ششمین همایش بین‌المللی افق های نویـن در
مهندسی برق، کامپیوتـر و مکانیک
6th International Conference on the New Horizons in
Electrical Engineering, Computer and Mechanical
www.mhconf.ir

توسعه نرم افزار امن با استفاده از هوش مصنوعی

محمدجوادحسین پور[۱،*]، فتاح شاکری حقیقی[۲]،سمیه رحمانی[۳]

[۱،*]عضو هیات علمی و استادیار بخش مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد استهبان،استهبان،ایران

Email:hosseinpoor.mohammadjavad@gmail.com

[۲]دانشجوی کارشناسی ارشد مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد استهبان،استهبان،ایران

Email:fatahshakeri@gmail.com

[۳] دانشجوی کارشناسی ارشد مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد استهبان،استهبان،ایران

Email:rahmanei1361@gmail.com

چکیده

امروز توسعه نرم افزار فرآیند خیلی پیچیده و چالشهای زیادی بیش از هر زمان دیگری در آن وجود دارد که اعمال نیازهای امنیتی در توسعه امن نرم افزار بسیار مهم خواهد بود. مدل های جدید توسعه نرم افزار در حال ظهور است که در امر توسعه نرم افزار به طور پیش فرض کمک می کند. نرم افزار امن بای بر اساس چارچوب اصولی سازمان و چارچوب اصولی نرم افزار سازمان تعریف گردد به این معنی که سطح هدف گذاری شده امنیت نرم افزار های مختلف یک سازمان وابسته به زمینه کسب وکار و درجه اهمیت اطالعات در آن زمینه مشخص می شود. در این مقاله ابتدا به بررسی مراحل مختلف توسعه نرم افزار می پردازیزم و سپس چگونگی اعمال نیازمندی های امنیتی مورد نیاز در هر مرحله را با در نظر گرفتن آخرین مسایل امنیتی مورد ارزیابی و بررسی قرار خواهیم داد.درواقع اعمال زودتر ملاحظات امنیتی در طی فعالیت های مراحل توسعه نرم افزار موجب میگردد تا سیستم اطلاعاتی پیاده سازی شده دارای یک امنیت پایه ایی گردد.همچنین امروزه متدلوژی های سریع به دلیل مزایایی که دارند بسیار محبوب هستند و در پروژه های مختلف خصوصا پروژه های کوچک و متوسط بسیار استفاده میشوند اما در توسعه نرم افزار با روش های سریع به امنیت توجه کافی نشده است.

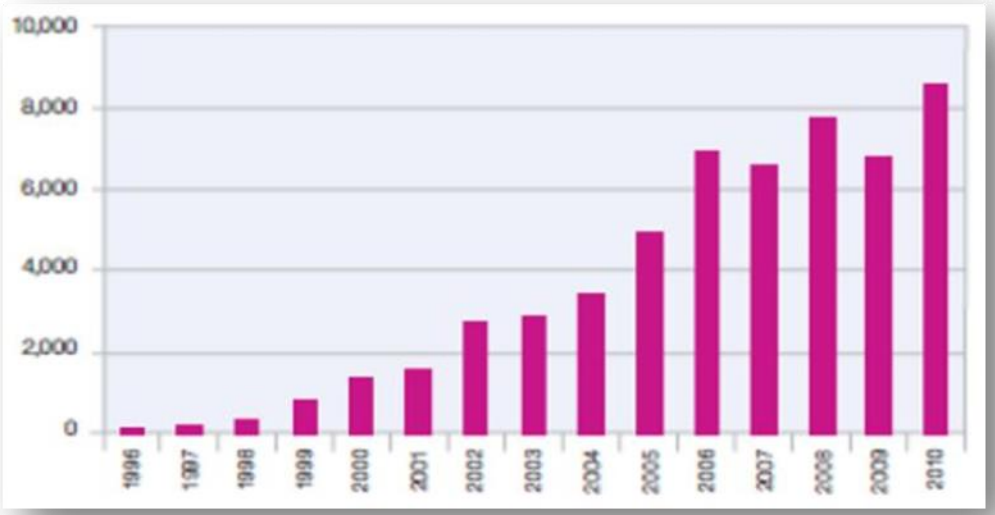
کلمات کلیدی:مهندسی نرم افزار، توسعه نرم افزار،هوش مصنوعی

۱- مقدمه

امروزه سازمان ها می دانند که باید از اطلاعات خود بخوبی محافظت کنند و همچنین انجام فعالیت های تجاری - اداری و آموزشی بدون استفاده از نرم افزار امکانپذیر نیست و نکته مهم اینکه اطلاعات حیاتی آنها بر روی بانکهای اطلاعاتی همین نرم افزار ها نگهداری می شوند. که کوچکترین خلل امنیتی در این نرم افزار ها می تواند این شرکت ها و سازمانها را در فضای رقابتی و پایداری درتجارت به نابودی بکشاند حال جایگاه امنیت در نرم افزار بخوبی روشن می شود نرم افزار امن تولید نخواهد شد مگر در قالب یک فرایند توسعه امن که مسایل امنیتی در تمامی مراحل تولید نرم افزار لحاظ شده باشد.یکی از مشکلات عمده نرم افزار ها در نظر نگرفتن نیازمندی های امنیتی و بطور کلی تمهیدات امنیتی در زمان توسعه آنها می باشد. غالباً در تولید نرم افزار های امن به میزان سطح هدف گذاری شده اطمینان هر سازمان توجه نمی شود و توسعه دهندگان به اجرای امنیت بر طبق یک سری استاندارد های مشترک و پایه اکتفا میکنند. با این وجود یک نظام و چارچوب امنیتی برای امن سازی فرایند توسعه امن نرم افزارامری حیاتی و ضروری به نظر می رسد هرچند در این زمینه فعالیتهای فراوان و خوبی صورت گرفته ولی تا هنگامی که افرادی سعی دارند با استفاده از علم خود در نرم افزار ها نفوذ کرده و ازاطلاعات آنها سوء استفاده نمایند باید تولید کنندکان نرم افزار تالش خود را در جهت ارتقا فرایند توسعه نرم افزار و در نتیجه تولید نرم افزار مطمئن مضاعف نمایند. یکی از روش های مطرح جهت افزایش امنیت سیستم های نرم افزاری روش مهندسی نیازمندی های کیفیت امنیت میباشد. این روش شیوهایی را ارائه میدهد که از طریق آن میتوان موضوعات امنیتی را هرچه زودتر در چرخه حیاط توسعه نرمافزار اعمال نمود. مهندسی نیازمندیهای کیفیت امنیت شامل نه مرحله میباشد که نیازمندی های امنیتی دسته بندی خواهد شد. قابل ذکر است این روش هرچند احتمال برای هرگونه پروژه ایی در مقیاس بزرگ قابل استفاده است اما در واقع برای ایجادسیستم های تکنولوژی اطلاعات طراحی شده است[1].امروز توسعه نرم افزار فرآیند خیلی پیچیده و چالش های زیادی بیش از هر زمان دیگری در آن وجود دارد نیازهای امنیتی در توسعه نرم افزار منجر به ایجاد چرخه زندگی به اصطلاح SSDLC شده است این مفهوم روش شناختی است که در چرخه زندگی توسعه نرم افزارکلاسیک گنجانده شده است به وسیله پنج فاز اصلی توضیح داده می شود آنالیز،.طراحی، پیاده سازی، تست، ارزیابی[1].توسعه محصول همواره با پدیده های جدید فناوری به طور مداوم در طول زمان تکامل می یابد امروزه چرخه عمر محصول PLCیا مدیریت فعالیت تجاری برای مدیریت به کارآمدترین روش محصولات شرکت در طول چرخه زندگی آنها است از همان ایده اول برای یک محصول در تمام راه تا پایان اتمام آن یکی از مهمترین بخش های فرآیند امروزه بدون شک توسعه نرم افزار است.

۲- ضرورت امن سازی فرایند توسعه نرم افزار

هزینه شناسایی و مدیریت ریسک های امنیتی نرم افزار در طول چرخه توسعه آن به مراتب کمتر از انتهای کار تولید و تحویل نرم افزار است.ضمن اینکه مطابق شکل ۱ آسیب پذیری های نرم افزار در بین سال های ۱۹۹۶ تا ۲۰۱۰ میلادی را نمایش می دهد که آسیب پذیری هرسال بیشتر شده است. با این وجود واضح خواهد بود که بدون داشتن یک خط تولید نرم افزار امن امکان برون داد یک نرم افزار امن از این چرخه تولید امری نا ممکن است همچنین فضای رقابتی کسب و کار سازمان ها وابسته به نرم افزار هایی است که در این زمینه دارند لذا با توجه به میزان آسیب پذیری های گزارش شده در حوزه نرم افزار وجود یک چرخه حیات امن نرم افزار که حاصل آن تولید و توسعه نرم افزار امن می باشد بسیار ضروری است.[2]



شکل ۱ : گزارش آمار آسیب پذیری نرم افزار[2]

۳- ویژگی های نرم افزار امن

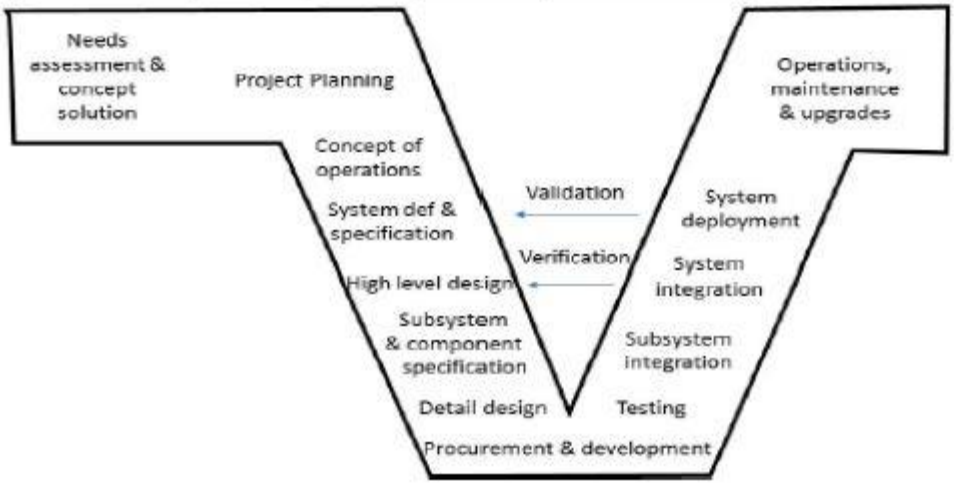
نرم افزار امن باید بتواند در شرایط بحرانی موجبات ادامه کسب و کار سازمان را فراهم و از دستبرد به اطلاعات حیاتی سازمان موجود در آن که کلید موفقیت سازمان محسوب میشود جلوگیری نماید.نبود حفره های امنیتی حساس در نرم افزار و همچنین عدم وجود نقاط آسیب پذیری کشنده در نرم افزار می تواند در حفظ تداوم کسب و کار و جلوگیری از نابودی و ورشکستگی سازمان نقشی کلیدی داشته باشد. نرم افزار امن باید از مکانیزم های خوبی در راستای سیاست ها، قوانین و چارجوب اصولی امنیت نرم افزار سازمان مانند چگونگی دسترسی پذیری، محرمانگی،یکپارچگی اطلاعات،سطوح دسترسی افراد برخوردار باشد.[3]

بنابراین نرم افزار امن در یک سازمان باید دارای ویژگی های ذیل باشد.

- محرمانگی اطلاعات را حفظ نماید
- سطوح دسترسی پذیری در آن کاملا مشخص و تعریف شده باشد
- یکپارچگی و جامعیت اطلاعات در بانک اطلاعاتی آن رعایت شود
- نرم افزار بر اساس یک متدلوژی امن تحت فرایند مدیریت امنیت نرم افزار سازمان ایجاد شده باشد
- در فرایند طراحی آن از مدل سازی تهدید بومی شده سازمان استفاده شده باشد
- سطح امنیت واقعی آن باید برابر با سطح امنیت هدف گذاری شده سازمان باشد

۳-۱- مهندسی نیازمندی های کیفیت امنیت

Phase 0	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
Concept exploration & benefits analysis	Project planning and concept of operations development	System definition & design	Systems development & implementation	Validation, operations, maintenance & upgrades	Systems retirement / replacement



در پژوهش انجام شده توسط cics8یک نقشه راه جهت توسعه پروژه هایی که موضوع امنیت در در آنها از نقش محوری برخوردار است ارائه شده است. در این راستا از فرایند یکپارچه رشنال بعنوان چهارچوب کاری استفاده شده است .همچنین روش مهندسی نیازمندی های کیفیت امنیت که حاوی توصیه ها و اقدامات امنیتی مورد نیاز در مراحل مختلف توسعه نرم افزار است را نیز بکار برده اند. این اقدامات شامل شامل روشهایی مانند تهیه سناریوهای موارد سوء استفاده ، درختهای هجوم، تحلیل ریسک و تهیه لیست ریسک های امنیتی،دسته بندی و اولویت بندی نیازمندی های امنیتی و قالبهای پیشنهادی جهت مستندسازی آنها میباشد. در واقع اعمال و بکارگیری روش مهندسی نیازمندیهای امنیتی در مراحل مختلف فرایند RUPمی تواند میزان موفقیت پروژه های نرم افزاری را به ویژه در مقوله امنیت تا حد زیادی افزایش دهد.[4] قابل ذکر است اگرچه نیازمندی های امنیتی تا حدود زیادی همان شیوه های عمومی نظیر بکارگیری رمز ورود، دیوارهای آتشین و ابزارهای ویروس یاب هستند اما آنها بطور مستقل از سایر فعالیت های مهندسی نیازمندی ها توسعه میابند و با جریان فعالیت های مرتبط با نیازمندی های یکپارچه نخواهند شد. فعالیت های امنیتی درطراحی مهندسی نرم افزار میتواند شامل موارد زیر باشد:

۲-۳- چرخه عمر توسعه سیستم

بسیاری از فرایندهای طراحی مهندسی وجود دارند که یکی از پرکاربردترین آنها چرخه عمر توسعه سیستم SDLC است تغییرات بسیاری در SDLC وجود دارد که متناسب با برخی از طرحها و پیاده سازی ها طراحی شده اند به طور کلی مراحل همچنین می تواند به صورت خطی ترنشان داده شوند.

شکل۲:چرخه عمرمهندسی سیستم های عمومی[5]

در نمودار Vکلی SDLC می توان خطی آن را همینطور در شکل ۲ میتوان مشاهده کرد بر اساس استاندارد IEEE - 1220برای کاربرد و مدیریت فرآیند مهندسی سیستم فعالیتهای زیر مربوط به فعالیت های اساسی مرتبط با هر مرحله از SDLCخطی است در شکل ۱ نشان داده شده است توجه داشته باشیم که نام های مختلفی برای هر مرحله ارائه میشود به این دلیل که در استانداردهای مختلف ازنام های مختلفی برای هر مرحله استفاده شده است جدول ۱ خالصه ای از نسخه های مختلف این جدول را نشان میدهد.

جدول ۱:فعالیت های مهندسی در SDLC

Phase	Engineering Activities
0: Concept Exploration Phase	Investigate new opportunities Explore technology readiness Conduct risk analysis Evaluate pre-concept match with users' needs
1: Requirements & Conceptualisation / Initiation Phase	Identify stakeholders' needs Evaluate alternate concepts Recommend possible solutions Define system requirements & specifications (functional, performance, operational, support etc). Define engineering project strategy.
2: Design & Development / Acquisition Phase	Develop detailed planning Design system architecture & allocate systems requirements to subsystems and components Develop detailed system design (logical & physical) Performs system synthesis analysis to assure system integration
3: Production & Implementation / Implementation / Production Phase	Produce system components & construct system Code/configure system functional components Conduct code walkthrough/unit tests Perform system integration Inspect and test (systems acceptance test)
4: Utilisation & Support / Operations / Maintenance Phase	Operate system to satisfy users' needs Provide sustained systems capability
5: Retirement / Disposition / Disposal Phase	Store, archive or dispose of system

۳-۳- ملاحظات امنیتی برای طراحی مهندسی

در مورد توسعه و استفاده از چرخه عمر توسعه سیستم امنیتی SSDLC نظریات زیادی وجود دارد اما در حال حاضر هیچ استانداردی برای SDLC وجود ندارد. جدول ۲ نشان میدهد که چگونه برخی ازکارشناسان ادغام و اجرای امنیت را در SDLC با توجه به فعالیتهای زیادی شده درSP ۸۰۰-۱۶۰ .VOL 1 و چهارچوب فنی تضمین اطلاعات AITF ارائه دادند باید توجه داشت که فعالیتهای ذکر شده جامع نیستند و فعالیت های دیگری وجود دارند ترتیب و فاز اجرایی فعالیتها نیز ممکن است در بسیاری از فریم ورک ها و فرآیندها متفاوت باشد خالصه ملاحظات امنیتی مندرج در جدول ۲ وضوح کافی در مورد انواع فعالیت های پیشنهادی برای هر مرحله از SDLC را ارائه می دهد.[5]

جدول۲:ملاحظات امنیتی در SDLC

Phase	Security Activities: SP 800-160	Security Activities: IATF
0: Concept Exploration Phase		Discover information protection needs Ascertain the system purpose
1: Requirements & Conceptualisation / Initiation Phase	Survey & understand the policies, standards, and guidelines. Define security classification & protection level Identify information assets Conduct preliminary risk assessment.	Identify information asset needs protection. Define System Security Requirements based on the protection needs.
2: Design & Development / Acquisition Phase	Conduct risk assessment Define security requirements & select security controls (categories & types) Perform cost/benefit analysis (CBA) Security planning (based on risks & CBA) Practice Information Systems Security Engineering (ISSE) process to develop security controls Develop security test & evaluation plan for verification & validation of security controls	Design System Security Architecture Design system architecture to meet on security requirements. Develop Detailed Security Design based on security architecture Design security functions & features for the system
3: Production & Implementation / Implementation / Production Phase	Implement security controls in accordance with system security plan (SSP) Perform Security Certification & Accreditation of target system.	Implement System Security Implement designed security functions and features into the syste.
4: Utilisation & Support / Operations / Maintenance Phase	Configuration management & perform change control Continuous monitoring Perform periodic security assessment	Assess Security Effectiveness & assess effectiveness of ISSE activities

۴-۳- روش توسعه نرم افزار امنیتی SECSDM

با وجود پیشنهادهای زیاد در مورد اینکه کجا و چگونه می توان امنیت را در طراحی مهندسی گنجانیده و پیچیدگی هایی فراوان در عملی بودن کار وجود دارد اما رهنمودهای شفاف و عملی ارائه نشده است مرجع ۱۱ متدولوژی SECSDM را همواره می تواند با یک ابزار نرم افزاری مربوطه توسعه داد زیرا آنها دریافتند که بسیاری از برنامه های کارشناسی فناوری اطلاعات از اهمیت دادن به ادغام امنیت اطلاعات درچرخه و نرم افزار قضاوت می کند. آنها SECSDM را به عنوان یک رویکرد عملی برای ادغام امنیت در طراحی نرم افزار میدانند و نرم افزار به صورت عملی به منظور ارائه یک رویکرد گام به گام روشن برای شناسایی کنترل امنیت مورد نیاز و چگونگی پیاده سازی عملی مکانیسم های امنیتی مربوط برای برآورده ساختن الزامات طراحی شده است. نگاشت SEDSDMبه SDLCشامل مراحلی بصورت اکتشاف مفهوم، شناسایی نیازمندیها، طراحی وتوسعه، پیاده سازی و تولید، پشتیبانی و بازنشستگی متدولوژی میباشد. در مرحله اکتشاف و ایده پردازی مهندس به طور عمده با اکتشاف آمادگی فناوری پیش از ارزیابی نیازهای کاربران و تجزیه تحلیل ریسک درگیر است در کنار ارزیابی نیازهای کاربران مهندس اکنون باید خطرات مرتبط با این پروژه را تعیین کند بنابراین تجزیه و تحلیل ریسک انجام شده می تواند برای شناسایی دارایی های اطلاعاتی و تهدیدها و آسیب پذیری های مرتبط با هم یک گسترش یابد الزامات امنیتی برای کاهش و سپس باید خطرات شناسایی شده

تعیین شود SECSDM پیشنهاد می کند. در مرحله شناسایی نیازمندی ها بر اساس پیش ارزیابی های نیاز ها و آمادگی فناوری و همچنین ارزیابی ریسک مهندس باید راه حل های ممکن را توصیف کند و الزامات کامل تیم را تعریف و مشخصات محصول خطرات شناسایی نشده در مرحله قبلی نیازهای دقیق و مشخصات محصول را بررسی و از این رو میتوان از الزامات امنیتی برای اطلاع از اینکه کدام سرویس های امنیتی باید انتخاب شود استفاده کرد و خطرات شناسایی شده را کاهش داد. هدف اصلی مرحله طراحی و توسعه این است که مهندسان بتوانند معماری سیستم را طراحی کنند و نیازهای سیستم را به زیر سیستم ها اختصاص دهند و سرویس های امنیتی مشخصات در مرحله قبل باید تعریف و ترسیم شده به مکانیزم های امنیتی خاص که نحوه اجرای خدمات SECSDM نشان میدهد را توصیه میکند **.در مرحله بعد** اجرای فرایند طراحی مهندسی شامل زیرساخت یکپارچه سازی و آزمایش سیستم سخت افزار و نرم افزار هنگام انجام دادن فعالیت های SECSDMدر SDLC باید مکانیزم های امنیتی شناخته شده را هنگام استفاده از نرم افزار پیاده سازی کند از آنجا که مهندسان ممکن است بر استانداردهای کدگذاری ایمن مهارت نداشته و یا از بهترین روشهای آگاهی نداشته باشند باید در مورد استانداردهای کدگذاری و بهترین روش ها برای استفاده در هنگام اجرای کنترل امنیتی خواست توصیه شود .در مرحله پشتیبانی در مدت زمان استفاده پشتیبانی SECDLC مهندس باید اطمینان حاصل کند که محصول مطابق با نیاز کاربران کار میکند و نرم افزار به مرحله استفاده و پشتیبانی اضافه میشود. مهندس باید تنها عملکرد کاربردی نرم افزار تعبیه شده روی دستگاه بلکه جنبه های امنیتی مرتبط با آن را نیز بداند. در مرحله بازنشستگی برای از کار افتادن یک SECDSMپروژه اختصاص ندارد با این حال از کار افتادن این محصول را پشت سر میگذارد که ممکن است.[6] متخصصان مهندسی نرم افزار متدولوژی های مختلفی را برای توسعه نرم افزار امن ارائه داده اند که معروف ترین آنها متدولوژی MCGraw[7] و SDL مایکروسافت[8] است. این متدولوژی ها در فازهای مختلف توسعه نرم افزار تکنیک ها و ابزارهای خاصی را تامین امنیت نرم افزار ارائه نموده اند. مدلسازی UMLsecدر فازهای تحلیل و طراحی توسعه امن[9] ، استفاده از ابزارهای مدلسازی دیگر نظیر نمودار Case Abuseکه سناریوی استفاده کاربر و سوء استفاده نفوذگر از سیستم را بصورت همزمان نشان میدهد[10,8]. مدلسازی تهدیدها با استفاده از رهیافت Strideمایکروسافت که در واقع نمودار آن نوعی DFDتوسعه یافته برای نشان دادن تهدیدها است[11] ترسیم درخت های حمله که نوعی درخت OR-ANDاست[12]. مرور کد در فاز کدنویسی[13] انجام تست نفوذ در انتهای مراحل توسعه نرم افزار[14] و مدیریت ریسکهای امنیتی در سرتاسر چرخه حیات نرم افزار [15]از جمله فعالیت های مهمی است که در توسعه نرم افزار انجام میشود. مایکروسافت به ایجاد مجموعه ایی از اصطلاحات به منظور دستیابی به امنیت در فرایند توسعه نرم افزار بنام SDLمبادرت ورزیده است.[8] متدولوژی SDL مایکروسافت بر مبنای اصول راهنمای امنیت و حریم خصوصی بنا نهاده شده است. پیروی از این اصول میتواند منجر به تولید نرمافزاری امن و قابل اعتماد گردد[16]. SDL تعداد آسیب پذیری ها در نرم افزار در حال ارسال را تا بیش از ۵۰ درصد کاهش میدهد اما از نظر متدولوژی چابک، روش SDLبسیار سنگین است زیرا برای حفظ امنیت محصولات بسیار عظیم مانند Windowsو Office Microsoftطراحی شده است که هردو دارای چرخه توسعه طولانی هستند[17]. کاربران روشهای چابک بخواهند از SDLاستفاده کنند باید دو تغییر ایجاد کنند. اول اینکه افزودن SDLبه فرایندهای چابک باید اندک باشد یعنی تیم توسعه برای هر ویژگی نرم افزار، فعالیت های SDLرا به اندازه کافی انجام دهد. دوم اینکه مراحل توسعه مرتبط با

SDLدر یک قالب سریعتر سازماندهی شوند زیرا روش های چابک را بکار نگرفته اند .جان و همکاران[18] برای شناسایی و جلوگیری از حملات زیر صفر یک مدل زودرس آسیب پذیری را پیشنهاد می دهد. مرتضی و همکاران[19] گرایش های الگوی آسیب پذیری در اعصار را تجزیه تحلیل میکند. محققان دیگر سعی می کنند آسیب پذیری ها را از مراحل اولیه توسعه یا به حداقل رساندن حملات مخرب از فرایند مهندسی نرمافزار امنیتی کاهش دهند و برآورد هزینه ها برای انجام یک توسعه نرم افزار آن ضروری است. در این مورد yang و همکاران [21]مدلی برای برآورد تلاش توسعه نرم افزار سیستم عامل ها در چین ایجاد کردند با این حال ارزیابی علل خاص خرابی نرم افزار پیچیده است و برخی دیگر از مشکلات تولید نرم افزاری ایمن توسط Sodiyaو همکاران[22] ارائه شده است. در توسعه برنامه هایی که استانداردهای ایمنی را برآورد میکنند و روند دشوار است همانطور که رحمان و صبا پیشنهاد میکنند حتی وقتی تکنیک های هوش مصنوعی استفاده می شود برآورد مشکل میشود. به گفته این سیاست و همکاران امنیت نرم افزار دیگرنباید اختیاری باشد بلکه اجباریست در این راستا بیشتر مطالعات با هدف توسعه ایمن با هدف معرفی ارائه شده و در چرخه زندگی توسعه نرمافزار SDLC بررسی و اندازه گیری شده اند. جونز استدلال می کند که امنیت باید در تمام مراحل چرخه عمر سیستم ها لحاظ شوند از طرف دیگر کریم و همکاران یک پسوند امنیتی را به مدل SDLCاضافه کردند[23].

۴- تکنیک های هوش مصنوعی در برنامه نویسی و تست نرم افزار

تکنیک های آموخته شده از تحقیقات هوش مصنوعی ,برنامه نویسی پیشرفته را بسیار ساده تر انجام میدهد، به خصوص با توجه به جریان و کنترل اطلاعات به عنوان نتیجه پیشرفت در ارایه دانش. در ادامه ما به تکنیک های هوش مصنوعی مورد استفاده در پشتیبانی از وظایف برنامه نویسی و تست نرم افزار می پردازیم[24] .

۴-۱- برنامه نویسی

مهندسان نرم افزار میتوانند از تکنیک های هوش مصنوعی برای کمک به اتوماسیون یا کمک به فرآیند برنامه نویسی استفاده کنند.

۴-۱-۱- استفاده از هوش مصنوعی برای کمک به اتوماسیون فرایند برنامه نویسی

ایده در اینجا تهیه یک برنامه کامال خودکار ترکیبی است. این کار با تعیین متخصصان انسانی نرم افزار مورد نظر انجام میشود .به گونه ای که یک سیستم میتواند "توابع ,ساختار داده ها یا کل برنامه ها "را مستقیما از مشخصات تولید کند. بسیاری از فناوری های هوش مصنوعی قابل استفاده در این زمینه هستند. بوش در سال ۱۹۸۶ ضمن توصیف روش تحلیل و طراحی شی گرا توضیحات بدون مرز را به انواع داده توضیح میدهد. می توان از استدلال آنالوگ در استفاده مجدد از نرم افزار استفاده کرد .این ایده برای پیدا کردن سیستم های مشابه از نظر نیازمندی ها است تا آن را اصلاح کنیم. اگر چه این روند عملی بنظر میرسد ,اما در مهندسی نرم افزار تا حد زیادی اثبات نشده است. استدلال مبتنی بر مورد CBR [25] بر این فرض استوار است که مشکالت مشابه با راه حل های مشابه بهتر حل می شوند. گفته میشود که استدلال مبتنی بر مورد تعدادی از تکنیک های مدیریت مزایا را ارایه میدهد. برای بازیابی سنتز برنامه از مخازن مولفه استفاده مجدد از تجربه موفقیت آمیز گذشته مهم است .به عنوان نمونه،

یکی از کاربرد های فناوری استدلال مبتنی بر مورد CBR این بود که بسیاری از دانش دیگر از استفاده مجدد از بسته های نرم افزاری در کتابخانه های زبان های برنامه نویسی آدا ADA و سی C پشتیبانی میکردند. ایده استفاده مجدد از تجربه، بلند پروازانه ترین شکل استفاده مجدد از پشتیبانی استدلال مبتنی بر مورد CBR با آنچه که کارخانه تجربه نامیده میشود نزدیک است. این زمینه همچنین به عنوان یادگیری سازمانی شناخته می شود روش ها و تکنیک های تحقیق را برای مدیریت، انتخاب و تطبیق مصنوعات قابل استفاده مجدد از پروژه های مهندسی نرم افزار. یک کارخانه تجربه بر پایه هایی همچون فرایند بازخورد ها، ذخیره سازی مناسب از یک تجربه و پشتیبانی از استفاده مجدد بازیابی بنا شده است. برنامه نویسی محدودیت یکی دیگر از تکنیک های تکنیک های هوش مصنوعی است که در مهندسی نرم افزار کاربرد دارد. به عنوان مثال،برنامه نویسی محدودیت برای طراحی سیستم PTIDEJ (شناسایی الگو،ردیابی و تقویت در جاوا) مورد استفاده قرار گرفته است." شناسایی الگو،ردیابی و تقویت در جاوا "یک سیستم خودکار است که برای شناسایی معماری های میکرو مانند الگو های طراحی در کد منبع شی گرا طراحی شده اند. یک میکرو معماری زیر مجموعه کلاس ها را در یک برنامه معترض گرا تعریف میکند. جذابیت اصلی PTIDEJ این است که می تواند برای پاسخ های خود توضیحی ارایه دهد. این واقعا جالب است زیرا مهندسی نرم افزار و کد نویسی اغلب نوعی هنر محسوب میشوند و جایی که سیستم های کامال خودکار همیشه مورد توجه کاربران بالقوه (یا برنامه نویسان) قرار نمی گیرند. مهندسی نرم افزار مبتنی بر جستجو SBSE یک موضوع تحقیق نو ظهور است که بر نمایندگی از جنبه های مهندسی نرم افزار به عنوان مشکلاتی که ممکن است توسط استفاده راه های فرا اکتشافی توسعه یافته در جستجو در هوش مصنوعی حل شوند. مهندسی نرم افزار مبتنی بر جستجو ,اصالح وظایف مهندسی نرم افزار به عنوان مشکلات بهینه سازی است. یکی از تکنیک های بهینه سازی و جستجو که می تواند مورد استفاده قرار بگیرد الگوریتم های ژنتیکی هستند. الگوریتم های ژنتیکی با بهینه سازی جمعیت راه حل های آزمایشی برای یک مشکل ،برای تولید خودکار کد استفاده می شوند. افراد حقیقی در جامعه برنامه های کامپیوتری هستند.

۲-۱-۴- تست کردن

آزمایش نرم افزار یک کار گران قیمت در فرایند توسعه است و یکی از مهمترین چالش ها مربوط به اتوماسیون احتمالی آن است. تکنیک های هوش مصنوعی می توانند در این زمینه نقش اساسی داشته باشند. یکی از این تکنیک ها تکنیک های حل محدودیت هستند. از آنجا که فعالیت اصلی افوت و دمیلو در سال ۱۹۹۱ در زمینه آزمایش جهش توجه زیادی به استفاده از تکنیک های حل محدودیت در اتوماسیون تست نرم افزار(آزمایش مبتنی بر محدودیت) شده است.به عنوان مثال ATGEN یک تولید کننده داده های تست نرم افزار است که بر اساس اجرای نمادین و برنامه نویسی منطق محدود برای برنامه های آدا ADA است[26]. روش های مختلف دیگری وجود دارد که چگونه تکنیک های هوش مصنوعی می توانند فرایند آزمایش را پشتیبانی کنند. یکی از اولین مطالعات برای ارائه یک سیستم دانش بنیان برای آزمایش ، توسط برینگ و کرافورد در سال 1988انجام شده است که یک سیستم خبره مبتنی بر زبان برنامه نویسی Prologرا توصیف می کنند که یک برنامه کوبول (زبان برنامه نویسی سطح بال برای کاربرد های تجاری) را به عنوان ورودی می گیرد ، ورودی را تجزیه می کند تا شرایط مربوط را شناسایی کند و سپس هدف آن

تولید داده های آزمون بر اساس شرایط است. زمینه فعال تر تحقیق از اواسط دهه ۱۹۹۰ استفاده از برنامه ریزی هوش مصنوعی برای آزمایش بوده است. یک برنامه ریز هوش مصنوعی می تواند موارد آزمایشی ، متشکل از یک توالی از دستورات را با نمایش دستورات به عنوان اپراتور ، ارائه حالت های اولیه و تعیین هدف به عنوان آزمایش برای رفتار صحیح سیستم ایجاد کند. برنامه ریزی هوش مصنوعی همچنین برای آزمایش سیستم های توزیع شده و برای تولید موارد آزمایش برای رابط های گرافیکی کاربر استفاده شده است. استوارت راسل و پیتر نورویگ در سال ۱۹۹۴ در مقاله خود مربوط به هوش مصنوعی: یک رویکرد مدرن توضیح می دهد که تجزیه و تحلیل فرایند شکسته شدن چیزی به قطعات یا مؤلفه ها با هدف درک مؤلفه های فردی است. یک مقاله توسط کباچی در سال ۲۰۰۷ نشان داده است که استفاده از الگوریتم های ژنتیکی برای بهینه سازی از دهه ۱۹۸۰ به طور قابل توجهی رشد کرده است[27]. این روند همچنین در استفاده از آنها در آزمایش وجود دارد ، با مطالعات متعدد با هدف استفاده از خواص آنها در تالش برای تولید موارد بهینه آزمایش است. برتولینو در سال ۲۰۰۷ چارچوب مفیدی را برای خالصه نمودن چالش هایی که برای رفع مشکالت اطمینان از مناسب بودن سیستم برای اهداف با آن روبرو شده است[28] ، ارائه می دهد همچنین تحقیقات دیگری را درمورد موارد زیر ارائه میدهند:

۱-تدوین تئوری جهانی آزمایش

۲-تست کاملا اتوماتیک

۳-طراحی برای تسهیل آزمایش

۴-توسعه استراتژی های یکپارچه که هزینه آزمایش های مکرر را به حداقل می رساند

واپلر و واگنر در سال ۲۰۰۶ اذعان می کنند که استفاده از یک عملکرد تناسب اندام به عنوان وسیله اصلی برای جلوگیری از توالی های غیرقانونی کارآمد نیست. در عوض ، آنها استفاده جدیدی از برنامه نویسی ژنتیک GP را ارائه میدهند، که هدف آن یادگیری عملکردها یا برنامه ها از طریق تکامل است. به عنوان مثال ، نویسندگان از الگوریتم های ژنتیکی برای آزمایش برنامه های شی گرا استفاده کرده اند که در آن هدف اصلی ساخت موارد آزمایشی متشکل از توالی تماس های متد است[29]. ناند،کوار و جین در سال ۲۰۰۷ در مقاله استفاده از منطق فازی در توسعه نرم افزار موضوعات در سیستم های اطلاعاتی در مورد استفاده از منطق فازی در تست نرم افزار برای مدیریت عدم قطعیت موجود در این مرحله از توسعه نرم افزار توضیح می دهد.[30]

۵- نتیجه گیری

بنابراین به این امر مهم می رسیم که توسعه دهندگان نرم افزار عالوه برای اصول مشترک امنیتی تولید نرم افزار که باید مستمراً(در هر فازی) کنترل گردد می بایست به چارچوب های امنیتی سازمان در خلال پیاده سازی هر فاز مراجعه نمایند. در نتیجه تولید نرم افزار امن در یک سازمان تحقق نمی یابد مگر در یک خط تولید امن نرم افزار و خط تولید امن ساخته نمی شود جز با شناخت

چارچوب های امنیت نرم افزاری هر سازمان، تعیین محدوده اطمینان نرم افزار و سیاستهای امنیتی اتخاذ شده توسط کارفرما و ایجاد یک فرایند مشخص درون سازمانی که در آن نیازمندی های امنیتی مختلف تعیین و بوسیله مدل های تهدید ارزیابی شده و آسیب پذیری ها بر اساس سطح اطمینان هدف گذاری شده سازمان شناسایی و کنترل شود.

مراجع

1. Fujdiak, R., et al. Managing the secure software development. in 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS). 2019. IEEE.

2. Palombo, H., et al. An Ethnographic Understanding of Software (In) Security and a Co-Creation Model to Improve Secure Software Development. in Sixteenth Symposium on Usable Privacy and Security ({SOUPS} 2020)

3. Boehm, B. and E. Venson, Secure Software Development Levels and Costs. 2020.
4. Zareen, S., A. Akram, and S.J.A.S. Ahmad Khan, Security requirements engineering framework with BPMN 2.0. 2 extension model for development of information systems.2020. **10**(14): p. 4981.
5. Solms, S.V. and L.A. Futcher, Adaption of a Secure Software Development Methodology for Secure Engineering Design. in IEEE Access, 2020.
6. Von Solms, S. and L.A.J.I.A. Futcher, *Adaption of a Secure Software Development Methodology for Secure Engineering Design*. 2020. **8**: p. 125630-125637.
7. Hainey, T.J.I.J.o.I.M., Information Systems Development Methodologies, Techniques & Tools, David Avison, Guy Fitzgerald, McGraw-Hill Education, Maidenhead, 2006 (656pp., ISBN 10 0-07-
7114175,£ 41.99). 2007. **27**(1): p. 58-59.
8. de Vicente Mohino, J., et al., *The application of a new secure software development life cycle (S SDLC) with agile methodologies*. 2019. **8**(11): p. 1218.
9. Lee, J., et al., A software development methodology for secure web application. 2019. **9**(1): p. 336-341.
10. Alenezi, M. and S. Almuairfi, ESSENTIAL ACTIVITIES FOR SECURE SOFTWARE DEVELOPMENT.
11. Lingham, A.D., et al., *Implementation of Security Features in Software Development Phases*. 2020.
12. Sabbah, M., et al., Development and properties of new chitosan-based films plasticized with spermidine and/or glycerol. 2019. **87**: p. 245-252.
13. Benni, B., et al., A delta- oriented approach to support the safe reuse of black- box code rewriters. 2019. **31**(8): p. e2208.
14. Sara, M., *Test performance and security measurements in software development*. 2019.
15. Dhir, S., D. Kumar, and V. Singh, Success and failure factors that impact on project implementation using agile software development methodology, in Software Engineering. 2019, Springer. p. 647-654.
16. Núñez, J.C.S., A.C. Lindo, and P.G.J.I.A. Rodríguez, A Preventive Secure Software Development Model for a Software Factory: A Case Study. 2020. **8**: p. 77653-77665.

17. Jimenez-Ramirez, A., et al. A method to improve the early stages of the robotic process automation lifecycle. in International Conference on Advanced Information Systems Engineering .2019Springer.
18. Tsen, E., R.K. Ko, and S.J.A.a.S. Slapničar, Organisational cyber resilience and its influence on cyber attack outcomes: an exploratory study of 1,145 publicised attacks. 2020.
19. Amankwah, R. Predicting Vulnerable Software Components via Bellwethers. in Trusted Computing and Information Security: 12th Chinese Conference, CTCIS 2018, Wuhan, China, October 18, 2018, Revised Selected Papers. 2019. Springer.
20. Abaimov, S. and G.J.I.A. Bianchi, CODDLE: Code-injection detection with deep learning. 2019. **7**: p. 128617-128627.
21. Jyoti tewari,Swati arya, Prem narayan singh ,"Approach of Intelligent Software Agents in Future Development",IJARCSSE, ISSN:2277128X , May 2013.
22. Hany H Ammar, Walid Abdelmoez and Mohamed Salah Hamdi, "Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems" – ICCIT 2012.
23. Mark Harman "The Role of Artificial Intelligence in Software Engineering" ACM computing surveys, 2011.
24. Prince Jain,"Interaction between Software Engineering and Artificial Intelligence-A Review"-International Journal on Computer Science and Engineering (IJCSE), ISSN: 0975-3397, Vol 3 No.12 December 2011.
25. Dr. Nachamai. M ,Senthil Vadivu and Tapaskar , "Enacted Software Development Process Based on Agent methodologies",IJEST ,VOL.3, NO.11, November 2011
26. Farid Meziane, Sunil Vadera," Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects" DOI: 10.4018/978-1-60566-7584.ch014. 2010
27. Farah Naaz Raza, "Artificial Intelligence Techniques in Software engineering International multiconference of Engineers and computer scientists 2009 , ISBN 978988.17012-2-0 Vol 1 – IMECS 2009.
28. Parveen Ranjan Srivastava, and Tai-hoon Kim, Application of Genetic Algorithm in Software Testing, International Journal of Software Engineering and its Applications.Vol. 3, No.4, October 2009.
29. Yang, H.-L., & Wang, C.-S. (2009). Recommender system for software project planning one application of revised CBR algorithm. ExpertSystems with Applications, 36(5), 8938– 8945. doi:doi:10.1016/j.eswa.2008.11.050.
30. Hooshyar, B., Tahmani, A., & Shenasa, M. (2008). A Genetic Algorithm to Time-Cost Trade off in project scheduling. In Proceedings of the IEEE World Congress on Computational Intelligence (pp. 3081-3086), Hong Kong. Washington DC: IEEE Computer Society.