مهندسی نرم افزار امن

دانشجویان دانشگاه داده پرداز انفورماتیک

ترم 911


بسم الله الرحمن الرحیم

فهرست **صفحه**

مقدمه :

در خودروها، کنترل گوشی های تلفن همراه، سرویس های ارائه شده در بانک‌ها، آب، گاز و پرواز از یک نقطه به نقطه دیگر، از نرم افزارها استفاده شده است.

حتی اگر متوجه نباشیم تمامی این موارد توسط سیستم های پیچیده نرم افزاری در بستر اینترنت محقق شده‌اند.

بنابراین؛

سازماندهی، استقرار، عملیات اجرایی یک نرم افزار بدون در نظر گرفتن مسائل امنیتی، مانند راه رفتن روی سیم بدون توری امنیتی در زیر آن است.

به همین دلیل در این درس موضوع مورد بحث امنیت، تعریف ابعاد مختلف اطمینان و امنیت نرم‌افزار است که به طور فزاینده یک مشکل نرم افزاری است.

فصل اول :

ضرورت امنیت در چرخه حیات :

امروزه ارزیابی امنیتی، تنها پویش پورتهای باز شبکه نبوده و کاوش در رفتار نرم‌افزار به عنوان بخش کلیدی و بحرانی از یک سیستم بسیار ضروری است.

بسیاری از ذینفعان تولید نرم‌افزار پس از تولید و در مرحله بکارگیری نرم‌افزار به فکر امنیت برنامه کاربردی خود می‌افتند و غالباً به انجام یک تست نفوذ بسنده می‌کنند. این موضوع باعث افزایش هزینه برطرف کردن نقص‌های امنیتی، در سطح برنامهٔ کاربردی به صورت موردی و پراکنده با امنیت برخورد کردن، درک نادرست و ناقص از نیازمندیهای امنیتی ذینفعان تولید نرم افزار و همچنین عدم مدیریت مناسب تغییرات و خط مشی‌های امنیتی می‌شود.

شناخت مسأله :

- استفاده روزافزون شرکتها از **ذخیره، پردازش و انتقال اطلاعات حساس** با استفاده از سیستم های نرم افزاری که بطور مستقیم به اینترنت متصل می شوند.
- استفاده شهروندان از سرویس های اداری و مالی بانک ها و ادارات که به صورت اینترنتی ارائه می شوند؛

نظیر پرداخت های مالی، بیمه ها، سرمایه گذاری‌ها، ثبت نام کودکان برای مدرسه و پیوستن به سازمان های مختلف و شبکه های اجتماعی.

- اتصال جهانی به منابع مختلف اطلاعاتی سیستم های نرم افزاری که آن را آسیب پذیرتر کرده است.
- دوران جنگ اطلاعات [دنینگ، 1998]؛

تروریسم سایبر، جرم و جنایت کامپیوتری با سرعت بالا در راه است.

- هدف قرار دادن کل حیطه سیستم های نرم افزاری از طریق نبوغ انسانی؛

تروریست، جرایم سازمان یافته و دیگر جنایتکاران از طریق نبوغ انسانی با هدف قرار دادن کل حیطه سیستم های نرم افزاری، راه خلاف قانون در پیش گرفته اند و در نفوذ به این سیستم ها معمولا موفق هستند.

- عدم انعطاف پذیری یا مقاومت کافی این سیستم ها در برابر این حملات؛

اکثر این سیستم ها به اندازه کافی در برابر این حملات انعطاف‌پذیر یا مقاوم نیستند.

بطور خلاصه نرم افزارهای فشرده و گسترده، یک محیط باز را برای افشای اطلاعات حساس هویتی و دسترسی‌های غیرمجاز به اطلاعات فراهم نموده است.

گزارشی تحت عنوان امنیت سایبری :

در گزارش ارائه شده به رئیس جمهور ایالات متحده با عنوان امنیت سایبری، اولویت‌بندی [PITAC 2005] اطلاعات کمیته مشورتی فناوری رئیس جمهور، در خصوص خلاصه بحران‌ها و مشکلات نرم‌افزار ناامن (Non secure) به شرح زیر اعلام شده است:

- توسعه نرم افزار هنوز یک علم نیست؛

توسعه نرم‌افزار هنوز یک علم یا یک نظم و انضباط دقیق نیست، همچنین روند توسعه و طراحی برای به حداقل رساندن آسیب‌پذیری در مقابل مهاجمان به اندازه کافی قوی و کنترل شده نیست.

- می‌توان نرم‌افزار را به یک مریض سرطانی تشبیه کرد که بدن او آماده آسیب‌پذیری در مقابل انواع بیماری‌هاست.
- نرم‌افزار آلوده می‌تواند مانند سرطان، خودش را در سرتاسر شبکه تکثیر کند و باعث آسیب در سیستم‌های دیگر شود.
- این فرایندهای مخرب ممکن است از دید افراد عادی و حتی کارشناسان متخصص نامرئی بوده و رشد خود را در خفا ادامه دهند.
- نقص نرم افزار با امنیت انشعابات؛

که شامل اشکالات برنامه نویسی (مانند سرریز بافر) و معایب طراحی (مانند خطاهای متناقض) می‌شود. (شکل 1-2)

- می‌توان سیستم‌ها را از این طریق مقایسه کرد.
- برنامه‌های کاربردی تحت اینترنت بیشتر هدف اینگونه حملات قرار می‌گیرند و باید به امنیت آنها بیشتر پرداخته شود.
- استفاده کدهای مخرب و Botnet [1] ها از این نقص‌ها به منظور دسترسی‌های غیرمجاز و راه‌اندازی حملات؛

بدافزارها، کد های مخرب و بات نت [1] از این نقص‌ها برای تحصیل دسترسی غیرمجاز و راه‌اندازی حملات استفاده می‌کنند.

بات‌نت مجموعه‌ای از کامپیوترهای متصل به اینترنت هستند که استحکامات امنیتی را نقض کرده و کنترل به یک حزب مخرب واگذار شده‌است. هر دستگاه توافق شده‌ای که به عنوان بات شناخته می‌شود، هنگامی که توسط توزیع تروجان، یک نرم‌افزار مورد نفوذ قرار بگیرد، ایجاد می‌شود و به عنوان یک نرم‌افزار مخرب شناخته می‌شود. کنترل‌کننده یک بات نت قادر است که فعالیت های این کامپیوترهای توافق شده را از طریق کانال های ارتباطی که توسط پروتکل‌های شبکه با مبنای استاندارد نظیر IRC و HTTP تشکیل شده‌اند نظارت کند.

بات نت در لغت :

لغتی است که از ایده شبکه‌ای از روباتها استخراج شده است. در ساده ترین شکل، یک روبات یک برنامه کامپیوتری خودکار است. در botnet ها، bot به کامپیوترهایی اشاره می‌کند که می‌توانند توسط یک یا چند منبع خارجی کنترل شوند. یک فرد مهاجم معمولا کنترل کامپیوتر را با ضربه زدن به آن کامپیوتر توسط یک ویروس یا یک کد مخرب بدست می‌گیرد و به این وسیله دسترسی فرد مهاجم به سیستم آسیب دیده فراهم می‌شود.

ممکن است کامپیوتر شما بخشی از یک botnet باشد ولی ظاهرا درست و عادی کار کند. Botnet ها معمولا برای هدایت فعالیتهای مختلفی مورد استفاده قرار می‌گیرند. این فعالیتها می‌تواند شامل انتشار هرزنامه و ویروس، یا انجام حملات انکار سرویس و یا فعالیتهای خرابکارانه دیگر باشد. Botnet ها از کامپیوترهایی تشکیل شده اند که توسط یک سرور خرابکار کنترل می‌شوند و عمده ترین تکنولوژی مورد استفاده جهت انتشار هرزنامه، بدافزار و برنامه های سرقت هویت هستند. زمانی که کامپیوترها آگاهانه یا ناآگاهانه توسط نرم افزاری که برای این منظور طراحی شده آسیب می‌بینند، دیگر قادر نخواهند بود در برابر دستورات مالک botnet مقاومت نمایند.

یک Botnet دقیقا چیست؟

Botnet ها شبکه هایی از کامپیوترهای آلوده هستند. این کامپیوترها تحت کنترل یک مجموعه دستورات هستند که از طریق نرم افزاری که تعمدا و یا نا آگاهانه نصب شده است، مدیریت شده و تغییر می‌کنند. این نرم افزار توسط یک کامپیوتر خرابکار کنترل می‌گردد. ممکن است botnet ها دارای کارکردهای قانونی نیز باشند، ولی در اغلب موارد با فعالیتهای مجرمانه برای انتشار هرزنامه، بدافزار یا حملات سرقت هویت در ارتباطند.

بر اساس مطالعات اخیر، حدود 10 درصد از تمامی کامپیوترهای موجود بر روی اینترنت توسط botnet ها آلوده شده اند. زمانی که یک کامپیوتر توسط نرم افزار botnet آلوده می‌شود، دیگر قادر نخواهد بود در برابر دستورات مالک botnet مقاومت کرده یا از اجرای آنها سر باز زند. برخی اوقات از کامپیوترهای موجود در Botnet ها بعنوان Zombie نام برده می‌شود. اندازه یک botnet به پیچیدگی و تعداد کامپیوترهای استخدام شده در این Botnet بستگی دارد. یک botnet بزرگ ممکن است از 10000 کامپیوتر منفرد تشکیل شده باشد. معمولا کاربران کامپیوترها از این موضوع که سیستمهایشان از راه دور کنترل شده و مورد سوء استفاده قرار می‌گیرد اطلاعی ندارند.

از آنجاییکه Botnet ها از تکنولوژیهای بدافزاری مختلفی تشکیل شده اند، توضیح دادن درباره آنها و پیچیدگی کار آنها چندان ساده نیست. افراد مهاجم تکنولوژیهای مختلف را به نحوی با هم ترکیب کرده اند که دسته بندی آنها را سخت می‌کند.

انواع حملاتی که Botnet ها می‌توانند صورت دهند:

- **حملات انکار سرویس توزیع شده**

یک botnet با هزاران عضوی که در سراسر جهان دارد می‌تواند یک حمله گسترده و هماهنگ را برای خراب کردن یا از کار انداختن سایتها و سرویسهای مهم راه اندازی نماید و منابع و پهنای باند این سیستمها را اشغال کند. حملات چندین گیگا بیت بر ثانیه توسط botnet ها حملاتی کاملا شناخته شده و معمول هستند. اغلب حملات معمول از UDP، ICMP، و TCP SYN استفاده می‌کنند. اهداف این حملات ممکن است شامل وب سایتهای تجاری یا دولتی، سرویسهای ایمیل، سرورهای DNS، ارائه دهنده های سرویس اینترنت، زیرساختهای اساسی اینترنت یا حتی تولید کنندگان ابزارهای امنیتی صنعت فناوری اطلاعات باشد. حملات همچنین ممکن است سازمانهای سیاسی یا مذهبی خاصی را هدف بگیرند. این حملات گاهی با باج گیری همراه می‌شوند. هر سرویس اینترنتی ممکن است هدف یک botnet قرار گیرد. این کار می‌تواند از طریق غرق کردن وب سایت مورد نظر در درخواستهای بازگشتی HTTP انجام شود. این نوع حمله که در آن، پروتکلهای سطوح بالاتر نیز برای افزایش تاثیر حمله به کار گرفته می‌شوند، بعنوان حملات عنکبوتی نیز مشهور است.

- **ابزار جاسوسی و بدافزار**

Botnet ها فعالیتهای تحت وب کاربران را بدون اطلاع یا رضایت کاربر کنترل کرده و گزارش می‌دهند. همچنین ممکن است Botnet ها نرم افزار دیگری را برای جمع آوری اطلاعاتی درباره آسیب پذیریهای سیستم نصب کرده و این اطلاعات را به دیگران بفروشند. علاوه بر این، یک ربات همچنین می‌تواند بعنوان یک وسیله استراق سمع به کار رفته و داده های مهم و حساس را که از یک سیستم آسیب دیده می‌گذرند گوش دهد. داده های نوعی که این رباتها به دنبال آن می‌گردند عبارتند از اسامی کاربری و کلمات عبوری که فرمانده Botnet می‌تواند برای اهداف شخصی خود از آنها استفاده کند. داده هایی درباره یک botnet رقیب که در همان واحد نصب شده است نیز می‌تواند هدف فرمانده botnet قرار بگیرد تا به این وسیله، botnet دیگر را نیز سرقت نماید.

- **سرقت هویت**

در اغلب موارد Botnet ها برای سرقت اطلاعات هویت شخصی افراد، داده های مالی و تجاری، یا کلمات عبور کاربران و سپس فروش یا استفاده مستقیم از آنها به کار می‌روند. همچنین Botnet ها در پیدا کردن و معرفی سرورهایی که می‌توانند برای میزبانی وب سایتهای

سرقت هویت مورد استفاده قرار گیرند کمک کنند. این وب سایتها خود را به جای یک وب سایت معتبر جا زده و کلمات عبور و داده های هویتی کاربران را سرقت می‌کنند.

- **ابزار تبلیغاتی**

ممکن است botnet ها بطور خودکار popup های تبلیغاتی را بر اساس عادات کاربر دانلود و نصب نمایند یا مرورگر کاربر را مجبور کنند که بطور متناوب وب سایتهای خاصی را مشاهده نماید.

- **هرزنامه**

یک botnet می‌تواند برای ارسال هرزنامه ها مورد استفاده قرار گیرد. پس از اینکه یک کامپیوتر مورد سوء استفاده قرار گرفت، فرمانده botnet می‌تواند از این zombie جدید به همراه سایر zombie های botnet استفاده کرده و با جمع آوری آدرسهای ایمیل به ارسال دسته ای هرزنامه و یا ایمیلهای سرقت هویت اقدام نماید. امروزه اغلب هرزنامه ها از طریق botnet ها انتشار می‌یابند. بر اساس مطالعات اخیر، در سال 2008 botnet ها مسوول انتشار بیش از 90 درصد از هرزنامه ها بودند.

- **گسترش** Botnet

Botnet ها همچنین می‌توانند برای گسترش سایر Botnet ها مورد استفاده قرار گیرند. این کار با متقاعد کردن کاربر برای دانلود کردن فایل اجرایی مورد نظر از طریق FTP، HTTP یا ایمیل انجام می‌شود.

- **فریب سیستمهای پرداخت به ازای هر کلیک**

Botnet ها می‌توانند برای مقاصد تجاری مورد استفاده قرار گیرند. آنها این کار را با کلیکهای خودکار روی یک سیستم که به ازای هر کلیک مبلغی را پرداخت می‌کند انجام می‌دهند. اعضای Botnet در هنگام آغاز به کار یک مرورگر بطور خودکار روی یک سایت کلیک می کنند. بعبارت دیگر، این Botnet ها تعداد کلیکهای یک آگهی تبلیغاتی را به شکل مصنوعی افزایش می‌دهند.

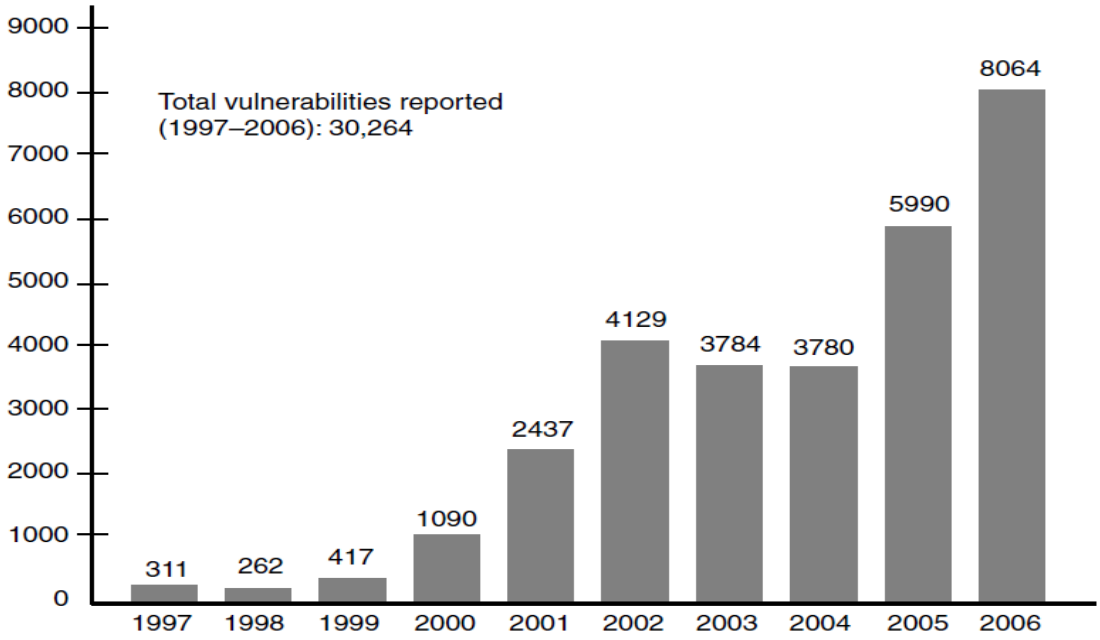


Figure 1–2: *Vulnerabilities reported to CERT*

با توجه به این روند، در اینجا یک نیاز واضح و مبرم به تغییر راه و رویکرد به منظور توسعه یک رویکرد منظم به امنیت نرم‌افزار برای ما (مدیران پروژه و مهندسین نرم افزارکامپیوتر) است. [مک 2006].

در سال 2007 بررسی امنیت جهانی نشان داد که، 87 درصد از پاسخ دهندگان به نظرسنجی، کیفیت ضعیف توسعه نرم‌افزار را به عنوان یک تهدید در12 ماه آینده دانستند.

تعریف امنیت نرم افزار

امنیت نرم افزار به معنای تضمین این است که در مرحله توسعه نرم افزار، کد امن یکپارچه‌ای جهت جلوگیری از آسیب‌پذیری به کار رفته باشد تا نرم‌افزار به خوبی بتواند مراحل مختلف مانند تست آسیب پذیری، اسکن نرم افزار و تست نفوذبخشی در چرخه عمر توسعه SDLC (System Development life Cycle) را بگذراند. [2007 Deloitte].

مناطق خطر مؤثر بر امنیت نرم افزار

- اتصال به اینترنت؛

فرایند رشد اتصال به اینترنت در کامپیوترها و شبکه ها و وابستگی کاربرهای مشابه در شبکه- سرویسهای فعال شده اعم از ایمیل، معاملات مبتنی بر وب، افزایش یافته است. این مساله روند امنیت نرم افزار را در معرض خطر بیشتری قرار می‌دهد.

- بروز رسانی ها؛

یکی دیگر از مناطق خطر که در امنیت نرم افزار موثر می‌باشد به روز رسانی است. این خطر زمانی بروز می‌کند که Update های سیستم پذیرفته شده است و امکانات آن توسعه یافته است. سیستمهای توسعه پذیر بسیار فریبنده هستند زیرا Feature های آنها قابلیت افزایش داشته و سرویسهای جدیدی را فراهم می‌کنند. در بروزرسانی هر پسوند جدید، امکانات جدید و رابط های گرافیکی جدید ایجاد می‌شود و بدین ترتیب خطرات جدید اضافه می‌شود و در نهایت منطقه خطر امنیت نرم افزار، به طور کنترل نشده، در یک مقیاس رشد کرده که این مساله منجر به پیچیدگی سیستم نرم‌افزاری می‌شود مانند سیستم عامل ویندوز.

پیچیدگی سیستم ها

نرم افزارها چه شرایطی دارند؟

- تولید و ساخت یک سیستم نرم افزاری به صورت بخشهای جداگانه و مجزا؛

تولید یک سیستم نرم افزاری قابل اعتماد، در مرحله ساخت و راه اندازی که به صورت مجزا و بخشهای جداگانه صورت می‌گیرد زمان زیادی نخواهد برد. تولید هریک از این قسمتهای مجزا باید منطبق بر هزینه و زمان بندی از پیش برنامه‌ریزی شده، صورت بگیرد.

- تشکیل یک کل عملیاتی؛

هر جزء جدید و یا به روز رسانی محیط عملیاتی موجود در نرم افزار، باید با حفظ ویژگیهایی که به ارث برده می‌شوند انجام شود و در نهایت این اجزاء جهت تشکیل یک کل عملیاتی، ادغام شوند.

- طراحی سیستم های جدید بر روی سیستمهای قدیمی و مبتنی بر وب کردن آنها؛

طراحی سیستم های جدید بر روی سیستم های قدیمی و وب بیس کردن آنها منجر به تولید سیستم‌هایی می‌شود که مملو از آسیب پذیری هستند.

گسترش دامنه و مقیاس سیستم، نیاز به تجدید نظر در تعدادی از مفروضات توسعه را بهمراه دارد که به طور کلی به امنیت نرم افزار مربوطند.

تجدیدنظر در خصوص برخی مفروضات توسعه

- کنترل متمرکز؛

به جای کنترل متمرکز، که در سیستمهای بزرگ تبدیل به یک هنجار می‌شود، مدیران پروژه باید به این مساله توجه داشته باشند که نقاط کنترل متعدد و اغلب مستقلی را برای سیستم‌ها درنظر بگیرند.

- افزایش یکپارچگی؛

افزایش یکپارچگی در سیستم‌ها به سرعت منجر به کاهش توانایی این سیستمها برای ایجاد تغییرات در مقیاس گسترده می‌شود. علاوه بر این، برای سیستم هایی که مدیریت آنها به طور مستقل صورت می‌گیرد، فرایند ارتقاء لزوما هماهنگ نیست. بنابراین مدیران پروژه نیاز به حفظ قابلیت های عملیاتی با امنیت مناسب در به روز رسانی خدمات و افزودن خدمات جدید دارند.

- ادغام سیستمهای مستقل و در حال توسعه؛

با ادغام سیستم‌های مستقل و در حال توسعه، مدیران پروژه ناگزیر از رویارویی با مجموعه‌ای ناهمگون از اجزاء، پیاده سازی‌های مختلف از رابط های معمول، ناسازگاری و تناقض در میان سیاست های امنیتی خود هستند.

- عدم تطابق و خطاهای سیستم های مستقل در حال توسعه؛

با عدم تطابق و خطاهای سیستم‌های مستقل در حال توسعه، نارسایی هایی در مدیریت بعضی از فرم‌ها ایجاد می‌شود، در این حالت باید به تطبیق و شکل دهی مجدد و پیچیده نیازمندی های امنیتی پرداخت.

تضمین کیفیت و امنیت نرم‌افزار

- قابلیت اطمینان یک نرم افزار بخش مهمی از کیفیت آن است. اگر یک نرم افزار به دفعات متعدد در اجرا با شکست مواجه شود، آنگاه یکی از مهمترین عوامل کیفی آن از بین رفته است.

قابلیت اطمینان نرم افزار به این شکل تعریف می شود: عملکرد بدون شکست یک نرم افزار در محیطی خاص و برای مدت زمانی معین.

توجه به این نکته حائز اهمیت است که شکست نرم افزار در اثر اشکالات طراحی است. در واقع تمام شکست‌های نرم افزاری به مشکلات طراحی یا پیاده سازی منتهی می‌شوند.

- امنیت نرم افزار یکی از شاخص‌های تضمین کیفیت نرم‌افزار است که متضمن تشخیص خطرات احتمالی که بر روی نرم‌افزار تاثیرات منفی دارند و موجبات شکست سیستم را بوجود می‌آورند، است.

اگر این خطرات احتمالی زود تشخیص داده شوند، اشکالات و نواقص طراحی نرم‌افزار قابل تشخیص بوده و حذف آنها از فرآیند مهندسی نرم-افزار امکان پذیر می‌گردد.

قابلیت اطمینان در نرم‌افزار از دیدگاه NIA

به گفته کمیته سیستم های امنیت ملی (National Information Assurance) "تضمین اطلاعات ملی آمریکا (IA) واژه نامه" [CNSS 2006]

اطمینان در نرم افزار (Software Assurance) به معنای عاری بودن محصول نرم افزاری از آسیب پذیری‌های تعمدی و یا اتفاقی است که در هر مرحله از چرخه حیات یک محصول نرم‌افزاری به آن تزریق شده باشد.

علل مؤثر بر کاهش امنیت نرم‌افزار

امروزه قابلیت و طرز کار نرم‌افزار به تنهایی نمی‌تواند عاملی برای عرضه آن و تبلیغ برای آن قلمداد شود، بلکه امن بودن و قابلیت اطمینان از آن اهمیت بالایی دارد.

- ارتباط ضعیف نرم افزار در اجرای موفقیت آمیز سیستم های نرم افزاری؛
- حجم نرم افزار و پیچیدگی آن مانع از اجرای آزمون جامع می‌شود.
- برون سپاری و استفاده از اجزاء زنجیره عرضه نرم افزار تایید نشده، منجر به افزایش در معرض خطر قرار گرفتن می‌شود.
- فریبندگی و پیچیدگی روز افزون؛

منجر به حملات مخفیانه برای بهره بردای از تسهیلات می‌شود.

- استفاده مجدد از برنامه های نرم افزار قبلی با برنامه های کاربردی دیگر منجر به عواقب ناخواسته، افزایش تعداد اهداف آسیب‌پذیر است.

رهبران کسب و کار تمایلی به قبول ریسک سرمایه گذاری در در نرم افزار های امنیتی ندارند.

مفاهیم تضمین نرم‌افزار

- قابلیت اطمینان نرم افزار

همچنین به عنوان نرم افزار تحمل خطا شناخته شده است.

- ایمنی نرم افزار
- امنیت نرم افزار

که توانایی های نرم افزار برای مقاومت در برابر تهدید های امنیتی، تحمل و بازیابی در برخورد با حوادث است.

هدف از امنیت نرم‌افزار

هدف اصلی از امنیت نرم افزار تولید نرم افزارهای قویتر، با کیفیت بالاتر و بدون نقص است که همواره بتواند به درستی در برابر حملات مخرب مقاومت کند. [مک 2006].

بعبارت دیگر

هدف از امنیت نرم افزار فراهم آوردن خواسته های زیر در سیستم های نرم افزاری است:

- مصونیت از آسیب پذیری و حملات احتمالی به سیستم؛
- کاهش اثرات خرابی و بازگردانی با بیشترین سرعت ممکن

کاهش اثرات خرابی ناشی از شکست های احتمالی در اثر حملات، اطمینان حاصل نمودن از اینکه اثرات هرگونه حمله منتشر نشود، و بازگردانی با بیشترین سرعت ممکن از آن شکست.

- مقاومت در برابر شکست ناشی از حملات؛

این سیستم همچنان بتواند بعد از انواع حملات تا حد ممکن به کار خود ادامه دهد و در برابر شکست ناشی از این حملات مقاومت نماید.

ویژگیهای نرم افزار امن توسعه‌یافته

نرم افزاری که بصورت امن توسعه یافته باشد بطور کلی در سراسر چرخه عمر توسعه، دارای خواص زیر است:

- اجرای قابل پیش بینی (Predictable execution)

یعنی نرم افزار در هنگام اجرا کاملا با انتظاراتی که از آن می‌رود، مطابقت داشته باشد.

- امانت (Trustworthiness)

به حداقل رساندن آسیب‌پذیری‌های نرم‌افزار تا حد ممکن و جلوگیری از تغییر اهداف کلی در اثر برنامه‌های مخرب.

- مطابقت (Conformance)

با فعالیت‌های برنامه‌ریزی شده، سیستماتیک و چند رشته‌ای، اطمینان حاصل شود که اجزای نرم افزارها، محصولات و سیستم مطابق با الزامات و استانداردهای قابل اجرا و همچنین مناسب برای استفاده مشخص است.

این اهداف و خواص باید تفسیر شوند:

- کلیه فرایندهای مرتبط با شناسایی، تحلیل و پاسخگویی به هرگونه عدم اطمینان؛

که شامل حداکثرسازی نتایج رخدادهای مطلوب و به حداقل رساندن نتایج وقایع نامطلوب می‌باشد (مدیریت ریسک).

- مقاوم در برابر حمله، تحمل حمله و تا حد امکان انعطاف‌پذیر در برابر حمله؛

علاوه بر اجرای قابل پیش بینی، اعتماد و انطباق، نرم افزار امن باید به عنوان سیستمی مقاوم در برابر حملات باشد، تحمل حمله و انعطاف-پذیری در برابر آن تاحدی که ممکن است. برای اطمینان از این موضوع مهندسین نرم افزار باید به طراحی مولفه‌هایی بپردازند که ورودی‌های شناخته شده و قابل قبول در سیستم را از حملات مخرب تشخیص دهد. این ورودی‌ها ممکن است افراد و یا فرایندها باشند.

- بازیابی از خرابی؛

برای رسیدن به انعطاف‌پذیری در برابر حملات، سیستم نرم‌افزار باید قادر به بازیابی از خرابی حاصله از حملات موفقیت‌آمیز باشد و با از سرگیری عملیات خود بتواند حداقل سطح قابل قبولی از خدمات را به موقع ارائه دهد. این سیستم در نهایت باید در بهبود خدمات کامل در سطح مشخصی از عملکرد برسد.

نقش فرایندها و روشها در نرم‌افزار امن

تاکنون رویکردهایی که تزریق امنیت در نرم‌افزار را به مرحله پس از تولید وا می‌گذارند، چندان در این امر کارآمد نبوده‌اند. از این رو، مکتب نوینی در این وادی به وجود آمده است که اعمال تمهیدات امنیتی در نرم‌افزار را در سراسر چرخه تولید نرم‌افزار مناسب و مطلوب می‌داند.

تعدادی از عواملی که امن بودن نرم افزار را تحت تاثیر قرار می دهد عبارتند از:

- آسیب‌پذیری‌های نرم افزار که می‌تواند از اتخاذ فرآیندها و شیوه‌های مورد استفاده در توسعه آن، از ابتدا تاثیر بگیرد.

این عوامل عبارتند از: تصمیم‌گیری‌های صورت گرفته توسط مهندسان نرم افزار، نقص معرفی آنها در طراحی و مشخصات، اشتباهات و سایر نقص‌هایی که به صورت سهوی یا عمدی در کد توسعه یافته نرم‌افزار ایجاد شده‌اند.

- عواملی مانند انتخاب زبان‌های برنامه نویسی، ابزار توسعهٔ مورد استفاده جهت توسعه نرم افزار و پیکربندی آن و همچنین رفتار اجزای نرم‌افزاری در توسعه خود و محیط‌های عملیاتی.
- ساختار فرآیندها و شیوه‌های مورد استفاده برای تعیین، طراحی و توسعه نرم افزار؛

با این حال به طور فزاینده‌ای مشاهده می‌شود که مهم‌ترین تفاوت بین نرم‌افزارهای امن و ناامن در ساختار فرآیندها و شیوه‌های مورد استفاده برای تعیین، طراحی و توسعه نرم افزار نهفته شده است.

تهدیدهای امنیت نرم‌افزار

در امنیت اطلاعات، تهدید منبع خطر است که این تهدید اغلب یک شخص است که قصد تخریب، با استفاده از یک یا چند نرم‌افزار مخرب را دارد.

نرم افزار در معرض دو دسته کلی از تهدیدات است:

- تهدید در طول توسعه (به طور عمده تهدیدات داخلی)

یک مهندس نرم‌افزار می‌تواند نرم‌افزار را در هر نقطه‌ای که از چرخه توسعه زندگی خود باشد از طریق محرومیت عمدی از شمول و یا تغییرات در ویژگیهای موردنیازش، مدل‌های تهدید، اسناد طراحی، کد منبع، مونتاژ و چارچوب ادغام، آزمون خرابکاری موارد و نتایج آزمون و یا نصب و راه‌اندازی و پیکربندی دستورالعمل ها و ابزار، تحت تاثیر قرار دهد.

- تهدید در طول اجرا (اعم از تهدیدات داخلی و خارجی)

هر سیستم نرم‌افزاری قابل اجرا بر روی پلت فرم شبکه است که از آنجایی که بیشتر در معرض حمله است، احتمال آسیب‌پذیری بالاتری دارد.

این حملات ممکن است با استفاده از آسیب‌پذیری‌های عمومی شناخته شده باشد مانند سرریز بافر، اجرای کد از راه دور و....

یک مثال

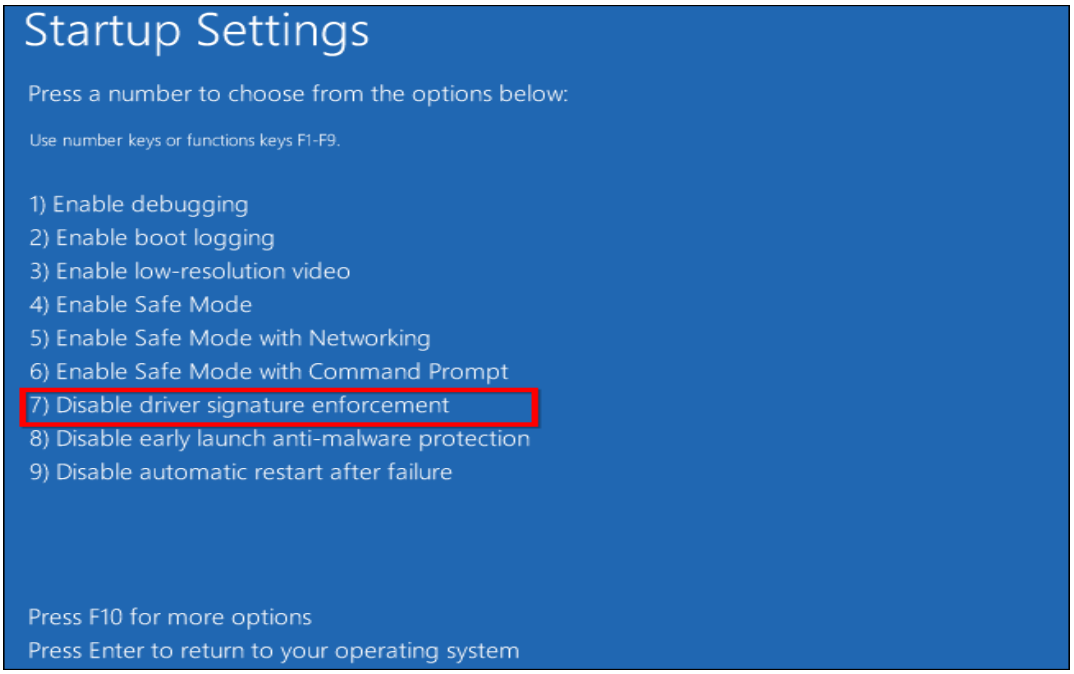
چرا نسخه 64 بیتی ویندوز به نسبت نسخه 32 بیتی آن امنیت بالاتری دارد؟

منظور از امنیت بالاتر این نیست که ویندوز ۶۴ بیتی نسبت به مخرب‌ها ایمن است بلکه مزیت آن داشتن امکانات امنیتی بیشتر است.

- تصادفی سازی چیدمان فضای آدرس؛

ASLR قابلیتی است که باعث می‌شود اطلاعات مربوط به یک نرم‌افزار در حافظه به صورت تصادفی چیده شوند. در واقع اگر تصادفی‌سازی چیدمان فضای آدرس‌ها یا ASLR وجود نداشته باشد، ممکن است مکان قرارگیری اطلاعات نرم‌افزار در حافظه قابل پیش‌بینی باشد و در نتیجه حملات امنیتی برای نرم‌افزارها خطرناک‌تر خواهند بود.
در مقابل اگر ASLR فعال باشد، کسی که قصد حمله و هک کردن سیستم را دارد می‌بایست موقعیت اطلاعات مربوط به نرم‌افزار موردنظر را به درستی حدس زده و از حفره‌های امینتی موجود استفاده کند. در واقع اگر حدس وی اشتباه از آب درآید، ممکن است نرم‌افزار بسته شده و حمله نافرجام تمام شود.

این ویژگی امنیتی در نسخه‌های ۳۲ بیتی ویندوز و سایر سیستم‌عامل‌ها نیز وجود دارد ولی در نسخه‌های ۶۴ بیتی بسیار قدرتمندتر است. یکی از دلایل آن داشتن فضای آدرس بیشتر در نسخه‌های ۶۴ بیتی است چرا که حافظهٔ رم بیشتری را پشتیبانی و آدرس‌دهی می‌کنند.



(شکل 1-3)

- بررسی الزامی امضای دیجیتالی درایورها؛

نسخهٔ ۶۴ بیتی ویندوز این الزام را ایجاد می‌کند که درایورهای سخت‌افزار از نظر امضای دیجیتالی چک شوند. این الزام در مورد درایورهای سخت‌افزاری برای Kernel (یا هستهٔ ویندوز) و همچنین درایورهای کاربری مثل درایور پرینتر وجود دارد.
بررسی الزامی امضای دیجیتالی درایورها باعث می‌شود که اگر فرضاً درایوری فاقد امضا است و در واقع یک مخرب به شکل درایور است، روی سیستم عامل اجرا نشود.
بررسی الزامی امضای دیجیتالی درایورها باعث می‌شود که اگر فرضاً درایوری فاقد امضا است و در واقع یک مخرب به شکل درایور است، روی سیستم عامل اجرا نشود. بنابراین سازندگان محصولات نرم‌افزاری مخرب، برای اجرا کردن درایور معیوب خود مجبور هستند هنگام بوت شدن سیستم با اجرای یک Rootkit، پردازش‌های مربوط به درایور خود را از دید سیستم عامل پنهان کنند. راه‌حل دیگر استفاده از یک امضای دیجیتالی معتبر است که از سازندگان قانونی درایورها به سرقت رفته است.

برای غیرفعال کردن بررسی امضای دیجیتالی درایورها به منظور توسعهٔ نرم‌افزارها در ویندوزهای ۶۴ بیتی می‌بایست به Debugger کرنل متصل شد و یا هنگام Reboot شدن ویندوز از گزینهٔ خاصی استفاده کرد که در حالت معمول وجود ندارد.

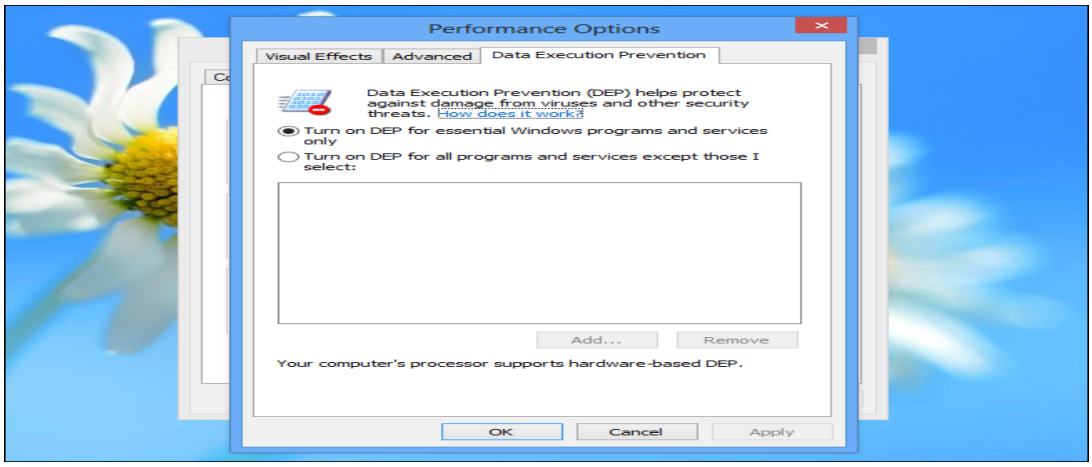
- محافظت Kernel در مقابل Patch یا وصله؛

KPP یا Kernel Patch Protection که با نام Guard Patch هم شناخته می‌شود، یک ویژگی امنیتی است که تنها در نسخه‌های ۶۴ بیتی ویندوز دیده می‌شود. پچ‌گارد به نرم‌افزارها و حتی درایورها اجازه نمی‌دهد که در حالت کرنلی اجرا شوند و کرنل ویندوز را با کدهایی وصله‌دار یا پچ کنند. از نظر فنی می‌توان چنین قابلیتی را در نسخه‌های ۳۲ بیتی ویندوز هم فعال کرد ولی هیچ وقت در نسخه‌های ۳۲ بیتی از چنین ویژگی‌ای پشتیبانی نشده است. برخی از نسخه‌های ۳۲ بیتی آنتی‌ویروس‌ها، با پچ کردن کرنل ویندوز به ارزیابی و تشخیص ویروس‌ها و محافظت سیستم در برابر آنها می‌پردازند.
پچ‌گارد به درایور وسایل سخت‌افزاری اجازه نمی‌دهد که کرنل را پچ کنند. به عنوان مثال مانع دستکاری کرنل ویندوز توسط روت‌کیت‌ها و در ادامه پنهان شدن در سیستم‌عامل می‌شود.

اگر هرگونه تلاش جهت دستکاری و پچ کردن کرنل ویندوز تشخیص داده شود، ویندوز سریعاً خاموش شده و صفحهٔ آبی مرگ نمایش داده می‌شود و یا ممکن است سیستم Reboot شود.

- ممانعت از اجرای اطلاعات؛

DEP یا Execution Prevention Data این اجازه را به سیستم عامل می‌دهد که برخی از نقاط حافظه را به عنوان غیرقابل اجرا مشخص و علامت‌گذاری کند. علامت‌گذاری این خانه‌های حافظه توسط NX Bit انجام می‌شود. در واقع نواحی خاصی از حافظه که تنها قرار است داده‌ای را نگه دارند، قابل اجرا نخواهند بود.
به عنوان مثال یک سیستم عامل بدون DEP را در نظر بگیرید، یک هکر با استفاده از نوعی Buffer Overflow (سرریزش حافظه) می‌تواند کدهایی را به ناحیه‌ای از حافظه که به یک نرم‌افزار خاص اختصاص داده شده، اضافه نماید. کد اضافه شده بعداً اجرا شده و هکر به مقصود خود می‌رسد. با استفاده از DEP هکر ممکن است کدی را اضافه کند ولی آن بخش از حافظه با علامت غیر قابل اجرا مشخص شده و اجرا نمی‌شود و در نتیجه حمله متوقف خواهد شد. تنظیمات پیش‌فرض برای نرم‌افزارهای ۶۴ بیتی سخت‌گیرانه‌تر و DEP همواره فعال است در حالی که در سیستم‌عامل ۳۲ بیتی به خاطر مشکلات سازگاری نرم‌افزارها، این ویژگی در حالت پیش‌فرض نافعال می‌باشد.



(شکل 1-4)

چرخه حیات نرم‌افزار :

فصل دوم :

خواص کلیدی

خواص کلیدی به طور معمول برای توصیف امنیت شبکه استفاده می شود. اما در اینجا تعاریف برای انطباق با حوزه امنیت نرم افزار تغییر داده شده است. این خواص عبارتند از: محرمانگی – جامعیت – در دسترس بودن – پاسخگویی – غیر قابل انکار بودن

- **محرمانگی**

در نرم افزار باید اطمینان حاصل شود که هر یک از ویژگی های آن مانند: محیط اجرایی، مدیریت دارایی و یا محتوای آن از اشخاص غیرمجاز پنهان شود. این اصل شامل مواردی مانند نرم افزار منبع باز هم می شود. با وجود اینکه این برنامه ها ویژگی ها و محتوای خود را دردسترس عموم مردم قرار می دهند اما هنوز هم باید محرمانگی را درخصوص دارایی های خود حفظ نمایند.

مثال نقض:

جمله ی موفق تزریق SQL در نرم افزار برای استخراج اطلاعات شخصی از پایگاه داده می تواند عدم وجود محرمانگی در نرم افزار را اثبات کند.

- **جامعیت**

نرم افزار . کاربردهای مدیریت شده آن باید در برابر براندازی، مقاوم و انعطاف پذیر باشد. براندازی ممکن است از طریق تغییرات غیرمجاز در کد نرم افزار، کاربردهای مدیریت شده در پیکربندی یا هرگونه تغییرات توسط رفتار اشخاص مجاز و یا غیرمجاز ایجاد شود. این تغییرات ممکن است شامل جایگذاری، فساد، دستکاری، تخریب، درج ناخواسته یا حذف باشد. جامعیت باید در طول مرحله ی توسعه نرم افزار و اجرای آن حفظ شده باشد.

- **دردسترس بودن**

این نرم افزار باید در هر زمان که برای کاربران مجاز خود مورد نیاز است عملیاتی و دردسترس باشد در عین حال عملکرد و مزایای آن باید همواره دور از دسترس کاربران غیرمجاز باشد.

مثال نقض:

کراس سایت اسکریپتینگ XSS یا حمله موفقیت آمیز علیه یک برنامه کاربردی وب می تواند در نقض هر دو مورد جامعیت و دردسترس بودن تاثیر گذار باشد.

- **پاسخگویی**

همه ی اقدامات امنیتی مربوط به نرم افزار به عنوان کاربر با مسئولیت های خاص باید ثبت و ردیابی شود. این ردیابی بلافاصله بعد از ثبت اقدامات انجام شده امکان پذیر باشد. زبان حسابرسی مرتبط با آن در سیاست های امنیتی نرم افزار نشان دهنده ی این است که اقدامات امنیتی مناسب مربوطه در نظر گرفته شده است.

- **غیر قابل انکار بودن**

این ویژگی به توانایی نرم افزار برای جلوگیری از رد یا انکار مسئولیت، برای اقدامات انجام شده توسط کاربر اشاره می کند. همچنین تضمین می کند که ویژگی های پاسخگویی را نمی توان از بین برد.

مثال نقض:

یک سرریز بافر موفق حمله ای است که با تزریق کد مخرب تلاش می کند اطلاعات مربوط به حساب کاربری را سرقت کرده و سپس با تغییر سیاهه های مربوط به پوشش دادن حمله خود تمام ویژگی های کلیدی امنیت را نقض کند.

خواص موثر در نرم افزار

برخی از خواص نرم افزار، اگرچه به طور مستقیم در نرم افزار امن تاثیر ندارند اما ممکن است برای توصیف چگونگی نرم افزار امن بکار می روند مانند:

- **اعتماد**

قابلیت اعتماد یک ویژگی از نرم افزار است که تضمین می کند این نرم افزار به همان عنوانی که در نظر گرفته شده است عمل خواهد کرد.

هیچ آسیب پذیری یا ضعف بهره برداری و ویژگی مخربی در نرم افزار وجود ندارد.

- **پیش بینی**

نرم افزار در هنگام اجرا ، هر آنچه که برای آن در نظر گرفته شده را بدون کم وکاست ارائه کند.

اجرای قابل پیش بینی باید این مفهوم را برساند که این نرم افزار به طور موثر علاوه بر اینکه انتظارات قابل اجرا را انجام دهد ، مواردی را هم که دور از انتظار است انجام ندهد.

- **صحت**

صحت یک ویژگی مهم از نرم افزار است که باید به طور مداوم تحت تمام شرایط عملیاتی در نظر گرفته شود.

یکی از مکانیسم های تعیین صحت نرم افزار حمله به امنیت نرم افزار می باشد.

نرم افزار باید تحت شرایط پیش بینی نشده امنیت خود را حفظ نماید. در مهندسی نرم افزار برای ایجاد صحت باید تعداد آسیب پذیری هایی که در نرم افزار وجود دارد را (که توسط مهاجمان مورد سواستفاده قرار می گیرد) کاهش داد.

- **ایمنی**

در مهندسی نرم افزار ، قابلیت های کارکرد عملیاتی، رابط کاربری و صفات کارایی باید در تایید این باشد که آنها تضمین کننده ایمنی نرم افزار خواهند بود.

- **اطمینان**

توانایی نرم افزار در ارائه سرویس های مشخص شده طبق انتظار کاربر و احتمال انجام عملیات بدون خطا در مدت زمان معین

راهکارهای تاثیرگذاری بر امنیت نرم افزار :

- تغییر دیدگاه از دیدگاه تدافعی به تهاجمی
- نیاز به منابع دانش و پوشش تمام چرخه عمر نرم افزار با استفاده از تکنیک هایی مثل نقاط تماس (Touch points)

استراتژی دفاعی

دیدگاه دفاعی معمولا شامل نگاه به نرم افزار از داخل به خارج است. تهدید های امنیت و آسیب پذیری ها را از این دریچه بررسی می کند.در اینصورت باید به موارد زیر توجه کرد:

- پرداختن به مسائل غیر منتظره از طریق اجتناب ،حذف و کاهش نقاط ضعف
- تلاش برای بهبود مستمر و تقویت مقاومت در برابر حملات
- پرداختن به مسائل امنیتی مورد انتظار از طریق طراحی معماری امنیتی مناسب

استراتژی دفاعی برای مسائل امنیتی مورد انتظار:

احراز هویت، اختیارها،کنترل دسترسی، مجوز، رمزنگاری و بطور کلی طراحی و معماری امنیت مناسب.

استراتژی دفاعی برای مسائل غیر منتظره :

- بکار گیری شیوه های متمرکز در شناسایی و کاهش نقاط ضعف در سیستم های نرم افزار

کشف نقاط آسیب پذیری در نرم افزار از طریق تست جعبه سیاه

تست جعبه سیاه

تست جعبه سیاه اصطلاحاً به شرایطی گفته می‌شود که اجراکنندگان تست، هیچ نوع اطلاعات اولیه‌ای در رابطه با ساختار و شرایط شبکه مورد بررسی و یا سیستم هدف ندارند.

گام های این تست عبارتند از:

- جمع آوری اطلاعات
- تست واقعی امنیت برای طرح شبکه
- ایده یابی در رابطه با افکار طراح و برنامه ریز شبکه
- تست هدف با ارزیابی کامل و هماهنگی ایده طراح و ایده حمله کننده

تست جعبه سیاه و سفید

- در واقع می‌توان گفت تست جعبه سیاه به شبکه از دید نفوذگران خارجی می‌نگرد که هیچ نوع اطلاعات اولیه از شبکه هدف ندارند و با این شرایط ممکن است اقدام به حمله علیه شبکه نمایند. در واقع می‌توان گفت تست جعبه سیاه به شبکه از دید نفوذگران خارجی می‌نگرد که هیچ نوع اطلاعات اولیه از شبکه هدف ندارند و با این شرایط ممکن است اقدام به حمله علیه شبکه نمایند اما تست جعبه سفید شبکه را از دید نفوذگران داخلی بررسی می‌کند که ممکن است به دلایلی قصد آسیب‌رساندن به شبکه را داشته باشند.
- در میان این دو دیدگاه کاملاً متفاوت می‌توان روش‌های بینابین‌تری را نیز در نظر گرفت که به آن‌ها اصطلاحاً تست جعبه خاکستری می‌گویند.

یک باور غلط

برخلاف تصور غلط رایج، امنیت نرم افزار تنها به عهده کدنویس و برنامه نویس نمی باشد بلکه افراد دیگری مثل افراد سازمان حمایت کننده و یا افراد تیم توسعه نیز می باشد. مثلا با مرور کدهای نرم افزار توسط افراد توسعه دهنده با تحربه می توان امنیت را در سیستم بالاتر برد و ....

سه روش مکمل برای بهبود امنیت در سیستم های تولید نرم افزار

1- اجتناب از خطر: سیستم به گونه ای طراحی شود که با خطر مواجه نشود.

2- تشخیص و رفع خطر: سیستم به گونه ای طراحی شود که خطرات قبل از منجر شدن به سانحه تشخیص و خنثی شود.

3- محدود کردن خسارت: سیستم به گونه ای طراحی شود که پیامدهای سانحه به حداقل برسد.

هدف نهایی در تلاش های امنیتی

این است که سیستم در حفاظت از خودش برای حفظ ویژگی های دفاعی در مقابل حملات عمدی توانایی کاملی داشته باشد. این تلاش های امنیتی عبارتند از:

1- مقاومت در برابر حمله

2- تحمل حمله

3- انعطاف پذیری در برابر حمله

در ادامه مطلب مهم دیگری که باید به آن توجه کرد مدیریت ریسک است برای داشتن یک نرم افزار مناسب امن در فرآیند تولید نرم افزار باید به این بخش اساسی توجه کرد.

مدیریت ریسک

شامل پیش بینی ریسک هایی است که ممکن است بر زمانبندی و کیفیت نرم افزار در حال توسعه موثر باشد. ریسک ها سه دسته اند:

- ریسک های پروژه: ریسک هایی که بر زمانبندی و منابع پروژه تاثرگذارند.
- ریسک های محصول: ریسک هایی که به کیفیت و کارایی نرم افزار در حال توسعه تاثیر می گذارند.
- ریسک های تجاری: ریسک هایی که بر اعتبار سازمان توسعه دهند یا تهیه کننده نرم افزار تاثیرگذارند.

فرآیند مدیریت ریسک

شامل 4 فعالیت:

1- شناسایی ریسک: ریسک های پروژه و محصول و کار شناسایی می شود.

2- تحلیل ریسک: احتمال و ترتیب ریسک ها مشخص می شود.

3- برنامه ریزی ریسک: شامل برنامه ریزی برای مقابله با ریسک، اجتناب از اثرات منفی ریسک یا حداقل کردن تاثیر منفی آن است.

4- نظارت بر ریسک: ریسک هایی که تخمین زده شده اند با جمع آوری اطلاعات بیشتر در مورد آنها برای تعدیل ریسک برنامه ریزی می شود.

چشم انداز مهاجمین

با توجه به روش های مکمل برای بهبود امنیت و مدیریت ریسک می توان راههای حمله به نرم افزار را یافت و الگوهای تکنیکی مورد استفاده برای این حملات را شناسایی کرد.

الگوهای حمله

عموما حملات چند مرحله ای هستند و مهاجم در هر مرحله ای از حمله، با امکاناتی که بدست می آورد می تواند برای حمله به نقطه هدف در سیستم پیشروی کند.

احتمال نفوذ

تعیین قدرت نفوذ در مراحل مختلف:

- الگوهای حمله در تمام مراحل چرخه عمر توسعه نرم افزار
- الگوهای حمله نیازمندی های امنیتی مثبت و منفی
- الگوهای حمله در معماری و طراحی

- الگوهای حمله در پیاده سازی و برنامه نویسی
- الگوهای حمله در تست نرم افزار امنیتی

راهکارهای تولید نرم افزار امن

- ساخت یک مورد تضمین امنیتی
- گنجاندن موارد تضمین در SDLC
- امنیت تضمین و تلاش در جهت پیروی از آن
- نگهداری و بهره مندی از موارد تضمین
- ضوابط مشترک
- قوانین و آئین نامه امنیتی، حفظ حریم شخصی کاربران

فصل سوم :

اهمیت مهندسی نیازمندی ها :

این عجیب نیست که مهندسی نیازمندی ها برای موفقیت هر پروژه بزرگ در حال توسعه مهم است. برخی از مطالعه ها نشان می دهد اگر نقص مهـندسی نیازمندی ها در سیستم عملیاتی را یکباره اصلاح کنیم هزینه اش نسبت به زمانی که نقص ها را در هنگام توسعه نیازمندی ها بر طرف کنیم، 10 تا 200 برابر هزینه اش بیشتر می شود و سایر مطالعات نشان می دهد که بازسازی، نیازمندی ها- طراحی و کدها نقصی است که بر روی بیشتر پروژه های توسعه نرم افزار وجود دارد و برابر با 40 تا 50٪ کل تلاش پروژه است. درصد نقص های اصلی در طول مهندسی نیازمندی ها بیش از 50٪ تخمین زده شده است. درصد کلی هزینه پروژه، بخاطر نقص نیازمندی ها بین 25 تا 40٪ تغییر می کند. به طور واضح، با توجه به هزینه های ناشی از نیازمندهای امنیتی ضعیف، یک بهبود کوچک در این زمینه ارزش بسیار بالایی دارد. نصب یک برنامه در محیط عملیاتی در جهت بهبود امنیت، به طور قابل توجهی بسیار سخت و هزینه بردار است.

مشکلات نیازمندی ها در بین مسائل مطرح شده به دلیل پدیده های نامطلوب زیر است:

1-پروژه، بیش از بودجه هزینه بردارد، پروژه، از برنامه عقب می مانند، پروژه، به طور قابل توجهی کاهش رتبه دارد و یا کنسل می شود.

2- تیم توسعه برنامه هایی با کیفیت پایین ارائه دهد.

3-محصولات قبل از تحویل استفاده نمیشوند.

در حال حاضر توسعه نرم افزار در یک محیط پویا انجام می گیرد که تغییرات زمانی انجام میگیرد که پروژه در حال توسعه است با این نتیجه که نیازمندی ها دائماً در حال تغییر هستند. این تغییرات می تواند الهام گرفته از درگیری بین گروه های ذینفع، تکامل بازارها با سرعت زیاد، تاثیر تصمیمات معاوضه و ...باشد .

علاوه بر این مهندسی نیازمندی ها بر روی پروژه های شخصی اغلب از مشکلات زیر رنج می برند:

1- به طور معمول شناسایی نیازمندی های همه ذینفعان را شامل نمیشود و از جدیدترین و مدرن ترین تکنیک ها استفاده نمی کند.

2-نیازمندی ها اغلب شرح محدودیت معماری یا مکانیزم های پیاده سازی است به جای شرح آنچه که سیستم باید انجام دهد.

3-نیازمندی ها اغلب به صورت مستقیم و بدون هر گونه تجزیه و تحلیل و یا مدلی هستند. به طور معمول این نیازمندی های عملیاتی کاربر نهایی محدود شده است به :1-نیازمندی های کیفیت مانند امنیت 2- سایر نیازمندی های عملیاتی و غیرعملیاتی 3- محدودیت های معماری، طراحی، پیاده سازی و تست.

4-خصوصیات نیازمندی ها به طور معمول توسط تمامی مخاطبان درنظر گرفته شده با نیازمندهای خاص مثل مبهم بودن، ناقص بودن، ناسازگار،عدم انسجام، غیرممکن، منسوخ اتفاقی است و نه آزمایشی، نه قابل اعتبارسنجی و قابل استفاده نیستند.

5-مدیریت نیازمندهای به طور معمول ضعیف است با داده ضبط شده غیر مفید و ویژگی های از دست رفته اغلب به ردیابی، برنامه ریزی و اولویت بندی بدون تغییر ردیابی یا سایر مدیریت تنظیمات محدود شده است. روش دیگر ممکن است به قابلیت های ارائه شده به وسیله یک وسیله خاص، با اندکی شانس برای بهبود محدود شود.

نیازمندی های کیفیت:

حتی زمانی که سازمان ها اهمیت نیازمندهای عملیاتی کاربر نهایی را تشخیص می دهند همچنان نیازمندیهای کیفیت را نادیده می گیرند مثل عملکرد، ایمنی، امنیت، قابلیت اطمینان و نگهداری. برخی از نیازمندی های کیفیت هستند نیازمندی های غیرکارکردی اما سایر موارد توضیح می دهد قابلیت های سیستم . حتی اگر این به طور مستقیم به نیازمندهای کاربر نهایی کمک نکند. ممکن است انتظار داشته باشید توسعه دهندگان انواع خاصی از سیستم های ماموریت بحرانی و سیستم هایی که زندگی بشر در آن درگیر باشند مانند شاتل فضایی مدت زمانی زیادی است که اهمیت نیازمندی های کیفیت آن به رسمیت شناخته شده است و در توسعه نرم افزار به کار گرفته شده است. در بسیاری از سیستم های دیگر اگرچه نیازمندهای کیفیت یا نادیده گرفته می شود و یا در یک راه ناقص رفتار می کنند. از این رو ما شکست نرم افزارهای مرتبط با سیستم های قدرت، سیستم های تلفن، فضاپیماهای بدون سرنشین و ... را دیدیم. اگر نیازمندی های کیفیت برای این نوع سیستم ها تنظیم نشود خیلی شانس کمی دارند که بر روی سیستم های کسب و کار عادی تمرکز کنند.این عدم توجه به نیازمندی های کیفیت به وسیله تمایل به پایین نگه داشتن هزینه ها و رو به رو شدن با برنامه های تهاجمی تشدید می شود و در نتیجه قراردادهای توسعه نرم افزار اغلب نیازمندیهای کیفیت خاص را شامل نمیشود.

مهندسی نیازمندی های امنیت:

اگر نیازمندیهای امنیت به طور موثر تعریف نشود سیستم نمیتواند برای موفقیت ها و یا شکست قبل از پیاده سازی ارزیابی شود. زمانی که نیازمندی های امنیت در نظر گرفته شود آن ها اغلب از سایر فعالیت های مهندسی نیازمندی ها به طور مستقل توسعه می یابند. به عنوان نتیجه نیازمندی های امنیتی خاص اغلب نادیده گرفته می شوند . در بررسی مستندات نیازمندی ها به طور معمول متوجه می شویم که نیازمندی های امنیتی در یک بخش هستند که به وسیله خودشان و از روی لیست اصلی ویژگی های امنیتی کپی شده است. تجزیه و تحلیل نیازمندی هایی که مورد نیاز هستند تا منجر به تولید یک مجموعه بهتر از نیازمندی های امنیتی شود به ندرت اتفاق می افتد. همان طور که قبلا اشاره شد محیط های عملیاتی و اهداف تجاری اغلب به صورت پویا تغییر می کنند در نتیجه توسعه نیازمندی های امنیتی یک فعالیت یکباره نیست.بنا براین فعالیت هایی که ما در این فصل توضیح میدهیم باید به صورت فعالیت های پی در پی برنامه ریزی شوند.

سو استفاده و موارد سو استفاده:

برای ایجاد نرم افزار امن و قابل اطمینان ما اول باید از رفتارهای ناهنجار سبقت بگیریم . ما به طور معمول در موارد استفاده، رفتارهای ناهنجار را شرح نمیدهیم و با UML توضیح نمیدهیم و ما باید راهی داشته باشیم تا در مورد آن و آماده سازی اش حرف بزنیم. استفاده نادرست از موارد می تواند به شما کمک کند که نرم افزارتان را از دیدگاهی که مهاجم انجام میدهد، ببینید. با فرا رفتن از ویژگی های

هنجاری می توانید به طور همزمان به رویدادهای منفی و غیرمنتظره فکر کنید که این گونه می توانید درک بهتری برای ایجاد یک نرم افزار امن و قابل اطمینان بدست آورید. Guttorm Sindre و Andreas Opdahl نمودار use-caseرا از طریق موارد سو استفاده فعالیت های سیستم که باید منع شوند، توسعه و نمایش دادند و از طریق تجزیه و تحلیل آنها امنیت و نیازمندی های خصوصی را پشتتبانی می کند. Ian Alexander مدافع بهره گرفته از سو استفاده و موارد آن در کنار هم هست که منجر به تجزیه و تحلیل خطر و تهدید در طول تحلیل نیازمندی ها می شود. امنیت، یک مجموعه از ویژگی ها نیست: هیچ منوی کشویی برای انتخاب امنیت وجود ندارد که بعد بتوانید به تماشای جادوی آن بنشینید. متاسفانه بسیاری از توسعه دهندگان نرم افزار به اشتباه فرض می کنند که مکانیزم ها و ویژگی های امنیتی در بخشی از نرم افزار خود منجر به امنیت در سرتاسر سیستم آن ها می شود.در بسیاری از موارد نشان می دهد در فروش محصولات برای تامین امینت از پروتکل SSL و رمزنگاری 128 بیتی استفاده کرده اند. امنیت یک خاصیت ضروری یک سیستم است و نه ویژگی آن. امنیت را باید از ابتدای ساخت محصول در نظر گرفت و به عنوان یک بخش مهم طراحی از ابتدا و در هر فاز از توسعه باید باشد و همه این ها منجر به یک سیستم کامل می شود. ایجاد امنیت از ابتدا در (Systems development life-cycle)SDLC بیانگر روشن شدن نیازمندی های سیستم به طور روشن و صریح است یعنی روش های فنی باید فراتر از ویژگی های آشکار برود. احراز هویت و مجوزها، در ابتدای برنامه نباید متوقف شود. بهترین و مقرون به صرفه ترین روش برای امنیت نرم افزار، باید فراتر از ویژگی های هنجاری رفت و فکر کردن در طول فرآیند توسعه، ادامه یابد. در هر بار که یک نیاز، ویژگی یاuse case جدیدی ایجاد می شود توسعه دهنده یا متخصص امنیت باید زمانی را برای فکر کردن در مورد این که آن آیتم ممکن است خواسته و یا عمدا مورد سو استفاده قرار گیرد، در نظر بگیرد. فکر کردن در مورد کاری که شما نمی توانید انجام دهید: مهاجمان، مشتریان استاندارد نیستند این ها افراد بدی هستند با نیت های مخرب که می خواهند نرم افزار شما به نفع خودشان باشد. اگر فرایند توسعه به رفتارهای ناهنجار و غیر منتظره نپردازد نفوذگر دارای مقدار زیادی مواد خام برای نفوذ است. نفوذگران افرادی بسیار خلاق هستند و در زمینه نفوذ خود در جاهای مشخص جستجو می کنند. مهاجمان باهوش تلاش خواهند کرد که پیش فرض های سیستم را تحلیل کنند. اگر یک طرح فرض کند که اتصالات وب سرور به پایگاه داده همیشه معتبر است به عنوان مثال یک مهاجم سعی می کند که برای دسترسی به داده های با ارزش به وب سرور درخواست نامناسب بفرستد. اگر در طراحی نرم افزار فرض شود که مشتری هرگز کوکی های وب بروزرش را قبل از آنکه برگردند به وب بروز مهاجم تغییرات عمدی بر روی کوکی ها اعمال خواهد شد. ساخت نرم افزار امن به ما می آموزد که مانند یک نگهبان مراقب پیش فرض ها باشیم . زمانی که ما یک سیستم را تجزیه تحلیل و طراحی می کنیم یعنی ما در یک موقعیت بسیار خوبی هستیم زیرا سیستم را از مهاجمان بالقوه بهتر می شناسیم. ما باید این قدرت نفوذ دانش را به نفع امنیت و قابلیت اطمینان را از طریق پرسش و پاسخ های زیر بدست آوریم:

1- کدام فرضیات در سیستم ما مطلق هستند؟

2-چه چیزهایی باعث می شود که فرضیات ما غلط شود؟

3-یک مهاجم از چه الگوهای حمله ای برای تحمیل به سیستم استفاده می کند؟

متاسفانه افراد خلق کننده یک سیستم بهترین آنالیزگز آن سیستم نیستند. به طور کلی در نظر گرفتن تمامی فرضیات برای کسانی که مجموعه ای از فرضیات مطلق را ساخته اند کار سختی است. خوشبختانه این حرفه ها، کارشناسان عالی موضوعی ایجاد می کنند. این متخصصان و تحلیلگران سیستم می توانند با هم طبق فرضیات یک سیستم راه های را که ممکن است یک مهاجم به سیستم داشته باشد را بررسی کنند. ایجاد موارد سوء استفاده های مفید: یکی از اهداف موارد سو استفاده از تصمیم گیری و مستندسازی فرضی است که نرم افزار در برابر استفاده نامشروع چگونه واکنش نشان می دهد؟

ساده ترین و عملی ترین روش برای ایجاد موارد سو استفاده از طریق فرآیند طوفان ذهنی انجام میگیرد. متدهای نظری به یک سیستم کامل با منطق و مدل های دقیق رسمی نیاز دارند اما این فعالیت ها به زمان و منابع بسیار زیادی نیاز دارند. برای هدایت طوفان ذهنی، متخصصین امنیت نرم افزار سوالات بسیاری از طراحان سیستم می پرسند تا به تشخیص مکانی که ممکن است سیستم ضعف داشته باشد کمک کنند. این طوفان ذهنی یک نگاه دقیق به همه رابط های کاربری و رویدادها را شامل می شود که یک شخص نمی تواند یا نمیخواهد، این نتوانستن و نخواستن اشکال مختلفی دارد. مثال کاربر نمیتواند بیشتر از 50 کاراکتر وارد کند چون کدهای جاوا اسکریپ به او این اجازه را نخواهد داد. در نتیجه این موارد را نمیتوان تغییر داد. متاسفانه مهاجم ها این نتوانستن ها و نخواستن ها را به واقعیت تبدیل کردند. فرآیند تعیین موارد سو استفاده باعث می شود که طراح، موارد استفاده درست از نادرست را تشخیص دهد. البته برای رسیدن به این نقطه طراح باید سوالات درستی را مطرح کند مانند: سیستم چگونه می تواند ورودی خوب را از بد تشخیص دهد؟ می تواند بگوید که یک درخواست چه از یک برنامه قانونی و یا از یک برنامه غیرقانونی باشد. ترافیک را بررسی می کند؟آیا همه سیستم ها نقاط آسیب پذیریشان بیشتر از ورودی های مستقیمشان هست؟ از سیم؟ در محیط های کاری؟ هر خط ارتباطی بین دو نقطه انتهایی یا دو جز یک مکانی است که مهاجم ممکن است خودش را در آن بین قرار دهد پس این مهاجم در این سیستم چه کاری می تواند انجام دهد؟ ترافیک ارتباطی چگونه است؟ تغییر و پخش کردن ترافیک؟ خواندن فایل های ذخیره شده در ایستگاه های کاری؟تغییر کلیدهای رجیستری یا تنظیمات فایل ها ؟

تلاش برای پاسخ دادن به چنین سوالاتی کمک به شفافیت سوالات طراحان نرم افزار و فرضیات معماری می کند که این گونه، طراح می تواند از مهاجم در زمینه شناسایی مشکلات پیشی گیرد.

نمونه ای از یک سو استفاده:

در این بخش یک مثال از مشکل امنیتی نرم افزار کلاسیک بر روی برنامه کلاینت/سرور را توضیح می دهیم. معماری به گونه ای تنظیم شده است که سرور براساس برنامه های کلاینت سایدی کار می کند که دستکاری های حساس مالی در دیتابیس که باید تمام دسترسی های مجاز به دیتابیس بررسی شود. شرایط بدتری را در نظر بگیرید، یک کپی کامل از دیتابیس به برنامه های کلاینی ارسال شود تا روی کامپیوتر های شخصی اجرا شود. در نتیجه یک کپی کامل از اطلاعات حساس بر روی کامپیوترهای کاربران در دسترس است. اگر کاربر به هارد دیسک برود و cache برنامه را ببیند می تواند اطلاعات را تفکیک کند و اطلاعات حساس را بدست آورد. کاربر پیام های که به سرور ارسال شده است را به اجرا گذاشته است این پیام ها به اعتبار کاربر وابسته نیستند. سرور فرض می کند که هر پیامی که از کلاینت به او می رسد قانونی است زیرا از سیستم کنترل دسترسی نرم افزار کلاینت عبور کرده است. با متوقف کردن ترافیک شبکه، مقادیر مشکوکی که در cache نرم افزار کلاینت است، ایجاد کلاینت متخاصم و کاربران مخرب داده به دیتابیس تزریق میکنند که آن ها قادر به خواندن آنها نیستند .تعیین کردن نتوانستن ها و نخواستن ها برای کسانی که فقط به جنبه های مثبت می پردازند سخت است. الگوهای حمله می تواند راهنمایی هایی در این زمینه ارائه کند. به طور مثال overflow کردن buffer از چندین الگوی استاندارد متفاوت پیروی می کند اما الگوها یک مقدار تغییر منصفانه موارد را اجازه می دهند. زمانی که ما تلاش می کنیم برای توسعه موارد استفاده نادرست و سو استفاده های آن می توانیم از الگوهای حمله کمک بگیریم.

مدل پردازش مربعی: Security Quality Requirements Engineering

مهندسی نیازمندی های کیفیت امنیت، یک مدل پردازشی است که در دانشگاه Carnegie Mellon توسعه یافته است که ابزاری برای استخراج، طبقه بندی و اولویت بندی نیازمندی های امنیتی برای سیستم های فناوری اطلاعات و برنامه ها فراهم می کند. تمرکز این مدل بر روی ایجاد مفاهیم امنیتی در مراحل اولیه ی SDLC است. این همچنین می تواند برای مستندسازی و تجزیه و تحلیل مفاهیم امنیتی سیستم ها استفاده شود. آن ها در یک زمینه پیاده سازی می شوند و به پیشرفت ها و تغییرات آن سیستم ها جهت می دهد. بعد از این توسعه اولیه، از SQUARE در یک سری از مطالعات موردی کلاینت استفاده شد. ابزار های Prototype برای پشتیبانی پردازش ها توسعه داده شد. پیش نویس فرآیند، به عنوان یک baseline بعد از اینکه موارد مطالعه شده کامل شود مورد تجدید نظر قرار میگیرد. فرآیند baselined در نه گام زیر نمایش داده شده است.

در اصل، گام های یک تا چهار فعالیت هایی هستند که قبل از مهندسی نیازمندی های امنیتی باید انجام شوند و اطمینان از موفقیت آنها ضروری است. (9 تا گام)

1- توافق در تعاریف
2- شناخت اهداف امنیتی
3- توسعه مصنوعات برای پشتیبانی از تعریف نیازمندی های امنیت
4- اجرای ارزیابی ریسک
5- انتخاب روش استخراج
6- استخراج نیازمندی های امنیتی
7- طبقه بندی نیازمندی ها
8- اولویت بندی نیازمندی ها
9- رسیدگی به نیازمندی ها

توضیح مختصری از SQUARE :

فرآیند SQUARE بهترین پاسخی است که به وسیله مهندسی نیازمندی های امنیتی و متخصصان امنیت در زمینه حمایت مدیریت اجرایی و ذینفعان داده می شود. این فرآیند زمانی بهترین عملکرد را دارد که استخراج بعد از ارزیابی ریسک رخ دهد و نیازمندی های امنیتی قبل از معماری بحرانی و تصمیمات طراحی مشخص شوند. بنا بر این ریسک های امنیتی بحرانی در یک تجارت در توسعه نیازمندی های امنیتی در نظر گرفته خواهد شد.

گام اول: توافق در تعاریف : به عنوان پیش نیاز مهندسی نیازمندی های امنیتی در نظر گرفته می شود. در یک پروژه افراد تیم تمایل دارند تعاریفی که در ذهن دارند را نگه دارند و براساس تجربیات گذشته خود کار کنند اما این تعاریف اغلب متفاوت هستند به طور مثال در

برخی از سازمان های دولتی برای پیاده سازی امنیتی از لیبل های امنیتی برای برخی ها استفاده می کنند در حالی که برای دیگران امنیت را به صورت فیزیکی یا سایبری پیاده می کنند. لازم نیست که شما تعاریف را خلق کنید می توانید از تعاریف سازمانهایی مانند IEEE و SWEBOK استفاده کنید و حتی سفارش دهید.

گام دوم: شناخت اهداف امنیتی: باید در سطوح سازمانی انجام گیرد و نیاز به حمایت از توسعه نرم افزار در طول پروژه است. این مرحله بررسی سازگاری با سیاست های سازمان و محیط امنیتی عملیاتی را فراهم می کند. ذینفعان مختلف دارای اهداف متفاوتی هستند. به طور مثال، یک ذینفع در منابع انسانی ممکن است در مورد حفظ محرمانگی سوابق کارکنان نگران باشند. در حالی که یک ذینفع در حوزه مالی ممکن است در مورد داده های مالی در مورد تغییر و یا مجوز آن نگران باشد.این مهم است که یک نماینده با تخصص های عملیاتی از ذینفعان داشته باشیم . زمانی که اهداف ذینفعان مختلف شناسایی شده است آن ها نیاز به اولویت بندی دارند. در صورت عدم توافق یک تصمیم اجرایی ممکن است نیاز به اولویت بندی آن ها باشد.

گام سوم: توسعه مصنوعات: برای حمایت از تمام فعالیت های مهندسی نیازمندی ها امنیتی بعدی ضروری است. اغلب، سازمان ها این موارد را ندارند: مستنداتی از مفاهیم عملیاتی یک پروژه، شرح اهداف پروژه، استفاده از مستندات عادی و سناریو های تهدیدات، سو استفاده و یا موارد سو استفاده و سایر موارد مورد نیاز برای حمایت از نیازمندی ها. در نتیجه کل فرآیند نیازمندی ها ساخته شده است بر اساس فرضیات مطرح نشده یا زمان زیادی را صرف backtracking می کند و تلاش زیادی را برای مستند سازی انجام میدهند.

گام چهارم:اجرای ارزیابی ریسک: نیاز به یک متخصص در زمینه ارزیابی خطر، حمایت از سهامداران و حمایت از مهندسی نیازمندی های امنیتی دارد. کارشناس ارزیابی ریسک می تواند یک روش خاص را براساس نیازهای خاص سازمان پیشنهاد دهد. مصنوعات گام سوم، ورودی فرآیند ارزیابی ریسک را مهیا میکند. شرکت هایی که به طور معمول ارزیابی ریسک انجام نمیدهند یک روش منطقی برای ریسک آن سازمان ندارند. زمانی که نیازمندی های امنیتی تشخیص داده می شود اما همچنان تمایل دارند که راه حل یا تکنولوژی خاصی مانند رمزنگاری انتخاب کنند بدون اینکه در نظر بگیرند که چه مشکلی را حل خواهد کرد.

گام پنجم: انتخاب روش استخراج: زمانی مهم است که پروژه ذینفعان متفاوتی دارد. مدل های معمول استخراج مانند روش نیازمندی سریع، طراحی نرم افزار مشترک یا مصاحبه ساخت یافته می توانند در غلبه کردن بر مسائل ارتباطی زمانی که ذینفعان با زمینه های فرهنگی متفاوت هستند، موثر باشند. در موارد دیگر استخراج ممکن است از یک ذینفع اولی تشکیل شود و تلاش می کند که نیازمندی های امنیتی ذینفعان را درک کند.

گام ششم: استخراج نیازمندی های امنیتی: فرآیندی است که استخراج واقعی را از طریق روش های انتخاب شده از طریق راهنمایی های دقیق در مورد چگونگی انجام استخراج انجام میدهد.

گام هفتم: طبقه بندی نیازمندی ها: به مهندسی نیازمندی های امنیتی اجازه می دهد که بین نیازهای اساسی، اهداف و محدودیت های معماری که ممکن است وجود داشته باشد تشخیص دهد. زمانی که معماری سیستم قبل از فرآیند نیازمندی ها انتخاب شود نیازمندی ها به طور معمول محدودیت ایجاد میکنند. این حالت برای زمانی که ارزیابی مربوط به خطرات وابسته به آن محدود می شود، خوب است. این طبقه بندی به اولویت بندی فعالیت ها کمک می کند.

گام هشتم: اولویت بندی نیازمندی ها: فقط به طبقه بندی وابسته نیست بلکه ممکن است درگیر تجزیه و تحلیل سود و زیان مالی باشد که در نظر می گیرد که نیازمندی های امنیتی یک ارتباط قوی با مسائل مالی دارد. اولویت بندی ممکن است به عواقب نقص های امنیتی مانند از دست دادن زندگی، از دست دادن اعتبار و از دست دادن اعتماد مشتری وابسته باشد.

گام نهم: رسیدگی به نیازمندی ها : می تواند در سطوح مختلف انجام شود. هنگامی که این رسیدگی انجام می شود تیم پروژه باید یک مجموعه از نیازمندی های امنیتی اولویت بندی شده اولیه داشته باشد. باید تشخیص دهیم که کدام ناحیه ناقص است و نیاز به بازرسی مجدد دارد. در نتیجه تیم پروژه باید درک کند که چه مناطقی وابسته به معماری و پیاده سازی خاص است و مجدداً به بازرسی بپردازند.

ابزارها: یک ابزار اولیه توسعه یافته برای حمایت از SQUARE است. این در درجه اول یک چهارچوب سازمانی برای اسناد مهیا می کند. ابزار ، ساختارهای پیچیده مانند تحلیل نیازمندی ها را اجرا نمی کند. استفاده از یک روش استخراج می تواند در تولید یک مجموعه منسجم و کامل از نیازمندی های امنیتی کمک کند. با این حال، همفکری گروهی و روش های استخراج برای نیازهای عادی نسبت به نیازمندی های امنیتی استفاده می شود و در نتیجه، در یک مجموعه منسجم و کامل از نیازهای امنیتی نیستند. دستاورد این سیستم نسبت به زمانی که نیازمندیها با یک روش سیستماتیک استخراج میشوند به احتمال زیاد از امنیت کمتری برخوردار است. در این بخش، ما به طور خلاصه درباره تعدادی از روش های استخراج و تجزیه و تحلیل که می تواند منجر به انتخاب یک روش مناسب شود صحبت می کنیم. مطالعات موردی را می توان در " مطالعات استخراج نیازمندی ها " یافت. در حالی که نتایج ممکن است از سازمانی به سازمان دیگر متفاوت باشد. استخراج نیازمندیها، یک حوزه پژوهشی فعال است و انتظار داریم که در آینده شاهد پیشرفت در این زمینه باشیم. در نهایت، مطالعات احتمالاً تعیین خواهد کرد که کدام روش برای استخراج نیازمندی های امنیتی موثر است. با این حال، در حال حاضر اطلاعات کمی برای مقایسه روش های مختلف برای استخراج نیازمندی های امنیتی وجود دارد.

بررسی اجمالی چندین روش استخراج:

فهرست زیر شناسایی چندین روش برای استخراج نیازمندی های امنیتی در نظر گرفته شده است را نشان می دهد . برخی به طور خاص با امنیت در ذهن (به عنوان مثال ، از موارد سوء استفاده) توسعه یافته اند ، در حالی که موارد دیگر در مهندسی سنتی استفاده شده است و به طور بالقوه می تواند به نیازمندی های امنیتی تمدید شود . در آینده ، ما ممکن است درک بهتری از چگونگی جنبه های منحصر به فرد از انتخاب روش های استخراج نیازمندی های امنیتی داشته باشیم. ما همچنین کار های اخیر در مورد استخراج نیازها را که به طور کلی می تواند در انجام این فرآیند انتخاب، مورد توجه باشد را در یک لیست در نظر گرفته و به طور خلاصه هر یک از روش ها را شرح می دهیم:

موارد سوء استفاده، روش سیستم های نرم، گسترش کیفیت عملکرد، نیازهای کنترل شده، سیستم های اطلاعات پایه، برنامه توسعه مشترک، تجزیه و تحلیل دامنه ویژگی، تحلیل گفتمان انتقادی، روش های تسریع نیازها

موارد سوء استفاده:

همانطور که قبلا اشاره شد، سو استفاده از مفهوم استفاده از یک سناریو یک وضعیت که صاحب سیستم نمیخواهد رخ دهد، در یک زمینه مورد استفاده منفی است. به عنوان مثال، رهبران کسب و کار ، برنامه ریزان نظامی و بازیکنان بازی، با استراتژی تجزیه و تحلیل بهترین حرکت حریف خود را به عنوان تهدید شناسایی می کنند. در مقابل، یک مورد استفاده به طور کلی توضیح رفتاری که صاحب سیستم می خواهد را نشان می دهد. مدل مورد استفاده و نمودار همراه آن(UCD ها) برای مشخصات مورد نیاز بسیار مفید هستند. با این حال،

مجموعه ای از موارد استفاده باید به عنوان یک جایگزین برای یک سند خصوصیات مورد نیاز استفاده شود، به عنوان این رویکرد می تواند در تسلط بر نیازمندیها، بحث استفاده از مدل های مورد استفاده برای سیستم و کیفیت استخراج نیازمندی ها می تواند قابل توجه باشد.

سیستم های نرم افزاری(روش SSM):

SSM به مشکلاتی که در شرایط اجتماعی، سیاسی و فعالیت های انسانی وجود دارد، می پردازد . SSM می تواند به " مشکلات نرم" به جای " مشکلات سخت" که بیشتر تکنولوژی گرا هستند، رسیدگی کند. نمونه هایی از مشکلات نرم شامل نحوه برخورد با بی خانمانی ، نحوه مدیریت برنامه ریزی فاجعه و چگونگی بهبود مراقبتهای پزشکی است . در نهایت مشکلات تکنولوژی گرا ممکن است از این مشکلات نرم ظاهر شوند، اما نیاز به تجزیه و تحلیل خیلی بیشتری برای رسیدن به آن مورد است. فایده اولیه SSM فراهم کردن ساختار وضعیت مشکل های نرم در شیوه ای سازمان یافته است. علاوه بر این، توسعه دهنده گان را وادار به کشف راه حل های فراتر از تکنولوژی می کند .

گسترش عملکرد کیفیت (QFD):

QFD " یک مفهوم کلی فراهم می کند که نیاز مشتری را با الزامات فنی مناسب برای هر مرحله از توسعه محصول و تولید شرح می دهد " ویژگی متمایز QFD تمرکز بر نیازهای مشتری در تمام فعالیت های توسعه محصول است. با استفاده از QFD ، سازمان ها می توانند کار گروهی را ترویج دهند، اولویت بندی فعالیتهای عملی، تعریف اهداف روشن ، و کاهش زمان توسعه داشته باشند.

گشترش کیفیت کارکرد (CORE):

CORE تجزیه و تحلیل نیازمندیها و روشی مشخص و واضح است که کاربر از خدمات توسط سیستم پیشنهادی تهیه میکند. در CORE ، مشخصات مورد نیاز توسط کاربر و توسعه یا یکی از آنها ایجاد شده است. مشکلات تجزیه و تحلیل را از دیدگاه کاربر و توسعه دهنده بررسی می کند . اطلاعات مجموعه ای از نقطه نظرات که مورد تجزیه و تحلیل قرار می گیرد، است. آخرین مرحله از معاملات CORE با تجزیه و تحلیل محدودیتها، از جمله محدودیت های محیط عملیاتی سیستم ، تحقیق در رابطه با درجه ای از عملکرد و بررسی قابلیت اطمینان می باشد.

شماره بر اساس سیستم های اطلاعاتی (IBIS):

روش IBIS بر این اصل است که فرایند طراحی برای مشکلات پیچیده ، که به نظر Rittel مشکلات نابکار ، ضرورتاً در میان ذینفعان است که در آن هر یک از ذینفعان چشم انداز فردی خود را به وضوح بر اساس مسائل مربوط به طراحی به ارمغان می آورند. هر مشکل، نگرانی، یا سوال می تواند یک مسئله باشد و ممکن است بحث و حل و فصل برای طراحی به ادامه نیاز داشته باشد.

توسعه کاربرد مشترک (JAD):

JAD روشی است که به طور خاص برای توسعه سیستم های کامپیوتری بزرگ طراحی شده است. هدف آن این است که شامل همه ذینفعان در مراحل طراحی محصول از طریق جلسات متمرکز باشد. در مراحل اولیه JAD، تیم مهندسی باید وظایف عنوان شده اش در مورد حقیقت یابی و جمع آوری اطلاعات مورد نیاز را انجام دهد. به طور معمول، خروجی این مرحله به عنوان استخراج نیازمندی های امنیتی و

اهداف امنیتی محصول می باشد. جلسه JAD بطور واقعی استفاده از اعتبار این اطلاعات از طریق ایجاد یک مجموعه توافق در مورد نیازمندی های امنیتی برای محصول است.

تجزیه و تحلیل دامنه ویژگی گرا (FODA):

FODA یک روش تجزیه و تحلیل دامنه و مهندسی روش ها است که بر روی توسعه دارایی های قابل استفاده مجدد تمرکز دارد. با بررسی سیستم های نرم افزاری مرتبط و نظریه اساسی از یک سری از سیستم ها، نشان می دهد که تجزیه و تحلیل دامنه می تواند یک توصیف کلی از الزامات این کلاس از سیستم در قالب یک مدل دامنه و مجموعه ای از روش ها برای اجرای آنها را فراهم کند. روش FODA در دو مفهوم مدل سازی شده است انتزاع و پالایش. انتزاع برای ایجاد مدل از برنامه های کاربردی خاص در دامنه است. برنامه های کاربردی خاص در این حوزه هستند و سپس به عنوان اصلاحات از مدل دامنه توسعه یافته و پالایش میشوند. به عنوان مثال دامنه مورد استفاده در گزارش های اولیه در FODA سیستم های مدیریتwindow است .

تحلیل گفتمان انتقادی (CDA):

CDA به استفاده از روش های زبانشناسی اجتماعی و تجزیه و تحلیل نوشتاری و گفتار به طور خاص میپردازد. از این روش در تجزیه و تحلیل استخراج نیازمندی ها، مصاحبه ها و روایات و داستان استفاده میشود.

روش نیازمندیهای تسریع شده (ARM):

روند ARM تسهیل استخراج نیازمندیها و شرح فعالیت ها است که شامل سه مرحله است:
مرحله آماده سازی، مرحله آسان سازی، مرحله نهایی تحویل روند ARM شبیه به JAD است، اما با برخی از تفاوت های قابل توجه از روش پایه JAD ، که به منحصر به فرد بودن آن کمک کند. به عنوان مثال، در این فرایند، برای تسهیل از ظرفیت و تکنیک های پویای گروه استفاده می شود متفاوت ازچیزی که در JAD می باشد ، از تکنیک همفکری گروهی و ثبت مدل های مفهومی مختلف استفاده می شود.

معیارهای ارزیابی استخراج:

در زیر به عنوان مثال معیارهای ارزیابی که ممکن است در انتخاب روش استخراج مفید باشند آمده است ، هر چند که مطمئنا شما هم می توانید از معیارهای دیگر استفاده کنید . نکته اصلی انتخاب یک مجموعه ای از ضوابط ودرک مشترک از معنی آنها است.

سازگاری :این روش را می توان برای تولید نیازمندیها در محیط های متعدد استفاده کرد . به عنوان مثال، روش استخراج به همان اندازه کاربرد دارد در یک محصول نرم افزاری که مراحل برنامه ریزی در تکمیل یک پروژه دارد.

پذیرش ذینفعان: سهامداران به احتمال زیاد در روش استخراج نیازمندیها برای تجزیه و تحلیل نیازهای خود به توافق میرسند. به عنوان مثال، این روش در یک محیط تجاری بیش از حد تهاجمی نیست.

اجرای آسان: روش استخراجی است که بیش از حد پیچیده نیست و می تواند به درستی و راحتی اجرا شود.

خروجی گرافیکی: روش تولید مصنوعات بصری به آسانی قابل درک است.

اجرای سریع: نیازهای مهندسان و سهامداران به طور کامل از روش استخراج در یک مقطع زمانی معقول میتواند تامین شود.

منحنی یادگیری سطحی: نیاز مهندسان و سهامداران به طور کامل می تواند در طول زمان معقول با روش استخراج درک شود.

بلوغ بالا: روش استخراج در معرض توجه و تجزیه و تحلیل نیازها و جامعه مهندسی نیازمندیها قرار گرفته است.

مقیاس پذیری: این روش می تواند برای استخراج نیازمندیهای پروژه های با اندازه های مختلف استفاده شود ، از سیستم های در سطح سازمانی تا برنامه های کاربردی در مقیاس کوچک.

توجه داشته باشید که در این روش فرض میشود که همه معیارها به یک اندازه مهم است. اگر برخی از معیارهای از بقیه مهمتر هستند ،به طور میانگین می توانند مورد استفاده قرار گیرند . به عنوان مثال ، در دسترس بودن ابزار CASE ممکن است از خروجی گرافیکی مهم تر باشد. طرح وزن معمولی می تواند معیار شود " ضروری" با وزن 3 ، " مطلوب" با وزن 2 ، و " اختیاری " با وزن 1 .
ما تصمیم به استفاده از JAD، ARM ، و IBIS در سه طرح مختلف گرفتیم. این سه روش ذهنی رتبه بندی به عنوان دو نامزد مناسب برای مطالعات موردی، با توجه به زمان و تلاش محدودیت برای این پروژه است به نظر ما فقط این نمره کل نیست: منحنی یادگیری یک عامل مهم بود و این تیم اقدام به انتخاب روش هایی که بیش از حد شبیه به یکدیگر نیستند کرد، بنابراین شامل برخی از تفاوت ها است. در مطالعات ما، ما بیشترین موفقیت با استفاده از ARM در شناسایی نیازمندی های امنیتی بود. نتایج دقیق برای هر سه روش را می توان در مهندسی نیازمندیها از بخش تولید امنیت در وب سایت بیابید .

`

توضیحات بیشتر :

این امکان وجود دارد که ترکیبی از روش های مختلف ممکن است بهتر کار کنند. شما باید این گزینه را به عنوان بخشی از فرایند ارزیابی در نظر بگیرید ، فرض کنید که شما وقت کافی و منابع ارزیابی به میزان روش های ممکن را ترکیب شده و در واقع آنها را ترکیب کرده اید . شما همچنین باید زمان لازم را برای اجرای یک روش استخراج خاص و زمان لازم برای یادگیری یک ابزار جدید که این روش را پشتیبانی می کند در نظر بگیرید. انتخاب روش استخراج مورد نیاز که پاسخگوی نیازهای گروه های مختلف ذینفعان باشد در پرداختن به طیف وسیع تری از نیازمندی های امنیتی کمک میکند .

اولویت نیازمندیها :

هنگامی که مجموعه ای از نیازمندی های امنیتی شما را شناسایی کرده اید ،معمولا می خواهید آنها را اولویت بندی کنید . با توجه به وجود محدودیت های زمان و بودجه ، استخراج همه نیازمندیها که برای یک سیستم مشخص لازمه سخت میشود. همچنین، نیازمندی های امنیتی اغلب در مراحل اجرا و اولویت بندی می تواند کمک کنند به تعیین این که کدام یک باید اول اجرا شود. بسیاری از سازمان ها پایین ترین هزینه را برای نیازمندیهای خود در ابتدا می پردازند ، بدون اینکه به اهمیت آن توجه کنند. راه آسانتری را میتوان انتخاب کرد به عنوان

مثال، با خرید یک راه حل COTS . این تنها روش برای رسیدن به اهداف امنیتی سازمان یا پروژه نیست. برای اولویت بندی نیازمندی های امنیتی در مد منطقی بیشتر، یک رویکرد اولویت بندی سیستماتیک توصیه می کنیم. در این بخش انتخاب یک روش مناسب اولویت بندی نیازمندیها و به طور خلاصه شرح تعدادی از روشها مورد بحث و تجزیه و تحلیل قرار میگیرد . ما همچنین یک روش اولویت بندی نیازها با استفاده از AHP را مورد بحث قرار می دهیم . پوشش گسترده از این مواد در دسترس است.

در حالی که نتایج ممکن است برای سازمان شما متفاوت باشد، بحث از تکنیک های مختلف باید مورد توجه باشد. کارهای زیادی باید انجام شود قبل از اولویت بندی نیازمندی های امنیتی در نظر گرفته شده در یک ناحیه، اما ازاینجا است که ما باید شروع به رسیدگی کنیم.

شناسایی روشهای اولویت دهی کاندید:

تعدادی از روش های اولویت بندی در مهندسی نیازمندیهای سنتی مفیدند و می توانند به طور بالقوه برای توسعه نیازمندی های امنیتی استفاده شود. ما به طور خلاصه در اینجا ذکر درخت جستجوی دودویی ، روش بر اساس اعداد تخصیص ، برنامه ریزی بازی، روش 100 نقطه ، نظریه -W ، ارزیابی نیازمندیها ، روش Wiegers ، چارچوب اولویت بندی نیازها و AHP . اطلاعات بیشتر را می توانید در ساخت امنیت در وب سایت و در منابع بیابید.

درخت جستجو دودویی (BST):

درخت جستجوی دودویی یک الگوریتم است که به طور معمول در یک جستجو برای اطلاعات مورد استفاده قرار گیرد و به راحتی می توانید در اولویت بندی بسیاری از نیازهای کوچک استفاده شود. رویکرد اساسی مورد نیاز به شرح زیر است:

1- قرار دادن تمام نیازهای در یک پایه
2- نگاهی به یک نیازمندی وقرار دادن آن را در یک گره ریشه
3- نگاهی به نیازهای دیگر و مقایسه آن با گره ریشه .
4- اگر اهمیت نیازمندی از ریشه کمتر است ، آن را نسبت به گره فرزند چپ. اگر نیاز از گره ریشه مهم تر است ، آن را نسبت به گره فرزند سمت راست مقایسه میکنند. اگر گره هیچ گره فرزندی نداشت، قرار دادن گره فرزند جدید را به سمت راست یا چپ ، بسته به اینکه آیا نیازمندی مهمتر است یا نه.
5- تکرار مراحل 3 و 4 تا وقتی که تمام نیازمندی ها مقایسه شوند ودر یک BST قرار داده شوند.
6- برای ارائه اهداف ، گذاشتن تمام BST مورد نیاز در یک لیست ، که کم اهمیت ترین نیاز در انتهای لیست و مهم ترین نیاز در آغاز این فهرست قرار گیرد.

روش تخصیص اعداد:

این روش برای هر نیاز مقیاس فراهم می کند. تقسیم نیازها به سه گروه : اجباری، دلخواه و غیر اصلی. اختصاص یک عدد در مقیاس 1 تا 5 برای نشان دادن اهمیت هر نیازمندی . رده بندی نهایی به طور متوسط از رتبه بندی تمام شرکت کنندگان برای هر مورد نیازمندی است .

برنامه ریزی بازی :

این یکی از ویژگی های فوق العاده برنامه نویسی است که توسط مشتریان برای اولویت بندی ویژگی های مورد نظر استفاده می شود ، این یک نوع متفاوت از روش تخصیص اعداد است، که در آن توزیع نیازها به سه گروه است : " کسانی که بدون سیستم عمل نمی کنند، آنهایی که ضرورت کمتری دارند اما شاخص های تجاری مهم ارائه میکنند ، " و " آنهایی که می خواهند به خوبی داشته باشند. "

روش 100 نقطه:

روش 100 نقطه اساسا یک طرح رای گیری است که در تمرینات همفکری گروهی استفاده می شود. به همه ذینفعان 100 امتیاز داده شده است که می توانند برای رای دادن به مهم ترین نیازها استفاده کنند. 100 امتیاز می تواند در هر راهی که ذینفعان می خواهند تقسیم شود. برای مثال، اگر چهار نیازمندی وجود دارد که دیدگاه ذینفعان برای اولویت بندی آنها برابر است، می توان 25 نقطه برای هرکدام قرار داد. اگر یک نیازمندی وجود دارد که ازدیدگاه ذینفعان مهمتر است، می تواند 100 داده شود. با این حال، این نوع از برنامه تنها برای رای اولیه کار می کند. اگرمجدد رای گیری شود، مردم به احتمال زیاد برای توزیع مجدد رای خود تلاش میکنند تا طرح خود را در اولویت قرار دهند.

نظریه – W:

نظریه – W (همچنین به عنوان " برنده برنده" شناخته می شود) در ابتدا در دانشگاه کالیفرنیای جنوبی در سال 1989 توسعه داده شد . این روش از مذاکره برای حل اختلافات در نیازمندیها پشتیبانی می کند، به طوری که هر یک از ذینفعان برنده باشند. این بر دو اصل متکی است : طرح پرواز و پروازطرح، شناسایی و مدیریت ریسک.
اصل اول به دنبال ایجاد برنامه های خوش ساخت است که با استانداردهای از پیش تعریف شده برای توسعه آسان ، طبقه بندی، و پرس و جو صورت می گیرد . " پرواز از طرح " پیشرفت زیربرنامه های اصلی را تضمین می کند.

اصل دوم ، " شناسایی و مدیریت ریسک ، " شامل ارزیابی ریسک و مدیریت ریسک است. که برای محافظت از شرایط "برد برد" سهامداران در برابرنقض قوانین استفاده می شود . در مذاکرات برنده برنده ، هر کاربر باید نیازمندیهای خصوصی خود را قبل از شروع مذاکرات رتبه بندی کند. در این روند رتبه بندی منحصر به فرد ، کاربر در نظر می گیرد که آیا او حاضر به واگزاری نیازمندیهای مسلم خود در شرایط خاص است، به طوری که شرایط برد و باخت را به طور کامل درک کند.

ارزیابی نیازمندیها :

ارزیابی نیازمندیها یک فرآیند چند مرحله ای است که شامل ایجاد اولویت های نسبی مورد نیاز، برآورد منابع مورد نیاز برای ارضای هر نیاز، و انتخاب یک زیر مجموعه از الزامات مورد نیاز برای بهینه سازی احتمال موفقیت محصول در بازار می باشد. این تکنیک به وضوح در توسعه دهندگان محصولات نرم افزاری در بازار تجاری دیده میشود. کتاب اخیر دیویس ، همکاری بین توسعه نرم افزار و بازاریابی را مفصل شرح می دهد، توصیه می کنیم اگر به این رویکرد توجه دارید آن را بخوانید. ارزیابی نیازمندیها یک روش منحصر به فرد است که ارزش بررسی دارد، هر چند به وضوح فراتر از اولویت بندی نیازمندیهای سنتی در عوامل کسب و کار تجاری است.

روش Wiegers:

روش Wiegers به طور مستقیم مربوط به ارزش هر نیاز مشتری است. اولویت محاسبه با تقسیم مقدار نیازمندیها از مجموع هزینه ها و ریسک های فنی در ارتباط با اجرای آن است. ارزش یک نیاز وابسته به ارزش های مشتری ، به مشتری ارائه شده و در صورتیکه نیازها از دست بروند جریمه پرداخت میشود. با توجه به این دیدگاه ، توسعه دهندگان باید هزینه های مورد نیاز و خطرات اجرای آن و همچنین جریمه نیازمندیهای از دست رفته را ارزیابی کنند . ویژگی ها در مقیاس 1 تا 9 مورد بررسی قرار می گیرد.

چارچوب مورد نیاز اولویت بندی:

الزامات چارچوب اولویت بندی و ابزار مرتبط با آن شامل هر دو استنباط و فعالیت های اولویت بندی است . این چارچوب برای رسیدگی به مسائل زیر در نظر گرفته شده است:

استخراج اهداف کسب و کار سهامداران این پروژه، امتیاز سهامداران با استفاده از مدل مشخصات ذینفعان، اجازه دادن به سهامداران برای درجه بندی اهمیت نیازمندیها و اهداف تجاری با استفاده از یک مقیاس درجه بندی گرافیک فازی، امتیاز نیازمندیها بر اساس معیارهای عینی، پیدا کردن وابستگی بین الزامات و نیازمندی های خوشه بندی شده به طوری که اولویت بندی برای آنها موثر باشد، با استفاده از روش تجزیه و تحلیل ریسک برای تشخیص گروه ها در میان سهامداران ، انحراف در میان ذینفعان برای رتبه بندی شخصی، و ارتباط بین سهامداران ورودی و رتبه بندی نهایی.

AHP:

AHP روشی است برای تصمیم گیری در شرایطی که اهداف چندگانه وجود داشته باشد. این روش از یک "جفت عاقلانه" استفاده میکند ماتریس برابری نسبت به محاسبه ارزش و هزینه های مورد نیاز امنیت نیازمندیها نسبت به یکدیگر است. با استفاده از AHP ، مهندسی نیازمندیها می تواند ثبات در نتیجه را تایید کند . AHP می تواند از اشتباهات فردی جلوگیری کند و احتمال وجود نتایج قابل اعتماد را افزایش دهد. آن توسط یک ابزار مستقل و همچنین با کمک محاسباتی SQUARE tool پشتیبانی می شود.

مقایسه تکنیک اولویت بندی:

ما توصیه می کنیم به مقایسه با چند تکنیک اولویت بندی برای کمک به انتخاب یک روش مناسب. برخی از معیارهای ارزیابی به عنوان مثال در اینجا ارائه شده است :

گام های روشن : تعریف روشنی است بین مرحله یا مراحلی که در روش اولویت بندی وجود دارد.

اندازه گیری کمی : خروجی عددی روش اولویت بندی به وضوح اولویت های مشتری برای همه نیازمندیها را نشان می دهد .

بلوغ بالا : این روش تا به حال در معرض توجه و تجزیه و تحلیل توسط جامعه مهندسی نیازمندیها قرار گرفته است .

کاربری پایین : A تعداد مناسب از ساعت وقت لازم است به درستی اجرا از روش اولویت بندی .

منحنی یادگیری سطحی : نیازمندیهای مهندسان و سهامداران به طور کامل می تواند در یک مقطع زمانی معقول درک شود.

توجه داشته باشید که در این روش رسیدگی به اهمیت هر معیار در نظر نیست. همچنین ممکن است یک متوسط وزنی ساخته شود برای وقتیکه می خواهید از روش مقایسه استفاده کنید. به عنوان مثال ، بلوغ ممکن است اهمیت بیشتری از منحنی یادگیری داشته باشد . این تفاوت را می توان با حساب های وزن نتایج و رتبه بندی معیارهای مختلف با عنوان " ضروری" با وزن 3 ، " مطلوب" با وزن 2 ، و " اختیاری " با وزن 1 در نظرگرفت. ماتریس مقایسه مورد استفاده در یک مطالعه موردی در جدول 3-5 نشان داده شده است . این مثال به عنوان یک توصیه واقعی به استفاده از روش های خاص در نظر گرفته نشده ، شما میتوانید معیار مقایسه و رتبه بندی خود را توسعه دهید.

توصیه هایی برای اولویت بندی نیازمندیها :

اولویت بندی نیازهای امنیتی یک فعالیت مهم است. ما توصیه می کنیم که سهامداران روش های اولویت بندی نامزد را انتخاب کنند، توسعه معیارهای انتخاب برای انتخاب یکی از روش های پیاده سازی نیازمندی های امنیتی. در طی فرایند اولویت بندی، ذینفعان می توانید کنترل کنید که هر کس درک و آگاهی در مورد شرایط امنیتی بدست آورند و بیشتر به بررسی هر شرایط مبهم بردازند و سپس به اتفاق نظر برسند ، نتایج حاصل از این اولویت بندی قابل اطمینان تر خواهد بود