رویکردی مبتنی بر شبکه‌های پتری و حساب رخداد برای تحلیل ایمنی رفتار نرم‌افزار

سید مرتضی بابامیر
استادیار گروه مهندسی کامپیوتر دانشگاه کاشان
babamir@kashanu.ac.ir

سعید جلیلی
استادیار گروه مهندسی کامپیوتر دانشگاه تربیت مدرس
sjalili@modares.ac.ir

چکیده – اگر نرم‌افزارهایی که سیستم‌های حساس به ایمنی را کنترل و هدایت می‌کنند رفتار نامطلوب داشته باشند، می‌توانند سیستم را با شکست روبرو کنند. ما در این مقاله ابتدا با استفاده از شبکه‌های پتری و حساب رخداد، رویکردی را برای تحلیل ایمنی رفتار نرم-افزار ارائه می‌دهیم و سپس با طرح مسئله پروتکل ارتباطی مطمئن در شبکه، چگونگی استفاده از رویکردمان را نشان می‌دهیم. رویکرد ما شامل چهار قدم است: (۱) استفاده از روش رسمی شبکه‌های پتری برای توصیف رفتار امن نرم‌افزار، (۲) ارائه طریقی برای استخراج مجموعه گزاره‌های منطقی پایا[1] از شبکه‌های پتری[2] و توصیف آنها با *حساب رخداد*[3]، (۳) توصیف نیازهای ایمنی با حساب رخداد و (۴) استفاده از گزاره‌های منطقی پایا برای تحلیل رفتار نرم‌افزار در برابر نیازهای ایمنی. گزاره‌های منطقی پایا، که هسته تحلیل‌گر رفتار نرم-افزار را تشکیل می‌دهند، گزاره‌های همیشه برقراری هستند که رفتار مطلوب (ارضا نیازهای ایمنی) نرم‌افزار را نشان می‌دهند.

کلید واژه- سیستم‌های حساس به ایمنی، تحلیل پویا ، نیازهای ایمنی، گزاره‌های پایا

۱- مقدمه

هر سیستمی، باید دو ویژگی ایمن‌بودن[4] و حیات‌داشتن[5] را دارا باشد. ایمن‌بودن معرف این است که رخداد بدی (مانند بن‌بست در تخصیص منابع به پردازه‌ها به وسیله سیستم عامل) رخ نمی‌دهد و حیات‌داشتن معرف این است که رخداد خوبی (مانند خاتمه‌پذیری یک برنامه) رخ می‌دهد[۱]. یک سیستم حساس به ایمنی[6] سیستمی است که تخلف از ویژگی ایمن-بودن، منجر به شکست آن می‌شود. شکست سیستم‌های حساس به ایمنی مانند سیستم‌های پزشکی، امنیتی، نظامی و کنترل پرواز، پرهزینه است و باعث از دست‌رفتن زندگی انسان‌ها می‌شود. این سیستم‌ها که عمدتا به صورت‌های توکار[7]، بی‌درنگ، واکنشی[8] و توزیع‌شده هستند، با نرم‌افزارهای حساس به ایمنی هدایت و کنترل می‌شوند. به منظور کاهش احتمال شکست این سیستم‌ها باید تلاش نمود تا قابلیت اعتماد به نرم‌افزار آنها افزایش یابد اما همانطور که نرم‌افزار به عنوان یک مولفه مهم و توکار در سیستم‌های حساس به ایمنی در حال گسترش است، اندازه و پیچیدگی آن نیز در حال افزایش است. افزایش اندازه و پیچیدگی نرم‌افزار، احتمال بروز خطاها را در آن بیشتر می‌کند و در نتیجه احتمال شکست سیستم را افزایش می‌دهد.

به منظور کاهش احتمال شکست سیستم، باید به درستی نرم-افزار پرداخت که به صورت ایستا یا به صورت پویا انجام می-شود. در رویکرد ایستا، با استفاده از روش‌های اثبات قضیه[9] یا بررسی مدل[10]، درستی کلی توصیف نرم‌افزار (یعنی درست-بودن توصیف نرم‌افزار به ازا تمام مقادیر متغیرهای آن) را بدون اجرای آن اثبات می‌کنند. در رویکرد پویا، نرم‌افزار را با تعدادی مقادیر آزمایشی اجرا و نتایج حاصل از اجرا را با نتایج مورد انتظار مقایسه می‌کنند.

اما دو رویکرد فوق‌الذکر با موانعی مواجه هستند. در رویکرد ایستا یعنی وارسی توصیف نرم‌افزار، اثبات قضیه و بررسی مدل در نرم‌افزارهای بزرگ و پیچیده دشوار و در برخی از موارد غیرممکن می‌شود و ممکن است بررسی درستی برخی از نیازها تصمیم‌ناپذیر شود. علاوه بر این در زمان توصیف نرم‌افزار، برخی از شرایط محیطی که نرم‌افزار در آن اجرا خواهد شد، ممکن است قابل پیش‌بینی نباشد. رویکرد پویا یعنی آزمون نرم‌افزار، به دلیل انتخاب تعداد محدودی از داده‌های آزمون و در نتیجه عدم آزمون تمام اجراهای ممکن، فقط حضور خطاها و نه غیبت آنها را نشان می‌دهد[۲]. تولید داده‌های مناسب برای آزمون نرم‌افزار محدودیت دیگر این رویکرد است که ما آن را در [۳] مورد بحث و بررسی قرار داده‌ایم. علاوه بر این نشان داده شده است که آزمون نرم‌افزار می‌تواند احتمال شکست نرم‌افزار را تا مقدار $۱۰^{-۶}$ کاهش دهد در حالی که در نرم‌افزار سیستم حساس به ایمنی، احتمال شکست نرم‌افزار تا

[1] Logical Invariants
[2] Petri Nets
[3] Event Calculus
[4] Safety
[5] Liveness
[6] Safety-Critical
[7] Embedded
[8] Reactive
[9] Theorem Proving
[10] Model Checking