

## ارائه یک روش جدید برای شناسایی بدافزارها در سطح مجازی‌ساز در ماشین‌های مجازی

حسین شیرازی<sup>۱</sup>، سید محمدرضا فرشچی<sup>۲\*</sup>

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 ماشین‌های مجازی در مدیریت بهینه و اثربخش منابع سیستمی نقش مهمی ایفا می‌کنند. مجازی‌سازی، به مفهوم ایجاد چند ماشین مجازی بر روی یک سخت‌افزار مهمان است که امکان استفاده بهینه از منابع سیستمی را فراهم می‌نماید. امروزه، با گسترش بدافزارها در ماشین‌های مجازی، ضرورت توجه به آسیب‌پذیری‌ها در این حوزه از سیستم‌های میزبان گسترش یافته است. رفتار یک بدافزار در ماشین مجازی، تغییر اشیاء سیستمی در گام اول، و در گام دوم، نفوذ به سیستم‌عامل میزبان ماشین مجازی در زمان اتمام کار، انجام فرآیندهای دلخواه به‌عنوان گام نهایی است. این مقاله برای اولین بار به ارائه یک روش امن، برای شناسایی، دسته‌بندی و امحاء بدافزارها در ماشین مجازی پرداخته است. روش پیشنهادی به نام، *SMS*، 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 شناسایی شده و به ماشین میزبان منتقل می‌شود. استفاده از جریان‌های اطلاعاتی فرآیندها در ماشین مجازی، دقت بسیار مطلوبی را برای مکانیزم پیشنهادی فراهم کرده است. در روش پیشنهادی، اولاً برخلاف روش‌های کنونی، تنها قسمتی از اطلاعات سیستمی مورد پردازش قرار می‌گیرد. ثانیاً، برخلاف کلیه ضد بدافزارهای موجود، بجای بررسی تک‌به‌تک اشیاء سیستمی، گروه‌های تشکیل شده 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، به شدت 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 *SMS* با ضد بدافزارهای تجاری کنونی، عملکرد بسیار مناسب در تشخیص و حذف بدافزارها و همچنین کاهش نرخ نمونه‌های غلط منفی به‌خوبی محرز شده است.

**واژه‌های کلیدی:** امنیت مجازی‌ساز، طبقه‌بندی بدافزار، امنیت در لینوکس، مدل رفتاری

### ۱. مقدمه

تغییرات مختلفی در سطح سیستم‌عامل رخ می‌دهد. از جمله تغییر در سیستم‌فایل، تغییر در رجیستری (این مفهوم در لینوکس با فایل‌های تنظیمات بیان می‌شود- 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 ماشین مجازی)، در لایه مجازی‌ساز، باید قادر به تفکیک فعالیت‌های غیرمجاز بدافزارها در سطح سیستم‌عامل بوده و بتواند کلیه فعالیت‌های صحیح کاربر را بازشناسی کرده و به سطح سیستم‌عامل میزبان منعکس کند. ارائه یک روش امن ترکیبی 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 پروسس‌های سالم به سطح سیستم‌عامل برای ذخیره‌سازی منعکس کند (به‌عنوان مثال، وب‌سرورها، پایگاه‌داده‌ها، یا فایل‌های تنظیمات نرم‌افزارهای کاربردی).

۲- بیشتر برنامه‌های زمان واقعی بر روی ماشین مجازی، دو نمونه از برنامه خود را اجرا می‌کنند. یک نسخه از پروسس مربوط به

یک ماشین مجازی در سطح سیستم‌عامل از مزیت‌های مختلفی، از جمله، هزینه راه‌اندازی/ غیرفعال‌سازی اندک [۱]، نیاز به حداقل منابع [۲]، و قابلیت تعمیم‌سازی گسترده از دید اشتراک محیط اجرای سیستم‌عامل میزبان، و محدودیت تغییر حالات در ماشین مجازی، برخوردار است. علاوه‌بر این، تحمل‌پذیری نسبت به اجرای دستورات غیرمجاز یا بروز خطا در ماشین میزبان، از ویژگی‌های عمومی استفاده از ماشین مجازی است. بنابراین امکان نسبی نصب نرم‌افزار، بدون نگرانی از آلودگی به بدافزار و تغییرات سیستمی برای کاربر فراهم می‌شود. به‌عبارت دیگر، یک ماشین مجازی آلوده به بدافزار را به‌راحتی می‌توان تعویض کرد. اگرچه این راه‌حل بدیهی به نظر می‌رسد اما نیازمند صرف هزینه مضاعف برای برگرداندن ماشین جدید به وضعیت کاری کاربر در ماشین مجازی قبلی خواهد بود. زمانی که کاربر با لیست پروسس‌های عادی در حال کار است،

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

برای فراهم کردن 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 حائز اهمیت می‌باشد. بنابراین مدل ارائه شده باید کمترین سربار را بر روی ماشین مجازی داشته باشد. از طرف دیگر ماشین مجازی خود دارای سرباری بر روی سیستم‌عامل میزبان است. بنابراین در صورت داشتن پیچیدگی زمانی یا مکانی، مشکل تنزل مطبوع [۸] (افت کلی عملکرد سیستم) نیز ممکن است رخ دهد. مشکل دوم اینکه مکانیزم پیشنهادی باید تفکیک صحیحی میان تمامی فعالیت‌های مشکوک یا تغییر حالت‌های ناشی از رفتار فرآیندها (یا بدافزارها) انجام دهد. بدیهی است، شناسایی صحیح رفتار کلیه بدافزارها براحتمالاً امکان پذیر نخواهد بود. روش‌های کنونی از قبیل بررسی رخدادهای 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 ماشین مجازی 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 تغییر حالت‌هایی که توسط بدافزارها انجام شده است، از ادغام آنها با سیستم‌عامل میزبان ممانعت به عمل می‌آورد. نتایج آزمایشات در این تحقیق نشان می‌دهد مکانیزم پیشنهادی سطح وسیعی از امنیت را برای فعالیت مناسب ماشین مجازی فراهم می‌کند. اگرچه روش‌های مبتنی بر ایجاد چندفرآیندی برای غلبه بر مشکل جامعیت منابع مشترک در [۱۰ و ۱۱] ارائه شده‌است، اما در اینجا از پرداختن به این مشکل خودداری شده است.

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

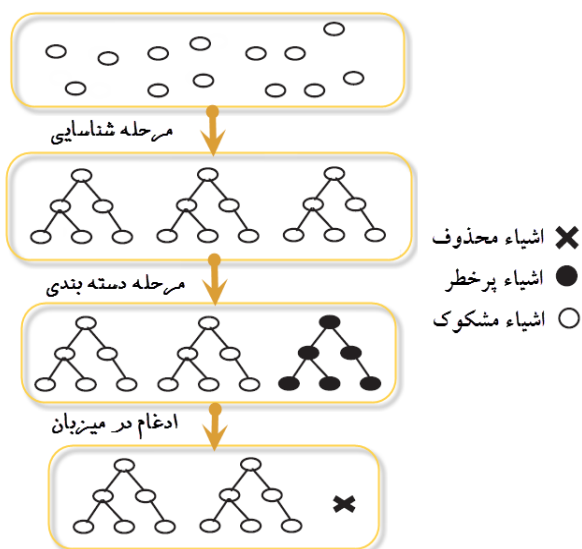
در روش پیشنهادی، SSM، از سه ویژگی جدید استفاده شده است. اولین ویژگی عدم توجه به فایل‌های ثبت وقایع (فایل‌های log) به دلیل حجیم بودن و سربار زمانی آن‌هاست. در حقیقت از جریان اطلاعاتی در سطح سیستم‌عامل و منطق بدافزارها برای دسته‌بندی استفاده شده است. نتایج بررسی‌ها در بخش‌های بعد، سربار اندک روش پیشنهادی نسبت به روش‌های مبتنی بر آنالیز فایل‌های ثبت وقایع که عملکرد نسبتاً ضعیفی دارند را نشان می‌دهد. ثانیاً، برعکس روش‌های معمول تشخیص نفوذ و بازیابی سیستم [۱۲، ۱۳، ۲۲، ۲۳، ۲۵] که اقدام به بررسی تک‌به‌تک اشیاء سیستمی

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 سیستم‌عامل میزبان منقل 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. به عبارت 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 کرده باشد. روش پیشنهادی امن ترکیبی، برای بازگرداندن فایل‌های موقتی این‌گونه فعالیت‌ها باید به‌درستی عمل کرده و فایل‌ها را به سطح سیستم‌عامل میزبان جهت ادغام بازگشت دهد.

با توجه به موارد فوق و همچنین با توجه به معماری‌های کنونی ماشین مجازی، تشخیص عملکرد صحیح از مخرب در سطح ماشین مجازی از اهمیت خاصی برخوردار است. تکنولوژی‌های کنونی ماشین مجازی 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 سیستم‌عامل FreeBSD و استفاده از ویژگی‌های CPUهای پر قدرت 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 به دلیل دشواری در ایزوله کردن guestها بعضی از سرویس‌ها به صورت 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 فراهم نشده و ثانیاً تغییرات ادغامی حالت‌ها به سیستم‌عامل میزبان بدون توجه به فاکتورهای امنیتی ذکر شده فوق انجام می‌گیرد.



شکل ۱. مراحل سه گانه روش SSM در حالت کلی

## ۱.۲. دسته‌بندی اشیاء مشکوک سیستمی

چنان که گفته شد، یکی از نوآوری‌های پژوهش پیشنهادی، بررسی اشیاء سیستمی به صورت گروهی است. سیستم‌های تشخیص بدافزار سنتی، از بررسی‌های منفرد بر روی هر یک از اشیاء، استفاده می‌کنند. این خصوصیت سبب ایجاد سربرازمانی، تنزل مطبوع و مصرف حافظه خواهد شد. با استفاده از ویژگی بررسی گروهی، اثرات تخریبی بدافزارها با دقت بیشتری شناسایی خواهد شد. چراکه کلیه رفتار بدافزار، بر روی ماشین مجازی شناسایی شده و در مرحله نهایی الگوریتم، از بین خواهند رفت. در حالی که در بررسی منفرد در روش‌های سنتی، تنها عملکردهای سیستمی مورد بررسی قرار می‌گیرند، و امکان عدم تشخیص کارکرد کلی 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:

- فرآیندهای منتج 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 به خود گرفته، یا بتواند، ماکروها را اجرا کند. برای دسته اول، BAT، VBS، JS، DLL و برای دسته دوم می‌توان، DOC، PPT، DOT، را نام برد. همچنین، اجازه تغییر توسعه فایل، توسط فرآیند اجرا شده را نخواهد داد. با استفاده از این دو قانون، هر کد اجرایی مخرب، از لحظه ورود به سیستم تحت بررسی قرار خواهد گرفت. در حقیقت، مشکل اصلی تضمین صحت عملکردی کل سیستم، از این طریق خواهد بود. به عبارت 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

می‌کنند، روش پیشنهادی از بررسی دسته به دسته استفاده می‌کند که از لحاظ زمانی کارایی 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 ماشین مجازی در سطح سیستم‌عامل که به منظور انتقال فرآیندها به میزبان است چهار مزیت فراهم شده است. اولاً استفاده از سه ویژگی بالا، سبب ایجاد یک روش سبک و کارا برای انتقال حالت‌ها به سیستم میزبان در صورت آلودگی به بدافزار خواهد شد. ثانیاً، یک الگوریتم دسته‌بندی در سطح ماشین مجازی ارائه شده است. این دسته‌بندی براساس جریان اطلاعاتی 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 جریان اطلاعاتی، تنها مبتنی بر دنبال کردن 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 کارایی مؤثر بهبود یافته نسبت به روش‌های کنونی 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 داده شده و سربراز اعمالی بر Xen هم تاثیر چندانی بر عملکرد سیستم نداشته است. علاوه بر این نسبت به مقایسه برخی از آنتی‌ویروس‌های تجاری در کشف بدافزارها در Xen، روش SSM توانایی معقول تری را به نمایش گذاشته 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 صحت داده‌ها در سیستم‌عامل میزبان و ماشین مجازی، باید به قاعده بازنویسی (ذخیره تغییرات)، ماشین مجازی توجه کرد. بعد از اتمام فرآیند کاری ماشین مجازی، فایل‌ها و اطلاعات رجیستری (در لینوکس فایل تنظیمات)، توسط جلسه کاری ماشین مجازی بازنویسی می‌شود. بنابراین مکانیزم ما، باید از بروزرسانی تغییرات بدافزار جلوگیری کند. لزوم تأکید بر توقف (عادی یا غیرعادی) کامل ماشین مجازی، از ساختار سیستم فایل و مدیر فرآیند نشات می‌گیرد (چراکه برخی از فرآیندها قابلیت بازنویسی در سیستم فایل میزبان را تا باز شدن 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 روش پیشنهادی را به نمایش گذاشته‌ایم.

جهت‌دار بوده و دارای نقطه شروعی، به‌عنوان ریشه خواهد بود. یال‌ها در این گراف همان جریان اطلاعاتی 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 هر یک 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 هر یک 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 اشیاء سالم نیز باشد. بنابراین ایده تقسیم هر گراف به زیرگراف انجام داده خواهد شد، 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 روش‌های انتقال بدافزار به میزبان را از دید کانال انتقال می‌توان در سه گروه دسته‌بندی کرد. 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 اسکرپیت‌های دیگر برای نفوذ کامل در میزبان. فراخوانی‌های حاصل از این فرآیندها در گروه *انشعاب طبقه‌بندی* می‌شوند (گروه *انشعاب B*، در شکل ۲ (ب)). دسته دوم، 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 طبقه‌بندی* می‌شوند. (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 براحتمی امکان‌پذیر نخواهد بود (مثل A به C در شکل ۲ (الف)). به‌عبارت دیگر زمانی که 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

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 فایل مشکوک 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 رفتارهای ثابت 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 این قوانین، از دنبال کردن فعالیت‌هایی از قبیل خواندن فایل‌ها، پوشه‌ها یا بسیاری از فراخوانی‌های سیستمی اجتناب کرده‌ایم. در واقع این فعالیت‌ها، همان جریان‌های اطلاعاتی 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 نسبتاً 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 تضمین می‌کنیم 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 مجدداً 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 راه، 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،

باشد، ریشه آن گره،  $v_r$ ، 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 آن گروه خواهد بود. بنابراین نتیجه می‌گیریم، که اگر اشیاء داخل یک گروه پرخطر باشد، ریشه آن گروه 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.

### ۳.۲. تشخیص گروه‌های پرخطر

در این تحقیق برای تشخیص بدافزارها (اشیاء با رفتارهای پرخطر)، اقدام به ایجاد یک هسته برخط، کرده‌ایم. روش‌های کنونی در تشخیص بدافزارها [۱۵، ۱۶، ۱۷]، با استفاده از جریان اطلاعات پویا، به استخراج ویژگی و دنبال کردن فرآیندها می‌پردازند. این روش‌ها سربرار بسیار زیادی را به سیستم اعمال می‌کنند، و برای عملیات برخط مناسب نیستند. روش‌های سنتی مبتنی بر تشخیص رشته بایت‌ها (تشخیص امضاء) نیز به دلیل 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 نمونه‌منفی غلط، 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 الگوی رفتاری تعریف شده باشد، پرخطر نامیده می‌شود (شکل ۳). الگوی رفتاری استفاده‌شده در این تحقیق، از فراخوانی‌های توابع API، (واسط برنامه‌نویسی کاربری)، و دنبال کردن جریان اطلاعاتی سطح سیستم‌عامل در ماشین مجازی تشکیل شده است. برخلاف بهترین روش‌های کنونی، مبنی بر دنبال کردن 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.

حال اجرا، دارای اولویت است. لذا اگر بدافزار اقدام به تولید نمونه فرآیندی بکند، برچسب خود 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 گروه‌های تشکیل شده، 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 آن گروه پرخطر خواهد بود. از آنجا 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 هر یک از اشیاء پرخطر، با اجرای یکی از قواعد سه‌گانه می‌تواند بدافزار را به سطح ماشین مجازی 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،

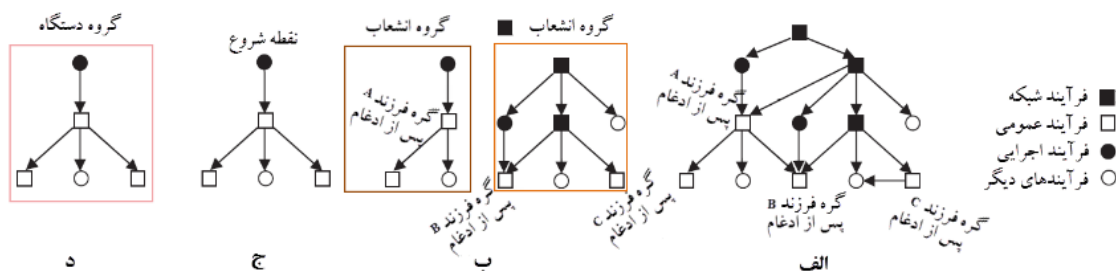
$$v_r.f = \text{پرخطر} \wedge (v_r, v_1) \in E \rightarrow v_1.f = \text{پرخطر}$$

$$v_r.f = \text{پرخطر} \wedge (v_1, v_2) \in E \rightarrow v_2.f = \text{پرخطر}$$

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

بنابراین، تمامی فرزندان گروه باید پرخطر باشد.

۲- اگر هر یک از اشیاء به‌صورت  $v_r \neq v_n$  در یک گروه پرخطر



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

خواهد بود (دقت کنید، طبق اثبات بالا می‌توان بجای اشاره به اشیاء  $c$  از گروه  $c$  (کوچک)، استفاده کرد.) بنابراین:

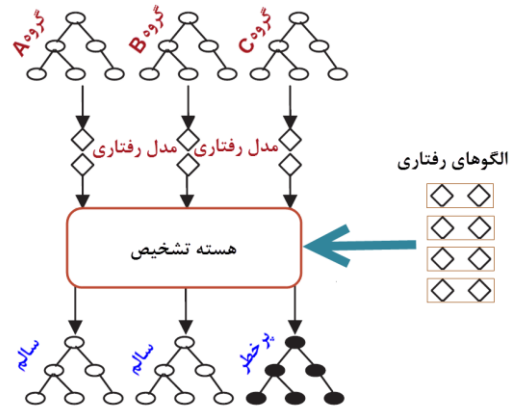
- ۱-  $\forall c \in C, b, p \notin C \rightarrow$  اقدامی انجام نده
- ۲-  $\exists c \in C, b, p \in c \wedge c.t = b.t \rightarrow$  نوع رفتاری نداریم
- ۳-  $\exists c \in C, b, p \in c \wedge b.t \neq c.t \wedge c.t \neq b.t \rightarrow$  نوع رفتاری نداریم
- ۴-  $\rightarrow c.f =$  پرخطر
- ۵-  $\exists c \in C, b, p \in c \wedge b.t = c.t \rightarrow$  اقدامی انجام نده
- ۶-  $\exists c \in C, b, p \in c \wedge b.t = t_b \rightarrow c.f$
- ۷- اجازه رفتار  $b$  داده نشود  $\wedge$  پرخطر  $\rightarrow$

قاعده اول اقدامی را در پی ندارد چرا که رفتار فرآیند در هیچ یک از گروه‌ها قرار نمی‌گیرد (توسط فرآیند پرخطر ایجاد نشده‌است). قاعده دوم، رفتار تشخیص داده شده را در گروه خود دسته‌بندی خواهد کرد. قاعده سوم، همان تشخیص وجود دو رفتار متفاوت در یک گروه است که قبلاً توضیح داده شد. لذا SSM، گروه را به‌عنوان پرخطر شناسایی خواهد کرد. قاعده چهارم، رفتار یکسان اشیاء داخل یک گروه است، که اقدام خاصی انجام نمی‌دهد. نهایتاً، قاعده پنجم، گروه مذکور را به‌عنوان پرخطر شناسایی کرده، اما به دلیل وجود رفتاری جدید از سمت فرآیند والد بدافزار، اجازه عملکرد به فرآیند داده نمی‌شود. لازم به ذکر است، SSM، تنها به دسته‌بندی و تشخیص دسته‌های پرخطر می‌پردازد و هیچ‌گونه توقف‌پردازی در این سطح در ماشین مجازی صورت نخواهد گرفت. برخی از آنتی‌ویروس‌های تجاری، مثل Avira، پس از تشخیص فرآیند پرخطر بلافاصله اقدام به بلاک کردن 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 بلاک کردن فرآیند سیستمی ممکن است اطلاعاتی از دست‌رفته یا عملکرد سیستم ناپایدار گردد.

#### ۴.۲. ادغام گروه‌های سالم

همان‌طور که ذکر شد ادغام پس از توقف کامل سرویس‌های ماشین مجازی 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 انجام می‌دهند. 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 به حساب می‌آیند. بنابراین در الگوریتم پیشنهادی از ادغام این اشیاء نیز جلوگیری شده است.



شکل ۳. هسته تشخیص بدافزار، ورودی مدل‌های 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 منفی غلط و منفی مثبت ثابت باقی نمی‌ماند، و بهترین انتخاب برای سطح ماشین مجازی 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 شروع به کار ماشین مجازی فعال کند.
- ۲- فرآیندی که گسترش فرآیند در سیستم عامل را در دستور کار خود قرار دهد.
- ۳- فرآیندی که ویژگی‌های مخفی‌سازی خود را فعال نماید، 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 انجام کاری نخواهد بود. بنابراین به جای تهیه لیست الگویی بسیار زیاد، از ۴ قاعده فوق برای مطابقت گروه‌ها و اشیاء داخل آن استفاده شده‌است. با توجه 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 رفتار کنونی مورد استفاده توسط هسته تشخیص SSM باشد، در این صورت،  $b, p$  بیانگر فرآیندی خواهد بود 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 گروه‌های تشکیل شده خواهد بود.  $c.t$  نوع رفتاری است که در اشیاء گروه  $c$ ، مورد استفاده قرار گرفته‌است. نهایتاً، نماد T پرخطر | سالم =  $c.f$ ، بیانگر سالم یا پرخطر بودن گروه  $c$ ،



فراخوانی‌های API در لینوکس (ویندوز)، برچسب‌گذاری تمامی اشیاء را با الگوریتم پیشنهادی، به راحتی می‌توان انجام داد. بررسی‌های نشان می‌دهد، اکثر فراخوانی‌ها در سطح هسته اتفاق افتاده و به این دلیل امکان عبور بدافزار از الگوریتم پیشنهادی به راحتی امکان پذیر نخواهد بود. برای اشیاء ثابت مسیر خود ماشین مجازی در فراخوانی پیش فرض تابع سیستمی، برای ذخیره‌سازی انتخاب و برای اشیاء موقت تصویری آن را به صورت فایل داده موقتی، ذخیره خواهیم کرد.

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

مرحله ادغام، آخرین روال موجود در شکل ۴ می‌باشد. همان طور که گفته شد، از آنجا که دستور ادغام یا حذف توسط درخواست کاربر صورت می‌گیرد. روال ادغام ابتدا به بررسی تغییرات حالت در ماشین مجازی پرداخته و سپس گروه‌های سالم را با شروع درخواست کاربر از مسیر ماشین مجازی برداشته و در سیستم عامل میزبان ادغام خواهد کرد.

### ۳.۱. خود تکثیری فرآیند

اگر شناسایی بدافزارها تنها بر اساس الگوی رفتاری باشد، شناسایی گروه‌های پرخطر به صورت پویا در بازه‌های زمانی معنایی نخواهد داشت. به عبارت دیگر، هسته تشخیص، قادر به شناسایی ویژگی‌های بدافزارهای جدید نخواهد بود. برای پوشش مطلوب این قضیه، از ساختاری شبیه [۱۸]، برای تعریف رفتارهای جدید استفاده کرده ایم، پس:

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

نحوه تعریف رفتار بدافزارهای جدید خواهد بود. منظور از عملیات، عملکرد فراخوانی‌های سیستمی مشابه خواهد بود. به عنوان 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 مخفی سازی فرآیند قرار خواهد گرفت. بنابراین در هر فراخوانی از تابع API مربوط به بررسی سه ویژگی اول، رفتار سالم یا پرخطر از فراخوانی مربوطه شناسایی می‌شود. نوع رفتار

بدین ترتیب، الگوریتم پیشنهادی به صورت زیر خواهد بود.

فرض کنید،  $c.s$  بیانگر هویت والد گروه و  $c.o \in A$ ، بیانگر اشیائی باشد که فعالیت خودراه اندازی رادارها باشند.  $o_1 \triangleright o_2, o_1 \in A$  بیانگر اجرای خودراه اندازی  $o_1$ ، 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، یا تغییر کرده ماشین مجازی برای ادغام، استفاده شده است. پس:

$$\forall c \in C, (c.f = \text{سالم}) \wedge (c.s \notin R) \wedge (\exists o \in C, o \in A) \rightarrow$$

پرخطر  $c.f$

$$\forall o_1 \in A, o_1 \in c_1, c_1 \in C, (c_1.f = \text{پرخطر})$$

$$\wedge (\exists o_2 \in c_2, o_1 \triangleright o_2, c_2.f = \text{سالم}) \rightarrow c_2.f = \text{پرخطر}$$

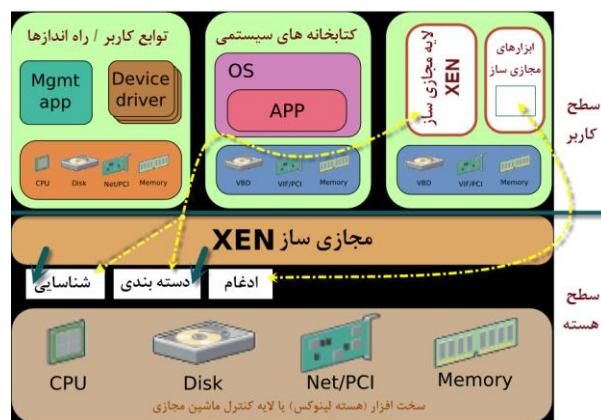
$$\forall c \in C, (c.f = \text{سالم}) \wedge \text{ادغام}(c) \rightarrow \forall o \in O, \forall c \in C, o \notin$$

$$\text{عدم ادغام} \rightarrow \text{پرخطر}(c.f) \forall c \in C, \text{ادغام}(o) \rightarrow c$$

برقرار است. همچنین لازم به ذکر است، جامعیت داخلی داده‌ها به دلیل 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 ماشین مجازی اثرگذار نخواهد بود.

### ۳. پیاده سازی SSM

برای نمایش ساختار دقیق مدل پیشنهادی از Xen بر روی لینوکس نسخه Redhat، استفاده شده است. پیاده سازی ادغام کننده در لایه مجازی ساز Xen صورت گرفته است. این لایه شامل راه اندازهای هسته و توابع سطح کاربر می‌باشد. علاوه بر این برای عمل ادغام یک واسط کنترلی در سطح کاربر نیز در نظر گرفته ایم. شکل ۴ بیانگر معماری پیاده سازی عمومی، SSM، می‌باشد.



شکل ۴. ساختار پیاده سازی SSM در لایه مجازی ساز

در این شکل وظیفه تشخیص اشیاء پرخطر برعهده روال شناسایی کننده خواهد بود. در حقیقت با استفاده از دنبال کردن

برای آزمایش مورد استفاده قرار گرفت. ما برای ارزیابی از ابزار MONIT در [۲۰]، استفاده کرده تا آنالیز فرآیندها، رجیستری (فایل تنظیم مدنظر است)، و عملیات فایل‌ها، را زیر نظر بگیریم. همچنین، از دو کامپیوتر و دو سرور، به صورت 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  $Z$  در جدول 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. وب سرور Apache، یک FTP سرور، و یک نرم‌افزار هم برای برقراری ارتباط بین سیستم‌ها به نام IRC، از جمله نرم‌افزارهای نصب شده بر روی سرور اول می‌باشد. سرور دوم هم به‌طور مشابه، دارای همین نرم‌افزارها بوده و به‌عنوان یک وب سرور سالم شروع به کار می‌کند. کامپیوترهای معمولی از مرورگر Chrome برای دانلود نمونه‌ها و اتصال به سرورها استفاده می‌کنند. همچنین از توزیع سیستم عامل Redhat برای اولین سیستم معمولی استفاده شده است (لازم به ذکر است، نوع میزبان‌ها و سیستم عامل یا استفاده از نرم‌افزارهای دیگر تأثیری بر اجرای مدل پیشنهادی ندارد). برای اعمال کاربردهای حقیقی به مرور صفحات وب آلوده، بارگذاری فایل‌های اجرایی بدافزارها، ارسال و دریافت رایانامه، دسترسی به پوشه‌های اشتراکی، انتقال فایل‌های آلوده و اجرای فرآیندها پرداخته شده است. بنابراین، ارزیابی SSM در این شرایط مصداق کاملی خواهد داشت، و مکانیزم پیشنهادی به‌دقت تحت بررسی قرار خواهد گرفت. 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 ماشین مجازی جدید اجرا، و در مرحله دوم بدافزار پس از فعال‌سازی SSM، به همراه نرم‌افزارهای سالم، در یک ماشین مجازی جدید به منظور ادغام، اجرا می‌شود. پس از

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 به صورت 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 دیگری استفاده می‌کنیم. این جدول برای استخراج نوع رفتار به دست آمده از یک بدافزار، مورد استفاده قرار می‌گیرد. تابع ASCII برای محاسبه مقدار اسکی رشته ورودی خواهد بود. طول جدول، بیانگر تعداد رفتارهای داخل جدول و طول رشته نیز، بیانگر تعداد کاراکترهای رشته ورودی خواهد بود. تنها نکته قابل ذکر از الگوریتم بارگذاری اطلاعات داخل جدول، ترتیب جداول خواهد بود. در این ترتیب ابتدا سه جدول اول، و سپس جدول چهارم بارگذاری خواهد شد. به عبارت دیگر ابتدا رفتار برای تطبیق به جدول اول ارسال خواهد شد، در صورت عدم وجود این رفتار سراغ جدول دوم رفته و آن را 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

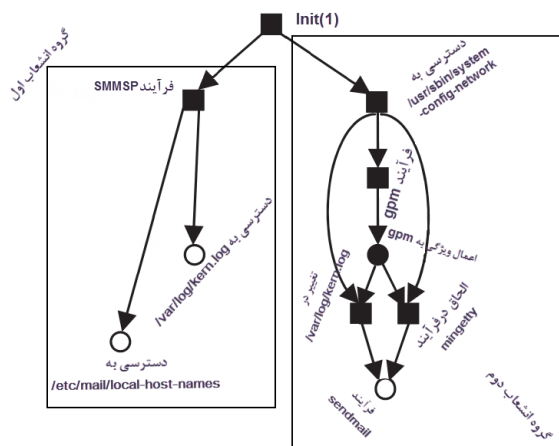
ارزیابی‌های دقیق ما بر روی مکانیزم پیشنهادی SSM، از سه قسمت تشکیل شده است. در بخش اول به بررسی مدل پیشنهادی برای گروه‌بندی بدافزارهای واقعی خواهیم پرداخت. در بخش دوم مقایسه روش پیشنهادی با ضد بدافزارهای تجاری امروزی انجام شده است. نهایتاً در بخش آخر به کارآیی مدل پیشنهادی اشاره خواهد شد.

##### ۴.۱. عملکرد در ادغام صحیح

برای نمایش عملکرد SSM، از ۶۰ بدافزار واقعی در پایگاه PUI [۱۹]، استفاده شده است. در این تحقیق سعی شده است تا حد امکان رفتار بدافزارها متفاوت تر از آنچه که در پروفایل رفتاری تعریف شده است، مورد استفاده قرار بگیرد. همچنین ۵۰ برنامه سالم هم



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 مشخص شود داده‌های صحیحی 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. علاوه بر این، نرم‌افزارهای سالمی که پس از الحاق بدافزار به آنها، ادغام شده‌اند، به‌راحتی در میزبان، به‌فعالیت خود ادامه داده و عملکردهای مخرب آنها در روال ادغام بی‌تأثیر شده است. نمونه‌های غلط‌مثبت، زمانی که دریافت بدافزارها بدون استفاده از یک برنامه کاربردی صحیح، مثل Chrome، (الگوی رفتاری تعریف شده در تحقیق)، بارگذاری شده‌اند، افزایش یافته است. با بررسی فایل‌های تغییرات 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 به نام، Cdorked.A، پرداخته شده است. این بدافزار در خانواده‌ای شبیه ویروس شبکه‌ای Blast قرار دارد. شکل ۵، خروجی سطح مجازی ساز 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 خودراه‌اندازی را شامل می‌شود و در دسته پرخطر قرار گرفته است. چرا که طبق جدول رفتاری، دو عملکرد مخرب را دنبال کرده است. بنابراین کل اشیاء گروه طبق تعریف بخش ۲، در دسته پرخطر قرار گرفته و ادغام نخواهند شد.



شکل ۵. گراف وابستگی بدافزار Cdorked.A

جدول ۲. انواع نمونه‌های مورد استفاده در آزمایشات

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

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

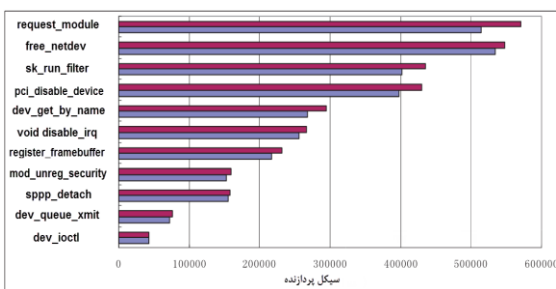
جدول ۱. انواع رفتاری در نظر گرفته شده برای ارزیابی SSM

نوع	تغییر رجیستری برای ایجاد خودراه‌اندازی (فایل تنظیمات)	ایجاد یا تغییر سرویس‌ها	ایجاد یا تغییر در راه‌اندازهای دستگاه
نوع ۲	خودتکرار شونده‌گی	فرآیندهای دیگر	ایجاد فرآیند به‌صورت غیرعادی
نوع ۳	تغییر در تنظیمات امنیتی	غیرفعال کردن نرم‌افزارهای امنیتی	غیرفعال کردن بازگردانی سیستم
نوع ۴	سرقت اطلاعات هویتی		

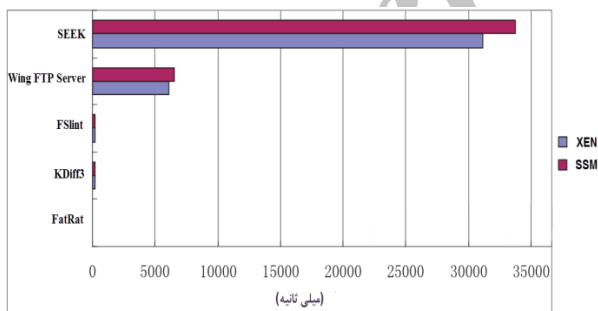
۲.۴. مقایسه با بدافزارهای تجاری

برای ارزیابی دقیق‌تر مکانیزم پیشنهادی، از ضد بدافزارهای

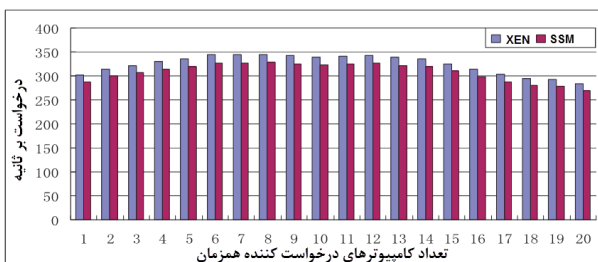
مورد بررسی قرار می‌دهد، تنزل مطبوع حالت اولیه برای کل سیستم، تغییر چندانی ندارد. برای ارزیابی زمانی برنامه‌ها، ما از اندازه‌گیری زمان میانگین، شروع تا خاتمه برنامه‌ها استفاده کرده‌ایم. شکل ۹ بیانگر اعمال ۰/۸ تا ۸/۷ درصدی سربرار به میانگین زمانی کلی اجرای برنامه‌ها، با وجود SSM نسبت به عملکرد معمول Xen می‌باشد. نهایتاً به منظور ارزیابی خدمات سرویس دهنده وب، توانایی عملیاتی Apache در شکل ۱۰ مورد بررسی قرار گرفته است. این بررسی با مقایسه کارایی 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، کارایی با وجود SSM، در حدود ۹۵ درصد کارایی در حالت Xen بدون وجود مدل پیشنهادی است. ارزیابی‌های انجام شده نشان می‌دهد، وجود مکانیزم SSM، علاوه بر ارائه قدرت تشخیص بالا و مقابله با رفتارهای پرخطر فرآیندها، از سربرار بسیار کمی در ماشین مجازی و لایه مجازی‌ساز برخوردار بوده است.



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

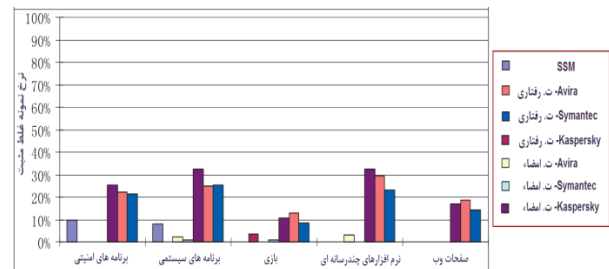


شکل ۹. سربرار ایجاد شده بر نرم‌افزارها با وجود SSM و حالت عادی Xen

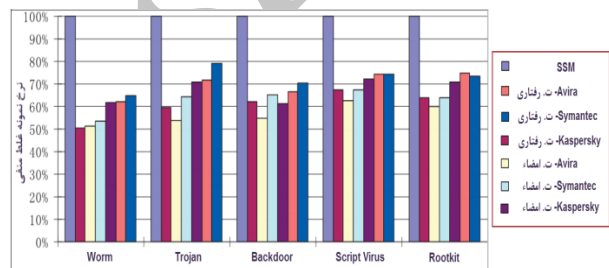


شکل ۱۰. بررسی کیفیت سرویس دهنده با اجرای SSM و حالت عمومی Xen

توجه به عدم تشخیص بدافزارهای جدید در روش‌های تشخیص امضاء این نکته قابل توجه خواهد 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 (بدون در نظر گرفتن روش مورد استفاده) اما SSM به خوبی تمامی تغییرات پرخطر را شناسایی کرده است.



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



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

### ۴.۳. سربرار ناشی از SSM

در این بخش از سه جنبه مختلف به بررسی سربرار سیستم SSM پرداخته شده است. ابتدا به بررسی سربرار فراخوانی‌های سیستمی تحت نظارت SSM پرداخته شده، سپس سربرار زمانی کل برنامه‌های مستقل، محاسبه گردیده است. در نهایت 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، FSInt، KDf3 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 بر روی هر دو کامپیوتر همراه با توزیع لینوکس Redhat، راه‌اندازی شده است. طبق توضیحات داده شده، سربرار 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 ۱/۳ درصد سربرار میانگین سیکل پردازنده را به Xen، در حالت کلی فراخوانی‌ها، تحمیل کرده است. از آنجا که SSM تنها قسمتی از فراخوانی‌ها را

## ۵. نتیجه‌گیری

در این پژوهش، یک مکانیزم امن، برای جلوگیری از رفتارهای پرخطر بدافزارها در ماشین‌های مجازی ارائه شده است. ماشین‌های مجازی 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 اعمال تغییرات سیستمی در ماشین‌های مجازی، سیستم‌عامل میزبان به شدت، تحت تأثیر قرار خواهد گرفت. برای اولین بار در این پژوهش مکانیزم پیاده‌سازی شده، با ارائه ویژگی‌های جدید، قادر به دنبال کردن فعالیت‌های فرآیندها در ماشین‌های مجازی 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 ماشین‌های مجازی، گروه‌هایی از اشیاء تشخیص داده شده در مرحله قبل، تشکیل می‌شود. در مرحله آخر، قبل از انجام ادغام، پروفایل‌های رفتاری مورد بررسی قرار گرفته و فعالیت‌های سالم به میزبان منتقل می‌شوند. روش پیشنهادی با اعمال سربرای بسیار ناچیز به ماشین‌های مجازی، عملکرد برنامه‌های اجرایی را تحت تأثیر قرار ندهد و به راحتی امکان ترکیب در سطح مجازی ساز را فراهم کرده است. ارزیابی‌های دقیق نسبت به ابزارهای تجاری موجود، قدرت مکانیزم پیشنهادی را نشان می‌دهد. نتایج نشان می‌دهند، روش پیشنهادی، با اعمال حداکثر ۸/۷ درصد سربرای، تمامی بدافزارها را قبل از ادغام در میزبان پاکسازی کرده است. کارهایی از قبیل تغییر عملکرد برنامه به حالتی که، خروج کامل از ماشین‌های مجازی برای اعمال تغییرات مورد نیاز نباشد، یا کنترل انجام این عمل به کاربر واگذار شود، در ادامه این مقاله می‌تواند انجام پذیرد.

## ۸. مراجع

- [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. Lanzil, 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/](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.
- [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.

## Implementing a Novel Malware Detection System in Virtual Machines

H. Shirazi<sup>1</sup>, S. M. R. Farshchi<sup>2\*</sup>

1- PhD, Malek Ashtar University of Technology

2- Malek Ashtar University of Technology

(Receiv: 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