سبک و کارا برای انتقال حالت‌ها به سیستم میزبان در صورت آلودگی به بدافزار خواهد شد. ثانیا، یک الگوریتم دسته‌بندی در سطح ماشین‌مجازی ارائه شده‌است. این دسته‌بندی براساس جریان‌اطلاعاتی سطح سیستم‌عامل و ایجاد دسته‌های مختلف برای گروه‌بندی اشیاء بر اساس تغییرات محیطی خواهد بود. ثالثاً، روش مورد استفاده در این تحقیق برای دنبال کردن جریان‌اطلاعاتی، تنها مبتنی بر دنبال‌کردن رفتار فرآیندهای مشکوک، فایل‌های اجرایی (که امکان الحاق بدافزار به آنها وجود دارد) و اشیاء IPC (عناصر ارتباط بین پردازش‌ها) استوار است. این انتخاب سبب کارآیی مؤثر بهبود یافته نسبت به روش‌های‌کنونی تشخیص بدافزار خواهد شد که در سطح سیستم‌عامل شناورند. رابعاً، پیاده‌سازی SSM بر روی Xen و در محیط لینوکس انجام شده است. نتایج نشان داده که بدافزارهای مورد بررسی قرار گرفته در SSM با دقت مناسبی تشخیص داده‌شده و سربار اعمالی بر Xen هم تاثیر چندانی بر عملکرد سیستم نداشته است. علاوه‌بر این نسبت به مقایسه برخی از آنتی‌ویروس‌های تجاری در کشف بدافزارها در Xen، روش SSM توانایی معقول‌تری را به نمایش‌گذاشته است. لازم به یادآوری است پیاده‌سازی SSM، برای سطوح دیگر مجازی‌سازی، نیازمند تغییراتی در کد برنامه خواهد بود، زیرا جزئیات فنی از محیط و نحوه عملکرد سیستم شبیه‌ساز به صورت دستی به SSM اعمال شده است.

۲. ساختار SSM

برای حفظ صحت‌داده‌ها در سیستم‌عامل میزبان و ماشین‌مجازی، باید به قاعده بازنویسی (ذخیره تغییرات)، ماشین‌مجازی توجه کرد. بعد از اتمام فرآیندکاری ماشین‌مجازی، فایل‌ها و اطلاعات رجیستری (در لینوکس فایل‌تنظیمات)، توسط جلسه‌کاری ماشین‌مجازی بازنویسی می‌شود. بنابراین مکانیزم ما، باید از بروزرسانی تغییرات بدافزار جلوگیری کند. لزوم تأکید بر توقف (عادی یا غیرعادی) کامل ماشین‌مجازی، از ساختار سیستم‌فایل و مدیر فرآیند نشات می‌گیرد (چراکه برخی از فرآیندها قابلیت بازنویسی در سیستم‌فایل‌میزبان را تا بازشدن قفل اشتراک فایل، ندارند). فرضیه مدل پیشنهادی، براساس مطالعه دقیق رفتار بدافزارها در گزارش تحلیل بدافزار، توسط سیمانتک در [۱۳]، ارائه شده است. در شکل ۱ مراحل عملکردی روش‌پیشنهادی را به‌نمایش گذاشته‌ایم.



شکل ۱. مراحل سه‌گانه روش SSM در حالت‌کلی

۲. ۱. دسته‌بندی‌اشیاء مشکوک‌سیستمی

چنان که گفته شد، یکی از نوآوری‌های پژوهش پیشنهادی، بررسی اشیاء سیستمی به‌صورت گروهی است. سیستم‌های تشخیص بدافزار سنتی، از بررسی‌های منفرد بر روی هریک از اشیاء، استفاده می‌کنند. این خصوصیت سبب ایجاد سربارزمانی، تنزل‌مطبوع و مصرف‌حافظه خواهد شد. با استفاده از ویژگی بررسی‌گروهی، اثرات تخریبی بدافزارها با دقت بیشتری شناسایی خواهد شد. چراکه کلیه رفتار بدافزار، بر روی ماشین‌مجازی شناسایی شده و در مرحله نهایی الگوریتم، از بین خواهند رفت. در حالی که در بررسی منفرد در روش‌های‌سنتی، تنها عملکردهای سیستمی مورد بررسی قرار می‌گیرند، و امکان عدم‌تشخیص کارکردکلی بدافزار وجود دارد.

از آنجا که کلیه بدافزارها، از طریق شبکه، یا هارد درایو خارجی (دستگاه‌های جداشونده)، منتقل می‌شوند، ما نقطه شروع بررسی را به دو قسمت تقسیم کرده‌ایم:

- فرآیندهای‌منتج به فراخوانی ارتباطی تحت شبکه یا،
- فایل‌های اجرایی (مثل توسعه EXE)، بر روی دستگاه‌های جداشونده.

منظور ما از بررسی فایل اجرایی، کلیه اشیاء با توسعه‌هایی است که یک فرآیند بتواند در آن حالت اجرایی به‌خود گرفته، یا بتواند، ماکروها را اجرا کند. برای دسته اول، BAT.، VBS، JS.، DLL. و برای دسته‌دوم می‌توان، DOC.، PPT.، DOT. را نام برد. همچنین، SSM اجازه تغییر توسعه فایل، توسط فرآیند اجرا شده را نخواهد داد. با استفاده از این دو قانون، هر کد اجرایی‌مخرب، از لحظه ورود به سیستم تحت بررسی قرار خواهد گرفت. در حقیقت، مشکل اصلی تضمین صحت عملکردی کل سیستم، از این طریق خواهد بود. به‌عبارت دیگر، ما معتقدیم، کل اشیاء سیستم، نباید مورد پیگیری قرار بگیرند. بنابراین باید یک مصالحه بین تعداد اشیاء مورد بررسی، و میزان ریسک آلودگی به بدافزار، ایجاد کرد. این مصالحه براساس

تمرکز بر روی اشیاء سیستمی که وظیفه تکثیر بدافزار را برعهده دارند، برقرار خواهد شد. بدین منظور،

- فایل‌ها، و مداخل رجیستری که توسط بدافزار یا فرآیند مشکوک ایجاد، دستکاری یا مورد دسترسی قرار گرفته، یا،
- فرآیندی که توسط یک فرآیند مشکوک در حال گسترش بوده، یا،
- فرآیندی که یک فایل‌مشکوک را فراخوانی کرده یا در حال اجرای یک فرآیند (منفعل) بوده،

را به‌عنوان یک رفتار پرخطر مورد بررسی قرار می‌دهیم.

در حقیقت، قاعده اول، تمامی رفتارهای‌ثابت در ماشین‌مجازی را مورد بررسی قرار می‌دهد. قاعده دوم و سوم، بر روی فرآیندهای درحال اجرا (فعال) و یا اجرای یک فرآیند منفعل (منظور برنامه ذخیره‌شده‌ای که آماده به اجراست) تمرکز دارد. با این‌قوانین، از دنبال‌کردن فعالیت‌هایی از قبیل خواندن فایل‌ها، پوشه‌ها یا بسیاری از فراخوانی‌های سیستمی اجتناب کرده‌ایم. در واقع این فعالیت‌ها، همان جریان‌های‌اطلاعاتی با خطرپذیری ناچیز می‌باشند. بنابراین تمامی فعالیت‌های رفتاری یک بدافزار طبق قاعده اول و سوم قطعاً منجر به یک ردیابی خواهد شد. چنان که در گزارش سیمانتک نیز ذکر شده است، در منطق بدافزارها، اجرای فرامین از فراخوانی یک فایل اجرایی شروع خواهد شد. هرچند، بدافزار ممکن است به فایل‌های سالم ملحق شده و فرامین خود را اجرا کند، که در قاعده دوم یا سوم، مورد بررسی SSM قرار خواهد گرفت.

قوانین پیشنهادی نسبتاً کامل است، اما تنها نکته باقیمانده، فرآیندهای نسبتا زیادی است که از IPC استفاده می‌کنند. ما تعاملات بین فرآیندها را در SSM به‌دلیل سربارسنگین و استفاده گسترده فرآیندها از این توابع، بررسی نمی‌کنیم، اما تضمین‌می‌کنیم که بدافزارها از این طریق با قواعد سه‌گانه امکان نفوذ ندارند. اولاً، حملاتی که از طریق IPC انجام می‌شود اندک بوده و من‌جمله، می‌توان، حمله DoS را مثال زد. ویژگی IPC، این است که بدافزارها امکان انتشار از طریق شبکه را بر روی ماشین‌مجازی یافته، ولی مجددا باید از طریق یک فایل اجرایی، منتشر شوند که در قاعده اول مورد بررسی SSM قرار خواهد گرفت. بنابراین امکان انتقال از طریق IPC و آلودگی ماشین‌مجازی به بدافزار با ویژگی‌های پیشنهادی، قطعاً منتفی خواهد بود. ایده انتقال بدافزارها از طریق IPC را، در معماری امنیت داخلی سیستم‌عامل ویندوز ۷ در [۱۴]، می‌توان مشاهده کرد.

۲. ۲. برچسب‌گذاری اشیاء پرخطر

در این بخش ما از یک گراف وابستگی، برای انتساب گروه به اشیاء پرخطر (اشیاء مشکوک در مرحله بعد)، استفاده می‌کنیم. در حقیقت برای هر یک از اشیاء پرخطر که منجر به تولید اشیاء سلسله‌مراتبی می‌شوند، یک گراف وابستگی تشکیل خواهیم داد. این گراف وابستگی با تکامل جریان اطلاعاتی تکمیل می‌شود. این گراف، جهت‌دار بوده و دارای نقطه شروعی، به‌عنوان ریشه خواهد بود. یال‌ها در این گراف همان جریان‌اطلاعاتی بوده که تحت عملیاتی خاص، بوقوع پیوسته‌اند. شکل ۲(الف) و (ج)، دو گراف وابستگی برای فرآیند شبکه، و اجرای یک فایل را به‌نمایش گذاشته است. نکته اصلی اینجاست که عملاً، از ایجاد گراف وابستگی استفاده نمی‌شود، زیرا روش SSM، برخط می‌باشد. در روش پیشنهادی هریک از اشیائی که پرخطر باشد، منقوش به یک برچسب خواهد شد، که دارای فیلدهای مختلفی است (مثل، شماره و زمان). این فیلدها از تطابق قاعده متناظر هریک از اشیاء سیستمی مورد بررسی و محاسبه قرار خواهند گرفت. به‌صورت متوالی با این برچسب کلیه لیست‌های مجاورتی ماشین‌مجازی، مورد بررسی قرار می‌گیرند. به‌عبارت دیگر ما از یک ساختار تشکیل گراف و سپس آنالیز آن پیروی نمی‌کنیم. اگر یک فرآیند تحت شبکه، به‌صورت پرخطر شناسایی شود، نباید تمامی اشیاء داخل آن را به‌صورت بدافزار تقسیم‌بندی کرد. ممکن است یک فرآیند دارای اشیاء زیادی باشد، که در دسته بدافزار طبقه‌بندی شده باشند، اما دارای اشیاءسالم نیز باشد. بنابراین ایده تقسیم هر گراف به زیرگراف انجام داده‌خواهدشد، تا هر گروه دقیقاً شامل اشیاء سالم یا پرخطر باشد.

طبق بررسی‌های ما بر روی بیش از هزاران بدافزار و مطالعه تحلیل‌گر بدافزار سیمانتک، کلیه روش‌های‌انتقال‌بدافزار به میزبان را از دید کانال انتقال می‌توان در سه گروه دسته‌بندی‌کرد. دسته اول، انتقال از طریق واسط شبکه یا اجرای فرآیندی با مجوز دسترسی به منابع، یا استفاده از باگ نرم‌افزارهای کاربردی، برای ورود، سپس گسترش یا ایجاد اسکریپت‌های‌دیگر برای نفوذ کامل در میزبان. فراخوانی‌های حاصل از این فرآیندها در گروه *انشعاب* طبقه‌بندی می‌شوند (گروه انشعاب B، در شکل ۲ (ب)). دسته‌دوم، از طریق تکنیک‌های مهندسی اجتماعی بوده و بدافزار مربوطه باید در میزبان، مقیم حافظه باشد. این گروه و تمامی فرآیندهای حاصل از آن در گروه *انشعاب* طبقه‌بندی خواهند شد (گروه انشعاب A، در شکل ۲ (ب)). دسته سوم، از طریق فایل اجرایی بر روی دستگاه‌های جداشونده خواهد بود که در گروه *دستگاه* طبقه‌بندی‌می‌شوند. (شکل ۲ (د)).

مشکل دیگری که در گراف وابستگی وجود دارد [۲۳، ۲۴]، وجود فرزندها با بیش از یک والد است (فرآیندهای تولیدی مشترک از نرم‌افزار سالم و بدافزار در حالت‌های متفاوت در چندین مرحله). ادغام این فرزندها براحتی امکان‌پذیر نخواهد بود (مثل A به C در شکل ۲ (الف)). به‌عبارت دیگر زمانی‌که والدین یک گره به گروه‌های مختلف تعلق دارند، تعیین دسته فرزند، از اهمیت خاصی برخوردار است. برای حل این مشکل در SSM، از اولویت اجرا استفاده می‌کنیم. در حقیقت ترتیب زمانی از، فایل برنامه اجرای فرآیند پدر تولید اشیاء خواهد بود. بنابراین فرزند ادغامی به‌صورت بدیهی از والد خود یا دسته متعلق به فایل برنامه ارث‌بری خواهد داشت. از طرفی اجرای فرآیند از بخش کد فایل منفعل برنامه، نسبت به بخش کد فرآیند در

حال اجرا، دارای اولویت است. لذا اگر بدافزار اقدام به تولید نمونه فرآیندی بکند، برچسب‌خود را از فایل برنامه اخذ خواهد کرد. از طرف دیگر اگر بیش از یک والد دارای اولویت زمانی یکسانی باشند، به‌صورت بدیهی، فرزند دارای برچسبی برعکس ترتیب زمانی خواهد بود (ترتیب ایجاد فرآیند). به‌عنوان مثال فرزندان A تا C در گروه‌های *انشعاب* ۱ و ۲ شکل ۲ (ب) قرار گرفته‌اند.

در نهایت نیز گروه‌های‌تشکیل‌شده، متشکل از یک دسته از اشیاء (سالم یا پرخطر) خواهد بود، و امکان وجود اشیاء از دو گروه در یک طبقه امکان‌پذیر نخواهد بود.

اثبات. یک گروه را با $c=(V,E)$ نمایش می‌دهیم که V مجموعه رئوس و E مجموعه یال‌های جهت‌دار برای اتصال رئوس خواهد بود. رئوس در این مجموعه بیانگر اشیاء سیستمی خواهد بود. از نماد، v_r برای اشاره به گره ریشه گروه، نماد c، استفاده شده است. یال (v_{n-1},v_n) بیانگر جریان اطلاعاتی عملیات مورد نظر از شروع والد گره v_{n-1}، تا گره فرزند v_n، می‌باشد. برای تناظر سالم یا پرخطر بودن اشیاء از نماد، پرخطر|سالم $= v.f$، استفاده می‌شود که:

۱- اگر ریشه گروه c، پرخطر باشد، تمامی اشیاء آن‌گروه پرخطر خواهد بود. ازآنجا که ریشه آن گراف تولیدکننده تمامی فرزندان در آن گراف خواهد بود. پس برای هر رأس دلخواه، v_n در گروه، حداقل یک مسیر انتشار بدافزار به‌صورت $\{(v_r,v_1),(v_1,v_2),(v_2,v_3),\dots,(v_{n-1},v_n)\}$ وجود خواهد داشت که طبق توضیحات بخش قبل هریک از اشیاء پرخطر، با اجرای یکی از قواعد سه‌گانه می‌تواند بدافزار را به سطح ماشین‌مجازی منتقل کند، پس:

$$(v_{m-1},v_m)\in E \wedge v_{m-1}.f = \text{پرخطر} \rightarrow v_m.f = \text{پرخطر}$$

در این صورت اگر گره ریشه پرخطر باشد داریم،

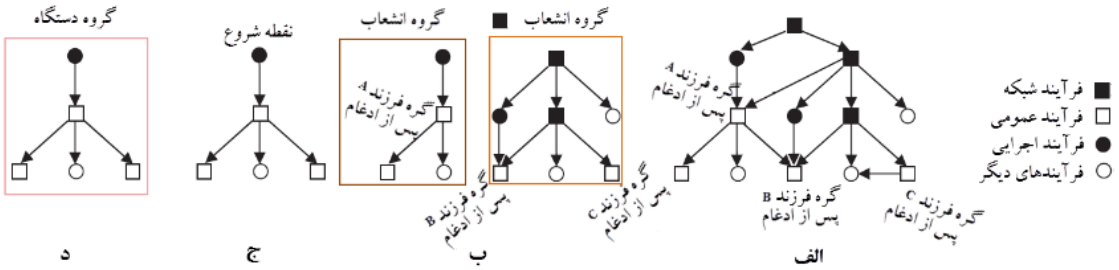
$$v_r.f = \text{پرخطر} \wedge (v_r,v_1)\in E \rightarrow v_1.f = \text{پرخطر}$$
$$v_r.f = \text{پرخطر} \wedge (v_1,v_2)\in E \rightarrow v_2.f = \text{پرخطر}$$
$$v_{n-1}.f = \text{پرخطر} \wedge (v_{n-1},v_n)\in E \rightarrow v_n.f = \text{پرخطر}$$

بنابراین، تمامی فرزندان گروه باید پرخطر باشند.

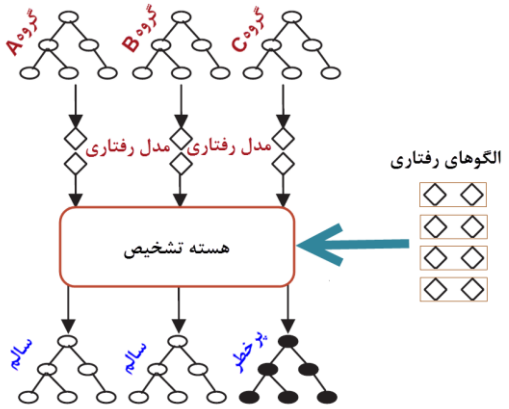
۲- اگر هر یک از اشیاء به‌صورت $v_r \neq v_n$ در یک گروه پرخطر باشد، ریشه آن‌گره، v_r، باید پرخطر بوده و یک مسیر انتقال بدافزار به گره فرزند، v_n، نیز باید موجود باشد. چرا که تنها منبع ایجاد بدافزار در یک گروه، ریشه آن‌گروه خواهد بود. بنابراین نتیجه می‌گیریم، که اگر اشیاء داخل یک گروه پرخطر باشد، ریشه آن‌گروه نیز پرخطر خواهد بود. لذا تنها حالت ایجاد چنین گروهی، دسته‌بندی منفرد گروه به اشیاء سالم یا پرخطر خواهد بود و امکان دسته‌بندی ترکیبی وجود نخواهد داشت.

۲. ۳. تشخیص گروه‌های پرخطر

در این تحقیق برای تشخیص بدافزارها (اشیاء با رفتارهای پرخطر)، اقدام به ایجاد یک هسته برخط، کرده‌ایم. روش‌های‌کنونی در تشخیص بدافزارها [۱۵، ۱۶ و ۱۷]، با استفاده از جریان اطلاعات پویا، به استخراج ویژگی و دنبال‌کردن فرآیندها می‌پردازند. این روش‌ها سربار بسیار زیادی را به‌سیستم اعمال می‌کنند، و برای عملیات برخط مناسب نیستند. روش‌های سنتی مبتنی بر تشخیص رشته بایت‌ها (تشخیص امضاء) نیز به‌دلیل دقت تشخیص پایین و ناتوانی عملکرد در مقابله با بدافزارهای جدید به‌تنهایی قابل استفاده نمی‌باشند. دسته دیگری از ضد بدافزارها از بررسی رفتار تک‌به‌تک بدافزارها استفاده می‌کنند. این روش‌ها تولید نمونه‌منفی‌غلط، نسبتاً زیادی دارند و بنابراین قابل استفاده در SSM، نمی‌باشند. در روش پیشنهادی برخلاف روش‌های ارائه‌شده کنونی از یک الگوریتم برخط استفاده شده‌است، که بجای بررسی تک‌به‌تک فرآیندها، گروه‌ها را موردپردازش قرار می‌دهد. اگر گروهی شامل حداقل دورفتار مشابه در اشیاء خود و مطابق الگوی‌رفتاری تعریف‌شده باشد، پرخطر نامیده می‌شود (شکل ۳). الگوی‌رفتاری استفاده‌شده در این تحقیق، از فراخوانی‌های توابع API، (واسط برنامه‌نویسی کاربر)، و دنبال‌کردن جریان اطلاعاتی سطح سیستم‌عامل در ماشین‌مجازی تشکیل شده است. برخلاف بهترین روش‌های‌کنونی، مبنی بر دنبال‌کردن جریان داده‌ای، در این پژوهش از جریان اطلاعاتی سطح سیستم‌عامل استفاده شده است. طبق نتایج ارائه شده، این استفاده منتج به بهینگی پیاده‌سازی SSM شده است و به‌طور همزمان، دقت تشخیص نیز نسبتاً بهبود یافته است.



شکل ۲. گراف وابستگی و گروه‌های تشکیل شده در مثال نمونه‌ای



شکل ۳. هسته‌تشخیص بدافزار، ورودی‌مدل‌های رفتاری به سیستم و دسته‌بندی گروه‌ها

فرضیه استفاده از حداقل دو رفتار مشابه، کاملاً بررسی شده است. اگرچه در این تحقیق سعی در افزایش بررسی رفتارهای اشیاء مشترک شده، اما تعادل بین نمونه‌های منفی‌غلط و منفی مثبت ثابت باقی نمی‌ماند، و بهترین انتخاب برای سطح ماشین‌مجازی با انتخاب دو رفتار پایدار می‌شد. نکته مهم دیگر توجه به تعداد نمونه‌های رفتاری تعریف شده است. اگر تعداد نمونه‌ها زیاد باشد، عملکرد کلی SSM، تحت تأثیر قرار می‌گیرد. در طرف مقابل اگر تعداد نمونه‌ها کافی نباشد، عملکرد SSM در یافتن اشیاء پرخطر تحت تأثیر قرار خواهد گرفت، لذا توجه به تعداد نمونه‌ها از اهمیت برخوردار است.

ما چهار نمونه رفتار را برای بررسی انتخاب کرده‌ایم:

۱- فرآیندی که روند خود راه‌اندازی را در زمان شروع‌به‌کار ماشین‌مجازی فعال کند.

۲- فرآیندی که گسترش فرآیند در سیستم‌عامل را در دستور کار خود قرار دهد.

۳- فرآیندی که ویژگی‌های مخفی‌سازی خودرا فعال‌نماید، و،

۴- فرآیندی که دارای اهداف و فراخوانی‌های پرخطر باشد.

عموم بدافزارها قابلیت‌های فوق را دارا هستند. به‌عنوان مثال بدافزار فاقد قاعده اول، پس از راه‌اندازی مجدد قادر به انجام‌کاری نخواهد بود. بنابراین به‌جای تهیه لیست الگویی بسیار زیاد، از۴ قاعده فوق برای مطابقت گروه‌ها و اشیاء داخل آن استفاده شده‌است. باتوجه به محدودیت صفحات مقاله، جمع‌بندی هسته تشخیص بدافزار با قواعد زیر بیان شده و از بررسی بیشتر قوانین خودداری شده است.

فرض کنید b بیانگر رفتارکنونی مورد استفاده توسط هسته تشخیص SSM باشد، در این صورت، $b.p$ بیانگر فرآیندی خواهدبود که رفتار b را اجرا کرده است. همچنین، $b.t \in \{t_1, t_2, t_3, t_4, t_b\}$، بیانگر نوع رفتار خواهد بود، و t_b بیانگر نوع رفتاری است که در الگوهای تشخیص هسته شناسایی پیشنهادی قرار نمی‌گیرد. یک گروه را به‌صورت $c \in C$ نمایش می‌دهیم که c بیانگر اشیاء سیستمی و C، بیانگر گروه‌های‌تشکیل‌شده خواهد بود. $c.t$ نوع رفتاری است که در اشیاء گروه c، مورد استفاده قرارگرفته‌است. نهایتاً، نماد T پرخطر|سالم $= c.c.f$، بیانگر سالم یا پرخطر بودن گروه c، خواهد بود (دقت کنید، طبق اثبات بالا می‌توان بجای اشاره به اشیاء c، از گروه c (کوچک)، استفاده کرد.) بنابراین:

۱- $\forall c \in C, b.p \notin C \longrightarrow$ اقدامی انجام نده

۲- $\exists c \in C, b.p \in c \wedge c.t =$ نوع رفتاری نداریم $\longrightarrow c.t = b.t$

۳- $\exists c \in C, b.p \in c \wedge b.t \neq c.t \wedge c.t \neq$ نوع رفتاری نداریم $\longrightarrow c.f =$ پرخطر

۴- $\exists c \in C, b.p \in c \wedge b.t = c.t \longrightarrow$ اقدامی انجام نده

۵- $\exists c \in C, b.p \in c \wedge b.t = t_b \longrightarrow c.f$ $\longrightarrow$ اجازه رفتار b داده نشود $\wedge$ پرخطر

قاعده اول اقدامی را درپی ندارد چراکه رفتار فرآیند در هیچ‌یک از گروه‌ها قرار نمی‌گیرد (توسط فرآیند پرخطر ایجاد نشده‌است). قاعده‌دوم، رفتار تشخیص داده شده را درگروه‌خود دسته‌بندی خواهد کرد. قاعده سوم، همان تشخیص وجود دو رفتار متفاوت در یک‌گروه است که قبلاً توضیح داده شد. لذا SSM، گروه را به‌عنوان پرخطر شناسایی خواهدکرد. قاعده چهارم، رفتار یکسان اشیاء داخل یک گروه است، که اقدام خاصی انجام نمی‌دهد. نهایتاً، قاعده پنجم، گروه مذکور را به‌عنوان پرخطرشناسایی‌کرده، اما به‌دلیل وجود رفتاری جدید از سمت فرآیند والد بدافزار، اجازه عملکرد به فرآیند داده نمی‌شود. لازم به‌ذکر است، SSM، تنها به دسته‌بندی و تشخیص دسته‌های پرخطر می‌پردازد و هیچ‌گونه توقف‌پردازشی در این سطح در ماشین‌مجازی صورت نخواهد گرفت. برخی از آنتی‌ویروس‌های تجاری، مثل Avira، پس از تشخیص فرآیند پرخطر بلافاصله اقدام به بلاک‌کردن کل فرآیند می‌کنند که می‌تواند عملکرد و پایداری کل سیستم را دچار مخاطره کند. به‌عنوان مثال، اگر بدافزار داخل یک روال سیستمی نهفته باشد، با بلاک‌کردن فرآیند سیستمی ممکن‌است اطلاعاتی از دست‌رفته یا عملکردسیستم ناپایدار گردد.

۲. ۴. ادغام گروه‌های سالم

همان‌طور که ذکر شد ادغام پس از توقف‌کامل سرویس‌های ماشین‌مجازی انجام خواهد شد. بنابراین ارتباطات IPC، فایل‌ها و تمامی اشیاء بسته‌شده‌اند. و تنها اشیاء ثابت، مثل فایل، رجیستری و... نیاز به ادغام خواهند داشت. الگوریتم پیشنهادی برای ادغام به‌صورت زیر خواهد بود.

در ابتدا تمامی گروه‌های سالم بررسی می‌شود، در گروه‌های سالم، ویژگی خودراه‌اندازی تمامی فرآیندها بررسی می‌شود. عمده بدافزارها عملیات خودراه‌اندازی را تنها برای الحاق به سیستم‌عامل میزبان و ادامه عملیات در آن محیط انجام‌می‌دهند. بنابراین با این بررسی، بدافزارهایی هم که از سیستم شناسایی در امان مانده‌اند، تغییراتشان در ادغام شرکت نخواهد داشت. بدیهی است پس از تشخیص یکی از اشیاء داخل گروه با فرآیند خودراه‌اندازی (قبل از ادغام)، اشیاء دیگر داخل گروه نیز پرخطر به‌حساب می‌آیند. بنابراین در الگوریتم پیشنهادی از ادغام این اشیاء نیز جلوگیری شده است.

بدین‌ترتیب، الگوریتم پیشنهادی به‌صورت زیرخواهد بود.

فرض‌کنید، $c.s$ بیانگر هویت والد گروه و $c.o \in A$، بیانگر اشیائی باشد که فعالیت خودراه‌اندازی رادارا باشند. $o_1 \rhd o_2, o_1 \in A$ بیانگراجرای خودراه‌اندازی o_1، متناظر با فایل اجرایی o_2، خواهد بود. R نیز بیانگر کلیه فرآیندهای موجود در سیستم خواهد بود. نهایتاً از نماد O، نیز برای اشاره به اشیاء ایجاد شده، یاتغییرکرده ماشین‌مجازی برای ادغام، استفاده شده‌است. پس:

۱-

$$\forall c \in C, \left(c.f = \text{سالم}\right) \wedge (c.s \notin \mathrm{R}) \wedge (\exists o \in C, o \in A) \longrightarrow c.f = \text{پرخطر}$$

۲-

$$\forall o_1 \in A, o_1 \in c_1, c_1 \in C, \left(c_1.f = \text{پرخطر}\right) \wedge (\exists o_2 \in c_2, o_1 \rhd o_2, c_2.f = \text{سالم}) \longrightarrow c_2.f = \text{پرخطر}$$

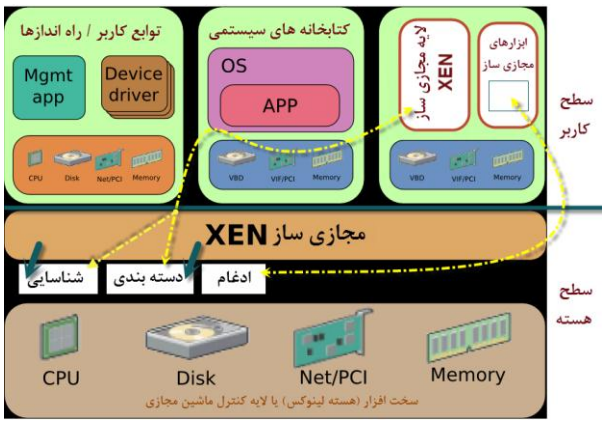
۳-

$$\forall c \in C, \left(c.f = \text{سالم}\right) \wedge \longrightarrow \text{ادغام}\,(c) \forall o \in O, \forall c \in C, o \notin c \longrightarrow \text{ادغام}\,(o) \forall c \in C, \left(c.f = \text{پرخطر}\right) \longrightarrow \text{عدم ادغام}$$

برقرار است. همچنین لازم به‌ذکر است، جامعیت داخلی‌داده‌ها به‌دلیل عدم ادغام گروه‌های پرخطر، همچنان برقرار خواهد بود، چرا که در بخش قبلی نیز توضیح داده شد، تغییرات بدافزارها تأثیری بر جامعیت داده‌ها ندارند و تغییرات آنها در فرآیند سیستمی ماشین‌مجازی اثرگذار نخواهد بود.

۳. پیاده‌سازی SSM

برای نمایش ساختار دقیق مدل‌پیشنهادی از Xen بر روی لینوکس نسخه Redhat، استفاده شده‌است. پیاده‌سازی ادغام‌کننده در لایه مجازی‌ساز Xen صورت گرفته است. این لایه شامل راه‌اندازهای هسته و توابع سطح کاربر می‌باشد. علاوه بر این برای عمل ادغام یک واسط کنترلی در سطح کاربر نیز در نظر گرفته‌ایم. شکل ۴ بیانگر معماری پیاده‌سازی عمومی، SSM، می‌باشد.



شکل ۴. ساختارپیاده‌سازی SSM در لایه‌مجازی‌ساز

در این شکل وظیفه تشخیص اشیاء پرخطر برعهده روال شناسایی‌کننده خواهدبود. در حقیقت با استفاده از دنبال کردن فراخوانی‌های API در لینوکس (ویندوز)، برچسب‌گذاری تمامی اشیاء را با الگوریتم پیشنهادی، به‌راحتی می‌توان انجام داد. بررسی‌های نشان می‌دهد، اکثر فراخوانی‌ها در سطح هسته اتفاق افتاده و به این دلیل امکان عبور بدافزار از الگوریتم پیشنهادی به‌راحتی امکان‌پذیر نخواهد بود. برای اشیاء ثابت مسیر خود ماشین‌مجازی در فراخوانی پیش‌فرض تابع سیستمی، برای ذخیره‌سازی انتخاب و برای اشیاء موقت تصویری آن را به‌صورت فایل‌داده موقتی، ذخیره خواهیم‌کرد.

هسته تشخیص از منطق ذکر شده بالا، برای اعمال مدل رفتاری و مقایسه آن با گروه‌های‌تشکیل شده برای تشخیص گروه‌های پرخطر استفاده می‌کند. عموم بدافزارها از چندین فراخوانی API، برای اقدامات‌خود استفاده می‌کنند. هسته تشخیص نیز از ابتدا با دنبال کردن این توابع، عمل نظارتی خود را انجام می‌دهد. به‌عنوان مثال برای الحاق یک فرآیند در فرآیند دیگر (رفتار عمومی بدافزارها)، در لینوکس توابع مختلفی از جمله vfork()، setsid()، signal() و... به‌صورت متوالی انجام می‌گیرد. بنابراین تنها دنبال‌کردن توابع ابتدایی اگر به‌درستی صورت‌گرفته باشد و امکان دنبال کردن آن در سطح هسته فراهم باشد، برای تشخیص‌کامل کافی خواهد بود.

مرحله ادغام، آخرین روال موجود در شکل ۴ می‌باشد. همان‌طور که گفته شد، ازآنجا که دستور ادغام یا حذف توسط درخواست کاربر صورت می‌گیرد. روال ادغام ابتدا به بررسی تغییرات حالت در ماشین‌مجازی پرداخته و سپس گروه‌های سالم را با شروع درخواست کاربر از مسیر ماشین‌مجازی برداشته و در سیستم‌عامل میزبان ادغام خواهد کرد.

۳. ۱. خودتکثیری فرآیند

اگر شناسایی بدافزارها تنها براساس‌الگوی رفتاری باشد، شناسایی‌گروه‌های پرخطر به‌صورت پویا در بازه‌های زمانی معنایی نخواهد داشت. به‌عبارت دیگر، هسته تشخیص، قادر به شناسایی ویژگی‌های بدافزارهای جدید نخواهد بود. برای پوشش مطلوب این قضیه، از ساختاری شبیه [۱۸]، برای تعریف رفتارهای‌جدید استفاده کرده‌ایم، پس:

رفتار جدید: (عملیات، اشیاء، پارامترها، نوع)

نحوه تعریف رفتار بدافزارهای جدید خواهد بود. منظور از عملیات، عملکرد فراخوانی‌های سیستمی مشابه خواهد بود. به‌عنوان مثال، یک فراخوانی سیستمی ممکن است حاصل چندین فراخوانی توابع API دیگر باشد. یا برعکس، تابع سیستمی خاصی با استفاده از چند تابع API دیگر، قابل استفاده باشد. اشیاء و پارامترها در تعریف فوق، همان رفتار استخراجی از تابع مربوطه بوده، و در نهایت یک رفتار در یکی از انواع (پارامتر آخر تعریف فوق)، مقیم حافظه در میزبان، انتشار فرآیند، یا مخفی‌سازی فرآیند قرار خواهدگرفت. بنابراین در هر فراخوانی از تابع API مربوط با بررسی سه ویژگی اول، رفتار سالم یا پرخطر از فراخوانی مربوطه شناسایی می‌شود. نوع رفتار

بدافزارها نیز در جدولی در ساختار Xen ذخیره گردیده و در فرآیند بوت از SSM، مورد استفاده قرار می‌گیرد. در طول زمان، با افزایش مدل‌های رفتاری بدافزارها، زمان جستجو افزایش خواهد یافت. بنابراین ما از پیشنهاد ساختار Hash، با چهار جدول و تابع Hash زیر استفاده کرده‌ایم:

طول جدول ٪ (طول رشته + (رشته) *ASCII*)
=(جدول و رشته)*Myfunc*

سه جدول اول مورد استفاده، دارای عناصری با طول چهار بیت و ساختاری به‌صورت زیر می‌باشد:

R	**E**	**S**	**S**

اولین بیت بیانگر رکورد *R* خواهد بود، که حاوی رفتار متناظر فرآیند بوده، و رکورد دوم، بیانگر وجود تکرار در رفتار تابع، و دو بیت آخر بیانگر یکی از انواع رفتار توضیح داده شده در بخش قبل است. برای ذخیره‌سازی کلیه اطلاعات چهارگانه فوق نیز از جدول دیگری‌استفاده می‌کنیم. این‌جدول برای استخراج نوع رفتار به‌دست‌آمده از یک بدافزار، مورد استفاده قرار می‌گیرد. تابع ()ASCII برای محاسبه مقدار اسکی رشته ورودی خواهد بود. طول جدول، بیانگر تعداد رفتارهای داخل‌جدول و طول رشته نیز، بیانگر تعداد کاراکترهای رشته ورودی خواهد بود. تنها نکته قابل ذکر از الگوریتم بارگذاری اطلاعات داخل جداول، ترتیب جداول‌خواهد بود. در این ترتیب ابتدا سه جدول اول، و سپس جدول چهارم بارگذاری خواهد شد. به‌عبارت دیگر ابتدا رفتار برای تطبیق به جدول اول ارسال خواهد شد، در صورت عدم وجود این رفتار سراغ جدول دوم رفته و آن‌را مورد پردازش قرار می‌دهیم، تا به جدول آخر برسیم. در این مقاله، تنها به الگوریتم یافتن رفتار اشاره می‌شود (مراجعه به الگوریتم ۱).

۴. شبیه‌سازی و نتایج تجربی

ارزیابی‌های دقیق ما بر روی مکانیزم پیشنهادی SSM، از سه قسمت تشکیل شده است. در بخش اول به بررسی مدل پیشنهادی برای گروه‌بندی بدافزارهای واقعی خواهیم پرداخت. در بخش دوم مقایسه روش پیشنهادی با ضد بدافزارهای تجاری امروزی انجام شده است. نهایتاً در بخش آخر به کارآیی مدل‌پیشنهادی اشاره خواهد شد.

۴. ۱. عملکرد در ادغام صحیح

برای نمایش عملکرد SSM، از ۶۰ بدافزار واقعی در پایگاه PU1 در[۱۹]، استفاده شده است. در این تحقیق سعی شده است تا حد امکان رفتار بدافزارها متفاوت‌تر از آنچه که در پروفایل رفتاری تعریف شده‌است، مورد استفاده قرار بگیرد. همچنین ۵۰ برنامه سالم هم برای آزمایش مورد استفاده قرار گرفت. ما برای ارزیابی از ابزار MONIT در [۲۰]، استفاده‌کرده تا آنالیز فرآیندها، رجیستری (فایل تنظیم مدنظر است)، و عملیات فایل‌ها، را زیر نظر بگیریم. همچنین، از دو کامپیوتر و دو سرور، به‌صورت یک شبکه محلی برای ارزیابی‌ها استفاده شده است.

الگوریتم ۱. هسته تشخیص رفتار با ساختار پیشنهادی

ورودی: *رفتار*، *P* / خروجی: رفتار متناظر عملکرد اشیاء

۱- (جدول ۱، عملیات *P*)*MP=Myfunc*
۲- اگر (جدول ۱)(*MP*).فیلد *R* = غلط)
۲-۱- رفتاری یافت نشده است.
۳- اگر (جدول ۱)(*MP*).فیلد E = غلط)
۳-۱- جدول ۲)(*MP*).فیلد *T* را بازگشت دهید.
۴- (جدول ۳، عملیات P)*MP=Myfunc*
۵- اگر (جدول ۳)(*MP*).فیلد E = غلط)
۵-۱- جدول ۳)(*MP*).فیلد *T* را بازگشت دهید.
۶- برای هر رفتار *Z* درجدول چهارم کارهای زیر را تکرار کن.
۶-۱- اگر (اشیاء *Z* با اشیاء *P* یکسان بود) و
(پارامتر *Z* با پارامتر *P* یکسان بود) و
(عملیات *Z* با عملیات *P* یکسان بود)
۶-۱-۱- نوع *Y* را بازگشت دهید.

سرور اول برای ذخیره‌سازی نمونه‌های بدافزارها انتخاب شده است. وب‌سرور Apache، یک FTP سرور، و یک‌نرم‌افزار هم برای برقراری ارتباط بین سیستم‌ها به‌نام IRC، از جمله نرم‌افزارهای نصب شده بر روی سرور اول می‌باشد. سرور دوم هم به‌طور مشابه، دارای همین نرم‌افزارها بوده و به‌عنوان یک وب‌سرور سالم شروع به‌کار می‌کند. کامپیوترهای معمولی از مرورگر Chrome برای دانلود نمونه‌ها و اتصال به سرورها استفاده می‌کنند. همچنین از توزیع سیستم‌عامل Redhat برای اولین سیستم معمولی استفاده شده است (لازم‌به‌ذکر است، نوع میزبان‌ها و سیستم‌عامل یا استفاده‌از نرم‌افزارهای دیگر تأثیری براجرای مدل پیشنهادی ندارد). برای اعمال کاربردهای حقیقی به مرور صفحات وب آلوده، بارگذاری فایل‌های‌اجرایی بدافزارها، ارسال و دریافت رایانامه، دسترسی به پوشه‌های اشتراکی، انتقال فایل‌های آلوده و اجرای فرآیندها پرداخته شده است. بنابراین، ارزیابی SSM در این شرایط مصداق‌کاملی خواهد داشت، و مکانیزم پیشنهادی به‌دقت تحت بررسی قرار خواهدگرفت. برای تشخیص گروه‌های پرخطر دسته‌بندی‌های رفتاری جدول ۱ در نظر گرفته می‌شود.

برای هر نمونه بدافزار سه آزمایش در نظر گرفته شده است. ابتدا، بدون اجرای SSM، نمونه بدافزار را در یک ماشین‌مجازی جدید اجرا، و در مرحله دوم بدافزار پس از فعال‌سازی SSM، به‌همراه نرم‌افزارهای سالم، در یک ماشین‌مجازی‌جدید به‌منظورادغام، اجرا می‌شود. پس از

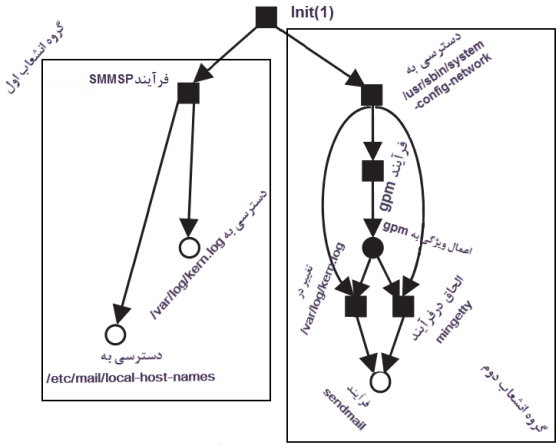
اجرا نام اشیایی که پس از ادغام، حذف شده‌اند، توسط روال ادغام، بررسی گردیده است. در مرحله سوم، نرم‌افزارهای سالم بعد از ادغام، برای نمایش جامعیت داخلی داده‌ها، مورد بررسی قرار گرفته است، تا مشخص‌شود داده‌های‌صحیحی در قالب ادغام، دچار مشکل شده‌اند، یاخیر. نتایج در جدول ۲، به نمایش گذاشته شده است. برای هر نوع از نمونه‌ها، نمونه‌های مورد ارزیابی قرار گرفته، و نمونه‌های غلط‌منفی و غلط‌مثبت، نشان داده شده است. با توجه به نتایج آزمایشات روش پیشنهادی، به‌خوبی تمامی نمونه‌های بدافزارها را به‌عنوان پرخطر دسته‌بندی کرده است، و تنها دو نرم‌افزار سالم را به اشتباه در این دسته گروه‌بندی کرده است. متناظراً در این دو حالت، نرخ نمونه غلط‌منفی و نرخ نمونه غلط‌مثبت برابر صفر و چهار درصد بوده است. علاوه‌بر این، نرم‌افزارهای سالمی که پس از الحاق بدافزار به آنها، ادغام شده‌اند، به‌راحتی در میزبان، به‌فعالیت خود ادامه‌داده و عملکردهای مخرب آنها در روال‌ادغام بی‌تأثیر شده است. نمونه‌های غلط‌مثبت، زمانی که دریافت بدافزارها بدون‌استفاده از یک برنامه کاربردی صحیح، مثل Chrome، (الگوی رفتاری تعریف شده در تحقیق)، بارگذاری شده‌اند، افزایش یافته است. با بررسی فایل‌های‌تغییرات مشخص شد، تغییرات برنامه‌های صحیح، شبیه رفتارهای چهارگانه ذکر شده در جدول بالا بوده است، بنابراین SSM، آنها را به اشتباه گروه‌بندی کرده است. در ادامه این بخش به بررسی جزئیات یکی از بدافزارها به‌نام، Cdorked.A، پرداخته شده است. این بدافزار در خانواده‌ای‌شبیه ویروس شبکه‌ای Blast قرار دارد. شکل ۵، خروجی سطح مجازی‌ساز Xen از نمایش گراف وابستگی فایل‌های تغییرات، پس از اجرای این بدافزار است. ریشه گراف با فرآیند init، آغاز شده است. گروه اول، شامل فرآیند، smmsp بوده و تحت عملکردهای صحیح شبکه‌ای قرار می‌گیرد. در حالی که، گروه دوم، تغییرات در تنظیمات برنامه‌ها، و فعالیت خوداندازی را شامل می‌شود و در دسته پرخطر قرار گرفته است. چرا که طبق جدول رفتاری، دو عملکرد مخرب را دنبال‌کرده است. بنابراین کل اشیاء گروه طبق تعریف بخش ۲، در دسته پرخطر قرارگرفته و ادغام نخواهند شد.

جدول ۱. انواع رفتاری در نظر گرفته شده برای ارزیابی SSM

نوع ۱	تغییر رجیستری برای ایجاد خوداندازی (فایل تنظیمات)	ایجاد یا تغییر سرویس‌ها	ایجاد یا تغییر در راه‌اندازهای دستگاه
نوع ۲	خودتکرارشوندگی	الحاق به فرآیندهای‌دیگر	ایجاد فرآیند به‌صورت غیرعادی
نوع ۳	تغییر در تنظیمات‌امنیتی	غیرفعال کردن نرم‌افزارهای امنیتی	غیرفعال کردن بازگردانی سیستم
نوع ۴	سرقت اطلاعات هویتی		

۴. ۲. مقایسه با بدافزارهای‌تجاری

برای ارزیابی دقیق‌تر مکانیزم پیشنهادی، از ضد بدافزارهای Kaspersky، Avira و Symantec استفاده شده است. از این ضد بدافزارها برای ارزیابی نتایج حاصل از ادغام ماشین‌مجازی، در سیستم‌عامل میزبان استفاده می‌گردد. در آزمایش اول، به بررسی ادغام در میزبان بدون حضور SSM، پرداخته شده است. علی‌رغم به‌روزرسانی کامل ضدبدافزارهای تجاری مورد استفاده، هیچ‌یک از آنها نتوانستند فرآیندهای آلوده به بدافزارها را پس از ادغام شناسایی کنند، و تنها روش پیشنهادی قادر به تشخیص مناسب این بدافزارها بود. در مرحله دوم، تمامی نمونه‌های جدول ۲، با استفاده از روش تشخیص امضاء و روش تشخیص رفتاری به تفکیک مورد بررسی قرار گرفته است. شکل ۶ بیانگر نرخ نمونه غلط مثبت، از اجرای ۵ دسته مختلف از نرم‌افزارهای سالم می‌باشد.



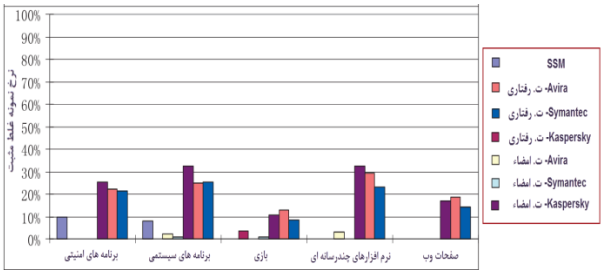
شکل ۵. گراف وابستگی بدافزار Cdorked.A

جدول ۲. انواع نمونه‌های مورد استفاده در آزمایشات

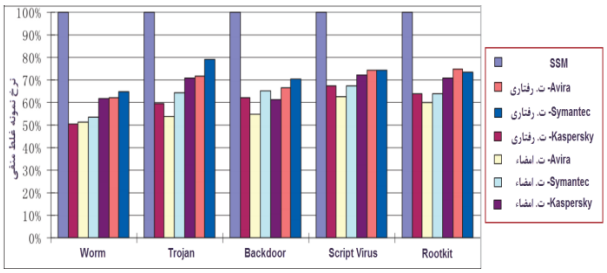
نمونه‌ها		کل	نمونه غلط‌منفی	نمونه غلط‌مثبت
بدافزارها	Worm	۲۰	۰	-
	Trojan	۱۹	۰	-
	Backdoor	۱۷	۰	-
	Script Virus	۲	۰	-
	Rootkit	۲	۰	-
	مجموع	۶۰	۰	-
برنامه‌های سالم	برنامه‌های امنیتی	۱۰	-	۱
	برنامه‌های سیستمی	۹	-	۱
	بازی	۵	-	۰
	نرم‌افزارهای چندرسانه‌ای	۵	-	۰
	صفحات وب	۲۱	-	۰
	مجموع	۵۰	-	۲

با توجه به شکل، نرخ نمونه غلط‌مثبت SSM، کمی بیشتر از روش‌های تشخیص امضاء و کمتر از روش‌تشخیص مبتنی بر رفتار می‌باشد. به‌طور کلی، روش‌های تشخیص رفتار، نرخ نمونه غلط مثبت بیشتری نسبت به روش‌های تشخیص امضاء تولید می‌کنند، اما با

توجه به عدم تشخیص بدافزارهای جدید در روش‌های تشخیص امضاء این نکته قابل توجه نخواهد بود. شکل ۷ بیانگر نرخ نمونه‌منفی، در میان ۵ دسته از بدافزارهای مورد آزمایش است. ابزارهای تجاری، قادر به تشخیص ۵۰ تا ۷۹ درصدی تغییرات پرخطر بوده‌اند (بدون‌درنظرگرفتن روش‌مورداستفاده) اما SSM به‌خوبی تمامی تغییرات پرخطر را شناسایی کرده است.



شکل ۶. نرخ نمونه غلط‌مثبت با استفاده از ابزارهای تجاری و SSM

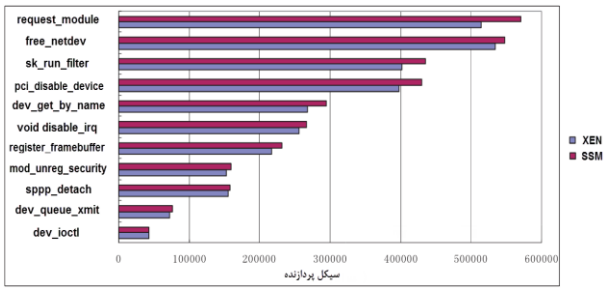


شکل ۷. نرخ نمونه غلط‌منفی با استفاده از ابزارهای تجاری و SSM

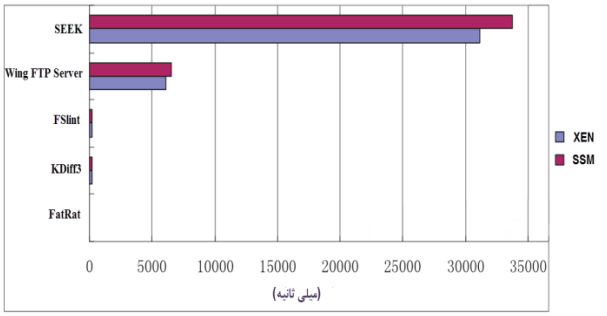
۴. ۳ . سربار ناشی از SSM

در این بخش از سه جنبه مختلف به بررسی سربار سیستم SSM پرداخته شده است. ابتدا به بررسی سربار فراخوانی‌های سیستمی تحت نظارت SSM پرداخته شده، سپس سربار زمانی کل برنامه‌های مستقل، محاسبه گردیده است. درنهایت نیز، تنزل مطبوع خدمات سرورهای در نظر گرفته شده را مورد بررسی قرار می‌دهیم. کامپیوتر اول، از پردازنده Pentium-4 2.8-GHz، با ۱ گیگابایت حافظه رم، برخوردار بوده و شامل نرم‌افزارهای مختلفی از قبیل، SEEK، Wing FTP Server، FSlint، KDiff3 و FatRat بوده است. کامپیوتر دوم هم، از پردازنده Intel Core 2 Duo 2-GHz، با ۲ گیگابایت حافظه رم، برخوردار بوده است. در این تحقیق، SSM برروی هر دو کامپیوتر همراه با توزیع لینوکس Redhat، راه‌اندازی شده است. طبق توضیحات داده شده، سربار SSM در واقع، ناشی از دنبال کردن فراخوانی‌های سیستمی می‌باشد، بنابراین میانگین کلی فراخوانی‌های سیستمی مورد بررسی SSM محاسبه گردیده است. میانگین سیکل پردازنده، برای هر تابع API، در شکل ۸ نمایش داده شده است. دقت در این شکل نشان می‌دهد، SSM حدود ۱٫۹ تا ۱۱٫۳درصد سربار میانگین سیکل پردازنده را به Xen، در حالت کلی فراخوانی‌ها، تحمیل کرده است. از آنجا که SSM تنها قسمتی از فراخوانی‌ها را مورد بررسی قرار می‌دهد، تنزل مطبوع حالت اولیه برای کل سیستم، تغییر چندانی ندارد. برای ارزیابی زمانی برنامه‌ها، ما از اندازه‌گیری زمان میانگین، شروع تا خاتمه برنامه‌ها استفاده کرده‌ایم. شکل ۹ بیانگر اعمال ۰٫۸ تا ۸٫۷ درصدی سربار به میانگین زمانی کلی اجرای برنامه‌ها، با وجود SSM نسبت به عملکرد معمول Xen می‌باشد. نهایتاً به‌منظور ارزیابی خدمات سرویس‌دهنده وب، توانایی عملیاتی Apache در شکل ۱۰ مورد بررسی قرار گرفته است. این بررسی با مقایسه کارآیی سه نمونه از این وب‌سرور در حالت عمومی Xen و وجود SSM انجام شده است. همچنین، از نرم‌افزار WebBench در [۲۱]، برای ارزیابی نتایج در ده تکرار استفاده شده است. در هر آزمایش، حدود ۱ تا ۲۰ درخواست برای وب‌سرور تولید شده است. نتایج نشان می‌دهد، کارآیی با وجود SSM، در حدود ۹۵ درصد کارآیی در حالت Xen بدون وجود مدل پیشنهادی است.

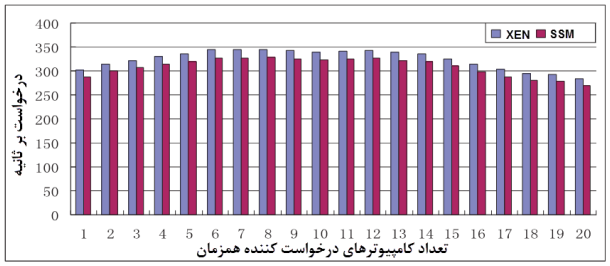
ارزیابی‌های انجام شده نشان می‌دهد، وجود مکانیزم SSM، علاوه‌بر ارائه قدرت تشخیص بالا و مقابله با رفتارهای پرخطر فرآیندها، از سربار بسیار کمی در ماشین‌مجازی و لایه مجازی‌ساز برخوردار بوده است.



شکل ۸. سربار ایجاد شده براساس سیکل پردازنده در فراخوانی‌های توابع حساس در Xen و SSM



شکل ۹. سربار ایجاد شده بر نرم‌افزارها با وجود SSM و حالت عادی Xen



شکل ۱۰. بررسی کیفیت سرویس‌دهنده با اجرای SSM و حالت عمومی Xen

۵. نتیجه‌گیری

در این پژوهش، یک مکانیزم امن، برای جلوگیری از رفتارهای پرخطر بدافزارها در ماشین‌مجازی ارائه شده‌است. ماشین‌مجازی که در سطح سیستم‌عامل پیاده‌سازی شده است، عموماً، پس از اتمام فعالیت فرآیندهایش، تغییرات ثابت و موقت خود را به سیستم‌عامل میزبان منتقل می‌کند، تا قابل استفاده مجدد باشد. با ورود بدافزارها و اعمال‌تغییرات سیستمی در ماشین‌مجازی، سیستم‌عامل میزبان به‌شدت، تحت تأثیر قرار خواهد گرفت. برای اولین‌بار در این پژوهش مکانیزم پیاده‌سازی شده، با ارائه ویژگی‌های‌جدید، قادر به دنبال کردن فعالیت‌های فرآیندها در ماشین‌مجازی خواهد بود. مکانیزم ابداعی، ابتدا با بررسی مدل‌های رفتاری فرآیندها، به تشخیص فرآیندهای والد می‌پردازد. با استفاده از دنبال کردن جریان اطلاعاتی فرآیندها و استفاده از ساختار Hash، پروفایل رفتاری مشابه (حداقل ۲ مورد) مورد شناسایی قرار خواهد گرفت. در مرحله بعد، با استفاده از یک مدل پیشنهادی طبقه‌بندی برای ماشین‌مجازی، گروه‌هایی از اشیاء تشخیص‌داده شده در مرحله قبل، تشکیل می‌شود. در مرحله آخر، قبل از انجام ادغام، پروفایل‌های رفتاری مورد بررسی قرار گرفته و فعالیت‌های سالم به میزبان منتقل‌می‌شوند. روش پیشنهادی با اعمال سربار بسیار ناچیز به ماشین‌مجازی، عملکرد برنامه‌های اجرایی را تحت تأثیر قرار نداده و به‌راحتی امکان ترکیب در سطح مجازی‌ساز را فراهم‌کرده‌است. ارزیابی‌های دقیق نسبت به ابزارهای تجاری موجود، قدرت مکانیزم پیشنهادی را نشان می‌دهد. نتایج نشان می‌دهند، روش پیشنهادی، با اعمال حداکثر ۸٫۷ درصد سربار، تمامی بدافزارها را قبل‌از ادغام در میزبان پاکسازی کرده است. کارهایی از قبیل تغییر عملکرد برنامه به‌حالتی که، خروج‌کامل از ماشین‌مجازی برای اعمال تغییرات مورد نیاز نباشد، یا کنترل انجام این عمل به کاربر واگذار شود، در ادامه این مقاله می‌تواند انجام‌پذیرد.

۸. مراجع

[1] Amani P., Khalozadeh H., Aref M., Proposing a Novel Sandbox using
AES Encryption Chaotic Shema, The Forth Iranian Conference on
Encryption, 2008, In Persian.

[2] Ajami M., Payandeh A., Aref M., A Power Attack on A5/1 Encryption
Algorithm, The Eight Iranian Conference on Encryption, 2012, In Persian.

[3] http://www.freebsd.org/.

[4] http://linux-vserver.org/.

[5] http://www.oracle.com/VMSystems/Zones/.

[6] http://openvz.org/.

[7] http://www.parallels.com/products/virtuozzo/.

[8] Zhu, Z. Jiang, Z. Xiao, and X. Li, "Optimizing the Performance of Virtual Machine Synchronization for Fault Tolerance," IEEE Transactions on Computers, vol. 60, no. 12, pp. 1718-1729, Dec. 2011.

[9] Jenni Susan Reuben, "A Survey on Virtual Machine Security," Draft Books on Network Security, TKK T-110.5290, Version 3, 2013.

[10] N. Li, Z. Mao, and H. Chen, "Usable Mandatory Integrity Protection for Operating Systems," Proc. IEEE Symp. Security and Privacy, pp. 164-178, May 2007.

[11] E. Kirda, C. Kruegel, G. Banks, G. Vigna, and R.A. Kemmerer,"Behavior-Based Spyware Detection," Proc. 15th Conf. USENIX Security Symp., article 19, 2006.

[12] L. Martignoni, E. Stinson, M. Fredrikson, S. Jha, and J.C. Mitchell, "A Layered Architecture for Detecting Malicious Behaviors," Proc. 11th Int'l Symp. Recent Advances in Intrusion Detection (RAID), Sept. 2008.

[13] Symantec, Inc., http://www.symantec.com/business/ security-response/threatexplorer/threats.jsp, 2013.

[14] Microsoft Security Bull., http://www.microsoft.com/ technet/ security/current.aspx, 2013.

[15] A. Lanzi1, M. Sharif, and W. Lee, "K-Tracer: A System for Extracting Kernel Malware Behavior," Proc. 17th Ann. Network and Distributed System Security Symp. (NDSS), 2009.

[16] C. Kolbitsch, P.M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang, "Effective and Efficient Malware Detection at the End Host," Proc. 18th Conf. USENIX Security Symp., pp. 351-366, 2009.

[17] O. Sukwong, H. Kim, and J. Hoe, "Commercial Antivirus Software Effectiveness: An Empirical Study," Computer, vol. 44, no. 3, pp. 63-70, Mar. 2011.

[18] S.A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion Detection Using Sequences of System Calls,"J. Computer Security, vol. 6, no. 3, pp. 151-180, 1998.

[19] http://www.csmining.org/.

[20] http://mmonit.com/.

[21] PC Magazine, "PC Magazine Benchmarks," http://www. pcmag. com/encyclopedia_term/WebBench.asp/., 2013.

[22] Yin, Song, Egele, Kruegel C., and Kirda E., "Panorama: Capturing System-Wide Information Flow for Malware Detection and Analysis," Proc. 14th ACM Conf. Computer and Comm. Security (CCS), 2007.

[23] Zhiyong Shan, Xin Wang; Tzi-Cker Chiueh, "Malware Clearance for Secure Commitment of OS-Level Virtual Machines," IEEE Transactions on Dependable and Secure Computing, vol.10, no.2, pp.70,83, March-April 2013.

[24] Hahn A., Ashok, A.; Sridhar, S.; Govindarasu, M., "Cyber-Physical Security Testbeds: Architecture, Application, and Evaluation for Smart Grid," IEEE Transactions on Smart Grid, vol.4, no.2, pp.847, 855.

[25] Bari, M.F., Boutaba, R. Esteves, R, Granville, L.Z. Podlesny, M., "Data Center Network Virtualization: A Survey", IEEE Communications Surveys & Tutorials, vol.15, no.2, pp.909,928.