به نام خدا

موضوع:

چگونگی استفاده از abuse case و risk analysis در طراحی امن نرم افزار

گردآوری شده توسط:

حوری دهش

استاد درس:

دکتر زینب آقائی

تیرماه ۱۴۰۲

فهرست

۱. مقدمه

یک فرآیند مهندسی امنیتی معتبر، همانطور که توسط معیارهای مشترک مشخص می‌شود، یک فعالیت پیچیده است که شامل بسیاری از محصولات کاری خاص است: اهداف امنیتی، الزامات امنیتی، سیاست‌های امنیتی، مشخصات عملکردی و مدل‌های سیاست امنیتی. این محصولات کاری در فرآیندی که هدف آن ایجاد محصولات امنیت اطلاعات قابل اعتماد است، ضروری هستند. اما درک محصولات کاری و روابط بین آنها ممکن است سخت باشد، حتی برای افرادی با پیشینه فنی قوی اما دانش کمی از مهندسی امنیت.

تجزیه و تحلیل ریسک مجموعه ای از تکنیک ها است که برای بررسی مشکلات ایجاد شده توسط عدم قطعیت و ارزیابی اثرات آنها استفاده می شود. تا به امروز در بسیاری از مناطقی استفاده شده که ایمنی بسیار مهم است. صنایعی که از این تکنیک ها استفاده می کنند شامل صنایع هسته ای، شیمیایی و فضایی است.

توسعه نرم افزار حوزه دیگری است که نویسندگان معتقدند تکنیک های تجزیه و تحلیل ریسک را می توان در آن به کار برد. با گسترش این حوزه، استفاده از رایانه در شرایط بحرانی ایمنی گسترش یافته است.

همچنین دشمنان و بازیگران خرابکار نیز ممکن است از روش های مختلف برای شناسایی ضعف ها و نقاط ضعف امنیتی در سیستم ها استفاده کنند. در اینجا استفاده دشمنان از risk analysis و abuse case به صورت زیر می‌تواند رخ دهد:

۱. استفاده از risk analysis: دشمنان ممکن است با استفاده از تحلیل ریسک، نقاط ضعف امنیتی را شناسایی کنند و در نتیجه بتوانند به طور مستقیم ریسک های امنیتی را مورد هدف قرار دهند. آنها ممکن است از فرصت هایی که از نتیجه تحلیل ریسک به دست می‌آورند، برای حملات و نفوذ به سیستم ها استفاده کنند.

۲. استفاده از abuse case: بازیگران خرابکار ممکن است از استفاده از abuse case برای شناسایی نقاط ضعف و آسیب‌پذیری ها در سیستم استفاده کنند. با تحلیل سناریوهای سواستفاده و رفتارهای غیرمجاز، آنها می‌توانند از این موارد برای برنامه ریزی و انجام حملات متقلبانه به سیستم ها استفاده کنند.

به طور کلی، هر دوی risk analysis و abuse case می‌توانند به عنوان ابزارهایی برای تهدید و حملات، در دسترس دشمنان و بازیگران خرابکار قرار بگیرند. استفاده آگاهانه و مناسب از این روش ها و اعمال تدابیر امنیتی مناسب می‌تواند به کاهش تهدیدات و محافظت از سیستم ها و نرم افزارها کمک کند.

۲. چرخه عمر توسعه نرم افزار

چرخه عمر توسعه نرم افزار، یک مدل فرایندی است که مراحل مختلف توسعه نرم افزار را به ترتیب مشخصی تعریف می‌کند. این مدل ها به طور کلی، مراحلی مانند تحلیل نیازمندی ها، طراحی، پیاده سازی، تست و نگهداری را شامل می‌شوند. هر مدل ممکن است دارای مراحل بیشتر یا کمتری باشد و ممکن است با توجه به نوع پروژه و رویکرد های توسعه متفاوتی داشته باشد.

به طور کلی، چرخه عمر توسعه نرم افزار می‌تواند به مراحل زیر تقسیم شود:

۱. تحلیل نیازمندی ها: در این مرحله، نیازمندی های سیستم جمع آوری و تحلیل می‌شوند تا یک تصویر دقیق از نیازهای کاربران و مشتریان به دست آید.

۲. طراحی: در این مرحله، یک طرح کلی از سیستم توسعه می‌شود که شامل ساختار، واسطها و رابطه بین اجزا مختلف است.

۳. پیاده سازی: در این مرحله، کدهای برنامه نویسی به زبان‌های مورد نظر نوشته می‌شوند و اجرای سیستم آغاز می‌شود.

۴. تست: در این مرحله، عملکرد و صحت سیستم بررسی می‌شود و خطاها و اشکالات شناسایی می‌شوند.

۵. نگهداری: بعد از اجرا، سیستم نیازمند نگهداری و به روزرسانی می‌باشد. این شامل تعمیر خطاها، به روزرسانی امکانات و پشتیبانی از کاربران است.

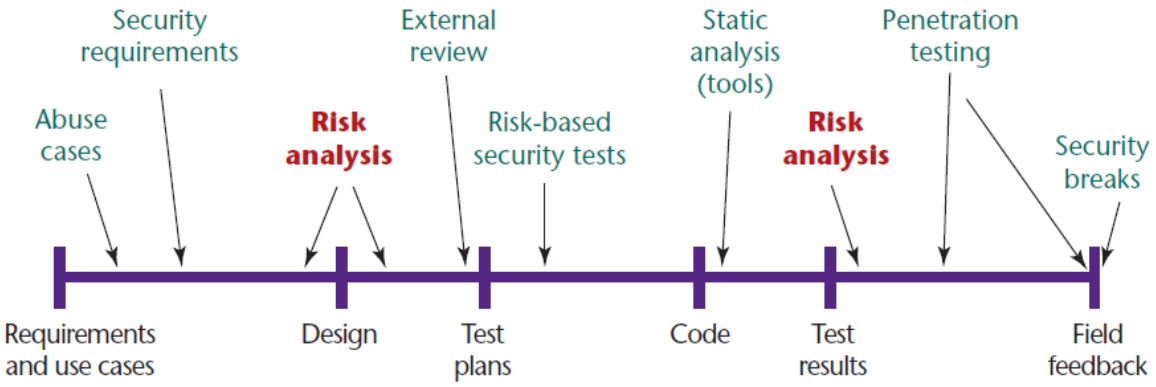
risk analysis و abuse case، در تمام مراحل چرخه عمر توسعه نرم افزار قابل شناسایی و مدیریت هستند:

- risk analysis: در ابتدای پروژه، تحلیل ریسک به منظور شناسایی و ارزیابی ممکن‌ترین ریسک‌های موجود در پروژه انجام می‌شود. این ریسک‌ها می‌توانند شامل عوامل فنی، بودجه، زمان، نیازمندی ها و مدیریت پروژه باشند. با شناسایی ریسک ها، اقداماتی برای کاهش و مدیریت آنها در طول چرخه عمر پروژه انجام می‌شود.

- abuse case: در طول فرایند توسعه نرم افزار، مواردی را که می‌توان به عنوان سواستفاده تلقی کرد، باید شناسایی و مدیریت کرد. موارد سواستفاده ممکن است شامل آسیب های امنیتی، خطرات حریم خصوصی، استفاده نادرست از سیستم و... باشند. در نتیجه امنیت و حفاظت در طول توسعه نرم افزار باید همواره در نظر گرفته شود و مدیریت مناسب برای کاهش خطرات سواستفاده انجام شود.

برای موفقیت پروژه، مدیریت ریسک و موارد سواستفاده باید به صورت مداوم در طول چرخه عمر توسعه نرم افزار توسط تیم توسعه و مدیران پروژه رصد و مدیریت شود.

یک نمایی از چرخه عمر توسعه نرم افزار:



در چرخه عمر توسعه نرم افزار، abuse case در ابتدا است چون از آنجایی که ممکن است تاثیر قابل توجهی بر روی نیازمندی ها و طراحی سیستم داشته باشد از ابتدای فرایند، تحلیل نیازمندی ها شروع می‌شوند یعنی قبل از اینکه تیم توسعه و طراحی به طراحی جزئیات و پیاده سازی بپردازند، باید موارد سواستفاده را شناسایی و در نظر بگیرند.

دلیل این امر این است که شناسایی موارد سواستفاده از ابتدای فرایند توسعه نرم افزار، امکان تعیین نیازمندی ها و طراحی امن را فراهم می‌کند. با شناسایی این موارد، تیم توسعه و طراحی می‌توانند برنامه‌ریزی های لازم را برای جلوگیری از سواستفاده از سیستم انجام دهند و اقدامات امنیتی مناسب را در طراحی و پیاده سازی سیستم اعمال کنند.

همچنین شناسایی abuse case در مراحل ابتدایی توسعه نرم افزار، به تیم توسعه و مدیران پروژه کمک می‌کند تا در تخصیص منابع و بودجه، اولویت بندی صحیح را در نظر بگیرند. با اولویت بندی موارد سواستفاده در مراحل ابتدایی، می‌توان هزینه های احتمالی بعدی که ناشی از تعمیر خطاها و به روزرسانی‌های امنیتی می‌باشد را کاهش داد و بهبود امنیت سیستم را تسهیل کرد.

بنابراین ابتدا شناسایی و تحلیل موارد سواستفاده در چرخه توسعه نرم افزار، بهبود امنیت سیستم را تضمین می‌کند و کاهش هزینه های مربوط به اصلاحات و به روزرسانی‌های امنیتی در مراحل بعدی را فراهم می‌کند.

همین طور در چرخه عمر توسعه نرم افزار، risk analysis در مراحل مختلف قرار می‌گیرد تا بهبود امنیت و کاهش خطرات پروژه را تضمین کند. این تحلیل معمولا قبل و بعد از طراحی و بعد از نتایج تست انجام می‌شود که بخاطر دلایل زیر می‌باشد:

قبل از طراحی (Pre-Design):

قبل از شروع فعالیت طراحی، تحلیل ریسک انجام می‌شود تا مشکلات و خطرات ممکن در فرآیند طراحی شناسایی شوند. در این مرحله، تحلیل ریسک بر روی نیازمندی ها و قابلیت های مورد نظر انجام می‌شود. هدف این تحلیل این است که ریسک های موجود شناسایی شوند و راهکارهای مناسبی برای کاهش و مدیریت آنها در طراحی ارائه شود. این اقدام می‌تواند از انتخاب راهکارهای مناسب تا جلوگیری از ایجاد مشکلات بزرگتر در مراحل بعدی توسعه نرم افزار کمک کند.

بعد از طراحی (Post-Design):

بعد از انجام فعالیت طراحی و ایجاد طرح کلی سیستم، تحلیل ریسک بر روی طرح طراحی شده صورت می‌گیرد. در این مرحله، تحلیل ریسک به منظور شناسایی و مدیریت ریسک های مرتبط با ساختار و واسطهای سیستم، اجزا مختلف و سیاست های امنیتی صورت می‌گیرد. با این تحلیل، می‌توان مسائل امنیتی محتمل را شناسایی کرده و راهکارهای امنیتی مناسب را در طراحی و پیاده سازی سیستم اعمال کرد.

بعد از نتایج تست (Post-Test Results):

همچنین، تحلیل ریسک بعد از دریافت نتایج تست نیز بسیار حیاتی است. نتایج تست نرم افزار نشان می‌دهد که آیا نرم افزار مطابق با نیازمندی ها و مشخصات فنی عمل می‌کند یا خیر. در این مرحله، تحلیل ریسک بر روی خطرات و مشکلات شناسایی شده در نتایج تست انجام می‌شود و راهکارهای مناسبی برای برطرف کردن آنها ارائه می‌شود. هدف این تحلیل، اطمینان حاصل کردن از کیفیت و عملکرد مطلوب نرم افزار و اطمینان از عدم وجود ریسک‌های جدی در محصول نهایی است.

بنابراین، تحلیل ریسک در چرخه توسعه نرم افزار در این مواقع صورت می‌گیرد تا مشکلات مرتبط با امنیت و کیفیت را شناسایی کند و راهکارهای مناسب را ارائه دهد. این اقدامات بهبود امنیت و اطمینان از عملکرد مطلوب نرم افزار را تضمین می‌کنند.

۳. معرفی abuse case

برای ایمن سازی فضای مجازی، ساختن نرم افزارهای ایمن بسیار مهم است. فعالیت ها و محصولات مرتبط با امنیت باید در هر یک از مراحل چرخه عمر توسعه نرم افزار ادغام شوند. یکی از فعالیت های مرتبط با امنیت، توسعه abuse case است. abuse case یک مورد استفاده از دیدگاه مهاجم با هدف آسیب رساندن به سیستم است. abuse case ممکن است به کاربران یا خود سیستم آسیب برساند. abuse case موارد کاربردی در استفاده از سیستم را تهدید می کند و باید از آن به عنوان یک پشتیبانی برای توسعه دهندگان، جهت استخراج الزامات امنیتی استفاده شود. توسعه abuse case به مهندسان نرم افزار اجازه می دهد تا از دیدگاه مهاجمان فکر کنند و تصمیم بگیرند و مستند کنند که نرم افزار چگونه باید در برابر استفاده نامشروع واکنش نشان دهد. براساس مواردی که مشاهده شد می توان اقدامات متقابل انجام داده تا مهاجم از طریق abuse case نتواند به سیستم آسیب برساند.

همین طور ساختار عمومی یک abuse case به صورت زیر است:

یک abuse case معمولا شامل سه عنصر اصلی است: بازیگر (Actor)، سناریو (Scenario) و ریسک (Risk) می‌باشد.

۱. بازیگر (Actor): بازیگر به فرد یا سیستم دیگری اشاره دارد که می‌تواند از سیستم مورد بررسی سواستفاده کند. ممکن است این بازیگر یک هکر، یک کاربر غیرمجاز، یک کاربر نادرست یا حتی یک سیستم خارجی باشد.

۲. سناریو (Scenario): سناریو توصیف می‌کند که چگونه بازیگر می‌تواند از سیستم سواستفاده کند. این شامل فعالیت هایی است که بازیگر انجام می‌دهد تا ریسکی را برای سیستم ایجاد کند، مانند نفوذ به سیستم، دسترسی غیرمجاز، تغییر داده ها، یا هر فعالیت مشابهی که می‌تواند سیستم را به خطر بیاندازد.

۳. ریسک (Risk): ریسک‌ها مشکلات امنیتی و آسیب‌پذیری‌هایی هستند که می‌توانند به دنبال سواستفاده بازیگران در سناریو باشند. این ممکن است شامل دسترسی غیرمجاز به داده‌ها، از بین بردن یا تغییر داده ها، انجام عملیات نامناسب یا هر گونه فعالیتی باشد که سلامت و امنیت سیستم را به خطر بیاندازد.

این ساختار عمومی است و می‌توان بر اساس نیازهای خاص پروژه، آن را سفارشی کرد و جزئیات بیشتری را در مورد بازیگران، سناریوها و ریسک‌های مربوطه به نرم افزار خود اضافه کرد.

موارد زیر جز abuse case ها محسوب می‌شود:

- گروهی از تراکنش های کامل بین یک یا چند کاربر و یک سیستم که باعث آسیب زایی میشود.
- نمودار های مورد استفاده مبتنی بر UML
- معمولا با استفاده از زبان طبیعی توصیف میشود همچین ممکن است از نمودار Tree/DAG استفاده شود.
- یک عضو از سیستم که از آن می توان برای امتیاز گیری و سپس سواستفاده، استفاده کرد.
- شامل طیفی از امتیازات امنیتی است که ممکن است مورد سواستفاده قرار گیرد.
- شامل آسیب ناشی از یک پرونده abuse case است.

استفاده از abuse case در طراحی امن نرم افزار شامل مراحل زیر است:

۱. شناسایی بازیگران: بعد از اینکه بازیگران مدل مورد استفاده، شناسایی شدند بازیگران مدل مورد abuse case را شناسایی کنید این بازیگران ممکن است کاربران غیر مجاز، هکرها یا سیستم های خارجی باشند. اگر یک بازیگر سعی کند از سیستم استفاده مضر کند، یک عامل مخرب مربوطه را به مدل abuse case اضافه کنید. پس از نمایش نقش‌های داخلی، برای هر مزاحمی که ممکن است مشکل ساز باشد، باید بازیگران اضافه شوند. افراد خارجی را در درجه اول بر اساس مهارت ها و منابع تشخیص دهید. اسناد الزامات ممکن است در شناسایی عوامل برای موارد سواستفاده کمک کند، اما باید تجزیه و تحلیل دقیقی از محیط سیستم نیز انجام شود. برای متخصص امنیت مهم است که در مورد بازیگران بالقوه با کاربران و مشتریان صحبت کند.

۲. شناسایی موارد سواستفاده: برای هر بازیگر، تعامل آنها با سیستم را مشخص کنید. هر مورد سواستفاده را نام ببرید. در این مرحله، ترسیم نمودار مورد سواستفاده مفید است.

۳. تعریف موارد سواستفاده: از آنجایی که رابط کاربری سیستم بیشتر اصلاح می شود و اجزای خاص شناسایی می شوند، می توان مورد سواستفاده را شرح داد. از آنجایی که از یک ساختار درختی برای توصیف نقاط سواستفاده احتمالی استفاده میشود، تعریف را تا زمانی به تعویق بیندازید که ساختار سیستم زمان کافی برای کار با آن داشته باشد.

۴. بررسی دانه بندی: ممکن است موارد abuse case بسیار کم یا ممکن است بسیار زیاد باشد. تصمیم گیری در مورد تعداد مورد نیاز عمدتا به تجربه و در نظر گرفتن سیستم هدف خاص بستگی دارد. در تجربه هایی که تا الان وجود دارد میتوان با موارد بیش از حد abuse case روبرو شد:

۱) شامل موارد احتمالی اما بعید

۲) مدل سازی با جزئیات بیش از حد

مشکل اخیر منجر به چندین مورد abuse case می شود که فقط با جزئیات نامناسب برای هدف مشخص می شوند. به عنوان مثال، در بیشتر موقعیت ها، نیازی به دو مورد abuse case از سرقت رمز عبور نداریم که فقط نوع پوسته فرمان مورد استفاده برای انجام سرقت متفاوت است. هنگام نادیده گرفتن یک مورد abuse case به عنوان نامحتمل، باید محتاط باشیم. برخی از موارد abuse case ممکن است بسیار پیچیده و برخی دیگر بیش از حد انتزاعی باشند. یک مدل مورد سواستفاده خوب، جزئیات یکنواختی را در موارد خود دارد، و نه تعداد زیادی از آنها.

۵. بررسی کامل بودن و حداقل بودن: هر شرح abuse case باید بررسی شود تا ببینیم آیا تعاملی را توصیف می کند که منجر به آسیب به کاربران سیستم می شود یا خیر، همچنین باید بررسی شود که آیا یک مورد abuse case مهم حذف شده است یا خیر. اسناد الزامات و مدل مورد استفاده باید به همراه توضیحات ویژگی های امنیتی پیش بینی شده بررسی شوند. باید با کاربران و مشتریان مشورت شود تا اطمینان حاصل شود که هیچ سواستفاده انتقادی نادیده گرفته نشده است.

همچنین به صورت دیگری نیز از abuse case برای طراحی امن نرم افزار استفاده میشود که شامل مراحل زیر می باشد:

۱. شناختن فرضیات: مشخص کنید که چه کسانی ممکن است به نرم افزار دسترسی داشته باشند و چه نوع داده ها و عملیاتی که می‌تواند انجام شود، را شناسایی کنید.

۲. شناسایی حملات محتمل: فرض کنید شما یک حمله‌کننده هستید و تلاش کنید تا به دنبال حملاتی باشید که ممکن است بر روی نرم افزار انجام شود. به عنوان مثال، حملات نفوذ، حملات دسترسی نامناسب، حملات انکار سرویس و...

۳. توصیف حملات سواستفاده: برای هر حمله محتمل، abuse case را توصیف کنید. این شامل مشخصات حمله، مراحل اجرای آن، منابعی که مورد سواستفاده قرار می‌گیرند و تاثیر حمله بر نرم افزار است.

۴. تحلیل اثر: ارزیابی کنید که هر حمله abuse case چه تاثیری بر روی نرم افزار، داده ها و کاربران آن خواهد داشت. این تحلیل به شما کمک می‌کند تا اولویت بندی کنید و روی مشکلاتی که بیشترین تاثیر را دارند، تمرکز کنید.

۵. پیشگیری و محافظت: برنامه‌ریزی و اعمال راه حل های امنیتی برای پیشگیری و محافظت در برابر حملات abuse case. این شامل استفاده از تکنیک های رمزنگاری، محدود کردن دسترسی به منابع مهم، استفاده از فیلترها و ورودی های صحت سنجی، بررسی هویت و سطوح دسترسی و... می‌شود.

abuse case به طور عمده برای شناسایی سناریوهای سواستفاده و رفتارهای غیرمجاز در نرم افزار استفاده می‌شود. این روش به طراحان نرم افزار کمک می‌کند تا رفتارهایی که بازیگران خرابکار ممکن است انجام دهند را شناسایی کنند و اقدامات امنیتی مناسب را در نرم افزار اعمال کنند.

در مورد حملات خاص که با استفاده از abuse case صورت گرفته‌اند، ممکن است موارد زیر را مشاهده کرد:

۱. حملات نفوذ (Intrusion Attacks): بازیگران خرابکار می‌توانند با استفاده از رفتارهای غیرمجاز و سواستفاده از نقاط ضعفی که در abuse case شناسایی شده‌اند، به سیستم نفوذ کنند و دسترسی غیرمجاز به منابع و اطلاعات را به دست آورند.

۲. حملات انکار سرویس (Denial of Service, DoS): بازیگران خرابکار می‌توانند با استفاده از سناریوهای سواستفاده در abuse case، سیستم را با حملات DoS برهم زنند و از دسترسی کاربران به سرویس‌ها جلوگیری کنند.

۳. حملات تقلب (Fraud Attacks): بازیگران خرابکار ممکن است از سناریوهای سواستفاده در abuse case برای انجام عملیات تقلبی مانند سرقت اطلاعات شخصی، سرقت اعتبار و ارسال اطلاعات تقلبی استفاده کنند.

۴. حملات خرابکارانه (Malicious Attacks): با استفاده از سناریوهای سواستفاده در abuse case، بازیگران خرابکار ممکن است برنامه های مخرب و مضر را در سیستم قرار داده و به نحوی سیستم را تحت تاثیر قرار دهند.

۵. حملات جاسوسی (Espionage Attacks): در صورتی که در abuse case رفتارها و عملکردهای مرتبط با حفظ حریم خصوصی شناسایی شده باشند، بازیگران خرابکار ممکن است از این اطلاعات برای جمع‌آوری و جاسوسی در سیستم استفاده کنند.

استفاده از abuse case می‌تواند به شناسایی و پیشگیری از این نوع حملات کمک کند. با تشخیص سناریوهای سواستفاده و رفتارهای غیرمجاز، تیم های امنیتی می‌توانند اقدامات امنیتی مناسب را برای جلوگیری از این نوع حملات در نرم افزار انجام دهند.

علاوه بر abuse case، در فرآیند طراحی امن نرم افزار، موارد دیگری نیز مورد ملاحظه قرار می‌گیرند که عبارتند از:

۱. Use Case (مورد استفاده): Use case ها توصیف می‌کنند که سیستم نرم افزاری چگونه باید تعامل کند و چه عملیات هایی باید انجام شود تا نیازهای کاربران برآورده شود. Use case ها در اصل توصیف می‌کنند که سیستم باید چه کاری انجام دهد.

۲. Functional Requirements (نیازمندی های کارکردی): نیازمندی های کارکردی شامل لیستی از عملکردهایی است که سیستم نرم افزاری باید انجام دهد. این نیازمندی ها به طور مشخص، مشخص می‌کنند که سیستم چه عملیاتی را اجرا کند و چه ویژگی‌هایی را داشته باشد.

۳. Non-functional Requirements (نیازمندی‌های غیرکارکردی): نیازمندی‌های غیرکارکردی به جنبه های غیرفنی نرم افزار مرتبط هستند. این نیازمندی‌ها ممکن است شامل عواملی مانند کارایی، امنیت، قابلیت حمل و نقل، قابلیت توسعه، راحتی استفاده و سازگاری باشد.

۴. User Stories (داستان کاربر): داستان کاربران توصیف کننده‌ی نیازها، هدف‌ها و فعالیت‌های مورد انتظار کاربران است. این داستان‌ها در قالب جملات ساده و قابل فهم بیان می‌شوند و به تیم توسعه راهنمایی می‌کنند که کاربران نیازمند چه چیزی هستند و سیستم چگونه باید برای آن‌ها عمل کند.

۵. Threat Modeling (مدل سازی تهدید): در این روش، تهدیدات امنیتی محتمل در محیط نرم افزار شناسایی و تحلیل می‌شوند. این شامل شناسایی منابع، تهدیدها، آسیب‌پذیری‌ها و تهدیدات مختلف است. از این طریق می‌توان نقاط ضعف و نقاط قوت سیستم را شناسایی کرده و برای رفع تهدیدات امنیتی اقدامات مناسبی را انتخاب کرد.

۶. Risk Assessment (ارزیابی ریسک): در این فرآیند، ریسک‌های مرتبط با نرم افزار شناسایی و ارزیابی می‌شوند. این شامل تعیین احتمال و اثرات ریسک‌ها بر امنیت و عملکرد سیستم است. با توجه به نتایج ارزیابی ریسک، می‌توان تدابیر امنیتی مورد نیاز را انتخاب و اعمال کرد.

۷. Security Controls (کنترل‌های امنیتی): این شامل اقدامات و فناوری های مورد استفاده برای حفاظت از سیستم در برابر تهدیدات امنیتی است. این شامل کنترل‌های فنی مانند رمزنگاری، کنترل دسترسی، نظارت و لاگ‌گیری، فایروال و همچنین سیاست‌ها و رویه‌های سازمانی است.

۸. Secure Coding Practices (روش‌های کدنویسی امن): با استفاده از روش‌های کدنویسی امن، می‌توان مشکلات امنیتی مرتبط با برنامه‌نویسی را کاهش داد. این شامل مواردی مانند اعتبارسنجی و فیلتر کردن ورودی‌ها، جلوگیری از حملات راه اندازهای مخرب (مانند حق‌تقلبی) و جلوگیری از نفوذ از طریق آسیب‌پذیری‌های رایج است.

۹. Security Testing (آزمون امنیتی): با اجرای آزمون‌های امنیتی، می‌توان به شناسایی آسیب‌پذیری‌ها و نقاط ضعف امنیتی پرداخت و اطمینان حاصل کرد که سیستم نرم افزاری مقاوم در برابر حملات است. این شامل آزمون نفوذ، آزمون نفوذ خودکار، آزمون آسیب‌پذیری و سایر فعالیت‌های مرتبط با آزمون امنیتی است.

تمامی این موارد همراه با abuse case می‌توانند به طراحان نرم افزار کمک کنند تا نیازها و اهداف کاربران را در نظر بگیرند، ریسک‌های امنیتی را شناسایی و ارزیابی کنند و تدابیر امنیتی مناسبی را پیاده سازی کنند تا یک نرم افزار ایمن و محافظه‌کار را به وجود بیاورند.

در آینده، انتظار می‌رود که استفاده از abuse case ها در طراحی نرم افزار و تضمین امنیت آن به موارد زیر منجر شود:

۱. توجه به تهدیدات جدید: با توسعه فناوری‌ها و تغییرات در محیط عملکردی نرم افزار، تهدیدات امنیتی نیز پیچیده‌تر و متنوع‌تر می‌شوند. استفاده از abuse case ها به طراحان نرم افزار کمک می‌کند تا به تهدیدات جدید و نوع جدیدی از حملات پاسخ دهند و به مواجهه با آنها آماده باشند.

۲. امنیت بیشتر: استفاده مداوم از abuse case ها در طراحی و توسعه نرم افزار منجر به بهبود امنیت کلی نرم افزار می‌شود. این موضوع می‌تواند به توسعه نرم افزارهایی کمک کند که دارای ساختار امنیتی استوارتر و اهداف مقاومتی قوی‌تری باشند.

۳. آگاهی بیشتر از ریسک‌ها: با استفاده از abuse case ها، تیم‌های طراحی و توسعه نرم افزار با ریسک‌های امنیتی محتمل در سیستم آشنا می‌شوند. این آگاهی به آنها کمک می‌کند تا مراقبت‌های لازم را انجام دهند و راهکارهای امنیتی مطلوب را برای حفاظت از سیستم اعمال کنند.

۴. اعتماد بهتر کاربران: با توجه به نگرانی‌ها و نیازهای روزافزون کاربران در خصوص امنیت نرم افزارها، استفاده از abuse case ها می‌تواند به ایجاد اعتماد بین کاربران و سازندگان نرم افزارها کمک کند. آنها می‌توانند از وجود تلاش‌هایی برای پیشگیری از سواستفاده مطلع شوند و به احساس بیشتری از امنیت و اعتماد در استفاده از نرم افزار برسند.

۵. تحقق تطابق با استانداردها و مقررات: در حوزه هایی مانند حریم خصوصی، امنیت داده‌ها، حوادث سایبری و مقررات قانونی، استفاده از abuse case ها می‌تواند به نرم افزارها کمک کند تا با استانداردها و مقررات مربوطه سازگار شوند و الزامات امنیتی را رعایت کنند.

با توجه به رشد روزافزون تهدیدات امنیتی، استفاده مداوم و بهبود یافته از abuse case ها در طراحی نرم افزارها می‌تواند به حفاظت و ایجاد نرم افزارهایی امن‌تر و مقاوم‌تر در برابر حملات کمک کند.

۴. معرفی risk analysis

اهمیت امنیت نرم افزار از زمانی کشف شد که اکثر حملات به سیستم های نرم افزاری توسط نرم افزارهایی که ضعیف طراحی شده اند، آشکار شد. علاوه بر این، نشان داده شده است که هر چه زودتر امنیت در یک سیستم نرم افزاری وارد شود، از نظر تلاش و هزینه بهتر خواهد بود بنابراین، در مقایسه با الگوهای طراحی، که هدف آنها ساختن نرم افزار به خوبی ساختار یافته و قابل استفاده مجدد است، الگوهای امنیتی پیشنهاد شده اند که هدفشان تحمیل یک مقداری از امنیت در مرحله طراحی است.

یکی از اهداف منطقی تحقیق تخمین امنیت تحمیل شده در یک سیستم نرم افزاری با بررسی اینکه کدام الگوهای امنیتی مورد استفاده قرار می گیرند و در کجای طراحی قرار دارند، خواهد بود. برای دستیابی به این هدف، ابتدا تعیین میشود که چندین الگوی امنیتی تا چه حد در برابر دسته های شناخته شده حملات مقاوم هستند.

مراحل اصلی risk analysis در طراحی امن نرم افزار عبارتند از:

۱. شناسایی ریسک ها: در این مرحله، تیم امنیتی باید به دقت نقاط ضعف احتمالی در نرم افزار را شناسایی کند. این شناسایی می‌تواند شامل آسیب‌پذیری های امنیتی محتمل، حملات ممکن، نقاط ضعف در طراحی، پیاده سازی یا مستندات باشد. روش‌های مختلفی مانند مرور کد (Code Review)، تحلیل ریسک تهدید (Threat Risk Analysis) و استفاده از پروتکل های امنیتی مانند OWASP Top 10 برای شناسایی ریسک ها استفاده میشوند.

۲. ارزیابی ریسک: در این مرحله، ریسک های شناسایی شده ارزیابی می‌شوند. این ارزیابی شامل ارزش‌گذاری ریسک ها بر اساس احتمال و اثر آنها است. احتمال حمله و اثرات مختلف ممکن از جمله از دست دادن داده‌ها، قطعی سرویس، خسارت مالی و آسیب به اعتبار مشتریان مورد بررسی قرار می‌گیرد.

۳. مدیریت ریسک: در این مرحله، اقدامات مناسبی برای مدیریت ریسک ها انتخاب و اعمال می‌شوند. این اقدامات می‌تواند شامل پیشگیری، کاهش، انتقال و قبول ریسک باشد. مثلا به روزرسانی و تعمیر آسیب‌پذیری‌ها، استفاده از مکانیزم‌های احراز هویت و دسترسی، رمزنگاری داده ها و مانیتورینگ مداوم جهت شناسایی حملات می باشد.

۴. ثبت و پیگیری ریسک: در این مرحله، ریسک های شناسایی شده و اقدامات انجام شده برای مدیریت آنها در یک سیستم ثبت می‌شوند و پیگیری انجام می‌شود. این فرآیند ممکن است شامل داشتن گزارشات و مستندات مربوطه، ارتباط با تیم های امنیتی و آپدیت های منظم باشد.

در risk analysis سناریوهای زیر تحلیل می شوند تا با رفع این موارد نرم افزار ایمنی داشته باشند:

- حمله Man-In-The-Middle: این حمله ارتباط بین دو سیستم را قطع می کند. مهاجم ارتباط مستقلی با قربانیان برقرار می‌کند و پیام هایی را بین آنها مخابره می‌کند و باعث می شود که آنها باور کنند که مستقیما با یکدیگر صحبت می‌کنند. امضای چنین حمله ای این است که یک ارتباط ناامن بین دو سیستم وجود داشته باشد یا اگر سیستم ها در یک منطقه غیرقابل اعتماد (همانند WiFi های رایگان بدون رمز) ارتباط برقرار کنند.
- حمله Denial-Of-Service (DOS): هدف این حمله این است که یک سیستم یا یکی از منابع کلیدی آن برای کاربران قانونی در دسترس نباشد. حملات DOS فرمت های متفاوتی با امضاهای متفاوت دارند. برخی از ورودی های نامعتبر استفاده می کنند. برخی دیگر به یک سیستم درخواست های زیادی ارسال می کنند تا سیستم کرش کند. امضاهای احتمالی چنین حمله ای عبارتند از:
(الف) یک سیستم در دسترس عموم که از کنترل اعتبار سنجی ورودی (یا فایروال) برای تایید درخواست های دریافتی استفاده نمی کند یا
(ب) یک فایروال که کنترل احراز هویت مناسب را برای فیلتر کردن درخواست ها اجرا نمی کند.
- حمله دستکاری داده ها: اگر داده ها به صورت متن ساده درست شوند، مهاجم می تواند داده ها را در حالت ذخیره سازی، در حین انتقال یا در حین پردازش دستکاری کند. امضاهای احتمالی این حملات عبارتند از:
(الف) یک جز سیستم که در یک میزبان غیرقابل اعتماد عمل می کند (خودی مخرب) یا
(ب) ارسال داده ها به صورت متن ساده بین سیستم ها یا
(ج) عدم وجود رمزنگاری مناسب
- حمله تزریقی: این حمله از فقدان کنترل‌های اعتبارسنجی ورودی برای عبور از ورودی های مخرب سواستفاده می‌کند که می‌توانند برای به دست آوردن امتیازات بالاتر، تغییر داده‌ها یا خراب کردن سیستم مورد استفاده قرار گیرند. انواع مختلف حملات تزریق شامل SQL Injection، OS Command Injection و XML Injection می باشد. امضا این حمله این است که سیستم، فیلتر ورودی مناسبی را روی ورودی های کاربر یا ورودی های سایر اجزای غیرقابل اعتماد، اعمال نمی‌کند.

توسعه معیارهای امنیتی برای استفاده در ارزیابی معماری نرم افزار نیز یک کار بسیار پیچیده است. معیارهای امنیتی مختلف با دامنه کاربرد متفاوت وجود دارد. این موارد عبارتند از: معیارهای تجزیه و تحلیل آسیب پذیری استاتیک، معیارهای تجزیه و تحلیل آسیب پذیری پویا، معیارهای امنیتی معماری استاتیک، و معیارهای امنیتی زمان اجرا. در زیر برخی از معیارهای شناخته شده مورد استفاده در ارزیابی امنیت معماری مورد بحث قرار می گیرند:

۱) معیارهای امنیتی معماری سیستم: این معیارها به ارزیابی سلامت امنیت معماری نرم افزار کمک می کند. این معیارها را می توان برای ارزیابی قرار گرفتن سیستم در معرض حمله با توجه به معماری و جزئیات کد نرم افزار استفاده کرد. این معیار ها عبارتند از:

- معیار سطح حمله: این معیار نسبت سیستمی را که مهاجم می تواند برای حمله به سیستم قربانی استفاده کند، اندازه گیری می کند. این معیار می تواند تعداد روش های سیستمی که داده ها را از محیط نرم افزار دریافت می کنند، تعداد روش هایی که داده ها را به محیط نرم افزار برمی گرداند، تعداد کانال های ارتباطی و تعداد موارد داده غیرقابل اعتماد را اندازه گیری میکند. هر چه مقدار سطح حمله بزرگتر باشد، سیستم بالقوه ناامن یا آسیب پذیرتر است.
- معیار تقسیم بندی: تقسیم بندی به این معنی است که سیستم ها و اجزای آنها در بخش های مختلف و جدا از یکدیگر اجرا می شوند. بنابراین سازش هیچ یک از آنها تاثیری بر دیگران ندارد. این معیار می تواند تعداد نرم افزار های مستقلی که به یکدیگر اعتماد ندارند (احراز هویت و مجوز نرم افزار از سایر اجزای سیستم جدا می باشد) که در سیستم اجرا می شوند، اندازه گیری کند. هر چه مقدار بخش بندی بیشتر باشد، سیستم امن تر است.
- معیار حداقل امتیاز: این معیار بیان می کند که هر جز و کاربر باید حداقل امتیازات لازم برای تکمیل وظایف خود داشته باشد. این معیار را می‌توان از دو نظر ارزیابی کرد: از نظر کنترل‌های امنیتی، می‌تواند امتیازات اعطایی کاربران را بررسی کند. از دیدگاه تحلیل معماری، این مورد می تواند به این صورت ارزیابی شود که چگونه سیستم به حداقل اجزای ممکن تجزیه می شود یعنی تعداد اجزایی که می توانند به داده های حیاتی دسترسی داشته باشند. هرچه این مقدار کوچکتر باشد، سیستم امن تر است.
- معیار شکست ایمن: سیستم هیچ داده ای را که معمولا در صورت خرابی سیستم نباید فاش شود، فاش نمی کند. این شامل داده های سیستم و همچنین داده های مربوط به سیستم می شود. این معیار را می‌توان از روی پاسخ‌های کنترل امنیتی ارزیابی کرد به عنوان مثال نحوه کنترل شدن سیستم در هنگام خرابی. از دیدگاه معماری سیستم، می‌توان آن را به عنوان تعداد ویژگی‌ها و روش‌های حیاتی که در یک جز معین قابل دسترسی هستند، ارزیابی کرد. هر چه مقدار معیار کوچکتر باشد، احتمال دارد سیستم در صورتی که خراب شود باز هم ایمن باشد.

۲) معیارهای معماری امنیتی: این معیارها به ارزیابی سلامت معماری و مکانیسم های امنیتی سیستم از جمله: توابع و اجزای امنیتی، الگوهای امنیتی و کنترل های امنیتی کمک می کند.

NIST (National Institute of Standards and Technology) مجموعه ای از اصول طراحی را معرفی می کند که باید در توسعه سیستم های ایمن اتخاذ شوند. این موارد عبارتند از: استفاده از امنیت

لایه ای، سادگی طراحی امنیتی، حفاظت از اطلاعات در حین پردازش، حمل و نقل و ذخیره سازی و عدم اعتماد به ورودی های خارجی. اصل های امنیتی NIST عبارتند از:

- اصل ۱. یک خط مشی امنیتی صحیح به عنوان بنیاد امنیتی ایجاد کنید:

یک خط مشی امنیتی یک سند مهم برای توسعه در هنگام طراحی یک سیستم اطلاعاتی است. سیاست امنیتی با تعهد اساسی سازمان به امنیت اطلاعات که به عنوان یک بیانیه خط مشی کلی فرموله شده است، آغاز می شود. سپس این خط مشی برای تمام جنبه های طراحی سیستم یا راه حل امنیتی اعمال می شود. این خط مشی اهداف امنیتی (مانند محرمانه بودن، یکپارچگی، در دسترس بودن، مسئولیت پذیری و اطمینان) را که سیستم باید از آن پشتیبانی کند، شناسایی می کند و این اهداف رویه ها، استانداردها و کنترل های مورد استفاده در طراحی معماری امنیت فناوری اطلاعات را هدایت می کند. این سیاست همچنین باید نیازمند تعریف دارایی های حیاتی، تهدید درک شده و نقش ها و مسئولیت های مرتبط با امنیت باشد.

- اصل ۲. امنیت را به عنوان بخشی جدایی ناپذیر از طراحی کلی سیستم در نظر بگیرید:

در طراحی سیستم اطلاعاتی باید امنیت را در نظر گرفت. تجربه نشان داده است که اجرای صحیح و موفقیت آمیز اقدامات امنیتی بعد از توسعه یک سیستم، دشوار و پرهزینه است، بنابراین باید به طور کامل در فرآیند چرخه حیات سیستم ادغام شود. این شامل ایجاد سیاست های امنیتی، درک الزامات امنیتی ناشی از آن، مشارکت در ارزیابی محصولات امنیتی و در نهایت مهندسی، طراحی، پیاده سازی و دفع سیستم است.

- اصل ۳. مرزهای امنیتی فیزیکی و منطقی که توسط سیاست های امنیتی مرتبط کنترل می شوند را به وضوح مشخص کنید:

فناوری اطلاعات در مکان های فیزیکی و منطقی و مرزهایی بین این مکان ها وجود دارد. درک اینکه چه چیزی باید در برابر عوامل خارجی محافظت شود می تواند به اطمینان حاصل شود که اقدامات حفاظتی کافی در جایی که بیشترین تاثیر را دارند اعمال می شود. گاهی اوقات یک مرز امنیتی توسط افراد، اطلاعات و فناوری اطلاعات مرتبط با یک مکان فیزیکی تعریف می شود. اما این واقعیت را نمی توان نادیده گرفت که در یک مکان واحد، بسیاری از سیاست های امنیتی مختلف ممکن است وجود داشته باشد، برخی اطلاعات در دسترس عموم و برخی اطلاعات حساس طبقه بندی نشده را پوشش می دهند. گاهی اوقات یک مرز امنیتی توسط یک سیاست امنیتی تعریف می شود که بر مجموعه خاصی از اطلاعات و فناوری اطلاعات حاکم است که می تواند از مرزهای فیزیکی عبور کند. پیچیدگی بیشتر موضوع این است که در بسیاری از مواقع، یک ماشین یا سرور واحد ممکن است هم اطلاعات دسترسی عمومی و هم اطلاعات حساس طبقه بندی نشده را در خود جای دهد. در نتیجه، چندین سیاست امنیتی ممکن است برای یک ماشین یا در یک سیستم منفرد اعمال شود. بنابراین، هنگام

توسعه یک سیستم اطلاعاتی، مرزهای امنیتی باید در نظر گرفته شود و در اسناد و سیاست های امنیتی سیستم مربوطه ابلاغ شود.

- اصل ۴. اطمینان حاصل کنید که توسعه دهندگان در مورد چگونگی توسعه نرم افزار ایمن آموزش دیده اند:

عاقلانه نیست که فرض کنیم توسعه دهندگان می دانند چگونه نرم افزار ایمن را توسعه دهند. بنابراین، قبل از توسعه سیستم، اطمینان حاصل کنید که توسعه دهندگان به اندازه کافی در زمینه توسعه نرم افزار ایمن آموزش دیده اند. این اصل شامل کاربرد رشته های مهندسی برای طراحی، توسعه، کنترل پیکربندی، و یکپارچه سازی و آزمایش است.

- اصل ۵. ریسک را تا حد قابل قبولی کاهش دهید:

ریسک به عنوان ترکیبی از:

(الف) احتمال اینکه یک منبع تهدید خاص اعمال شود (به طور عمدی یا ناخواسته یک آسیب پذیری سیستم دستورات خاصی را اعمال کند) و

(ب) تاثیر نامطلوب ناشی از آن، بر عملیات سازمانی، دارایی های سازمانی یا افراد تعریف می شود. اجتناب از ریسک، یک هدف رایج امنیت فناوری اطلاعات بود. با درک بهتر ماهیت ریسک، این موضوع تغییر کرد. امروزه مشخص شده است که حذف همه ریسک ها مقرون به صرفه نیست. برای هر کنترل پیشنهادی باید یک تحلیل هزینه و سود انجام شود. در برخی موارد، مزایای یک سیستم امن تر ممکن است هزینه های مستقیم و غیرمستقیم را توجیه نکند. مزایای یک سیستم امن شامل مواردی است که از ضرر مالی جلوگیری می کند. برای مثال، کنترل ها ممکن است برای حفظ اعتماد و اطمینان عمومی ضروری باشند. هزینه های مستقیم شامل هزینه خرید و نصب یک فناوری معین است. هزینه های غیر مستقیم شامل کاهش عملکرد سیستم و آموزش اضافی است. هدف این اصل کاهش ریسک ماموریت/کسب و کار تا سطح قابل قبولی است.

- اصل ۶. فرض کنید سیستم های خارجی ناامن هستند:

اصطلاح دامنه اطلاعاتی از عمل پارتیشن بندی منابع اطلاعاتی بر اساس کنترل دسترسی، نیاز و سطوح حفاظت مورد نیاز ناشی می شود. سازمان ها اقدامات خاصی را برای اجرای این پارتیشن بندی و فراهم کردن جریان عمدی اطلاعات مجاز بین حوزه های اطلاعاتی اجرا می کنند. مرز یک دامنه اطلاعاتی محیط امنیتی آن دامنه را نشان می دهد.

دامنه خارجی دامنه ای است که تحت کنترل شما نیست. به طور کلی، سیستم های خارجی را باید ناامن در نظر گرفت. تا زمانی که یک دامنه خارجی «معتمد» تلقی نشود، مهندسان سیستم، معماران و متخصصان فناوری اطلاعات باید تدابیر امنیتی یک سیستم خارجی را با اقدامات یک سیستم داخلی قابل اعتماد متفاوت فرض کنند و بر این اساس ویژگی‌های امنیتی سیستم را طراحی کنند.

- اصل ۷. معاوضه های بالقوه بین کاهش ریسک و افزایش هزینه ها و کاهش سایر جنبه های اثربخشی عملیاتی را شناسایی کنید:

برای برآورده ساختن الزامات امنیتی اعلام شده، یک طراح سیستم، معمار یا کارشناس امنیتی باید همه نیازهای عملیاتی رقیب را شناسایی و برطرف کند. ممکن است به دلیل سایر الزامات عملیاتی، اصلاح یا تنظیم اهداف امنیتی ضروری باشد. در اصلاح یا تنظیم اهداف امنیتی، پذیرش ریسک و هزینه بیشتر ممکن است اجتناب ناپذیر باشد. با شناسایی و پرداختن به این مبادلات در اسرع وقت، تصمیم گیرندگان فرصت بیشتری خواهند داشت و می توانند به سیستم های موثرتری دست یابند.

- اصل ۸. اجرای تدابیر امنیتی مناسب برای دستیابی به اهداف امنیتی سازمانی:

به طور کلی، اقدامات امنیتی فناوری اطلاعات بر اساس نیازهای منحصر به فرد سازمان تنظیم می شود. در حالی که عوامل متعددی مانند الزامات اصلی ماموریت و راهنمایی باید در نظر گرفته شوند، مسئله اساسی محافظت از ماموریت یا تجارت در برابر تاثیرات منفی مرتبط با امنیت فناوری اطلاعات است. از آنجایی که نیازهای امنیتی فناوری اطلاعات یکسان نیستند، طراحان سیستم و متخصصان امنیت باید سطح اعتماد را هنگام اتصال به سایر شبکه‌های خارجی و زیر دامنه‌های داخلی در نظر بگیرند. شناخت منحصر به فرد بودن هر سیستم اجازه می دهد تا از یک استراتژی امنیتی لایه ای استفاده شود.

- اصل ۹. از اطلاعات در حین پردازش، حمل و نقل و ذخیره سازی محافظت کنید:

خطر تغییر یا تخریب غیرمجاز داده‌ها، افشای اطلاعات و ممانعت از دسترسی به داده‌ها در حین انتقال باید همراه با خطرات مرتبط با داده‌هایی که در ذخیره‌سازی یا در حال پردازش هستند در نظر گرفته شود. بنابراین مهندسان سیستم، معماران و متخصصان فناوری اطلاعات باید اقدامات امنیتی را برای حفظ یکپارچگی، محرمانه بودن و در دسترس بودن داده‌ها از جمله نرم‌افزار کاربردی در حین پردازش، حمل و نقل و ذخیره سازی در صورت لزوم اجرا کنند.

- اصل ۱۰. برای دستیابی به امنیت کافی، محصولات سفارشی را در نظر بگیرید:

طراحان باید بدانند که در برخی موارد نمی‌توان با سیستم‌هایی که به طور کامل از محصولات COTS ساخته شده‌اند، اهداف امنیتی را برآورده کرد. در چنین مواردی افزایش COTS با مکانیسم های غیر COTS ضروری خواهد بود.

- اصل ۱۱. در برابر تمام حملات احتمالی از سیستم محافظت کنید:

در طراحی کنترل‌های امنیتی، باید چندین کلاس از «حملات» در نظر گرفته شود. آن دسته از طبقاتی که منجر به ریسک غیرقابل قبول می شوند باید کاهش یابند. نمونه‌هایی از کلاس‌های «حمله» عبارتند از: نظارت غیرفعال، حملات شبکه فعال، بهره‌برداری توسط افراد داخلی، حملاتی که نیاز به دسترسی فیزیکی یا نزدیکی دارند، درج درهای پشتی و کدهای مخرب در طول توسعه و/یا توزیع نرم‌افزار.

- اصل ۱۲. در صورت امکان، امنیت را بر اساس استانداردهای از پیش تعریف شده قرار دهید:

بیشتر سازمان ها برای انجام ماموریت یا تجارت خود به طور قابل توجهی به سیستم های اطلاعاتی توزیع شده وابسته هستند. این سیستم ها، اطلاعات را هم در سازمان خود و هم بین سازمان های دیگر توزیع می کنند. برای اینکه قابلیت‌های امنیتی در چنین محیط‌هایی موثر باشد، طراحان برنامه‌های امنیتی باید تمام تلاش خود را به کار گیرند تا قابلیت همکاری و قابلیت حمل را در تمام اقدامات امنیتی، از جمله سخت افزار و نرم افزار و شیوه های پیاده سازی بگنجانند.

- اصل ۱۳. از زبان مشترک در توسعه الزامات امنیتی استفاده کنید:

استفاده از یک زبان مشترک هنگام توسعه الزامات امنیتی به سازمان‌ها اجازه می‌دهد تا محصولات و ویژگی‌های امنیتی ارزیابی شده در یک محیط آزمایشی مشترک را ارزیابی و مقایسه کنند. هنگامی که یک فرآیند ارزیابی "مشترک" بر اساس الزامات یا معیارهای مشترک استوار باشد، سطحی از اطمینان می تواند ایجاد شود که تضمین کند عملکردهای امنیتی محصول با الزامات امنیتی سازمان مطابقت دارد. معیارهای مشترک منبعی از عبارات مشترک برای نیازهای مشترک فراهم می کند و از یک روش ارزیابی مشترک پشتیبانی می کند.

- اصل ۱۴. اقدامات امنیتی طراحی کنید تا امکان پذیرش منظم فناوری جدید، از جمله فرآیند ارتقای فناوری امن و منطقی را فراهم کند:

با تغییر ماموریت و فرآیندهای تجاری و محیط تهدید، الزامات امنیتی و روش های حفاظت فنی باید به روز شوند. خطرات مرتبط با فناوری اطلاعات برای ماموریت/کسب و کار در طول زمان متفاوت است و تحت ارزیابی دوره ای قرار می گیرد. ارزیابی دوره ای باید انجام شود تا طراحان و مدیران سیستم را قادر سازد تا تصمیمات مدیریت ریسک آگاهانه را در مورد پذیرش یا کاهش ریسک های شناسایی شده با تغییرات یا به روز رسانی در قابلیت امنیتی اتخاذ کنند. عدم شناسایی به موقع از طریق ارزیابی مجدد راه حل های امنیتی منسجم و اصلاح آسیب پذیری های فناوری اطلاعات در حال تحول و کاربردی منجر به اعتماد کاذب و افزایش خطر می شود. هر مکانیزم امنیتی باید بتواند از انتقال به فناوری جدید یا ارتقا ویژگی های جدید بدون نیاز به طراحی مجدد سیستم پشتیبانی کند. طراحی امنیتی باید ماژولار باشد تا بتوان بخش‌های جداگانه طراحی امنیتی را بدون نیاز به اصلاح کل سیستم ارتقا داد.

- اصل ۱۵. استفاده از کنترل امنیتی ایجاد شده راحت باشد:

هر چه حفظ و اجرای یک کنترل امنیتی دشوارتر باشد، احتمالا این کنترل کمتر موثر خواهد بود بنابراین کنترل های امنیتی باید به گونه ای طراحی شوند که با مفهوم عملیات و سهولت استفاده به عنوان یک ملاحظه مهم سازگار باشد. تجربه و تخصص مدیران و کاربران باید متناسب با عملکرد کنترل امنیتی باشد. یک سازمان باید منابع لازم را برای اطمینان از آموزش صحیح مدیران و کاربران

سیستم سرمایه گذاری کند. علاوه بر این، مقرون به صرفه بودن کنترل امنیتی، هزینه‌های آموزش مدیر و کاربر همراه با هزینه‌های عملیاتی چرخه عمر باید در نظر گرفته شود.

- اصل ۱۶. امنیت لایه‌ای را پیاده‌سازی کنید (مطمئن شوید که هیچ نقطه آسیب‌پذیری وجود ندارد):
طرح‌های امنیتی باید یک رویکرد لایه‌ای را برای مقابله یا محافظت در برابر یک تهدید خاص یا کاهش آسیب‌پذیری در نظر بگیرند. به عنوان مثال استفاده از یک مسیریاب فیلترینگ بسته همراه با یک دروازه برنامه و یک سیستم تشخیص نفوذ برای افزایش ضریب کاری که مهاجم باید برای حمله موفقیت آمیز به سیستم صرف کند، ترکیب می شود. افزودن کنترل های رمز عبور خوب و آموزش کافی کاربر، وضعیت امنیتی سیستم را حتی بیشتر بهبود می بخشد. نیاز به محافظ های لایه ای به ویژه هنگامی که از محصولات تجاری خارج از قفسه (COTS) استفاده می شود مهم است. تجربه عملی نشان داده است که پیشرفته ترین کیفیت امنیتی در محصولات COTS درجه بالایی از محافظت در برابر حملات پیچیده را ارائه نمی دهد. می توان با قرار دادن چندین کنترل به صورت سری، به کاهش این وضعیت کمک کرد، که مستلزم کار اضافی توسط مهاجمان برای دستیابی به اهداف خود است.
- اصل ۱۷. طراحی و راه اندازی یک سیستم فناوری اطلاعات برای محدود کردن آسیب و انعطاف پذیری در پاسخ:
سیستم های اطلاعاتی باید در برابر حمله مقاوم باشند، آسیب را محدود کنند و در صورت وقوع حملات باید به سرعت بهبود یابند. اصل پیشنهاد داده شده در اینجا نیاز به فناوری‌های حفاظتی کافی در همه سطوح برای اطمینان از مقابله موثر با هرگونه حمله سایبری احتمالی را تشخیص می‌دهد. آسیب‌پذیری‌هایی هستند که قابل رفع نیستند، آسیب‌پذیری‌هایی هستند که هنوز رفع نشده‌اند، آنهایی که ناشناخته هستند و آنهایی که نمی‌توان آنها را برطرف کرد (به عنوان مثال، سرویس‌های مخاطره‌آمیز از طریق فایروال‌ها) تا امکان افزایش قابلیت‌های عملیاتی را فراهم کنند. علاوه بر دستیابی به یک حالت اولیه ایمن، سیستم های ایمن باید بعد از خرابی وضعیت کاملا مشخصی داشته باشند یا از طریق یک روش بازیابی به یک وضعیت امن شناخته شده بروند. سازمان‌ها باید قابلیت‌های شناسایی و پاسخ دهی ایجاد کنند، نقاط ضعف واحد را در سیستم‌های خود مدیریت کنند و یک استراتژی گزارش دهی و پاسخ را پیاده‌سازی کنند.
- اصل ۱۸. اطمینان حاصل کنید که سیستم مقاوم است و مقاوم می ماند:
این انتظارات را معمولا می توان به عنوان ارائه مقاومت کافی در برابر نفوذ مستقیم و تلاش برای دور زدن کنترل های امنیتی خلاصه کرد. درک خوب محیط تهدید، ارزیابی مجموعه‌های نیازمندی‌ها در رشته‌های مهندسی سخت افزار و نرم افزار و ارزیابی محصول و سیستم، اقدامات اولیه‌ای هستند که برای دستیابی به اطمینان استفاده می‌شوند. علاوه بر این، مستندسازی تهدیدات خاص و در حال

تحول در ایجاد تنظیمات به موقع در امنیت کاربردی و حمایت استراتژیک از افزایش پیشرفت‌های امنیتی مهم است.

- اصل ۱۹. آسیب‌پذیری ها را محدود کنید:

اگر آسیب‌پذیری وجود داشته باشد، آسیب می‌تواند محدود شود و به سایر عناصر سیستم اطلاعاتی اجازه دهد تا به درستی کار کنند.

- اصل ۲۰. سیستم های دسترسی عمومی را از منابع حیاتی ماموریت (مانند داده ها، فرآیند ها و...) جدا کنید:

در حالی که گرایش به سمت زیرساخت های مشترک در بسیاری از موارد دارای شایستگی قابل توجهی است، اما به طور کلی قابل اجرا نیست. در مواردی که حساسیت یا بحرانی بودن اطلاعات زیاد است، سازمان‌ها ممکن است بخواهند تعداد سیستم‌هایی را که داده‌ها در آنها ذخیره می‌شود را محدود کرده و آنها را از نظر فیزیکی یا منطقی جدا کنند. جداسازی فیزیکی ممکن است شامل اطمینان از عدم وجود ارتباط فیزیکی بین منابع اطلاعاتی دسترسی عمومی سازمان و اطلاعات حیاتی سازمان باشد. هنگام پیاده سازی راه حل‌های جداسازی منطقی، لایه‌هایی از سرویس‌های امنیتی و مکانیسم‌ها باید بین سیستم‌های عمومی و سیستم‌های امن مسئول حفاظت از منابع حیاتی ماموریت، ایجاد شود. لایه های امنیتی ممکن است شامل استفاده از طرح های معماری شبکه مانند مناطق غیرنظامی و زیرشبکه های غربال شده باشد. در نهایت، طراحان و مدیران سیستم باید سیاست ها و رویه های امنیتی سازمانی را در مورد استفاده از سیستم های دسترسی عمومی اعمال کنند.

- اصل ۲۱. از مکانیسم های مرزی برای جداسازی سیستم های محاسباتی و زیرساخت های شبکه استفاده کنید:

برای کنترل جریان اطلاعات و دسترسی در سراسر مرزهای شبکه در زیرساخت‌های محاسباتی و ارتباطی و برای اعمال جداسازی مناسب گروه‌های کاربر، باید از مجموعه‌ای از دستگاه های کنترل دسترسی و سیاست های کنترل دسترسی همراه استفاده شود.

موارد زیر را برای ارتباطات در سراسر مرزهای شبکه تعیین کنید:

- چه رابط های خارجی مورد نیاز است.
- این که آیا اطلاعات تحت فشار قرار می گیرند یا کشیده می شوند.
- پورت ها، پروتکل ها و خدمات شبکه مورد نیاز است.
- چه الزاماتی برای تبادل اطلاعات سیستم وجود دارد.

- اصل ۲۲. طراحی و اجرای مکانیزم های حسابرسی برای تشخیص استفاده غیرمجاز:

سازمان ها باید گزارش های حسابرسی را برای شناسایی استفاده غیرمجاز و اطمینان از عملکرد صحیح منابع سیستم، نظارت و ثبت را به طور دوره ای بررسی کنند. در برخی موارد، ممکن است از سازمان ها خواسته شود که اطلاعات به دست آمده از طریق مکانیسم های حسابرسی را برای اشخاص

ثالث مناسب، از جمله مقامات مجری قانون یا متقاضیان قانون آزادی اطلاعات (FOIA) افشا کنند. بسیاری از سازمان‌ها رضایت خود را برای نظارت بر این سیاست‌ها اعمال کرده‌اند که بیان می‌کنند شواهد استفاده غیرمجاز (مثلا مسیرهای حسابرسی) ممکن است برای پشتیبانی از تحقیقات اداری یا جنایی مورد استفاده قرار گیرد.

- اصل ۲۳. برای اطمینان از در دسترس بودن مناسب، روش‌های بازیابی اضطراری را توسعه و اعمال کنید:

تداوم طرح های عملیات یا رویه های بازیابی فاجعه به تداوم عملیات سازمان در صورت بروز فاجعه یا قطع خدمات طولانی مدت که بر ماموریت سازمان تاثیر می‌گذارد، می‌پردازند. چنین طرح هایی باید به مرحله واکنش اضطراری، مرحله بازیابی و بازگشت به مرحله عملیات عادی بپردازند. مسئولیت های پرسنل در طول یک حادثه و منابع موجود باید شناسایی شود. در واقع، طرح های بازیابی اضطراری و بلایای طبیعی به هر سناریو یا فرضیه‌ای نمی‌پردازند بلکه بر روی رویدادهایی که احتمال وقوع دارند تمرکز می کنند و روش قابل قبولی را برای بهبودی سیستم شناسایی می کنند. به طور دوره ای، برنامه ها و رویه ها باید به کار گرفته شوند تا اطمینان حاصل شود که آنها موثر و به خوبی درک شده اند.

- اصل ۲۴. برای سادگی تلاش کنید:

هرچه مکانیسم پیچیده تر باشد، احتمال دارد که دارای نقص های بیشتری باشد. مکانیسم های ساده معمولا نقص‌های قابل بهره برداری کمتری دارند و نیاز به نگهداری کمتری دارند. علاوه بر این از آنجایی که مسائل مدیریت پیکربندی ساده شده است، به روزرسانی یا جایگزینی یک مکانیسم ساده فرآیندی ساده و آسان تری دارد.

- اصل ۲۵. عناصر سیستم مورد اعتماد را به حداقل برسانید:

اقدامات امنیتی شامل افراد، عملیات و فناوری است. در جایی که از فناوری استفاده می شود سخت افزار، سیستم عامل و نرم افزار باید طوری طراحی و پیاده سازی شوند که برای حفظ حفاظت، نیاز به حداقل تعداد عناصر سیستم مورد اعتماد باشد. علاوه بر این، برای اطمینان از تایید مقرون به صرفه بودن ویژگی های امنیتی سیستم، مهم است که مقدار نرم افزار و سخت افزار مورد انتظار برای ارائه امن ترین عملکردها به حداقل رسانده شود.

- اصل ۲۶. کمترین امتیاز را اجرا کنید:

مفهوم محدود کردن دسترسی، یا «حداقل امتیاز»، صرفا به این معناست که برای انجام عملکردهای مورد نیاز، مجوزی بیش از حد لازم ارائه نشود. این شاید اغلب در مدیریت سیستم اعمال شود.

هدف آن کاهش ریسک با محدود کردن تعداد افرادی است که به کنترل‌های امنیتی سیستم حیاتی دسترسی دارند. به عنوان مثال، کنترل افرادی که مجاز به فعال یا غیرفعال کردن ویژگی های امنیتی سیستم یا تغییر امتیازات کاربران یا برنامه ها هستند. بهترین روش نشان می دهد که بهتر است

چندین مدیر با دسترسی محدود به منابع امنیتی به جای یک نفر با مجوزهای "فوق کاربر" داشته باشید.

باید به اجرای کنترل های دسترسی مبتنی بر نقش برای جنبه های مختلف استفاده از سیستم، نه تنها مدیریت، توجه شود. سیاست امنیتی سیستم می تواند نقش های مختلف کاربران یا فرآیندها را شناسایی و تعریف کند. به هر نقش مجوزهای مورد نیاز برای انجام وظایف خود اختصاص داده میشود. هر مجوز یک دسترسی مجاز به یک منبع خاص را مشخص می کند (مانند دسترسی "خواندن" و "نوشتن" به یک فایل یا دایرکتوری مشخص، دسترسی "اتصال" به یک میزبان و پورت معین و..). تا زمانی که مجوز به طور صریح داده نشده باشد، کاربر یا فرآیند نباید به منبع محافظت شده دسترسی داشته باشد. علاوه بر این، نقش‌ها یا مسئولیت هایی را که برای اهداف امنیتی باید جدا باقی بمانند، مشخص کنید، که معمولا به آن «تفکیک وظایف» می‌گویند.

- اصل ۲۷. مکانیسم های امنیتی غیر ضروری را اجرا نکنید:

هر مکانیزم امنیتی باید از یک سرویس امنیتی یا مجموعه ای از خدمات، پشتیبانی کند و هر سرویس امنیتی باید از یک یا چند هدف امنیتی پشتیبانی کند. اگر اقدامات اضافی از یک سرویس یا هدف امنیتی شناخته شده پشتیبانی نکند، نباید اجرا شود. چنین مکانیزم هایی می‌توانند پیچیدگی‌های غیرضروری را به سیستم اضافه کنند و منابع بالقوه آسیب‌پذیری های اضافی هستند.

یک مثال: رمزگذاری فایل است که از سرویس کنترل دسترسی پشتیبانی می کند که به نوبه خود از اهداف محرمانه بودن و یکپارچگی با جلوگیری از دسترسی غیرمجاز به فایل پشتیبانی می کند. اگر رمزگذاری فایل بخشی ضروری برای دستیابی به اهداف باشد، مکانیسم مناسب است. با این حال، اگر این اهداف امنیتی به اندازه کافی بدون گنجاندن رمزگذاری فایل پشتیبانی شوند، آن مکانیسم یک پیچیدگی سیستم غیر ضروری خواهد بود.

- اصل ۲۸. از امنیت مناسب در خاموش کردن یا دفع یک سیستم اطمینان حاصل کنید:

اگر چه ممکن است یک سیستم خاموش شود ولی اطلاعات حیاتی همچنان در سیستم وجود دارد و می تواند توسط یک کاربر یا سازمان غیرمجاز بازیابی شود. دسترسی به سیستم های اطلاعاتی حیاتی باید همیشه کنترل شود.

در پایان چرخه حیات یک سیستم، طراحان سیستم باید رویه هایی را برای دفع دارایی های یک سیستم اطلاعاتی به شیوه ای مناسب و ایمن ایجاد کنند. رویه هایی باید اجرا شوند تا اطمینان حاصل شود که هارد دیسک های سیستم، حافظه فرار و سایر رسانه ها تا حد قابل قبولی پاک می‌شوند و اطلاعات باقی مانده را حفظ نمی‌کنند.

- اصل ۲۹. شناسایی و جلوگیری از خطا ها و آسیب پذیری های رایج:

بسیاری از خطا ها به وفور تکرار می شوند. خطاهایی مانند سرریز شدن بافر، شرایط مسابقه، خطا های قالب بندی رشته ها، بررسی نکردن اعتبار ورودی و امتیازات بیش از حد به برنامه ها داده می شود. یادگیری از گذشته نتایج آینده را بهبود می بخشد.

- اصل ۳۰. امنیت را از طریق ترکیبی از اقدامات توزیع شده به صورت فیزیکی و منطقی اجرا کنید:

اغلب یک سرویس امنیتی واحد با همکاری عناصر موجود در ماشین های جداگانه به دست می‌آید. به عنوان مثال، احراز هویت سیستم معمولا با استفاده از عناصری از رابط کاربری در ایستگاه کاری از طریق عناصر شبکه تا یک برنامه کاربردی در سرور تایید اعتبار انجام می شود. مهم است که همه عناصر را با خدمات امنیتی که ارائه می کنند مرتبط کنیم. این مولفه ها احتمالا برای دستیابی به امنیت در بین سیستم ها به اشتراک گذاشته می شوند، زیرا منابع زیرساختی با توجه به بودجه عملیاتی تامین می شوند.

- اصل ۳۱. تدابیر امنیتی را برای رسیدگی به حوزه های اطلاعاتی متعددی که همپوشانی دارند، تدوین کنید:

حوزه اطلاعاتی مجموعه ای از موجودیت های فعال (شخص، فرآیند یا دستگاه) و اشیا داده آنهاست. ممکن است یک دامنه اطلاعاتی مشمول چندین خط مشی امنیتی باشد. یک خط مشی امنیتی واحد ممکن است دامنه های اطلاعاتی متعددی را در برگیرد. یک قابلیت امنیتی کارآمد و مقرون به صرفه باید بتواند سیاست‌های امنیتی متعددی را برای محافظت از حوزه های اطلاعاتی متعدد بدون نیاز به جداسازی فیزیکی اطلاعات و سیستم های اطلاعاتی مربوطه که داده ها را پردازش می‌کنند، اعمال کند. این اصل برای دور شدن از رویه سنتی ایجاد شبکه های محلی و زیرساخت های جداگانه برای سطوح مختلف حساسیت (به عنوان مثال، طبقه‌بندی امنیتی یا عملکرد تجاری مانند توسعه پیشنهاد) و حرکت به سمت راه‌حل هایی است که امکان استفاده از زیرساخت های مشترک و با حفاظت مناسب در سطح سیستم عامل، برنامه و ایستگاه کاری را فراهم می‌کند.

علاوه بر این برای انجام ماموریت ها و حفاظت از عملکرد های حیاتی، سازمان های دولتی و بخش خصوصی انواع مختلفی از اطلاعات را برای محافظت در اختیار دارند. با در نظر گرفتن این اصل، مهندسان سیستم، معماران و متخصصان فناوری اطلاعات باید قابلیت امنیتی ایجاد کنند که به سازمان هایی با سطوح مختلف حساسیت اطلاعاتی، اجازه دهد تا به اهداف امنیتی اساسی به شیوه‌ای کارآمد دست یابند.

- اصل ۳۲. احراز هویت کاربران و فرآیند ها برای اطمینان از تصمیمات کنترل دسترسی مناسب در داخل و بین دامنه ها:

احراز هویت فرآیندی است که در آن یک سیستم تایید اعتبار یک انتقال، پیام یا ابزاری را برای تایید واجد شرایط بودن یک فرد، فرآیند یا ماشین برای انجام یک اقدام مورد نظر تعیین می‌کند و در نتیجه

تضمین می‌کند که امنیت توسط یک غیرقابل اعتماد به خطر نیفتد. منبع برای اجرای سیاست های امنیتی و دستیابی به اهداف امنیتی ضروری است که احراز هویت کافی به دست آید. علاوه بر این، سطح اعتماد همیشه در هنگام برخورد با تعاملات بین دامنه ای یک مسئله است. راه حل این است که یک خط مشی احراز هویت ایجاد کنید و در صورت لزوم آن را در تعاملات بین دامنه ای اعمال کنید.

توجه: یک کاربر ممکن است حق استفاده از بیش از یک نام را در چندین دامنه داشته باشد. علاوه بر این، حقوق ممکن است در میان دامنه ها متفاوت باشد که به طور بالقوه منجر به نقض خط مشی امنیتی می شود.

- اصل ۳۳. برای اطمینان از مسئولیت پذیری از هویت های منحصر به فرد استفاده کنید:

یک هویت ممکن است یک کاربر واقعی یا یک فرآیند با هویت خاص خود را نشان دهد به عنوان مثال، برنامه ای که دسترسی از راه دور را انجام می دهد. هویت‌های منحصر به فرد، یک عنصر ضروری هستند تا بتوانید:

- مسئولیت پذیری و ردیابی یک کاربر یا فرآیند را حفظ کنید.
- حقوق خاصی را به یک کاربر یا فرآیند اختصاص دهید.
- عدم انکار را فراهم کنید.
- تصمیمات کنترل دسترسی را اجرا کنید.
- هویت یک همتا را در یک مسیر ارتباطی امن ایجاد کنید.
- از ظاهر شدن کاربران غیرمجاز به عنوان یک کاربر مجاز جلوگیری کنید.

risk analysis در طراحی امن نرم افزار در موارد زیر می‌تواند مورد استفاده قرار بگیرد:

۱. در مرحله طراحی اولیه: از risk analysis در مرحله اولیه طراحی نرم افزار استفاده می‌شود تا ریسک های امنیتی احتمالی شناسایی شوند و در طراحی امنیتی سیستم در نظر گرفته شوند. این کمک می‌کند تا از ابتدا مسائل امنیتی مربوط به نرم افزار را مدنظر قرار داده و اقدامات مورد نیاز را اعمال کنید.

۲. در مرحله ارزیابی امنیت: هنگامی که نرم افزار در دسترس قرار گرفته و قبل از عرضه به کاربران، می‌توان از risk analysis برای ارزیابی امنیت نهایی استفاده کرد. با انجام risk analysis، می‌توان نقاط ضعف امنیتی را شناسایی و برنامه های بهبود را برای آنها اجرا کرد.

۳. در صورت تغییرات: هرگاه که در نرم افزار تغییراتی صورت می‌گیرد، می‌توان از risk analysis استفاده کرد تا تاثیر تغییرات بر امنیت نرم افزار را بررسی کنید. این کمک می‌کند تا اطمینان حاصل شود که تغییرات جدید هیچ ریسک امنیتی جدیدی ایجاد نمی‌کنند و تدابیر امنیتی مناسب برای مقابله با هر تغییر اعمال می‌شود.

۴. در واکنش به رویداد های امنیتی: در صورت وقوع رویداد های امنیتی مثل نفوذ یا حملات، risk analysis می‌تواند به تیم های امنیتی کمک کند تا ریسک های امنیتی جدید را شناسایی و اقدامات امنیتی فوری را به منظور جلوگیری از وقوع مشکلات بیشتر اتخاذ کنند.

risk analysis در واقع یک روش برای شناسایی و ارزیابی ریسک های امنیتی است و به طور مستقیم حملات را انجام نمی‌دهد. به عبارتی، risk analysis در حد ارزیابی ریسک های امنیتی و تعیین اقدامات امنیتی برای کاهش آنها مورد استفاده قرار می‌گیرد.

اما با استفاده از اطلاعات حاصل از تحلیل ریسک، حملات مختلفی ممکن است انجام شود. برخی از حملات معروف که می‌توانند براساس نقاط ضعف و ریسک های امنیتی شناسایی شده از روش risk analysis برنامه‌ریزی شوند، عبارتند از:

۱. حملات نفوذ (Intrusion Attacks): با استفاده از آسیب‌پذیری ها و نقاط ضعف کشف شده در طراحی و پیاده سازی نرم افزار، حمله کنندگان می توانند به سیستم نفوذ کنند و دسترسی غیر مجاز به منابع و اطلاعات محرمانه را به دست آورند.

۲. حملات ردپاگذاری (Reconnaissance Attacks): اطلاعاتی که از نتیجه عملیات risk analysis به دست آمده است، ممکن است برای جمع آوری اطلاعات و ردپاگذاری در نظر گرفته شوند. حمله کنندگان با استفاده از این اطلاعات، سیستم را بررسی و به دنبال ضعف ها و نقاط ضعف امنیتی می‌گردند.

۳. حملات رمزگشایی (Cryptanalysis Attacks): اگر رمزنگاری مورد استفاده در نرم افزار ضعیف باشد و در ریسک های امنیتی شناسایی شده باشد، حمله‌کنندگان ممکن است سیستم را هدف قرار داده و با تلاش برای شکست رمزگشایی، دسترسی غیرمجاز به اطلاعات را به دست آورند.

۴. حملات ردیابی (Tracking Attacks): اگر در تحلیل ریسک به نقاط ضعف مربوط به حفظ حریم خصوصی و ردیابی کاربران پی برده شود، حمله کنندگان ممکن است با استفاده از این نقاط ضعف، کاربران را ردیابی و اطلاعات شخصی آنها را جمع‌آوری کنند.

مهم است به یاد داشته باشید که روش risk analysis به منظور شناسایی ریسک‌ها و برنامه‌ریزی برای مقابله با آنها استفاده می‌شود و حملات خاص را مستقیما انجام نمی‌دهد. انجام اقدامات امنیتی مناسب و استفاده از روش‌های حفاظتی مناسب می‌تواند به کاهش ریسک حملات و حفظ امنیت نرم افزار کمک کند.

به طور کلی استفاده از risk analysis در طراحی امن نرم افزار در شناسایی، ارزیابی و مقابله با ریسک های امنیتی در طول عمر نرم افزار بسیار مفید است.

به جز risk analysis، استفاده از موارد زیر می‌تواند به بهبود امنیت مربوط به طراحی نرم افزار کمک کند:

۱. Threat Modeling (مدل سازی تهدیدات): این روش برای شناسایی و تحلیل تهدیدات امنیتی در طراحی نرم افزار استفاده می‌شود. در مدل سازی تهدیدات، تهدیدات محتمل به نرم افزار شناسایی شده، ارزیابی و سپس تدابیر امنیتی مناسب جهت مقابله با آنها اعمال می شوند.

۲. Secure Coding Practices (روش های برنامه نویسی امن): استفاده از روش های برنامه نویسی امن، از جمله استفاده از ورودی تصفیه شده، پیشگیری از حملات حقوق دسترسی (Access Control)، رمزنگاری، اعتبارسنجی و سانسور داده‌ها (Data Sanitization)، به کاهش آسیب‌پذیری ها و حفاظت از نرم افزار در برابر حملات کمک می‌کند.

۳. Security Testing (آزمون امنیتی): با استفاده از تکنیک های آزمون امنیتی، نرم افزار مورد آزمون قرار می‌گیرد تا آسیب‌پذیری ها، ضعف ها و نقاط ضعف امنیتی شناسایی شود. این شامل آزمون نفوذ، آزمون آسیب‌پذیری، آزمون اعتبارسنجی و سایر فعالیت‌های آزمون امنیتی است.

۴. Security Awareness Training (آموزش آگاهی امنیتی): آموزش کارکنان و تیم های مرتبط با نرم افزار در خصوص مسائل امنیتی و بهترین شیوه ها و روش های مقابله با تهدیدات امنیتی می‌تواند به افزایش آگاهی و دانش امنیتی در سازمان کمک کند.

۵. Security Governance (حاکمیت امنیتی): تعیین سیاست ها و رویه های امنیتی سازمان، ارتقا مدیریت امنیتی و تامین منابع لازم برای اجرای اقدامات امنیتی، بهبود امنیت نرم افزار را بهبود می‌بخشد.

۶. Secure Development Lifecycle (چرخه توسعه امن): استفاده از چرخه توسعه امن (Secure SDLC) که شامل مراحلی مانند تحلیل ریسک، طراحی امن، کد نویسی امن، آزمون امنیتی و مراحل نهایی است، به تضمین امنیت در هر مرحله از فرآیند توسعه کمک می‌کند.

استفاده مشترک از risk analysis و موارد فوق می‌تواند به طراحان نرم افزار، در بهبود امنیت محصولات و حفاظت از آنها در برابر تهدیدات امنیتی کمک کند.

از risk analysis برای کارهای آینده می توان انتظارات زیر را داشت:

۱. شناسایی ریسک های جدید: با پیشرفت فناوری و تغییرات در عملکرد سیستم، ریسک های جدید در زمینه امنیت نرم افزار پدید می‌آید. استفاده از risk analysis به طراحان کمک می‌کند تا ریسک های جدید را شناسایی کنند و برنامه ریزی لازم برای مقابله با آنها را انجام دهند.

۲. تحلیل عمیق‌تر ریسک ها: روش های تحلیل ریسک به تیم های امنیتی کمک می‌کند تا ریسک ها را به طور عمیق تر و جامع تر تحلیل کنند. این شامل بررسی عوامل احتمال، اثرات، آسیب پذیری ها، تهدیدات و ارزیابی تاثیر آنها بر سیستم است.

۳. ارتقا استراتژی ها و راهکارها: با تحلیل ریسک ها، تیم های امنیتی قادر خواهند بود استراتژی ها و راهکارهای خود را بروزرسانی کنند. این به آنها اجازه می‌دهد تا رویکرد های بهتری را برای کاهش ریسک ها و پاسخگویی به تهدیدات امنیتی انتخاب کنند.

۴. مدیریت بهتر ریسک ها: با تحلیل ریسک ها، تیم های امنیتی قادر خواهند بود بهترین رویکردها را برای مدیریت ریسک های امنیتی تعیین کنند. این شامل اولویت بندی ریسک ها، انتخاب و پیاده سازی تدابیر امنیتی، طراحی نقشه جامع ریسک و برنامه ریزی برای پاسخگویی به ریسک های بالاتر است.

۵. پیشگیری و پاسخگویی به حملات: تحلیل ریسک به تیم های امنیتی این امکان را می‌دهد که پیشگیری از حملات را افزایش دهند و بهبود بخشند. با شناسایی ریسک های امنیتی و مقابله با آنها، تیم ها می‌توانند برنامه‌ریزی مناسبی برای پاسخگویی به حملات احتمالی انجام دهند و بهبود جریان کار را در مواجهه با رخدادهای امنیتی فراهم کنند.

به طور خلاصه، استفاده مداوم و بهبود یافته از روش‌های تحلیل ریسک در آینده منجر به تشخیص بهتر ریسک‌های امنیتی، ارتقا استراتژی ها و راهکارها، مدیریت بهتر ریسک‌ها و پیشگیری از حملات می‌شود. این بهبودها و تحولات در تحلیل ریسک به تیم های امنیتی اجازه می‌دهد تا بهترین تدابیر امنیتی را انتخاب کنند و به طور موثرتری بر ریسک های امنیتی واکنش نشان دهند.

در آخر به تفاوت بین risk analysis و abuse case اشاره می شود که در زمینه مورد استفاده، تمرکز و هدف آنها است:

۱. استفاده:

risk analysis برای شناسایی، ارزیابی و مدیریت ریسک‌های امنیتی در سیستم ها و نرم افزارها استفاده می‌شود. این روش به طراحان و تیم های امنیتی کمک می‌کند تا ریسک های امنیتی را شناسایی کنند و اقدامات مناسب را برای کاهش آنها برنامه ریزی کنند. از سوی دیگر، abuse case برای شناسایی و توصیف سناریوهای سواستفاده از نرم افزار توسط بازیگران خطرناک استفاده می‌شود. abuse case متمرکز بر شناسایی رفتارها و فعالیت های غیرمجاز است که ممکن است توسط بازیگران خرابکار انجام شود.

۲. تمرکز:

در risk analysis، تمرکز اصلی بر شناسایی و ارزیابی ریسک های امنیتی است. فعالیت‌ها برای تشخیص و سنجش احتمال وقوع ریسک ها و تاثیر آنها بر سیستم و منابع اطلاعاتی مربوطه متمرکز می‌شود. از طرف دیگر، abuse case متمرکز بر شناسایی و توصیف سناریوهای سواستفاده است. این سناریوها به طراحان نرم افزار کمک می‌کند تا رفتارها و فعالیت‌های غیرمجازی که بازیگران خرابکار ممکن است انجام دهند را توصیف کنند.

۳. هدف:

هدف اصلی risk analysis ، کاهش ریسک های امنیتی و بهبود امنیت نرم افزار است. تحلیل ریسک به طراحان امکان می‌دهد تا راهکارها و تدابیر امنیتی مناسب را اعمال کنند تا ریسک ها را به حداقل برسانند. از سوی دیگر، هدف abuse case، توصیف و درک بهتر از رفتارها و فعالیت هایی است که بازیگران خرابکار ممکن است در سیستم انجام دهند. این به طراحان این امکان را می‌دهد تا از ابتدا این رفتارها را در نظر بگیرند و اقدامات مناسب برای پیشگیری و جلوگیری از سواستفاده را اتخاذ کنند.

بنابراین، اگرچه هر دوی risk analysis و abuse case در حوزه امنیت نرم افزار مورد استفاده قرار می‌گیرند، اما هدف و تمرکز آنها متفاوت است. risk analysis بر ریسک های امنیتی تمرکز دارد، در حالی که abuse case بر رفتارهای غیرمجاز و سواستفاده تمرکز دارد.

۵. نتیجه گیری

استفاده از نرم افزارهای ایمن، باعث افزایش قابلیت بهره وری و اطمینان سیستم می شود و همچنان از هزینه های از کار افتادن سیستم به علت اختلال در امنیت و دزدی داده ها می کاهد.

از abuse case در ابتدای ایمن سازی نرم افزار برای جلوگیری از مواردی که باعث سواستفاده میشود، استفاده می کنند.

از risk analysis در مراحل مختلف ساخت نرم افزار استفاده میشود تا امنیت آن افزایش پیدا کند همچنین در فاز به روزرسانی و توسعه نرم افزار نیز برای افزایش کارایی و امنیت، باید به کار گرفته شود.