نگاشت چرخهٔ McGraw به متدولوژی RUP برای توسعهٔ نرم‌افزار امن

کیوان رحیمی‌زاده[*1]، محمدعلی ترکمانی[2] و عباس دهقانی[3]

[1و3]دانشکده فنی و مهندسی، گروه مهندسی کامپیوتر، دانشگاه یاسوج، یاسوج، ایران

[2]کارخانجات مخابراتی ایران، شیراز، ایران

چکیده

امنیت نرم‌افزار از چالش‌های مهم در توسعه نرم‌افزار است. هر روز آسیب‌پذیری‌ها و نفوذهای زیادی در نرم‌افزارهای مشهور گزارش می‌شود. همان‌طور که برای حل مشکل بحران نرم‌افزار بحث مهندسی نرم‌افزار مطرح شد، مهندسی نرم‌افزار امن در کاهش چالش‌های امنیتی نرم‌افزار مؤثر است. چرخه McGraw به‌عنوان یکی از رهیافت‌های توسعه نرم‌افزار امن تعدادی نقطه تماس امنیت نرم‌افزار را معرفی می‌کند که شامل مجموعه‌ای از دستورالعمل‌های صریح و مشخص در راستای اِعمال مهندسی امنیت در نیازمندی‌ها، معماری، طراحی، کدنویسی، اندازه‌گیری و نگهداری نرم‌افزار است. نقاط تماس امنیت نرم‌افزار برای استفاده در ساخت نرم‌افزار، مستقل از پروسه نرم‌افزاری است و به هر فرآیند تولید نرم‌افزار قابل‌اعمال است. بنابراین، می‌توان با تغییر چرخه توسعه نرم‌افزار مورد نظر و اعمال نقاط تماس، چرخه توسعه نرم‌افزار امن را ایجاد کرد. در این پژوهش، راه‌کاری برای نگاشت چرخه McGraw به متدولوژی RUP؛ به‌عنوان متدولوژی سنگین وزن توسعه نرم‌افزار؛ و تلفیق این دو متدولوژی در راستای ایجاد یک متدولوژی ساده و کارآمد برای توسعه نرم‌افزار امن (که RUPST نام دارد) ارائه و همچنین، فراورده‌های جدید RUP برای توسعه نرم‌افزار امن به تفکیک هر نظم ارائه و چهار نقش جدید نیز برای انجام فعالیت‌های مرتبط با امنیت نرم‌افزار تعریف می‌شود. راه‌کار پیشنهادی در یک پروژه واقعی در شرکت کارخانجات مخابراتی ایران مورد استفاده و ارزیابی قرار گرفت. دست‌آوردها نشان می‌دهد که بهره‌گیری و اجرای صحیح این رهیافت توسط توسعه‌دهندگان، به پیاده‌سازی و توسعه امن‌تر و مستحکم‌تر نرم‌افزار منجر می‌شود.

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

Mapping of McGraw Cycle to RUP Methodology for Secure Software Developing

Keyvan RahimiZadeh[*1], Mohamamd Ali Torkamani[2] & Abbas Dehghani[3]

[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

*نویسندهٔ عهده‌دار مکاتبات

• تاریخ ارسال مقاله: ۱۳۹۷/۰۷/۳۰ • تاریخ پذیرش: ۱۳۹۸/۰۶/۱۱ • تاریخ انتشار: ۱۳۹۹/۰۶/۲۴ • نوع مطالعه: پژوهشی

سال ۱۳۹۹ شمارهٔ ۲ پیاپی ۴۴

[DOI: 10.29252/jsdp.17.2.46]

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

۱- مقدمه

امروزه نرم‌افزار به بخش جدایی‌ناپذیر از زندگی بشر تبدیل شده است. چرخه حیات تولید نرم‌افزار، از موضوعات مهم در تولید یک سامانه نرم‌افزاری است. برنامه‌ریزی و هدف‌گذاری، تحلیل، تولید، آزمایش، استقرار و نگهداری سامانه‌های نرم‌افزاری از گام‌های مهم این چرخه هستند [1]. با توجه به فراگیری نرم‌افزارها، فراهم‌سازی امنیت، قابلیت اطمینان نرم‌افزار و کارایی بسیار مهم و در مواردی امری حیاتی است. در این راه کاهش خطاهای نرم‌افزار نقش مهمی را ایفا می‌کند [2].

امنیت برنامه‌های کاربردی و امنیت شبکه‌های رایانه‌ای به‌تنهایی نمی‌توانند امنیت مورد نیاز کاربران را برآورده سازد. امنیت برنامه‌های کاربردی به معنی حفاظت از نرم‌افزار و سامانه‌ای که نرم‌افزار پس از ساخت و توسعه روی آن اجرا می‌شود، است نرم‌افزارها با چالش‌های هم‌چون، جداسازی (ایزوله) کد در جعبه شنی[1] (برای نمونه ماشین‌مجازی جاوا)، حفاظت در قبال کد مخرب، قفل‌کردن برنامه‌های اجرایی، پایش برنامه‌ها به هنگام اجرا (به‌خصوص ورودی برنامه‌ها)، اعمال سیاست‌های استفاده از نرم‌افزار و سامانه‌های قابل توسعه روبه‌رو هستند.

هرگاه نرم‌افزار از درون آسیب‌پذیر باشد، هیچ راه‌کاری به‌جز مسدودکردن و عدم استفاده از نرم‌افزار وجود ندارد. راه‌حل اصلی رهایی از این مشکل، مهندسی نرم‌افزار امن است. در مهندسی نرم‌افزار امن، از نخستین گام تا گام پایانی توسعه نرم‌افزار، امنیت نرم‌افزار در نظر اعمال می‌شود و ریسک‌های ممکن نرم‌افزار مدیریت می‌شوند. در این راستا، با استفاده از ابزارهای مناسب آزمون‌های امنیتی مختلف را روی سامانه انجام می‌شود و کدهای نرم‌افزاری نیز از نظر امنیتی مرور شده تا برنامه‌ها عاری از آسیب‌پذیری باشند و نرم‌افزاری امن و قابل اطمینان تولید شود.

[1] Sandboxing

برای حل مشکل امنیت نرم‌افزار، متخصصان مهندسی نرم‌افزار متدولوژی‌های گوناگونی را برای توسعه نرم‌افزار امن ارائه کرده‌اند که می‌توان به چرخه McGraw [1, 3, 4] و چرخه توسعه امنیت نرم‌افزار مایکروسافت (SDL)[2] [5, 6, 7] به‌عنوان مهمترین متدولوژی‌ها اشاره کرد. McGraw چرخه پیشنهادی خود را روی مدل خطی توسعه نرم‌افزار ارائه داده است. اما هدف از ارائه این چرخه آن است که برای تمامی متدولوژی‌های توسعه نرم‌افزار قابل پیاده‌سازی باشد. درواقع هدف ارائه‌دهنده آن، گری مک گرو، این بوده است که چرخه McGraw مستقل از متدولوژی باشد و روی هر متدولوژی قابل اعمال باشد. سایر راه‌حل‌ها به‌طورکامل اختصاصی هستند و فقط برای یک متدولوژی خاص ارائه شده‌اند. از سوی دیگر، متدولوژی RUP یک متدولوژی شناخته‌شده و محبوب برای توسعه نرم‌افزار است و بسیاری از شرکت‌ها از آن برای توسعه نرم‌افزار استفاده می‌کنند؛ اما این متدولوژی از ابتدا با ذهنیت امنیت طراحی نشده است. بنابراین، باید دستورالعمل‌های صریح و مشخص در راستای اِعمال مهندسی امنیت در سرتاسر چرخه توسعه نرم‌افزار به RUP اضافه شود. این دستورالعمل‌ها که بر اساس بهترین تجربیات در حوزه مهندسی نرم‌افزار امن هستند، شامل تعدادی فعالیت اضافی‌اند که می‌بایست توسط تیم توسعه نرم‌افزار انجام شوند. در پژوهش‌های پیشین راه‌کاری در خصوص اعمال نقاط امنیت نرم‌افزار روی RUP انجام نشده است. در این پژوهش، یک مدل برای نگاشت چرخه McGraw به متدولوژی RUP ارائه می‌شود. همچنین فراورده‌های[3] جدید RUP برای توسعه نرم‌افزار امن به تفکیک هر نظم ارائه و چهار نقش جدید نیز برای انجام فعالیت‌های مرتبط با امنیت نرم‌افزار تعریف می‌شود.

در مدل ارائه‌شده، نقاط تماس امنیت نرم‌افزار را که در چرخه McGraw معرفی و به مدل آبشاری توسعه نرم‌افزار

[2] Security development lifecycle
[3] Artifacts

[DOI: 10.29252/jsdp.17.2.46]

اضافه شده است، به متدولوژی RUP اعمال می‌شود. استفاده از این مدل، که RUPST[1] نامیده می‌شود، به توسعه‌دهندگان کمک می‌کند تا نرم‌افزاری امن‌تری تولید کنند. این مدل در مقایسه با پژوهش‌های پیشین که سعی در امن‌کردن متدولوژی RUP داشته‌اند و بیشتر به تحلیل و طراحی نیازمندی‌های نرم‌افزار توجه کرده‌اند، بسیار جامع است و تمامی گام‌ها و مراحل توسعه نرم‌افزار را دربر می‌گیرد. از آنجا که اصول مهندسی نرم‌افزار امن در تمام مراحل این مدل در نظر گرفته شده است، نرم‌افزاری که به این روش توسعه یابد، از جنبه‌هایی همانند معماری، طراحی و پیاده‌سازی امن خواهد بود. مدل پیشنهادی RUPST در یک پروژه واقعی مورد استفاده قرار گرفته است. نتایج ارزیابی این مدل نشان می‌دهد که بهره‌گیری و اجرای صحیح این رهیافت توسط توسعه‌دهندگان، به پیاده‌سازی و توسعه امن‌تر و مستحکم‌تر نرم‌افزار منجر می‌شود.

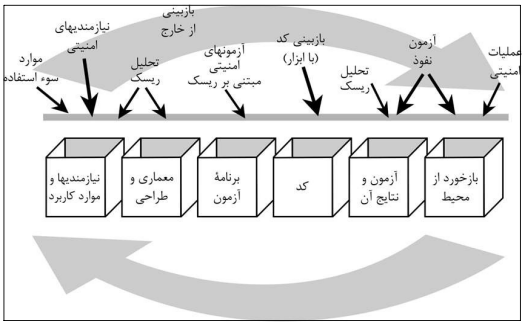
ساختار ادامه مقاله به این ترتیب است که در بخش دوم، چرخه McGraw برای توسعه نرم‌افزار امن بررسی می‌شود. در بخش سوم، متدولوژی RUP، که یکی از متدولوژی‌های مهم توسعه نرم‌افزار است به‌اختصار بیان می‌شود. در بخش چهارم، پژوهش‌های پیشین بررسی می‌شود. در بخش پنجم، یک مدل برای نگاشت چرخه McGraw به متدولوژی RUP در راستای ایجاد یک متدولوژی ساده و کارآمد برای توسعه نرم‌افزار امن (RUPST) ارائه می‌شود و بخش ششم به یک مطالعه موردی درخصوص استفاده از RUPST در یک پروژه واقعی اختصاص دارد. در بخش هفتم مزایای مدل و به‌دنبال آن در بخش هشتم مقایسه روش پیشنهادی با سایر روش‌ها از جنبه‌های مختلف عرضه و در بخش پایانی نیز دست‌آوردهای این پژوهش ارائه می‌شود.

۲- چرخهٔ McGraw

چرخه McGraw (شکل (۱)) نخستین‌بار در مجله امنیت و محرمانگی IEEE Security & Privacy در سال ۲۰۰۴ ارائه شد. این مدل صرف‌نظر از اینکه از کدام یک از فرایندهای پایه توسعه نرم‌افزار همانند حلزونی، توسعه مبتنی بر مؤلفه، RUP، XP، Agile استفاده می‌شود، قابل‌استفاده است. تعریف چرخه توسعه امن دلخواه با اضافه‌کردن یک مجموعه نقاط تماس امنیت نرم‌افزار[2] به فرآیند موجود امکان‌پذیر است؛ بنابراین، می‌توان با تغییر چرخه توسعه نرم‌افزار (SDLC[3])، چنانچه نقاط تماس را داشته باشد، چرخه توسعه نرم‌افزار امن را ایجاد کرد [4].

نقاط تماس امنیت نرم‌افزار، مجموعه‌ای از بهترین تجارب امنیت نرم‌افزار است و شامل دستورالعمل‌های صریح و مشخص به‌منظور دخیل‌کردن مهندسی امنیت در نیازمندی‌ها، معماری، طراحی، کدنویسی، اندازه‌گیری و نگهداری نرم‌افزار است. این دستورالعمل‌ها می‌بایست در قالب مراحل و نقاط مشخص و قابل‌لمس معرفی شوند. این مراحل به هر فرآیند تولید نرم‌افزار قابل اعمال هستند. به عبارت دیگر، نقاط تماس امنیت نرم‌افزار، مستقل از فرآیند توسعه‌ای که به کار برده شده است، طراحی‌شده‌اند. نقاط تماس تمامی مراحل چرخه حیات نرم‌افزار را شامل می‌شود و هدف آن، اتخاذ مجموعه‌ای ساده و روشن از بهترین تجارب مهندسی که برای ورود امنیت به فرآیند توسعه نرم‌افزار طراحی‌شده‌اند، است.

نقاط تماس هم‌زمان با تکامل نرم‌افزار بیش از یک‌بار تکرار می‌شوند و شامل سازوکارهای امنیتی (همانند کنترل دسترسی) و طراحی برای امنیت است. این سازوکارها هم شامل فعالیت‌های کلاه‌سیاه (مخرب و مربوط به نفوذگر) و هم فعالیت‌های کلاه‌سفید (سازنده و مربوط به توسعه‌دهنده) است. طبیعتاً، برخی نقاط تماس قدرتمندتر از سایرین‌اند و اولویت بیشتری دارند. نقاط تماس به‌ترتیب اولویت عبارتند از مرور کد، تحلیل ریسک‌های معماری، آزمون نفوذ، آزمون‌های امنیتی مبتنی بر ریسک، سناریوهای سوء موارد کاربری، نیازمندی‌های امنیتی، عملیات (اپراتوری) امنیتی [3]. در ادامه اولویت‌های نقاط تماس مورد بررسی قرار می‌گیرند.



(شکل-۱): چرخه McGraw [3]
(Figure-1): McGraw Cycle [3]

۲-۱- مرور کد

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

[1] RUP with Security Touchpoints
[2] Software security touchpoints
[3] Software Development Life Cycle

[DOI: 10.29252/jsdp.17.2.46]

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 پیداکردن 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:

- افراز 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[1] 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.

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.

۲-۴- آزمون امنیتی مبتنی بر ریسک (ریسک مبنا)

براساس شکل (۱)، جایگاه آزمون امنیت بر مبنای ریسک در مرحله آزمون و نتایج آن است. آزمون امنیت، فعالیتی فراتر از قلمرو پویش پورت شبکه است که به‌وسیلهٔ بسیاری از نرم‌افزارهای آزمون نفوذ انجام می‌شود. اگر آزمون امنیتی مبتنی بر ریسک به‌درستی انجام شود، تأثیر آن بسیار دقیق‌تر و عمیق‌تر از آزمون‌های جعبهٔ سیاه همانند آزمون نفوذ و جعبهٔ 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 پیشران‌های 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 ره‌یافت 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 فرایند یک‌پارچه‌سازی[2] 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 که حمله روی آن موفق می‌شود، تمرکز و ریسک‌ها را

[1] Penetration testing

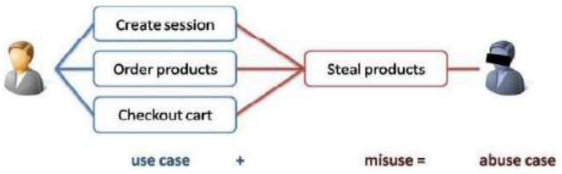
[2] Integration

[DOI: 10.29252/jsdp.17.2.46]

شناسایی کند. از نظر تئوری، اگر هر مؤلفه به‌صورت امن پیاده‌سازی و معیارهای طراحی بین مؤلفه برآورده شده باشد، سامانه‌ای که از کنار هم قرارگرفتن این مؤلفه‌ها به‌دست می‌آید، نیز به شکل امن خواهد بود؛ اما آزمون امنیتی باید در سطح سامانه نیز ادامه بیابد و سطح تجمیع‌شده را نیز در برگیرد (نقطه تلاقی آزمون نفوذ و آزمون امنیتی). بنابراین، در ادامه باید آزمون امنیت در سطح سامانه انجام شود و خواص نرم‌افزار مجتمع‌شده نیز ارزیابی شود. این نقطه به‌طوردقیق همان‌جایی است که آزمون نفوذ و آزمون امنیت با یکدیگر هم‌پوشانی دارند.

۲-۵- نمودار سوء موارد کاربری

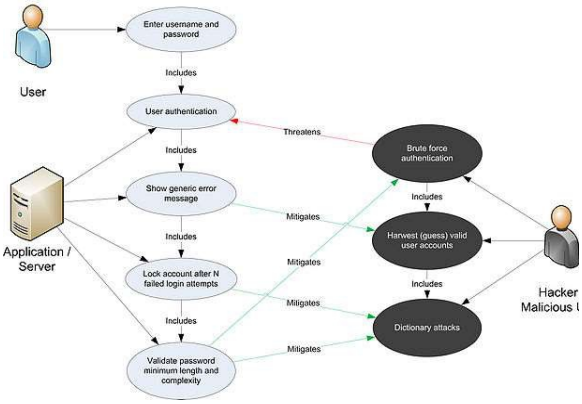
نمودار موارد کاربری[1] در UML برای مدل‌سازی کارکردهای برنامه و آنچه برنامه باید انجام دهد استفاده می‌شود. از نمودار سوء موارد کاربری[2] برای مواردی که نمی‌خواهیم در برنامه اتفاق بیافتد، استفاده می‌کنیم؛ درواقع، نمودار موارد کاربری نشان‌دهنده مواردی است که کاربر انجام می‌دهد اما نمودار سوء موارد کاربری مواردی را نشان می‌دهد که حمله‌کننده مایل است انجام دهد؛ بنابراین موارد سوء استفاده[3]، شبیه سناریوی کاربری هستند، با این تفاوت که آنها رفتار برنامه را در شرایط حمله نشان می‌دهند. 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؟ همان‌طور که در شکل (۲) دیده می‌شود، Abuse case از دو قسمت use case و misuse case تشکیل‌شده است. شکل‌های (۳) و (۴) دو نمونه نمودار موارد کاربری را نشان می‌دهد.



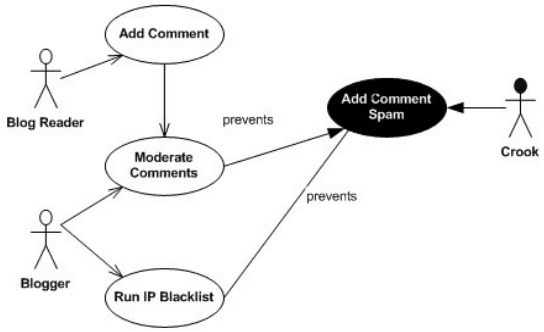
(شکل-۲): اجزای نمودار سوء موارد کاربری
(Figure-2): Abuse cases diagram

۲-۶- نیازمندی‌های امنیتی

نیازمندی‌های امنیتی سامانه باید با دقت و به‌طور کامل مدل‌سازی شوند. مدل‌سازی نیازمندی‌های امنیتی نرم‌افزار با UMLSec در فازهای تحلیل و طراحی یکی از فعالیت‌ها در توسعه نرم‌افزار امن است. UMLSec ابتدا توسط جرجینس و همکارانش [5] در سال ۲۰۰۲ ارائه شد؛ سپس شرح کامل آن برای مدل‌سازی سامانه‌هایی که در آنها امنیت یک مسألهٔ بحرانی است در [6] ارائه شد. درواقع، UMLsec بسطی از UML است که امکان نمایش اطلاعات مربوط به امنیت را درون نمودارهای ترسیم‌شده برحسب مشخصات سامانه فراهم می‌کند.



(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 می‌شود. بنابراین نیاز به فراهم‌آوری محیطی مناسب در سطح شبکه است تا سیستم‌عامل و البته تنظیمات درست امنیتی خود برنامه کاربردی فراهم شود. دیگر نقاط قابل‌طرح در این گام عبارتند از ثبت‌کردن مرور امنیتی نهایی، مدیریت تغییرات (CAB[4]) و برنامه رصد امنیتی و پاسخ‌گویی.

[1] Use case
[2] Abuse cases diagram
[3] Abuse case
[4] Change Advisory Board

[DOI: 10.29252/jsdp.17.2.46]

۳- متدولوژی RUP

در این بخش متدلوژی 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[1]، که در آن محدوده پروژه مشخص می‌شود. در گام تشریح[2]، برنامه‌ریزی پروژه برای دست‌یابی به معماری پایا[3] 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، RUP فرایندی دوبعدی 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:

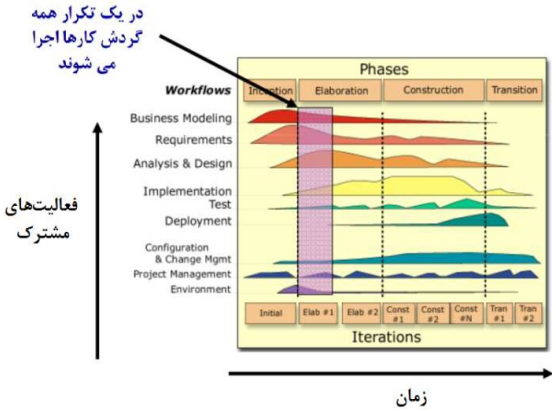
- جنبه‌های کیفی، مهندسی و روش‌های طراحی.
- فعالیت‌ها، نظم‌ها، خروجی‌ها و نقش‌ها.

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 نشان‌دهندهٔ گردش کار است. در این شکل، به فعالیت‌هایی که زیر واژه " work flows" نوشته‌شده (مانند Business Modeling، Requirements، Analysis & Design و...) نظم می‌گویند. نظم عبارت است از مجموعه‌ای از فعالیت‌های مرتبط که به یکی از نواحی مهم پروژه وابسته است. نواحی مهم اشاره‌ای به مراحل سنتی فرآیند 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 فراورده 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 فرآیند تولید نرم‌افزار، ایجاد، استفاده، یا به‌روزرسانی می‌شوند.

[1] Inception
[2] Elaboration
[3] Architecture baseline

[DOI: 10.29252/jsdp.17.2.46]

در RUP فرآورده‌ها در نه مجموعه طبقه‌بندی می‌شوند که در جدول (۱) نمایش داده شده‌اند. درروش توسعه افزایشی این نه مجموعه در طی چرخه توسعه نرم‌افزار تکامل می‌یابد [1].

(جدول-۱): دسته‌بندی فرآورده‌ها در RUP [1]
(Table-1): Artifacts in RUP [1]

نظم	فرآورده
مدل‌سازی تجاری	مدل موارد کاربری تجاری[1]، مستند معماری تجاری، مستند دور نمای تجاری
نیازمندی‌ها	مستند دورنما، نیازمندی‌ها (شامل نیازهای ذینفعان، مدل موارد کاربری و مشخصات تکمیلی)، برنامه مدیریت نیازمندی‌ها
تحلیل و طراحی	مدل تحلیل، مستند معماری نرم‌افزار، مدل طراحی، مدل استقرار
پیاده‌سازی	کد منبع و فایل‌های اجرایی، فایل‌های داده‌ای موردنیاز
آزمایش	برنامه آزمایش، روال آزمایش، مدل آزمایش ، داده‌های آزمایش
استقرار	برنامه استقرار، محصول نهائی، مستندات کاربر، مواد آموزشی
مدیریت پیکربندی	برنامه مدیریت پیکربندی، مستند درخواست تغییر
مدیریت پروژه	فرآورده‌های برنامه‌ریزی (مانند برنامه توسعه نرم‌افزار، نقشه تکرار، فهرست ریسک‌ها)، فرآورده‌های عملکردی مانند توصیف نشرها، تشخیص وضعیت پروژه و....
محیط	مورد توسعه، راهنمایی‌های مدل‌سازی تجاری، راهنمایی‌های طراحی

در RUP هر فرد می‌تواند چند نقش را ایفا کند و چند نفر می‌توانند یک نقش را ایفا کنند. هم افراد متخصص و هم بهره‌برداران سامانه جزء نقش‌های RUP هستند. برخی از مهم‌ترین نقش‌های RUP عبارتند از: تحلیل‌گر فرآیند کاری[2]، بهره‌برداران، بازبین کننده تجاری، تحلیل‌گر سامانه، معمار سامانه، طراح واسط کاربر، طراح پایگاه داده، بازبین‌کننده‌های معماری و طراحی، طراح آزمایش، یک‌پارچه‌ساز سامانه، مدیر استقرار، مدیر پیکربندی، مدیر پروژه، مدیر کنترل تغییرات، مهندس فرایند و متخصص ابزار. نمونه‌ای از فعالیت‌ها در گام بازبینی طراحی یافتن موارد کاربری و عامل‌ها توسط تحلیل‌گر سامانه، بازبین‌کننده طراحی است [9].

۴- پیشینهٔ پژوهش

امنیت نرم‌افزار به‌عنوان یکی از مهم‌ترین گام‌های ساخت و توسعه نرم‌افزار است [14, 15, 16]. امروزه، بروز حملات فراوان به نرم‌افزارها، چالش‌های فراوانی را برای کاربران و فراهم‌کنندگان نرم‌افزارها به‌وجود آورده است. کاستی‌های امنیتی نرم‌افزارها از جنبه‌های مختلفی چالش برانگیز است. از یک سو، ناامن‌بودن نرم‌افزارها هزینه زیادی را برای سازندگان به همراه دارد و از سوی دیگر، تمایل کاربران به استفاده از نرم‌افزاری که تمهیدات امنیتی را در آنان مورد توجه قرار نگرفته است کاهش می‌دهد. مهندسی نرم‌افزار امن می‌تواند در راستای کاهش هزینه‌ها و افزایش شمار کاربران دارد.

پژوهش‌های زیادی در زمینه امن‌سازی نرم‌افزارها انجام گرفته است [25-19]. در [8, 9] یک مدل فرایندی برای مهندسی نیازمندی‌های امنیتی موسوم به Square RUP که تأکید آن بر تحلیل نیازمندی‌های امنیت و مدیریت ریسک و همچنین استفاده از misuse cases است، ارائه شده است. در [10] یک مدل سبک‌وزن برای امنیت نرم‌افزار به نام CLASP[3] معرفی شده است در محیط‌های RUP استفاده می‌شود. نویسندگان فعالیت‌هایی را برای مهندسی نیازمندی‌های امنیتی ارائه کرده است که مهم‌ترین فعالیت‌های آن تحلیل نیازمندی‌های امنیتی، استفاده از misuse cases، طراحی رابط کاربری برای امنیت، استفاده از اصول امنیت برای طراحی، انجام آزمون‌های امنیتی و تعیین پیکربندی‌های امن پایگاه داده است. در [11] مدلی به نام RUPSec ارائه و نظم‌های مدل‌سازی کسب کار و نیازمندی‌ها در RUP برای توسعه سامانه‌های امن توسعه داده شده است. RUPSec تعدادی فراورده جدید برای گام‌های تحلیل و طراحی همانند مدل تهدید، مدل misuse، مدل امنیتی موارد کاربری[4]، مدل misactor ارائه کرده است. همچنین، به تعریف فعالیت‌های جدید برای مهندسی نیازمندی‌های امنیتی مانند تعریف سیاست‌های امنیتی، پیداکردن تهدیدهای سامانه و موارد کاربری امنیتی به امنیت نرم‌افزار در گام‌های تحلیل و طراحی توجه کرده است. هرچند روش انجام برخی از کارها به‌طوردقیق مشخص نشده است. برای نمونه، مشخص نشده از چه روشی باید برای مشخص‌کردن تهدیدهای سامانه استفاده شود. بیشتر اینکه، درخصوص مدل‌سازی تهدیدها نیز بحث نشده است. نویسندگان در [12] فعالیت‌های امنیتی مورد نیاز را در نظم نیازمندی‌های RUP اضافه کرده‌اند. همچنین، یک نقش با عنوان خبره امنیت[5] به نقش‌های RUP اضافه شده و به مدل‌سازی نیازمندی‌ها با استفاده misuse case توجه شده است. نویسندگان در [13] یک مدل توسعه نرم‌افزار امن

[1] Business usecase model
[2] Business process analyst
[3] Comprehensive, lightweight application security process
[4] Security use case model
[5] Security expert role

[DOI: 10.29252/jsdp.17.2.46]

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[1] و توسعه‌دهنده امنیت[2] درنظر 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 اِعمال 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 آبشاری 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 درنظر 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 متدولوژیRUP مشخص می‌شود. نتیجه این کار در جدول (۲) نشان داده شده است. بر اساس جدول (۲) 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. همان‌طورکه 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

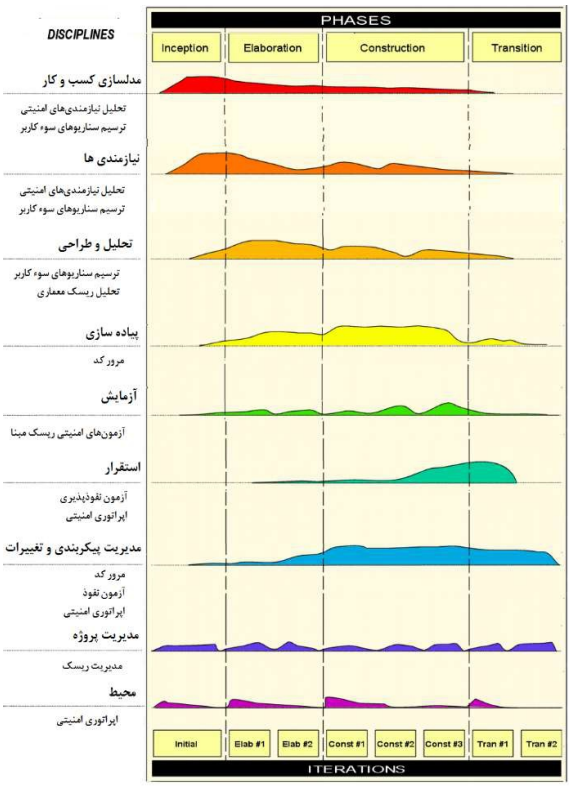
نظم های RUP	فعالیت چرخه McGraw
مدل‌سازی تجاری	تحلیل نیازمندی‌های امنیتی ترسیم سناریوهای سوء موارد کاربری
نیازمندی‌ها	تحلیل نیازمندی‌های امنیتی ترسیم سناریوهای سوء موارد کاربری
تحلیل و طراحی	ترسیم سناریوهای سوء موارد کاربری تحلیل ریسک معماری
پیاده‌سازی	مرور کد
آزمایش	آزمون‌های امنیتی ریسک مبنا
استقرار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

تحلیل نیازمندی‌های امنیتی یکی از فعالیت‌های کلان در چرخه McGraw است و می‌تواند شامل فعالیت‌هایی مانند مشخص‌کردن 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

[1] Security analyst
[2] Security developer

[DOI: 10.29252/jsdp.17.2.46]

یا مرور کد انجام می‌شود، تا آسیب‌پذیری‌های موجود در کد برنامه مشخص شود. درصورتی که برنامه‌نویس‌ها از کتابخانه‌های ناامن یا منسوخ استفاده کرده باشند یا اصول کدنویسی امن توسط تیم توسعه رعایت نشده باشد، در این نظم آسیب‌پذیری مشخص خواهد شد. بسیاری از حملات سایبری رایج نظیر سرریز بافر، XSS و تزریق SQL ناشی از کدهای ضعیف و آسیب‌پذیر هستند که می‌توان از آنها به وسیله فعالیت مرور کد پیش‌گیری کرد. در نظم آزمایش، آزمون‌های امنیتی ریسک مبنا انجام می‌شود. آزمون‌های امنیتی بر اساس الگوی حملات، نتایج تحلیل ریسک و سناریوهای سوء موارد کاربری که در نظم‌های قبل تهیه شده است، انجام می‌شود. در نظم استقرار نیز دو فعالیت مهم انجام می‌شود که عبارتند از آزمون نفوذ و اپراتوری امنیتی. در آزمون نفوذ نرم‌افزار با رویکردی از بیرون به درون مورد ارزیابی امنیتی قرار می‌گیرد تا آسیب‌پذیری‌ها و حفره‌های امنیتی سامانه مشخص شود. این کار با استفاده از نرم‌افزارهایی نظیر Acunetix،Meta sploit و Kali linux انجام می‌شود.



(شکل-۶): فعالیت‌های پیشنهادی در راستای تولید و توسعه نرم‌افزار امن در متدولوژی RUP

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

(جدول-۳): فراورده‌های جدید RUP برای توسعه نرم‌افزار امن به‌تفکیک نظم

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

نظم	فراورده جدید
مدل‌سازی تجاری	مدل سوء موارد کاربری تجاری
نیازمندی‌ها	مدل سوء موارد کاربری ، برنامه مدیریت نیازمندی‌های امنیتی
تحلیل و طراحی	مدل تحلیل ریسک های امنیت، مستند معماری نرم‌افزار امن، مدل های تحلیل و طراحی امن با UMLSec، مدل استقرار امن
پیاده‌سازی	گزارش مرور کد
آزمایش	برنامه آزمون‌های امنیتی، روال آزمون‌های امنیتی، مدل آزمون‌های امنیتی، داده‌های آزمایش امنیتی
استقرار	گزارش آزمون نفوذ، مستند نصب و پیکربندی امن نرم‌افزار
مدیریت پیکربندی	گزارش آزمون نفوذ، گزارش مرور کد
مدیریت پروژه	مستند مدیریت ریسک‌های امنیتی
محیط	مستند نصب و پیکربندی امنیتی ابزارها

(جدول-۴): نقش‌های جدید برای انجام فعالیت‌های مرتبط با توسعه نرم‌افزار امن در RUP

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

نقش جدید	وظیفه
تحلیل‌گر و معمار امنیت[1]	مشخص کردن مکانیزم‌های امنیتی مورد نیاز در پروژه، تحلیل نیازمندی‌های امنیتی، تحلیل ریسک‌های امنیتی، ترسیم نمودارهای Abuse case، ترسیم درخت حمله، مدل‌سازی تهدیدها به روش STRIDE
طراح نیازهای امنیتی[1]	ترسیم نمودارهای Abuse case، ترسیم نمودارهای UMLSec برای نیازمندی‌های امنیتی سیستم، تعریف موارد آزمون امنیتی[1]
مرورکننده کد[1]	مرور کد با استفاده از ابزار و مشخص کردن آسیب‌پذیری‌ها و مشکلات امنیتی موجود در کد.
مسئول آزمون نفوذ [1]	انجام آزمون نفوذ با استفاده از ابزارهای مختلف، ارائه گزارش آزمون نفوذ

در اپراتوری امنیتی نیز تنظیمات و پیکربندی امن نرم‌افزار انجام می‌شود. بخش‌هایی از نرم‌افزار که استفاده از آنها خطرناک است و نیاز به دقت دارد، نباید به‌صورت پیش‌فرض فعال باشد. به‌عنوان مثال در گوشی‌های تلفن همراه خاصیت Developer یا Root گوشی به‌صورت پیش‌فرض غیرفعال است. تنظیمات پیش‌فرض محافظه‌کارانه، اجتناب از تغییرات پیش‌فرض خطرناک و غیرفعال‌کردن خدماتی که کمتر به‌کار برده می‌شوند، اصول مهمی در مهندسی نرم‌افزار امن هستند که باید مورد توجه قرار گیرند [6, 7, 18]. در اپراتوری امنیتی همچنین راهنمای

[DOI: 10.29252/jsdp.17.2.46]

استقرار امن نرم‌افزار و ابزارهای استقرار وصله[1] 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- مطالعهٔ موردی: استفاده از RUPST در یک پروژهٔ 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 علاوه‌بر 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 علاوه‌بر آزمون‌های عملکردی رایج در مهندسی 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 کاربری بر اساس سناریوهای مختلف انجام شد. فهرست برخی از ابزارهای استفاده‌شده در این پروژه در جدول (۵) ارائه شده است.

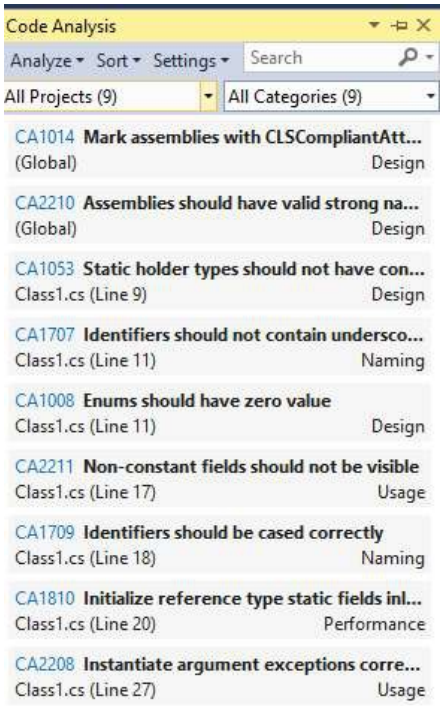
(جدول-۵): ابزارهای استفاده شده در این پروژه
(Table-5): Tools for evaluation the RUPST

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 نمایی اجرا شوند که این موضوع می‌تواند برای اکسپلویت‌کردن و منع سرویس استفاده شود. نرم‌افزار Regex وجود این آسیب‌پذیری اجرایی نمایی 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 ارتقاع 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 ردیافت 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

[1] Patch deployment

[DOI: 10.29252/jsdp.17.2.46]

شکل (۷) نمونه ابزار تحلیل کد در ویژوال استودیو را نشان می‌دهد. همان‌طورکه ملاحظه می‌شود، این ابزار آسیب‌پذیری‌هایی را در نرم‌افزار یافته است و رهنمودهایی را نیز به منظور رفع آسیب‌پذیری‌های امنیتی کد برنامه ارائه می‌دهد. در نظم استقرار، پیکربندی نرم‌افزار بر اساس اصول امنیت اطلاعات نظیر تنظیمات پیش‌فرض محافظه‌کاران، اجتناب از تغییرات پیش‌فرض خطرناک و غیرفعال‌کردن خدماتی که کمتر به‌کاربرده می‌شوند، توسط پیش‌فرض که درقبل 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 می‌دهد. استفاده از این رهیافت در توسعه از پورتال کارخانجات مخابراتی ایران موجب شده است که بسیاری از خطاهای امنیتی در گام‌های ابتدایی توسعه نرم‌افزار و قبل از مراحل کدنویسی کشف شوند که این موضوع علاوه‌بر افزایش امنیت سامانه، باعث کاهش هزینه‌های توسعه نرم‌افزار شده است. بنابراین استفاده از این متدولوژی باعث جلوگیری از آسیب‌پذیری در کدهای نرم‌افزار و امنیت بیشتر سامانه شده است.

(جدول-۶): برخی از خطاهای کشف شده در اجرای راه‌کار پیشنهادی در توسعه بخشی از پورتال کارخانجات مخابراتی ایران
(Table-6): Reported bugs of implementing RUPST in ITMC

محل خطا	عنوان
معماری و طراحی نرم‌افزار	۱- عدم رمزنگاری اطلاعات در پایگاه داده SQl Server در سمت سرویس دهنده ۲- استفاده از یک معماری قدیمی برای توسعه نرم‌افزار ۳- درنظر نگرفتن مباحث امنیتی در طراحی اولیه
کدنویسی	۱- استفاده از یک کتابخانه شخص ثالث که دارای آسیب‌پذیری است. ۲- عدم اعتبارسنجی در ورودی یکی از صفحات طراحی‌شده ۳- امکان انجام حمله SQL Injection روی یکی از صفحات طراحی‌شده ۴- تعریف نامناسب متغیرهای یکی ازرده‌ها ۵- وجود آسیب‌پذیری در اسمبلی‌ها ۶- نام‌گذاری نامناسب ۷- خطای امنیتی در Class Library
آزمون امنیتی مبتنی بر ریسک	۱- عدم کارکرد مناسب پنجره تعیین رمز عبور (پذیرش رمزهای ضعیف به‌عنوان رمز عبور کاربران) ۲- کندبودن تابع رمزنگاری روی برخی از کلاینت‌هایی که از نظر قدرت پردازشی ضعیف‌تر هستند ۳- نشت داده‌ها از سطوح دسترسی بالاتر به پایین‌تر
آزمون نفوذ	۱- یافتن سه نوع آسیب‌پذیری در سامانه در ارتباط با وجود پورت‌های باز و حمله XSS

۷- مزایای مدل پیشنهادی (RUPST)

به‌اختصار، برخی از مزایای مدل پیشنهادی عبارتند از:

۱- درنظرگرفتن اصول مهندسی نرم‌افزار امن در تمامی مراحل توسعه نرم‌افزار.

۲- امکان به‌کارگیری روش‌های مدل‌سازی امنیت نظیر UMLSec در کنار نمودارهای سنتی UML در RUP.

۳- مبتنی‌بودن بر سناریوهای سوء موارد کاربری و مدل‌سازی تهدیدها به روش STRIDE.

۴- دارای نقش‌های مناسب برای فعالیت‌های امنیتی.

۵- دارای فراورده‌های مورد نیاز در توسعه نرم‌افزار امن.

۶- کارآمدی، سادگی و سهولت استفاده بر اساس مطالعه موردی.

۷- یافتن خطاها و نقص‌های امنیتی در ابتدای پروژه.

۸- قابلیت توسعه.

۹- عدم وابستگی به ابزارهای امنیتی خاص.

[DOI: 10.29252/jsdp.17.2.46]

۸- مقایسه ره‌یافت پیشنهادی با سایر روش‌ها

جدول (۷) مقایسه RUPST با تعدادی از روش‌های دیگر را نشان می‌دهد. این مقایسه بر اساس هفده عامل مهم انجام شده است. بیشتر جنبه‌های مقایسه ذکرشده در این جدول جزء بهترین تجربیات در مهندسی نرم‌افزار امن هستند. بر اساس این مقایسه، روش پیشنهادی از نظر فرایند و متدولوژی توسعه و چارچوب کاری از روش‌های دیگر جامع‌تر است.

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

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

جنبه مقایسه	McGraw	SDL	[11,12]	[13]	[14]	[15]	[17]	RUPST
تحلیل نیازمندی‌های امنیتی در فازهای تحلیل و طراحی و معماری	*	❊	❊	❊	*	❊	*	❊
مدل‌سازی UMLSec	❊							*
مدلMisuse	*	❊	❊	❊			*	❊
استفاده از نمودارسوء 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 فراورده‌های جدید برای امنیت	*	*	*	*	*	❊	*	❊
تاکید بر آموزش امنیت		❊						❊
ورود بحث حریم خصوصی در کنار امنیت به‌صورت مجزا		❊						
تطبیق با فازهای RUP			❊	❊	*	❊	*	❊

۹- نتیجه‌گیری

در این پژوهش، راه‌کاری نوین برای توسعه نرم‌افزار امن با بهره‌گیری از نگاشت چرخه McGraw به متدولوژی RUP ارائه شد. در مدل ارائه‌شده 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. علاوه‌براین، 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 درنظر گرفته شده است، نرم‌افزاری که به 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.

[DOI: 10.29252/jsdp.17.2.46]

10- References ۱۰- مراجع

[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 نرم‌افزار"، پردازش علائم و داده‌ها. 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.

[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). [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

[DOI: 10.29252/jsdp.17.2.46]

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.infsof.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 هم اکنون 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-نرم‌افزار 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-نرم‌افزار در سال ۱۳۹۰ از دانشگاه شهید بهشتی تهران دریافت کرد و در حال حاضر دانشجوی دکترا در رشته مهندسی کامپیوتر نرم‌افزار در دانشگاه آزاد اسلامی یاسوج است. حوزه تخصصی ایشان مهندسی نرم‌افزار 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 به‌ترتیب در سال‌های ۱۳۸۷و 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

[DOI: 10.29252/jsdp.17.2.46]