سیستم تشخیص نفوذ مبتنی بر پروسس فازی از طریق تکنیک داده کاوی

محمد اکبرپور سکه
دانشگاه آزاد اسلامی واحد شیروان
moh_ak_s@yahoo.com

محسن کاهانی
دانشگاه فردوسی مشهد
kahani@ferdowsi.um.ac.ir

رضا منصفی
دانشگاه فردوسی مشهد
rmonsefi@ferdowsi.um.ac.ir

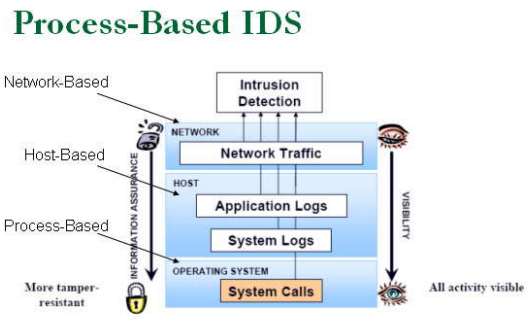
چکیده - سیستم تشخیص نفوذ[1] یکی از مهمترین مباحث امنیت شبکه[2] است که وجود آن در کنار سیستم جلوگیری از نفوذ[3] باعث بالا رفتن درجه امنیت سیستم خواهد شد. دراین مقاله برای تشخیص نفوذ از ردگیری دنباله فراخوانی‌های سیستمی استفاده شده است و هر پروسس[4] به صورت یک جعبه سیاه در نظر گرفته می‌شود و فقط فراخوانی های سیستمی آن مورد بررسی قرار می‌گیرد. از این طریق می توان رفتارهای نرمال و غیر نرمال کاربران را برروی پروسس‌های در حال اجرای سیستم‌عامل شناسایی کرد. برای تجزیه و تحلیل فراخوانی‌های سیستمی از روش داده‌کاوی فازی[5] استفاده شده است. در این طرح, هدف کاهش اندازه پایگاه‌داده و زمان مورد نیاز برای تشخیص نفوذ و نرخ هشدارهای نادرست می‌باشد.

کلید واژه: تشخیص نفوذ مبتنی بر پروسس، تشخیص نفوذ رفتارهای غیرمتعارف[6]، داده‌کاوی، دنباله فراخوانی سیستمی[7] ، قواعد اشتراکی[8]، سرریز بافر[9]، تشخیص نفوذ فازی[10]

[1] Intrusion Detection System
[2] Network Security
[3] Intrusion Prevention System
[4] Process
[5] Fuzzy Datamining
[6] Anomaly-based IDS
[7] System call sequences
[8] Association rule
[9] Buffer overflow
[10] Fuzzy IDS

۱- مقدمه و پیشینه تحقیق

اولین ایده سیستم‌های تشخیص نفوذ مبتنی بر پروسس[11] در سال ۱۹۹۶ توسط پروفسور فارست[12] [Forrest96] ارائه شد (انواع مختلف سیستمهای تشخیص نفوذ در شکل ۱ قابل مشاهده است). پس از آن در سال ۱۹۹۸ وی کار خود را کمی تعمیم داد و datasetهای نرمال و غیر نرمال را بر روی سیستم عامل لینوکس برای چندین پروسس ایجاد نمود. از جمله محققینی که در این زمینه مقالاتی ارائه داده اند می‌توان به آقای Wagner در سال [Wagner02] و Martin و همکارانش از دانشگاه تگزاس در سال ۲۰۰۵ [Cmartin05] و A.Hafmeyr در سال ۲۰۰۷ [Hafmeyr07] اشاره نمود.



شکل ۱ -مقایسه سطوح مختلف تشخیص نفوذ

در یک سیستم کامپیوتری اگر نفوذگر بتواند از همه تجهیزات امنیتی و دیواره آتش عبور کرده و به سیستم وارد شود و در حال اختلال به سیستم باشد, سیستم دیگری به نام سیستم تشخیص نفوذ رفتار کاربران را مورد بررسی قرار داده و نفوذ احتمالی را شناسایی می‌نماید. یکی از انواع سیستم‌های تشخیص نفوذ سیستم‌های تشخیص نفوذ مبتنی بر پروسس می‌باشد که جستجوی خود را بر روی پروسس‌ها در لایه هسته سیستم‌عامل انجام می‌دهد.از آنجایی که با اجرای هر پروسس تعداد زیادی فراخوانی‌سیستمی اجرا می‌شود, می‌توان برای هر پروسس این فراخوانی‌های‌سیستمی را جمع‌آوری کرده و با اجرای مکرر پروسس در حالت‌های امن مختلف یک بانک اطلاعاتی نرمال ایجاد نمود. به طور مشابه جمع‌آوری فراخوانی‌های‌سیستمی پروسس مخرب نیز امکان پذیر می‌باشد. (به عنوان مثال از طریق strace در لینوکس)

۲- ساختار سیستم‌های تشخیص نفوذ

انواع مختلفی از سیستم‌های تشخیص نفوذ وجود دارد که می‌تواند در لایه‌های مختلف شبکه و سیستم‌عامل و هسته سیستم‌عامل نفوذ را تشخیص دهد. از جمله این سیستم‌ها می‌توان به سیستم‌های تشخیص نفوذ مبتنی بر شبکه, مبتنی بر میزبان, مبتنی بر پروسس اشاره نمود. دسته‌بندی دیگری نیز برای این سیستم‌ها وجود درد که بر اساس نوع برخورد آنها با رفتارهای نرمال و غیر نرمال می‌باشد مانند سیستم‌های تشخیص نفوذ غیر متعارف و سیستم‌های تشخیص نفوذ سوء استفاده. در این طرح از نوع تشخیص نفوذ مبتنی بر پروسس و مبتنی بر رفتار های غیر متعارف استفاده شده است. که در این دیدگاه همه رفتارهای نرمال پروفایل می‌شوند و هر رفتار جدید

[11] Process-based Intrusion Detection System
[12] Forrest

که با رفتارهای نرمال منطبق نباشد به عنوان نفوذ تشخیص داده می‌شود. می‌توان گفت برای کشف نفوذ در سطح پروسس و هسته سیستم عامل باید مراحل زیر دنبال شوند:

جمع آوری فراخوانی‌های‌سیستمی اجرا شده که توسط یک پروسس در حالت‌های مختف امن اجرا می‌شوند و همچنین پروفایل کردن رفتار کاربران. (در این مرحله اندازه بانک اطلاعاتی ایجاد شده بسیار بزرگ و جستجو در آن بسیار زمان گیر می‌باشد.)

جمع‌آوری فراخوانی‌های سیستمی اجرا شده پروسس در حالت‌های مختلف نفوذ

تجزیه و تحلیل فراخوانی‌های‌سیستمی نرمال و تبدیل آنها به تعدادی قاعده

مقایسه بانک اطلاعاتی فراخوانی‌های پروسس جدید و در حال اجرا با فراخوانی‌های نرمال

در صورت وجود اختلاف در ترتیب فراخوانی‌های‌سیستمی پروسس جدید با فراخوانی‌های‌سیستمی نرمال و مورد انتظار, نفوذ تشخیص داده خواهد شد.

۳- طرح پیشنهادی

برای کشف رفتارهای نرمال یک پروسس باید از فراخوانی‌های سیستمی آن پروسس استفاده نمود. در این حالت هر پروسس به صورت یک "جعبه سیاه" در نظر گرفته می‌شود و لازم نیست اطلاعاتی از دستورات داخل پروسس وجود داشته باشد. فقط کافی است دنباله فراخوانی‌های سیستمی پروسس را هنگام اجرا ذخیره کرده و با استفاده از آن‌ها رفتارهای نرمال و غیرنرمال پروسس را شناسایی و در نهایت نفوذ احتمالی را کشف نمود.

درطرح پیشنهادی برای تجزیه و تحلیل بانک اطلاعاتی نرمال و کشف نفوذ از تکنیک داده‌کاوی فازی استفاده شده است با این روش کلیه فراخوانی های سیستمی نرمال به تعدادی قاعده تبدیل خواهد شد که با این روش بانک اطلاعاتی نرمال با اندازه بسیار بزرگ به یک فایل کوچک با تعداد کمی قاعده تبدیل خواهد شد.(شکل ۲ و شکل ۳ به طور کامل گویای طرح پیشنهادی می باشند.)

۳-۱) محیط کار و Dataset استفاده شده

برای ثبت رفتارهای نرمال یک کاربر می‌توان از ردگیری برنامه‌هایی که کاربر اجرا می‌کند استفاده نمود. برای ردگیری پروسس‌های در حال اجرا می‌توان آن پروسس‌ها را در شرایط مختلف چندین بار اجرا کرده و فراخوانی‌های سیستمی اجرا شده توسط آنها را درون فایلی ذخیره نمود. برای انجام این کار می‌توان از دستور strace در حساب ریشه[۱۳] در لینوکس استفاده نمود.

datasetهای متعددی توسط دانشگاه‌های New mexico وMIT روی سیستم عامل لینوکس و سان ایجاد شده است که رفتارهای نرمال پروسس‌های مختلف و چندین نفوذ بر روی این پروسس‌ها را برای آزمایشات خود ایجاد کرده‌اند. با استفاده

[13] *Root*

از datasetهای موجود در سایت UNM می‌توان الگوی خاصی برای رفتارهای نرمال تهیه کرده و قواعدی از آن استخراج نمود، تا درآینده بتوان با استفاده از این قواعد نرمال هر رفتار مشکوک و در نتیجه هر نوع نفوذ را شناسایی نمود. چون معمولاً نفوذها رفتار نرمال نداشته و در قواعد استخراج شده نرمال پروسس مربوطه صدق نمی‌کنند. دانشگاه UNM در دو مرحله این datasetها را ایجاد کرده است. در مرحله اول رفتار پروسس‌ها را در شرایط کاملاً امن و نرمال ردگیری نموده و یک بانک اطلاعات بزرگ از فراخوانی‌های سیستمی ایجاد کرده است. در مرحله دوم رفتار پروسس‌ها را هنگام نفوذهای شناخته شده‌ای مانند – buffer overflow - lprcp sunsendmailcp ردگیری کرده و بانک اطلاعاتی دیگری برای ثبت این فراخوانی‌های غیر نرمال فراهم نموده است. در این ردگیری‌ها پارامترهای ورودی و خروجی فراخوانی‌های سیستمی نادیده گرفته شده است. [Forrest96].

همچنین به جز datasetهای دانشگاه UNM فایل‌هایی نیز برای ردگیری فراخوانی‌های سیستمی چند پروسس مربوط به سیستم عامل لینوکس فدورا در سیستم شخصی ایجاد گردیده، که هنگام تست سیستم به کار برده شده اند.

برای پیاده سازی و اجرای برنامه های و داده های آزمایشی و همچنین محاسبات انجام شده از یک سیستم با مشخصات زیر استفاده‌شده است: پردازنده پنتیوم IV با ۵۱۲ مگابایت حافظه، سیستم‌عامل لینوکس فدورا و زبان برنامه نویسی C++.

همچنین datasetهای دریافت شده از سایت UNM به عنوان مبنای کار قرارگرفته که پروسس‌های استفاده شده در این مقاله عبارتند از Stide, Named, Xlock, Ps, Login.

۳-۲) پیاده سازی

همان‌طور که در [شکل ۲] مشاهده می شود مراحل ذیل برای تشخیص نفوذ باید به ترتیب اجرا شوند:

در مرحله "normal trace" کلیه فراخوانی‌های نرمال پروسس‌ها جمع‌آوری می‌شود، که در این‌جا همان datasetهای نرمال UNM می‌باشند پس فقط آنها را تهیه کرده و به عنوان ورودی به سیستم می‌دهیم.

در مرحله "Data mining" باید دنباله فراخوانی‌های سیستمی فوق با استفاده از تکنیک قواعد اشتراکی (که از تکنیک‌های داده‌کاوی می‌باشد) به یک سری قواعد تبدیل شود.

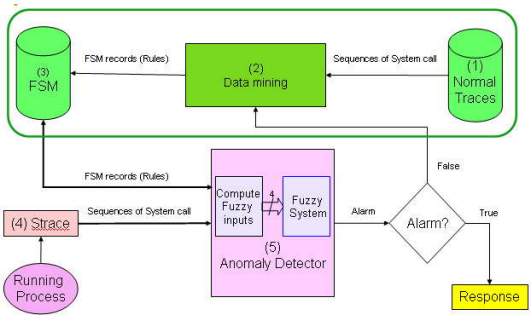
درمرحله"FSM" باید کلیه قواعدنرمال استخراج شده با یک قالب مشخصی‌درماشین حالات‌متناهی[14] ذخیره شود تا بتوان در فازهای بعدی با سرعت دستیابی بالا از آنها استفاده نمود.

کلیه مراحل فوق به صورت برون خطی[15] انجام می‌شود و بسیار وقت گیر می‌باشد. در صورتی که مراحل بعدی که مربوط به تشخیص رفتارهای غیر نرمال پروسس در حال اجرا هستند به صورت بلادرنگ انجام می‌شوند.

[14] Finite State Machine
[15] Off-line

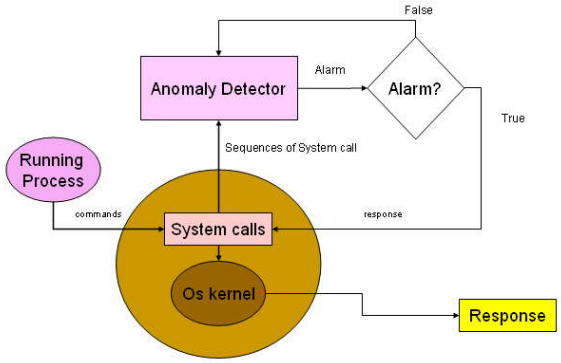
در مرحله "Strace" کلیه فراخوانی‌های سیستمی پروسس در حال اجرا، توسط هسته سیستم عامل استخراج شده و به عنوان ورودی به مرحله بعدی داده می‌شود.

در مرحله "Anomaly Detector" بخشی با نام سیستم فازی وجود دارد که می‌تواند یک مقدار فازی را به عنوان خروجی تولید کند تا به صورت یک هشدار به لایه بالاتر ارسال گردد. در این مرحله ابتدا باید چندین پارامتر ورودی فازی از روی اطلاعات دریافت شده از بخش Strace و مقایسه آن با FSM محاسبه شوند و سپس سیستم فازی از روی این پارامترهای ورودی یک پارامتر خروجی به نام Decision نتیجه می‌دهد که با استفاده از آن می‌توان یک هشدار امنیتی به مدیر شبکه ارسال کرد.



شکل ۲ – جزئیات و مراحل طرح پیشنهادی

نمایی دیگر از شکل فوق در [شکل ۳] مشاهده می‌شود که ارتباط بین بخش‌های مختلف سیستم پیشنهادی را با هسته[16] نشان می‌دهد.



شکل ۳- ارتباط مراحل مختلف سیستم پیشنهادی و هسته

۳-۳) داده کاوی[17]

برای استخراج قواعد نهفته موجود در فایل نرمال و تجزیه و تحلیل آن، از روش قواعد اشتراکی مربوط به تکنیک داده‌کاوی استفاده شده است. به این صورت که برای هر itemset یک درجه پشتیبانی[18] و یک درجه اطمینان[19] محاسبه می‌شود. و به این طریق قواعد اشتراکی به صورت مثال زیر ایجاد شده و در نهایت می‌توان آنها را در یک ماشین حالات‌متناهی قرار داد.

فرمول محاسبه درجه پشتیبانی و درجه اطمینان هر قاعده به صورت زیر می باشد: (با فرض وجود قاعده $x \rightarrow y$)

$$s = \sup port(x \cup y)$$

$$confidence = c = \frac{\sup port(x \cup y)}{\sup port(x)}$$

[16] *OS Kernel*

[17] *Datamining*
[18] *support*
[19] *confidence*

مثالی ساده از قواعد اشتراکی به صورت زیر می‌باشد:

open → read , c, s : 5 →3 , c, s
open → getrlimit , c, s : 5 →76, c, s
mmap → close ,c, s : 90 →6, c, s

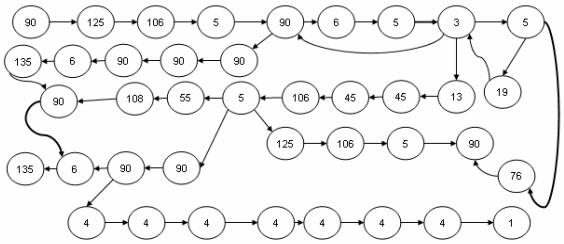
در این مثال بعد از open همیشه read یـا getrlimit فراخـوانی می‌شود، یعنی اگر بعد از open یک فراخوانی دیگـر مـثلاً write اجرا گردد غیر نرمال شـناخته خواهـد شـد. بخـشی از دنبالـه فراخوانی‌هـای سیـستمی پروسـس sendmail را در [شـکل ۴] آورده شده است.

```
read() write() close() munmap() sigprocmask() wait4()
sigprocmask() sigaction() alarm() time() stat() read()
alarm() sigprocmask() setreuid() fstat() getpid()
time() write() time() getpid() sigaction() socketcall()
sigaction() close() flock() getpid() lseek() read()
kill() lseek() flock() sigaction() alarm() time()
stat() write() open() fstat() mmap() read() open()
fstat() mmap() read() close() munmap() brk() fcntl()
setregid() open() fcntl() chroot() chdir() setreuid()
lstat() lstat() lstat() lstat() open() fcntl() fstat()
lseek() getdents() fcntl() fstat() lseek() getdents()
close() write() time() open() fstat() mmap() read()
close() munmap() brk() fcntl() setregid() open() fcntl()
chroot() chdir() setreuid() lstat() lstat() lstat()
lstat() open() fcntl() brk() fstat() lseek() getdents()
lseek() getdents() time() stat() write() time() open()
getpid() sigaction() socketcall() sigaction() umask()
sigaction() alarm() time() stat() read() alarm()
getrlimit() pipe() fork() fcntl() fstat() mmap() lseek()
close() brk() time() getpid() sigaction() socketcall()
sigaction() chdir() sigaction() sigaction() write()
munmap() munmap() munmap() exit()
```

شکل4 - نمونه‌ای از دنباله فراخوانی سیستمی یک پروسس

حال با فرمول‌های confidence, support روش قواعد اشتراکی داده‌کاوی می‌توان قواعد نرمال را استخراج نمـود. بـرای کـار بـا فراخـوانی‌هـای سیـستمی بـه هـر کـدام از آنهـا یـک شناسـه اختصاص می‌دهیم و از این پس با استفاده از این شناسـه‌هـا بـا آنها کار می‌کنیم.

بعد از استخراج قواعد باید آنها به همراه درجه اطمینان در یـک FSM ذخیـره شـوند. مثـال سـاده‌ای از دنبالـه فراخـوانی‌هـای سیستمی که باید در FSM ذخیـره شـوند در [شـکل ۵] آورده شده است.



شکل ۵- ترتیب اجرای فراخوانی‌ها در یک پروسس(FSM)

پس از چندین بار اجرای برنامه مشاهده شد که یـک سـری از دنباله های منحصر به فـرد از فراخـوانی هـای سیـستمی دائـم تکرار می شوند، در این قسمت هدف یـافتن ایـن دنبالـه هـای منحصر به فرد است.

رفتارهای مجاز سیـستم را مـی‌تـوان بـه دو دسـته قـانونی[20] و نرمال تقسیم کرد. تفاوتی بین رفتارهای قانونی و نرمـال وجـود دارد. به این صورت که ممکن اسـت رفتـاری در فایـل نرمـال نباشد ولی غیر متعارف نیز نیست. این حالـت مـواقعی رخ مـی دهد یک خطای شناخته شده‌ای از پروسس رخ دهد و بخـشی از کد مربوط به این خطا اجرا گردد که قبلا هنگام ایجاد رفتـار نرمال چنین حالتی در نظر گرفته نشده است. به این رفتار یک رفتار قانونی گویند که غیر نرمال تشخیص داده مـی‌شـود، و در نتیجه باعث یک هشدار اشتباه[21] خواهد شد.

نکته ۱: نمی‌توان گفت که همیشه نفوذ دارای یک رفتار غیـر نرمال است و یا این‌که همیشه رفتارهای نرمال نفوذ نیستند.

نکته۲: باید روش محاسبه درجه اطمینان هر کـدام از قواعـد مـستقل از طـول مجموعـه باشـد، چـون در غیـراین صـورت

[20] Legal
[21] False alarm

نفوذگر برای تغییر این درجه عمل خود، از ترتیب فراخوانی‌های بسیار طولانی و نرمال استفاده می‌نماید.

۳-۴) سیستم فازی

در این طرح چهار پارامتر ورودی فازی استفاده شده است که هر یک از آنها به نوعی انعطاف سیستم را بیشتر می‌کند و همچنین می‌توانند تعداد تشخیص‌های نادرست را کاهش دهند

این چهار پارامتر فازی عبارتند از:

تعداد عدم تطابق[۲۲]

میانگین درجه اطمینان قواعد[۲۳]

درجه خطر برای قواعد مشکوک[۲۴]

تعداد هر کدام از قواعد[۲۵] در پروسس

که همه آنها می‌توانند مقادیر فازی Small, Medium, Large داشته باشند. در [شکل ۶] جزئیات این پارامترها مشخص شده است و مقادیر a, b, c در نمودارهای مربوط به مجموعه‌های فازی این پارامترها در بخش بعدی (توابع عضویت مجموعه‌های فازی) آمده است. این چهار پارامتر بر اساس دنباله فراخوانی‌های پروسس در حال اجرا و قواعد استخراج شده ماشین حالات متناهی (FSM) محاسبه شده و به سیستم تشخیص نفوذ فازی وارد می‌شوند. سیستم فازی نیز پس از انجام محاسبات فازی خروجی y را به عنوان هشدار ایجاد می‌کند که این خروجی نیز می‌تواند مقادیرفازی نرمال[۲۶]، مشکوک[۲۷] و نفوذ[۲۸] داشته باشد.

سیستم فازی طرح پیشنهادی دارای مشخصات زیر است [شکل ۶ و ۷]:

تعداد مجموعه‌های فازی: پنج مجموعه (چهار مجموعه برای متغییرهای ورودی و یکی برای متغیر خروجی).

فازی‌ساز[۲۹]: منفرد.

غیرفازی‌ساز[۳۰]: میانگین مراکز.

موتور استنتاج فازی[۳۱]: ضرب.

پایگاه قواعد فازی[۳۲]: جدول جستجو.

با توجه با نوع سیستم فازی فوق می‌توان خروجی y را به صورت زیر محاسبه نمود:

فرمول محاسبه در موتور استنتاج ضرب:

$$\mu_{B'}(y) = Max_{l=1}^{M}\left[\sup(\mu_{A'}(x).\prod_{i=1}^{n}\mu_{A_i^l}(x_i)\mu_{B^l}(y))\right]$$

$$f(x) = y^* = \frac{\sum_{l=1}^{M}\bar{y}^l(\prod_{i=1}^{n}\mu_{A_i^l}(x_i))}{\sum_{l=1}^{M}(\prod_{i=1}^{n}\mu_{A_i^l}(x_i))}$$

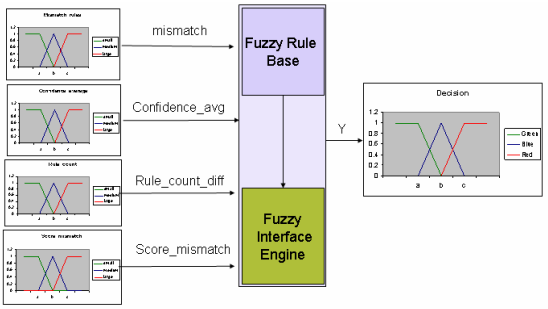
Fuzzy set	a	b	c
Mismatch	0	1	2
Confidence	50	80	90

[22] Mismatch
[23] Confidence_average
[24] Score_mismatch
[25] Rule_count

[26] Green(Normal)
[27] Blue(Dubious)
[28] Red(Danger)
[29] Fuzzifier
[30] Defuzzifier
[31] Fuzzy Inference Engine
[32] Fuzzy Rulebase

Rule_count	200	700	1300
Score_mismatch	50	55	60
Decision	0.2	0.6	0.8

شکل ۶ - مقادیر a, b, c برای توابع عضویت فازی



شکل ۷- مشخصات سیستم فازی

برای تشخیص نفوذ ابتدا باید پارامترهای فازی فوق محاسبه شده و به سیستم فازی داده شوند. سیستم فازی نیز بر اساس ساختاری که دارد قواعد مشکوک را شناسایی کرده و در نهایت یک خروجی به عنوان نتیجه کار به مدیر سیستم ارسال می‌کند. به طور مثال یکی از رفتارهای مشکوک اجرای فراخوانی close بعد از فراخوانی lseek می‌باشد. و یا همیشه بعد از open یک mmap اجرا میگردد تا یک نگاشت حافظه به اشاره گر آن فایل صورت گیرد. به طور مثال:

fp= open("/usr/password","r")
address=
mmap(0,len,prot_read,map_private,fp,offset)

یعنی اگر بعد از open فراخوانی دیگری اجرا گردد غیر نرمال خواهد بود.

۴- نتایج برنامه و ارزیابی

با چندین بار اجرای برنامه‌های فوق برای پروسس‌های مختلف Stide, [Z], Xlock, Ps, Login نتایج کلی زیر حاصل شد [شکل ۸]. همان‌طور که دیده می‌شود از تعداد ۱۰۵ مورد نفوذ موجود در پروسس stide تعداد ۶۱ مورد توسط سیستم، Red و تعداد ۴۳ مورد، Blue و یک مورد، Green گزارش شده است. این یک مورد Green یک تشخیص نادرست می‌باشد، چون باید سیستم آن‌را Blue یا Red معرفی می‌کرد. از مزایای روش مبتنی بر پروسس و طرح پیشنهادی، می‌توان به موارد زیر اشاره کرد:

توانایی تشخیص نفوذ بلادرنگ[۳۳] وجود دارد (به دلیل اندازه کم رفتارهای نرمال)

همه فعالیت‌ها قابل مشاهده هستند و فعالیت پنهانی وجود ندارد.

اطمینان از صحت اطلاعات[۳۴] استخراج شده در این روش بیشتر از روش‌های دیگر است.

الگوریتم‌های این روش به سیستم‌عامل خاصی بستگی نداشته و در تمام سیستم‌های عامل می‌توان از آنها استفاده نمود.

با این روش حملاتی را که دیگر سیستم‌های تشخیص نفوذ نمی‌توانند شناسایی کنند تشخیص داده می‌شود (مانند Buffer overflow، decode، sunsendmailcp و lprcp و ...).

[33] Real time IDS
[34] Information Assurance

ایجاد الگو و قواعد نرمال در این روش ساده‌تر از دیگر روش‌هاست (چون فراخوانی‌های سیستمی محدود هستند).

با توجه به خروجی های شکل ۸ مشاهده می‌شود که برای بخش داده‌کاوی که به صورت برون خطی اجرا می‌شود، زمان زیادی صرف خواهد شد (به عنوان مثال برای کشف رفتارهای نرمال Stide به اندازه ۱۱۱۶۰ ثانیه زمان نیاز است).

پس از چندین بار اجرای برنامه کشف نفوذ برای هر پروسس نفوذدار حداقل یک مورد رفتار غیر نرمال شناسایی شد، که دنباله های زیر از جمله این رفتارهای غیر نرمال می باشند:

process=3 read , brk , lseek , close , munmap , fstat , mmap , read , write
process=44 brk ,read ,lseek ,close ,munmap ,fstat ,mmap ,read ,write
process=79 read ,brk ,read ,brk ,brk ,brk ,brk ,brk ,brk
process=79 brk ,read ,lseek ,close ,munmap ,fstat ,mmap ,read ,write
process=85 brk ,brk ,exit ,mprotect ,mprotect ,stat ,open ,mmap ,close
process=102 mmap ,mmap ,sigreturn
process=102 mmap ,sigreturn ,mprotect
process=102 sigreturn ,mprotect ,open ,read ,mmap ,mmap ,mmap ,mmap ,close
process=103 setup ,setup ,setup

با توجه به جدول زیر (شکل ۸) مشاهده می‌شود که تعداد تشخیص‌های نادرست در پروسس"Login" رضایت بخش نمی‌باشد، و بعد از چندین بار اجرای برنامه برای این پروسس رفتار نرمال آن استخراج نشده و مشخص است که با داده‌های نرمال موجود نمی‌توان رفتار نرمال این پروسس را کشف نمود، چون در این پروسس بیشتر از رفتار نرمال خود پروسس، رفتار کاربر نیز دخالت دارد. و نمی‌توان تنها با پروفایل مربوط به رفتار پروسس رفتار نرمال مناسبی برای کل پروسس بدست آورد. پس یکی از مشکلات سیستم پیشنهادی مناسب نبودن آن برای پروسس Login (و پروسس‌های که رفتار آنها به نحوی به رفتار کاربر نیز بستگی داشته باشند) می‌باشد. اگر بخواهیم سیستم برای چنین پروسس‌هایی نیز بهتر کار کند باید پروفایل پروسس, همراه با پروفایل کاربر ایجاد شود که پیچیدگی کار را افزایش خواهد داد.

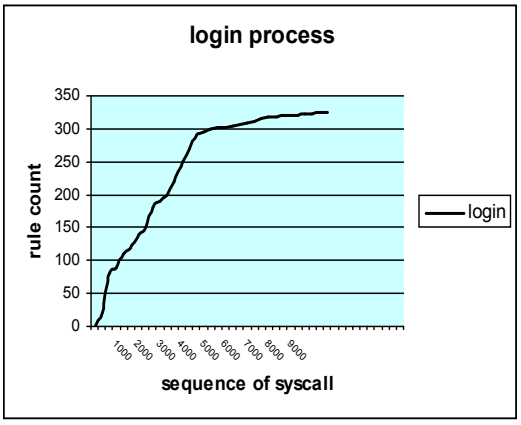
Data mining - (off-line)				Datasets information						Linux process
				intrusive		normal				
mining time (Second)	end time	start time	No. association rule	No. sys call (for test)	No. Trace (for test)	No. sys call		No. Trace		
						for learning	all	for learning	all	
11160	09:18:31.26	06:12:15.23	191	99514	105	5230018	19524102	5000	13726	stide
2276	19:38:01.44	19:00:25.32	446	1800	6	545589	9230572	13	27	named
231	20:04:40.13	20:00:41.83	188	3231	2	100000	16598639		72	Xlock
10	14:42:12.36	14:42:02.15	142	612	31	6144	6144	24	24	ps
39	15:34:04.17	15:33:25.12	312	4857	15	8906	8906	33	33	login

Detect - (real-time)				Linux process
False alarm	Detect			
	Green	Blue	Red	
1	1	43	61	stide
0	0	2	4	named
0	0	0	2	Xlock
0	0	0	31	ps
4	4	1	10	login

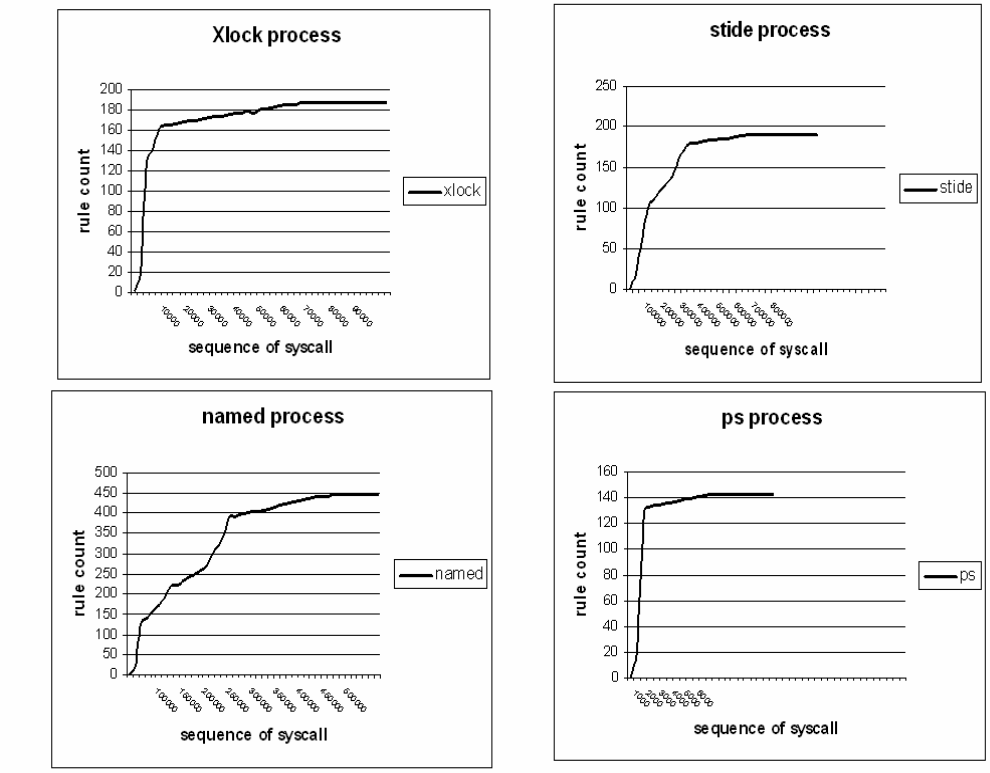
شکل ۸ - نتایج نهایی برنامه بعد از بخش برون خطی و بخش بلادرنگ

مطابق نتایج زیر (شکل ۹ و شکل ۱۰) می‌توان گفت که با افزایش تعداد بانک اطلاعاتی نرمال تعداد قواعد پروسس‌های Stide, Xlock از ۲۰۰ بیشتر نمی‌شود، ولی در پروسس login با افزایش تعداد داده‌های آزمایشی تعداد قواعد نیز افزایش می‌یابد و هیچ گاه تعداد قواعد به یک حد آستانه نرسیده است (Sliding window=5).

با توجه به نمودار شماره ۹ مشخص است که رفتار پروسس Login به راحتی قابل تبدیل شدن به تعدادی قاعده منحصر به فرد نیست.



شکل ۹ – نمودارهای تعداد دنباله های منحصر به فرد در پروسسlogin



شکل ۱۰ – نمودارهای تعداد دنباله های منحصر به فرد در پروسسهای stide و xlock و ps و named

۵- توصیه‌های آتی

روش پیشنهاد شده در این تحقیق برای تشخیص تعدادی از نفوذهای مورد بررسی در سایت UNM مناسب است، ولی بعضی از حمله‌های جدید ممکن است رفتارهای نرمال داشته باشند در صورتی‌که پارامترهای فراخوانی‌های آنها غیر نرمال باشد، در نتیجه با این روش نمی‌توان چنین حمله‌هایی را شناسایی نمود. پس بهتر است در کنار استفاده از روش‌های ذکر شده از پارامترهای ورودی و خروجی فراخوانی‌های سیستمی نیز جهت تشخیص نفوذ استفاده نمود تا بتوان حمله‌های پیشرفته‌تر امروزی را نیز تشخیص داد. به عنوان مثال دنباله فراخوانی‌های سیستمی زیر از دید این طرح برای اکثر پروسس‌ها نرمال می‌باشد، در صورتی که نفوذگر در حال عوض کردن رمز کاربر ریشه و یا ایجاد یک کاربر جدید می باشد [شکل ۱۱]:

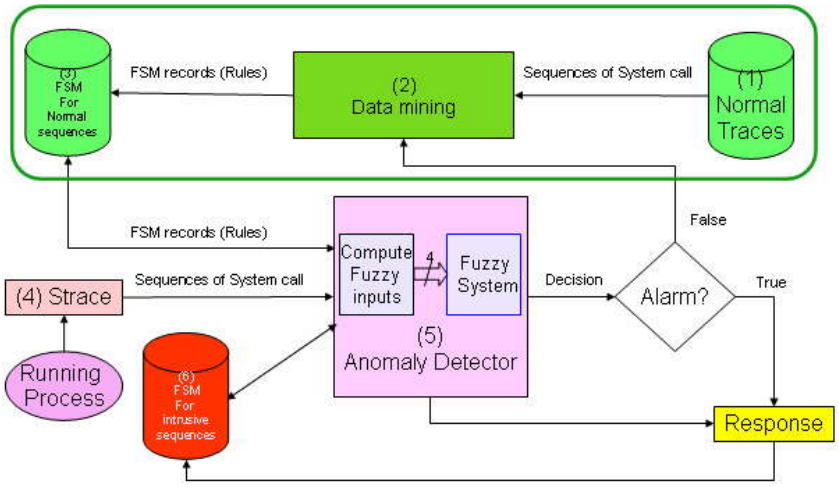
```
Normal trace: (open- mmap- read- lseek- write- close- exit)
Dubious trace:
                open("../../etc/shadow","RDWR")
                mmap(0,…fp,offset)
                read(fp,user_inf)
                lseek(fp,root_record)
                write(fp,new_pass)
                close(fp)
                exit(0)
```

شکل ۱۱- مشکوک بودن یک دنباله فراخوانی نرمال با استفاده از تجزیه و تحلیل پارامترها

دلیل اینکه در این طرح از پارامترهـای ورودی و خروجـی فراخوانی‌های سیستمی استفاده نشده , فراهم نشدن محـیط مطالعاتی و datasetهای مناسب می‌باشد.

بنابراین درکارهایی که درآینده در این زمینه کاری انجام می‌شود توجه به نکات زیر ضروری خواهد بود:

- کنترل ترتیب اجرای فراخوانی‌های سیستمی
- نظارت بر پارامترهای ورودی خروجی کلیه فراخوانی‌هـای اجرا شده
- ایجاد پروفایل‌هایی از رفتار کاربران برای تـشخیص نفوذ پروسس‌هایی ماننـد"login" کـه نمـی‌تـوان بـه راحتـی بـرای آن بـا روش پیـشنهادی قواعـد نرمـال استخراج نمود.
- امکان استفاده هم‌زمان از تشخیص نفوذ anomaly و misuse
- امکان ایجـاد یـک سیـستم جلـوگیری-تـشخیص نفوذ(IDPS)



شکل ۱۲ – طرح پیشنهادی برای کارهای آتی

۶- مراجع و منابع

[1] [Hofmeyr07] "Intrusion Detection using Sequences of System Calls" By: Stephanie Forrest ,Steven A.Hofmeyr 2007

[2] [Forrest96] "A Sense of Self for Unix Processes" By: Stephanie Forrest ,Steven A.Hofmeyr, A. Longstaff University of New Mexico IEEE 1996

[3] [Wagner02] "Mimicry Attacks on Host-Based Intrusion Detection Systems" By: David Wagner , University of Colifornia , 2002

[4] [Hoang03] "A Multi-layer Model for Anomaly Intrusion Detection Using Program Sequences of System Calls"
By: Xuan Dau Hoang, University of Melbourne, Australia IEEE 2003.

[5] [Cmartin05] "A Comparison of System Call Feature Representations for Insider ThreatDetection"
By: Cheryl Martin,Alexander Liu,University of Texas at Austin, United States Military Academy IEEE 2005

[6] [Nguyen03] "Detecting Insider Threats by Monitoring System Call Activity"
By: Nam Nguyen ,University Of California , Los Angeles United States Military Academy Proceedings of the 2003 IEEE

[7] [GTandon03] "Learning Rules from System Call Arguments and Sequences for Anomaly Detection" By: Gaurav Tandon ICDM 2003

[8] [Eskin02] " Modeling System Calls for Intrusion Detection with Dynamic Window Sizes"
By: Eleazar Eskin & Wenke Lee , Computer Science Department Columbia University 2002

[9] [Libenzi04] "Guarded Memory Move (GMM) Buffer Overflow Detection And Analysis"
By: Davide Libenzi , January 17, 2004

[10] [Kyung02] "Type-Assisted Dynamic Buffer Overflow Detection"
By: Kyung-suk Lhee and Steve J. Chapin, Center for Systems Assurance Syracuse University 2002

[11] [HLA02] "Linux System Calls for HLA Programmers" By: 2002