



نگاشت چرخه McGraw به متدولوژی RUP برای توسعه نرم افزار امن

کیوان رحیمی زاده^{۱*}، محمدعلی ترکمانی^۲ و عباس دهقانی^۳

^۱دانشکده فنی و مهندسی، گروه مهندسی کامپیوتر، دانشگاه یاسوج، یاسوج، ایران

^۲کارخانجات مخابراتی ایران، شیراز، ایران

چکیده

امنیت نرم افزار از چالش های مهم در توسعه نرم افزار است. هر روز آسیب پذیری ها و نفوذهای زیادی در نرم افزارهای مشهور گزارش می شود. همان طور که برای حل مشکل بحران نرم افزار بحث مهندسی نرم افزار مطرح شد، مهندسی نرم افزار امن در کاهش چالش های امنیتی نرم افزار مؤثر است. چرخه McGraw به عنوان یکی از رهیافتهای توسعه نرم افزار امن تعدادی نقطه تماس امنیت نرم افزار را معرفی می کند که شامل مجموعه ای از دستورالعمل های صریح و مشخص در راستای اعمال مهندسی امنیت در نیازمندی ها، معماری، طراحی، کدنویسی، اندازه گیری و نگهداری نرم افزار است. نقاط تماس امنیت نرم افزار برای استفاده در ساخت نرم افزار، مستقل از پروسه نرم افزاری است و به هر فرآیند تولید نرم افزار قابل اعمال است. بنابراین، می توان با تغییر چرخه توسعه نرم افزار مورد نظر و اعمال نقاط تماس، چرخه توسعه نرم افزار امن را ایجاد کرد. در این پژوهش، راه کاری برای نگاشت چرخه McGraw به متدولوژی RUP؛ به عنوان متدولوژی سنگین وزن توسعه نرم افزار؛ و تلفیق این دو متدولوژی در راستای ایجاد یک متدولوژی ساده و کارآمد برای توسعه نرم افزار امن (که RUPST نام دارد) ارائه و همچنین، فرآورده های جدید RUP برای توسعه نرم افزار امن به تفکیک هر نظم ارائه و چهار نقش جدید نیز برای انجام فعالیت های مرتبط با امنیت نرم افزار تعریف می شود. راه کار پیشنهادی در یک پروژه واقعی در شرکت کارخانجات مخابراتی ایران مورد استفاده و ارزیابی قرار گرفت. دست آوردها نشان می دهد که بهره گیری و اجرای صحیح این رهیافت توسط توسعه دهندگان، به پیاده سازی و توسعه امن تر و مستحکم تر نرم افزار منجر می شود.

واژگان کلیدی: مهندسی نرم افزار امن، چرخه توسعه نرم افزار، طراحی نرم افزار، نقاط تماس، فرآورده

Mapping of McGraw Cycle to RUP Methodology for Secure Software Developing

Keyvan RahimiZadeh^{1*}, Mohamamd Ali Torkamani² & Abbas Dehghani³

^{1,3} Department of Computer Engineering, Yasouj University, Yasouj, Iran

² Iranian Telecommunication Manufacturing Company, Shiraz, Iran

Abstract

Designing a secure software is one of the major phases in developing a robust software. The McGraw life cycle, as one of the well-known software security development approaches, implements different touch points as a collection of software security practices. Each touch point includes explicit instructions for applying security in terms of design, coding, measurement, and maintenance of software. Developers are able to provide secure and robust software by applying such touch points. In this paper, we introduce a secure and robust approach to map McGraw cycle to RUP methodology, named RUPST. The traditional form of RUP methodology is revised based on the proposed activities for software security. RUPST adds activities like security requirements analysis, abuse case diagrams, risk-based security testes, code review, penetration testing, and security operations to the RUP disciplines. In this regard, based on RUP disciplines, new touch points of software security are presented as a table. Also, RUPST adds new roles such as security

* Corresponding author

*نویسنده عهده دار مکاتبات

architect and requirement analyzer, security requirement designer, code reviewer and penetration tester which are presented in the form of a table along with responsibilities of each role.

This approach introduces new RUP artifacts for disciplines and defines new roles in the process of secure software design. The offered artifacts by RUPST include security requirement management plan, security risk analysis model, secure software architecture document, UMLSec model, secure software deployment model, code review report, security test plan, security testes procedures, security test model, security test data, penetration report, security risks management document, secure installation and configuration document and security audit report.

We evaluate the performance of the RUPST in real software design process in comparison to other secure software development approaches for different security aspects. The results demonstrate the efficiency of the proposed methodology in developing of a secure and robust software.

Keywords: Secure software engineering, software development lifecycle, software design, RUP, artifact

برای حل مشکل امنیت نرم افزار، متخصصان مهندسی نرم افزار متدولوژی های گوناگونی را برای توسعه نرم افزار امن ارائه کرده اند که می توان به چرخه McGraw [1, 3, 4] و چرخه توسعه امنیت نرم افزار مایکروسافت (SDL) [5, 6, 7] به عنوان مهمترین متدولوژی ها اشاره کرد. McGraw چرخه پیشنهادی خود را روی مدل خطی توسعه نرم افزار ارائه داده است. اما هدف از ارائه این چرخه آن است که برای تمامی متدولوژی های توسعه نرم افزار قابل پیاده سازی باشد. درواقع هدف ارائه دهنده آن، گری مک گرو، این بوده است که چرخه McGraw مستقل از متدولوژی باشد و روی هر متدولوژی قابل اعمال باشد. سایر راه حل ها به طور کامل اختصاصی هستند و فقط برای یک متدولوژی خاص ارائه شده اند. از سوی دیگر، متدولوژی RUP یک متدولوژی شناخته شده و محبوب برای توسعه نرم افزار است و بسیاری از شرکت ها از آن برای توسعه نرم افزار استفاده می کنند؛ اما این متدولوژی از ابتدا با ذهنیت امنیت طراحی نشده است. بنابراین، باید دستورالعمل های صریح و مشخص در راستای اعمال مهندسی امنیت در سرتاسر چرخه توسعه نرم افزار به RUP اضافه شود. این دستورالعمل ها که بر اساس بهترین تجربیات در حوزه مهندسی نرم افزار امن هستند، شامل تعدادی فعالیت اضافی اند که می بایست توسط تیم توسعه نرم افزار انجام شوند. در پژوهش های پیشین راه کاری در خصوص اعمال نقاط امنیت نرم افزار روی RUP انجام نشده است. در این پژوهش، یک مدل برای نگاشت چرخه McGraw به متدولوژی RUP ارائه می شود. همچنین فرآورده های جدید RUP برای توسعه نرم افزار امن به تفکیک هر نظم ارائه و چهار نقش جدید نیز برای انجام فعالیت های مرتبط با امنیت نرم افزار تعریف می شود.

در مدل ارائه شده، نقاط تماس امنیت نرم افزار را که در چرخه McGraw معرفی و به مدل آشناری توسعه نرم افزار

۱- مقدمه

امروزه نرم افزار به بخش جدایی ناپذیر از زندگی بشر تبدیل شده است. چرخه حیات تولید نرم افزار، از موضوعات مهم در تولید یک سامانه نرم افزاری است. برنامه ریزی و هدف گذاری، تحلیل، تولید، آزمایش، استقرار و نگهداری سامانه های نرم افزاری از گام های مهم این چرخه هستند [1]. با توجه به فراگیری نرم افزارها، فراهم سازی امنیت، قابلیت اطمینان نرم افزار و کارایی بسیار مهم و در مواردی امری حیاتی است. در این راه کاهش خطاهای نرم افزار نقش مهمی را ایفا می کند [2].

امنیت برنامه های کاربردی و امنیت شبکه های رایانه ای به تنهایی نمی توانند امنیت مورد نیاز کاربران را برآورده سازد. امنیت برنامه های کاربردی به معنی حفاظت از نرم افزار و سامانه ای که نرم افزار پس از ساخت و توسعه روی آن اجرا می شود، است نرم افزارها با چالش های هم چون، جداسازی (ایزوله) کد در جعبه شنی^۱ (برای نمونه ماشین مجازی جاوا)، حفاظت در قبال کد مخرب، قفل کردن برنامه های اجرایی، پایش برنامه ها به هنگام اجرا (به خصوص ورودی برنامه ها)، اعمال سیاست های استفاده از نرم افزار و سامانه های قابل توسعه روبه رو هستند.

هرگاه نرم افزار از درون آسیب پذیر باشد، هیچ راه کاری به جز مسدود کردن و عدم استفاده از نرم افزار وجود ندارد. راه حل اصلی رهایی از این مشکل، مهندسی نرم افزار امن است. در مهندسی نرم افزار امن، از نخستین گام تا گام پایانی توسعه نرم افزار، امنیت نرم افزار در نظر اعمال می شود و ریسک های ممکن نرم افزار مدیریت می شوند. در این راستا، با استفاده از ابزارهای مناسب آزمون های امنیتی مختلف را روی سامانه انجام می شود و کدهای نرم افزاری نیز از نظر امنیتی مرور شده تا برنامه ها عاری از آسیب پذیری باشند و نرم افزاری امن و قابل اطمینان تولید شود.

¹ Sandboxing

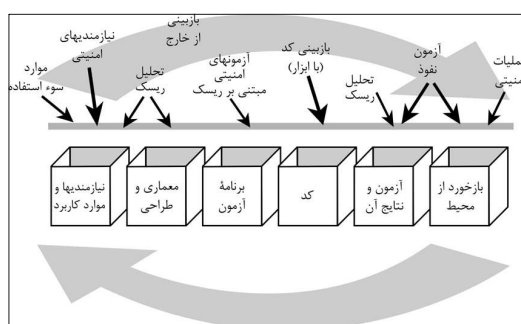
² Security development lifecycle

³ Artifacts

چنانچه نقاط تماس را داشته باشد، چرخه توسعه نرم افزار امن را ایجاد کرد [4].

نقاط تماس امنیت نرم افزار، مجموعه ای از بهترین تجارب امنیت نرم افزار است و شامل دستورالعمل های صریح و مشخص به منظور دخیل کردن مهندسی امنیت در نیازمندی ها، معماری، طراحی، کدنویسی، اندازه گیری و نگهداری نرم افزار است. این دستورالعمل ها می بایست در قالب مراحل و نقاط مشخص و قابل لمس معرفی شوند. این مراحل به هر فرآیند تولید نرم افزار قابل اعمال هستند. به عبارت دیگر، نقاط تماس امنیت نرم افزار، مستقل از فرآیند توسعه ای که به کار برده شده است، طراحی شده اند. نقاط تماس تمامی مراحل چرخه حیات نرم افزار را شامل می شود و هدف آن، اتخاذ مجموعه ای ساده و روشن از بهترین تجارب مهندسی که برای ورود امنیت به فرآیند توسعه نرم افزار طراحی شده اند، است.

نقاط تماس هم زمان با تکامل نرم افزار بیش از یک بار تکرار می شوند و شامل سازوکارهای امنیتی (همانند کنترل دسترسی) و طراحی برای امنیت است. این سازوکارها هم شامل فعالیت های کلاه سیاه (مخرب و مربوط به نفوذگر) و هم فعالیت های کلاه سفید (سازنده و مربوط به توسعه دهنده) است. طبیعتاً، برخی نقاط تماس قدرتمندتر از سایرین اند و اولویت بیشتری دارند. نقاط تماس به ترتیب اولویت عبارتند از: مرور کد، تحلیل ریسک های معماری، آزمون نفوذ، آزمون های امنیتی مبتنی بر ریسک، سناریوهای سوء موارد کاربری، نیازمندی های امنیتی، عملیات (اپراتوری) امنیتی [3]. در ادامه اولویت های نقاط تماس مورد بررسی قرار می گیرند.



شکل-۱: چرخه McGraw [3]

(Figure-1): McGraw Cycle [3]

۲-۱- مرور کد

هدف از مرحله مرور کد، یافتن خطاها و آسیب پذیری های امنیتی موجود در کد برنامه است. مرور کد توسط ابزارهای

اضافه شده است، به متدولوژی RUP اعمال می شود. استفاده از این مدل، که RUPST^۱ نامیده می شود، به توسعه دهندگان کمک می کند تا نرم افزاری امن تر تولید کنند. این مدل در مقایسه با پژوهش های پیشین که سعی در امن کردن متدولوژی RUP داشته اند و بیشتر به تحلیل و طراحی نیازمندی های نرم افزار توجه کرده اند، بسیار جامع است و تمامی گام ها و مراحل توسعه نرم افزار را دربر می گیرد. از آنجا که اصول مهندسی نرم افزار امن در تمام مراحل این مدل در نظر گرفته شده است، نرم افزاری که به این روش توسعه یابد، از جنبه هایی همانند معماری، طراحی و پیاده سازی امن خواهد بود. مدل پیشنهادی RUPST در یک پروژه واقعی مورد استفاده قرار گرفته است. نتایج ارزیابی این مدل نشان می دهد که بهره گیری و اجرای صحیح این رهیافت توسط توسعه دهندگان، به پیاده سازی و توسعه امن تر و مستحکم تر نرم افزار منجر می شود.

ساختار ادامه مقاله به این ترتیب است که در بخش دوم، چرخه McGraw برای توسعه نرم افزار امن بررسی می شود. در بخش سوم، متدولوژی RUP، که یکی از متدولوژی های مهم توسعه نرم افزار است به اختصار بیان می شود. در بخش چهارم، پژوهش های پیشین بررسی می شود. در بخش پنجم، یک مدل برای نگاشت چرخه McGraw به متدولوژی RUP در راستای ایجاد یک متدولوژی ساده و کارآمد برای توسعه نرم افزار امن (RUPST) ارائه می شود و بخش ششم به یک مطالعه موردی درخصوص استفاده از RUPST در یک پروژه واقعی اختصاص دارد. در بخش هفتم مزایای مدل و به دنبال آن در بخش هشتم مقایسه روش پیشنهادی با سایر روش ها از جنبه های مختلف عرضه و در بخش پایانی نیز دست آوردهای این پژوهش ارائه می شود.

۲- چرخه McGraw

چرخه McGraw (شکل (۱)) نخستین بار در مجله امنیت و محرمانگی IEEE Security & Privacy در سال ۲۰۰۴ ارائه شد. این مدل صرف نظر از اینکه از کدام یک از فرایندهای پایه توسعه نرم افزار همانند حلزونی، توسعه مبتنی بر مؤلفه، RUP، Agile، XP استفاده می شود، قابل استفاده است. تعریف چرخه توسعه امن دلخواه با اضافه کردن یک مجموعه نقاط تماس امنیت نرم افزار^۲ به فرآیند موجود امکان پذیر است؛ بنابراین، می توان با تغییر چرخه توسعه نرم افزار (SDLC^۳)،

^۱ RUP with Security Touchpoints

^۲ Software security touchpoints

^۳ Software Development Life Cycle

مختلف انجام می‌شود. درواقع، کد منبع برنامه باید جستجو و آسیب‌پذیری‌های آن کشف شود. مرور کد یک عملیات لازم ولی نه کافی برای رسیدن به نرم‌افزار امن است. بهترین اکتشافی که مرور کد می‌تواند انجام دهد، آشکارسازی بخش مهمی از مشکلات امنیتی است. یافتن مشکلات معماری نرم‌افزار از کد برنامه، کار بسیار دشواری است. یک راه‌برد کامل، ترکیب مرور کد و تحلیل ریسک‌های معماری است. نمونه‌ای از مخاطراتی که به‌عنوان خروجی این مرحله مشخص می‌شود، مشاهده خطای سرریز بافر در یکی از خطوط برنامه است.

۲-۲- تحلیل ریسک معماری

در تولید نرم‌افزار، فرآورده‌ها عبارتند از محصولات (یا قطعات اطلاعاتی) که در طی فرآیند تولید نرم‌افزار، ایجاد، استفاده، یا به‌روزرسانی می‌شوند. فرآورده مهم بعدی در توسعه نرم‌افزار، توصیف سامانه و طراحی است. هدف از مرحله تحلیل ریسک معماری، یافتن عیب‌های معماری و طراحی است. حدود نیمی از مشکلات امنیتی به خطاهای طراحی برمی‌گردد. اتکا به کد برای پیدا کردن این عیوب کارساز نیست و نمی‌توان خطاهای طراحی را از روی کد پیدا کرد. برای انجام این کار یک درک سطح بالا از مسأله نیاز است. طراحان، معماران و تحلیل‌گران نرم‌افزار باید مفروضات را به‌گونه‌ای صریح مستند کنند که شناسایی حملات احتمالی، ممکن باشد. در این نقطه، تحلیل‌گران امنیتی عیب‌های معماری را کشف و رتبه‌بندی می‌کنند تا بتواند اصلاح و رویه کاهش ریسک آغاز شود. نمونه‌هایی از این‌گونه ریسک‌ها به‌صورت زیر است:

- افزاز نامناسب و محافظت ضعیف از داده‌های حساس
- ناتوانی وب‌سرویس از احراز اصالت کد فراخوانی‌شده و کاربر آن
- مناسب‌نبودن تصمیمات مربوط به کنترل دسترسی

۲-۳- آزمون نفوذ

براساس شکل (۱)، محل انجام آزمون نفوذ^۱ در چرخه McGraw در دو گام آخر مدل خطی توسعه نرم‌افزار است. آزمون نفوذ به‌عنوان یک تلاش قانونی و مجاز به بهره‌برداری از سامانه‌های رایانه‌ای برای ساخت سامانه امن‌تر و محافظت بیشتر توصیف شده است. آزمون نفوذ مناسب همیشه با توصیه‌های خاص برای تقویت امنیت به پایان می‌رسد.

^۱ Penetration testing

تفاوت آزمون نفوذ با آزمون‌های کارکردی رایج در نرم‌افزار آن است که آزمون کارکردی، آزمون مثبت‌ها است و به‌طورمعمول به‌دنبال ویژگی‌های عملکرد صحیح برنامه بر اساس کارکردهای تعیین شده است؛ ولی، آزمون نفوذ یعنی آزمون منفی‌ها و هدف آن یافتن حفره‌هایی است که ممکن است توسط هکرها مورد سوء استفاده قرار گیرد. نکته مهم این است که باید از آزمون نفوذ به‌عنوان «آخرین راه‌کار» و نه نخستین گام امنیتی فرآیند توسعه نرم‌افزار، استفاده شود.

۴-۲- آزمون امنیتی مبتنی بر ریسک (ریسک

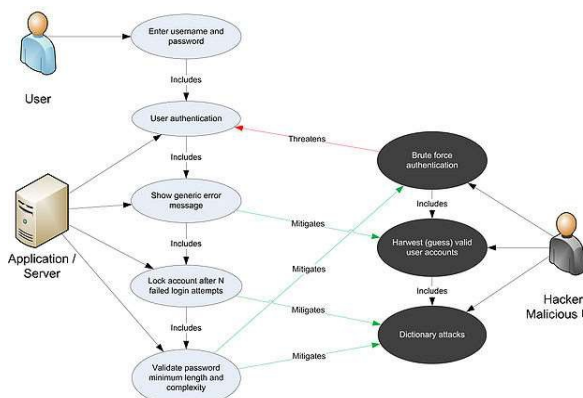
مینا)

براساس شکل (۱)، جایگاه آزمون امنیت بر مبنای ریسک در مرحله آزمون و نتایج آن است. آزمون امنیت، فعالیتی فراتر از قلمرو پوشش پورت شبکه است که به‌وسیله بسیاری از نرم‌افزارهای آزمون نفوذ انجام می‌شود. اگر آزمون امنیتی مبتنی بر ریسک به‌درستی انجام شود، تأثیر آن بسیار دقیق‌تر و عمیق‌تر از آزمون‌های جعبه سیاه همانند آزمون نفوذ و جعبه سفید است. این آزمون حتی فراتر از آزمون عملکردی سازوکارهای امنیتی نرم‌افزار خواهد بود. آزمایش‌کنندگان باید رویکردی مبتنی بر ریسک را هم در زمینه معماری سامانه و هم درخصوص رفتار یک مهاجم استفاده کنند. با تعیین ریسک‌های سامانه و ایجاد پیش‌رمان‌های آزمون با درنظرگرفتن این ریسک‌ها، یک آزمون‌کننده امنیت نرم‌افزار می‌تواند به‌درستی بر نواحی کد که ممکن است، روی آن‌ها حملات موفقی انجام شود، تمرکز کند.

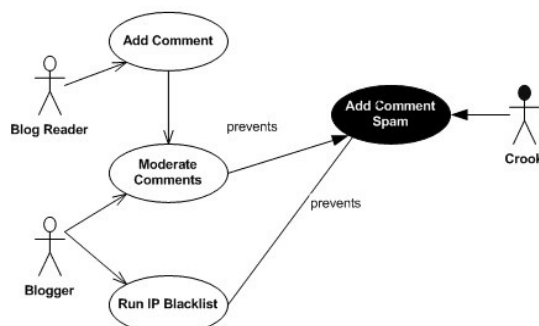
تفاوت اصلی آزمون امنیت و آزمون نفوذ، سطح رده‌یافت و زمان اختصاص داده‌شده به آزمون است. آزمون نفوذ وقتی انجام می‌شود که نرم‌افزار به‌طورکامل در محیط‌های عملیاتی در حال اجرا باشد. اما آزمون امنیت می‌تواند قبل از تکمیل نرم‌افزار در سطح واحد انجام شود. به عبارت دیگر، قبل از فرآیند یک‌پارچه‌سازی^۲ باید از سطح مؤلفه یا واحد شروع کرد. همچنین، آزمون نفوذ رویکردی بیرون به درون دارد و تا حدودی نیز کوتاه، اما رویکرد آزمون امنیتی مبتنی بر ریسک هم از بیرون به درون و هم از درون به بیرون سامانه است. آزمون امنیتی این مزیت را دارد که می‌توان امنیت سامانه را به تعدادی از قطعات گسسته تجزیه کرد و آزمون‌گر به‌طوردقیق می‌تواند روی بخش‌هایی از کد حمله روی آن موفق می‌شود، تمرکز و ریسک‌ها را

^۲ Integration

شناسایی کند. از نظر تئوری، اگر هر مؤلفه به صورت امن پیاده سازی و معیارهای طراحی بین مؤلفه برآورده شده باشد، سامانه ای که از کنار هم قرارگرفتن این مؤلفه ها به دست می آید، نیز به شکل امن خواهد بود؛ اما آزمون امنیتی باید در سطح سامانه نیز ادامه بیاید و سطح تجمیع شده را نیز در برگرد (نقطه تلاقی آزمون نفوذ و آزمون امنیتی). بنابراین، در ادامه باید آزمون امنیت در سطح سامانه انجام شود و خواص نرم افزار مجتمع شده نیز ارزیابی شود. این نقطه به طور دقیق همان جایی است که آزمون نفوذ و آزمون امنیت با یکدیگر هم پوشانی دارند.



(شکل-۳): یک نمونه نمودار سوء کاربرد
(Figure-3): A sample of misuse case diagram



(شکل-۴): نمودار سوء موارد کاربردی انتشار هرزنامه در یک وبلاگ

(Figure-4): Misuse case diagram for spamming

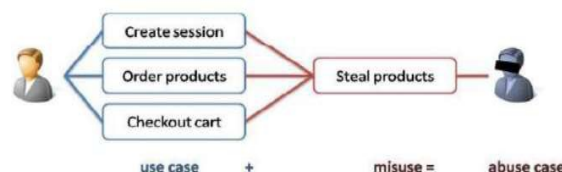
۲-۷- اپراتوری امنیتی

پیکربندی درست برنامه برای محیط کاربردی خاص در این گام انجام می شود. بنابراین نیاز به فراهم آوری محیطی مناسب در سطح شبکه است تا سیستم عامل و البته تنظیمات درست امنیتی خود برنامه کاربردی فراهم شود. دیگر نقاط قابل طرح در این گام عبارتند از ثبت کردن مرور امنیتی نهایی، مدیریت تغییرات (CAB⁴) و برنامه رصد امنیتی و پاسخ گویی.

⁴ Change Advisory Board

۵-۲- نمودار سوء موارد کاربری

نمودار موارد کاربری^۱ در UML برای مدل سازی کارکردهای برنامه و آنچه برنامه باید انجام دهد استفاده می شود. از نمودار سوء موارد کاربری^۲ برای مواردی که نمی خواهیم در برنامه اتفاق بیافتد، استفاده می کنیم؛ در واقع، نمودار موارد کاربری نشان دهنده مواردی است که کاربر انجام می دهد اما نمودار سوء موارد کاربری مواردی را نشان می دهد که حمله کننده مایل است انجام دهد؛ بنابراین موارد سوء استفاده^۳، شبیه سناریوی کاربری هستند، با این تفاوت که آنها رفتار برنامه را در شرایط حمله نشان می دهند. برای ایجاد نمودار سوء موارد کاربری، لازم است پاسخ صریحی به این سؤال داده شود: از چه چیزی، از چه کسی و تا چه زمانی باید محافظت شود؟ همان طور که در شکل (۲) دیده می شود، Abuse case از دو قسمت use case و misuse تشکیل شده است. شکل های (۳) و (۴) دو نمونه نمودار موارد کاربری را نشان می دهد.



(شکل-۲): اجزای نمودار سوء موارد کاربری

(Figure-2): Abuse cases diagram

۶-۲- نیازمندی های امنیتی

نیازمندی های امنیتی سامانه باید با دقت و به طور کامل مدل سازی شوند. مدل سازی نیازمندی های امنیتی نرم افزار با

¹ Use case

² Abuse cases diagram

³ Abuse case

۳- متدولوژی RUP

در این بخش متدولوژی RUP مورد بررسی قرار می‌گیرد. فرایند RUP یک راه‌کار نظام‌مند برای تخصیص کارها و مسئولیت‌ها در یک تیم توسعه نرم‌افزار است و هدف آن توسعه نرم‌افزار با کیفیت بالا است تا نیازهای کاربران نهایی را توسط یک برنامه با بودجه قابل پیش‌بینی تأمین کند. RUP قابلیت سفارشی‌سازی را دارد. این موضوع به این علت است که هیچ فرایند واحدی برای همه نرم‌افزارها مناسب نیست. در RUP دوران حیات یک نرم‌افزار به چهار گام تقسیم می‌شود [1]. در گام آغازین^۱، که در آن محدوده پروژه مشخص می‌شود. در گام تشریح^۲، برنامه‌ریزی پروژه برای دستیابی به معماری پایا^۳ انجام می‌گیرد. در گام سوم (گام ساخت)، محصول نرم‌افزاری ساخته می‌شود و در گام نهایی (مرحله انتقال)، نرم‌افزار به جامعه کاربران تحویل می‌شود.

در طول چرخه حیات نرم‌افزار، مجموعه‌ای از نشرها و نسخه‌ها برای نرم‌افزار تولید می‌شود تا آن را کامل کند. در هر تکرار نشرهایی صورت می‌گیرد و در برخی از گام‌ها بیش از یک تکرار وجود دارد. برخلاف فرایندهای توسعه سنتی که یک‌بُعدی هستند، فرایندی دوبُعدی است (شکل (۵)). این ابعاد عبارتند از:

۱- بعد فنی (تکنیکی) یا ایستا: در محور عمودی این بعد گردش کارهای اصلی را نشان می‌دهد و شامل جنبه‌های زیر است:

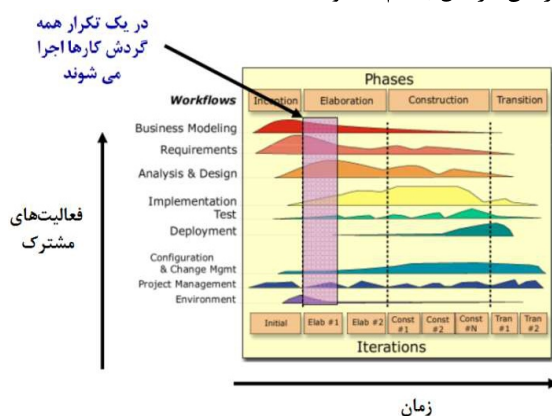
- جنبه‌های کیفی، مهندسی و روش‌های طراحی.
 - فعالیت‌ها، نظم‌ها، خروجی‌ها و نقش‌ها.
- ۲- بعد مدیریتی (بعد زمانی یا بعد پویا): این بعد در محور افقی ساختار چرخه توسعه نرم‌افزار را در RUP در بستر زمان نشان می‌دهد و شامل جنبه‌های زیر است:
- جنبه‌های مالی، راه‌بردی، تجاری و کنترل منابع.
 - چرخه‌ها، گام‌ها، تکرارها

براساس شکل (۵)، محور عمودی نشان‌دهنده گردش کار است. در این شکل، به فعالیت‌هایی که زیر واژه "work flows" نوشته شده (مانند Business Modeling، Analysis & Design، Requirements، ...) نظم می‌گویند. نظم عبارت است از مجموعه‌ای از فعالیت‌های مرتبط که به یکی از نواحی مهم پروژه وابسته است. نواحی مهم اشاره‌ای

¹ Inception
² Elaboration
³ Architecture baseline

به مراحل سنتی فرایند تولید آبشاری را دارد. RUP دارای نه نظم مدل‌سازی حرفه، نیازمندی‌ها، تحلیل و طراحی، پیاده‌سازی، آزمایش، استقرار، مدیریت پیکربندی، مدیریت پروژه و محیط است. در هر نظم توسط تعدادی فعالیت انجام می‌شود تا به یک مجموعه فرآورده‌های خاص دست‌یافته شود. در یک تکرار، همه این فعالیت‌ها اجرا می‌شوند. محور افقی گام‌های چهارگانه نرم‌افزار را نشان می‌دهد [1].

در محور عمودی نیز تکرارها و گردش کارهایی که انجام می‌شود و جزء فعالیت‌های مشترک است، وجود دارد. در همه گام‌ها (آغازین، تشریح، ساخت و انتقال) ما درگیر همه فعالیت‌ها خواهیم شد؛ اما کم‌وکیف آن‌ها با هم متفاوت است. همان‌طور که دیده می‌شود، سطوح زیر منحنی هرکدام از این مراحل با هم متفاوت است.



(شکل-۵): دو بعد فنی و مدیریتی RUP [9]
(Figure-5): Technical and management dimensions RUP [9]

از یک فرایند توسعه نرم‌افزار انتظار می‌رود که مشخص کند چه کسی، چه چیزی، به چه صورت، و در چه زمانی را باید انجام دهد. RUP، چهار عنصر مدل‌سازی زیر را برای این چهار سؤال ارائه می‌کند:

- ۱- نقش: چه کسی باید انجام دهد (Who).
- ۲- فعالیت: چگونه باید انجام شود (How).
- ۳- فرآورده: چه چیزی باید تولید شود (What).
- ۴- نظم: در چه زمانی باید انجام شود (When).

بنابراین در فرایند RUP تعدادی نقش وجود دارد و هر نقش تعدادی فعالیت انجام می‌دهد. در انجام هر فعالیت با تعدادی فرآورده سروکار داریم. یک نقش می‌تواند مسئول تعدادی فرآورده باشد؛ درعین حال یک فرآورده در قالب تعدادی نظم طراحی می‌شود. فرآورده‌ها عبارتند از محصولات (یا قطعات اطلاعاتی) که در طی فرایند تولید نرم‌افزار، ایجاد، استفاده، یا به‌روزرسانی می‌شوند.

امنیتی نرم افزارها از جنبه های مختلفی چالش برانگیز است. از یک سو، ناامن بودن نرم افزارها هزینه زیادی را برای سازندگان به همراه دارد و از سوی دیگر، تمایل کاربران به استفاده از نرم افزاری که تمهیدات امنیتی را در آنان مورد توجه قرار نگرفته است کاهش می دهد. مهندسی نرم افزار امن می تواند در راستای کاهش هزینه ها و افزایش شمار کاربران دارد.

پژوهش های زیادی در زمینه امن سازی نرم افزارها انجام گرفته است [19-25]. در [8, 9] یک مدل فرایندی برای مهندسی نیازمندی های امنیتی موسوم به Square RUP که تأکید آن بر تحلیل نیازمندی های امنیت و مدیریت ریسک و همچنین استفاده از misuse cases است، ارائه شده است. در [10] یک مدل سبک وزن برای امنیت نرم افزار به نام CLASP³ معرفی شده است در محیط های RUP استفاده می شود. نویسندگان فعالیت هایی را برای مهندسی نیازمندی های امنیتی ارائه کرده است که مهم ترین فعالیت های آن تحلیل نیازمندی های امنیتی، استفاده از misuse cases، طراحی رابط کاربری برای امنیت، استفاده از اصول امنیت برای طراحی، انجام آزمون های امنیتی و تعیین پیکربندی های امن پایگاه داده است. در [11] مدلی به نام RUPSec ارائه و نظم های مدل سازی کسب کار و نیازمندی ها در RUP برای توسعه سامانه های امن توسعه داده شده است. RUPSec تعدادی فراورده جدید برای گام های تحلیل و طراحی همانند مدل تهدید، مدل misuse، مدل امنیتی موارد کاربری⁴، مدل misactor ارائه کرده است. همچنین، به تعریف فعالیت های جدید برای مهندسی نیازمندی های امنیتی مانند تعریف سیاست های امنیتی، پیداکردن تهدیدهای سامانه و موارد کاربری امنیتی به امنیت نرم افزار در گام های تحلیل و طراحی توجه کرده است. هرچند روش انجام برخی از کارها به طور دقیق مشخص نشده است. برای نمونه، مشخص نشده از چه روشی باید برای مشخص کردن تهدیدهای سامانه استفاده شود. بیشتر اینکه، درخصوص مدل سازی تهدیدها نیز بحث نشده است. نویسندگان در [12] فعالیت های امنیتی مورد نیاز را در نظم نیازمندی های RUP اضافه کرده اند. همچنین، یک نقش با عنوان خبره امنیت⁵ به نقش های RUP اضافه شده و به مدل سازی نیازمندی ها با استفاده از misuse case توجه شده است. نویسندگان در [13] یک مدل توسعه نرم افزار امن

در RUP فراورده ها در نه مجموعه طبقه بندی می شوند که در جدول (۱) نمایش داده شده اند. در روش توسعه افزایشی این نه مجموعه در طی چرخه توسعه نرم افزار تکامل می یابد [1].

(جدول ۱): دسته بندی فراورده ها در RUP [1]

(Table-1): Artifacts in RUP [1]

نظم	فراورده
مدل سازی تجاری	مدل موارد کاربری تجاری ^۱ ، مستند معماری تجاری، مستند دور نمای تجاری
نیازمندی ها	مستند دورنما، نیازمندی ها (شامل نیازهای ذینفعان، مدل موارد کاربری و مشخصات تکمیلی)، برنامه مدیریت نیازمندی ها
تحلیل و طراحی	مدل تحلیل، مستند معماری نرم افزار، مدل طراحی، مدل استقرار
پایه سازی	کد منبع و فایل های اجرایی، فایل های داده ای مورد نیاز
آزمایش	برنامه آزمایش، روال آزمایش، مدل آزمایش، داده های آزمایش
استقرار	برنامه استقرار، محصول نهایی، مستندات کاربر، مواد آموزشی
مدیریت پیکربندی	برنامه مدیریت پیکربندی، مستند درخواست تغییر
مدیریت پروژه	فراورده های برنامه ریزی (مانند برنامه توسعه نرم افزار، نقشه تکرار، فهرست ریسک ها)، فراورده های عملکردی مانند توصیف نشرها، تشخیص وضعیت پروژه و
محیط	مورد توسعه، راهنمایی های مدل سازی تجاری، راهنمایی های طراحی

در RUP هر فرد می تواند چند نقش را ایفا کند و چند نفر می توانند یک نقش را ایفا کنند. هم افراد متخصص و هم بهره برداران سامانه جزء نقش های RUP هستند. برخی از مهم ترین نقش های RUP عبارتند از: تحلیل گر فرایند کاری^۲، بهره برداران، بازبین کننده تجاری، تحلیل گر سامانه، معمار سامانه، طراح واسط کاربر، طراح پایگاه داده، بازبین کننده های معماری و طراحی، طراح آزمایش، یک پارچه ساز سامانه، مدیر استقرار، مدیر پیکربندی، مدیر پروژه، مدیر کنترل تغییرات، مهندس فرایند و متخصص ابزار. نمونه ای از فعالیت ها در گام بازبینی طراحی یافتن موارد کاربری و عامل ها توسط تحلیل گر سامانه، بازبین کننده طراحی است [9].

۴- پیشینه پژوهش

امنیت نرم افزار به عنوان یکی از مهم ترین گام های ساخت و توسعه نرم افزار است [14, 15, 16]. امروزه، بروز حملات فراوان به نرم افزارها، چالش های فراوانی را برای کاربران و فراهم کنندگان نرم افزارها به وجود آورده است. کاستی های

³ Comprehensive, lightweight application security process

⁴ Security use case model

⁵ Security expert role

¹ Business usecase model

² Business process analyst

مبتنی بر RUP برای توسعه سرویس‌های ارزش افزوده ارائه داده‌اند. در راستای این هدف، به گام‌های مختلف RUP، فعالیت‌های امنیتی لازم برای توسعه سرویس‌های ارزش‌افزوده اضافه شده است. همچنین نقش‌های لازم همانند تحلیل‌گر امنیتی^۱ و توسعه‌دهنده امنیت^۲ در نظر گرفته شده است.

درکل تمرکز بیشتر پژوهش‌گران بر مهندسی نیازمندی‌ها بوده است و اغلب، تمامی گام‌های توسعه نرم‌افزار را مورد توجه قرار نداده‌اند. برخی از فعالیت‌ها نیز به‌طور کلی بررسی شده و راه‌کار نظام‌مندی برای انجام کار ارائه نشده است.

در این پژوهش، یک مدل برای نگاشت چرخه McGraw به متدولوژی RUP ارائه می‌شود. هدف، افزودن تعدادی فعالیت به متدولوژی RUP در راستای اعمال مهندسی امنیت است. این فعالیت‌های اضافی بر اساس بهترین تجربیات در حوزه مهندسی نرم‌افزار امن است و باید توسط تیم توسعه نرم‌افزار انجام شوند. درواقع در مدل ارائه‌شده، نقاط تماس امنیت نرم‌افزار را که توسط McGraw معرفی و به مدل آشناری توسعه نرم‌افزار افزوده شده است، به متدولوژی RUP اعمال می‌شود. استفاده از این مدل به توسعه‌دهندگان کمک می‌کند تا نرم‌افزاری امن‌تر تولید کنند. از آنجا که اصول مهندسی نرم‌افزار امن در تمام مراحل این مدل در نظر گرفته شده است، نرم‌افزاری که به این روش توسعه یابد، جنبه‌هایی همانند معماری، طراحی و پیاده سازی امن را دارد. این مدل بسیار کارآمد و به‌کارگیری آن نیز ساده است.

۵- راه‌کار پیشنهادی برای نگاشت چرخه McGraw و متدولوژی RUP (RUPST)

در این بخش یک راه‌کار برای نگاشت چرخه McGraw به متدولوژی RUP و تلفیق این دو متدولوژی در راستای ایجاد یک متدولوژی ساده و کارآمد برای توسعه نرم‌افزار امن (RUPST) ارائه می‌شود. برای ارائه راه‌کار پیشنهادی، فعالیت‌های نظم‌های مختلف RUP در نظر گرفته شده‌اند و سپس جایگاه هر یک از نقاط تماس چرخه McGraw در نظم‌های متدولوژی RUP مشخص می‌شود. نتیجه این کار در جدول (۲) نشان داده شده است. بر اساس جدول (۲)

¹ Security analyst

² Security developer

می‌توان شکل متدولوژی سنتی RUP را بر اساس فعالیت‌های جدید تصحیح کرد. شکل (۶) فعالیت‌های اضافی را (پیشنهادی) در راستای تولید و توسعه نرم‌افزار امن در متدولوژی RUP نشان می‌دهد. همان‌طور که در این شکل مشاهده می‌شود، در نظم‌های مدل‌سازی تجاری و نیازمندی‌ها، دو فعالیت اضافی نسبت به RUP سنتی انجام می‌شود که عبارتند از تحلیل نیازمندی‌های امنیتی و ترسیم سناریوهای سوء موارد کاربری. تحلیل نیازمندی‌های امنیتی موجب می‌شود که توسعه‌دهندگان توجه خاصی به مدل‌سازی نیازهای امنیتی داشته باشند. با استفاده از سناریوهای سوء موارد کاربری توسعه‌دهندگان می‌توانند سناریوهای مختلف نرم‌افزار را از دیدگاه نفوذگران بررسی و مدل‌سازی کنند.

(جدول-۲): نقاط تماس امنیت نرم‌افزار بر اساس نظم‌های RUP

(Table-2): Touch points in RUP

نظم‌های RUP	فعالیت چرخه McGraw
مدل‌سازی تجاری	تحلیل نیازمندی‌های امنیتی ترسیم سناریوهای سوء موارد کاربری
نیازمندی‌ها	تحلیل نیازمندی‌های امنیتی ترسیم سناریوهای سوء موارد کاربری
تحلیل و طراحی	ترسیم سناریوهای سوء موارد کاربری تحلیل ریسک معماری
پیاده‌سازی	مرور کد
آزمایش	آزمون‌های امنیتی ریسک مبنا
استقرار	آزمون نفوذ اپراتوری امنیتی
مدیریت پیکربندی	مرور کد آزمون نفوذ اپراتوری امنیتی
مدیریت پروژه	مدیریت ریسک‌های امنیتی
محیط	اپراتوری امنیتی

تحلیل نیازمندی‌های امنیتی یکی از فعالیت‌های کلان در چرخه McGraw است و می‌تواند شامل فعالیت‌هایی مانند مشخص کردن سازوکارهای امنیتی مورد نیاز در پروژه، تحلیل ریسک‌های امنیتی، ترسیم درخت حمله، مدل‌سازی تهدیدها به روش STRIDE و یا سایر فعالیت‌های مرتبط با امنیت اطلاعات بر اساس نیازهای پروژه و تشخیص متخصصین امنیت نرم‌افزار در توسعه نرم‌افزار باشد. در نظم تحلیل و طراحی نیز ترسیم سناریوهای سوء موارد کاربری و تحلیل ریسک‌های معماری انجام می‌شود. معماری نرم‌افزار اهمیت خاصی در توسعه نرم‌افزار دارد. بنابراین توسعه‌گران باید به تحلیل ریسک‌های معماری توجه خاصی کنند. در نظم پیاده‌سازی، مرور کد با ابزارهای تحلیل

(جدول-۳): فراورده های جدید RUP برای توسعه نرم افزار امن

به تفکیک نظم

(Table-3): New artifacts for developing secure software development in RUP

نظم	فراورده جدید
مدل سازی تجاری	مدل سوء موارد کاربری تجاری
نیازمندی ها	مدل سوء موارد کاربری ، برنامه مدیریت نیازمندی های امنیتی
تحلیل و طراحی	مدل تحلیل ریسک های امنیت، مستند معماری نرم افزار امن، مدل های تحلیل و طراحی امن با UMLSec، مدل استقرار امن
پایاده سازی	گزارش مرور کد
آزمایش	برنامه آزمون های امنیتی، روال آزمون های امنیتی، مدل آزمون های امنیتی، داده های آزمایش امنیتی
استقرار	گزارش آزمون نفوذ، مستند نصب و پیکربندی امن نرم افزار
مدیریت پیکربندی	گزارش آزمون نفوذ، گزارش مرور کد
مدیریت پروژه	مستند مدیریت ریسک های امنیتی
محیط	مستند نصب و پیکربندی امنیتی ابزارها

(جدول-۴): نقش های جدید برای انجام فعالیت های مرتبط با

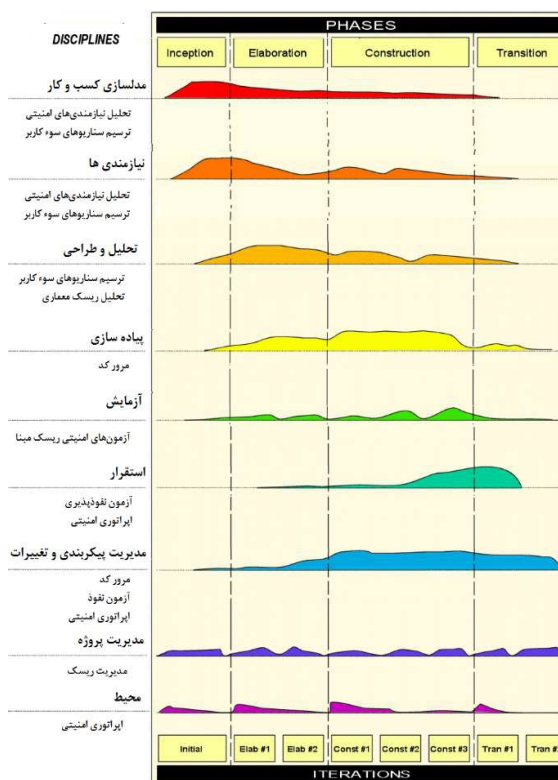
توسعه نرم افزار امن در RUP

(Table-4): New roles for developing secure software development in RUP

نقش جدید	وظیفه
تحلیل گر و معمار امنیت ^۱	مشخص کردن مکانیزم های امنیتی مورد نیاز در پروژه، تحلیل نیازمندی های امنیتی، تحلیل ریسک های امنیتی، ترسیم نمودارهای Abuse case، ترسیم درخت حمله، مدل سازی تهدیدها به روش STRIDE
طراح نیازهای امنیتی ^۱	ترسیم نمودارهای Abuse case، ترسیم نمودارهای UMLSec برای نیازمندی های امنیتی سیستم، تعریف موارد آزمون امنیتی ^۱
مرور کننده کد ^۱	مرور کد با استفاده از ابزار و مشخص کردن آسیب پذیری ها و مشکلات امنیتی موجود در کد.
مسئول آزمون نفوذ ^۱	انجام آزمون نفوذ با استفاده از ابزارهای مختلف، ارائه گزارش آزمون نفوذ

در اپراتوری امنیتی نیز تنظیمات و پیکربندی امن نرم افزار انجام می شود. بخش هایی از نرم افزار که استفاده از آنها خطرناک است و نیاز به دقت دارد، نباید به صورت پیش فرض فعال باشد. به عنوان مثال در گوشی های تلفن همراه خاصیت Developer یا Root گوشی به صورت پیش فرض غیرفعال است. تنظیمات پیش فرض محافظه کارانه، اجتناب از تغییرات پیش فرض خطرناک و غیرفعال کردن خدماتی که کمتر به کار برده می شوند، اصول مهمی در مهندسی نرم افزار امن هستند که باید مورد توجه قرار گیرند [6, 7, 18]. در اپراتوری امنیتی همچنین راهنمای

یا مرور کد انجام می شود، تا آسیب پذیری های موجود در کد برنامه مشخص شود. در صورتی که برنامه نویسی ها از کتابخانه های ناامن یا منسوخ استفاده کرده باشند یا اصول کدنویسی امن توسط تیم توسعه رعایت نشده باشد، در این نظم آسیب پذیری مشخص خواهد شد. بسیاری از حملات سایبری رایج نظیر سرریز بافر، XSS و تزریق SQL ناشی از کدهای ضعیف و آسیب پذیر هستند که می توان از آنها به وسیله فعالیت مرور کد پیش گیری کرد. در نظم آزمایش، آزمون های امنیتی ریسک مینا انجام می شود. آزمون های امنیتی بر اساس الگوی حملات، نتایج تحلیل ریسک و سناریوهای سوء موارد کاربری که در نظم های قبل تهیه شده است، انجام می شود. در نظم استقرار نیز دو فعالیت مهم انجام می شود که عبارتند از آزمون نفوذ و اپراتوری امنیتی. در آزمون نفوذ نرم افزار با رویکردی از بیرون به درون مورد ارزیابی امنیتی قرار می گیرد تا آسیب پذیری ها و حفره های امنیتی سامانه مشخص شود. این کار با استفاده از نرم افزارهایی نظیر Acunetix، Kali linux و Meta sploit انجام می شود.



(شکل-۶): فعالیت های پیشنهادی در راستای تولید و توسعه

نرم افزار امن در متدولوژی RUP

(Figure-6): Recommended activities for developing secure software in RUP

استقرار امن نرم‌افزار و ابزارهای استقرار وصله^۱ برای کمک به استقرار وصله ارائه می‌شوند. نظم مدیریت پیکربندی با تغییرات نرم‌افزار مرتبط است. بنابراین مرور کد، آزمون نفوذ و اپراتوری امنیتی از جمله فعالیت‌هایی هستند که در این نظم انجام می‌شوند. همچنین، در نظم مدیریت پروژه، باید مدیریت ریسک انجام شود. این کار شامل ایجاد فهرست جدید ریسک‌ها یا به‌روزرسانی فهرست ریسک‌ها و امتیازدهی و اولویت‌بندی ریسک‌های امنیتی و رسیدگی به ریسک‌ها می‌شود. در نظم محیط نیز باید اپراتوری امنیتی برای استفاده امن از ابزارهای مورد نیاز در توسعه نرم‌افزار انجام شود. به‌خصوص ابزارهایی که برای آزمون امنیت نرم‌افزار به‌کار می‌روند، اهمیت ویژه‌ای دارند.

انجام فعالیت‌های اضافی ذکرشده در این قسمت می‌تواند منجر به ساخت نرم‌افزاری امن‌تر شود. به‌منظور انجام فعالیت‌هایی در راستای تأمین امنیت نرم‌افزار، نیاز به تخصص‌ها و نقش‌های جدیدی در تیم توسعه و همچنین تولید فراورده‌های اضافی در RUP است. مطابق با چرخه McGraw فراورده‌های جدید RUP برای توسعه نرم‌افزار امن در جدول (۳) نمایش داده شده است؛ همچنین نقش‌های جدیدی مورد نیاز در گروه توسعه در جدول (۴) نمایش داده شده است.

۶- مطالعه موردی: استفاده از RUPST در یک پروژه واقعی

راه‌کار ارائه‌شده در این پژوهش در توسعه بخشی از پورتال کارخانجات مخابراتی ایران مورد استفاده قرار گرفته است. برخلاف روش‌های سنتی که درقبل استفاده می‌شد، در ابتدای پروژه علاوه بر تحلیل نیازمندی‌های نرم‌افزار، تحلیل نیازمندی‌های امنیتی به‌عنوان یکی از فعالیت‌های اضافی در نظم‌های مدل‌سازی حرفه و نیازمندی‌ها انجام شد. همچنین، فهرست ریسک‌های امنیتی ایجاد شد. به مدیریت ریسک‌های امنیتی در سرتاسر پروژه توجه شد و فهرست ریسک‌های امنیتی در گام‌های مختلف توسعه به‌روزرسانی و در نظم‌های مدل‌سازی حرفه، نیازمندی‌ها و تحلیل و طراحی، نمودارهای سوء موارد کاربردی ترسیم و به‌روزرسانی شد.

در نظم پیاده‌سازی، مرور کد برای یافتن آسیب‌پذیری‌های امنیتی موجود در برنامه انجام و در نظم آزمایش نیز علاوه بر آزمون‌های عملکردی رایج در مهندسی

^۱ Patch deployment

نرم‌افزار، آزمون‌های امنیتی ریسک مبنا انجام شد. در این نظم، آزمون‌های کارکردهای امنیتی و آزمون‌های امنیتی بر اساس الگوی حملات، نتایج تحلیل ریسک و سناریوهای سوء موارد کاربردی بر اساس سناریوهای مختلف انجام شد. فهرست برخی از ابزارهای استفاده‌شده در این پروژه در جدول (۵) ارائه شده است.

(جدول-۵): ابزارهای استفاده شده در این پروژه

(Table-5): Tools for evaluation the RUPST

ابزار	شرح
ابزار مرور کد در Visual Studio	بررسی آسیب‌پذیری‌های امنیتی موجود در کد برنامه با استفاده از مجموعه قوانین امنیتی مایکروسافت.
Attack Surface Analyzer	یک ابزار تحلیل سطح حمله است که تصویر لحظه‌ای از حالت سیستم قبل و بعد از نصب محصول گرفته و تغییرات را در عناصر کلیدی سطح حمله در ویندوز نمایش می‌دهد [14].
SDL Threat Modeling Tool	این ابزار برای مدل‌سازی تهدیدها به کار می‌رود. ابزاری برای استفاده در فاز طراحی برای معماران نرم‌افزار و مهندسان جهت تحلیل طراحی و معماری قبل از آغاز پیاده‌سازی است.
ابزار فاز آزمون پایه فایل MiniFuzz	کار این ابزار تولید ورودی‌های تصادفی برای آزمون کد اداره فایل (file-handling) است.
ابزار فاز آزمون Regex	الگوهای عبارت‌های منطقی (Regex) می‌توانند در زمان‌های نمای اجرا شوند که این موضوع می‌تواند برای اکسپلویت‌کردن و منع سرویس استفاده شود. نرم‌افزار Regex وجود این آسیب‌پذیری اجرایی نمای را بررسی می‌کند و ابزاری است با هدف پیش‌گیری آسیب‌پذیری‌های حملات منع سرویس است.
FxCop	مایکروسافت FxCop ابزاری برای تحلیل باینری است که تجمع‌های چارچوب (قالب) NET کامپایل شده را برای نقاط ضعف رایج و مشترک برنامه‌نویسی امتحان می‌کند. FxCop می‌تواند کدهای اسمبلی را مدیریت کرده و گزارش‌های مربوط به آن را مورد ارزیابی قرار دهد که می‌تواند شامل طراحی‌های احتمالی، محلی‌سازی، کارایی و ارتفاع سطح امنیتی باشد [15].
CAT.NET	ابزار تحلیل باینری کد است که به توسعه‌دهندگان کمک می‌کند تا انواع آسیب‌پذیری‌های رایج نظیر XSS، تزریق SQL و تزریق Xpath را پیدا کنند [16]
Microsoft Anti-Cross-Site Scripting Library	Anti-XSS library را برای محافظت از برنامه‌های کاربردی تحت وب در مقابل حملات XSS به کار می‌رود [17]. این کتابخانه یک ره‌یافت فهرست سفید مناسب‌تر از متدهای رمزگذاری native در NET. ارائه می‌دهد. همچنین ویژگی پشتیبانی از globalization را نیز دارد که در کتابخانه Net. وجود ندارد.
Microsoft Source Code Analyzer for SQL Injection	این ابزار تحلیل کد ایستا را برای یافتن آسیب‌پذیری‌های تزریق SQL در کدهای ASP فراهم می‌سازد [16]
Kali Linux	انجام آزمون نفوذ روی سامانه

(جدول-۶): برخی از خطاهای کشف شده در اجرای راه کار

پیشنهادی در توسعه بخشی از پورتال

کارخانجات مخابراتی ایران

(Table-6): Reported bugs of implementing RUPST in ITMC

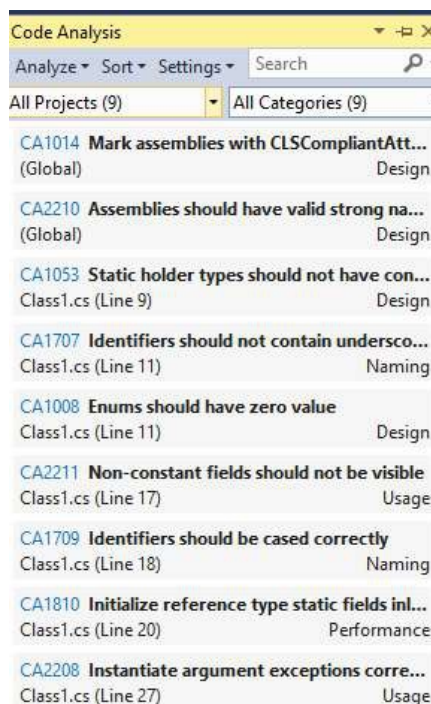
عنوان	محل خطا
۱- عدم رمزنگاری اطلاعات در پایگاه داده SQL Server در سمت سرویس دهنده ۲- استفاده از یک معماری قدیمی برای توسعه نرم افزار ۳- در نظر نگرفتن مباحث امنیتی در طراحی اولیه	معماری و طراحی نرم افزار
۱- استفاده از یک کتابخانه شخص ثالث که دارای آسیب پذیری است. ۲- عدم اعتبارسنجی در ورودی یکی از صفحات طراحی شده ۳- امکان انجام حمله SQL Injection روی یکی از صفحات طراحی شده ۴- تعریف نامناسب متغیرهای یکی از رده ها ۵- وجود آسیب پذیری در اسمبلی ها ۶- نام گذاری نامناسب ۷- خطای امنیتی در Class Library	کدنویسی
۱- عدم کارکرد مناسب پنجره تعیین رمز عبور (پذیرش رمزهای ضعیف به عنوان رمز عبور کاربران) ۲- کند بودن تابع رمزنگاری روی برخی از کلانتهایی که از نظر قدرت پردازشی ضعیف تر هستند ۳- نشت داده ها از سطوح دسترسی بالاتر به پایین تر	آزمون امنیتی مبتنی بر ریسک
۱- یافتن سه نوع آسیب پذیری در سامانه در ارتباط با وجود پورت های باز و حمله XSS	آزمون نفوذ

۷- مزایای مدل پیشنهادی (RUPST)

به اختصار، برخی از مزایای مدل پیشنهادی عبارتند از:

- ۱- در نظر گرفتن اصول مهندسی نرم افزار امن در تمامی مراحل توسعه نرم افزار.
- ۲- امکان به کارگیری روش های مدل سازی امنیت نظیر UMLSec در کنار نمودارهای سنتی UML در RUP.
- ۳- مبتنی بودن بر سناریوهای سوء موارد کاربری و مدل سازی تهدیدها به روش STRIDE.
- ۴- دارای نقش های مناسب برای فعالیت های امنیتی.
- ۵- دارای فرآورده های مورد نیاز در توسعه نرم افزار امن.
- ۶- کارآمدی، سادگی و سهولت استفاده بر اساس مطالعه موردی.
- ۷- یافتن خطاها و نقص های امنیتی در ابتدای پروژه.
- ۸- قابلیت توسعه.
- ۹- عدم وابستگی به ابزارهای امنیتی خاص.

شکل (۷) نمونه ابزار تحلیل کد در ویژوال استودیو را نشان می دهد. همان طور که ملاحظه می شود، این ابزار آسیب پذیری هایی را در نرم افزار یافته است و رهنمودهایی را نیز به منظور رفع آسیب پذیری های امنیتی کد برنامه ارائه می دهد. در نظم استقرار، پیکربندی نرم افزار بر اساس اصول امنیت اطلاعات نظیر تنظیمات پیش فرض محافظه کاران، اجتناب از تغییرات پیش فرض خطرناک و غیرفعال کردن خدماتی که کمتر به کار برده می شوند، توسط پیش فرض که در قبل شرح داده شدند، انجام شد. در آخرین گام از نظم استقرار همچنین آزمون نفوذ روی سامانه انجام شد.



(شکل-۷): نمونه ابزار تحلیل کد در ویژوال استودیو

(Figure-7): Sample code analysis in Visual Studio

جدول (۶) نتایج حاصل از اجرای RUPST در این پروژه را نشان می دهد. استفاده از این ره یافت در توسعه از پورتال کارخانجات مخابراتی ایران موجب شده است که بسیاری از خطاهای امنیتی در گام های ابتدایی توسعه نرم افزار و قبل از مراحل کدنویسی کشف شوند که این موضوع علاوه بر افزایش امنیت سامانه، باعث کاهش هزینه های توسعه نرم افزار شده است. بنابراین استفاده از این متدولوژی باعث جلوگیری از آسیب پذیری در کدهای نرم افزار و امنیت بیشتر سامانه شده است.

۸- مقایسه رهیافت پیشنهادی با سایر روش‌ها

جدول (۷) مقایسه RUPST با تعدادی از روش‌های دیگر را نشان می‌دهد. این مقایسه بر اساس هدفه عامل مهم انجام شده است. بیشتر جنبه‌های مقایسه ذکر شده در این جدول

(جدول-۷): مقایسه روش‌های مختلف با روش پیشنهادی از جنبه‌های مختلف

(Table-7): Comparison of RUPST with other methods

جنیه مقایسه	McGraw	SDL	[11,12]	[13]	[14]	[15]	[17]	RUPST
تحلیل نیازمندی‌های امنیتی در فازهای تحلیل و طراحی و معماری	*	*	*	*	*	*	*	*
مدل‌سازی UMLSec	*	*	*	*	*	*	*	*
مدل Misuse	*	*	*	*	*	*	*	*
استفاده از نمودار سوء موارد کاربری	*	*	*	*	*	*	*	*
مرورکد	*	*	*	*	*	*	*	*
آزمون نفوذ	*	*	*	*	*	*	*	*
آزمون‌های امنیتی ریسک مبنا	*	*	*	*	*	*	*	*
اپراتوری امنیتی	*	*	*	*	*	*	*	*
مدل‌سازی تهدیدها به روش STRIDE	*	*	*	*	*	*	*	*
تعریف دقیق فرایندهای امنیتی در تمامی فازهای توسعه نرم‌افزار	*	*	*	*	*	*	*	*
مشخص‌کردن ابزارهای مورد نیاز برای توسعه امنیت در فازهای مختلف توسعه نرم‌افزار	*	*	*	*	*	*	*	*
مدیریت ریسک‌های امنیتی	*	*	*	*	*	*	*	*
ایجاد نقش‌های جدید متناسب با فرایندهای امنیتی	*	*	*	*	*	*	*	*
تعریف فرآورده‌های جدید برای امنیت	*	*	*	*	*	*	*	*
تاکید بر آموزش امنیت	*	*	*	*	*	*	*	*
ورود بحث حریم خصوصی در کنار امنیت به صورت مجزا	*	*	*	*	*	*	*	*
تطبیق با فازهای RUP	*	*	*	*	*	*	*	*

این روش توسعه یابد، از منظر تمامی جنبه‌ها مانند معماری، طراحی و پیاده‌سازی امن است.

RUPST در یک پروژه واقعی در کارخانجات مخابراتی ایران مورد استفاده قرار گرفت و ارزیابی شد. نتایج حاصل از به‌کارگیری این متدولوژی نشان می‌دهد که استفاده و اجرای صحیح این رهیافت توسط توسعه‌دهندگان، می‌تواند منجر به تولید یک نرم‌افزار بسیار امن‌تر و مستحکم‌تر شود. هر چند اجرای صحیح این رهیافت نیازمند آشنایی توسعه‌دهندگان با اصول مهندسی نرم‌افزار امن (مانند اصول مایکروسافت برای امنیت و حریم خصوصی)، روش‌های مدیریت ریسک‌های امنیتی، تکنیک‌های تحلیل و طراحی نرم‌افزار امن (نظیر مدل‌سازی امنیتی با UMLSec، مدل‌سازی سوء موارد کاربری و مدل‌سازی تهدیدها با استفاده از رهیافت STRIDE مایکروسافت)، تکنیک‌ها و ابزارهای کدنویسی امن و آزمون نفوذ برای توسعه نرم‌افزار امن، است.

۹- نتیجه‌گیری

در این پژوهش، راه‌کاری نوین برای توسعه نرم‌افزار امن با بهره‌گیری از نگاشت چرخه McGraw به متدولوژی RUP ارائه شد. در مدل ارائه شده که RUPST نامیده شد، نقاط تماس امنیت نرم‌افزار به متدولوژی RUP اعمال شد. در این راه، مشخص کردیم که در هریک از نظم‌های RUP می‌بایست کدامیک از نقاط تماس چرخه McGraw انجام شود. همچنین فرآورده‌های جدید RUP برای توسعه نرم‌افزار امن به تفکیک هر نظم ارائه شد. علاوه بر این، چهار نقش جدید برای انجام فعالیت‌های مرتبط با امنیت نرم‌افزار تعریف شد که عبارتند از تحلیلگر و معمار امنیت، طراح نیازهای امنیتی، مرورکننده کد و مسئول آزمون نفوذ. استفاده از این مدل به توسعه‌دهندگان کمک می‌کند که نرم‌افزاری امن‌تر تولید کنند. از آنجا که اصول مهندسی نرم‌افزار امن در تمام مراحل این مدل در نظر گرفته شده است، نرم‌افزاری که به

10- References

۱۰- مراجع

- [11] P. Jaferian, G. Elahi, M. R. A. Shirazi, and B. Sadeghian, "RUPSec: extending business modeling and requirements disciplines of RUP for developing secure systems," in *31st EUROMICRO Conference on Software Engineering and Advanced Applications*, 2005, pp.232-239. DOI: 10.1109/EUROMICRO.2005.51
- [12] H. Mohd and et al., "A secured e-tendering model based on rational unified process (RUP) approach: inception and elaboration phases," *International Journal of Supply Chain Management*. Vol. 5, no 4, pp. 114-120, 2016.
- [13] H. Belani, Z. Car, and A. Caric, "RUP-based process model for security requirements engineering in value-added service development," in *2009 ICSE Workshop on Software Engineering for Secure Systems*, 2009, pp.54-60. DOI: 10.1109/IWSESS.2009.5068459
- [14] "Microsoft Attack Surface Analyzer." [Online]. Available: <https://www.microsoft.com/en-us/download/details.aspx?id=24487>. [Accessed: 15-May-2019].
- [15] "FxCop | Microsoft Docs." [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-3.0/bb429476\(v=vs.80\)](https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-3.0/bb429476(v=vs.80)). [Accessed: 15-May-2019].
- [16] "Microsoft Code Analysis." [Online]. Available: <http://microsoft.github.io/CodeAnalysis/>. [Accessed: 15-May-2019].
- [17] "Microsoft Anti-Cross Site Scripting Library V4.3 from Official Microsoft Download Center." [Online]. Available: <https://www.microsoft.com/en-us/download/details.aspx?id=43126>. [Accessed: 15-May-2019].
- [18] "Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution." [Online]. Available: <https://www.kali.org/>. [Accessed: 15-May-2019].
- [19] C. E. de Barros Paes and C. M. Hirata, "RUP Extension for the Development of Secure Systems," in *Fourth International Conference on Information Technology (ITNG'07)*, 2007, pp. 643-652. DOI: 10.1109/ITNG.2007.171
- [20] R. Kneuper, "Software Processes in the Software Product Life Cycle," in *Software Processes and Life Cycle Models*, Cham: Springer International Publishing, 2018, pp. 69-157. DOI: https://doi.org/10.1007/978-3-319-98845-0_3
- [21] Y. Mufti, M. Niazi, M. Alshayeb, and S. Mahmood, "A Readiness Model for Security
- [1] J. H. Allen, *Software security engineering: a guide for project managers*. Addison-Wesley, 2008.
- [۲] مهدی افتخاری، مریم مجیدی مومن آبادی، مجتبی خمر، "ارائه یک روش فازی-تکاملی برای تشخیص خطاهای نرم افزار"، پردازش علائم و داده ها. ۱۳۹۷، دوره ۱۵، شماره ۴، صفحات ۱۶-۳.
- [2] M. Eftekhari, M.M. Momenabadi, M. Khamar, "Proposing an evolutionary-fuzzy method for software defects detection", *JSDP*. Vol. 15, NO. 4, pp.3-16, 2009.
- [3] G. McGraw, "Software Security: Building Security in," in *2006 17th International Symposium on Software Reliability Engineering*, 2006, pp. 6-16.
- [4] M. Howard and S. Lipner, "The security development lifecycle: SDL, a process for developing demonstrably more secure software", Microsoft Press, 2006.
- [5] "Microsoft SDL Process Guidance updates, version 5.2 - Microsoft Security." [Online]. Available: <https://www.microsoft.com/security/blog/2012/05/23/now-available-microsoft-sdl-process-guidance-updates-version-5-2/>. [Accessed: 14-May-2019].
- [6] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," Springer, Berlin, Heidelberg, 2002, pp. 412-425. DOI: https://doi.org/10.1007/3-540-45800-X_32
- [7] J. Jürjens, *Secure Systems Development with UML*. Springer-Verlag Berlin, Heidelberg, 2010.
- [8] N. R. Mead, T. Stehney, N. R. Mead, and T. Stehney, "Security quality requirements engineering (SQUARE) methodology," in *Proceedings of the 2005 workshop on Software engineering for secure systems building trustworthy applications - SESS '05*, 2005, vol. 30, no. 4, pp. 1-7. DOI: <https://doi.org/10.1145/1082983.1083214>
- [9] N. R. Mead, V. Viswanathan, and J. Zhan, "Incorporating security requirements engineering into standard lifecycle processes," *International Journal of Security and Its Applications*, vol. 2, no. 4, pp. 67-79, 2008. DOI: 10.1109/COMPSAC.2008.85
- [10] H. Assal and S. Chiasson, "Security in the Software Development Lifecycle," in *Fourteenth Symposium on Usable Privacy and Security*, 2018, pp. 281-296.

دانشجوی دکترا در رشته مهندسی کامپیوتر نرم افزار در دانشگاه آزاد اسلامی یاسوج است. حوزه تخصصی ایشان مهندسی نرم افزار و امنیت اطلاعات است و تاکنون دو مورد ثبت اختراع و بیش از شصت عنوان کتاب و چندین مقاله علمی از ایشان منتشر شده است.

نشانی رایانامه ایشان عبارت است از:

Torkamani@itmc.ir



عباس دهقانی کارشناسی مهندسی کامپیوتر را در سال ۱۳۸۰ از دانشگاه شیراز، کارشناسی ارشد و دکترای مهندسی معماری سامانه‌های رایانه‌ای را به ترتیب در سال‌های ۱۳۸۷ و ۱۳۹۴ از دانشگاه اصفهان دریافت کرده است. وی هم‌اکنون عضو هیأت علمی گروه مهندسی برق و کامپیوتر دانشگاه یاسوج است. حوزه‌های پژوهشی مورد علاقه ایشان شامل معماری رایانه، طراحی سامانه‌های روی تراشه و شبکه‌های بی‌سیم روی تراشه است.

نشانی رایانامه ایشان عبارت است از:

Dehghani@yu.ac.ir

Requirements Engineering,” *IEEE Access*, vol. 6, pp. 28611–28631, 2018.
DOI: 10.1109/ACCESS.2018.2840322

- [22] C. Gonzalez and E. Liñan, “A Software Engineering Methodology for Developing Secure Obfuscated Software,” Springer, Cham, 2020, pp. 1069–1078. DOI: https://doi.org/10.1007/978-3-030-12385-7_72
- [23] S. K. Jha and R. K. Mishra, “Predicting and Accessing Security Features into Component-Based Software Development: A Critical Survey,” Springer, Singapore, 2019, pp. 287–294. DOI: https://doi.org/10.1007/978-981-10-8848-3_28
- [24] P. Morrison, D. Moye, R. Pandita, and L. Williams, “Mapping the field of software life cycle security metrics,” *Information and Software Technology*, vol. 102, pp. 146–159, Oct. 2018. DOI: <https://doi.org/10.1016/j.infsoc.2018.05.011>
- [25] H. Maleki, A. Jamshidi, and M. Mohammadi, “A Framework for Effective Exception Handling in Software Requirements Phase,” Springer, Singapore, 2019, pp. 397–411. DOI: https://doi.org/10.1007/978-981-10-8672-4_30.



کیوان رحیمی‌زاده کارشناسی مهندسی کامپیوتر را در سال ۱۳۸۱ از دانشگاه شیراز، کارشناسی ارشد مهندسی شبکه‌های رایانه‌ای را در سال ۱۳۸۶ از دانشگاه صنعتی امیرکبیر و دکترای معماری سامانه‌های رایانه‌ای را در سال ۱۳۹۴ از دانشگاه علم و صنعت دریافت کرده است. وی هم‌اکنون عضو هیأت علمی گروه مهندسی برق و کامپیوتر دانشگاه یاسوج است. حوزه‌های پژوهشی مورد علاقه ایشان شامل مدل سازی و ارزیابی سامانه‌های رایانه‌ای، محاسبات ابری، تحلیل داده‌های کلان و شبکه‌های رایانه‌ای است.

نشانی رایانامه ایشان عبارت است از:

RahimiZadeh@yu.ac.ir



محمدعلی ترکمانی کارشناسی خود را در سال ۱۳۸۰ در رشته مهندسی کامپیوتر-نرم افزار از دانشگاه آزاد اسلامی واحد شیراز دریافت کرد و از آن سال تاکنون در کارخانجات مخابراتی ایران (ITMC) مشغول به کار است. وی مدرک کارشناسی ارشد خود را در رشته مهندسی کامپیوتر-نرم افزار در سال ۱۳۹۰ از دانشگاه شهید بهشتی تهران دریافت کرد و در حال حاضر