نشریه علمی «نوآوری‌های فناوری اطلاعات و ارتباطات کاربردی»

سال اول، شماره ۳، پاییز ۱۴۰۰؛ ص ۶۶- ۵۵

علمی – تخصصی

مبهم‌سازی نرم‌افزار به‌وسیله‌ی تحلیل سلسله مراتبی و شبکه‌های پتری

محمد خانجانی[۱]، سعید پارسا[۲]

۱- فارغ التحصیل کارشناسی ارشد دانشگاه علم و صنعت ایران، تهران

۲-دانشیار، گروه آموزشی مهندسی کامپیوتر-نرم افزار، دانشگاه علم و صنعت ایران، تهران

(دریافت: ۱۴۰۰/۰۵/۱۵، پذیرش: ۱۴۰۰/۰۹/۲۱)

چکیده

مبهم‌سازی در مفهوم کلی، تکنیکی است که باعث می‌شود تا درک و تحلیل کدهای یک برنامه سخت‌تر شود که باهدف امنیت نرم‌افزار موردنظر صورت می‌گیرد. برای مهندسی معکوس به‌طورمعمول از گراف جریان کنترلی استفاده می‌شود. درنتیجه می‌بایست این گراف جریان کنترلی را به طریقی مبهم نمود. در این مقاله گراف جریان کنترلی با استفاده از شبکه‌های پتری چند نخی همراه با انتخاب‌کننده سلسله مراتبی مبهم می‌شود. بحث اصلی بر سر تقسیم کدها به قسمت‌هایی می‌باشد به‌طوری‌که برنامه در نخ‌های جداگانه اجرا شود. این نخ‌ها می‌بایست توسط مدیر نخ، مدیریت شوند تا در اجرای آن‌ها مشکلی پیش نیاید. هر چه شبکه پتری که برای مدل‌سازی برنامه استفاده می‌شود پیچیده‌تر باشد، مهندسی معکوس آن نیز دشوارتر می‌شود. کوچکترین تغییر در روند اجرایی با استفاده از نقاط توقف در انتخاب انتخاب‌کننده سلسله مراتبی تاثیر می گذارد و در نتیجه باعث می‌شود برنامه روند اصلی خود را طی نکند.

کلیدواژه‌ها: شبکه‌های پتری چند نخی، انتخاب‌کننده سلسله مراتبی، گراف جریان کنترلی، مهندسی معکوس، محافظت نرم‌افزار، مدیریت نخ.

۱- مقدمه

به‌طورکلی بحث مبهم‌سازی نرم‌افزار بدین معنی می‌باشد که در کدهای برنامه و یا در اجرای برنامه تغییراتی ایجاد شود که فهم برنامه را دشوار و یا در مواردی غیرممکن سازد [۱]. با گسترش روش‌ها و ابزارهای جدید مهندسی معکوس می‌بایست راهکارهایی را اندیشید که در مقابل این روش‌ها مقاومت نمایند تا نرم‌افزار به‌راحتی مورد مهندسی معکوس قرار نگیرد. برای مقاومت در برابر مهندسی نرم‌افزار نیاز نیست که حتماً به‌طور کامل جلوی آن گرفته شود بلکه اگر بتوان مدت تحلیل را به مقداری بسیاری افزایش داد نیز خود راهکاری برای مقاومت می‌باشد.

به‌طورکلی مبهم‌سازی یک مسیر یک‌طرفه می‌باشد که می‌تواند با افزودن افزونه‌هایی در جهت محافظت از نرم‌افزار استفاده شود. در اینجا گفته شد یک‌طرفه، بدین معنی که امکان برگشت به کد اولیه وجود ندارد و در صورتی هم که امکان‌پذیر باشد، تنها در شرایط خاص، بسیار پیچیده و مشکل می‌باشد [۲].

مهندسی معکوس که امروزه صورت می‌گیرد، معمولاً با تحلیل پویا انجام می‌شود بدین معنی که نقاط توقفی در برنامه قرار داده می‌شود و نحوه عملکرد برنامه با استفاده از این نقاط توقف و ردیابی جریان کنترلی انجام می‌شود.

به‌طورکلی مبهم‌سازی باید دارای قوانینی باشد. برای اینکه برنامه p1 بتواند به برنامه p2 که مبهم شده برنامه p1 می‌باشد طی تغییرات T تبدیل شود، می‌بایست:

- برنامه p2 از برنامه p1 ساخته‌شده باشد و p2 قابل‌اجرا و همچنین دارای همان عملکرد p1 باشد.
- تحلیل و مهندسی معکوس برنامه p2 می‌بایست دارای پیچیدگی بیشتری نسبت به برنامه اولیه p1 باشد.
- هیچ راه مؤثری برای برگشت از برنامه p2 به برنامه p1 وجود نداشته باشد [۳].

با استفاده از مبهم‌سازی می‌توان از استفاده‌ی غیرقانونی از نرم‌افزار جلوگیری نمود.

مبهم‌سازی می‌تواند در چند سطح صورت گیرد که عبارت‌اند از[۴]:

- کد منبع برنامه
- کد میانه که می‌توان آن را بین کد ماشین و کد منبع دانست برای مثال IL در زبان‌های دات نت.
- کد ماشین

* رایانامه نویسنده مسئول: parsa@iust.ac.ir

مبهم‌سازی در سطح کد منبع دارای معایبی می‌باشد همچون اینکه خیلی از روش‌ها را کامپایلرها نمی‌توانند قبول و ایجاد نمایند و یا نمی‌توانند در مقابل روش‌های ترکیبی ایستا و پویا مقاومت کنند چراکه در سطح کد برنامه می‌باشند و دارای قدرت بالایی نمی‌باشند. به‌طورکلی کد نویسی آن‌ها در سطح بالایی می‌باشد[۵][۶].

مبهم‌سازی در سطح کد میانه دارای فواید بسیاری می‌باشد ولی پیاده‌سازی آن کمی دشوار می‌باشد مخصوصاً درصورتی‌که نیاز باشد که چند محیطی[1] نیز باشد در آن صورت نیازمند به ترجمه گر[2] می‌باشد تا بتواند در محیط‌های مختلف کارایی خود را حفظ نماید.

مبهم‌سازی در سطح ماشین که خاص‌منظوره می‌باشد ولی دارای امنیت بسیار بالایی نسبت به بقیه می‌باشد. مبهم‌سازی در این سطح حتی از مبهم‌سازی در سطح کد میانه نیز دارای دردسرهای بیشتری می‌باشد.

در مقاله [۷] روشی جهت مبهم‌سازی شرط وقوع رفتارها در کد برنامه ارائه داده شده است تا درصورت تحلیل نمادین کد، کاربران از شروط واقعی یک رخداد مطلع نگردند. برای این منظور در مقاله [۷]، یک راه‌کار نوین با اعمال معادلات خطی ارائه داده شده است. در این روش با جایگذاری برخی شروط غیرواقعی و جدید در مسیر برنامه و مرتبط‌نمودن متغیرهای آن با متغیرهای اصلی برنامه، حل‌کننده را دچار اشتباه می‌نماید. این امر موجب ایجاد شاخه-های متعدد و غیرواقعی در درخت نمادین برنامه می‌گردد. لذا، تحلیل کد را دچار پیچیدگی می‌کند. این روش دارای دقت به نسبت خوبی است ولی پیچیدگی و سربار بالایی دارد که استفاده از آن را تقریبا در برنامه های بزرگ غیر ممکن می‌کند.

در مقاله [۸] از ایده مبهم‌سازی برای پنهان سازی اطلاعات استفاده کرده است و ایده اصلی روش پیشنهادی در این مقاله پنهان کردن اطلاعات، جایگزینی برخی رشته‌های تولید شده به‌طور تصادفی که بخشی از کد مرده معرفی‌شده هستند، با پیام مخفی کدگذاری شده است.

در مقاله [۹] راهکاری با نام Loki ارائه شده است که ادعا شده است قادر است تا برابر حملات بسیار مختلف و شناخته شده مقاوم عمل کند. این راهکار مبتنی روشی ترکیبی می باشد که از ترکیب‌های ۳ تا ۴ روش ایجاد شده است و امنیت به نسبت خوبی دارد ولی سربار پیاده‌سازی آن بسیار بالاست.

در این مقاله قصد بر این است که گراف جریان کنترلی برنامه مبهم شود تا با این کار برنامه در مقابل مهندسی معکوس مقاومت کند و یا به‌طورکلی بعضی از روش‌های مهندسی معکوس را غیرممکن نماید و همچنین مبهم‌سازی مورداستفاده در این مقاله در سطح کد میانه می‌باشد.

در زمینه مبهم‌سازی با شبکه‌های پتری تلاش‌هایی صورت گرفت ولی به دلیل سادگی شبکه‌های پتری استفاده‌شده، دارای نقاط ضعف فراوانی بودند و کاربر بدخواه با تحلیل عمیق برنامه می‌توانست روند درست برنامه را دریابد و همین‌طور در زمینه مدیریت نخ‌ها نیز مشکلات فراوانی وجود داشت ولی در این مقاله از مدیر نخ برای مدیریت نخ‌ها استفاده‌شده است و همین‌طور اینکه شبکه‌های استفاده‌شده نیز با استفاده از روش‌هایی همنچون استفاده از شبکه‌های پتری سه بعدی می‌توان مبهم‌تر نمود. سعی شده است که در این مقاله مشکلات روش‌های قبلی برطرف شود البته باید خاطرنشان کرد که استفاده از شبکه‌های پتری چند نخی در مبهم‌سازی، بحث جدیدی است که در رابطه با آن مقالات زیادی موجود نمی‌باشد. برای تغییر حالت در شبکه‌های پتری با استفاده از قاعده‌ای ساده صورت می‌گیرد که این می‌تواند باعث مشکلات فراوانی شود (کاربر متخاصم می‌تواند با اندکی کوشش روند آن را درک نماید). ولی با استفاده از انتخاب‌کننده سلسله مراتبی که در اینجا از آن استفاده شده است تقریبا راه تحلیل بطور کامل بسته شده است چرا که کوچکترین تغییری در روند اجرا باعث تاثیری گذاری روی وزن معیارها و گزینه ها می‌شود و در نتیجه انتخابی اشتباه صورت گرفته و روند اصلی برنامه پیموده نمی‌شود.

در این مقاله، ابتدا شبکه پتری توضیح داده می‌شود. بعدازآن تحلیل سلسله مراتبی که در انتخاب‌کننده مورد نظر در اینجا از آن کمک گرفته شده است توضیح داده می‌شود. در ادامه روش پیشنهادی بازگو می‌شود و در انتها ارزیابی، نتیجه‌گیری و منابع آورده شده است.

۲- شبکه‌های پتری

تئوری شبکه‌های پتری از کارهای A.W.Holt Jack Denis، Carl Adam Petri و ... توسعه یافت و عموماً به‌عنوان ابزاری برای مطالعه و مدل‌سازی سیستم‌ها تلقی می‌گردد. یک شبکه پتری یک توصیف ریاضی است ولی در ضمن یک نمایش گرافیکی یا بصری نیز می‌باشد [۱۰].

[1] Multi-Platform
[2] Translator

شبکه‌های پتری از ۴ جز پایه برای مدل کردن سیستم تشکیل شده‌اند[۱۱]:

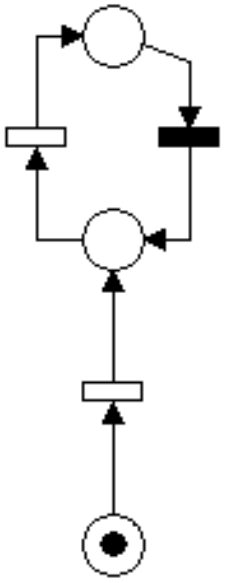
- مکان(P): برای نمایش اجزا سیستم و حالت آنها به کار می‌رود همانند درایور دیسک، یک برنامه یا منابع دیگر.
- تراکنش (T): برای توصیف رویدادهای مختلفی که منجر به وضعیت‌های مختلفی در سیستم می شود به کار می‌رود. به‌عنوان مثال عمل خواندن یک قطعه از روی دیسک یا نوشتن بر روی دیسک
- Arc: روابطی که بین تراکنش‌ها و مکان‌ها وجود دارد را نشان می‌دهند. برای اتصال مکان‌ها و تراکنش‌ها به کار می‌رود. در حقیقت مثل راهی برای فعال کردن یک تراکنش هستند.
- توکن (Token): برای تعریف وضعیت شبکه‌های پتری به کار می‌روند. توکن‌ها در مدل‌های اولیه، نشانه های غیرتوصیفی هستند که در مکان‌ها قرار می‌گیرند و در تعریف علامت‌گذاری شبکه‌های پتری به کار می‌روند. هر مکان می‌تواند فاقد توکن یا دارای یک یا چند توکن باشد.

شبکه‌های پتری به کمک نمایش مجموعه‌ها نیز قابل توصیف هستند که به صورت پنج تایی M = (P,T,I,O,MP) نشان داده میشوند. اعضای این پنج تایی عبارتند از: P مجموعه مکان‌ها، T مجموعه تراکنش‌ها، I توابع ورودی تراکنش‌ها که مکان‌ها را به تراکنش‌ها نگاشت می‌کنند، O توابع خروجی تراکنش‌ها که تراکنش‌ها را به مکان‌ها می‌نگارند، MP که نشانه‌گذاری شبکه را به کمک توکن‌های موجود در مکان‌ها نشان می‌دهد.

از طرفی نمایش گرافیکی، شبکه‌های پتری را به‌عنوان گراف دو قسمتی جهتدار نیز بیان می‌کند. در گراف دو قسمتی جهتدار، گره‌ها به دو مجموعه مجزا تقسیم می‌شوند به قسمی که برای هر یال موجود در این گراف، یک انتهای آن در یک مجموعه گره و انتهای دیگرش در مجموعه دیگر است. اشتراک این دو مجموعه گره تهی و اجتماع آنها کل گره‌های گراف را تشکیل می‌دهد.

در شبکه‌های پتری مکان‌ها و تراکنش‌ها به‌عنوان این دو مجموعه مجزا از گره‌ها در نظر گرفته می‌شوند و arcها در حکم یال اتصال دهنده آنها می‌باشند که یه سر آنها به مکان‌ها و سر دیگرشان به تراکنش‌ها متصل است[۱۲].

شکل (۱) نمونه‌ای از یک شبکه پتری را نمایش می‌دهد.



شکل (۱): یک نمونه شبکه پتری [۱۲]

۲-۱- علامت‌گذاری شبکه پتری

علامت m، علامت‌دهی یا قرار دادن یک سری مهره در مکان‌های شبکه پتری است. یک‌مهره، مفهوم ابتدایی نظیر T,P از شبکه پتری را دارد. پس‌ازاینکه مهره‌ها تعریف شدند و داخل مکان‌های (P) شبکه پتری قرار گرفتند، تعداد و وضعیت مهره‌ها در اثنای اجرای یک شبکه پتری می‌تواند تغییر کند. به عبارت دیگری می‌توان گفت که مهره‌ها مورداستفاده قرار می‌گیرند تا بتوانند اجرای شبکه پتری را تعریف کنند.

M=(c,m) یک شبکه پتری C=(P,T,I,O) و یک علامت m است که اغلب نوشته می‌شود: M=(P,T,I,O,m)، بردار تعداد مهره‌های آن مکان را نشان می‌دهد.

شبکه پتری، به‌وسیله شلیک گذرها اجرا می‌شود. یک گذر T به‌وسیله برداشتن (مهره) از مکان‌های ورودی و ایجاد مهره‌های جدید مجزا در مکان‌های خروجی اجرا می‌گردد. یک گذر در صورتی شلیک می‌کند که مجاز یا فعال باشد و در صورتی مجاز است که هرکدام از مکان‌های ورودی گذر حداقل دارای تعداد مهره‌ای برابر با کمان‌های ورودی به آن گذر (که به‌صورت پیکان از مکان P به گذر T است) باشد [۱۳].

۳- تحلیل سلسله مراتبی

با توجه به محدودیت عقلائی که هر انسان به تنهائی دچار آن است، شاید تشریک مساعی گروهی تنها راه دستیابی به یک تصمیم‌گیری منطقی، منظم، جامع و کامل باشد. فرایند تحلیل

سلسله‌مراتبی[1] یکی از معروفترین فنون تصمیم‌گیری چندمعیاره است که اولین بار توسط توماس ال ساعتی عراقی الاصل در دهه ۱۹۷۰ ابداع گردید. این روش هنگامیکه تصمیم‌گیری با چند گزینه رقیب و معیار تصمیم‌گیری روبرو است می‌تواند استفاده گردد. فرآیندهای تحلیل سلسله مراتبی از قسمت های طراحی پرسشنامه خبره، تعیین وزن معیارها، مقایسه زوجی گزینه‌ها براساس معیارها و محاسبه اولویت‌ها تشکیل می شوند.

۳-۱- طراحی پرسشنامه خبره

پرسشنامه مورد استفاده برای تحلیل‌های سلسله‌مراتبی و تصمیم‌گیری چندمعیاره به پرسشنامه خبره موسوم است. برای تهیه پرسشنامه خبره از مقایسه زوجی گزینه‌ها استفاده می‌شود. برای هر سطح از سلسله مراتب یک پرسشنامه خبره تهیه می‌شود. برای امتیاز دهی از مقیاس نه درجه ساعتی به صورت جدول (۱) استفاده می‌شود. البته در اینجا این پرسشنامه از افراد متخصص در زمینه تحلیل برنامه استفاده شده است تا بتوان بهترین معیارها را برای مبهم‌سازی استفاده نمود[۱۴][۱۵].

جدول (۱): مقیاس نه درجه ساعتی

ارزش	وضعیت مقایسه i نسبت به j	توضیح
۱	ترجیح یکسان	شاخص i نسبت به j اهمیت برابر دارد و یا ارجحیتی نسبت به هم ندارند.
۳	کمی	گزینه یا شاخص i نسبت به j کمی مهمتر است.
۵	خیلی	گزینه یا شاخص i نسبت به j مهمتر است.
۷	خیلی زیاد	گزینه i دارای ارجحیت خیلی بیشتری از j است.
۹	کاملاً	گزینه i از j مطلقاً مهمتر و قابل مقایسه با j نیست.
۲-۴-۶-۸	بینابین	ارزش‌های بینابین را نشان می‌دهد مثلا ۸، بیانگر اهمیتی زیادتر از ۷ و پایین‌تر از ۹ برای iاست.

پژوهشگران معمولا از طیف پنج نقطه جدول (۲) استفاده می‌کنند که ساده‌تر بوده و نتایج یکسانی به‌دست می‌دهد.

جدول (۲): طیف پنج نقطه استفاده شده توسط پژوهشگران

ترجیح یکسان	کمی بهتر	بهتر	خیلی بهتر	کاملا بهتر
۱	۳	۵	۷	۹

با استفاده از این مقیاس هیات مدیره هر یک از گزینه‌ها را براساس هر یک از عوامل به صورت زوجی مقایسه می‌کنند.

۳-۲- تعیین وزن معیارها

سطح اول سلسله مراتب را معیارهای اصلی تشکیل می‌دهد. پرسشنامه خبره با مقایسه زوجی معیارهای اصلی براساس هدف به تعیین اولویت هر یک از معیارهای اصلی می‌پردازد. بنابراین باید معیارها را براساس هدف دوبه‌دو با هم مقایسه نمود.

اکزل و ساعتی (۱۹۸۳) استفاده از میانگین هندسی را بهترین روش برای ترکیب مقایسات زوجی معرفی کرده‌اند. بنابراین از داده‌های هر سطر میانگین هندسی گرفته می‌شود. وزن‌های به‌دست آمده نرمال نیستند. منظور از وزن نرمال آن است که جمع اوزان برابر ۱ باشد. بنابراین میانگین هندسی به‌دست آمده در هر سطر بر مجموع میانگین هندسی ستون‌ها تقسیم می‌شود. ستون جدید که حاوی وزن نرمال شده هر معیار است را بردار ویژه گویند. وزن نهایی هر ماتریس همان ستون بردار ویژه است[۱۶].

هر معیار ممکن است خود از یک مجموعه زیرمعیار تشکیل شده باشد. در اینصورت یک سطح دیگر به مدل تحلیل سلسله مراتبی اضافه می‌شود (تحلیل شبکه‌ای[2]) یعنی دیگر نمی‌توان از مدل تحلیل سلسله مراتبی استفاده نمود و می بایست از تحلیل شبکه‌ای استفاده نمود که در چندین سطح می‌تواند معیارها را در نظر بگیرد ولی در راه حل پیشنهادی که در این مقاله ارائه می‌شود تنها یک سطح کافی است از اینرو از تحلیل سلسله مراتبی در این مورد استفاده می‌شود. باید خاطر نشان کرد که هر چه سطح استفاده شده چپایین تر باشد، سربار محاسباتی نیز خیلی پایین تر می آید و در این راه حل که با روند اجرایی درگیر است سرعت از معیارهای مهمی می باشد که نمی‌توان از آن چشم پوشی کرد.

۳-۳- مقایسه زوجی گزینه‌ها براساس معیارها

پس از تعیین وزن هر یک از معیارها در گام بعد باید گزینه‌ها به‌صورت زوجی براساس هر معیار مقایسه شوند. بعد از اینکه مقایسه ها انجام شد داده‌ها را به ماتریسی منتقل می‌کنند که همان ماتریس مقایسه زوجی است.

[1] Analytical Hierarchy Process(AHP)

[2] Analytical Network Process(ANP)

گام بعدی تعیین اولویت است. برای تعیین اولویت از مفهوم نرمال‌سازی که در گام قبلی توضیح داده شد استفاده می‌شود. پس از نرمال کردن وزن هر گزینه براساس معیار مورد نظر به‌دست خواهد آمد. به عبارت دیگر محاسبه مقدار ویژه هر سطر با تخمین میانگین هندسی (میانگین هندسی آن سطر به جمع میانگین هندسی سطرها) می باشد[۱۷].

می‌توان از نرم افزارهای Expert Choice و یا Super Decision برای سرعت در روند محاسبه استفاده نمود. مقایسه‌های زوجی برای تمامی معیارها انجام می‌شود. به این ترتیب اولویت هر گزینه انتخابی براساس هر معیار محاسبه می‌شود.

۳-۴- محاسبه اولویت‌ها

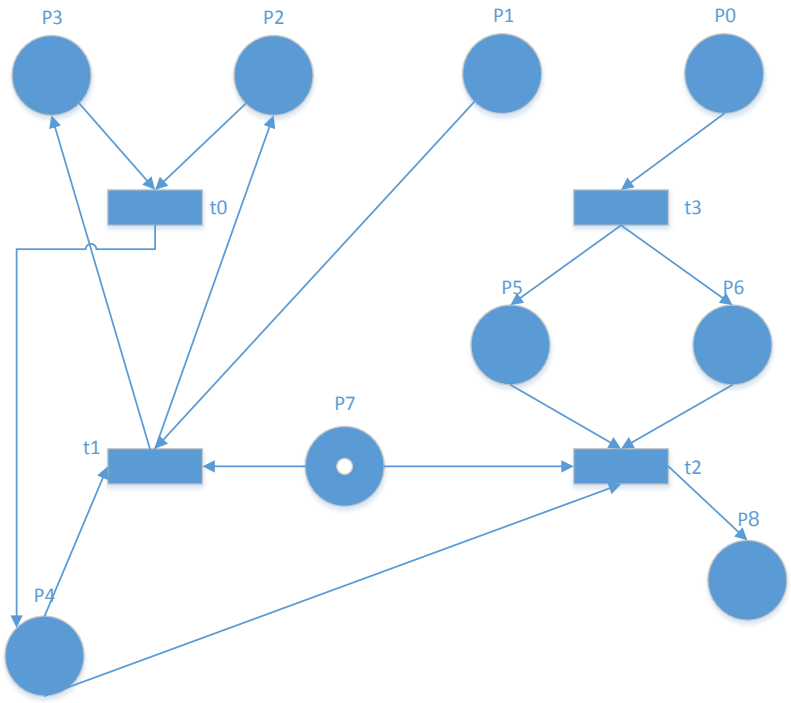
اکنون به سادگی با استفاده از میانگین موزون نود مورد نظر انتخاب می‌شود. برای محاسبه امتیاز از فرمول (۱) استفاده می‌شود.

فرمول (۱): امتیاز هر گزینه = مجموع حاصل‌ضرب اولویت آن گزینه براساس معیار i ضربدر اولویت آن معیار

گزینه‌ای که دارای بیشترین امتیاز باشد به‌عنوان گزینه مورد نظر انتخاب می‌شود.

۴- راه‌حل پیشنهادی برای مبهم‌سازی

برای مبهم‌سازی برنامه می‌توان از شبکه‌های پتری استفاده نمود. در این روش با استفاده از شبکه‌های پتری می‌توان روند درک برنامه را بسیار طولانی کرد و همچنین مدت‌زمان تحلیل را نیز بسته به شبکه‌ی پتری استفاده‌شده، بیشتر نمود. این روش به خودی خود روند درک برنامه را بسیار مشکل می نماید اما برای پیچیده تر کردن این روند می‌توان از انتخاب‌کننده استفاده نمود طوریکه رفتن از حالتی به حالت دیگر طبق انتخاب این انتخاب‌کننده انجام می‌شود و حالت دیگر برای تغییر حالت توسط همین قسمت انتخاب‌کننده انجام می‌شود. در این روش روند اجرای برنامه به شبکه پتری مدل می‌شود. نمونه‌ای از شبکه پتری که می‌توان استفاده نمود را در شکل (۲) می‌توان مشاهده نمود.



شکل (۲): نمونه‌ای از شبکه پتری برای مبهم‌سازی

در این راه‌حل در نظر گرفته‌شده است که هر یک از گذرهای t0…t3، بیانگر یک قطعه از کد می‌باشند که دارای زمان اجرای خاصی می‌باشند. زمان اجرا را می‌توان طبق [۱۲] و [۱۸] تخمین زد (منظور از زمان اجرای قطعه کد در شبکه‌های پتری، زمان مورد نیاز برای شلیک می‌باشد). این تخمین بدین صورت می‌باشد که برنامه به قطعاتی قابل جداسازی تقسیم می‌شود و برای هرکدام از این قسمت‌ها یک دامنه زمانی در نظر گرفته می‌شود که یک حداکثری را دارا می‌باشد. ابزار مهندسی معکوس موجود

برای بررسی می‌بایست نقاط توقفی را در برنامه ایجاد نمایند البته این نقاط توقف می‌تواند متفاوت ولی به‌هرحال از مدت‌زمانی که هر گذر نیاز دارد مطمئناً بیشتر می‌باشد. چون در این حالت زمان شلیک گذرها به علت وجود نقاط توقف تغییر می‌کنند بنابراین می‌توان شبکه پتری که قرار است برنامه به آن مدل شود را طوری ایجاد نمود که با کمی تغییر در زمان شلیک جریان گذرها از مسیر درست دور شود، در نتیجه برنامه به پایان درست نمی‌رسد. البته چون جریان اجرایی برنامه به‌صورت چند نخی می‌باشد و گذرها متمایز از هم عمل می‌کنند، روند مهندسی معکوس بسیار پیچیده می‌شود و در صورت استفاده از نقاط توقف جریان اصلی برنامه از هم می‌پاشد و برنامه به پایان درست نمی‌رسد و حمله‌کننده هم نمی‌تواند از جریان برنامه چیزی درک نماید چراکه جریان اصلی طی نشده است. در صورتی که تنها از شبکه پتری استفاده شود، امکان دارد که در روند اجرای درست نیز به ندرت مشکلاتی پیش آید و این مشکلات موجب می‌شود که برنامه در شرایط درست نیز روند اصلی خود را پیمایش نکند. یکی از مشکلات اصلی در روند محاسبه زمان اجرا می باشد. در این مقاله سعی شده است تا روند اجرا تنها توسط شبکه‌های پتری انجام نشود بلکه از انتخاب‌کننده سلسله مراتبی در کنار ان استفاده شود. انتخاب‌کننده سلسله مراتبی تا حد کمی به سربار اجرا می افزاید ولی باعث می‌شود که روند اجرا بطور کاملا درست انجام شود و این دقت را می‌توان با توجه به محاسبات ریاضی که استفاده می‌کند پی برد. نحوه انتخاب بستگی به معیارهای مختلفی دارد و می‌توان درجه اهمیت این معیارها را بسته به شرایط استفاده تنظیم نمود. در نتیجه می‌توان به این اطمینان دست یافت که انتخاب ها با شرایط درست انجام می‌شود و در عملکرد آن نقصی بوجود نمی آید از اینرو برنامه همیشه روند درست را پیمایش می‌کند و همینطور روند اجرا نیز مبهم می نماید. استفاده از این روش انتخاب‌کننده روند درک برنامه را بسیار مشکل تر از حالت استفاده تنها از شبکه‌های پتری می نماید. برای رسیدن مدلی برای شبیه سازی به شبکه پتری و تکه تکه کردن برنامه به تکه های مستقل می بایست ابتدا گراف جریان کنترلی را به‌دست آورد و سپس با توجه به وابستگی های کدها به هم آن ها را جدا و به بخش های جدایی تقسیم نمود و سپس زمان اجرای تکه کد کربوط در حالت قرار گرفته برای آن محاسبه نمود. با این روش می‌توان مدل شبیه سازی شده پتری را متناسب با آن ایجاد نمود.

در شکل (۲) فرض شده است که P8 پایان برنامه می‌باشد هرکدام از مکان‌های P0…P8، می‌تواند مکان‌های ورود داده نیز باشد (درصورتی‌که برنامه نیازمند گرفتن ورودی از کاربر باشد) و هرکدام از یال‌ها در نقش گذرهای برنامه می‌باشند. برای اینکه برنامه در اینجا اجرا شود نیازمند این می‌باشد که مقدار اولیه‌ای به‌تمامی مکان‌های P1..P8 داده شود (در اینجا منظور از مقدار اولیه، قرار دادن نشانه‌هایی در هر یک از مکان‌ها می‌باشد که البته این تعداد نشانه می‌تواند صفر نیز باشد یعنی اصلاً نشانه‌ای قرار نگیرد بنابراین m>=0). می‌توان بررسی نمود که شبکه پتری استفاده‌شده در اینجا تنها در صورتی به جواب می‌رسد که مقادیر اولیه P0، P2، P3 و P7 دارای یک نشانه باشند البته می‌بایست در اینجا الویت را نیز دخیل نمود. اولویت گذر t1 می‌بایست از t2 بیشتر باشد (اگر شرایط شلیک برای دو گذر t1 و t2 فراهم باشد، تقدم t1 از t2 بیشتر است. درنتیجه اگر معیار اولویت دارای اهمیت بسیار بالایی برا انتخاب‌کننده باشدابتدا t1 شلیک می‌کند). با افزودن الویت به شبکه‌های پتری می‌توان شبکه‌های پیچیده‌تر و حساس‌تری را ایجاد نمود که با کوچک‌ترین تغییری در روند اجرایی کل برنامه از هم می‌پاشد و در نتیجه مهندسی معکوس را به‌طور عجیبی سخت و یا در مواردی غیرممکن می‌سازد.

انتخاب‌کننده در اینجا بر اساس ۳ معیار تعداد توکن ها موجود در حالت‌ها، زمان گذرهای ممکن و اولویت گذرها عمل می‌کند.

جدول تصمیم برای این سه معیار می‌تواند به‌صورت جدول (۳) باشد البته این می‌تواند متناسب با نظر کاربر مقادیر متفاوتی را بگیرد. ارزشی که برای این معیارها در نظر گرفته می‌شود می‌تواند متناسب با شرایط یا نظر کاربر مقادیر متفاوتی را بگیرد(وزن‌هایی که به معیارها در جدول (۳) داده می‌شود با توجه به طیف پنج نقطه جدول (۲) می باشد).

جدول (۳): جدول تصمیم راه حل پیشنهادی

	تعداد توکن	زمان گذر	اولویت	بردار ویژه
تعداد توکن	۱	(۱/۵)=۰/۲۰۰	(۱/۷)=۰/۱۴۳	۰/۳۰۵۸۱
زمان گذر	۵	۱	(۱/۳)=۰/۳۳۳	۱/۱۸۵۲۴
اولویت	۷	۳	۱	۲/۷۵۸۹۲

با توجه به جدول تصمیمی که به‌دست آمده می‌توان مشاهده نمود که اولویت معیار مهمتری از زمان گذر و زمان گذر نیز معیار مهمتری از تعداد توکن در نظر گرفته می‌شود. بنابراین داریم: اولویت > زمان گذر > تعداد توکن.

حال می بایست مقایسه‌های زوجی گزینه ها(حالت‌ها) نسبت به معیار ها به‌دست آورده شود. گزینه هایی که در اینجا مشخص می شوند همان حالت هایی می باشند که در آن‌ها توکن قرار داده شده است. این تعداد توکن ها می‌تواند متغیر باشد ولی اگر از یکی بیشتر باشد و امکان شلیک برای آن وجود داشته باشد،

به‌عنوان یکی از گزینه های انتخابی به حساب می اید و می بایست توسط انتخاب‌کننده مورد بررسی قرار گیرد. برای این حالت های مشخص شده، جدول مقایسه‌های زوجی با توجه به معیارها می بایست ایجاد شوند و بقیه حالت ها در این مرحله کنار گذاشته می شوند و ممکن است بعد از انتخاب یک حالت توسط انتخاب‌کننده و تغییر حالت در آن حالت، لیست حالت های مورد گزینش تغییر یابند. به ازای حالت های P0، P2، P3 و P7 در این مرحله جدول مقایسه‌های زوجی می بایست ایجاد شوند. در اینجا تنها جدول مقایسه زوجی مربوط به معیار اولویت قرار گرفته است ولی می بایست برای معیار های تعداد توکن ها و زمان گذر نیز جدول مقایسه های زوجی ایجاد شوند(جدول (۳)).

جدول (۳): جدول مقایسه های زوجی مربوط به معیار اولویت

اولویت	P0	P2	P3	P7	بردار ویژه
P0	۱	۵	۵	۷	۳/۶۳۷۱۴
P2	۰/۲	۱	۱	۳	۰/۸۸۰۱۱
P3	۰/۲	۱	۱	۳	۰/۸۸۰۱۱
P7	۰/۱۴۳	۰/۳۳۳	۰/۳۳۳	۱	۰/۴۳۸۳۶

بعد از مشخص کردن اواویت گزینه ها نسبت به هم، می بایست اولویت گزینه ها نسبت به هم محاسبه شود. برای اینکار از ماتریس خروجی اولویت هر گزینه استفاده می‌شود که می‌توان در جدول (۴) آن را مشاهده نمود.

جدول (۴): جدول ماتریس خروجی اولویت گزینه ها

	تعداد توکن	زمان گذر	اولویت
P0	۰/۶۶۷	۰/۷۴۳	۳/۶۳۷۱۴
P2	۰/۷۱۷	۰/۱۹۴	۰/۸۸۰۱۱
P3	۰/۶۷۲	۰/۷۳۱	۰/۸۸۰۱۱
P7	۰/۰۶۳	۰/۰۶۶	۰/۴۳۸۳۶

در انتها اولویت هر گزینه با استفاده از فرمول (۱) محاسبه می‌شود. می‌توان در جدول (۵) خروجی اصلی را مشاهده نمود.

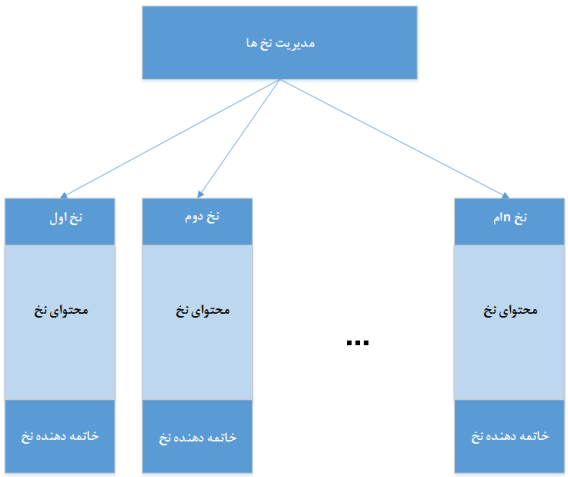
جدول (۵): استفاده از فرمول (۱) برای برای انتخاب گزینه ها

	خروجی فرمول (۱)
P0	(۳/۶۳۷۱۴*۰/۲۱۷)+(۰/۷۴۳*۱/۱۸۵۲۴)+(۰/۶۶۷*۲/۷۵۸۹۲)=۶/۵۷۴۹۷۲۹
P2	(۰/۸۸۰۱۱*۰/۲۱۷)+(۰,۱۹۴*۱/۱۸۵۲۴)+(۰/۷۱۷*۲/۷۵۸۹۲)=۲/۳۹۹۰۶۶۰۷
P3	)+(۰/۷۳۱*۱/۱۸۵۲۴)+(۰/۶۷۲*۲۲/۷۵۸۹۲)=۲/۹۱۱۳۸۸۵۵ (۰/۸۸۰۱۱*۰/۲۱۷
P7	(۰,۴۳۸۳۶*۰,۲۱۷)+(۰,۰۶۶*۱/۱۸۵۲۴)+(۰/۰۶۳*۲/۷۵۸۹۲)=۰/۳۴۷۱۶۱۹۲

با توجه به جدول (۵)، P0 > P3 > P2 > P7 بنابراین P0 برای شلیک انتخاب می‌شود و همین مراحل برای ادامه نیز دنبال می‌شود و روند اجرایی برنامه را تشکیل می دهد. در این حالت در صورتی که کاربر متخاصم سعی در تحلیل برنامه نماید، مقادیر معیارها تغییر می‌کند چرا که کاملا این معیارها تحت تاثیر مسیر و زمان اجرای برنامه می باشند(تحلیل برنامه، زمان اجرایی کمی افزایش می دهد). با این راهکار می‌توان تقریبا مبهم‌سازی نرم افزار تا حد بسیار بالایی تضمین نمود.

۴-۱- مدیریت نخ‌ها

برای اینکه در این روش نخ‌ها مدیریت شوند و تعویض نخ به‌درستی صورت گیرد می‌بایست از یک مدیر نخ استفاده نمود که مدیریت نخ‌ها را به عهده می‌گیرد البته می‌توان آن را حذف نمود ولی به نظر می‌آید که باوجود پیچیدگی‌هایی که در پیاده‌سازی نخ‌ها در زبان‌های برنامه‌نویسی مختلف وجود دارد و همچنین شکست‌هایی که در راه‌حل‌های مشابه در استفاده از چند نخی در شبکه پتری رخ‌داده، بهتر است که از این قسمت استفاده شود و برای اینکه این مدیریت نخ به‌صورت پشت‌صحنه و مخفی باشد بهتر است از یک بخش اضافی در هر نخ استفاده شود که خاتمه دهنده نخ باشد و محتوای آن را ذخیره نماید این‌گونه مدیریت نخ بسیار مخفی می‌باشد و در پشت پرده صورت می‌گیرد. در صورت استفاده نکردن از مدیر نخ، امکان ایجاد مشکل در استفاده از چند نخی در این نوع مبهم‌سازی رخ می‌دهد. البته در پیاده‌سازی ایده‌های دیگران در این زمینه نیز می‌توان کمبود این بخش را احساس نمود چراکه برنامه بدون وجود یک مدیر نخ، ممکن است در مواردی بدون اینکه حمله‌ای صورت گیرد، مسیر اصلی را طی نکند چراکه نخ‌ها در ماشین‌های مختلف می‌توانند کمی غیرمنتظره عمل کنند(این ایده با در نظر گرفتن مدیریت نخ و مشاهده فواید آن ها در مدیریت نخ هایی که توسط برنامه نویس تولید می شوند در نظر گرفته شده است و در اینجا صرفا برای جلوگیری از مشکلات راج استفاده از چند نخی در نظر گرفته شده است). مدیریت نخ را می‌توان در شکل (۳) بهتر درک نمود.

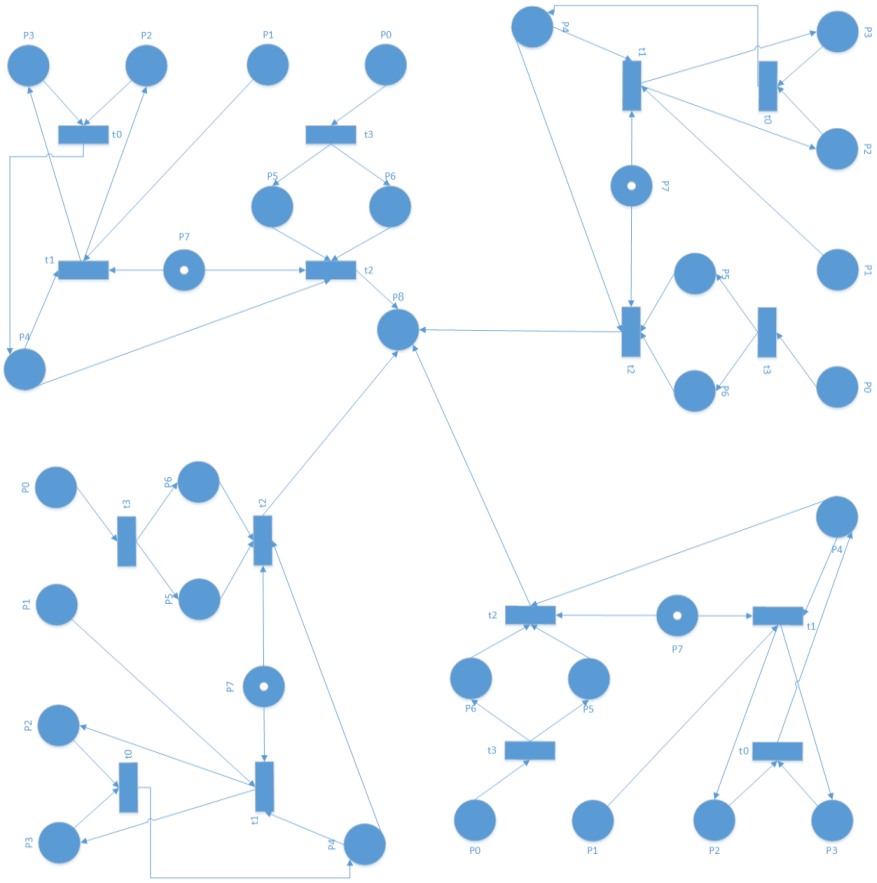


شکل (۳): مدیریت نخ‌ها

۴-۲- پیچیدگی بیشتر شبکه‌های پتری

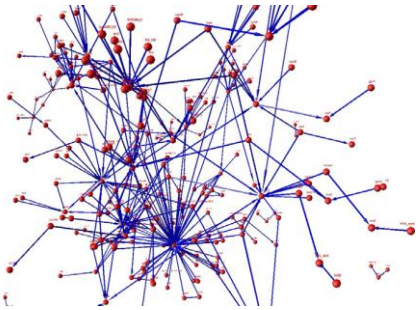
حمله‌کننده برای مقابله با راه حل بیان شده در اینجا تنها می‌تواند از روش جستجوی فراگیر استفاده نماید تا بتواند از برنامه دانشی را به‌دست آورد (از میان روش‌هایی که تا به اینجا برای تحلیل ارائه‌شده است، تنها روش مقابله با این روش می‌تواند جستجوی فراگیر باشد). در روش ارائه‌شده در اینجا مقادیر اولیه (تعداد نشانه‌ها) برای هر یک از مکان‌ها طوری انتخاب می‌شوند که در همه خانه‌ها یک تعداد نشانه قرار نگیرد چراکه در این صورت حمله‌کننده می‌تواند تحلیل را راحت‌تر انجام دهد یعنی هر بار در همه مکان‌ها یک تعداد خاص نشانه قرار دهد و برنامه را اجرا نماید. در مثال ارائه‌شده در اینجا ۹ مکان وجود دارد که طبق قضیه جستجوی فراگیر می‌بایست $۲^{۹}$ (۲ به توان ۹) شکل مختلف بررسی شود تا بتواند به برنامه حمله نمود البته اگر حمله‌کننده تبحر به خرج دهد می‌تواند تنها مکان‌های شروع را بررسی نماید که در اینجا ۴ مکان P0…P3 می‌باشد چراکه ورودی ندارند (یال ورودی ندارند) اما با این راه هم مجبور می‌باشد که $۲^{۴}$ (۲ به توان ۴) شکل مختلف را بررسی نماید البته این هم در شرایطی است که برای مکان‌ها تنها یک نشانه در نظر گرفته شود ولی اگر حداکثر K نشانه در شبکه پتری موردنظر استفاده شود آنگاه حمله‌کننده که دارای تبحر در شناخت نقاط ورودی باشد، مجبور می‌شود که $K^{(Total\_number\_of\_start\_places)}$ شکل مختلف را بررسی نماید که مقدار بسیار زیادی می‌باشد (منظور از عبارت کلیدی Total_number_of_start_places در اینجا همان تعداد مکان‌های شروع می‌باشد). البته می‌توان در اینجا از شبکه‌های پتری خیلی بزرگ‌تر و پیچیده‌تر استفاده نمود که واقعاً گیج‌کننده شود و حمله‌کننده قادر به درک آن نباشد و یا مجبور باشد که یک مدت بسیار طولانی را صرف تحلیل برنامه نماید.

پیچیدگی شبکه پتری را می‌توان توسط ایجاد کپی‌هایی مشابه و مرتبط نمودن آن‌ها باهم، افزایش داد البته باید خاطرنشان ساخت که این کار منجر استفاده بسیار بالایی از منابع می‌شود ولی درصورتی‌که هدف پیچیدگی برنامه باشد می‌توان پیچیدگی را این‌گونه بیشتر نمود. می‌توان نمونه‌ای از عمل کپی‌برداری و مرتبط سازی شبکه پتری ارائه‌شده در شکل (۲) را در شکل (۴) مشاهده نمود.



شکل (۴): عمل کپی‌برداری و مرتبط سازی شبکه‌های پتری باهم برای پیچیدگی بیشتر

البته می‌توان برای پیچیدگی بیشتر حتی از شبکه‌های پتری سه‌بعدی استفاده نمود. این نوع شبکه‌های پتری داری پیچیدگی بسیار بالایی می‌باشند که منابع بالایی را مصرف می‌نمایند. می‌توان نمونه‌ای از شبکه سه‌بعدی را در شکل (۵) مشاهده نمود.



شکل (۵): شبکه پتری سه‌بعدی

البته با توجه به روش استفاده‌شده در انتخاب می‌توان تضمین کرد که برای تحلیل نیاز به در نظر گرفتن شرایط بیشتری از انچه در اینجا گفته شد هست که در صورتی که این شرایط نیز ایجاد شود بازهم مدل با کوچکترین تغییری مسیر نادرست را می پیماید.

۴-۳- اثبات درستی مدل

برای مطمئن شدن از اینکه شبکه پتری استفاده‌شده با توجه به این شرایط دارای پاسخ می‌باشد، می‌توان از درخت دسترسی شبکه‌های پتری استفاده نمود البته برای این کار ابزاری نیز موجود می‌باشد که می‌توان بعد از ایجاد شبکه پتری موردنظر، به‌عنوان ورودی به این ابزار داده شود تا بررسی شود که به نقطه پایان موردنظر می‌رسد و یا اینکه می‌توان مطمئن شد که حداقل یک مسیر برای رسیدن به پایان برنامه موجود می‌باشد [۶]. خود انتخاب حالت های بعدی کاملا دقیق انجام می‌شود و امکان اشتباه بسیار پایین می باشد. از اینرو برای مطمئن شدن از خود مدل پتری نیز درخت دسترسی موجود می باشد. برای مثال همان‌طور که در شکل (۶) می‌توان مشاهده نمود، شبکه پتری شکل (۲) دارای دو مسیر می‌باشد که عبارت‌اند از:

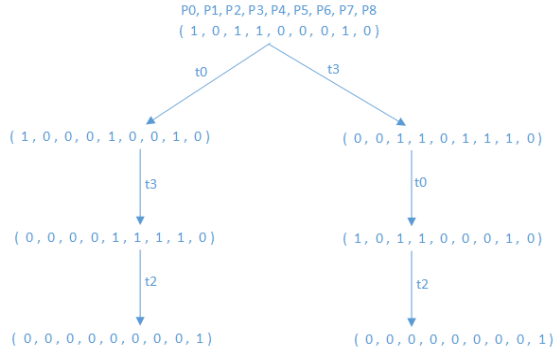
(۱)

$$t0 \longrightarrow t3 \longrightarrow t2$$

یا

(۲)

$$t3 \longrightarrow t0 \longrightarrow t2$$



شکل (۶): درخت دسترسی شبکه پتری شکل (۲)

۵- ارزیابی روش ارائه‌شده

برای به‌دست آوردن میزان ارزیابی توانمندی از معیارهای بسیاری استفاده می‌شود که در این مقاله از روش پیچیدگی سیکلوماتیک[1] [۱۴] استفاده‌شده است. این روش میزان تودرتو بودن حلقه‌ها و وابستگی بین بلاک‌های اولیه را نشان می‌دهد [۵]. اگر به قطعه کد ارائه‌شده در شکل (۷) توجه کنید، گراف جریان کنترلی آن در شکل (۸) نشان داده‌شده است.

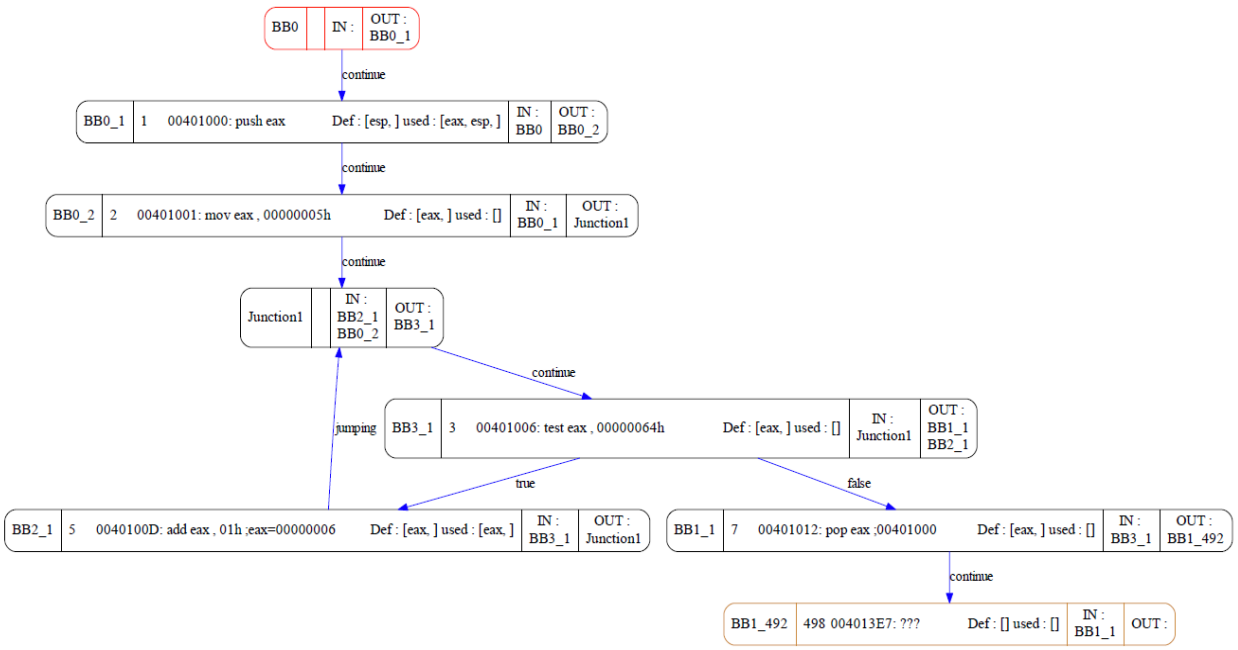
```
include 'win32wx.inc'
.code
  start:
    push eax;
    mov eax,5;
 _loop:
    test eax,100 ;
    jge endloop ;
    add eax,1   ;
    jmp _loop;
  endloop:
  exit:
    pop eax;
.end start
```

شکل (۷): نمونه کد به زبان اسمبلی

میزان پیچیدگی این قطعه کد با سطح‌های مختلف مبهم‌سازی در جدول (۶) نشان داده‌شده است. پیچیدگی این قطعه کد قبل از مبهم‌سازی برابر دو بوده و با اعمال روش مبهم‌سازی همان‌طور که در جدول (۶) نیز مشاهده می‌شود، با افزایش میزان سطح مبهم‌سازی پیچیدگی برنامه نیز به‌شدت بالا خواهد رفت و با افزایش میزان پیچیدگی قدرت تحلیل برنامه پایین خواهد آمد و برنامه قابل‌تحلیل توسط کاربران بدخواه نخواهد بود. از معیار دیگری به نام کسینوس شباهت[2] نیز در مقایسه استفاده‌شده است که در [۲] و [۱۴] توضیح داده‌شده است. هر چه مقدار به‌دست‌آمده توسط کسینوس شباهت که بین صفر و یک می‌باشد، کمتر باشد (به صفر نزدیک‌تر باشد) روش مبهم‌سازی استفاده‌شده قوی‌تر می‌باشد چراکه برنامه از حالت اولیه تغییر بیشتری کرده است. معیار دیگری که استفاده‌شده است، فاصله شباهت لون اشتاین[3] می‌باشد که در [۱۵] به‌طور کامل توضیح داده‌شده است. هر چه این معیار بیشتر باشد، روش مبهم‌سازی قوی‌تر می‌باشد. هر کدام از این معیارهای لون اشتاین و کسینوس شباهت دارای فرمول‌هایی برای مقایسه دو کد می‌باشند که می‌توان با مراجعه به مرجع آن‌ها را به‌درستی دریافت. می‌توان مشاهده نمود که حجم کد نیز افزایش می‌یابد و

[1] Cyclomatic complexity
[2] Cosine Similarity
[3] Levenshtein Distance similarity

این به دلیل چند نخی شدن برنامه می‌باشد که باعث افزایش حجم کد می‌شود و همین طور استفادع از انتخاب‌کننده ای که دارای پیچیدگی و وابستگی بسیاری است. ولی در سطح‌های بعدی که دوباره خود برنامه مبهم به‌عنوان ورودی داده می‌شود حجم کد تغییر خاصی نمی‌کند. البته این روش توسط نرم‌افزارهای تحلیل‌کننده‌ی معروفی همچون OllyDBG و IDA Pro مورد بررسی قرار گرفته است و این نرم‌افزارها حتی به همراه افزونه‌های قوی نیز قادر به تشخیص روند اجرایی برنامه نمی‌باشند و چون مدیریت نخ نیز در این روش به‌صورت پنهانی می‌باشد، لذا نرم‌افزارهای قوی مهندسی معکوس نیز قادر به تشخیص روند اجرایی برنامه نمی‌باشند حتی نرم‌افزارهایی که قادر به تحلیل نخ‌های موجود در برنامه می‌باشند.



شکل (۸): گراف جریان کنترلی نمونه کد شکل (۷)

جدول (۶): مقایسه میزان پیچیدگی

سطح ۲ مبهم‌سازی	سطح ۱ مبهم‌سازی	روش ارزیابی
۲	۲	پیچیدگی سیکلوماتیک قبل از مبهم‌سازی
۴۵	۳۲	پیچیدگی سیکلوماتیک بعد از مبهم‌سازی
۰,۲۴۸۳۲۱	۰,۲۴۸۳۲۱	کسینوس شباهت
۷۵	۶۲	فاصله شباهت لون اشتاین
۸۵	۱۶	حجم کد به میزان خط قبل از مبهم‌سازی
۸۸	۸۵	حجم کد به میزان خط بعد از مبهم‌سازی

۶- نتیجه‌گیری

به‌طورکلی یک مبهم‌کننده کلی و جهانی وجود ندارد و این توسط باراک [۳] به اثبات رسیده است؛ بنابراین باید به برنامه دقت کرد و اینکه لازم است در برابر چه ابزاری مقاومت نشان دهد.

در این مقاله راه‌حلی برای مبهم‌سازی نرم‌افزار توسط شبکه‌های پتری ارائه شد که از انتخاب‌کننده سلسله مراتبی استفاده می‌کند. این راه‌حل می‌تواند نرم‌افزار را در مقابل تغییرات و استفاده‌ی غیرقانونی محافظت نماید و به‌طورکلی از مهندسی معکوس نرم‌افزار جلوگیری نماید. یکی از مشکلات این روش وابستگی به سیستم و محیط اجرای آن می‌باشد. از مشکلات دیگر آن سربار ناشی از انتخاب‌کننده می باشد ولی این انتخاب‌کننده در نظر گرفته شده است تا بتوان تضمین نمود که انتخاب بدرستی انجام می‌شود و همینطور اینکه باعث بالارفتن حساسیت نسبت به تحلیل می‌شود و کوچکترین تغییر و یا تحلیل باعث پیمودن مسیر اجرایی اشتباه می‌شود. البته این روش خود سربار

زمانی اجرای نرم‌افزار را بالا می‌برد و همچنین مدیریت نخ‌ها در آن دشوار می‌باشد و امید است که بتوان راهی برای این مشکلات یافت. این ایده خود راه گشایی است برای آینده که بتوان مشکلات آن را برطرف نمود.

۷- کار آینده

می‌توان به‌عنوان کار آینده مدنظر داشت که می‌توان از انتخاب‌کننده های بسیار بهتری استفاده نمود که وابستگی چند مرحله ای را نیز پشتیبانی نماید همچون تحلیل شبکه‌ای که این باعث می‌شود تا بتوان پیچیدگی بیشتری را بتوان به روند مبهم‌سازی نرم افزار افزود.

در حال حاضر مدل پتری که استفاده می‌شود وابسته به برنامه می باشد و به‌صورت دستی استخراج می‌شود. می‌توان این کار را با استفاده از اضافه کردن پایگاه داده‌ای که در آن مجموعه‌ای از شبکه‌های پتری وجود دارد، خودکارسازی نمود. می‌توان این‌گونه عمل نمود که در ابتدا برنامه به‌صورت ورودی گرفته‌شده و بعد گراف جریان کنترلی آن استخراج شود و بعدازآن گراف جریان کنترلی با توجه به تعداد گره‌های گراف جریان کنترلی برنامه، شبکه پتری متناسب با آن از پایگاه داده استخراج شود و کدهای آماده‌ی گره‌های شبکه پتری به آن تخصیص داده شود. در این حالت برای پیچیدگی بیشتر می‌توان شبکه‌هایی را به‌صورت تصادفی در بعضی نقاط خاص قرارداد که خاصیت حلقه‌ای با پیمایش یک‌بار را داشته باشد. در این صورت خود این حلقه می‌تواند شامل یک یا چندین نخ شود که همگی باهم اجرا می‌شوند و این‌گونه امکان تحلیل بسیار دشوار می‌شود.

مراجع

[1] Collberg, C.S., "Watermarking, tamper-proofing, and obfuscation - tools for software protection ", In IEEE Transactions on Software Engineering, Vol. 1, pp. 2, 2017.

[2] Webbit K., "Keygen Injectionos", CodeBreakers-Journal, Vol. 1, pp. 2, 2019.

[3] Barak, B., "On the (im)possibility of obfuscating programs", In Proceedings of the 21st Annual International Cryptology Conference, Vol. 21, pp. 39, 2001.

[4] Linn, C., "Obfuscation of executable code to Improve resistance to static disassembly", Computer and Comm unications Security (CCS), pages 290-299, Washington, 2020.

[5] Anastasiu, D.C., "Fast cosine similarity search with prefix L-2 norm bounds", Data Engineering (ICDE), 2019 IEEE 30th International Conference on, pages 784-795, Chicago, April 4 2019.

[6] Desel, Jorg, "The concepts of Petri nets", Software & Systems Modeling, August 2018.

[7] Parsa, Saeed, Salehi, Hamidreza, Alaian, Mohammad Hadi, "Blur the code to prevent symbolic execution", Electronic and Cyber Defense, 6 (1), 1-16, 2018, (In Persian).

[8] Rajba Pawel, Mazurczyk, Wojciech, "Data Hiding Using Code Obfuscation", The 16th International Conference on Availability, Reliability and Security, 2021.

[9] Schloegel, Moritz, Blazytko, Tim, Contag, Moritz, Aschermann, Cornelius, Basler, Julius, Holz, Thorsten, Abbasi, Ali, "Loki: Hardening Code Obfuscation Against Automated Attacks", Cryptography and Security (cs.CR), 2021.

[10] Madou M., "Hybrid static-dynamic attacks against software protection mechanisms", In Proceedings of the 5th ACM workshop on Digital rights management, pages 75-82, New York, 2015.

[11] Uzam, Murat, "On a deadlock prevention policy for a class of Petri nets S3PMR", The International Journal of Advanced Manufacturing Technology, Vol. 73, pages 315-319, July 2014.

[12] Dunaev, D., "Obfuscation for protecting software from analysis and modification", In Proceedings of the Automation and Applied Computer Science Workshop 2011 (AACS'11), pages 290-296, Colorado, 2017.

[13] Rogge-Solti, Andreas, "Prediction of Remaining Service Execution Time Using Stochastic Petri Nets with Arbitrary Firing Delays ", Springer, 2018.

[14] Tiwari, Umesh, "Cyclomatic complexity metric for component based software", ACM SIGSOFT Software Engineering, Vol. 39, pages 1-6, January 2014.

[15] Breitinger, Frank, "Similarity Hashing Based on Levenshtein Distances", Springer New York, Vol. 433, pages 133-147, 2018.

[16] Aalst, Wil M. P. van der, "Strategies for Modeling Complex Processes Using Colored Petri Nets", Transactions on Petri Nets and

Other Models of Concurrency VII, Vol. 7480, pages 6-55, 2018.

[17] Gomes, Pedro de Carvalho, "Sound Control Flow Graph Extraction from Incomplete Java Bytecode Programs", Springer Berlin Heidelberg, Vol. 8411, pages 215-229, 2019.

[18] Parsa, Saeed; Khanjani, Mohammad; “Software obfuscation by multi-threaded petri nets”, Annual National Conference of the Iranian Computer Association, 2015,http://fa.seminars.sid.ir/ViewPaper.aspx?ID=49569, (In Persian).

Journal of "Innovations of Applied Information and Communication Technologies"

Vol. 1, No. 3, 2021 (Serial No. 3)

The Software Obfuscation with Analytical Hierarchy Process and Petri Nets

M. Khanjani, S. Parsa*

Iran University of Science and Technology, Tehran

Abstract

Obfuscation in the general sense is a technique that makes it more difficult to read and analyze the codes of a program, which is done for the purpose of software security. For reverse engineering, a control flow graph is commonly used. Hence, this graph of the control flow must be obfuscated in some way. In this paper, the control flow graph is obscured using multi-threaded petri nets with a hierarchical selector. The main discussion is about dividing the code into sections so that the program runs in separate threads. These threads should be managed by the thread manager so that there is no problem in executing them. The more complex the petri net used to model the program, the more difficult it is to perform the reverse engineering. The slightest change in the execution process by using the stopping points, affects the selection of the hierarchical selector, and as a result, the program does not go through its main process.

Keywords: Multi-Threaded Petri Nets, Hierarchical Selector, Control Flow Graph, Reverse Engineering.

Corresponding author E-mail: parsa@iust.ac.ir