شناسایی مزاحمت داخلی و حفاظت سیستم با استفاده از داده کاوی و روش های کالبد سنجی

چکیده

امروزه اکثر سیستم های کامپیوتری از ID و رمز عبور کاربران به عنوان الگوهای دخول به سیستم استفاده می کنند تا کاربران را تصدیق نمایند. بهرحال، برخی از افراد الگوهای دخول به سیستمشان را با همکارانشان به اشتراک می گذارند و از این همکاران درخواست کمک در کارها را دارند، لذا این الگو را تبدیل به یکی از ضعیف ترین نقاط امنیت کامپیوتر می نماید. مهاجمان داخلی، کاربران معتبر یک سیستم هستندکه از داخل به سیستم حمله می کنند و چون اکثر سیستم های شناسایی تهاجم، شناسایی دیوارهای آتش و جداسازی رفتارهای مخرب تنها از خارج دنیای سیستم اجرا شده اند، شناسایی این مهاجمان داخلی سخت می باشد. علاوه بر این، برخی از مطالعات مدعی هستند که تحلیل تماس های سیستم (SCs) که با دستور تولید شده اندف می تواند این دستورات را شناسایی کند تا بوسیله ی آن حمله ها را دقیقا مشخص نماید. لازم به ذکر است که الگوهای حمله، ویژگی های یک حمله هستند. بنابراین، در این مقاله، یک سیستم امنیت، با نام شناسایی تهاجم داخلی و سیستم حفاظت (IIDPS) ارائه شده است تا عادات استفاده ی کاربران را به عنوان ویژگی های کالبد سنجی آنها رهگیری کند و بوسیله ی مقایسه ی رفتارهای مصرف کاربر با الگوهای جمع شده در پروفایل شخصیِ نگه دارنده ی حساب، مشخص نماید که نگه دارنده ی حساب یک کاربرِ معتبر وارد شده به سیستم است یا نه. نتایج تجربی نشان می دهد که دقت شناسایی کاربر IIDPS برابر با 94.29% است، در حالیکه زمان پاسخ کمتر از 0.45s است. این مسئله نشان می دهد که IIDPS می تواند یک سیستم محافظت شده را به طور موثر و کارآمد از حمله های خودی محافظت نماید.

اصطلاحات شاخص: داده کاوی، حمله ی خودی، شناسایی و محافظت تهاجم، تماس سیستم (SC)، رفتارهای کاربر

1. دیباچه

در دهه های گذشته، سیستم های کامپیوتری به طور گسترده ای به کار گرفته شده اند تا زندگی های راحت تر و ساده تری را برای کاربران فراهم کنند. اما امنیت در زمانی که مردم از قابلیت های توانمند و قدرت پردازش سیستم های کامپیوتری استفاده می کنند، یکی از مسائل جدی در حوزه ی کامپیوتر می باشد چون مهاجمان اغلب سعی می کنند تا به سیستم های کامپیوتری نفوذ و به طور مخرب رفتار کنند، برای مثال می توان دزدیدن داده ی حیاتی شرکت، سیستم ها را از کار خارج کردن یا حتی تخریب سیستم ها را نام برد.

به طور کلی،حمله ی خودی در میان تمام تهاجم های شناخته شده مانند حمله ی فارمینگ، انکار خدمت توزیع شده (DDoS)، حمله ی استراق سمع، و حمله ی ماهیگیری با نیزه، یکی از دشوارترین موارد برای شناسایی است چون دیوارهای آتش در سیستم های شناسایی تهاجم (IDs) معمولا دربرابر تهاجم های خارجی دفاع می کنند. امروزه اکثر سیستم ها برای تصدیق کاربران، ID کاربر و کلمه ی عبور را به عنوان الگوی ورود به سیستم کنترل می کنند. بهر حال، مهاجمین می توانند برای سرقت الگوی ورود به سیستم قربانی، تروجان هایی نصب کنند یا یک آزمونی با ابعاد بزرگ و با کمک لغت نامه در پیش یگرند تا رمز عبور کاربر را بدست آورند. در زمانی که موفق می شوند، می توانند در ادامه به سیستم وارد شوند، به فایل های خصوصی کاربران دسترسی پیدا کنند یا تنظیمات سیستم را خراب کرده یا تغییر دهد. خوش بختانه، اکثر سیستم های کنونیِ میزبان بنیان و IDSs شبکه بنیان می توانند یم تهاجم شناخته شده را به صورت بهنگام شناسایی کنند. اما شناسایی اینکه مهاجم کیست، دشوار است چون بسته های حمله معمولا با IP های ساختگی مطرح می شوند یا مهاجمین می توانند با الگوهای ورود به سیستم معتبر به یک سیستم وارد شوند. اگرچه، تماس های سیستم سطح OS (SCs) در شناسایی مهاجمین و تشخیص هویت کاربران بسیار مفید است اما پردازش حجم بزرگی از SCs، کاویدنِ رفتارهای مخرب از آنها و شناسایی مهاجمین محتمل برای یک تهاجم هنوز چالش های مهنسی هستند.

بنابراین، در این مقاله، یک سیستم امنیتی با نام شناسایی تهاجم داخلی و سیستم حفاظت (IIDPS) ارائه می کنیم که رفتارهای مخرب شناسایی شده ای که در سطح SC به سمت یک سیستم روانه شده را شناسایی می کند. IIDPS از داده کاوی و روش های پروفایل کردن کالبد سنجی برای کاویدنِ الگوهای تماس سیستم (الگوهای SC) استفاده می کند. لازم به ذکر است که این الگوها به عنوان طولانی ترین ترتیب تماس سیستم (ترتیب SC) تعریف شده اند که برای کاربر چندین بار به طور مکرر در فایل ثبت وقایعِ کاربر ظاهر شده اند. ویژگی های کالبد سنجی کاربر از تاریخچه ی استفاده ی کامپیوتر کاربر بدست می آیند. این ویژگی ها به عنوان یک الگوی SC تعریف می شوند که مکررا در ترتیبات SC ارائه شده ی کاربر ظاهر شده اما به ندرت بوسیله ی دیگر کاربران استفاده می گردند.

مشارکت های این مقاله عبارتند از : 1) ویژگی های کالبدسنجی کاربر را بوسیله ی تحلیل SCs متناظر برای تقویت دقت در شناسایی تهاجم مشخص کند؛ 2) قادر است تا IIDPS را به سیستم موازی انتقال دهد تا زمان پاسخ شناسایی آن را کمتر کند؛ و 3) به طور کارآمد دربرابر تهاجم خودی مقاومت کند.

باقی این مقاله به صورت پیش رو سازماندهی می شود. بخش II کارهای مرتبط این مقاله را معرفی می کند. بخش III چارچوب و الگوریتم های IIDPS را تعریف می کند. نتایج تجربی به ترتیب در بخش های IV و V نشان داده می شوند و مورد بحث قرار می گیرند. بخش VI این مقاله را نتیجه گیری می کند.

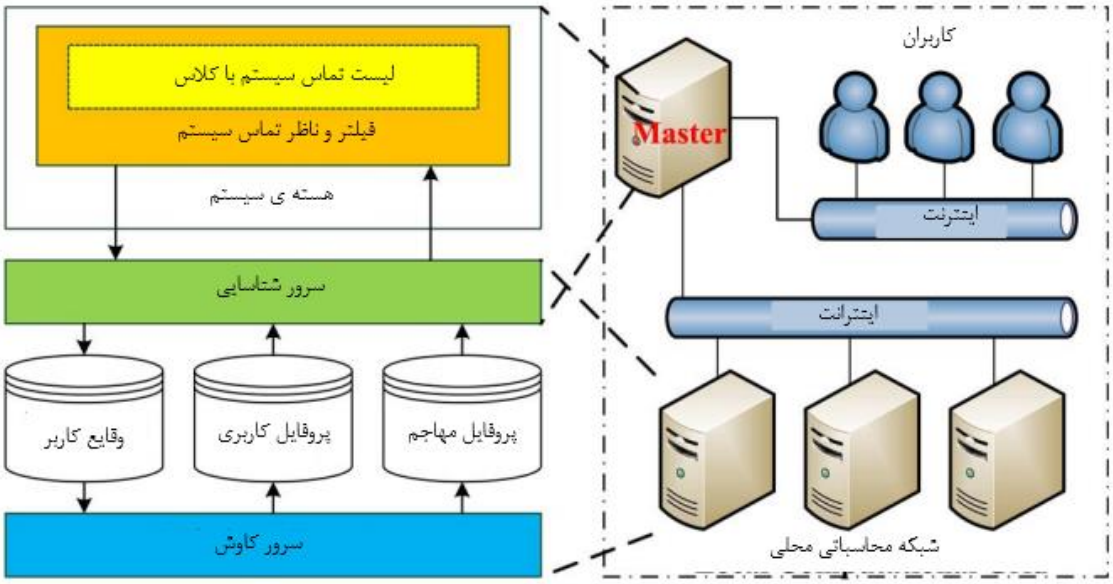
2. کارهای مرتبط

علم کالبدسنجی کامپیوتر که سیستم های کامپیوتر را به عنوان صحنه های جرم می بیند، حقایق و عقاید در مورد اطلاعات جمع آوری شده برای یک واقعه ی امنیتی را شناسایی، نگهداری، بازیابی، تحلیل و ارائه می کند. این علم کاری که مهاجمین انجام داده اند را تحلیل می کنند. این کارها مانند انتشار ویروس های کامپیوتری، بدافزارها و کدهای مخرب و ایجاد تهاجم های DDoS می باشد. اکثر روش های شناسایی تهاجم روی این موضوع که چگونه رفتارهای شبکه ی مخرب یافته شود و ویژگی های بسته های حمله (الگوهای تهاجم، برمبنای تارخچه ی ضبط شده در فایل های ثبت وقایع) بدست آیند، تمرکز کرده اند. Qadeer از کاشف بسته ی خود توسعه داده استفاده

کرده است تا بسته های شبکه را جمع کند. او حمله های شبکه را با کمک حالات شبکه و توزیع بسته و از طریق این بسته ها ی شبکه ی جمع آوری شده تمیز می دهد. O’ Shaughnessy and Gray تهاجم شبکه و الگوهای حمله را از فایل های ثبت وقایع سیستم بدست آورده اند. این فایل ها شامل اثراتی از استفاده ی غلط کامپیوتر می شوند. بدین معنی که این اثرات و الگوهای استفاده ی غلط می تواند از فایل های ثبت وقایعی تولید شده از روی قاعده دقیقتر تولید شود. Wu and Banzhaf پیشرفت تحقیقِ روش های به کاربرده شده ی هوش محاسباتی (از جمله شبکه های عصبی مصنوعی، سیستم های فازی، محاسبات تکاملی، سیستم های مصنوعی ایمن و هوش گروهی) را بررسی کردند تا رفتارهای مخرب را شناسایی کنند. این نویسندگان به طور سیستماتیک روش های شناسایی تهاجم مختلفی را مقایسه و خلاصه کرده اند تا به ما امکان دیدن این چالش های تحقیق موجود را به وضوح دهد.

این روش ها و کاربردهای فوق الذکر حقیقتا در امنیت شبکه مشارکت می کنند. بهرحال، آنها نمی توانند به سادگی کاربرانی که از راه دور به سیستم وارد می شوند را به سادگی تصدیق کنند و انواع خاص خاصی از تهاجم (برای مثال زمانی که یک کاربر بدون اعتبار با رمز عبور و ID کاربر معتبر وارد سیستم شود) را شناسایی نمایند. یک سیستم امنیتی که ویژگی های کالبد سنجی را به جای سطح SC برای سطح دستور جمع اوری می کند، در کار قبلی ما ایجاد شد. علاوه بر این، اگر مهاجمین از جلسات متعددی برای اجرای حمله (حمله های چند مرحله ای یا اجرای حمله های DDoS) استفاده کرده باشند لذا شناسایی الگوهای حمله برای چنین سیستمی ساده نیست. Hu یک IDS سبک وزن هوشمند ارائه کرد که از یک روش کالبدسنجی برای پروفایل کردنِ رفتارهای کاربر و روش داده کاوی استفاده کرده اند تا حمله های تعاونی را اجرا نماید. این نویسندگان مدعی شدند که سیستم می تواند تهاجمات را به طور موثر و کارآمد و بهنگام شناسایی کند. آنها فیلتر SC را ذکر نکردند. Giffin مثالی دیگر برای یکپارچه سازی کالبدسنجی های کامپیوتر با سیستم های دانش بنیان فراهم کرد. این سیستم یک مدل از پیش تعریف شده را اتخاذ می کند که به ترتیبات SC امکان اجرای عادی را می دهد و بوسیله ی سیستم شناسایی به کار گرفته می شود تا اجرای برنامه را محدود کرده و امنیت سیستم محافظت شده را تضمین نماید. مطرح کردن

مجموعه هایی از SCs های مخرب برای کاربردهای شناسایی مفید است و شناسایی ترتیبات حمله بر مبنای دانش جمع آوری شده است. هنگامی که یک حمله ی شناسایی نشده ارائه شده باشد، این سیستم به طور مکرر ترتیب حمله را در 2s (همانند زمان محاسباتی اش) پیدا می کند. Fiore اثربخشیِ رویکرد شناسایی را برمبنای یادگیری ماشین استفاده کننده از ماشین افتراقی محدود بولتزمن بررسی کردند تا توان تشریحیِ مدل های تولید کننده با قابیلت های طبقه بندی دقیق تجمیع کند و از این رو بخشی از دانشش را از داده ی آموزش ناقص استنتاج نماید به طوریکه طرح شناسایی ناهنجاری شبکه می تواند سطح حفاظتی مورد قبولی از هر دو تهدید داخلی و خارجی فراهم گرداند. Faisal احتمال استفاده از رشته کاوی داده را تحلیل کرد تا امنیت زیرساخت سنجش پیشرفته را از طریق یک IDS تقویت نماید. زیرساخت سنجش پیشرفته که یکی از مهمترین مولفه های حیاتی کارت هوشمند است،



شکل ۱. IIDPS چارچوب سیستم

به عنوان پلی برای فراهم کردن جریان دوطرفه ی اطلاعات بین حوزه ی کاربر و حوزه ی تامین عمل می کند. این نویسندگان با IDS به عنوان مقیاس امنیت خط دوم بعد از خط اولِ روش های امنیتی زیرساخت سنجش پیشرفته ی اولیه مانند رمز گذاری، اختیار دادن و احراز هویت رفتار می کنند.

IIDPS .3

در این بخش، در ابتدا چارچوب IIDPS را معرفی کرده و مولفه های IIDPS را به تفصیل تشریح می نماییم. دو الگوریتم نیز برای تولید یک فایل عادت کاربر و شناسایی یک متجاوز داخلی ارائه می شوند

A. چارچوب سیستم

همان طور که در شکل 1 نشان داده شده، IIDPS متشکل از یک ناظر و فیلتر SC ، یک سرور کاوش، یک سرور شناسایی، یک شبکه ی محاسباتی محلی و سه مخزن است. لازم به ذکر است که این مخازن، فایل های ثبت وقایع کاربر، پروفایل های کاربر و پروفایل متهاجم هستند. فیلتر و ناظر SC به عنوان یک ماژول قابل بارگذاری قرار داده شده در هسته ی سیستم در نظر گرفته شده است و SCs هایی که به هسته ارسال شده اند را جمع آوری کرده و به شکل (uid,pid,SC) در سیستم محافظت شده ذخیره می نماید. Uid، pid و SC به ترتیب ID کاربر، پردازش ID و c SC بوسیله ی کاربر اساسی ارسال شده است ($c \in SCs$). همچنین ورودی های کاربر را در فایل ثبت وقایع کاربر که فایلی برای نگهداری SCs های ارسالی از طرف کاربرِ دنبال کننده ی ترتیب ارسال شده اش است، ذخیرهمی نماید. سرور کاوش، داده ی ثبت وقایع را با روش های داده کاوی تحلیل می کند تا عادات استفاده از کامپیوتر کاربر را به عنوان الگوهای رفتار کاربر شناسایی نماید. سرور شناسایی، الگوهای رفتارهای کاربران را با الگوهای SC که در پروفایل متهاجم جمع شده اند (الگوهای تهاجم) و الگوهای جمع شده در پروفایل کاربر مقایسه می کند تا به ترتیب رفتارهای مخرب را شناسایی کرده و بهنگام مشخص کند که مهاجم واقعی کیست. هنگامی که یک تهاجم شناسایی می شود، سرور شناسایی به فیلتر و ناظر SC اعلام می کند تا کار بر را از سیستم حفاظت شده مجزا کند. هدف این کار جلوگیری از مهاجم است تا پیوسته به سیستم حمله نکند.

هر دو سرور شناسایی و کاوش روی شبکه ی کحاسباتی محلی راه اندازی می شوند تا سرعت شناسایی برخط IIDPS و کاوش را افزایش داده و قابلیت های کاوش و شناسایی ان را تقویت نماید. اگر یک کاربر با استفاده از الگوی ورود به سیستم فرد دیگری، به سیستم وارد شده باشد، IIDPS با محاسبه ی امتیازت تشابه بین ورودی های کنونی کاربر

(SCs) و الگوهای رفتار ذخیره شده در پروفایل های کاربرِ کاربران مختلف، تشخیص می دهد که کاربر اساسی چه کسی است. SCs جمع آوری شده در لیست SC کلاس محدود به عنوان مولفه ی کلیدیِ فیلتر و ناظر SC، SCs هستند که برای استفاده شدن بوسیله ی کلاس ها/ گروه ها ی محتلفی از کاربران در سیستم اساسی ممنوع گردیده اند. برای مثال یک منشی نمی تواند از برخی از مزایای خاص SCs بهره ببرد بنابراین، دستوراتی که این SCs را تولید می کند، برای استفاده ی منشی ها ممنوع خواهد شد.

B. فیلتر و ناظر SC

SC در حقیقت یک واسط بین کاربرد کاربر و خدمات ایجاد شده بوسیله ی هسته است. به طور کلی، مقدار بسیار زیادی از SCs در زمان اجرای کار (برای مثال یک فعالیت یا پردازش) تولید می شوند. برای مثال، زمانی که یک کاربر رمز عبور خود را بوسیله ی ارائه کردن یک دستور برنامه ی واسط "رمز عبور" به سیستم در حال کار لینوکس تغییر داده باشد، تا SCs 2916 از جمله بازکردن ()، بستن ()، خواندن()، نوشتن() و غیره تولید می شود. بنابراین، نظارت همه ی SCs به طور همزمان برای یک سیستم دشوار است، خصوصا در زمانی که کاربران زیادی برنامه هایشان را اجرا می کنند. به عنوان یک نتیجه، نیاز داریم تا برخی از SCs های ایمنی که مکررا استفاده می شوند را فیلتر کنیم.

برای فهمیدن این مسئله که کدام SCs ، موارد تولید شده ی معمولی بوسیله ی دستور برنامه ی رابط می باشند، مدل استاتیکِ تناوب شرط-تناوب معکوس سند (TF-IDF) استفاده می شود تا اهمیت SCs جداشده ای که در فایل ثبت وقایع کاربر جمع شده اند را تحلیل نماید. رابطه ی بین یک شرط و یک سند در حوزه ی بازیابی اطلاعات همانند رابطه ی بین Sctiو دستور (برای مثال j که ti را تولید می کند) است. تناوب شرط (TF) به کار گرفته شده برای اندازه گیری وزن تناوب یک SC تولید شده بوسیله ی j به صورت زیر تعریف می شود:

$$\mathrm{TF}_{i,j} = \frac{n_{i,j}}{\sum_{k=1}^{k=h} n_{k,j}} \qquad (1)$$

که $n_{i,j}$ تعداد دفعاتی است که ti در زمان اجرای j ارسال می شود، h تعداد SCs مختلفِ تولید شده در زمان اجرای j است و مقسوم علیه $\sum_{k=1}^{k=h} n_{k,j}$ تعداد دفعاتی که همه ی این SCs ارسال می شوند را جمع می کند. تناوب معکوس سند (IDF)، معیار اهمیت ti در میان همه ی دستورات برنامه ی رابط در نظر گرفته شده، به صورت زیر تعریف می شود:

$$\mathrm{IDF}_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|} \qquad (2)$$

که $|D|$ (عدد اصلی D) تعداد کل دستورات برنامه ی واسط در مجموعه ی در نظر گرفته شده است و $\{j : t_i \in d_j\}$ مجموعه ی دستورات برنامه ی رابط dj می باشد که در طول اجرایش در هر عضو ti تولید می کند. وزن TD-IDF برای ti که بوسیله ی j تولید می شود به صورت زیر تعریف می گردد:

$$(\mathrm{TF\text{-}IDF})_{i,j} = \mathrm{TF}_{i,j} \times \mathrm{IDF}_i. \qquad (3)$$

در واقع، وزن TF-IDF به عنوان یکی از روش های وزن دهی در داده کاوی و حوزه های بازیابی اطلاعات، متناسب با تعداد دفعاتی که یک SC در فایل ثبت وقایع کاربر ظاهر می شود، افزایش می یابد و درواقع می تواند اهمیت SC مشخصی را نشان دهد..

Command	No. of SCs	System calls generated
chmod	94	close(19), read(3), open(18), execve(1), access(3), brk(3), umask(1), munmap(2), mprotect(4), mmap2(20), stat64(1), fstat64(17), set_thread_area(1), fchmodat(1)
kill	47	read(4), open(5), close(5), execve(1), getpid(1), access(4), kill(1), brk(3), munmap(2), mprotect(5), mmap2(11), fstat64(4), set_thread_area(1)
date	133	read(70), write(1), open(21), close(22), execve(1), clock_gettime(1), …
rm	102	mmap2(20), read(3), open(18), close(20), execve(1), unlinkat(1), …

جدول I: SCها و تناوبات تولیدشان در طول اجرای چهار دستور خاص

جدول 1 چهار دستور برنامه ی واسط متشکل از chmod، kill، date و rm و SC هایی که تولید کرده اند می باشد را لیست می کند. SC close(19) که در ردیف chmod ظاهر می شود، این موضوع که SC close() در زمانی که chmod اجرا می شود، برای مجموعا 19 دفعه تکرار می گردد را ارائه می کند.

بنابراین، وزن TF برای close() با استفاده از (1) درجایی که تعداد کل دفعاتی که SCs در هنگام اجرای chmod تولید می شود، 94 است، برابر با $0.2021(= 19/94)$ می باشد. وزن TF برای SC kill() در زمان اجرای دستور kill برابر با $0.02128(= 1/47)$ است.

وزن های IDF برای open() و unlinkat() براساس جدول 1 و (2) به ترتیب برابر با $0(= \log 4/4)$ و $2(= \log 4/1)$ هستند، از آنجایی که مورد دوم تنها بوسیله ی دستور rm تولید می شود به معنای در میان چهار دستور است، unlinkat() یک ارائه دهنده ی SC ی rm است و open() بوسیله ی هر چهار دستور مطرح می گردد، این مسئله نشان می دهد که این دستور هیچ کدام از این چهار دستور را ارائه نمی کند. وزن TF-IDF برای SCs باقی مانده می تواند در روشی مشابه محاسبه گردد.

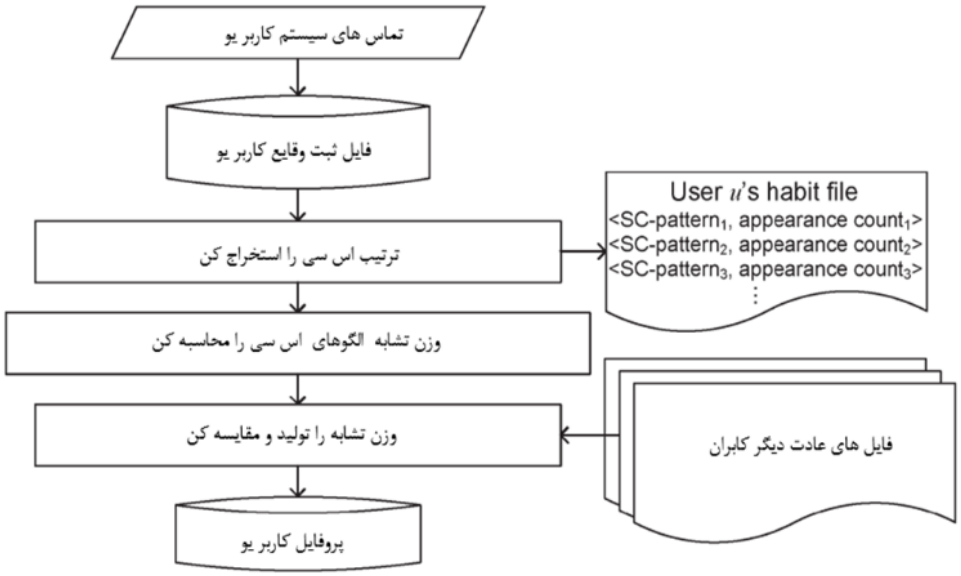
همه ی داده های جمع آوری شده در IIDPS، بوسیله ی یک ابزار داده کاوی تحلیل می شوند (تحلیلگر iData) که قابلیت پیشبینی شدن کلاس و پیش بینی پذیری کلاس دو پارامتر استفاده شده در آن هستند. این دو پارامتر به ترتیب برای ارزیابی کلاس داخلی و وزن های کلاس داخلیِ یک ویژگی در کلاس هایی با ویژگی طبقه بندی شده استفاده می شوند. در ادامه، قابلیت پیش بینی شدن کلاس در تعریف 1 و پیش بینی پذیری کلاس در معادله ی 2 تعریف می شوند.

تعریف 1 (قابلیت پیش بینی شدن کلاس): با توجه به کلاس C و ویژگی مطلق A با $v_1, v_2, v_3, \ldots,$ and v_n به عنوان مقادیر کاندید آن، امتیاز قابلیت پیش بینی شدنِ کلاس C روی A برابر است با vi، که بوسیله ی $P(A = v_i)$ علامت گذاری شده، به عنوان درصدی تعریف می شود که مقدار A در C به اندازه ی آن vi است. مجموع امتیازات قابلیت پیش بینی شدن برای ویژگی A روی همه ی مقادیر کاندید آن در C برابر با 1 است ($\sum_{\forall T(A=v_i) \in C} P(A = v_i) = 1$) که در آن $T(A = v_i)$ نمونه ای از C با A=vi می باشد. امتیاز قابلیت پیش بینی

شدنِ ti با توجه به SC ti و دستور j، درصدی را اندازه می گیرد که به همراه آن $t_i \in Q$ می باشد و Q در آن مجموعه ای از SCs تولید شده در زمان اجرای j است. (به یاد داشته باشید که دیگر دستورات نیز می توانند ti را در زمان اجرایشان تولید کنند). امتیاز قابلیت پیش بینی پذیری ti (بوسیله ی p(ti) علامت گذاری شده) که اجرای j را شناسایی می کند، این معیار را محقق می نماید: $\sum_{\forall t_i \in Q} P(t_i) = 1$

Name	Value	Frequency	Predictability	Predictiveness
Syscall	set_thread_area	1	0.02	0.07
	read	4	0.09	0.02
	open	5	0.11	0.02
	munmap	2	0.04	0.03
	mprotect	5	0.11	0.05
	mmap2	11	0.23	0.03
	kill	1	0.02	1.00
	getpid	1	0.02	0.33
	fstat64	4	0.09	0.01
	execve	1	0.02	0.07
	close	5	0.11	0.01
	brk	3	0.06	0.07
	access	4	0.09	0.05
command	kill	47	1.00	1.00

شکل2: نتایج یادگیری نظارت شده که با فراخوانی ابزار داده کاوی تحلیل گر IData در جهت محاسبه ی امتیازات قابلیت پیش بینی شدن و پیش بینی پذیری SC ()kill تولید شده اند. امتیاز بالای پیش بینی پذیری نشان می دهد که ()kill نشانگر SC برای دستور kill است.



شکل ۳. جریان کنترل تولید پروفایل کاربر

تعریف 2 (پیش بینی پذیری کلاس): امتیاز پیش بینی پذیری مقدار ویژگی vi با توجه به کلاس C و یک ویژگی اصلی A با $v_1, v_2, v_3, \ldots, \text{and } v_n$ به عنوان مقادیر کاندید آن، $P(A = v_i)$، به عنوان احتمالی که یک نمونه ی T با A=vi در C قرار دارد، تعریف می شود. مجموع امتیازات پیش بینی پذیری برای A در همه ی مقادیر کاندید A در میان کلاس هایی در $\{C\}$ برابر با 1 است که $\{C\}$ در آن مجموعه ای از کلاس ها یی می باشد که هر عضو در آن، A را به عنوان یکی از ویژگی هایش دارد ($\sum_{\forall T(A=v_i)\in\{C\}} P(A = v_i) = 1$).. اجازه دهید Sj مجموعه ای از SC های تولید شده بوسیله ی j باشد. امتیاز پیش بینی پذیری یک SC ti (با $P'(t_i)$) علامت گذاری شده) به کار گرفته می شود تا احتمال اینکه ti عضوی از Sj در میان همه ی دستورات برنامه ی واسطِ تولید کننده ی ti در زمان اجرایشان باشد را اندازه گیری کند. امتیاز پیش بینی پذیریِ ti در میان همه ی مولفه های $\{j : t_i \in S_j\}$ برابر است با:

$$\sum_{\forall t_j \in S_j, j \in \{k : t_i \in S_k\}} P'(t_i) = 1. \qquad (4)$$

به طور کلی، نمونه های آموزشِ روند کاوش باید شامل همه ی دستورات برنامه ی واسطِ مرتبط و SC هایی که از طریق این دستورات ایجاد می شوند، باشند. برگه ی کلاس خروجیِ یادگیری نظارت شده ی دستور kill در شکل 2 نشان داده می شود. SC kill() معمولی دارای امتیاز قابلیت پیش بینی کمی است ($0.02(= 1/47)$)، این موضوع، بدین معنی است که تنها یک نمونه از kill() در زمان اجرای دستور kill تولید می شود و SC امتیاز قابلیت پیبینی شدنِ زیادی بدست می اورد ($1(= 1/1)$) (در میان همه ی دستورات کلاس) این موضوع نشان می دهد که kill() یکی از نشانگر های SC برای دستور kill است.

C. سرور کاوش

همان طور که در شکل 3 نشان داده شده، یک سرور کاوش ترتیب SC تولید شده بوسیله ی کاربر u را فایل ثبت وقایع u استخراج می کند، تعداد دفعاتی که یک الگوی خاص SC در فایل ظاهر می شود را می شمارد و نتیجه به شکل (الگوی SC، تعداد حضور) را در فایل عادت u ذخیره می کند

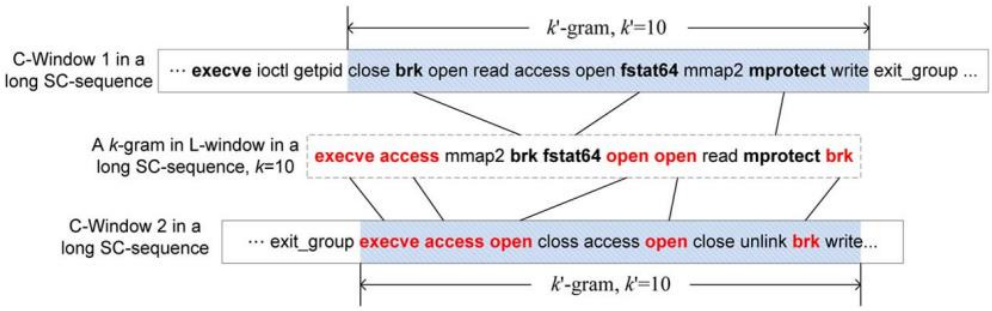
الگوریتم ۱: الگوریتم برای تولید یک فایل عادت

ورودی: فایل ثبت وقایع یو که یو یک کاربر سیستم اساسی است
خروجی: فایل عادت یو

1. G = |log file| - |Sliding window| ;
   /* |Sliding windows|=|L-window|=|C-window| */
2. for (i=0 ; $i \leq G-1$; i++) {
3. for (j=i+1 ; $j \leq G$; j++) {
4. for (each of $\sum_{k=2}^{|\text{Sliding window}|}(|\text{Sliding window}| - k + 1)$ k-grams in current L-window){
5. for (each of $\sum_{k'=2}^{|\text{Sliding window}|}(|\text{Sliding window}| - k' + 1)$ k'-grams in C-window){
6. گرم هارا با بلندترین الگوریتم توالی رایج مقایسه کن
7. اگر الگوی اس سی مشخص شده پیش از این در فایل عادت وجود داشته باشد
8.
9. تعداد الگوی اس سی را افزایش بده
10. الگوی اس سی را با حساب برابر با یک در فایل عادت قرار بده

شکل ۴ . الگوریتم برای تولید فایل عادت کاربر یو

.بعد از آن، تشابه وزن های الگوهای SC محاسبه می شوند تا الگوهای SC که معمولا بوسیله ی همه یا اکثر کاربران استفاده می شوند، فیلتر گردند. سپس، نتیجه ی خروجی با همه ی فایل های عادت دیگر کاربران در سیستم اساسی مقایسه می شود تا الگوهای خاص u، شناسایی شود. در نهایت، تشابه وزن محاسبه می شود تا پروفایل کاربر u را تولید کند. جزییات این روندها در ادامه توضیح داده خواهد شد.



شکل 5: مثالی از مقایسه بین گرم های k در یک پنجره ی L و دو گرم k’ در دو پنجره ی C برای SC 10

1) کاوش عادات متهاجم و کاربر: IIDPS ، SCs های تجمیع شده در فایل ثبت وقایع u را پردازش می کند. این پردازش ها با پنجره ی لغزانی با نام پنجره ی لغزان ثبت وقایع (به اختصار پنجره ی L) انجام داده می شود که برای شناسایی SCs متوالی برای اندازه ی | پنجره ی لغزان| در ترتیب ارائه شده ی آنها استعمال می گردد و SC ها را در پنجره به k گرم تقسیم می کند (k تعداد DC های متوالی |پنجره ی لغزان| است، k=1,2,3,4,...) در این مقاله، |پنجره ی لغزان| برابر با 10 است. علاوه بر این، پنجره ی لغزانی با اندازه ی یکسان (همان تعداد SC ها) ، با نام پنجره ی لغزان مقایسه شده (به اختصار پنجره ی C) به کار گرفته می شود تا دیگر الگوهای SC در فایل ثبت وقایع u را نیز شناسایی کنند. این دفعه، SC های متوالی k که ترتیب ارائه شده ی آنها را حفظ می کند، از پنجره ی C استخراج می شود تا مجموعی از {گرم ها-k' (|پنجره ی لغزان|-k'+1)}، |پنجره ی لغزان| ,k'=2,3,4,... تولید کند.

سرور کاوش الگوریتم 1 که در شکل 4 نشان داده شده است را فراخوانی می کند تا گرم های k' و گرم های k را مقایسه نماید. در ابتدا، همه ی SC های تجمیع شده در فایل ثبت وقایع u به عنوان یک ترتیب SC بلند مورد برخورد قرار می گیرند. در زمانی که همه ی $\sum_{k=2}^{|\text{Sliding window}|}$ گرم ها-k' (|پنجره ی لغزان|-k'+1) از پنجره ی L استنتاج شده اند، با $\sum_{k'=2}^{|\text{Sliding window}|}(|\text{Sliding window}|-k'+1)\ k'\text{-grams}$ که بوسیله ی استفاده از طولانی ترین الگوریتم متداول بعدی لستنتاج شده است، مقایسه شده اند. این عمل تشابه بین دو رشته را با حذف نویز ها آشکار می سازد، پنجره ی C یک ورودی SC (برای مثال شروع اصلی در موقعیت x، و در ادامه به موقعیت شروع در x+1 حرکت می کند.) را به سمت راست انتقال می دهد و مقایسه ی فوق الذکر تازمانی که پنجره ی C آخرین ترتیبِ اندازه ی |پنجره ی لغزان| را پوشش دهد، دوباره اجرا می شود. پنجره ی L بعدی SC را یکی به راست جابه جا می کند و مقایسه ی بالا ادامه می یابد. این روند زمانی ادامه پیدا می کند که این مقایسه در جایی که $G = |\text{log file}| - |\text{Sliding window}|$ باشد کامل شود. لازم به ذکر است که در این مقایسه، پنجره ی L ترتیب دوم SC برای اندازه ی |پنجره ی لغزان| را تا انتها پوشش می دهد (i=G-1) و پنجره ی C شامل آخرین ترتیب SC

برای اندازه ی |پنجره ی لغزان| (j=G) می شود. نظریه ی 1 نشان می دهد که زمان اجرای الگوریتم مقایسه ی (گرم های k، گرم های k’) برابر با $O(n^6)$است.

نظریه ی 1: زمان اجرای الگوریتم الگوریتم 1 $O(n^6)$ که n اندازه ی پنجره ی لغزان است.

اثبات: اجازه دهید $\text{Let } m = |\text{SC-sequence}| - (|\text{Sliding window}| - 1)$، که برابر با تعداد پنجره های لغزانی است که می توانند در ترتیب SC داده شده شناسایی شوند. در ادامه، یک پروفایل کاربر با $|m^*(m-1)/2|$بار فراخوانیِ مقایسه ی جفت (پنجره ی L، پنجره ی C) ایجاد می شود و هر مقایسه ی جفت (پنجره ی L، پنجره ی C) دارای

$$\sum_{k=2}^{|\text{Sliding window}|} (|\text{Sliding window}| - k + 1) * \sum_{k'=2}^{|\text{Sliding window}|} (|\text{Sliding window}| - k' + 1) \qquad (5)$$

دفعه مقایسه ی (گرم k، گرم k’) است. اجازه دهید n برابر با |پنجره ی لغزان| باشد و $l = |\text{SC-sequence}|$؛ مجموع دفعات مقایسه ی (گرم k، گرم k’) که با Ttotal علامت گذاری شده برابر است با:

$$\begin{aligned} T_{\text{total}} &= \frac{(l-n+1)(l-n)}{2} \times \sum_{k=2}^{n}(n-k+1) \times \sum_{k'=2}^{n}(n-k'+1) \\ &= \frac{(l-n+1)(l-n)}{2} \times \frac{n(n-1)}{2} \times \frac{n(n-1)}{2} \\ &\cong \frac{1}{8}(l-n)^2(n)^4. \end{aligned} \qquad (6)$$

این موضوع بدین معنا است که زمان اجرای الگوریتم مقایسه ی (گرم k، گرم k’) برابر با $O(n^6)$ است. قطعا اگر زمان اجرای الگوریتم را در l در نظر گرفته باشیم، $O(l^2)$ خواهد بود.

شکل 5 مثالی از مقایسه ی (گرم k، گرم k’) را ارائه می دهد. مستطیل های خط پیوسته، دو ترتیب SC مقایسه شده را لیست می کند. سطح هاشور خورده یک پنجره ی C است. مستطیل با خط نقطه چین شامل یک ترتیب SC (گرم k) که از یک پنجره ی L با k=10 استخراج شده، می باشد. در مستطیل بالایی (بوسیله ی پنجره ی C 1 علامت گذاری شده است)، SC هایی که با گرم k’ در زمان k’=10 تطابق می کنند عبارتند از brk,fstat64 و

mprotect ((omitting متعلق به یک SC برای سادگی). SC های باقی مانده از جمله close,open,read,access,open,mmap2 و write، نویز هستند و از این رو صرف نظر می شوند. هنگامی که k'=k=10، طولانی ترین توالی رایج بین گرم k و گرم k' در پایین ترین مستطیل (با پنجره ی C 2 علامت گذاری شده است) شامل execve,access,open,open و brk می شود.

2) ایجاد پروفایل های مهاجم و کاربر: در شکل 3 یک پروفایل کاربر متعلق به کاربر، یک فایل عادت است که تعداد حضورِ الگوی SC با تشابه وزن متناظرش جایگزین شده است. در ادامه الگوهای SC با تشابه وزنی کمتر از یک آستانه ی از پیش تعیین شده را حذف می کنیم چون SC های نشانگری که می توانند کاربر را در اینترانت اساسی از دیگر کاربران تمیز دهند، نمی باشند. یک الگوی حمله (یا یک امضا)، که می تواند الگوی خاص مهاجم یا الگوی رایج استفاده شده بوسیله ی مهاجمین باشد، می تواند به روشی مشابه شناسایی شود. به همین شکل، یک الگوی حمله که یک مهاجم مکررا استفاده می کند اما دیگران به ندرت یا اصلا از آن بهره نمی برند به عنوان یکی از نشانگرهای الگوهای حمله ی مهاجم در نظر گرفته خواهد شد و تشابه وزنی بالایی بدست خواهد آورد. در نتیجه، امضاهایی که در یک پروفایل حمله تجمیع شده اند (شکل 1 را ببینید) می تواند به امضاهای رایج و امضاهای مختص به مهاجم دسته بندی شوند. مورد دوم می تواند برای شناسایی این مسئله به کار گرفته شود که در زمانی که یک سیستم محافظت شده، بوسیله ی امضاهای مختص به مهاجم مورد تهاجم قرار می گیرد، چه کسی مهاجم است.

TABLE II
EXAMPLE OF USER HABIT FILES D, SC-PATTERNS T, AND THEIR CORRESPONDING D_i'S

D	T/D_i				
	CS_1/ D_1	CS_2/D_2	CS_3/D_3	...	CS_k/D_k
UH_1	✓	✓			✓
UH_2	✓		✓		✓
UH_3			✓		
...					
UH_{N-1}		✓	✓		✓
UH_N	✓	✓			

با توجه به مجموعه فایل های عادت کاربر $D = \{UH_1, UH_2, \ldots, UH_N\}$ که N برابر با تعداد کاربران است، اجازه دهید تا $T = \{CS_1, CS_2, \ldots, CS_k\}$ مجموعه ای از الگوهای SC بازیابی شده از مولفه های D باشد. اجازه دهید تا $D_i = \{UH'_1, UH'_2, \ldots, UH'_{Mi}\},\ 1 \leq i \leq k$ مجموعه ای از UH باشد که هر UH در آن شامل مولفه ی خاص $CS_i,\ CS_i \epsilon T,\ D_i \subseteq D, \text{ and } |D_i| = M_i$ می شود. مثال های Di در جدول II لیست می شوند که در آن

$$D_1 = \{UH_1, UH_2, UH_N\},\ D_2 = \{UH_1, UH_{N-1}, UH_N\},\ D_3 = \{UH_2, UH_3, UH_{N-1}\},$$

و

$$D_k = \{UH_1, UH_2, UH_{N-1}\}$$

معادله ی 7 که مکررا برای تخصیص یک وزن به یک شرط در حوزه ی بازیابی اطلاعات استفاده می شود، برای دادن تشابه وزنی Wi,j به هر الگوی SC به کار گرفته می شود. Wi,j وزن CSi در UHi است که

$$W_{ij} = \frac{f_{ij}}{f_{ij} + 0.5 + \frac{1.5 \times ns_j}{ns_{\text{avg}}}} \times \frac{\log\left(\frac{N+0.5}{M_i}\right)}{\log(N+1)}$$
$$i = 1, 2, 3, \ldots, k, \text{ and } j = 1, 2, 3, \ldots, N \qquad (7)$$

که در آن fi,j تعداد حضور CSi در UHi است، nsj تعداد کل الگوهای SC تجمیع شده در HUj می باشد و nsavg تعداد میانگین الگوهای SC در D است و $\log((N+0.5)/M_i)/\log(N+1)$ تناوب معکوس پروفایل مشخصات (ICPF) می باشد.

D.سرور شناسایی

سرور شناسایی، SC های ارسال شده به هسته را از طریق کاربر اساسی u در زمانی که u دستورات برنامه ی واسط را اجرا می کنند، دریافت می نماید و SC ها را در فایل ثبت وقایع u ذخیره می گرداند.

الگوریتم ۲: شناسایی یک متجاوز با یک مهاجم داخلی

ورودی: سی ان سی های ورودی کنونی یو(هر دفعه تنها یک اس سی ورودی است) و پروفایل های کاربری همه ی

خروجی: یو به عنوان متجاوز یا یک مهاجم شناخته شده ی داخلی مظنون است

1. $NCS_u = \emptyset$;
2. while (receiving u's input SC, denoted by h) {
3. $NCS_u = NCS_u \cup \{h\}$;
4. if ($|NCS_u| > |$Sliding window$|$) {
5. L-window = Right(NCS_u, | Sliding window|); /* Right(x, y) retrieves the last L-window of y from x */
6. for ($j = |NCS_u| - |$Sliding window$|$; $j > 0$; j--) {
7. C-window = Mid ($NCS_u, j, |$Sliding window$|$);/* Mid (x, y, z) retrieves a sliding window of size z beginning at the position of y from x */
8. گرم هارا با استفاده از منطق مقایسه ی به کاربرده شده در الگوریتم ۱ مقایسه کن تا ان اچ اف تولید کنی
9. for (each user g, $1 \le g \le N$)
10. امتیاز تشابه سیم (آی، جی) را با فراخوانی معادله ی۸ بین ان اس سی ورودی مهاجم است یا نگه دارنده ی حساب
11. if (($|NCS_u|$ mod paragraph size) == 0) { /* paragraph size = 30, meaning we judge whether u is an attacker or the account holder for every 30 input SCs */
12. امتیاز تشابه را برای همه ی کاربران دسته بندی کن
13. آستانه ی ۱ باند پایینی از پیش تعریف شده ی نرخ قطعی پروفایل کاربری کابر یو است در حالی که آستانه ی ۲ باند بالایی از پیش تعریف شده ی نرخ قطعی میانگین پروفایل مهاجم
14. به جای خود یو، به مدیر سیستم در مورد اینکه یو مهاجم مورد ظن است هشدار بده

شکل ۶. سرور شناسایی تشخیش می دهد که آیا یو یک متجاوز داخلی یا یک مهاجم است یا نه

در ادامه، سرور سعی می کند تا این قضیه که آیا u نگهدارنده ی حساب اساسی است یا خیر را با محاسبه ی امتیاز تشابه میان SC های جدیدا تولید شده در ورودی های کنونی u (در فایل ثبت وقایع u) و عادات کاربرد (امضاهای کالبد سنجی و همچنین الگوهای رفتار) ذخیره شده در پروفایل کاربر u در جهت اعتبارسنجی u، محاسبه نماید. لازم به ذکر است که SC های جدیدا تولید شده با NSCu علامت گذاری شده اند. مدل Okapi که برای محاسبه ی امتیاز تشابه بین UHj پروفایل کاربر برای کاربرِ j و ترتیب SC ورودی کنونی یک کاربر ناشناس u به کار می رود، بوسیله ی Sim(u,j) علامت گذاری شده و به صورت زیر تعریف می شود:

$$Sim(u,j) = \sum_{i=1}^{p} F_{iu} \cdot W_{ij} \qquad (8)$$

که در آن p تعداد الگوهای SC پدیدار شده در هر دو مورد NDCu و UHj است، Fiu تعداد حضور الگوی SC ِi تجمیع شده در NCSu می باشد و Wij که بوسیله ی فراخوانی 7 تولید می شود برابر با تشابه وزنی i در UHj است. هر چقدر که Sim(u,j) بالاتر باشد، احتمال اینکه u همان فرد j که NCSu را ارسال کرده باشد، بیشتر است.

الگوریتم 2 نشان داده شده در شکل 6یک مهاجم داخلی را شناسایی می کند. سرور شناسایی نیاز دارد تا فایل عادت موقت u را بوسیله ی تحلیل NCSu ایجاد کند، به طوریکه Fiu برای محاسبه قابل دسترسی باشد. این مفهوم مشابه با عملکرد سرور کاوش است با این تفاوت که مقایسه ی بین پنجره ی L و پنجره ی C هر بار در زمانی که SC ورودی از طریق u باشد، از عقب به جلو است برای مثال در زمانی که پنجره ی L متشکل از آخرین SC های|پنجره ی لغزان| باشد، پنجره ی C یک SC از پنجره ی L را به چپ انتقال می دهد تا گرم های k و گرم های k' که به طور منحصر به فرد از آنها استنتاج شده اند را مقایسه کند. در ادامه، دو پنجره برای هر جابه جایی به چپ در پنجره ی C مقایسه می شوند. این مقایسه تا زمانی که پنجره ی C اولین SCs |پنجره ی لغزان| متعلق به NCSu را پوشش دهد، ادامه می یابد.

در ادامه دو مثال داده شده است تا نشان داده شود کخ پنجره ی L و پنجره ی C در الگوریتم های 1 و 2 چگونه مقایسه می شوند. اجازه دهید $\{1, 2, 3, 4, 5, 6, 7, 8\}$ ترتیب SC داده شده باشد و اجازه دهید اندازه ی پنجره ی لغزان (|پنجره ی لغزان|) برابر با 5 باشد. مقایسه ی انجام شده بوسیله ی سرور کاوش در الگوریتم 1 در زیر نشان داده می شود. $\langle 12345, 45678\rangle_3 \rightarrow \langle 23456, 34567\rangle_4 \rightarrow \langle 23456, 45678\rangle_5 \rightarrow \langle 34567, 45678\rangle_6$

$\langle 12345, 23456\rangle_1 \rightarrow \langle 12345, 34567\rangle_2 \rightarrow$ که شکل (X,Y)z نشان می دهد که ترتیب SC X که در پنجره ی L قرار داشته و ترتیب SC Y که در پنجره ی C تجمیع شده ، در z امین قیاس در الگوریتم 1 مقایسه می شوند. لازم به ذکر است که در الگوریتم 1 $|X| = |Y| = |\text{Sliding window}|$.

در زمانی که الگوریتم 2 بوسیله ی سرور شناسایی فراخوانی می شود، پنجره ی لغزان به سمت چپ جابه جا می شود (عقب رفتن) تا ترتیبات SC را روی هر SC ورودی مشخص نماید. اگر ورودی های کاربر کمتر یا برابر با |پنجره ی لغزان| (یعنی 5) باشد، هیچ عملیاتی انجام داده نمی شود. در زمانی که کاربر ششمین SC را وارد کرده باشد،

پنجره ی L شامل SC های 2 و 3 و 4 و 5 و 6 می باشد و پنجره ی C ، SC های 1 و 2 و 3 و 4 و 5 را پوشش می دهد.(یعنی $\langle 23456, 12345\rangle_1$). بعد از اینکه هفتمین SC وارد می شود، مقایسه $\langle 34567, 12345\rangle_2$ $\rightarrow \langle 34567, 23456\rangle_4$ خواهد بود. سرور شناسایی هنگام دریافت هشتمین SC مقایسه های $\langle 45678, 34567\rangle_6 \rightarrow \langle 45678, 23456\rangle_5 \rightarrow \langle 45678, 12345\rangle_3$ را اجرا می کند. در این الگوریتم، همه ی الگوهای SC شناسایی شده، به جای فایل عادت j که حساب j همانی است که u به آن وارد شده، ورودی NHFu فایل جدید عادت u هستند.

با توجه به مثال ها می بینیم که این دو الگوریتم جفت های مقایسه ی یکسانی را حاصل می کنند، برای مثال $\langle 12345, 23456\rangle_1$ به $\langle 34567, 45678\rangle_6$ با الگوریتم 1 و $\langle 23456, 12345\rangle_1$، $\langle 45678, 34567\rangle_6$, $\langle 34567, 12345\rangle_2$, $\langle 34567, 23456\rangle_4$, $\langle 45678, 12345\rangle_3$ and $\langle 45678, 23456\rangle_5$ بوسیله ی الگوریتم ، این موضوع بدین معنی است که این دو الگوریتم منطق عکس نقیض مقایسه ی یکسانی را به کار می برند و NHFu یکسانی تولید و NCSu یکسانی ارائه می کنند. فرضیه ی 2 نشان می دهد که زمان اجرای الگوریتم شناسایی یک متجاوز یا مهاجم داخلی برابر با $O(gn^5)$ است.

نظریه ی 2: زمان اجرای الگوریتم شناسایی یک متجاوز یا یک مهاجم داخلی برابر با $O(gn^5)$ است که n اندازه ی پنجره ی لغزان و g تعداد کاربران قانونی است.

اثبات: اجازه دهید n برابر با |پنجره ی لغزان| و g تعداد کاربران قانونی باشد. هر دفعه که کاربر اساسی u اخیرا یک SC را وارد کند و تعداد SC های وارد شده بزرگتر از n باشد، آخرین SC های n به عنوان یک پنجره ی n شناسایی خواهند شد. اجازه دهید $|NCS_u| = l$ باشد، اکنون در NCSu، مجموعی از l-n+ پنجره ی لغزان می تواند یافته شود، این موضوع بدین معنی است که زمان مقایسه ی پنجره ی لغزان بین پنجره ی L و l-n پنجره ی C باقی مانده برابر با l-n است. براساس نظریه ی 1، برای هر جفت مقایسه ی پنجره ی لغزان، $(n(n-1)/2) \times (n(n-1)/2)$ از مقایسه ی (گرم k، گرم k') وجود دارد. اجازه دهید Tcomp تعداد کل مقایسه های (گرم k، گرم k') در NCSu باشد:

$$T_{\text{comp}} = (l-n) \times \frac{n(n-1)}{2} \times \frac{n(n-1)}{2}. \qquad (9)$$

از آنجایی که g کاربر در IIDPS وجود دارد، زمان کل مقایسه های (گرم k، گرم k’) برابر است با:

$$T'_{\text{total}} = g \times T_{\text{comp}} = g \times (l-n) \times \frac{n(n-1)}{2} \times \frac{n(n-1)}{2}. \quad (10)$$

این موضوع نشان می دهد که زمان اجرای الگوریتم برای الگوریتم 2 برابر با $O(gn^5)$ است.

در این مقاله، |پنجره ی لغزان| محدود به کمتر از 10 است. در نتیجه، زمان های مصرف شده برای مقایسه های (گرم k، گرم k’) بوسیله ی الگوریتم های 1 و 2، هر دو کوتاه هستند.

تعریف 3 (نرخ قطعی): با توجه به NHFu متعلق به کاربر ناشناس u و امتیاز تشابه بین NHFu و پروفایل کاربری UHq متعلق به کاربر q ($1 \le q \le N$) که N در آن تعداد کاربران است، نرخ قطعی برای سرور شناسایی برای $r(1 \le r \le N)$ (با x علامت گذاری شده است) در میان N کاربر به صورت $x = (N - rank(r))/N 0 \le x \le 1$ تعریف می شود، که در آن $rank(r)$ برابر با مرتبه ی کاربر r از بالا است. لازم به ذکر است که این مرتبه پس از دسته بندی شدن امتیازات تشابه در جایی که حساب r، حسابی است که u به ان وارد می شود، می باشد.

نرخ قطعیِ پروفایل کاربری r (x) باید در بالای $y * 100\%$ باشد که y آستانه ی تصمیم است. اگر این مسئله وجود نداشته باشد، u به جای r به عنوان یک متجاوز در نظر گرفته می شود.

1) انواع حمله: سه نوع تهاجم در این مقاله ایجاد می شود. حمله ی نوع I به عنوان موقعیت تعریف می شود که یک کاربرؤ گروه خاص، یک SC که اعضای گروه از استعمال آن منع شده اند را ارسال می کند. حمله ی نوع II، حمله ای است که یک SC حساس را ارسال می نماید که به عنوان موردی تعریف می شود که می تواند داده حساس یا تنظیمات سیستم را حذف کند یا تغییر دهد. یک حمله ی نوع III متشکل از الگوهای حمله ی سطح SC (الگوهای SC) هستند که هر کدام از آنها به عنوان یک مرحله از حمله مورد برخورد قرار می گیرد. در حقیقت، یک مهاجمِ ترکیب کننده ی SC خاص می تواند در بعضی اوقات، با موفقیت به سیستم نفوذ کند.

به طور کلی، الگوهای SC که یک حمله ی سرریز بافر را تولید می کنند، معمولا شامل یک SC نرمال با پارامتری طولانی تر از طول تعریف شده ی بافر هستند. یک مهاجم با چنین نوعی از حمله می تواند با استفاده از یک حمله ی

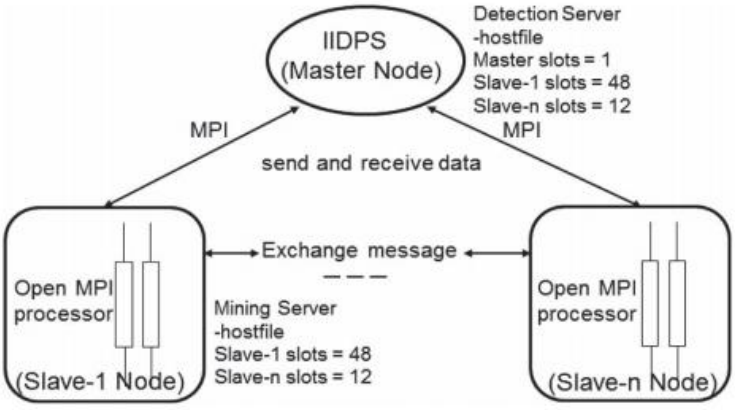
نوع III به یک سیستم نفوذ کند و در ادامه می تواند برای حمله به سیستم، به حق دسترسی بالاتری دستیابد (برای مثال، کرک کردن رمز عبور یا کسب مزیت روت کردن). SC هایی که حمله ی نوع III را ارسال می کنند، الگوی حمله ی چندمرحله ای نامیده می شوند. اساسا، یک الگوی حمله ی تک SC دسته ای خاص از الگوهای حمله ی چند مرحله ای در زمانی است که تعداد مراحل و تعداد SC ها برابر با یک باشد.

حمله های نوع I و نوع II (که در لیست کلاس محدود SC گروه تعریف شده است) می تواند از طریق مقایسه ی یک ورودی SC مستقیما با SC های جمع شده در این لیست ، شناسایی شود. حمله ی نوع III می تواند از طریق فراخوانی الگوریتم 2 تعریف شود.

همان طور که پیش از این مطرح شد، همه ی الگوهای SC مهاجمین در شکل یک پروفایل نیز ارائه می شوند. می توانیم با توجه به NCSu مشخص کنیم که آیا NCSu شامل الگوهای حمله ی مختص به مهاجم می شود سا نه. این عمل با به کار بردن فرایندی مشابه به فرایند داوری در مورد اینکه آیا u نگهدارنده ی حسابی که u به آن وارد شده است یا نه، می باشد. بعد از محاسبه ی امتیازات تشابه بین NHFu متناظر و پروفایل های کاربریِ همه ی کاربران، اگر نرخ قطعیِ پروفایل مهاجم بیشتر از y*100% باشد، در ادامه مشکوک می شویم که u مهاجم است و IIDPS یک پیام هشدار syslog تولید می کند و همراه با ID کاربر یک پیغام "نامطمئن" را پاسخ می دهد تا ناظر و فیلتر SC را مطلع سازد، این عمل کاربر را به طور بهنگام از سیستم جدا می کند تا از حمله ی پیوسته ی این کاربر به سیستم محافظت شده جلوگیری کند. البته، اگر مهاجم یک کاربر داخلی نباشد، یک سیستم رهگیری یا سیستم های شناسایی دیگر ملزوم می شود.

E. شبکه ی محاسباتی

شبکه ی محاسباتی شامل مجموعه ای از کامپیوترهای منفرد با اتصال داخلی است که به عنوان یک منبع محاسباتی یکپارچه کار می کنند.



شکل 7: SMP با معماری MPI باز

در این مقاله، سرور کاوش و یک سرور شناسایی در شبکه پیاده سازی می شوند تا پردازش داده را سرعت بخشند. محاسبه ی خوشه، عملیات های محاسباتی بزرگی را به زیرفعالیت های کوچکتری تقسیم می کند که از طریق به کار بردن چندین پردازش گر کامپیوتری به طور همزمان اجرا می شوند. هر پردازش گر مسئول اجرای بخشی از محاسبات است، (برای مثال، پردازش زیرمجموعه ای از داده ی ورودی). معمولا، گره ارباب، پردازش توزیع شده را تعدیل می کند و نتایج پردازش همه ی پردازش گرها را در جهت تکمیل تمام محاسبات، تجمیع می کند. شکل 7 خوشه ی چند پردازشی متقارن (SMP) را با معماری رابط باز عبوز پیام (MPI) نشان می دهد. تابع MPI یک محیط استاندارد شده و قابل جابه جایی را برای پردازنده ها فراهم می کند تا پیام ها را به طور مشترک تبادل کند. توابع سرور کاوش از طریق شبکه ی پردازش گرها با یک فایل میزبان که منابع محاسبه را برای سرور کاوش تعیین می کند، بدست می آیند. سرور شناسایی روی گره اربابِ شبکه ی محاسباتی با که منابع محاسبه را برای سرور شناسایی تعریف می کند، پیاده سازی می شود.

F. افزایش سرعت موازی

افزایش سرعت S برای سیستم موازی به صورت زیر تعریف می شود:

$$S = \frac{\text{The time consumed by a single processor to finish a job}}{\text{The time spent by a parallel system to finish a job}}$$

(S برابر است با زمان مصرف شده بوسیله ی یک پردازنده برای اتمام کار تقسیم بر زمان گذرانده شده بوسیله ی سیستم موازی برای اتمام کار).

S پارامتری است که برای اندازه گیری عملکرد سیستم موازی با n پردازنده استفاده شده است. به طور کلی، افزایش سرعت موازی به صورت زیر تعریف می شود:

$$S = \frac{1}{f + \max(f'_1, f'_2, \ldots, f'_n)} \qquad (12)$$

زمان کل لازم برای یک برنامه در جهت انجام دادن یک کار 1 است، f قسمت ترتیبی مصرف شده بوسیله ی یک پردازنده است، f'_i زمان سپری شده بوسیله ی پردازنده ی i در جهت انجام دادن کار محوله ی خود است، $\max(f'_1, f'_2, \ldots, f'_n)$ زمان پاسخ برای n پردازنده است و $\sum_{j=1}^{n} f'_j = 1 - f$ که در آن (1-f) تناسبی است که می تواند بوسیله ی به کاربردن n پردازنده ی موازی، موازی شود (از موازی شدن بهره ببرد). علاوه بر این، براساس قانون Amdahl، S محدود است به:

$$S \leq \frac{1}{f + \frac{1}{n}(1 - f)} \qquad (13)$$

که در آن اگر n پردازنده ی به کارگرفته شده برای انجام دادن کار، یکسان باشند، $(1 - f)/n$ زمان پاسخ است .

No. of processing cores	SC-sequence length			
	64	128	256	512
1	16.34	123.53	635.02	2910.70
36	0.45	2.11	8.94	40.45
48	0.43	1.79	6.92	28.19
60	0.61	1.88	5.98	23.14

جدول III: زمان پاسخ الگوریتم 1 با استفاده از محاسبات موازی (واحد: ثانیه)

با توجه به این محدودیت، هر چقدر که n به سمت بی نهایت میل کند، افزایش سرعت بیشینه به 1/f نزدیک می شود. در عمل، اگر f مولفه ای کوچک باشد، تناسب عملکرد به قیمت هر چقدر که n افزایش پیدا کند، به سرعت افت می کند.

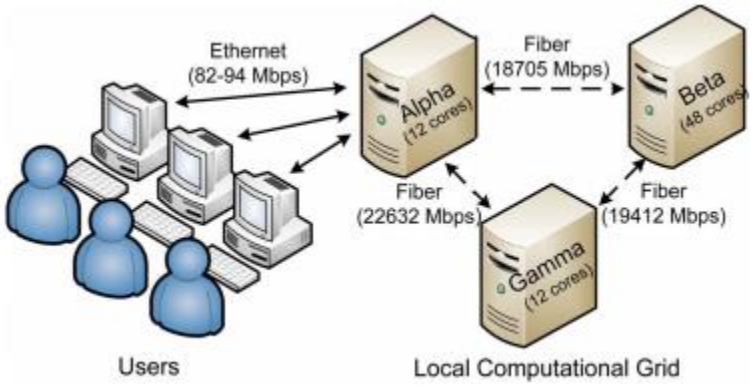
برمبنای نظریه ی تخمین Minsky، بیش از هزاران پردازش، نرخ های افزایش سرعت سرور شناسایی متناسب با $\log_2 n$ هستند که در آن n تعداد پردازنده ها می باشد. این موضوع بدین معنا است که سرور شناسایی که قسمت های ترتیبی بسیار زیادی دارد، نمی تواند به طور موازی اجرا شود. علاوه بر این، مقدار زیادی از پیغام ها در میان پردازنده های موازی تحویل داده می شوند. در زمانی که عملکردِ شبکه ی محاسبات محلی به کار گرفته شده، معلوم باشد، می توانیم در ادامه بر مبنای طول ترتیب SC داده شده تعداد مناسبی از گره های شبکه را مشخص کرده و به کار ببریم تا به عنوان سرور کاوش و سرور شناسایی برخط عمل کنند.

همان طور که در شکل 1 نشان داده شده، در IIDPS، گره ارباب Pm برای شبکه ی محاسباتی محلی، تعداد کل جفت های مقایسه ی (گرم k، گرم k') را محاسبه می کند، اندازه ی داده ی محاسبه را برای هر گره برده ی Pi محاسبه کرده و بخش فایل ثبت وقایع را شناسایی می نماید، این اندازه بوسیله ی $LS(i)$ برای $P_i, 1 \leq i \leq NP$ علامت گذاری شده است که NP تعداد گره های برده ی قابل دسترسی استو در هر مقایسه ی (گرم k، گرم k')طول های گرم k، گرم k' برابر هستند و گرم k، گرم k' هر دو از NCSu بازیابی شده اند.. $k = 2, 3, \ldots, |\text{Sliding window}|$ که در آن $|\text{Sliding window}| = |L - \text{window}| = |C - \text{window}|$. در ادامه، Pm، LS(i) را در زمان دریافت NCSu به Pi ارسال می کند، Pi جفت های مشخص شده ی گرم k و گرم k' در LS(i) را مقایسه می کند و الگوهای SC شناسایی شده و تعداد حضور متناظر Pm آنها را باز می گرداند. هنگامی که Pm نتایج را از گره های برده دریافت می کند، تعداد حضورِ الگوهای SC را جمع می کند تا پروفایل کاربر را ایجاد نماید. بعد از جمع کردن پروفایل همه ی کاربران، Pm ، وزن تشابه هر الگوی SC را برای پروفایل کاربری هر کاربر بوسیله ی فراخوانی 7 محاسبه می کند و همه ی پروفایل های کاربری را در هر کامپیوتر موازیِ شبکه ی محاسباتی محلی توزیع می کند. زمان مورد نیاز بوسیله ی Pm به منظور انتظار برای تکمیل شناسایی الگوهای SC و تعداد حضورشان، بوسیله س پردازش موازی به شدت کاهش می یابد. جدول III زمان پاسخ الگوریتم 1 را با استفاده از محاسبه ی موازی لیست می کند.

4. آزمایشات

برای تصدیق عملیاتی بودن و دقت IIDPS، سه آزمایش اجرا شد. اولین مورد آستانه ی نرخ قطعی را بین پروفایل کاربری ایجاد شده برای u و هر کدام از پروفایل های کاربری دیگر کاربران تعریف کرده است. مورد دوم دقت را برای سرور شناسایی برخط در زمانی که NCSu بوسیله ی u ارسال شده باشد را مورد مطالعه قرار می دهد. مورد سوم IIDPS را با چندین IDS مدرن که بر پایه ی میزبان هستند (HIDSs) مقایسه کرده است.

شکل 8 ، پیکربندی تجربی را نشان می دهد که متشکل از 12 کاربر است: یک کامپیوتر محافظت شده، آلفا اسمی و دو عضو دیگر از شبکه ی محاسباتی، بتا اسمی با 48 هسته ی پردازش و گاما با 12 هسته ی پردازش. همه ی کامپیوترها با سیستم عامل لینوکس فعالیت می کنند و اندازه ی حافظه برای هر کامپیوتر حداقل 25 GB می باشد. پهنای باند اندازه گیری شده و انواع شبکه میان دو نقطه ی تصادفیِ شبکه ی محاسباتی نیز در شکل 8 نشان داده می شود.



شکل 8: پیکربندی منطقیِ بستر آزمونِ استفاده شده در این مقاله

فیلتر و ناظر SC در بستر IIDPSD، در ابتدا در کامپیوتر آلفا نصب می شود تا فایل ثبت وقایعِ هر کار بر را در طول زمان تناوب بین یکم فوریه ی 2014 و 31 ام جولای 2014 بدست آورد. SC ها از 12 دسته ی مختلف کاربران به عنوان داده ی تجربی جمع اوری می شوند. این داده ها عبارتند از اوراکل سیستم (روت)، صف بندی پیغام (mq)، رزرو کردن، بلیط گرفتن، مالی، استراتژی عملیاتی (os)، پشتیبانی کردن، مدیریت پیکربندی (cm)، کاربرد شبکه،

اصل تجارت و کاربران بازرس. مجموعا 193613 SC از 12 فایل ثبت وقایع که طول پنجره ی لغزان در آن 10 است، جمع آوری شده است.

A. آستانه ی نرخ قطعی

75% رکوردها برای مشخص کردن آستانه ی نرخ قطعی به عنوان داده ی آموزش از هر کدام از 12 فایل ثبت وقایع انتخاب می شود تا 12 پروفایل کاربری متناظر تولید کند و 25٪ باقی مانده داده های آزمون هستند. این آزمایش ده بار انجام داده شد. با توجه به ترتیب SC ورودی کنونیِ کاربر u (NCSu)، اگر امتیاز تشابه بین پروفایل کاربری NCSu و u در بالای y*100% رتبه بندی می شود. همان طور که پیش از این ذکر شده، آستانه ی نرخ قطعی در میان امتیاز تشابه بین NCSu و هر کدام از پروفایل های کاربری 12 کاربر، y است. نتایج تجربیِ نرخ قطعی در جدول IV نشان داده می شوند. میانگین نرخ قطعی ر این جدول 0.9312 است. بنابراین، آستانه ی y روی 0.9 تنظیم شده است تا این مسئله که آیا کاربر اساسی u برای یک حساب، نگهدارنده ی همان حساب است یا نه را ارزیابی کند. علاوه بر این، اندازه ی پاراگراف به صورت $3 * |\text{Sliding window}|$ (SC 30) تعریف می شود. این موضوع بدین معنا است که سرور شناسایی، نرخ قطعی برای u در زمانی که طول SC های ورودی کنونی u به k برابری طول پاراگراف برسد را حساب می کند (k یک عدد صحیح مثبت است). هدف این کار اجتناب از محاسبه ی پیوسته ی رتبه بندی در هر ورودی SChsj.

شکل 9 نرخ قطعی بین ورودی های کنونی u و هر دو پروفایل کاربری مهاجم و u را نشان می دهد. ID ، Logid ترتیب ثبت وقایع است، uid، ID کاربر است (u با ID=1000)، ncslength طول ترتیب SC ورودی می باشد (که برابر است با $30 * k, k = 1, 2, 3, \ldots$)، حوزه ی مهاجم، نرخ قطعی را برای امتیاز تشابه بین ورودی کنونی u و امضاهای تهاجم (الگوی حمله) در پروفایل مهاجم را رکورد می کند و حوزه ی کاربر نرخ قطعی را برای امتیاز تشابه بین ورودی کنونی u و الگوهای SC تجمیع شده در پروفایل کاربری u ارائه می کند. اگر حوزه ی کاربر کمتر از 90٪

آستانه ی از پیش تعریف شده باشد (0.9=) یا حوزه ی مهاجم بیشتر از 90% آستانه باشد، کاربر به عنوان یک مهاجم مورد ظن قرار می گیرد.

جدول IV: نرخ های قطعی 12 کاربر از طریق مقایسه ی متقاطعِ فایل های ثبت وقایع کاربران (25٪ از داده ی آزمون)

User log file	Decisive rate	Standard Deviation
oracle.log	0.9333	6.24
cm.log	0.9249	5.83
mq.log	0.9166	5.27
business.log	0.9333	5.0
web.log	0.9249	5.83
reservation.log	0.9416	5.34
os.log	0.9166	5.27
financial.log	0.9083	4.49
ticketing.log	0.9499	5.53
root.log	0.9499	4.09
backup.log	0.9166	3.37
audit.log	0.9583	5.59
Average decisive rate = 0.9312 Average Standard Deviation=5.15		

نرخ تعیین میانگین برابر است با 0.9312

انحراف استاندارد میانگین برابر است با 5.15

```
logid=0, uid=1000, ncslength=30, attacker= 0.3211, user= 0.9788
logid=0, uid=1000, ncslength=60, attacker= 0.4003, user= 0.9634
logid=0, uid=1000, ncslength=90, attacker= 0.2387, user= 0.9875
logid=0, uid=1000, ncslength=120, attacker= 0.6541, user= 0.8924 -> alert
logid=0, uid=1000, ncslength=150, attacker= 0.9153, user= 0.8073 -> alert
```

شکل 9: سرور شناسایی، نرخ های قطعی را در زمانی که تعداد ورودی های کنونی SC به k برابر اندازه ی پاراگراف برسد، محاسبه می کند که SC برابر با سی است و k=1,2,3,....

همان طور که نشان داده شده، در زمانی که طول ترتیب SC برابر با 120 باشد، user = 0.8924 خواهد بود که کمتر از 0.9 می باشد. بنابراین، سرور شناسایی به مدیر سیستم هشدار می دهد که کاربر کنونی می تواند نگهدارنده ی حساب نباشد. در زمانی که ncslength =150 باشد، attacker = 0.9153 خواهد بود که بزرگتر از 0.9 است

و user = 0.8073 می شود که کمتر از 0.9 است. بنابراین، سرور شناسایی به مدیر سیستم هشدار می دهد که کاربر خودی است.

B. دقت شناسایی

در آزمایش دوم، مجددا به طور تصادفی 75٪ از تاریخچه ی استفاده ی کاربران را به عنوان داده ی آموزش برای ایجاد 12 پروفایل کاربری در نظر می گیریم و 25٪ باقی مانده داده ی آزمون هستند تا ورودی های برخطِ کاربر u را شبیه سازی کنند. هدف این کار بدست اوردن امتیازات تشابه میان u و هم ی کاربران است به طوریکه IIDPS بتواند در مورد اینکه کاربر u در اینترانت چه کسی است داوری کند.

Account ID	|Training data|	|Habit file|	|User profile|
root	9,531	122,805	73,683
oracle	10,544	126,710	77,293
mq	8,644	109,825	63,698
reservation	16,357	142,155	85,293
ticketing	18,288	145,380	79,959
financial	19,868	153,215	90,396
os	10,202	118,215	72,111
backup	5,678	29,224	16,365
cm	8,643	103,705	64,297
web	11,964	110,203	69,659
business	15,738	139,659	82,398
audit	9,753	105,490	66,458

جدول V: آمار 12 پروفایل کاربری تولید شده از طریق سرور کاوش به صورت موازی

اطلاعات اماری برای 12 پروفایل کاربریِ تولید شده بوسیله ی سرور کاوش در جدول V تولید می شود. در این جدول ID حساب، ID کاربر را نشان می دهد، |داده ی آموزش| تعداد SC ها در داده ی اموزش و |فایل عادت| تعداد الگوهای SC (به جای SC ها) تجمیع شده در فایل عادت می باشد و |پروفایل کاربری| تعداد الگوهای SC تجمیع شده در پروفایل کاربریِ کاربر است. در حدود 40٪ از الگوهای رایج کاربران حذف شدند چون وزن های تشابه آنها کمتر از آستانه ی از پیش تعیین شده ی 0.001 است.

آمار دقت شناسایی کاربر در جدول VI لیست می شوند. در این جدول "شماره ی پاراگراف" تعداد دفعات محاسبه ی نرخ تعیین در زمان ارزیابی داده ی آزمون وقتی که اندازه ی پاراگراف 30 SC باشد، است، "تعداد دفعات نگهدارنده

ی حساب بودن" تعداد میانگینِ مقدار ده برابر آن است که نرخ قطعی بزرگتر از آستانه ی از پیش تعیین شده باشد، و "تعداد دفعات مهاجم بودن" تعداد دفعات میانگینِ مقدار ده برابر آن است که نرخ قطعی کنتر از آستانه ی از پیش تعیین شده باشد (تعداد دفعاتی که سرور شناسایی به مدیر سیستم هشدار می دهد که کاربر کنونی مهاجم است). "دقت شناسایی"، دقت تصدیقِ هویت کاربر است. سیستم مدیریت پایگاه داده مورد استفاده قرار گرفته برای ذخیره و پردازش داده، SQLite است. مجموعا 100-GB DRAM و 10-TB فضای هارد دیسک در IIDPS وجود دارد و کمتر از 1 GB برای ذخیره سازی پروفایل های کاربران استفاده می شود.

C. مقایسه با دیگر HIDS ها

IIDPS یک HIDS معمولی است که اتفاقات داخلی سیستم را نظارت می کند. یک HIDS معمولا اطلاعات را جمع آوری و تحلیل می کند و بوسیله ی کاربران در سیستم استفاده می شود تا تهدیدات محتمل را شناسایی کند. برای تحقیق در مورد قابلیت های تهاجم سیستم، IIDPS در آزمایش سوم با چهار HIDS مقایسه می شود (OSSEC,AIDE,SAMHAIN و Symantec CSP). OSSEC داده ی ثبت وقایع را تحلیل می کند، یکپارچگی فایل را کنترل می نماید، سیاست های تنظیم را نظارت، rootkit ها را شناسایی می کند و حمله های مورد ظن را هشدار داده و فعالانه پاسخ می دهد. این سیستم دارای مامور یادگیری همکاری است که فایل های ثبت وقایع را برای شناسایی حمله های ساده ی نوع III تحلیل می کند. AIDE (سیم تله) فایل و یکپارچگی اداره کنندگان را برای یک فاصله ی زمانی از پیش تعیین شده که از طریق مدیر سیستم داده شده، کنترل می کند. SAMHAIN کنترل یکپارچگی فایل و نظارت و تحلیل فایل ثبت وقایع را فراهم می کند. همچنین rootkit ها را شناسایی، درگاه ها را نظارت، قابلیت های اجرای مزیت ریشه ی سرکش را شناسایی می کند و پردازش های مخفی که حمله های نوع I و نوع II استفاده می کنند را تشخیص می دهد. Symantec CSP به عنوان یک ابر مجموعه ی IDS میزبان سیمانتیک می تواند بخشی از حمله های نوع III و حمله های DDoS ارسال شده از طرف سیستم را شناسایی کند. این سیستم حمله های DDoS ارائه شده بوسیله ی یک سیستم را از طریق نظارت ترافیک ورودی سیستم تشخیص

می دهد. بهر حال، این موضوع می تواند هشدارهای غلطی را تحریک کند خصوصا در زمانی که کاربران یا برنامه های معمولی داده را در اینترنت بارگذاری می کنند.

Account ID	No. of Paragraph	Times of being an account holder	Times of being an attacker (Alarm)	Detection accuracy
root	106	100.03	5.97	94.36%
oracle	117	110.2	6.8	94.18%
mq	96	90.43	5.57	94.19%
reservation	182	171.89	10.11	94.44%
ticketing	203	191.58	11.42	94.37%
financial	220	208.78	11.22	94.90%
os	113	105.67	7.33	93.51%
backup	63	56.68	6.32	89.97%
cm	96	91.02	4.98	94.81%
web	132	124.81	7.91	94.55%
business	174	163.77	10.23	94.12%
audit	108	105.89	2.11	98.04%
Average	-	-	-	**94.29%**

جدول VI: دقت مشخصات کاربرِ IIDPS

Security System	Attack type / response time (second)				
	Identify User	Type-I	Type-II	Type-III	DDoS
OSSEC	× / -	✓ / 60	✓ / 60	× / -	× / -
AIDE	× / -	✓ / 60	✓ / 60	× / -	× / -
SAMHAIN	× / -	✓ / 60	✓ / 60	× / -	× / -
Symantec CSP	× / -	✓ / 2	✓ / 2	Δ / 3	Δ / 3.5
IIDPS	✓ / 0.45	✓ / 0.001	✓ / 0.001	✓ / 0.45	✓/0.45

جدول VII: مقایسه ی IIDPS با دیگر HIDSها

در این مقاله، یک مهاجم خودی می تواند با استفاده از ID و رمز عبور ورود به سیستم کاربر دیگری ، وارد سیستم شود و کاری مخرب انجام دهد. همان طور که پیش از این مطرح شد، حمله ی نوع I، حمله ای است که گروهی خاص در ان مانع از استفاده می شوند. حمله ی نوع II از rootkit ها استفاده می کند تا SC های حساس را ارائه نماید (()unlinkatو()kill) تا داده یا منبع حساس سیستم را تغییر دهد. حمله ی نوع III، GDB (یک ابزار اشکال زدایی برنامه) را به کار می برد تا روندهای در حال اجرا را رهگیری کند و یک حمله ی DDoS را در جهت تجاوز به دنیای خارجی سیستم راه اندازی نماید.

نتایج مقایسه ی این طرح ها در جدول II نشان داده می شود که "✓" در آن به معنی این است که سیستم دارای تابع تخصیص داده شده است و "x" اینکه سیستم این تابع را فراهم نمی کند را نشان می دهد و "Δ" نشان می دهد که سیستم تابع را دارد اما کاملا تابع IIDPS را محقق نمی کند. همه ی این سیستم ها به جز IIDPS، تابع شناساییِ کاربر محتمل را دارا نمی باشند. زمان های پاسخ گوییِ سیستم برای همه ی نوع های حمله نیز در جدول VII نشان داده می شود که در آن IIDPS بهتر از دیگر سیستم های آزمایش شده عمل می کند.

5. مباحثه

در این مقاله، یک IIDPS توسعه داده شده است تا حمله های خودی در سطح SC را با استفاده از داده کاوی و روش های کالبد سنجی شناسایی کند. نتایج تجربی نشان می دهد که IIDPS می تواند به طور موثر در برابر چندین حمله ی فوقالذکر مقاومت کند. خروجی، ویژگی های 16 را توسعه می دهد، این عمل تایید می کند که داده کاوی و روش های کالبد سنجیِ استفاده شده برای شناسایی تهاجم، مقاومت کارآمدی در برابر حمله ارائه می کند.

آزمایش دوم نشان می دهد که دقت میانگین شناسایی برابر با 94.29٪ است. دقت پشتیبانی کردن کاربر در جدول VI برابر با 89.97٪ است چون فایل های ثبت وقایع پشتیبانی SC های رایج تری در مقایسه با فایل های ثبت وقایع دیگر کاربران دارد. همچنین نشان داده شده است که IIDPS می تواند در زمانی که عادت کاربران ناگهانی تغییر می کند، به اشتباه شناسایی را انجام دهد. با این اوصاف، IIDPS در بیشتر موارد هنوز می تواند قانونی بودنِ کاربر وارد شده را تشخیص دهد.

در زمانی که کاربر یک دستور را وارد می کند، صدها یا هزاران SC تولید خواهد شد. تحلیل تعداد زیادی از SCها معمولا زمان زیادی می برد. همان طور که در جدول VII نشان داده شده، IIDPS 0.45 ثانیه برای شناسایی کاربر صرف می کند. اگرچه دیگر سیستم ها نسبت به IIDPS زمان بیشتری را برای تحلیل داده صرف می کنند، اما چگونگی کاوش SC ها در یک روش کارآمد باید مطرح شود. به کار بردن یک شبکه ی محاسباتی محلی می تواند سرعت پردازش سرور کاوش و سرور شناسایی را بالا ببرد. به طور کلی، ویژگی های کالبدسنجی کاربران که از

عملیات های بنیادی آنها بازیابی شده اند، در شناسایی رفتارهای مخرب کاربر مفید هستند و به ما می گوید که مهاجمین احتمالی چه کسانی هستند. این سیستم می تواند رفتارهای مخرب را نیز برای سیستم ها در جهت به کار بردن واسط GUI شناسایی کند. تعدادی دستور برنامه ی واسط شخص ثالث از جمله دستورات استفاده شده در پایگاه داده ی اوراکل، منطق شبکه ی اوراکل، MQ کره ی وب IBM، و برخی برنامه های توسعه داده شده بوسیله ی کاربر ایجاد شده است. نیاز داریم تا SC های تولید شده و الگوهای SC ایجاد شده بوسیله ی این دستورات را مطالعه کنیم، به طوریکه IIDPS می تواند رفتارهای مخرب مطرح شده بوسیله ی آنها را تشخیص داده و در ادامه سیستم محافظت شده را از مورد هجوم قرار گرفتن حفظ کند. علاوه بر این، کاویدن پروفایل های کاربری با استفاده از یک رویکرد خوشه ی ناظر می تواند عملکرد روند کاوش را نیز بهبود بخشد چون پردازش داده ی بزرگ در واقع یک چالش مهندسی است. علاوه بر این، برای شناسایی یک حمله و کاهش زمان پاسخ گویی متناظر، به یک ناظر بارکار خوشه، فیلتر سریعتر، الگوریتم شناسایی کارآمد و محیط مقاوم در برابر خرابیِ فراهم شده بوسیله ی یک شبکه ی محاسباتی نیازمندیم. علاوه بر این، یک تحلیل ریاضی روی رفتارهای IIDPS در استنتاج عملکرد رسمی و مدل های هزینه ی آن مفید است. کاربر می تواند بوسیله ی این موارد عملکرد و هزینه ی IIDPS را پیش از استفاده ی آن پیش بینی کند. مدل پیشنهاد شده در 36 می تواند در ادامه برای افزایش دقت شناسایی و بهبود نرخ قطعی استفاده شود.

علاوه بر این، ممکن است این سوال پرسیده شود که رکورد رفتار چگونه برای کاربری جدید ایجاد شده است و IIDPS چگونه یک پروفایل کاربری را به روز رسانی می کند. پاسخ سوال اول این است که وقفتی کاربر جدید k وجود داشت باشد، IIDPS فایل ثبت وقایع، فایل عادت و پروفایل کاربری k را در اولین ورود به سیستم این کاربر ایجاد می کند و در ادامه فرایند نشان داده شده در شکل 3 را دنبال می کند تا پروفایل کاربری k را تولید نماید. پاسخ سوال دوم این است که در هر مان که k به سیستم وارد می شود، علاوه بر SC kکه مستقیما ارسال می شود، IIDPS ترتیبات SC تولید شده بوسیله ی دستورات ارسالی را نیز شناسایی می کند. این ترتیبات SC در ادامه از طریق فراخوانی الگوریتم 2 برای شناسایی این مسئله که k مهاجم است یا نه، به کار گرفته می شود. وقتی k به

عنوان یک کاربر قانونی شناسایی می شود، IIDPS به جمع آوری SC های k ادامه می دهد و هنگامی که از سیستم خارج می شود، پروفایل این کاربر را براساس فرایند نشان داده شده در شکل 3 به روز رسانی می کند. وقتی پروفایل کاربری k به روز رسانی شود، می تواند برای شناسایی بعدی به گار گرفته شود.

6. نتیجه گیری

در این مقاله، رویکردی ارائه کردیم که روش های داده کاوی و کالبد سنجی را برای شناسایی الگوهای SC نشانگر برای یک کاربر به کار می برد.زمانی که الگوهای SC همیشگی در فایل ثبت وقایع کاربر ظاهر می شود، محاسبه می گردد، الگوهای SC که بیشترین استفاده را دارند، فیلتر می شوند و در ادامه یک پروفایل کاربر ایجاد می گردد. IIDPS با شناسایی الگوهای SC یک کاربر به عنوان عادات استفاده ی کامپیوترش از SC های ورودی کنونی، دربرابر حمله های احتمالی مقاومت می کند. نتایج تجربی نشان می دهد که دقت شناسایی میانگین در زمانی که آستانه ی نرخ تعیین 0.9 باشد، بیشتر از 94٪ است. این موضوع نشان می دهد که IIDPS می تواند به مدیر سیستم در مشخص کردن خودی یا مهاجم در یک محیط بسته کمک کند. مطالعه ی بعدی از طریق بهبود دادن عملکرد IIDPS و تحقیق در مورد دستورات برنامه ی واسط شخص ثالث انجام داده خواهد شد.

REFERENCES

[1] S. Gajek, A. Sadeghi, C. Stuble, and M. Winandy, "Compartmented security for browsers—Or how to thwart a phisher with trusted computing," in *Proc. IEEE Int. Conf. Avail., Rel. Security*, Vienna, Austria, Apr. 2007, pp. 120–127.

[2] C. Yue and H. Wang, "BogusBiter: A transparent protection against phishing attacks," *ACM Trans. Int. Technol.*, vol. 10, no. 2, pp. 1–31, May 2010.

[3] Q. Chen, S. Abdelwahed, and A. Erradi, "A model-based approach to self-protection in computing system," in *Proc. ACM Cloud Autonomic Comput. Conf.*, Miami, FL, USA, 2013, pp. 1–10.

[4] F. Y. Leu, M. C. Li, J. C. Lin, and C. T. Yang, "Detection workload in a dynamic grid-based intrusion detection environment," *J. Parallel Distrib. Comput.*, vol. 68, no. 4, pp. 427–442, Apr. 2008.

[5] H. Lu, B. Zhao, X. Wang, and J. Su, "DiffSig: Resource differentiation based malware behavioral concise signature generation," *Inf. Commun. Technol.*, vol. 7804, pp. 271–284, 2013.

[6] Z. Shan, X. Wang, T. Chiueh, and X. Meng, "Safe side effects commitment for OS-level virtualization," in *Proc. ACM Int. Conf. Autonomic Comput.*, Karlsruhe, Germany, 2011, pp. 111–120.

[7] M. K. Rogers and K. Seigfried, "The future of computer forensics: A needs analysis survey," *Comput. Security*, vol. 23, no. 1, pp.12–16, Feb. 2004.

[8] J. Choi, C. Choi, B. Ko, D. Choi, and P. Kim, "Detecting web based DDoS attack using MapReduce operations in cloud computing environment," *J. Internet Serv. Inf. Security*, vol. 3, no. 3/4, pp. 28–37, Nov. 2013.

[9] Q. Wang, L. Vu, K. Nahrstedt, and H. Khurana, "MIS: Malicious nodes identification scheme in network-coding-based peer-to-peer streaming," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, 2010, pp. 1–5.

[10] Z. A. Baig, "Pattern recognition for detecting distributed node exhaustion attacks in wireless sensor networks," *Comput. Commun.*, vol. 34, no. 3, pp. 468–484, Mar. 2011.

[11] H. S. Kang and S. R. Kim, "A new logging-based IP traceback approach using data mining techniques," *J. Internet Serv. Inf. Security*, vol. 3, no. 3/4, pp. 72–80, Nov. 2013.

[12] K. A. Garcia, R. Monroy, L. A. Trejo, and C. Mex-Perera, "Analyzing log files for postmortem intrusion detection," *IEEE Trans. Syst., Man, Cybern., Part C: Appl. Rev.*, vol. 42, no. 6, pp. 1690–1704, Nov. 2012.

[13] M. A. Qadeer, M. Zahid, A. Iqbal, and M. R. Siddiqui, "Network traffic analysis and intrusion detection using packet sniffer," in *Proc. Int. Conf. Commun. Softw. Netw.*, Singapore, 2010, pp. 313–317.

[14] S. O'Shaughnessy and G. Gray, "Development and evaluation of a data set generator tool for generating synthetic log files containing computer attack signatures," *Int. J. Ambient Comput. Intell.*, vol. 3, no. 2, pp. 64–76, Apr. 2011.

[15] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Appl. Soft Comput.*, vol. 10, no. 1, pp. 1–35, Jan. 2010.

[16] F. Y. Leu, K. W. Hu, and F. C. Jiang "Intrusion detection and identification system using data mining and forensic techniques," *Adv. Inf. Comput. Security*, vol. 4752, pp. 137–152, 2007.

[17] Z. B. Hu, J. Su, and V. P. Shirochin "An intelligent lightweight intrusion detection system with forensics technique," in *Proc. IEEE Workshop Intell. Data Acquisition Adv. Comput. Syst.: Technol. Appl.*, Dortmund, Germany, 2007, pp. 647–651.

[18] J. T. Giffin, S. Jha, and B. P. Miller, "Automated discovery of mimicry attacks," *Recent Adv. Intrusion Detection*, vol. 4219, pp. 41–60, Sep. 2006.

[19] U. Fiore, F. Palmieri, A. Castiglione, and A. D. Santis, "Network anomaly detection with the restricted Boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, Dec. 2013.

[20] M. A. Faisal, Z. Aung, J. R. Williams, and A. Sanchez, "Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study," *IEEE Syst. J.*, vol. 9, no. 1, pp. 1–14, Jan. 2014.

[21] R. J. Roger and M. W. Geatz, *Data Mining: A Tutorial-Based Primer*. Reading, MA, USA: Addison-Wesley, 2002.
[22] S. J. Shyu and C. Y. Tsai, "Finding the longest common subsequence for multiple biological sequences by ant colony optimization," *Comput. & Oper. Res.*, vol. 36, no. 1, pp. 73–91, Jan. 2009.
[23] D. Zhu and J. Xiao, "R-tfidf, a Variety of tf-idf Term Weighting Strategy in Document Categorization," in *Proc. Int. Conf. Semantics, Knowledge Grids*, Beijing, China, Oct. 2011, pp. 83–90.
[24] S. E. Robertson, S. Walker, M. M. Beaulieu, M. Gatford, and A. Payne, "Okapi at TREC-4," in *Proc. 4th text Retrieval Conf.*, 1996, pp. 73–96.
[25] S. Yu, K. Sood, and Y. Xiang, "An effective and feasible traceback scheme in mobile internet environment," *IEEE Commun. Lett.*, vol. 18, no. 11, pp. 1911–1914, Nov. 2014.
[26] B. Sayed, I. Traore, I. Woungang, and M. S. Obaidat, "Biometric authentication using mouse gesture dynamics," *IEEE Syst. J.*, vol. 7, no. 2, pp. 262–274, Jun. 2013.
[27] S. C. Arseni, E. C. Popovici, L. A. Stancu, O. G. Guta, and S. V. Halunga, "Securing an alerting subsystem for a keystroke-based user identification system," in *Proc. Int. Conf. Commun.*, Bucharest, Romania, 2014, pp. 1–4.
[28] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proc. AFIPS Spring Joint Comput. Conf.*, New Brunswick, NJ, USA, 1967, pp. 1–4.
[29] M. Minsky, "Form and content in computer science," *J. ACM*, vol. 17, no. 2, pp. 197–215, Apr. 1970.
[30] OSSEC. [Online]. Available: http://www.ossec.net/
[31] AIDE. [Online]. Available: http://aide.sourceforge.net/
[32] SAMHAIN. [Online]. Available: http://www.la-samhna.de/samhain/
[33] Symantec CSP. [Online]. Available: http://www.symantec.com/critical-system-protection
[34] Symantec CSP, "Executing operating system commands from PL/SQL," Oracle, Redwood City, CA, USA, White Paper, Jul. 2008.
[35] J. Böhm-Mäder, "The WebSphere MQ for UNIX administration tool," Free Softw. Found., Boston, MA, USA, May 2013.
[36] P. Angin and B. Bhargava, "An agent-based optimization framework for mobile-cloud computing," *J. Wireless Mobile Netw., Ubiquitous Comput., Dependable Appl.*, vol. 4, no. 2, pp. 1–17, Jun. 2013.