مهندسی نرم‌افزار امن

با رویکرد علمی و کاربردی

مؤلف:

دکتر محمدعلی ترکمانی

سرشناسه : ترکمانی، محمدعلی، ۱۳۵۴ -
عنوان و نام پدیدآور : مهندسی نرم‌افزار امن با رویکرد علمی و کاربردی/ مولف محمدعلی ترکمانی.
مشخصات نشر : مشهد: ارسطو،۱۳۹۵.
مشخصات ظاهری : ۲۴۲ ص.: مصور،جدول،نمودار.
شابک : 978-600-432-064-1
وضعیت فهرست نویسی : فیپا
موضوع : کامپیوترها -- ایمنی اطلاعات
موضوع : Computer security
موضوع : نرم‌افزار -- مهندسی
موضوع : Software engineering
رده بندی کنگره : ۱۳۹۵ ۴۲ ت ۹ ک / ۹/ ۷۶ QA
رده بندی دیویی : ۸/۰۰۵
شماره کتابشناسی ملی : ۴۳۷۰۵۷۸

نام کتاب : مهندسی نرم‌افزار امن با رویکرد علمی و کاربردی

مؤلف : دکتر محمدعلی ترکمانی

ناشر : ارسطو (با همکاری سامانه اطلاع‌رسانی چاپ و نشر ایران)

صفحه‌آرایی ، تنظیم و طرح جلد : محمدعلی ترکمانی و علی بیات

تیراژ : ۱۰۰۰ جلد

نوبت چاپ : چهارم– ۱۳۹۸

چاپ : مدیران

قیمت : ۶۰۰۰۰ تومان

شابک : 978-600-432-064-1

تلفن‌های مرکز پخش : ۰۹۱۷۷۱۶۴۹۴۰ – ۵۰۹۶۱۴۶ - ۰۵۱۱

فهرست مطالب

مقدمه:

امروزه نرم‌افزار به جزئی جدایی‌ناپذیر از زندگی بشر تبدیل‌شده است. نرم‌افزارها همه‌جا هستند و تقریباً زندگی بدون نرم‌افزار غیرممکن است. لذا امنیت و قابلیت اطمینان نرم‌افزار بسیار مهم و در مواردی امری بحرانی است. امنیت برنامه‌های کاربردی و امنیت شبکه‌های رایانه‌ای به‌تنهایی نمی‌توانند امنیت موردنیاز کاربران را برآورده کنند.

امنیت برنامه‌های کاربردی یعنی حفاظت از نرم‌افزار و سیستمی که نرم‌افزار روی آن اجرا می‌گردد، بعد از ساخته‌شدن آن! موضوعات بحرانی مرتبط با امنیت برنامه‌های کاربردی عبارتند از:

- ایزوله کردن کد در جعبه شنی یا Sandboxing (مثل ماشین‌مجازی جاوا)
- حفاظت در قبال کد مخرب
- قفل‌کردن برنامه‌های اجرایی
- رصد برنامه‌ها به هنگام اجرا (مخصوصاً ورودی برنامه‌ها)
- اعمال سیاست‌های استفاده از نرم‌افزار
- سیستم‌های قابل توسعه

این نوع امنیت می‌تواند تا حدودی مشکل را برطرف کند. به قول پزشکان تسکین‌بخش است اما درمان قطعی نیست. وقتی نرم‌افزار از درون مشکل داشته باشد، هیچ راهکاری به‌جز مسدود کردن و عدم استفاده از نرم‌افزار وجود ندارد. راه‌حل اصلی این مشکل مهندسی نرم‌افزار امن است که به عقیده کارشناسان از امنیت شبکه و امنیت برنامه‌های کاربردی مهم‌تر است. مهندسی نرم‌افزار امن به این موضوع می‌پردازد که از فاز آغازین تا فاز انتهایی پروژه مسئله امنیت را در نظر گرفته و برای دستیابی به آن برنامه‌ریزی کرده و ریسک‌های نرم‌افزار را مدیریت نماییم. همچنین با استفاده از ابزارهای مناسب تست‌های امنیتی مختلف را روی سیستم انجام می‌دهیم. همچنین کدهای نوشته‌شده را ازنظر امنیتی مرور می‌کنیم تا برنامه فاقد آسیب‌پذیری باشد. به‌این‌ترتیب نرم‌افزاری امن و قابل‌اطمینان تولید خواهد شد. با توجه به اهمیت این موضوع، دروس توسعه امن نرم‌افزار و مهندسی نرم‌افزار امن در سرفصل‌های مصوب وزارت علوم، تحقیقات و فناوری قرارگرفته است. علیرغم نیاز دانشجویان و اساتید به مرجعی مناسب برای این درس، تاکنون هیچ کتابی در این زمینه به چاپ نرسیده است. لذا تصمیم گرفتم این کتاب را تألیف نمایم تا بتواند برای

دانشجویان این درس مفید باشد و به افزایش کیفیت آموزشی این درس کمک نماید. در این کتاب سعی شده است که مطالب با زبانی ساده ارائه گردد. همچنین در پایان هر فصل نیز تعدادی سؤال تشریحی و چهارگزینه‌ای به همراه کلید سؤالات آورده شده است تا دانشجویان گرامی بهتر بتوانند خود را برای آزمون‌هایی که در پیش رودارند آماده نمایند. برای بخش عملی درس نیز مدلسازی UML و UMLSec با نرم‌افزار ویژوال پارادایم و مرور کد در Visual C# آموزش داده‌شده است.امید است این اثر موردتوجه همکاران و دانشجویان گرامی قرار گیرد. از اساتید و دانشجویان عزیز تقاضا دارم نقطه نظرات خود را از طریق ایمیل m.a.torkamani@gmail.com با مؤلف در میان بگذارند تا انشالله در ویرایش‌های بعدی اشکالات یا کاستی‌های احتمالی کتاب مورد تجدید نظر قرار گیرد. در پایان وظیفه خود می‌دانم از آقای مهندس علی بیات به خاطر طراحی جلد و همچنین از مدیریت انتشارات ارسطو و سامانه اطلاع رسانی چاپ و نشر ایران جناب آقای حسین قنبری به خاطر مساعدت در کار چاپ تشکر و قدردانی نمایم.

محمدعلی ترکمانی

تابستان ۱۳۹۵

فصل اول

انگیزه

۱-۱-مقدمه

امروزه نرم‌افزار همه‌جا هست و زندگی بدون نرم‌افزار تقریباً غیرممکن است. نرم افزار اتومبیل شمارا می‌راند. گوشی تلفن همراه شمارا کنترل می‌کند. چگونگی دستیابی شما به سرویس‌های مالی بانک شماست؛ چگونه برق، آب و گاز طبیعی را دریافت می‌نمایید؛ و چگونه از یک کرانه به کرانه‌ای دیگر پرواز می‌کنید. چه بدانیم یا ندانیم، همه‌ی ما بر سیستم‌های اطلاعاتی پیچیده، به هم مرتبط، و نرم‌افزاری متکی هستیم که از اینترنت به‌عنوان ابزاری برای مخابره و انتقال اطلاعات استفاده می‌کنند.

ساخت، استقرار، راه‌اندازی و استفاده از نرم‌افزاری که بر اساس و با ذهنیت امنیت تولید نشود، می‌تواند ریسکی بزرگ باشد. امنیت و قابلیت اطمینان نرم‌افزار بسیار مهم و در مواردی امری بحرانی است.

۱-۲- مسئله امنیت

به لحاظ تاریخی مسئله امنیت، مسئله تازه‌ای نیست. دفاع از قلعه‌ها و شهرها نمونه‌ای از مسئله امنیت درگذشته است. رویکرد دفاع پیرامونی (Defending perimeters) و دفاع در مرزهای سیستم (دیوار آتش) بازمانده مسائل امنیت فیزیکی است. بنابراین حمله‌کنندگان نیز از همان رویه‌های فریب دفاع پیرامونی تقلید می‌کنند. دفاع پیرامونی به تنهایی کافی نیست زیرا حتی نمی‌توانیم به درون سیستم‌ها اعتماد کنیم. آسیب پذیریهای موجود در نرم افزارها میتواند توسط نفوذگران مورد سوء استفاده قرار گیرد و در این رابطه فایروال هیچ کمکی به ما نخواهد کرد.

نرم‌افزارهای مبتنی بر وب اهداف معمول می‌باشند که به‌راحتی قابل بهره‌برداری هستند. حفره‌های امنیتی در نرم‌افزار وجود دارد و نرخ رشد آسیب پذیریهای نرم افزار رو به افزایش است. بنابراین تنها هزینه کردن برای تأمین امنیت شبکه مشکل امنیت را حل نمی‌کند بلکه باید نرم‌افزارهای بهتری ساخته‌شود. یک جنبه اصلی و بحرانی از امنیت کامپیوتر(Computer Security)، امنیت نرم‌افزار است.

برای امنیت نرم‌افزار چه‌کاری باید انجام دهیم؟ ایده مهندسی نرم‌افزار برای حل این مشکل این است که چنانچه نرم‌افزار تحت حملات مخرب قرار گرفت، بازهم بتواند به کار خود ادامه دهد.

واضح است که برای طراحی و ساخت یک نرم‌افزار غیرقابل تخریب باید هزینه کنیم.

۳-۱- مهندسی نرم افزار امن

مهندسی نرم افزار امن یعنی:

- توجه به امنیت نرم افزار در تمامی فازهای توسعه نرم افزار.
- درک ریسک‌های نرم‌افزاری و مدیریت آن‌ها
- تحلیل، طراحی، ساخت و تست نرم‌افزار برای ایمنی.

مهندسی نرم افزار امن تلاش می‌کند نرم‌افزاری بسازد که بتواند با دفاع پیش از حمله به بقای خود ادامه دهد.

قلب امنیت رایانه ، شناسایی و حذف مشکلات نرم‌افزار است.

برخی از رهیافت‌های امنیتی خوب به شرح ذیل است:

- تفکر به امنیت در تمامی مراحل چرخه حیات نرم‌افزار خصوصاً به امنیت در ابتدای چرخه حیات نرم‌افزار
- طراحی، ساخت و آزمون نرم‌افزار برای امنیت
- درک مشکلات عمومی(معایب مربوط به زبان برنامه‌نویسی)
- همه فرآورده‌های نرم‌افزاری را با تحلیل ریسک و تست عمقی محک بزنیم(تحلیل ریسک و آزمون همه مصنوعات نرم‌افزاری)
- آموزش درباره دخیل کردن امنیت (توسعه‌دهندگان، طراحان و کاربران)

۴-۱-امنیت برنامه‌های کاربردی در مقایسه امنیت نرم‌افزار

امنیت برنامه‌های کاربردی یعنی حفاظت از نرم‌افزار و سیستمی که نرم‌افزار روی آن اجرا می‌گردد، بعد از ساخته شدن آن! امنیت برنامه‌های کاربردی بیشتر به دنبال مانیتورینگ برنامه‌ها و جلوگیری از تخطی آن‌ها از وظایف و مسئولیت‌هایشان است. اما هدف از مهندسی نرم افزار امن، درنظر گرفتن امنیت در تمامی چرخه توسعه نرم افزار و تولید یک نرم افزار امن میباشد. بنابراین:

- امنیت نرم‌افزار، امنیت برنامه کاربردی نیست.

Software Security # Application Security

- امنیت نرم‌افزار، نرم‌افزار امنیتی نیست.

Software Security # Security Software

موضوعات بحرانی مرتبط با امنیت کاربرد عبارتند از:

- ایزوله کردن کد در جعبه شنی یا Sandboxing (مثل ماشین‌مجازی جاوا)
- حفاظت در قبال کد مخرب
- قفل‌کردن برنامه‌های اجرایی
- رصد برنامه‌ها به هنگام اجرا (مخصوصاً ورودی برنامه‌ها)
- اعمال سیاست‌های استفاده از نرم‌افزار
- سیستم‌های قابل توسعه

در امنیت برنامه کاربردی، همان رویکرد شبکه گرا به امنیت را داریم. استفاده از رویکردهای استانداردی مثل نفوذ و وصله، پالایش ورودی و غیره وجود دارد. به‌طورکلی، رویکردی منفعلانه و واکنشی است که بر اساس کشف و رفع مشکلات امنیتی شناخته‌شده بعد از سوءاستفاده (exploit) در سیستم‌های میدانی(عملیاتی) است. مثلاً کشف سرریز بافر از طریق کنترل ترافیک http روی پورت ۸۰. بنابراین تمرکز روی کاربرد، ندیدن تصویر بزرگ‌تر یعنی امنیت نرم‌افزار است.

بنابراین امنیت نرم‌افزار بر امنیت برنامه کاربردی برتری دارد. متأسفانه توسعه اکثر سیستم‌ها بدون حضور افراد امنیتی انجام می‌شود.

۱-۴-۱-ابزارهای آزمون امنیت برنامه کاربردی

محصولات تست امنیتی برنامه کاربردی راه‌حلی برای نرم‌افزار غیر ایمن است. با استفاده از این ابزارها یافتن و از بین بردن خطاها در زمان اجرای کد انجام می‌شود. مشکلاتی که در این رهیافت وجود دارد عبارتند از:

- ابزارهای آزمون محصول به‌عنوان راه‌کار تشخیص نرم‌افزار ناامن استفاده می‌شوند اما برای برطرف کردن آن کمکی نمی‌نماید.
- فاقد اثربخشی برای کشف و تشخیص مسئله، ناکارآمد برای برطرف کردن آن
- برخورد جعبه سیاهی و خیلی ساده با برنامه‌های کاربردی
- یافتن و حذف اشکالات بعد از توسعه کد
- محکی برای میزان " بد نبودن"badness-ometers!"
- نقطه‌ضعف اصلی این‌گونه ابزارها :: تمرکز روی ترافیک پورت ۸۰
- عدم توجه به ورودی‌های دیگر کاربردهای جدیدتر مثل SSL، متغیرهای محیطی، کتابخانه‌های خارجی، اجزای توزیع‌شده و ....
- عدم توجه به مسائل ورای ورودی‌ها نظیر درستی معماری، امنیت داده‌ها، کنترل دسترسی و ...

تنها کاربرد مناسب این ابزارها، آزمون محصولات تجاری موجود است، ولی حل مشکلات کشف‌شده نیاز به توسعه نرم‌افزار بهتر دارد!

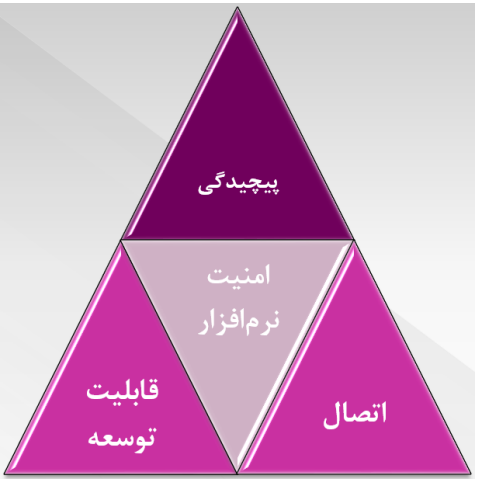
رفتار حمله‌کننده به‌صورت زیر است :

- تجزیه برنامه
- فهم نحوه کارکرد هر بخش
- انجام رفتاری که از کاربران عادی انتظار نمی‌رود
- سوء رفتار برنامه

با توجه به موارد فوق، به نظر شما، مؤثرترین روش برای حفاظت نرم‌افزار کدام است؟ امنیت نرم‌افزار یا برنامه کاربردی؟

۱-۵- مشکلات یا ابعاد سه‌گانه مسئله امنیت نرم‌افزار

سؤال این است که چرا امنیت نرم‌افزار در حال حاضر یک مشکل بزرگ است. مطابق شکل ۱-۱ سه دلیل برای این سؤالات وجود دارد:

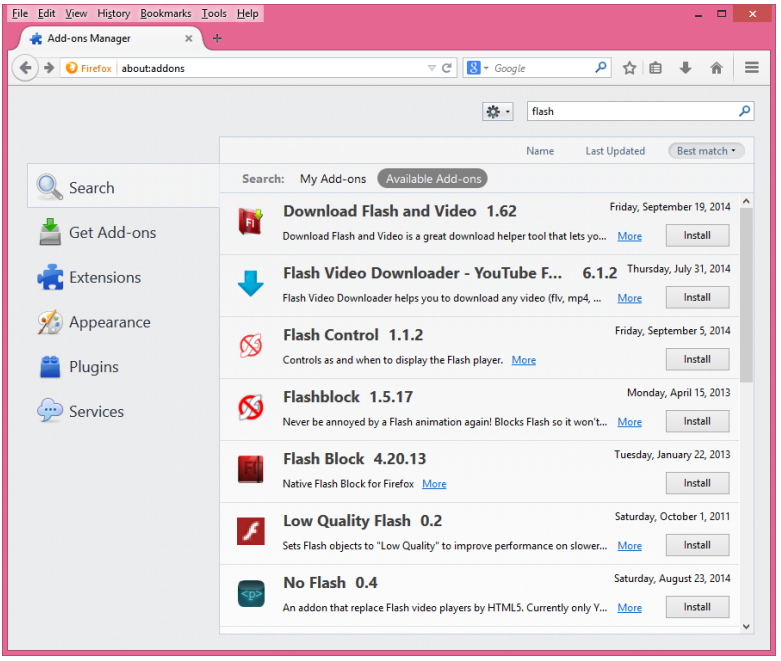


شکل ۱-۱: ابعاد سه‌گانه مسئله امنیت نرم‌افزار

۱) اتصال: رشد ارتباطات شبکه‌ای در نرم‌افزارها باعث افزایش تعداد حملات و سهولت انجام آن‌ها شده است.

- کارهایی که قبلاً offline انجام می‌شد، هم‌اکنون online انجام می‌شود.
- سیستم‌ها نسبت به حملات نرم‌افزاری از منابع راه دور آسیب‌پذیر شده‌اند.
- در حال حاضر حملات اتوماتیک آسان است.
- سیستم‌های نرم‌افزاری بیشتری موجودند که مورد تهاجم قرار گیرند.
- با شبکه‌ای شدن نرم‌افزارها، مهاجمان همین الان در یک‌قدمی شما هستند.
- سازمان‌های بزرگ درگیر دو مشکل هستند: سرویس‌های وب، معماری سرویس گرا(SOA). برنامه‌های کاربردی موروثی که هرگز طبیعت درون‌شبکه‌ای نداشتند، شبکه‌ای شده‌اند و به‌عنوان سرویس انتشاریافته‌اند.

۲) قابلیت توسعه: قابلیت توسعه یعنی درجه قابلیت گسترش سیستم‌ها. سیستم‌هایی که قابل بروز رسانی یا قابل توسعه‌اند، کدهای قابل‌حمل را فراخوانی می‌کنند تا عملکرد سیستم را ارتقاء دهند. به‌عنوان نمونه ساختار Plug-in در مرورگرهای وب(شکل ۱-۲) به‌منظور باز کردن انواع جدید فایل‌ها تعبیه شده است.



شکل ۱-۲: Plug-in در مرورگر FireFox

سیستمهای عامل از درایور دستگاه‌ها[1] که به‌صورت DLL ارائه می‌شوند، پشتیبانی می‌کنند. قابلیت توسعه برنامه‌های کاربردی از طریق اسکریپت نویسی، مؤلفه‌ها و اپلت‌ها که ازنقطه‌نظر اقتصادی جذاب به نظر می‌رسد، انجام می‌شود. متأسفانه این کار، پیشگیری از آسیب‌پذیری‌های نرم‌افزار و جلوگیری از ورود به سیستم به شکل اضافات نرم‌افزاری ناخواسته را دشوار می‌کند.

۳) پیچیدگی: منظور از پیچیدگی رشد بدون کنترل سیستم‌های نرم‌افزاری جدید از بعد اندازه و پیچیدگی است. برخی از مشکلات مربوط ب این حوزه عبارتند از:

1 device drivers

- وابستگی ما به عملکرد مناسب Kernel! Kernel به‌تنهایی شامل ۱۰ میلیون خط کد است.
- زمانی که سیستم‌ها خیلی بزرگ می‌شوند، خطاها اجتناب‌ناپذیرند. هرچقدر نرم‌افزار بزرگتر باشد، تأمین امنیت آن مشکل‌تر است.
- ویندوز XP ۴۰ میلیون خط کد دارد.
- رشد نرخ نقص[1] به نسبت مربع تعداد کدهای برنامه بالا می‌رود.
- استفاده از زبان‌های برنامه‌نویسی غیر ایمن پیچیدگی را افزایش می‌دهد.
- در تئوری ما می‌توانیم برنامه‌های کوچک را تحلیل و تثبیت نماییم، اما در عمل و برای برنامه‌های بزرگ چطور؟
- تعداد خطوط کد درحال‌توسعه روزانه در حال افزایش می‌باشند.
- تعداد خطوط بیشتر یعنی خطاهای[2] زیادتر.
- گاهی پایگاه کد[3] رشد می‌کند درحالی‌که به نظر می‌رسد حجم آن کم است!
- استفاده از سکوهای .NET و J2EE به نظر کاهش اندازه کد پایه! در عمل افزایش آن!
- پیچیدگی بیشتر درنتیجه استفاده از سیستم‌های مدیریت شناسه(ID)، XML، سرورهای برنامه کاربردی، پایگاه‌های داده.

۶-۱-حل مسئله امنیت نرم‌افزار

امنیت نرم‌افزار فعالیتی در حال پیشرفت است که نیازمند تغییرات فرهنگی می‌باشد. برخی از راه‌حل‌های موجود برای امنیت نرم افزار عبارتند از:

- چرخه توسعه امن McGraw 2004
- چرخه توسعه امن مایکروسافت

1 defect rate
2 Bugs
3 code base

- چارچوب پیشنهادی OWASP
- استانداردها و گواهی‌نامه‌های ناظر به توسعه امن

۷-۱-مشکلات امنیتی در نرم افزار

۱-۷-۱- نقص(Defect)

آسیب‌پذیری‌های مربوط به دو فاز طراحی و پیاده‌سازی ، هردو جزء نقص‌ها است. نقص‌ها ممکن است برای سال‌ها در نرم‌افزار غیرفعال باشد و تنها در یک سیستم اطلاعاتی با نتایج مهم بروز کند. نقص ها دو نوع هستند: اشکال یا خطا(Bug) و عیب(Flaw).

۱-۱-۷-۱-اشکال یا خطا(Bug)

Bug یک مشکل نرم‌افزاری در سطح پیاده‌سازی است. Bug ها ممکن است در کد موجود باشد اما هیچ‌وقت اجرا نشوند. Bug ها به‌آسانی قابل‌کشف و تصحیح می‌باشند.
ابزارهای مؤثری برای شناسایی برخی از انواع Bug (مثل سرریز، شرایط رقابتی فرایندها یا race Conditions) وجود دارد. تقریباً ۴۵% مسائل امنیت نرم‌افزار که به CERT گزارش می‌شود، ریشه در سرریز بافر دارد. به عنوان مثال در زبان C بسیار راحت است که جند بایت را تعریف کنید و سپس مقدار بیشتری از آن را استفاده کنید. زبان C این موضوع را کنترل نمی‌کند.

```
char x[12];
x[12] = '\0';
```

دو مجرای اصلی سرریز بافر عبارتند از اختصاص بافر در پشته و تخصیص بافر در heap. سرریزبافر استکی بسیار معمول است. استفاده از gets برای دریافت ورودی نامحدود :

```
void main() {
  char buf[1024];
  gets(buf);
}
```

۲-۱-۷-۱-عیب(Flaw)

Flaw یک مشکل در سطح عمیق‌تر است. یک عیب قطعاً در کد نرم‌افزار وجود دارد و قابل‌مشاهده است ولی می‌تواند مربوط به زمان طراحی هم باشد (یا نباشد). تکنیکی خودکار برای

کشف عیوب در سطح طراحی وجود ندارد. تحلیل ریسک به‌صورت دستی ممکن است نقاط ضعف را شناسایی کند. در عمل نسبت اشکالات به عیوب ۵۰/۵۰ است.

حذف خطاها از راه مرور کد تنها حدود نیمی از مشکل را حل می‌کند.

مایکروسافت گزارش داد بیش از ۵۰% مسائلی که شرکت در حین فشارهای امنیتی خود کشف کرده، ریشه در معماری نرم‌افزار دارند.

عموماً مهاجمین به این موضوع که آسیب‌پذیری به علت یک نقص (flaw) یا اشکال (bug) رخ‌داده است، توجهی ندارند، اگرچه به‌سادگی می‌توان از اشکالات، بهره‌برداری (سوءاستفاده) نمود. آسیب‌پذیری‌های موجود در سطح طراحی سخت‌ترین نوع آسیب‌پذیری هستند که باید به آن‌ها رسیدگی شود. جدول ۱-۱ نمونه‌هایی از عیب و اشکال را نشان می‌دهد.

جدول ۱-۱: نمونه‌هایی از عیب و اشکال را نشان می‌دهد.

BUGS	**FLAWS**
Buffer overflow: stack smashing	Method over-riding problems (subclass issues)
Buffer overflow: one-stage attacks	Compartmentalization problems in design
Buffer overflow: string format attacks	Privileged block protection failure (`DoPrivilege()`)
Race conditions: TOCTOU	Error-handling problems (fails open)
Unsafe environment variables	Type safety confusion error
Unsafe system calls (`fork()`, `exec()`, `system()`)	Insecure audit log design
Incorrect input validation (black list vs. white list)	Broken or illogical access control (role-based access control [RBAC] over tiers)
	Signing too much code

۲-۷-۱-ریسک

عیب‌ها و اشکالات می‌توانند منجر به ریسک گردند. ریسک‌ها باعث شکست یا توقف کار[1] نمی‌شوند. ریسک احتمالی را که یک اشکال یا عیب می‌تواند بر هدف نرم‌افزار (به صورت منفی) اثرگذار باشد در نظر می‌گیرد.

risk = probability × impact

probability احتمال و impact تأثیر است.

۳-۷-۱-محدوده نواقص(defects)

تعیین اشکال(bug) یا عیب (flaw)بودنِ یک نقص (defect) نرم‌افزاری مشکل است.

انواع نقص‌های نرم‌افزاری شامل موارد ذیل هستند:

- خطا(error)های پیاده‌سازی محلی مثل استفاده از gets
- آسیب‌پذیری‌های متوسط شامل تعامل بیش از یک ناحیه از کد
- خطاهای واسط میان رویه‌ای مثل شرایط رقابت(race Condition)
- اشتباهات سطح بالاتر زمان طراحی مثل اداره خطا(error-handling)
  - نیاز به تخصص بیشتر برای کشف
  - سخت برای کشف، سخت‌تر برای کشف به‌صورت خودکار

محدوده نواقص به موارد ذیل بستگی دارد:

- میزان کدی که باید در نظر گرفته شود تا آسیب‌پذیری معلوم گردد
- چه میزان جزئیات محیط اجرا را باید بدانیم تا آسیب‌پذیری‌ها را درک کنیم؟
- آیا یک توصیف در سطح طراحی بهترین روش برای تعیین وجود یک آسیب‌پذیری است؟
- آسیب‌پذیری‌های میانی شامل فعل و انفعالات متقابل بین چند موقعیت از کد است.
- آسیب‌پذیری‌های سطح طراحی: نیاز به تجربه زیادی دارند. این‌گونه آسیب‌پذیری‌ها نه‌تنها تشخیص آن‌ها مشکل است بلکه مخصوصاً یافتن اتوماتیک آنها دشوار است.

---

1 failures

۸-۱-سؤالات تشریحی

۱-تفاوت امنیت نرم‌افزار با امنیت شبکه چیست؟

۲-با ترسیم شکل ابعاد سه‌گانه امنیت در نرم‌افزار را توضیح دهید.

۳-اصطلاحات زیر را توضیح دهید.

الف-نقص ب-اشکال ج- عیب

۴- چهار نمونه از نقصهای موجود در نرم افزار را نام ببرید.

۵-تفاوت امنیت نرم افزار و امنیت برنامه کاربردی چیست؟