همایش پژوهش های نوین در علوم و فناوری
Conference on Recent Research in Science and Technology

# تست امنیت نرم افزار کاربردی براساس SPN

**مهران آقائی**[1]

[1]عضو هیات علمی دانشگاه پیام نور

m.aghaei@pnu.ac.ir

## چکیده

امنیت یکی از ویژگی‌های مهم نرم افزار است. برخی روش‌های تست امنیت نرم افزار، مبتنی بر پارامترهای دقیق یا بر مبنای ساختار تابع هستند و نمی‌توانند با محیط پیچیده و نیازهای ایمنی سازگار شوند، به ویژه در محیط‌های توزیع شده و بزرگ. در این مقاله براساس تئوری پتری نت تصادفی (SPN)[1]، و با استفاده از پارامترهای اندازه‌گیری قابلیت اعتماد SPN، امنیت نرم افزار کاربردی را مورد تحلیل قرار می‌دهیم و سپس آنالیز و طراحی سیستم نرم افزاری را ارائه می‌دهیم تا یک روش تست امنیت نرم افزار براساس مدل SPN مورد بررسی قرار گیرد. با انجام آنالیز نمونه، مشخص است که این روش می‌تواند به صورت طبیعی، سریع و کارآمد، وضعیت سیستم غیرفعال یا ترکیب حالت‌ها را بیابد، در حالی‌که یک روش جدید برای تست امنیت نرم افزار کاربردی ارائه می‌دهد.

**کلمات کلیدی:** تئوری پتری نت تصادفی، تست قابلیت اعتماد نرم افزار، مدل زنجیره مارکوف، آنالیز امنیت نرم افزار

## 1. مقدمه

تولید و توسعه برنامه‌ها در سیستم‌های هوشمند پیچیده همچون رایانش ابری، ابرداده‌ها و اینترنت اشیاء نشان می‌دهد که چالش‌های زیادی در سبک زندگی انسان‌ها وجود دارد و تمایل بشر مدرن به علم و فناوری، به ویژه اتکا به فناوری اطلاعات به عنوان بخش مهم زندگی را نمایش می‌دهد. همچنن در بخش محصولات سخت افزاری نیاز است که قابلیت اعتماد برای فعالیت‌های انسانی و تحقیقات علمی فراهم شود. در ضمن، نیازهای امنیتی نرم افزار نیز بهبود یابد. تست امنیت نرم افزار، پروسه‌ای برای بررسی نیازهای عملکردی مورد انتظار و نیازهای امنیتی است. همچنین مشکلات امنیتی بالقوه در نرم افزار نیز مشخص می‌شود و عموما شامل احراز هویت کاربر، تست امنیت پایگاه داده و شبکه است. راهکارهای تست به دلیل نیازهای امنیتی مختلف، متفاوت هستند. در حال حاضر، روش‌های تست امنیت نرم افزار عموما شامل تست آنالیز ایستا، تست مدگرا، تست فازی، تست مبتنی بر انتشار خطا، تست مبتنی بر درخت خطا، تست نفوذ و غیره هستند.

چن و همکاران معتقدند که تست نرم افزار با انتشار خطا، سبب سربار اضافی نرم افزار می‌شود و بر کارایی آن تاثیرگذار است و خطای متناوب یا خطای نامعلوم تولید می‌کند[1]. زو و همکاران از فناوری درخت خطای رسمی برای بررسی نیازهای امنیت نرم افزار استفاده کردند و پیشنهاد کردند که از زیرمجموعه مینیمم رخدادهای امنیتی برای تولید مجموعه

[1] Stochastic Petri net

تست استفاده شود[2]. یانگ و همکاران مسیر رفتار تهدیدآمیز را براساس مدل تهدید امنیت مشخصی ایجاد کردند و پیشنهاد کردند که براساس راهکار تست خاص، حالت‌های تست تولید شود. از این روش برای تست رفتار برنامه در محدوده کتابخانه مدل و تهدیدات امنیتی، استفاده می‌شود. هرچند این روش‌ها، برخی مبانی تئوری را ارائه می‌دهند، اما به سادگی مشخص است که مبتنی بر برخی پارامترهای مشخص هستند و نمی‌توان به صورت تصادفی از آنها در برنامه‌های عملی نرم افزاری استفاده کرد[3].

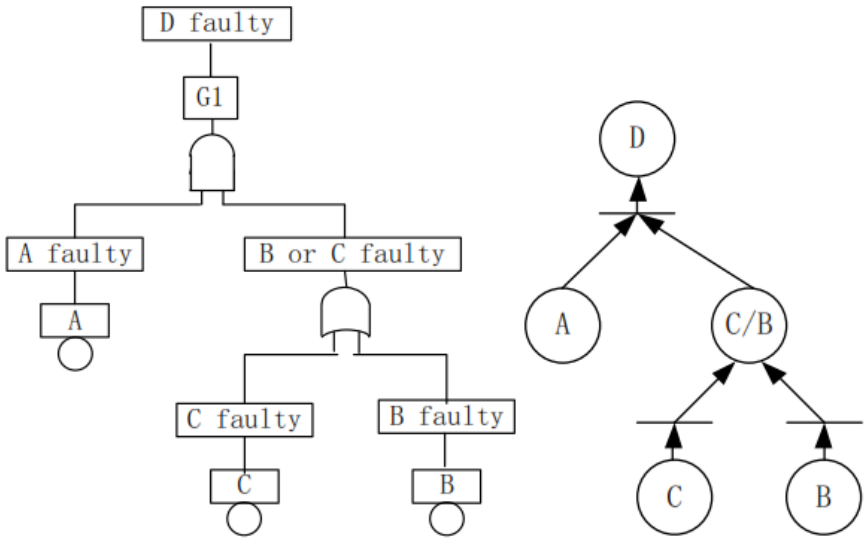
به دلیل اینکه ارزیابی ایمنی نرم افزار، چندین عامل را در نظر می‌گیرد، بنابراین عامل‌ها به صورت تصادفی انتخاب می شوند. زو و همکاران اثبات کردند که با استفاده از پتری نت (PN) نه تنها می‌توان معماری نرم افزار را تشریح نمود، بلکه همچنین می‌توان پروسه کاری نرم افزار و روابط بین ماژول‌های نرم افزاری را نیز مشخص کرد. وی و همکاران اثبات کردند که با استفاده از روش PN در آنالیز امنیتی نرم افزار می‌توان به سرعت، زیرمجموعه مینیمم را برای مجموعه تست تولید کرد. بنابراین در این مقاله، از تئوری SPN به عنوان مبنایی برای آنالیز تست امنیت نرم افزار برنامه (از نظر خطاهای امنیتی نرم افزار) استفاده می‌شود. نتایج شبیه‌سازی نشان می‌دهد که این روش می‌تواند در محیط های مختلف، سازگاری را حفظ کند و ارزیابی مشخصه‌های آن، دقت بالا و هزینه اندکی دارد. همچنین یک روش موثر و قابل اعتماد برای ارزیابی امنیت نرم افزار ارائه می‌دهد.

## 2 .مدلسازی قابلیت اعتماد نرم افزار و اندازه گیری پارامترها براساس SPN

در یک سیستم نرم افزاری، ابتدا تست قابلیت اعتماد نرم افزار صورت می‌گیرد، یا قابلیت اعتماد در پروسه کنترل ارزیابی می‌شود. زیرا محیط و انتخاب مسیر برنامه در زمان اجرا به صورت تصادفی هستند. بنابراین خطای نرم افزار تصادفی است و اجرای ماژول‌های سیستم به صورت تصادفی انجام می‌شود[4].

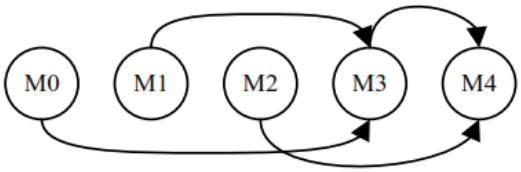
در هنگام ایجاد ماژول ارزیابی SPN، مدل ارزیابی قابلیت اعتماد معماری نرم افزار براساس SPN برای مولفه‌های نرم افزاری ساخته می‌شود. در شکل 1، یک درخت خطای نرم افزاری ساده و مدل SPN نمایش داده شده است. در این مدل، مکان S بیانگر یک ماژول است و توکن‌ها بیانگر ماژول کنترل و وضعیت کاری هستند و با انتقال توکن، مسیرها و شرایط در زمان اجرا مشخص می‌شود. گذر T فعالیتهای پروسه ماژول در ارزیابی منطقی یا تست را تشریح می‌کند، یا روابط بین ماژول‌ها را بیان می‌کند که سبب تغییر وضعیت ماژول می‌شود. بنابراین گذر فوری، با پیاده‌سازی شرط منطقی در تست قابلیت اعتماد نرم افزار، رابطه مستقیم دارد. کمان F وضعیت محلی و انتقال از وضعیت محلی (به دلیل اجرای عملیات) را نشان می‌دهد.

مدل SPN سیستم نرم افزاری را به صورت همریخت با زنجیره مارکوف زمان پیوسته تعریف می‌کند، همانطور که در شکل 2 نشان داده شده است.




شکل 1: مدل پتری نت و درخت خطای ساده

| | C | B | A | C/B | D |
|---|---|---|---|---|---|
| M0 | 1 | 1 | 1 | 0 | 0 |
| M1 | 0 | 0 | 0 | 1 | 0 |
| M2 | 0 | 0 | 0 | 1 | 0 |
| M3 | 0 | 0 | 0 | 0 | 1 |
| M4 | 0 | 0 | 0 | 0 | 1 |




شکل 2: مدل زنجیره مارکوف شکل 1

بنابراین گذر t یک متغیر تصادفی پیوسته $x_i$ از لحظه‌ی قابل پیاده سازی است تا بتوان این لحظه را پیاده‌سازی کرد و از تابع توزیع زیر استفاده می‌کند:

$$F_i(x) = P(X_i \leq x)$$

از سوی دیگر، تابع توزیعِ مرتبط با هر گذر را می‌توان به صورت یک تابع توزیع نمایی در مدل SPN تعریف کرد:

$$\forall t \in T : F_i \doteq 1 - e^{-\lambda_i x}$$

پارامتر واقعی $\lambda_i > 0$ میانگین نرخ آتش برای گذر t است و متغیر $x \geq 0$ یک متغیر تصادفی برای زمان t است.

فرض کنید S ,.., 2, 1}=n{ یک مجموعه از حالت‌های نرم افزار باشد و ,.., 2, 1}=kW{ حالت‌های کاری را برای ماژول سیستم نشان دهد. F={k, k+1,…2, +n} حالت خطا را برای سیستم نرم افزاری نمایش می‌دهد و t)X( حالت سیستم در زمان t است. t)X( یک پروسه مارکوف همگن است که S را به عنوان فضای حالت درنظر می‌گیرد. فرض

کنید $p_i(t) = P[X(t) = i]$ احتمال حالت i باشد که در زمان $\lambda_i(t)$ در S قرار دارد و $p_{ij}(t) = P[X(t + \Delta t) = j|X(t) = t]$ احتمال شرطی سیستم در حالت t در زمان $t + \Delta t$ است، تحت این شرط که سیستم در زمان t، در حالت j باشد. فرض کنید $P(t) = (p_1(t), p_2(t), \dots p_n(t))$ و $P(t + \Delta t) = P(t)Q(t)$ توزیع احتمال فضای حالت در هنگامی است که سیستم در زمان t قرار دارد و داریم:

$$Q(t) = \begin{pmatrix} p_{1,1}(t) & p_{1,2}(t) & \cdots & p_{1,n}(t) \\ \cdots & \cdots & \cdots & \cdots \\ p_{i,1}(t) & p_{i,2}(t) & \cdots & p_{i,n}(t) \\ \cdots & \cdots & \cdots & \cdots \\ p_{n,1}(t) & p_{n,2}(t) & \cdots & p_{n,n}(t) \end{pmatrix} = [q_{ij}]$$

واضح است برای $\Delta t \to 0$ داریم:

$$\begin{cases} q_{ij} = \lim_{\Delta t \to 0} \dfrac{p_{ij}(t + \Delta t)}{\Delta t} = \lambda_k & i \neq j \\ q_{ii} = \lim_{\Delta t \to 0} \dfrac{1 - p_{ii}(t + \Delta t)}{\Delta t} = -\sum_k \lambda_k & i = j \end{cases} \quad (1)$$

که $k \in W$ تعداد سرویس‌های ارائه شده توسط سیستم را در گذر فعالیت نشان می‌دهد. براساس فرضیه عمومی مدل ارزیابی قابلیت اعتماد نرم افزار، داریم:

تعریف 1: اگر وظایف اجرا شده توسط هر ماژول نرم افزاری در زمان خاص، بتواند توزیع احتمال مورد نظر را برآورده کند، می‌گوییم که نرم افزار در تمامی زمان‌ها در دسترس است.

لم 1: دسترس‌پذیری سیستم نرم افزاری در زمان t برابر است با مجموع احتمال اجرای نرمال هر ماژول نرم افزاری در طول زمان.

$$A(t) = P[X(t) = j, j \in W] = \sum_{j \in W} p_j(t) \quad (2)$$

اگر $A_s = \lim_{t \to \infty} A(t)$، وضعیت پایدار سیستم در دسترس است.

به دلیل اینکه فضای حالت نرم افزار در هنگام اجرا، پروسه مارکوف همگن است، بنابراین در هنگام $\Delta t \to 0$، با استفاده از فرمول احتمال کل به سادگی می‌توان ثابت کرد که:

$$\begin{cases} P(t) = \dfrac{dP(t)}{dt} Q^{-1} \\ P(0) = (p_1(0), p_2(0), \cdots, p_n(0)) \end{cases} \quad (3)$$

ماتریس Q باید به فرم استاندارد جردن تبدیل شود و در هنگام استفاده از معادله پازلی، با استفاده از تبدیل لاپلاسی حل می‌شود. در SPN زمان پیوسته، به راحتی می‌توان اندازه قابلیت اعتماد نرم افزار را در شرایط $x_i(t)$ به دست آورد:

$$R_i(t) = 1 - F_i(t) = e^{-\lambda_{ii} x_i(t)} \quad (4)$$

اندازه قابلیت اعتماد سیستم را می‌توان به صورت زیر به دست آورد:

$$R(t) = e^{\sum_i [\lambda_i(t) p_i(t)]} \quad (5)$$

بنابراین $\lambda_i(t)$ بیانگر نرخ گذر i امین ماژول در زمان t است و $p_i(t)$ بیانگر احتمال کارآمد حالت کاری i امین ماژول در زمان t است. بنابراین می‌توان نتیجه گرفت که:

اندازه خطا به صورت زیر است:

$$F(t) = 1 - R(t) = 1 - e^{\sum_i [\lambda_i(t) p_i(t)]} \quad (6)$$

شدت خطا برابر است با:

$$f(t) = \frac{dF(t)}{dt} \quad (7)$$

زمان متوسط برای خطای اول (MTTFF) برابر است با:

$$MTTFF = \int_0^t R(t)dt \quad (8)$$

هر مکان در مدل SPN بر دو حالت مبنا دلالت دارد: وضعیت پایدار و حالت خطا. بنابراین احتمال وضعیت پایدار سیستم نرم افزاری در زمان t برابر p است و $S = \{p, 1-p\}$. در صورتی که احتمال وضعیت پایدار هر ماژول در زمان t برابر $p_i(t)$ است و میانگین فاصله خطا در سیستم نرم افزاری به صورت زیر است:

$$MTTBS = (I - P)(-Q)^{-1} \quad (9)$$

از طریق آنالیز اندازه‌گیری شاخص قابلیت اعتماد، می‌توانیم مدل قابلیت اعتماد نرم افزار را براساس SPN آنالیز کنیم و حالت‌های سیستم نرم‌افزاری را بررسی کنیم، در حالی‌که کارایی حالت‌ها در پروسه طراحی نیز چک می‌شود.

### 3. آنالیز امنیت نرم افزار براساس SPN

هدف از استفاده از SPN برای آنالیز امنیت نرم افزار، یافتن دلایل نرم افزاری رخداد خطا است. براساس حالت خطا می توان از تست امنیت نرم افزار و طراحی ایمن استفاده کرد. بنابراین نرم افزار می‌تواند نیازهای امنیتی مورد انتظار را داشته باشد. از طریق گام‌های زیر و با استفاده از SPN می‌توان نرم افزار را تست و آنالیز نمود:

ابتدا لازم است که ریسک نرم افزار بررسی شود تا درخواست‌های امنیتی در توسعه نرم افزار تایید شده و حالت غیرایمن نرم افزار، تعریف شود. در این مرحله، طراحان نرم افزار باید درک خاصی از پروسه طراحی نرم افزار داشته باشند و حالت‌های خطرناک غیرقابل قبول را براساس اهداف طراحی نرم افزار، نیازهای امنیتی و مشخصه‌های طراحی، تعیین کنند.

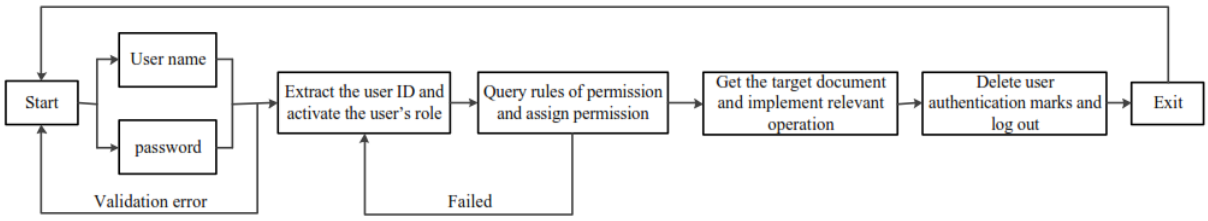
سپس براساس پروسه آنالیز امنیت نرم افزار، می‌توانیم مدل SPN نرم افزار را بسازیم. بنابراین برای ساخت گراف قابلیت اعتماد نرم افزار، مجموعه علائم قابل دسترسی برای هر علامت در گراف دسترسی، لیست می‌شود و هر مجموعه حالت دردسترس مورد آنالیز قرار می‌گیرد و مجموعه‌های از علائم دردسترس که به سادگی سبب ایجاد خطا در سیستم می‌شوند، مشخص می‌شود.

در نهایت، براساس آنالیز وضعیت خطای نرم افزاری، می‌توانیم به مرحله طراحی نرم افزاری برگردیم و سپس از آنالیز قابلیت اعتماد نرم افزاری و تست امنیتی نرم افزار، تست حالت‌های طراحی، قدرت طراحی و تست ماژول‌ها استفاده کنیم تا به سادگی بتوان خطای نرم افزاری را تشخیص داد و از ترکیب حالت‌های خطای نرم افزاری جلوگیری کرد.

از طریق آنالیز مدلسازی نرم افزار قابل اعتماد، می‌توان ترکیب حالت‌های خطای نرم افزاری را مشخص نمود. بنابراین طراحان نرم افزار می‌توانند از ترکیب حالت‌های خطا در پروسه طراحی نرم افزار جلوگیری کنند و ایمنی طراحی نرم افزار را بهبود بخشند.

**4 .مثال**

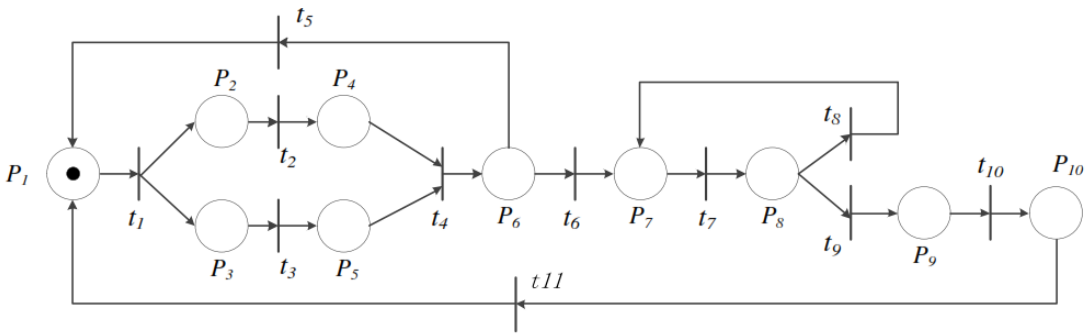
به دلیل اینکه تست امنیت نرم افزار کاربری، عموما شامل تست احراز هویت کاربر، تست شبکه سیستم و تست پایگاه داده است، بنابراین در این مقاله، کنترل دسترسی کاربر به برنامه نرم افزاری به عنوان یک مثال در نظر گرفته می شود تا روش تست امنیت برنامه کاربردی براساس SPN نمایش داده شود. شکل 3 فلوچارت عملیات منطقی ساده را برای احراز هویت امنیتی کاربر در نرم افزار کاربردی نشان می‌دهد.




شکل 3: فلوچارت عملیات منطقی برای احراز هویت امنیتی کاربر

براساس شکل 3، مدل SPN برای عملیات منطقی برای کنترل دسترسی کاربر ایجاد می‌شود، همانطور که در شکل 4 نشان داده شده است. در توصیف SPN در معماری نرم افزار، هر مکان بیانگر یک حالت جزئی از سیستم نرم افزاری است و هر گذر برای سیستم نرم افزاری، از یک حالت به حالت دیگر رخ می‌دهد. در پروسه در حال اجرا برای کل سیستم نرم افزاری، توکن به صورت ثابت انتقال می‌یابد و انتقال، تابعی از یک نرم افزار برای اجرای عملیات مختلف در پروسه در حال اجرا است. پروسه خاص به صورت زیر است: در هنگام استفاده از نرم افزار کاربردی، نام کاربری و گذرواژه توسط کاربر وارد می شود. اگر احراز هویت موفقیت آمیز باشد، سیستم شناسه کاربر را استخراج نموده و نقش کاربر را فعال می‌کند. سپس براساس نقش‌های مجموعه مجوزها، مجوز را تخصیص می‌دهد. کاربر بعد از به دست آوردن مجوز می‌تواند عملیات متناظر را انجام دهد. سیستم، علائم احرازهویت کاربر را پاک کرده و آن را خارج می‌کند و در نهایت به حالت اولیه بازمی‌گردد.

در شکل 4، مکان‌های P1 تا P10 بیانگر حالت نرم افزار، بعد از هر عملیات هستند. گذرهای 2t تا t4 برای ارزیابی نام کاربر و گذرواژه مورد استفاده قرار می‌گیرند. گذرهای t5 تا 11t عملیاتی هستند که به نقش‌ها تخصیص داده می‌شوند و مجوزهای کنترل دسترسی را فراهم می‌کنند. پروسه عملیات مورد استفاده ما، گذرهای میانگین نرخ آتش را در کل سیستم عملیاتی شبیه‌سازی می‌کند. مکان، گذر و وضعیت‌ها در جدول 1 ارائه شده است و مجموعه‌های علائم دسترس پذیر در جدول 2 نشان داده شده‌اند. میانگین نرخ آتش گذر برابر است با: {λ={1, 2, 2, 3, 1, 3, 2, 1, 1, 1, 1. براساس فرمول 3، فرمول احتمال حالت پایدار، احتمال حالت پایدار را برای هر وضعیت محاسبه می‌کنیم: M[P0]=0.1875، M[P1]=M[P2]=M[P3]=0.0468 و M[P4]=0.0625 و M[P5]=0.0468 و M[P6]=M[P7]=M[P8]=M[P9]=0.1406. بنابراین اندازه قابلیت اعتماد برای ماژول کنترل دسترسی برابر است با: R(t)=0.4286.




شکل 4: مدل SPN عملیات منطقی برای کنترل دسترسی کاربر

براساس نتایج محاسبه، قابلیت اعتماد کنترل دسترسی نرم افزار، پایین است. دلیل آن، استفاده از مدل احراز هویت هویت مبنا است و اینکه نقش‌های مجاز سیستم، منطقی نیستند. بنابراین در شروع طراحی نرم افزار، باید روش‌های متنوع احراز هویت، قوانین تقسیم هویت و طراحی و حالت‌های تست طراحی را اضافه کرد تا دو مسیر احتمالی را پوشش دهند و امنیت نرم افزار، بهبود یابد.

جدول 1: مکان، گذر و حالت‌های گراف دسترس‌پذیر

| مکان | گذر | تغییر حالت |
|---|---|---|
| P1 | t1 | ورود کاربر به سیستم نرم افزار با استفاده از t1 |
| P2 | t2 | وارد کردن نام کاربری با استفاده از t2 |
| P3 | t3 | وارد کردن گذرواژه با استفاده از t3 |
| P4 | t4 | تایید صحت نام کاربری و گذرواژه با استفاده از t4 |
| P5 | t5 | خطا در هنگام ورود، بازگشت به حالت اولیه |
| P6 | t6 | استخراج شناسه کاربر و فعال‌سازی نقش متناظر با t6 |
| P7 | t7<br>t8 | نقش‌های مجاز پرس و جو در سیستم نرم افزاری و تقسیم مجوزها بین کاربران<br>در صورتیکه تقسیم مجوزها با خطا همراه باشد، سیستم مجددا شناسه کاربر را برای ارزیابی استخراج می‌کند. |
| P8 | t9 | به دست آوردن مستندات متناظر و انجام برخی عملیات توسط t9 |
| P9 | t10 | حذف علامت احراز هویت کاربر و خروج کاربر توسط t10 |
| P10 | t11 | بازگشت به حالت اولیه |

جدول 2: مجموعه علائم دسترس پذیر سیستم

| NO | Marks |
|---|---|
| $M_0$ | (1,0,0,0,0,0,0,0,0,0) |
| $M_1$ | (0,1,1,0,0,0,0,0,0,0) |
| $M_2$ | (0,0,1,1,0,0,0,0,0,0) |
| $M_3$ | (0,1,0,0,1,0,0,0,0,0) |
| $M_4$ | (0,0,0,1,1,0,0,0,0,0) |
| $M_5$ | (0,0,0,0,0,1,0,0,0,0) |
| $M_6$ | (0,0,0,0,0,0,1,0,0,0) |
| $M_7$ | (0,0,0,0,0,0,0,1,0,0) |
| $M_8$ | (0,0,0,0,0,0,0,0,1,0) |
| $M_9$ | (0,0,0,0,0,0,0,0,0,1) |

براساس آنالیز تئوری، مثال، آنالیز و مقایسه روش‌های موجود برای تست ایمنی، همانطور که در جدول 3 نشان داده شده است، استفاده از SPN برای آنالیز امنیت نرم افزار کاربردی ساده است و به سادگی قابل درک است. همچنین می‌تواند دامنه آنالیز امنیت نرم افزار را محدود نموده و به سادگی، حالت ریسک نرم افزار را مشخص کند.

جدول 3: مقایسه روش‌های عمومی تست امنیتی

| روش تست امنیت | دامنه کاربرد | مزایا و معایب |
|---|---|---|
| محاسبه اشکالات | عموما برای تست عملیات امنیتی نرم افزار مورد استفاده قرار می‌گیرد. | این پروسه عموما تحت تاثیر عوامل مورد نظر است، درخواست‌های زیادی برای تست کننده وجود دارد و عموما مبتنی بر تجربه تست کننده است. |
| مبتنی بر مجموعه قواعد ساختاری یک زبان | عموما در نرم افزارهایی مورد استفاده قرار می گیرد که فاصل آنها، قواعد ساختاری مشخصی دارد. | حوزه کاربرد آن محدود است و برای نرم افزارهایی مناسب است که فاصل آنها، قواعد ساختاری واضحی دارد. |
| مدلگرا | دامنه آن به توانایی مدل‌سازی عملیات امنیتی وابسته است و برای تست عملیات امنیتی برای احراز هویت، کنترل دسترسی و غیره کاربرد دارد. | احتمال خطای انسانی را از بین می‌برد، اما کارایی تست پایین است. مدل تست را به سختی می‌توان ایجاد کرد و به سختی می‌توان سیستم را برای حالت‌های غیرکراندار موجود تست نمود. |
| تئوریگرا | متعلق به یک روش رسمی است و تطبیق‌پذیری زیادی دارد. | دامنه کاربردی آن گسترده است، اما کارایی تست اندک است. پروسه اثبات تئوری زمان‌گیر است و به سختی می‌تواند پروسه تست خودکار را پیاده‌سازی کند. |
| آنالیز درخت خطا (FTA) | برای سیستم‌های مختلط در مقیاس بزرگ مناسب است، همچون سیستم کنترل جنگی، سیستم‌های نظارت طبی، تشخیص حمله در شبکه و غیره. | تحقیق برای مجموعه برشی حداقل، سخت است. تبدیل نیازهای امنیتی به درخت خطای رسمی و آنالیز درخت خطا دشوار است. |

| | | |
|---|---|---|
| مدل SPN | مهمترین ویژگی SPN شامل همگام‌سازی، موازی‌سازی، عدم قطعیت و توانایی تشریح و آنالیز سیستم توزیع شده است. بنابراین می توان از آن در بسیاری از سیستم‌های عملی استفاده کرد. | محدوده گستردهای از کاربردها را شامل می‌شود، توانایی زیادی برای شبیه‌سازی دارد، می‌تواند دامنه آنالیز ریسک را کاهش دهد و حالت‌های خطرناک را مشخص کند. همچنین می‌تواند حالت‌های تست جامعی را بسازد و به سادگی، یک پروسه تست خودکار را به دست آورد. |

## 5 .نتیجه گیری

روش‌های بسیاری برای تست امنیت نرم افزار وجود دارد، اما اغلب آنها دامنه کاربردی خاص و شرایط ویژه‌ای دارند. در یک محیط تصادفی، استفاده از یک روش تست امنیت نرم افزار که سریع و کارآمد باشد، یکی از مسائل مهمی است که باید حل شود. هدف این مقاله، استفاده از SPN برای آنالیز امنیت نرم افزار است تا بتوان ماژول‌ها و حالت‌هایی که منجر به خطای نرم افزار می‌شوند، شناسایی کرد و از آنها برای طراحی بهتر ماژول‌های نرم افزاری در فاز طراحی نرم افزار استفاده کرد، تا از تولید حالت‌های خطا جلوگیری شود. این نوع از روش‌ها، نه تنها به سرعت می‌توانند مکان را مشخص کنند، بلکه همچنین به سادگی می‌توانند حالت‌های خطا را تشخیص دهند و ریسک اجرای برنامه را کاهش دهند. همچنین می‌توان از آن‌ها در پروسه توسعه سریع نرم افزار استفاده کرد، و هزینه توسعه نرم افزار را کاهش داد و روش جدیدی برای تست امنیت نرم افزار ارائه نمود.


## تشکر و قدردانی

تقدیم به مادر عزیزم که هر چه دارم از اوست.


## مراجع


[1] Chen J F, Lu Y S, Xie X D. Research on Software Fault Injection Testing. Journal of Software,2009, 20(6):1425-1443

[2] Xu Z W, Wu M F. Formal Fault Tree Analysis Modeling and Software Safety Testing. Journal of Tongji University (Natural Science),2001, 29(11):1299-1302.

[3] Yang G H, Qi X, Shi Y S. Software Security Testing Based on Threat Model. Computer Security, 2010,02:11-13+17.

[4] Zhao C G, Pu Z B. A software reliability assessment model andITSPetri net description. Computer Applications & Software, 2012, 29(1): 141-144.

[5] Wei Z, Zhou X, Bao H J, et al. Research on Safety Test for InterLocking Software Based on PetriNets. Computer Engineering and Applications, 2005, 17:123-125+138.

[6] Dugan J. B, Trivedi K S, Geist R M, et al. Extended stochastic Petri nets: Applications and analysis. In, Performance' 84, Proc 10th Int'l Symp on Computer Performance Modelling, Measurement and Evaluation, Amsterdam, Elsevier, 1984,507-519

[7] Mellor P. Analysis of software failure data(l): adnptation of coarse data. ICL Techical, Vol.1(2),1984.

[8] Muss TD, Okumoto K, Application of basic and logatithmic Poisson execution time model in software reliability measurement. In: 1988. RELIABLITY AND MAITAINABLITY Sympes IUM.190-194.

[9] Littbewoood B, Verrall JL, On the Likelihood function of a debugging model for software reliability. IEEE Trans on Reliability, Vol R-30,1981:115-118.

[10] Downs T, Garrone P. Some new model of software testing with performance comparisons, IEEE on Rrliab, l: ty Vol R-40(3),1991,322-328. [11] Zuberek W M. Performance evaluation using unbound timed Petri nets. In: proc of the Third Int'l Workshop on Petri Nets and Performance Models, Kyoto. Janpan, 1989,180-186.

# Software security test based on SPN

Mehran Aghaei
Department of Engineering, Faculty of Payame Noor University(PNU),
P.O.Box, 19395-4697, Tehran, Iran, E-mail: m.aghaei@pnu.ac.ir

**Abstract.** Software security is an important characteristic of software, some of the current software security testing methods are often based on the accurate parameters or on the basis of function structure, and cannot adapt to the complex environment and safety requirements, particularly in scale distributed environment. Based on Stochastic Petri nets (SPN) theory, this paper analyzes the security of the application software by the SPN reliability measurement parameters, and then react up on the analysis and design of the software system, in order to explore a kind of application software security testing method based on SPN model. With example analysis, the method can find the disable system state or state combination intuitively, rapidly, and effectively, meanwhile provides a new method for the application software security test.

**Keywords:** Stochastic Petri net, Reliability software testing, Markov chain model, Software Security Analysis