علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۱۲، شماره ۱ (الف)، ۱۳۹۳
صفحات ۴۹-۵۹
مقاله عادی

ارائه یک مدل تحلیلی برای بررسی امنیت سیستم‌های نرم‌افزاری موجود از روی معماری آن

حبیب ایزدخواه ایاز عیسی‌زاده جابر کریم‌پور

دانشکده علوم ریاضی، دانشگاه تبریز، تبریز، ایران

چکیده

امروزه ارزیابی امنیت نرم‌افزار به عنوان یکی از صفات کیفی مهم نرم‌افزار دارای اهمیت زیادی است. تعداد زیادی سیستم نرم‌افزاری وجود دارند که امنیت در آن‌ها در نظر گرفته نشده است که آن‌ها را مستعد بروز خطرات امنیتی می‌کند. معماری نرم‌افزار یکی از طراحی‌های مهم نرم‌افزار است که روی کیفیت نهایی نرم‌افزار تاثیر می‌گذارد. تحقیقات زیادی روی صفات کیفی مانند کارایی و قابلیت اطمینان در سطح معماری نرم‌افزار بررسی شده است ولی از تاثیر معماری نرم‌افزار بر امنیت گزارشی ارائه نشده است. هدف از این مقاله، ارائه یک روش تحلیلی مبتنی بر ریاضیات است که امنیت نرم‌افزار را در سطح معماری نرم‌افزار ارزیابی کند. برای این منظور، در ابتدا با استفاده از روش مهندسی نرم‌افزار مبتنی بر تجربه، با در نظر گرفتن داده‌های حفره‌های امنیتی کشف شده روی مرورگر وب موزیلا فایر فاکس، ارتباط بین انواع اتصال (coupling) و حفره‌های امنیتی را نشان خواهیم داد. بعد از نشان دادن وجود چنین رابطه‌ای، یک رابطه ریاضی با استفاده از مفاهیم آماری رگرسیون بین انواع اتصال و میزان آسیب‌پذیری در سطح یک مولفه ارائه داده و سپس با استفاده از امید ریاضی (انتظار) رابطه‌ای ریاضی برای مشخص کردن میزان آسیب‌پذیری کل یک سیستم نرم‌افزاری ارائه خواهیم داد. در نهایت، با استفاده از ابزار استخراج معماری Bunch، معماری نرم‌افزار را استخراج و بازیابی کرده و معماری بازیابی شده را به زنجیره‌های مارکوف تبدیل نموده و از روی این زنجیره‌ها و رابطه ریاضی ارائه شده مبادرت به پیش‌بینی و ارزیابی امنیت یک سیستم نرم‌افزاری خواهیم نمود.

کلمات کلیدی: سیستم‌های نرم‌افزاری، معماری نرم‌افزار، امنیت و زنجیره مارکوف.

۱- مقدمه

این مقاله، حاصل قسمتی از یک طرح پژوهشی مشترک بین دانشگاه تبریز و دانشگاه صنعتی Middle East کشور ترکیه است. هدف از طرح پژوهشی ایجاد معماری پویا و تطبیق‌پذیر از روی کد منبع سیستم‌های نرم‌افزاری موجود می‌باشد. پیش‌بینی محققان مهندسی نرم‌افزار این است که پویایی و تطبیق‌پذیری با محیط از ویژگی‌های اصلی نسل بعدی سیستم‌های نرم‌افزاری و سیستم‌های با دسترسی بالا و طول عمر زیاد خواهد بود. سیستم نرم‌افزاری در صورت لزوم، باید بتواند خود را با محیطی که پیوسته در حال تغییر بوده تطبیق داده و در شرایط لازم عملکرد خود را در واکنش به تغییرات محیط تغییر دهد. سیستم‌های نرم‌افزاری پویا دارای معماری پویایی می‌باشند؛ بنابراین معماری نرم‌افزار رابطه بسیار نزدیکی با پویایی و تطبیق‌پذیر بودن سیستم نرم‌افزاری با محیط دارد. در واقع این معماری نرم‌افزار است که نحوه عملکرد سیستم نرم‌افزاری در شرایط مختلف و در تعامل با محیط را مشخص می‌کند. در واقع وقتی سیستم با موارد امنیتی مواجه می‌شود از معماری مبتنی بر امنیت استفاده کند و یا وقتی می‌خواهد روی سیستم‌های توزیعی استقرار گردد از معماری که موارد مرتبط با توزیع را رعایت می‌کند استفاده کند. تمام روش‌های ارائه شده برای ایجاد معماری پویا، روش‌هایی قابل اعمال در مراحل اولیه توسعه نرم‌افزار هستند و این روش‌ها فقط یک چارچوب کلی برای تعریف معماری فراهم آورده‌اند و هیچ الگویی برای این‌که معماری چگونه بتواند امنیت را فراهم کند، توزیع شدگی داشته باشد، کارآیی داشته باشد یا موجب مصرف بهینه انرژی شود؛ ارائه نداده‌اند. هدف ما از طرح پژوهشی مشترک، ایجاد انواع معماری‌ها از کد منبع و قرار دادن آن‌ها در داخل کد منبع برنامه می‌باشد تا در مواقع لزوم

نرم‌افزار بتواند با استفاده از معماری‌های داخل خود، خودش را با محیط تطبق دهد.

هدف این مقاله فقط تاکید روی قسمتی از بخش مرتبط با معماری امنیت طرح پژوهشی است و آن بررسی میزان آسیب‌پذیری کد منبع می‌باشد. اکثر کارشناسان حوزه امنیت، تهدید را از ناحیه بستر شبکه (که خود شامل سخت افزارهای شبکه، سیستم عامل‌های شبکه و پروتوکول‌ها می‌باشد) می‌دانند. حال آنکه آمارهای قابل استناد جهانی که در موسسات معتبر آمده خلاف این موضوع را بیان می‌کند. بر طبق بررسی موسسه گارتنر حدود ۷۵٪ از حملات، به برنامه‌های تحت شبکه صورت گرفته که فقط ۲۵٪ از آن حملات به شبکه و سرویس‌های شبکه بوده در حالیکه از هزینه‌های صرف شده در حوزه امنیت ۱۰٪ به بخش برنامه‌های تحت شبکه اختصاص داده شده و ۹۰٪ هزینه‌ها به بخش شبکه اختصاص یافته است. یک نرم‌افزار آسیب‌پذیر معمولا دارای نقصی است که این نقص می‌تواند روی امنیت آن تاثیرگذار باشد و به مهاجمان اجازه می‌دهد تا از آن برای اهداف خرابکارانه خود استفاده کنند [۱] [۲].

اتصال (coupling) یکی از صفات کیفی سیستم‌های نرم‌افزاری است که تاثیر زیادی روی کیفیت نهایی یک نرم‌افزار دارد [۳]. مهندسان نرم‌افزار معتقدند [۴] یک سیستم نرم‌افزاری دارای معماری خوب باید دارای اتصال حداقل و پیوستگی حداکثر باشد. اتصال به سطح ارتباطات خارجی یک پیمانه (مولفه) اطلاق می‌شود. دو پیمانه دارای اتصال قوی هستند اگر هر یک بر اساس دیگری عمل کند. اتصال زیاد بین پیمانه‌ها فهم برنامه را با مشکل مواجه می‌سازد [۵]. در حالت کلی پنج نوع از اتصال وجود دارد [۴] که بصورت مرتب از اهمیت بیشتر به کمتر عبارتند از (۱) content (۲) common (۳) control (۴) stamp و (۵) data. در مهندسی نرم‌افزار بدترین حالت زمانی است که ارتباط بین دو پیمانه از نوع content باشد و بهترین حالت زمانی است که ارتباط بین دو پیمانه از نوع data باشد. منظور از وابستگی از نوع data این است که ارتباط بین دو پیمانه فقط از نوع داده باشد. منظور از وابستگی از نوع stamp این است که ارتباط بین دو پیمانه از نوع ساختمان داده باشد. منظور از وابستگی از نوع control این است که یک پیمانه پارامتری را به پیمانه دیگر ارسال می‌کند که کنترل عملیات پیمانه دیگر را به دست بگیرد. منظور از وابستگی از نوع common این است که دو پیمانه از یک متغیر سراسری برای انجام ارتباطات استفاده می‌کنند. منظور از وابستگی از نوع content این است که یک پیمانه محتوای پیمانه دیگر را تغییر می‌دهد.

این کاری پیچیده است که یک سیستم نرم‌افزاری بدون استفاده از یک یا چند از انواع اتصال پیاده‌سازی کرد. به هر حال این قابل قبول است که تا حد ممکن باید از control، common و content اجتناب نمود. بدون در نظر گرفتن انواع اتصال، وقتی اطلاعات میان پیمانه‌ها مبادله می‌شود احتمال اینکه به خطر بیافتد زیاد است. تعدادی تحقیق نشان داد که اتصال زیاد فهم، توسعه، تست و نگهداری یک نرم‌افزار را مشکل می‌کند و به عنوان یک اثر جانبی ممکن است ریسک خطرپذیری نرم‌افزار را افزایش دهد [۶-۱۲]. در [۱۳] و [۱۴] نشان داده شد که اتصال بالا به افزایش احتمال انتشار خطر وقتی یک مشکل امنیتی اتفاق می‌افتد می‌تواند منجر شود. آینام [۱۵] در تحقیقات خود نشان داد که حملاتی از نوع SQL injection و Buffer overflow بر اثر انواع خاصی از اتصال بالا است. در مرجع [۱۶] با استفاده از روش‌های مهندسی نرم‌افزار مبتنی بر تجربه روی نرم‌افزار Denial of Service ارتباط بین اتصال و میزان آسیب‌پذیری نرم‌افزار نشان داده شده است.

در مرجع [۱۴] با استفاده از روش‌های مهندسی نرم‌افزار مبتنی بر تجربه روی نرم‌افزار موزیلا فایرفاکس ارتباط بین اتصال و میزان آسیب‌پذیری نرم‌افزار نشان داده شده است. آن‌ها برای نشان دادن این کار از یادگیری ماشین استفاده کرده‌اند و هیچ رابطه‌ای ریاضی برای نشان دادن این ارتباط ارائه ننموده‌اند. در واقع ابزار ارائه شده توسط آن‌ها تعداد اتصال به همراه تعداد پیوستگی و پیچیدگی یک برنامه را گرفته و میزان امنیت سیستم را پیش‌بینی می‌کند. ما در این تحقیق کار آن‌ها را ادامه داده و با استفاده از داده‌های گردآوری شده توسط آن‌ها برای نرم‌افزار موزیلا فایرفاکس فرمول خود را ایجاد می‌کنیم. تفاوت کار ما با کار آن‌ها در این است که:

۱. آن‌ها اتصال بین پیمانه‌ها را یک مفهوم کلی فرض نموده و ارتباط یک موجودیت با موجودیت دیگر را اتصال فرض می‌کنند در حالی‌که اتصال انواع متفاوتی دارد و نتایج آزمایش‌های ما نشان داد که تاثیر هر کدام روی امنیت متفاوت با بقیه است.

۲. روش پیشنهاد شده توسط آن‌ها فقط آسیب‌پذیر بودن یا نبودن یک مولفه یا فایل را مشخص می‌کند و هیچ اطلاعاتی در زمینه میزان آسیب‌پذیر بودن فراهم نمی‌آورد.

۳. روش پیشنهاد شده توسط آن‌ها فقط در سطح مولفه یا فایل کار می‌کند و هیچ اطلاعاتی در زمینه آسیب‌پذیر بودن کل سیستم فراهم نمی‌آورد.

اگرچه اثر معیار اتصال بطور تجربی و به طور موفقیت آمیزی در امنیت یک سیستم نرم‌افزاری نشان داده شده است ولی هیچ رابطه ریاضی برای نشان دادن چگونگی تاثیر اتصال روی امنیت ارائه نشده است. همچنین در رابطه با تاثیر انواع اتصال بحثی نشده است.

۱-۱- مساله

مساله کلی بررسی شده در این مقاله این است که چگونه می‌توان یک روش کاربردی و مفید و همچنین مبتنی بر ریاضیات برای پیش‌بینی میزان آسیب‌پذیری و امنیت سیستم‌های نرم‌افزاری موجود از روی معماری آن‌ها و تعداد انواع اتصال ارائه نمود. یک راه‌حل برای این مساله باید دارای ویژگی‌های زیر باشد:

۱- بررسی احتمال وجود ارتباط بین انواع اتصال و میزان آسیب‌پذیری.

۲- در صورت وجود ارتباط، ارائه یک رابطه ریاضی که چگونگی ارتباط بین انواع اتصال و میزان آسیب‌پذیری را نشان دهد.

۳- بررسی امنیت سیستم نرم‌افزاری موجود از روی کد منبع با توجه به رابطه ریاضی به دست آمده در بند ۲.

برای رسیدن به مسائل ارائه شده در بالا، در این مقاله یک رابطه ریاضی بر اساس انواع اتصال و امنیت ارائه خواهیم داد و سپس با استفاده از آن مبادرت به پیش‌بینی امنیت نرم‌افزار از روی معماری نرم‌افزار خواهیم پرداخت. برای این منظور، در بخش (۳) با استفاده از مفاهیم آنالیز آماری، از روی داده‌های حفره‌های امنیتی جمع‌آوری شده از روی نرم‌افزار موزیلا فایرفاکس، همبستگی بین انواع اتصال و میزان آسیب‌پذیری یک سیستم نرم‌افزاری را نشان خواهیم داد. بعد از مشخص کردن همبستگی، با استفاده از رگرسیون آماری رابطه‌ای ریاضی که نشان دهنده چگونگی ارتباط بین انواع اتصال و میزان آسیب‌پذیری یک سیستم نرم‌افزاری ارائه خواهیم داد. در نهایت در بخش (۴) برای ارزیابی ایمنی یک سیستم نرم‌افزاری، ابتدا با استفاده از ابزار Bunch معماری آن را از کد منبع استخراج خواهیم نمود و معماری استخراج شده را به زنجیره‌های مارکوف برای ارزیابی تبدیل خواهیم نمود. از روی زنجیره مارکوف و رابطه ارائه شده در بخش (۳) به ارزیابی سیستم نرم‌افزاری از روی معماری آن خواهیم پرداخت.

۲- پیش‌زمینه

در این بخش به بررسی مفاهیم معماری نرم‌افزار، موزیلا فایرفاکس، زنجیره مارکوف، همبستگی و رگرسیون می‌پردازیم. این مفاهیم در تشخیص میزان آسیب‌پذیری نرم‌افزار استفاده شده است.

۲-۱- معماری نرم‌افزار

تعریف مورد نظر ما از معماری نرم‌افزار همان تعریف ارائه شده در IEEE STD 610, 12 [۱۷] می‌باشد. طبق IEEE STD 610, 12 معماری یعنی ارائه توصیفی فنی از یک سیستم که نشان دهندهٔ ساختار اجزاء آن، ارتباط بین آن‌ها و اصول و قواعد حاکم بر طراحی و تکامل آن‌ها در گذر زمان باشد. در واقع طبق این تعریف معماری نرم‌افزار چگونگی کنار هم قرار گرفتن اجزای اصلی یک نرم‌افزار را نشان می‌دهند. پیمانه‌بندی سامانه نرم‌افزار، فعالیت اصلی پیدا کردن معماری مناسب می‌باشد [۱۸، ۱۹].

به عبارت دیگر با پیمانه‌بندی سامانه نرم‌افزاری به زیر سیستم‌ها، می‌توان به معماری دلخواه و مناسب رسید. هدف پیدا کردن یک تقسیم‌بندی از کلاس‌ها است به طوری‌که ارتباط و اتصال بین کلاس‌های دو پیمانه مختلف تا جای ممکن کاهش پیدا کرده و اتصال بین کلاس‌های یک پیمانه بیشینه شود؛ یعنی کلاس‌هایی که بسیار به هم وابسته هستند، در یک زیر سیستم (پیمانه) قرار بگیرند و برعکس، کلاس‌هایی با وابستگی کم یا مستقل، در زیر سیستم‌های متفاوتی قرار بگیرند.

از ابزارهای معروف برای استخراج معماری نرم‌افزار می‌توان به DAGC [۱۸] و Bunch [۱۹] اشاره نمود. در این مقاله ما از Bunch برای پیدا کردن معماری نرم‌افزار استفاده خواهیم کرد.

۲-۲- مروری بر نرم‌افزار موزیلا فایرفاکس

به دلیل اینکه داده‌های استفاده شده در این تحقیق از نرم‌افزار موزیلا فایرفاکس می‌باشد بنابراین در این بخش بطور مختصر بررسی شده است. نرم‌افزار موزیلا فایرفاکس یکی از مرورگرهای مشهور برای استفاده از وب می‌باشد و بیش از دو میلیون خط کد دارد و همچنین در حدود ۳۲۰ میلیون کاربر از آن استفاده می‌کنند [۲۰]. علاوه بر این، یک تاریخچه قوی از حفره‌های امنیتی آن که در طول چهار سال (January 26, 2005 to April 27, 2009) کشف و برطرف شده‌اند را دارا می‌باشد.

ما در این تحقیق ورژن‌های متفاوتی از این نرم‌افزار را مورد بررسی قرار داده‌ایم. تا March 1, 2009، پنجاه و دو ورژن از ورژن R1.0 تا ورژن R3.0.6 دارای حفره‌های امنیتی بودند که کشف و بر طرف شده‌اند [۲۱]. در طول دوره چهار ساله ذکر شده و روی پنجاه دو ورژن، ۷۱۸ فایل از کل ۱۱۱۳۹ فایل دارای ۱۴۵۰ مشکل امنیتی بودند که کشف و برطرف شدند.

۲-۳- زنجیره مارکوف

در این بخش به معرفی زنجیره مارکوف گسسته می‌پردازیم که برای مدل کردن معماری نرم‌افزار استفاده شده است [۲۲، ۲۳]. زنجیره مارکوف، دنباله‌ای از متغیرهای تصادفی است که همگی این متغیرهای تصادفی دارای فضای نمونه‌ای یکسان هستند اما توزیع احتمالات آنها می‌تواند متفاوت باشد و هر متغیر تصادفی در یک زنجیره مارکوف تنها به متغیر قبل از خود وابسته است.

$$p_{ij}(n) = P(X_{n+1} = j \mid X_n = i) \qquad (۱)$$

یک زنجیره مارکوف با حالت‌ها و تابع احتمال انتقال بین حالت‌ها مشخص می‌شود. در این مقاله از زنجیره مارکوف جاذب استفاده شده است. یک زنجیره مارکوف، جاذب نامیده می‌شود اگر یک حالتی در زنجیره وجود داشته باشد که آن حالت خروجی نداشته باشد. هر زنجیره مارکوف با چندین حالت جاذب می‌تواند به یک زنجیره مارکوف با یک حالت نهایی تبدیل شود. این کار بدین صورت انجام می‌گیرد که یک حالت به زنجیره اضافه می‌شود و سپس از تمام حالت‌های نهایی به حالت جاذب یالی رسم می‌شود. ماتریس انتقالات بین حالات در زنجیره مارکوف بصورت زیر می‌باشد:

$$P \stackrel{\text{def}}{=} \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{pmatrix}$$

می‌توان ماتریس انتقالات را برای یک زنجیره مارکوف جاذب بصورت زیر تقسیم‌بندی نمود.

$$P = \begin{bmatrix} 1 & 0 \\ C & Q \end{bmatrix} \qquad (۲)$$

اگر یک زنجیره مارکوف دارای n حالت و m حالت جاذب باشد Q یک زیر ماتریس $(n-m)\times(n-m)$ است که احتمال انتقال بین حالت‌هایی که جاذب نیستند را نشان می‌دهد. 1 یک ماتریس $m\times m$ همانی را نشان می‌دهد، 0 یک ماتریس $n\times(n-m)$ صفر است و C نشان دهنده احتمال انتقال بین حالت‌های غیرجاذب و جاذب را نشان می‌دهد. ماتریس اساسی F بصورت زیر نشان داده می‌شود:

$$F = (I-Q)^{-1} \qquad (۳)$$

I نشان دهنده ماتریس همانی است. فرض کنید $X_{i,j}$ نشان دهنده انتظار تعداد پیمایش‌های (امید ریاضی) حالت j با شروع از حالت i قبل از وارد شدن به یک حالت جاذب باشد که این با استفاده از عنصر (i, j) ماتریس اساسی F مشخص می‌شود.

$$E[X_{i,j}] = m_{i,j} \qquad (۴)$$

واریانس امید ریاضی تعداد پیمایش‌ها از روی ماتریس اساسی می‌تواند محاسبه شود. فرض کنید $\sigma_{i,j}$ نشان دهنده واریانس تعداد پیمایش‌های حالت j با شروع از حالت i باشد.

تعریف- ماتریس $F_D = [md_{i,j}]$ یک ماتریس قطری با قطرهای ماتریس اساسی می‌باشد که به صورت زیر تعریف می‌شود:

$$md_{i,j} = \begin{cases} m_{i,j} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \qquad (۵)$$

اگر F_2 را به صورت زیر تعریف کنیم:

$$F_2 = [m^2_{i,j}] \qquad (۶)$$

در این صورت خواهیم داشت:

$$\sigma^2 = F(2F_D - I) - F_2 \qquad (۷)$$

از اینرو:

$$Var\ [X_{i,j}] = \sigma^2_{i,j} \quad (۸)$$

برای یک نرم‌افزار شامل تعدادی پیمانه، می‌توانیم معماری نرم‌افزار را با یک زنجیره مارکوف نشان دهیم. حالت‌های زنجیره مارکوف نشان دهنده پیمانه‌ها و یال‌های بین حالت‌ها نشان دهنده انتقال کنترل از یک پیمانه به پیمانه دیگر می‌باشد.

۲-۴- ضرایب همبستگی و رگرسیون

ضریب همبستگی، یکی از معیارهای مورد استفاده در تعیین همبستگی دو متغیر است. ابزاری آماری برای تعیین نوع و درجه رابطه یک متغیر کمی با متغیر کمی دیگر است. ضریب همبستگی شدت رابطه و همچنین نوع رابطه (مستقیم یا معکوس) را نشان می‌دهد. این ضریب بین ۱ تا ۱- است و در عدم وجود رابطه بین دو متغیر، برابر صفر است.

متغیر مستقل، متغیری است که بر متغیرهای دیگر اثر می‌گذارد. در تحقیق آزمایشی، متغیر مستقل متغیری است که توسط محقق دستکاری می‌شود تا تأثیرش بر متغیر وابسته مشخص شود. متغیر وابسته، متغیری است که ارزش یا مقدار آن به متغیر مستقل بستگی دارد. متغیر وابسته در اختیار محقق نیست و محقق نمی‌تواند در آن دخل و تصرف و دستکاری به عمل آورد.

ما وجود یا عدم وجود ارتباط بین انواع اتصال و حفره‌های امنیتی را به وسیله محاسبه کردن همبستگی بین انواع اتصال و تعداد حفره‌های امنیتی در نرم‌افزار موزیلا فایرفاکس را بررسی خواهیم کرد. مقدار همبستگی میزان وابستگی را نشان می دهد. کوهن [۲۴] پیشنهاد کرد که همبستگی کمتر از ۰/۳ یعنی همبستگی ضعیف، بین ۰/۳ تا ۰/۵ همبستگی متوسط و بیشتر از ۰/۵ یعنی همبستگی قوی. در این تحقیق از فرمول همبستگی پیرسون [۲۴] برای نشان دادن همبستگی بین انواع اتصال و تعداد حفره‌های امنیتی در موزیلا فایرفاکس استفاده کرده‌ایم.

تحلیل رگرسیونی یک ابزار آماری مفید برای دیدن رابطه‌ی بین متغیرها بکار می‌رود. تحلیل رگرسیون فن و تکنیکی آماری برای بررسی و به مدل در آوردن ارتباط بین متغیرهاست. تحلیل رگرسیون این امکان را برای محقق فراهم می‌کند تا تغییرات متغیر وابسته را از طریق متغیر مستقل پیش‌بینی نماید و سهم هر یک از متغیرهای مستقل را نیز در تبیین متغیر وابسته تعیین نماید. تفاوت رگرسیون با ضریب همبستگی در این است که رگرسیون به دنبال پیش‌بینی است در حالی که ضریب همبستگی تنها میزان وابستگی ۲ متغیر را با هم مقایسه می‌کند و زمانی که بین دو متغیر همبستگی وجود داشته باشد می‌توان از طریق رگرسیون مقدار یک متغیر (Y) را از روی یک متغیر دیگر (X) پیش‌بینی یا برآورد کرد، و هر چه همبستگی بین متغیرها بالاتر باشد، به همان اندازه پیش‌بینی دقیق‌تر است.

برای مشخص کردن ضرایب متغیرهای وابسته معمولا از روش کمترین مربعات استفاده می‌کنند. کمترین مربعات در واقع روشی برای برازش (fit) داده‌ها است. در روش کمترین مربعات، بهترین مدل برازش‌شده بر مجموعه‌ای از داده‌ها مدلی است که در آن مجموع مربع باقی مانده‌ها (Sum of Squared Residuals) کمینه باشد. منظور از باقی مانده‌ها، اختلاف بین داده مشاهده شده و مقداری است که از مدل به دست می‌آید.

بعد از انجام رگرسیون از ضریب تعیین R^2 برای مشخص کردن میزان معنادار بودن رابطه بدست آمده استفاده خواهد شد. اگر $R^2 = 0.850$ باشد نشان دهنده این موضوع است که 85% تغییر در متغیر وابسته y به وسیله تغییر در متغیرهای مستقل x می‌باشد. بقیه 15% بوسیله عوامل نامشخص است.

۳- تشخیص میزان آسیب‌پذیری یک پیمانه

در تنها تحقیق در زمینه ارتباط بین صفات داخلی و خارجی در زمینه امنیت، در سال ۲۰۱۱ که در ژورنال system architecture منتشر شده است [۱۴] با استفاده از نتایج تجربی که روی نرم‌افزار موزیلا فایرفاکس انجام داده است نشان داده است که بین اتصال و امنیت رابطه وجود دارد (نوع رابطه مشخص نشده است.) آلکساندر ایوانوف [۲۵] در رساله دکتری خود در سال ۲۰۰۵ با بررسی کدهای نرم‌افزارهای مختلف نشان داد که امنیت نرم‌افزار با حفره‌های درون کد ارتباط معکوس دارد.

در تحقیقی که در سال ۲۰۱۱ در دانشگاه کوئینز کانادا انجام شده است [۱۴] مشخص شده است بین حفره‌های درون کد و اتصال ارتباط وجود دارد (نوع رابطه مشخص نشده است.) پرسمن [۴] در کتاب مهندسی نرم‌افزار خود انواع اتصال‌ها را به صورت زیر تقسیم‌بندی کرد:

Data - Stamp - Control - Common - Content

از موارد بیان شده می‌توان نتیجه گرفت: احتمالا بین موارد پنج‌گانه مورد نظر پرسمن و امنیت نرم‌افزار رابطه وجود دارد. برای نشان دادن وجود یا عدم وجود این ارتباط از ضریب همبستگی پیرسون استفاده شده است. جدول (۱) نشان دهنده این موضوع است که ارتباط بین انواع اتصال و میزان آسیب‌پذیری درست است و ارتباط معنی‌داری بین آن‌ها وجود دارد. بنابراین بعد از وجود چنین همبستگی، هدف ما ارائه رابطه ریاضی است که ارتباطی بین ارتباط انواع اتصال و میزان آسیب‌پذیری را نشان دهد.

با توجه به جدول (۱) مشاهده می شود که انواع اتصال رابطه مثبتی با تعداد حفره‌های امنیتی در موزیلا فایرفاکس را دارد. هدف ارائه یک رابطه‌ای (که با $P_m(x)$ نشان داده می‌شود) که ارتباط بین انواع اتصال و میزان آسیب‌پذیری را در سطح یک پیمانه نشان دهد، می‌باشد. در حالت کلی بین انواع اتصال و میزان آسیب‌پذیری رابطه مستقیم و بین آن‌ها و امنیت رابطه معکوس وجود دارد. یعنی:

$$\text{Coupling} \equiv \text{Vulnerability} \equiv 1/\text{Security} \quad (۹)$$

برای ارائه رابطه بین انواع اتصال و میزان آسیب‌پذیری در سطح یک پیمانه انواع رابطه‌ها را مورد بررسی قرار داده و R^2 آن‌ها را محاسبه نموده و مشاهده کردیم یک رابطه چند جمله‌ای می‌تواند برآورد خوبی داشته باشد. بنابراین در ابتدا رابطه (۱۰) را با حداکثر درجه ۳ پیشنهاد داده و با استفاده از داده‌های جمع‌آوری شده از نرم‌افزار موزیلا فایرفاکس و تحلیل رگرسیون ضرایب آن‌ها را مشخص نموده و در نهایت سعی خواهیم کرد با استفاده از آنالیز آماری جملاتی که در رابطه تاثیر چندانی ندارند حذف کنیم. تحلیل رگرسیون برای تعیین ضرایب با استفاده از نرم‌افزار SPSS (www.spsstools.net) انجام گرفته شده است.

$$P_m(x) = c + \sum_{i=1}^{5} w_i C_i + \sum_{i=1}^{5} \alpha_i C_i^2 + \sum_{i=1}^{5} \beta_i C_i^3 + \sum_{i,j=1, i\neq j}^{5} \varphi_{ij} C_i C_j \quad (۱۰)$$

$$m = 1 \ldots \text{Numberof Modules}$$

$|C_1|$= number of Data Dependency in a module
$|C_2|$= number of Control Dependency in a module
$|C_3|$= number of Stamp Dependency in a module
$|C_4|$= number of Common Dependency a in module
$|C_5|$= number of Content Dependency in a module

جدول ۱- همبستگی بین انواع اتصال و حفره‌های امنیتی در موزیلا فایرفاکس

Type of copling	Correlations with vulnerabilities				
	R1.0	R1.5	R2.0	R3.0	R3.0.6
Data	0.471	0.462	0.458	0.458	0.454
Stamp	0.482	0.482	0.482	0.471	0.469
Control	0.510	0.514	0.515	0.510	0.509
Common	0.537	0.536	0.536	0.521	0.519
Content	0.641	0.641	0.634	0.631	0.631

For all the correlations, $p<0.05$.

جدول ۲- ضرایب بدست آمده برای رابطه (۶)

c	w_1	w_2	w_3	w_4	w_5	α_1	α_2	α_3	α_4	α_5
0.177608	0.011917	0.024287	0.100231	0.347872	0.428921	0.000911	0.000923	0.000981	0.100533	0.110542
β_1	β_2	β_3	β_4	β_5	$\varphi_{1,2}$	$\varphi_{1,3}$	$\varphi_{1,4}$	$\varphi_{1,5}$	$\varphi_{2,3}$	$\varphi_{2,4}$
0.000012	0.000034	0.000038	0.000124	0.000139	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
$\varphi_{2,5}$	$\varphi_{3,4}$	$\varphi_{3,5}$	$\varphi_{4,5}$							
0.000000	0.000000	0.000000	0.000000							

c، w_i، α_i، β_i، φ_{ij} وزن و اهمیت انواع اتصال هستند. هدف تعیین c، $w_i, \alpha_i, \beta_i, \varphi_{ij}$ می‌باشد. C_iC_j نشان دهنده تاثیر انواع اتصال بر یکدیگر می‌باشند. تعداد کل این ضرایب مجهول ۲۶ می‌باشد. برای تعیین ضرایب از داده‌های نرم‌افزار موزیلا فایرفاکس استفاده شده است. ضرایب ذکر شده با روش حداقل مجموع مربعات بدست می‌آید. این مقادیر با کمینه کردن مجموع مربعات انحراف‌ها، یا $\sum(P_i(x)-\hat{P}_i(x))^2$ به دست می‌آید. برای این کار باید مجموع مربعات خطا نسبت به تک تک ضرایب ذکر شده مشتق گرفته و مساوی صفر قرار گیرد. برای تعیین C_i ها از نرم‌افزار موزیلا فایرفاکس استفاده شده است. این نرم‌افزار دارای ۱۱۱۳۹ فایل است که از این تعداد فایل، ۷۱۸ فایل (از ویرایش R1.0 تا ویرایش R2.0.0.9 جمعا ۳۲ ویرایش) دارای مشکلات امنیتی بوده‌اند که درست شده‌اند. در این مقاله برای مشخص کردن ضرائب فرمول خود از این ۷۱۸ فایل استفاده شده است بنابراین یک دستگاه معادلات با ۷۱۸ معادله و ۲۶ مجهول بایستی حل شود.

روش انجام کار بدین صورت است که برای هر فایل موزیلا فایرفاکس با استفاده از نرم‌افزار Ndepend (www.ndepend.com) تعداد و نوع اتصال و همچنین با استفاده از تاریخچه حفره‌های امنیتی موزیلا فایرفاکس [۲۶] تعداد و نوع حفره‌ها را مشخص می‌کنیم (تعداد حفره‌ها در فایل i را با $P_i(x)$ نمایش داده شده است) و آن‌ها را در رابطه (۱۰) جاگذاری می‌کنیم. ضرایب به دست آمده در جدول (۲) نشان داده شده است.

بعد از بدست آمدن ضرایب، ضریب تعیین یا R^2 محاسبه شد که این مقدار برای مدل رابطه (۱۰)، 0.58 می‌باشد. هرچه مقدار R^2 بیشتر باشد بیانگر برازش بهتر مدل می‌باشد. عدد 0.58 نشان می‌دهد ارتباط بین متغیرهای مستقل و متغیر وابسته معنی‌دار است. حال به بررسی هر یک از معیارها (متغیرهای مستقل) با در نظر گرفتن ضرایب آن‌ها می‌پردازیم. هدف این است که بررسی کنیم کدامیک در چند جمله‌ای تاثیر معنی‌داری دارد. برای این منظور از t-Statistic استفاده کردیم. برای بررسی معناداری هر ضریب بایستی مقدار t محاسباتی با مقدار t جدولی با درجه آزادی n-k مقایسه گردد.

به طور تقریبی برای معناداری هر ضریب در سطح ۹۵٪ بایستی داشته باشیم: $|t| \geq 2$. جدول (۳) مقدار t-Statistic برای تمام ضرایب بدست آمده در جدول (۲) را نشان می‌دهد.

بنابراین مدل کلی پیشنهاد شده برای نشان دادن ارتباط بین تعداد انواع اتصال و میزان آسیب‌پذیری هر پیمانه بعد از حذف معیارهایی که تاثیر معناداری روی آسیب‌پذیری ندارند (یعنی $|t|<2$) بصورت زیر می‌باشد:

$$P_m(x) = c + (\sum_{i=1}^{5} w_i C_i) + w_4 C_4^2 + w_5 C_5^2 \qquad (۱۱)$$
$$m = 1 \ldots \text{Number of Modules}$$

بعد از حذف متغیرهایی که تاثیر معناداری روی متغیر وابسته نداشتند دوباره مقدار R^2 را محاسبه شد که 0.79 شد. در واقع 0.79 نشان می‌دهد که مدل چند جمله‌ای رابطه (۱۱) می‌تواند بخوبی ارتباط بین انواع اتصال و میزان آسیب‌پذیری در سطح یک پیمانه را نشان دهد. مقدار $P_m(x)$ می‌تواند از صفر تا یک عدد بزرگ باشد. سوال اینجاست که چه مقدار $P_m(x)$ چه مقدار آسیب‌پذیری را نشان می‌دهد. با تغییر کوچکی در $P_m(x)$ می‌توان آن را به بازه [0, 1] نگاشت نمود.

$$DV_m = 1 - (1 / P_m(x)) \text{ where } 0 < DV_m < 1$$
$$m = 1 \ldots \text{Number of Modules} \qquad (۱۲)$$

که در این صورت می‌توان گفت هرچقدر DV_m به عدد یک نزدیکتر آسیب‌پذیری پیمانه بیشتر خواهد بود.

۳-۱- اندازه‌گیری کارایی پیش‌بینی‌کننده میزان آسیب‌پذیری یک پیمانه

کارایی یک پیش‌بینی کننده با استفاده از چندین روش می‌تواند اندازه‌گیری شود. اغلب روش‌های متداول برای اندازه‌گیری کارایی پیش‌بینی کننده عبارتند از:

accuracy, recall, precision, false positive rate, and false negative rate

این روش‌ها برای اندازه‌گیری کارایی از ماتریس نامنظم نشان داده شده در جدول (۴) استفاده می‌کنند. این ماتریس، واقعی در مقابل پیش‌بینی شده را نشان می‌دهد که در آن:

جدول ۳- مقدار t-Statistic برای همه ضرایب رابطه (۶)

C	w_1	w_2	w_3	w_4	w_5	α_1	α_2	α_3	α_4	α_5
2.618319	2.076532	2.087455	5.12145	13.46926	24.54306	1.002316	1.074754	1.439125	5.000328	5.010423
β_1	β_2	β_3	β_4	β_5	$\varphi_{1,2}$	$\varphi_{1,3}$	$\varphi_{1,4}$	$\varphi_{1,5}$	$\varphi_{2,3}$	$\varphi_{2,4}$
0.363781	0.363841	0.473241	0.498342	0.843962	0.001280	0.025712	0.024329	0.876575	0.004759	0.700932
$\varphi_{2,5}$	$\varphi_{3,4}$	$\varphi_{3,5}$	$\varphi_{4,5}$							
0.039341	0.459121	0.042461	0.981241							

True Negative (TN): تعداد فایل‌هایی که پیش‌بینی شده‌اند احتمال آسیب‌پذیری ندارند بطوریکه هیچ آسیب‌پذیری در آن‌ها کشف نشده است.

False Negative (FN): تعداد فایل‌هایی که پیش‌بینی شده‌اند که احتمال آسیب‌پذیری ندارند که در واقع آن‌ها آسیب‌پذیری دارند.

False Positives (FP): تعداد فایل‌هایی که پیش‌بینی شده‌اند که دارای احتمال آسیب‌پذیری هستند در حالیکه واقعا آن‌ها آسیب‌پذیر نیستند.

True Positives (TP): تعداد فایل‌هایی که پیش‌بینی شده‌اند که احتمال آسیب‌پذیری دارند که آن‌ها واقعا آسیب‌پذیر هستند.

جدول ۴- ماتریس نامنظم

Actual	Predicted as	
	Not Vulnerable	Vulnerable
Not Vulnerable	TN=True Negatives	FP=False Positives
Vulnerable	FN= False Negatives	TP= True Positives

از روی ماتریس نامنظم چندین اندازه‌گیری کننده کارایی مانند زیر می‌تواند استخراج شود:

Accuracy, precision, recall, F-measures, false positive rate, and false negative rate

Accuracy: معروف است به نرخ طبقه‌بندی درست و برابر است با تعداد فایل‌هایی که به درستی پیش‌بینی شده‌اند (آسیب‌پذیر بودن یا نبودن) به تعداد کل فایل‌ها، که در رابطه (۱۳) نشان داده شده است.

$$Accurancy = \frac{TP + TN}{TP + FP + TN + FN} \quad (۱۳)$$

Precision: برابر است با تعداد فایل‌هایی به درستی آسیب‌پذیر پیش‌بینی شده‌اند به تعداد کل فایل‌هایی که آسیب‌پذیر پیش‌بینی شده‌اند که در رابطه (۱۴) نشان داده شده است.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (۱۴)$$

Recall: به این صورت تعریف می‌شود نرخ تعداد فایل‌هایی که به درستی آسیب‌پذیر پیش‌بینی شده‌اند به تعداد کل فایل‌هایی که واقعا آسیب‌پذیر هستند. فرمول برای محاسبه Recall در رابطه (۱۵) نشان داده شده است.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (۱۵)$$

هر دو معیار precision و recall معیارهای مهمی برای ارزیابی کارایی روش‌های پیش‌بینی کننده هستند. Precision بیشتر، زمان کمتری برای تست و بررسی تلف می‌شود؛ و recall بیشتر، یعنی فایل‌های آسیب‌پذیری کمتری تشخیص داده نشده است. باید یک تعادلی بین precision و recall وجود داشته باشد. برای مثال، اگر یک پیش‌بینی کننده فقط یک فایل را به عنوان فایل آسیب‌پذیر پیش‌بینی کند و آن فایل واقعا آسیب‌پذیر باشد در این صورت precision برابر ۱۰۰٪ خواهد بود. به هر حال، اگر فایل‌های آسیب‌پذیر دیگری وجود داشته باشد که شناسایی نشده باشند در این صورت recall کم خواهد بود. در یک مثال دیگر، اگر پیش‌بینی کننده همه فایل‌ها را به عنوان آسیب‌پذیر پیش‌بینی کند در این صورت recall برابر با ۱۰۰٪ خواهد بود اما precision آن کم خواهد بود. بنابراین بهتر خواهد بود که precision و recall با یکدیگر ترکیب شوند.

F-measure: این معیار میانگین وزن دار precision و recall می‌باشد. در بهترین حالت مقدار آن صد و در بدترین حالت مقدار آن صفر می‌باشد.

$$F_\beta - measure = \frac{(1+\beta^2) \times \text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \quad (۱۶)$$

F_1-measure باعث می‌شود اهمیت هر دو معیار precision و recall یکسان باشد و این همان میانگین هارمونیک precision و recall خواهد بود. در F_2-measure وزن و اهمیت recall دو برابر precision در نظر گرفته می‌شود و در $F_{0.5}$-measure وزن و اهمیت precision دو برابر recall در نظر گرفته می‌شود. اغلب محققان از معیارهای false positive rate (FP rate) و false negative rate (FN rate) به جای precision و recall استفاده می‌کنند. استراند و ویوکر [۲۷] اهمیت و کارایی این دو معیار ذکر شده را در تحقیقاتشان نشان دادند. ما نیز معتقد هستیم این اندازه‌گیرها در ارزیابی کارایی مدل‌های پیش‌بینی کننده میزان آسیب‌پذیری کارا هستند. این معیارها به صورت زیر تعریف می‌شوند:

$$\text{FP rate} = \frac{FP}{FP + TN} \quad (۱۷)$$

$$\text{FN rate} = \frac{FN}{TP + FN} \quad (۱۸)$$

FN بیشتر، نشان می‌دهد که یک ریسکی در تشخیص کامل آسیب‌پذیری‌ها وجود دارد یعنی آسیب‌پذیری وجود دارد ولی تشخیص داده نشده است؛ و در حالیکه FP بیشتر، نشان می‌دهد که تلاش ممکن است در تشخیص موجودیت‌های آسیب‌پذیر پیش‌بینی شده به هدر رود یعنی آسیب‌پذیری وجود ندارد ولی اشتباها تشخیص داده شده است. این معیارها به طور خیلی زیادی به precision و recall مرتبط هستند. بنابراین اینکه همه آن‌ها را در مشخص کردن کارایی یک پیش‌بینی

کننده به کار ببریم زیاد است. در این تحقیق ما همانند مرجع [۲۸] از accuracy، recall و FN rate استفاده کردیم. نتایج همه اندازه‌گیری ها به درصد بیان شده‌اند. حال می‌خواهیم کارایی رابطه (۱۲) را در پیش‌بینی میزان آسیب‌پذیری پیمانه‌ها بررسی کنیم. برای این کار، از ۲۰ ویرایش R2.0.0.10 تا R3.0.6 نرم‌افزار موزیلا فایرفاکس استفاده می‌کنیم. هدف این است که مشخص کنیم رابطه (۱۲) تا چه حدی می‌تواند در پیش‌بینی میزان آسیب‌پذیری‌ها در سطح یک مولفه یا پیمانه درست عمل کند. نتایج ارزیابی کارایی رابطه (۱۲) در جدول (۵) آورده شده است.

جدول ۵- کارایی رابطه (۱۲) در پیش‌بینی میزان آسیب‌پذیری در نرم‌افزار موزیلا فایرفاکس از ویرایش‌های R2.0.0.10 تا R3.0.6

Accuracy	Recall	FP rate	F_1-measure
72.51	69.22	28.12	68.00

برای بررسی بیشتر رابطه (۱۲) جهت پیش‌بینی میزان آسیب‌پذیری مولفه‌ها، ده مولفه دارای بیشترین حفره‌های امنیتی پیدا شده در نرم‌افزار Apache مورد بررسی قرار گرفته است. بنابراین، میزان آسیب‌پذیری این مولفه‌ها توسط رابطه (۱۲) محاسبه شده است. نتایج در جدول (۶) آورده شده است. همانگونه که در جدول (۶) مشاهده می‌شود مقدار DV_m برای این مولفه‌های آسیب‌پذیر بالا و نزدیک عدد یک می‌باشد. این جدول نشان می‌دهد که رابطه (۱۲) می‌تواند به خوبی برای نشان دادن میزان آسیب‌پذیری یک مولفه به کار رود.

۴- تشخیص میزان آسیب‌پذیری کل سیستم نرم‌افزاری

بعد از محاسبه کردن میزان آسیب‌پذیری یک پیمانه، به بررسی میزان آسیب‌پذیری کل سیستم نرم‌افزاری می‌پردازیم. اگر DV_i نشان دهنده شاخص آسیب‌پذیری پیمانه i باشد و $X_{1,i}$ نشان دهنده متغیر تصادفی متناظر با تعداد بازدیدهای حالت i و با شروع از حالت ۱ است بنابراین احتمال اینکه یک پیمانه در طول اجرای خود در سیستم نرم‌افزاری بدون مشکلات امنیتی عمل کند برابر است با $[(1-DV_i)^{X_{1,i}}]$ از اینرو، احتمال اینکه پیمانه iام دارای مشکلات امنیتی در طول اجرای یک سیستم نرم‌افزاری باشد برابر است با:

$$1-[(1-DV_i)^{X_{1,i}}] \quad (۱۹)$$

بنابراین برای یک سیستم نرم‌افزاری با n پیمانه (یا مولفه) شاخص آسیب‌پذیری می‌تواند به صورت زیر تعریف شود:

$$DV = 1-\prod_i^n (1-DV_i)^{X_{1,i}} \quad (۲۰)$$

که DV متغیر تصادفی است که نشان دهنده شاخص آسیب‌پذیری کل یک سیستم نرم‌افزاری است. بنابراین امید ریاضی (انتظار) شاخص آسیب‌پذیری کل یک سیستم نرم‌افزاری به صورت زیر می‌باشد:

$$\begin{aligned} E[DV] &= E\left[1-\prod_i^n (1-DV_i)^{X_{1,i}}\right] \\ &= 1-E\left[\prod_i^n (1-DV_i)^{X_{1,i}}\right] \\ &= 1-\prod_i^n E[(1-DV_i)^{X_{1,i}}] \end{aligned} \quad (۲۱)$$

با استفاده از بسط سری تیلور، می‌توان رابطه $E[(1-DV_i)^{X_{1,i}}]$ بصورت زیر نوشت.

$$\begin{aligned} E[(1-DV_i)^{X_{1,i}}] &= (1-DV_i)^{m_{1,i}} + \\ &\frac{1}{2}((1-DV_i)^{m_{1,i}} \times (\log(1-DV_i))^2 \sigma_{1,i} \end{aligned} \quad (۲۲)$$

بطوریکه تعداد پیمایش آخرین پیمانه همیشه ۱ است. یعنی $E[X_{1,n}]=1$ و $Var[X_{1,n}]=0$ بطوریکه:

$$E[(1-DV_n)^{X_{1,n}}] = 1-DV_n \quad (۲۳)$$

از این رو، انتظار شاخص آسیب‌پذیری یک سیستم نرم افزاری از مرتبه دو بصورت زیر است.

$$E(DV) = 1-\left[\prod_i^{n-1}\begin{bmatrix}(1-DV_i)^{m_{1,i}} + \frac{1}{2}((1-DV_i)^{m_{1,i}} \\ \times(\log(1-DV_i))^2\sigma_{1,i}\end{bmatrix}\right](1-DV_n) \quad (۲۴)$$

اگر از واریانس تعداد بازدیدها صرف نظر شود، از بسط سری تیلور $(1-DV_i)^{m_{1,i}}$ رابطه زیر بدست می‌آید:

$$1-E[(1-DV_i)^{X_{1,i}}] \cong 1-\left[1+\sum_k^n \binom{m_{1,i}}{k}(-)^k DV_i^k\right] \quad (۲۵)$$

و اگر از جمله های مرتبه بالا صرف نظر کنیم، داریم:

$$1-E[(1-DV_n)^{X_{1,i}}] \cong 1-[1-m_{1,i}DV_i] = m_{1,i}DV_i \quad (۲۶)$$

جدول ۶- ارتباط میان DV_m و ده مولفه دارای بیشترین میزان آسیب‌پذیری در نرم‌افزار Apache

Component	Bug Count	DV_m
Jsobj. cpp	24	0.937
nsCSSFrameConstructor.cpp	17	0.913
Jsfun.cpp	15	0.891
nsScriptSecurityManager.cpp	15	0.881
NsGlobalWindow.cpp	14	0.891
Jscsript.cpp	14	0.894
Jsinterp.cpp	14	0.894
nsDOMClassinfo.cpp	10	0.892
nsGenericElement.cpp	10	0.892
nsDOCShell.cpp	9	0.832

ما $m_{1,i}DV_i$ را شدت تاثیر یک مولفه روی ناامنی یک سیستم نرم افزاری تعریف می‌کنیم. گذرگاه امنیت یک سیستم نرم‌افزاری مولفه‌ای است که دارای بیشترین مقدار $1-E[(1-DV_n)^{X_{1,i}}]$ یا بطور ساده دارای بیشترین مقدار $m_{1,i}DV_i$ (اگر از واریانس تعداد بازدیدها و از جملات مرتبه دوم صرف نظر شود) است. جملات مرتبه دوم زمانی می‌تواند صرف نظر شود که DV_i ها بسیار کوچک باشند.

۴-۱- آنالیز حساسیت

در این بخش می‌خواهیم میزان اثرپذیری کل سیستم نرم‌افزاری از تغییر در تعداد پارامترهای مرتبط با امنیت (به عنوان مثال تغییر در تعداد حفره‌های امنیتی یک پیمانه) یک پیمانه را بررسی کنیم. برای بررسی اثر تغییرات در پارامترهای مرتبط با امنیت مولفه k بر روی کل سیستم نرم‌افزاری از دیفرانسیل استفاده می‌کنیم. برای انجام این کار مشتق امید ریاضی شاخص آسیب‌پذیری یک سیستم بر DV_k را محاسبه می‌کنیم. بنابراین:

$$\frac{dE[DV]}{dDV_k}=\left[\begin{array}{l} m_{i,k}(1-DV_k)^{m_{1,k}-1}+ \\ \frac{1}{2}\sigma_{1,k}^2\left(m_{1,k}(1-DV_k)^{m_{1,k}-1}(\log(1-DV_k))^2+\frac{2\log(1-DV_k)}{(1-DV_k)}(1-DV_k)^{m_{1,i}}\right)\end{array}\right]\times \left[\prod_{i=1,i\neq k}^{n-1}\left((1-DV_i)^{m_{1,i}}+\frac{1}{2}((1-DV_i)^{m_{1,i}})(\log(1-DV_i))^2\sigma_{1,i}^2\right)\right](1-DV_n) \quad (۲۷)$$

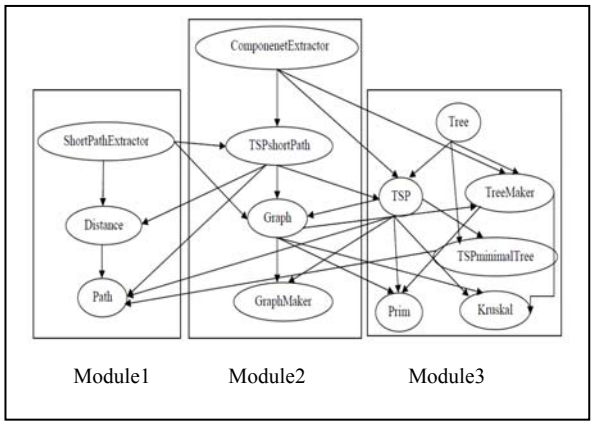
با در نظر گرفتن شاخص آسیب‌پذیری تغییر یافته یک سیستم نرم‌افزاری بصورت DV_{rev}، تعریف می‌کنیم ΔDV_k را بصورت تغییر در شاخص امنیت در مولفه k و اگر شاخص اصلی امنیت DV_{org} باشد داریم:

$$E[DV_{rev}]=E[DV_{org}]+(dE[DV]/dDV_n)\Delta DV_k \quad (۲۸)$$

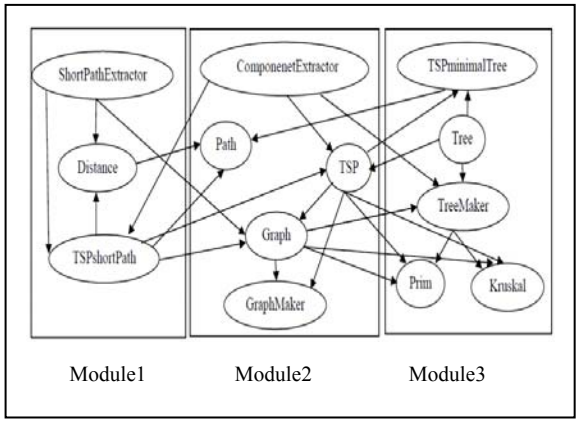
۵- مورد مطالعه

در این بخش به ارزیابی روش پیشنهاد شده می‌پردازیم. برای این منظور از مساله معروف فروشنده دوره‌گرد (TSP) استفاده می‌کنیم. برای ارزیابی روش پیشنهادی باید معماری TSP را با استفاده از ابزار Bunch از کد منبع آن استخراج شود. ورودی ابزار Bunch گراف فراخوانی و خروجی آن معماری نرم‌افزار می‌باشد. برای استخراج گراف فراخوانی TSP از روی کد منبع آن از ابزار تجاری NDepend استفاده شده است.

ابزار NDepend برای بیشتر زبان‌های برنامه‌سازی مشهور دنیا می‌تواند گراف فراخوانی را از کد منبع استخراج کند. بعد از استخراج گراف فراخوانی بایستی آن را برای استخراج معماری مناسب، پیمانه‌بندی نمود. برای ارزیابی امنیت TSP از روی معماری آن، دو معماری با کیفیت تقریبا یکسان ولی با پیمانه‌بندی‌های مختلف از روی کد آن استخراج استخراج شده است. شکل (۱) دو معماری استخراج شده برای مساله TSP از روی کد منبع آن را نشان می‌دهد. یال بین دو پیمانه نشان دهنده وابستگی از نوع اتصال می‌باشد. علاوه بر نوع اتصال تعداد آن‌ها نیز مهم است. بعد از استخراج معماری، برای ارزیابی امنیت، آنها را به زنجیره‌های مارکوف تبدیل می‌کنیم. زنجیره مارکوف برای معماری ۱ (الف) بصورت شکل (۲) و برای معماری ۱ (ب) بصورت شکل (۳) می‌باشد. اعداد روی یال‌ها نشان دهنده احتمال حرکت از یک پیمانه به پیمانه دیگر است. به عنوان مثال در شکل ۱ (الف) از پیمانه ۳ کلا ۵ یال خارج شده است که ۲ تا از آن‌ها به پیمانه ۱، دوتا به پیمانه ۲ و با ۱ یال به پیمانه ۴ رفته است. پیمانه ۴ به عنوان حالت نهایی به زنجیره مارکوف اضافه شده است.

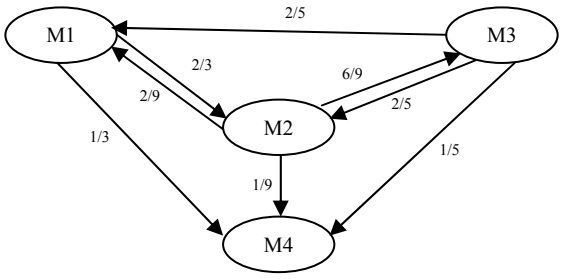


(الف)

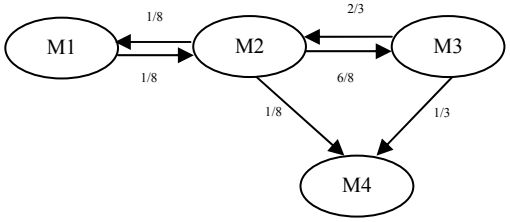


(ب)

شکل ۱- پیمانه‌بندی‌های به دست آمده برای TSP با استفاده از Bunch



شکل ۲- زنجیره مارکوف بدست آمده برای شکل ۱ (الف)



شکل ۳- زنجیره مارکوف بدست آمده برای شکل ۱ (ب)

ماتریس احتمال انتقال برای شکل (۲) و (۳) بصورت شکل (۴) است.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/3 & 0 & 2/3 & 0 \\ 1/9 & 2/9 & 0 & 6/9 \\ 1/5 & 2/5 & 2/5 & 0 \end{bmatrix}$$

(الف)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1/8 & 0 \\ 1/8 & 1/8 & 0 & 6/8 \\ 1/3 & 0 & 2/3 & 0 \end{bmatrix}$$

(ب)

شکل ۴– (الف) ماتریس انتقال برای شکل (۲) ؛ (ب) ماتریس انتقال برای شکل (۳)

ماتریس اساسی F برای شکل ۴ (الف) و ۴ (ب) بصورت شکل (۵) خواهد شد.

$$F = \begin{bmatrix} 1.766 & 1.584 & 1.045 \\ 1.161 & 2.400 & 1.584 \\ 1.175 & 1.594 & 2.052 \end{bmatrix}$$

(الف)

$$F = \begin{bmatrix} 1.028 & 0.227 & 0.150 \\ 0.227 & 1.822 & 1.202 \\ 0.150 & 1.202 & 1.793 \end{bmatrix}$$

(ب)

شکل ۵– (الف) ماتریس اساسی برای شکل ۴ (الف)؛ (ب) ماتریس اساسی برای شکل ۴ (ب)

DV_i برای پیمانه‌های ۱، ۲ و ۳ شکل‌های ۱ (الف) و ۱ (ب) در جدول ۷ نشان داده شده است.

جدول (۸) شاخص امنیت یعنی DV را برای هر دو معماری با در نظر گرفتن جدول (۷)، نشان می‌دهد.

جدول ۷- DV_i برای پیمانه‌بندی‌های (الف) و (ب) در شکل (۱)

	پیمانه‌بندی (الف)			پیمانه‌بندی (ب)		
	Module 1	Module 2	Module 3	Module 1	Module 2	Module 3
DV_i	0.81	0.89	0.91	0.47	0.81	0.65

جدول ۸- DV برای پیمانه‌بندی‌های (الف) و (ب) در شکل (۱)

	پیمانه‌بندی (الف)	پیمانه‌بندی (ب)
DV	0.75	0.46

با توجه به جدول (۸) اگر امنیت مهم باشد پیمانه‌بندی (ب) پیشنهاد می‌شود.

۶- محدودیت‌های تحقیق

در ارائه یک رابطه ریاضی مرتبط با امنیت کد منبع با محدودیت‌هایی روبرو بودیم. در زیر به دو مورد از این محدودیت‌ها پرداخته شده است:

اول، ما از این حقیقت آگاه هستیم که فاکتورهای زیادی است که می‌تواند به آسیب‌پذیری در سیستم‌های نرم‌افزاری منجر شود. با توجه به اینکه هدف ما از این تحقیق فقط تاکید روی معماری نرم‌افزار بود؛ ما پیشنهاد کردیم که انواع اتصال می‌تواند به عنوان مهمترین فاکتور برای ارزیابی امنیت در سطح معماری مورد استفاده قرار گیرد. اما ما اعلام می‌کنیم، به هیچ وجه، انواع اتصال به تنهایی معیار کاملی نخواهد بود. معیار دیگری که در سطح معماری استفاده می‌شود چسبندگی (cohesion) است که می‌تواند روی آسیب‌پذیری تاثیرگذار باشد. به هر حال تاثیر اتصال روی آسیب‌پذیری بیشتر از تاثیر چسبندگی است زیرا اتصال با ارتباطات خارجی یک پیمانه در ارتباط است و چسبندگی با ارتباطات داخلی یک پیمانه.

دوم، فرمول ارائه شده در این بخش، بر اساس حفره‌های امنیتی بود که تا زمان جمع آوری آن‌ها کشف و گزارش شده بودند. حفره‌های امنیتی که کشف نشده بودند یا اعلام همگانی نشده‌اند در این تحقیق در نظر گرفته نشده‌اند.

۷- نتیجه‌گیری

هدف از این مقاله ارائه روشی برای مشخص کردن میزان آسیب‌پذیری سیستم‌های نرم‌افزاری موجود از روی کد منبع آنها بود. برای انجام دادن این کار، یک روش برای مشخص کردن میزان آسیب‌پذیری در سطح معماری نرم‌افزار با استفاده از روش مهندسی نرم‌افزار مبتنی بر تجربه ارائه دادیم. از آنجائیکه نرم‌افزار موزیلا یک نرم‌افزار کد منبع باز است و تاریخچه همه حفره‌های امنیتی در آن مشخص است از آنها برای نشان دادن همبستگی بین انواع اتصال و آسیب‌پذیری نرم‌افزار استفاده کردیم. نتیجه آزمایش‌های ما نشان داد که همبستگی معناداری بین بین انواع اتصال و میزان آسیب‌پذیری نرم‌افزار وجود دارد. سپس یک رابطه ریاضی برای نشان دادن ارتباط بین تعداد اتصال با تعداد مشکلات امنیتی در یک پیمانه و در نهایت در کل سیستم نرم‌افزاری ارائه داده شد. در نهایت، با استفاده از ابزار استخراج معماری Bunch، معماری نرم‌افزار را استخراج و بازیابی کردیم و معماری بازیابی شده را به زنجیره‌های مارکوف تبدیل نموده و از روی این زنجیره‌ها مبادرت به پیش‌بینی و ارزیابی امنیت سیستم نرم‌افزار نمودیم.

مراجع

[1] A. Isazadeh, I. Elgedawy, J. Karimpour, and H. Izadkhah, "An Analytical Security Model for Existing Software Systems," *Journal of Applied Mathematics and Information Sciences*, vol. 8, no. 2, pp. 691-702, 2014.

[2] I. Krsul, *Software Vulnerability Analysis*, Ph. D. Dissertation, Purdue University, West Lafayette, Indiana, USA, 1998.

[3] F. Tsui, and O. Karam, *Essentials of Software Engineering*, Jones and Bartlett Publishers, 2010.

[4] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 2010.

[5] J. K. Kearney, R. L. Sedlmeyer, W. Thompson, M. Gray, and M. Adler, "Software complexity measurement," *ACM Trans. Communications*, vol. 29, no. 11, pp. 1044-1050, 1986.

[6] G. Koru, and J. Tian, "An empirical comparison and characterization of high defect and high complexity modules," *Journal of Systems and Software*, vol. 67, no.3, pp. 153-163, 2003.

[7] M. Janes, W. Scotto, B. Pedrycz, M. Russo, and G. Stefanovic, "Identification of defect-prone classes in telecommunication software systems using design metrics," *Journal of Systems and Software*, vol. 17, no. 6, pp. 3711-3734, 2006.

[8] G. Succi, W. Pedrycz, M. Stefanovic, and J. Miller, "Practical assessment of the models for identification of defect-prone classes in object-oriented commercial systems using design metrics," *Journal of Systems and Software*, vol. 65, no. 2, pp. 1-12, 2003.

[9] K. O. Elish, and M. O. Elish, "Predicting defect-prone software modules using support vector machines," *Journal of Systems and Software*, vol. 81, no. 1, pp. 649-660, 2008.

[10] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," *Proc. IEEE Intl Conf. Software Engineering*, pp. 452-461, 2006.

[11] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Software Engineering*, vol. 33, no. 9, pp. 2-13, 2007.

[12] H. Zhang, X. Zhang, and M. Gu, "Predicting defective software components from code complexity measures," *Proc. IEEE Intl Symp. Dependable Computing*, pp. 93-96, 2007.

[13] I. Chowdhury, B. Chan, and M. Zulkernine, "Security metrics for source code structures," *Proc. IEEE Intl Workshop on Software Engineering for Secure Systems*, pp. 57-64, 2008.

[14] I. Chowdhury, and M. Zulkernine, "Using Complexity, Coupling, and Cohesion Metrics as Early Indicators of Vulnerabilities," *Journal of Systems Architecture*, vol. 57, no. 5, pp. 294-313, 2011.

[15] V. S. Ayanam, *Software Security Vulnerability vs Software Coupling: A Study with Empirical Evidence*, Ms. Thesis, Southern Polytechnic State University, Marietta, Georgia, USA, 2009.

[16] M. Y. Liu, and I. Traore, "Empirical Relations between Attackability and Coupling: A case Study on DoS," *Proc. ACM SIGPLAN Workshop on Programming Languages and Analysis for Security*, pp. 57-64, 2006.

[17] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Boston: Addison-Wesley, 2003.

[18] O. Bushehrian, "A New Encoding Scheme and a Framework to investigate Genetic Clustering Algorithms," *Journal of Research and Practice in Information Technology*, vol. 37, no. 1, pp. 127-143, 2005.

[19] B. S. Mitchell, *A Heuristic Search Approach to Solving the Software Clustering Problem*, Ph. D. Dissertation, Drexel University, Philadelphia, 2002.

[20] "SciTools Inc. Blog," http://scitools.com/blog/2008/10/tip-understand-the countpath-metric.html, June 2008.

[21] "Mozilla Vulnerabilities," http://www.mozilla.org/projects/security/knownvulnerabilities, May 2007.

[22] U. N. Bhat, *Elements of Applied Stochastic Processes*, John Wiley and Sons, 1984.

[23] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, John Wiley and Sons, 2001.

[24] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, New York: Academic Press, 1988.

[25] I. Alexander, *Automatic Vulnerability Detection Using Static Source Code Analysis*, Ph. D. Dissertation, Graduate School of the University of Alabama, Alabama, 2005.

[26] Bugzilla, http://www.bugzilla.org, May 2006.

[27] T. J. Ostrand, and E. J. Weyuker, "How to Measure Success of Fault Prediction Models," *Proc. IEEE Intl Workshop on Software Quality Assurance*, pp. 25-30, 2007.

[28] L. Kuang, and M. Zulkernine, "An Anomaly Intrusion Detection Method Using the CSIKNN Algorithm," *Proc. ACM Symp. Applied Computing*, pp. 921-926, 2008.

حبیب ایزدخواه دارای مدرک دکتری در زمینه سیستم‌های کامپیوتری از دانشگاه تبریز است. او هم اکنون استادیار گروه علوم کامپیوتر دانشگاه تبریز است. مباحث مربوط به مهندسی نرم‌افزار از جمله علایق پژوهشی وی است.

آدرس پست‌الکترونیکی ایشان عبارت است از:

izadkhah@tabrizu.ac.ir

ایاز عیسی‌زاده دارای مدرک دکتری در رشته علوم کامپیوتری از دانشگاه کوئیز کانادا است. او هم اکنون استاد گروه علوم کامپیوتر دانشگاه تبریز است. مباحث مربوط به مهندسی نرم‌افزار و روش‌های رسمی از جمله علایق پژوهشی وی است.

آدرس پست الکترونیکی ایشان عبارت است از:

isazadeh@tabrizu.ac.ir

جابر کریم‌پور دارای مدرک دکتری در زمینه سیستم‌های کامپیوتری از دانشگاه تبریز است. او هم اکنون استادیار گروه علوم کامپیوتر دانشگاه تبریز است. مباحث مربوط به روش‌های رسمی در مهندسی نرم‌افزار و امنیت سیستم‌های کامپیوتری از جمله علایق پژوهشی وی است.

آدرس پست الکترونیکی ایشان عبارت است از:

karimpour@tabrizu.ac.ir

اطلاعات بررسی مقاله:

تاریخ ارسال: ۹۲/۱/۳۰

تاریخ اصلاح: ۹۲/۴/۲۴

تاریخ قبول شدن: ۹۲/۵/۲۴

نویسنده مرتبط: دکتر حبیب ایزدخواه، دانشکده علوم ریاضی، دانشگاه تبریز، تبریز، ایران.