



مهندسی نرم افزار امن

با رویکرد علمی و کاربردی

مؤلف:

دکتر محمدعلی ترکمانی

سرشناسه	: ترکمانی، محمدعلی، ۱۳۵۴ -
عنوان و نام پدیدآور	: مهندسی نرم افزار 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	: <b>Computer security</b>
موضوع	: نرم افزار -- مهندسی
موضوع	: <b>Software engineering</b>
رده بندی کنگره	: ۱۳۹۵ ت ۴۲ ک ۹ / ۹ / ۷۶ QA
رده بندی دیویی	: ۸/۰۰۵
شماره کتابشناسی ملی	: ۴۳۷۰۵۷۸

نام کتاب : مهندسی نرم افزار امن با رویکرد علمی و کاربردی

مؤلف : دکتر محمدعلی ترکمانی

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

صفحه آرای، 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

تلفن های مرکز پخش : ۰۹۱۷۷۱۶۴۹۴۰ - ۵۰۹۶۱۴۶ - ۰۵۱۱

این اثر مشمول قانون حمایت از مولفان و مصنفان و هنرمندان است. هر کس تمام یا قسمتی از این اثر را بدون اجازه مولف نشر یا پخش یا عرضه کند، مورد پیگرد قانونی قرار خواهد گرفت.

## فهرست مطالب

### فصل اول: انگیزه ..... ۱۵

۱-۱- مقدمه ..... ۱۵

۱-۲- مسئله امنیت ..... ۱۵

۱-۳- مهندسی نرم افزار امن ..... ۱۶

۱-۴- امنیت برنامه‌های کاربردی در مقایسه امنیت نرم افزار ..... ۱۷

۱-۴-۱- ابزارهای آزمون امنیت برنامه کاربردی ..... ۱۸

۱-۵- مشکلات یا ابعاد سه‌گانه مسئله امنیت نرم افزار ..... ۱۹

۱-۶- حل مسئله امنیت نرم افزار ..... ۲۱

۱-۷- مشکلات امنیتی در نرم افزار ..... ۲۲

۱-۷-۱- نقص (Defect) ..... ۲۲

۱-۷-۱-۱- اشکال یا خطا (Bug) ..... ۲۲

۱-۷-۱-۲- عیب (Flaw) ..... ۲۲

۱-۷-۲- ریسک ..... ۲۴

۱-۷-۳- محدوده نواقص (defects) ..... ۲۴

۱-۸- سؤالات تشریحی ..... ۲۵

### فصل دوم: چرخه \_\_\_\_\_ MCGRAW برای توسعه نرم افزار

### امنیت ..... ۲۷

۲-۱- چرخه MCGRAW برای حل مشکل امنیت ..... ۲۷

۲-۲- رویکرد سه ستونه در امنیت نرم افزار ..... ۲۸

- ۲۹-۲-۲-۱-ستون اول: مدیریت ریسک..... ۲۹
- ۲۹-۱-۱-۱-مراحل RMF..... ۲۹
- ۳۲-۲-۲-۲-ستون دوم- نقاط تماس امنیت نرم افزار..... ۳۲
- ۳۴-۲-۲-۱-مرور کد..... ۳۴
- ۳۴-۲-۲-۲-تحلیل ریسک معماری..... ۳۴
- ۳۵-۲-۲-۳-تست نفوذ..... ۳۵
- ۳۵-۲-۲-۴-تست امنیتی مبتنی بر ریسک(ریسک مبنا)..... ۳۵
- ۳۶-۲-۲-۴-۱-تفاوت تست امنیتی با تضمین کیفیت (QA)..... ۳۶
- ۳۶-۲-۲-۵-موارد سوءاستفاده یا سناریوهای سوء کاربرد..... ۳۶
- ۳۶-۲-۲-۶-نیازمندی های امنیتی..... ۳۶
- ۳۷-۲-۲-۷-عملیات امنیتی(اقدامات اپراتوری امنیت)..... ۳۷
- ۳۷-۲-۲-۸-تحلیل خارجی..... ۳۷
- ۳۷-۲-۲-۹-رویکرد دو کلاهی..... ۳۷
- ۳۹-۲-۲-۱۰-هرچه سریع تر، بهتر..... ۳۹
- ۳۹-۲-۲-۱۱-ایجاد یک گروه امنیت نرم افزار..... ۳۹
- ۴۰-۲-۲-۳-ستون سوم: دانش..... ۴۰
- ۴۳-۲-۳-مرور کد به وسیله نرم افزار..... ۴۳
- ۴۴-۲-۳-۱-ریشه مشکلات چیست؟..... ۴۴
- ۴۴-۲-۳-۲-تحلیل ایستای کد ( Static analysis )..... ۴۴
- ۴۵-۲-۳-۳-روش های تحلیل کد..... ۴۵
- ۴۵-۲-۳-۴-تحلیل کد باینری..... ۴۵
- ۴۶-۲-۳-۵-هدف؛ خوب بودن است و نه کامل بودن!!..... ۴۶
- ۴۷-۲-۳-۶-مشکل قواعد..... ۴۷
- ۴۷-۲-۳-۷-رهیافت های با رویکردهای تحلیل استاتیک..... ۴۷
- ۴۸-۲-۳-۸-تولیدکننده های عمده در بازار مرور کد..... ۴۸
- ۵۰-۲-۳-۹-مشخصات کلیدی یک ابزار..... ۵۰

- ۱۰-۳-۲- ویژگی‌هایی که یک ابزار نباید داشته باشد..... ۵۰
- ۴-۲- تحلیل ریسک‌های معماری..... ۵۱
- ۱-۴-۲- theme های رایج در تجزیه و تحلیل ریسک‌های امنیتی..... ۵۱
- ۲-۴-۲- فعالیت‌های اصلی در نمونه‌سازی تحلیل ریسک..... ۵۲
- ۳-۴-۲- رویکرد سنتی تحلیل ریسک..... ۵۳
- ۱-۴-۳-۲- یک مجموعه از تعاریف..... ۵۴
- ۲-۴-۳-۲- نیازمندی دانش ( Knowledge Requirement )..... ۵۵
- ۳-۴-۳-۲- دید در سطح جنگل ( Forest – level view )..... ۵۵
- ۴-۴-۲- روش سنتی محاسبه ریسک..... ۵۶
- ۴-۴-۲- اولویت‌بندی ریسک‌ها بر اساس احتمال وقوع و شدت خطر..... ۵۶
- ۲-۴-۴-۲- روش بیان ریسک به صورت زیان مالی..... ۵۶
- ۵-۴-۲- یک واقعیت در مورد ROI..... ۵۷
- ۶-۴-۲- محدودیت‌های رهیافت‌های سنتی تحلیل ریسک..... ۵۷
- ۷-۴-۲- تحلیل ریسک مدرن..... ۵۸
- ۸-۴-۲- ریسک در سطح نیازمندی‌ها..... ۵۹
- ۹-۴-۲- آنچه باید انجام گیرد..... ۵۹
- ۱۰-۴-۲- فرایند نقطه تماس: ریسک معماری..... ۶۰
- ۱-۴-۱۰-۱- تحلیل مقاومت در برابر حمله..... ۶۲
- ۲-۴-۱۰-۲- تحلیل ابهامات..... ۶۳
- ۳-۴-۱۰-۳- تحلیل ضعف‌ها..... ۶۴
- ۵-۲- تست نفوذ چیست؟..... ۶۶
- ۱-۵-۲- تفاوت تست نفوذ با تست کارکردی برنامه..... ۶۷
- ۲-۵-۲- آسیب‌پذیری‌های هدف..... ۶۷
- ۳-۵-۲- تست نفوذ نیازمند هر دو کلاه..... ۶۷

- ۶۷ .....وضعیت فعلی ۲-۵-۴
- ۶۸ .....مراحل تست نفوذ بر اساس استاندارد PTES ۲-۵-۵
- ۷۲ .....مرور ابزارهای تست نفوذ ۲-۵-۶
- ۷۲ .....۱-۵-۶-۲-فواید استفاده از ابزار ۲-۵-۶-۱
- ۷۳ .....۲-۵-۶-۲-ابزارهای تست برنامه تحت وب ۲-۵-۶-۲
- ۷۳ .....۳-۵-۶-۲-ابزارهای تست نفوذ نرم افزار ۲-۵-۶-۳
- ۷۳ .....۱-۵-۶-۳-۲-مکانیزم تزریق خطا ۲-۵-۶-۳-۱
- ۷۴ .....Holodeck ۲-۵-۶-۴
- ۷۴ .....۵-۵-۶-۲-مکانیزم fuzzing ۲-۵-۶-۵
- ۷۵ .....۱-۵-۶-۵-۲-دسته بندی کلی fuzzerها ۲-۵-۶-۵-۱
- ۷۵ .....۲-۵-۶-۵-۲-ابزارهای fuzzing ۲-۵-۶-۵-۲
- ۷۶ .....۲-۵-۷-محدودیت های تست نفوذ ۲-۵-۷
- ۷۶ .....۲-۵-۸-جمع بندی تست نفوذ ۲-۵-۸
- ۷۷ .....۲-۶-تست امنیتی ریسک مبنا ۲-۶
- ۷۷ .....۱-۶-۲-تفاوت تست امنیت و تست نفوذ ۲-۶-۱
- ۷۸ .....۲-۶-۲-از خارج به داخل و از داخل به خارج ۲-۶-۲
- ۷۹ .....۳-۶-۲-تفاوت تست ریسک مبنا با تست های معمول نرم افزار ۲-۶-۳
- ۷۹ .....۴-۶-۲-مدیریت ریسک و تست امنیتی ۲-۶-۴
- ۸۰ .....۵-۶-۲-چگونگی رهیافت تست امنیتی ۲-۶-۵
- ۸۲ .....۶-۶-۲-تفکر در مورد ورودی بدخواهانه ۲-۶-۶
- ۸۲ .....۷-۶-۲-غلبه بر ورودی ۲-۶-۷
- ۸۳ .....۸-۶-۲-توجه به زمان در دو بعد ۲-۶-۸
- ۸۳ .....۷-۲-نمودارهای سوء کاربرد ۲-۷
- ۸۳ .....۱-۷-۲-مدل موارد کاربری ۲-۷-۱

- ۲-۷-۲- موارد سوءاستفاده ( Abuse Cases ) ..... ۸۶
- ۲-۷-۳- امنیت مجموعه‌ای از ویژگی‌ها نیست ..... ۸۷
- ۲-۷-۴- چه کاری می‌توان انجام داد؟ ..... ۸۷
- ۲-۷-۵- نسبت این مرحله با مراحل قبل ..... ۸۸
- ۲-۷-۶- وقتی شما در جای حمله‌کننده قرار می‌گیرید ..... ۸۸
- ۲-۷-۷- وقتی ما در جای خودمان قرار می‌گیریم ..... ۸۹
- ۲-۷-۸- ایجاد و توسعه موارد سوءاستفاده مفید ..... ۸۹
- ۲-۷-۹- فرایند توسعه موارد سوءاستفاده ..... ۹۰
- ۲-۷-۱۰- ترسیم Abuse case diagram ..... ۹۱
- ۲-۸- عملیات امنیت ..... ۹۴
- ۲-۸-۱- جایگاه متخصصان امنیت در فازهای مختلف توسعه نرم‌افزار ..... ۹۴
- ۲-۸-۱-۱- نیازمندی‌ها : موارد سوء کاربرد ..... ۹۵
- ۲-۸-۱-۲- طراحی: تحلیل ریسک‌های کسب و کار ..... ۹۵
- ۲-۸-۱-۳- طراحی: تحلیل ریسک‌های معماری ..... ۹۵
- ۲-۸-۱-۴- طراحی تست: تست امنیت ..... ۹۶
- ۲-۸-۱-۵- پیاده‌سازی: مرور کد ..... ۹۶
- ۲-۸-۱-۶- تست سیستم: تست نفوذ ..... ۹۶
- ۲-۸-۱-۷- سیستم میدانی : استقرار و عملیات ..... ۹۷
- ۲-۹- سؤالات تشریحی ..... ۹۷

## فصل سوم: چرخه توسعه امنیت (SDL) مایکروسافت ..... ۱۰۱

- ۳-۱- مقدمه ..... ۱۰۱
- ۳-۲- اصول راهنما برای SD3+C ..... ۱۰۲
- ۳-۳- اصول راهنما برای PD3+C ..... ۱۰۵
- ۳-۴- مرور چرخه مایکروسافت ..... ۱۰۶

- ۱۰۸..... ۳-۵-مقایسه SDL با چرخه مک گرا؟
- ۱۰۹ ..... ۳-۶- برخی از ابزارهای ارائه شده SDL
- ۱۰۹..... ۳-۶-۱- ابزار Attack Surface Analyzer
- ۱۰۹ ..... ۳-۶-۲- ابزار SDL Threat Modeling Tool
- ۱۱۰..... ۳-۶-۳- ابزار فاز تست پایه فایل MiniFuzz
- ۱۱۰..... ۳-۶-۴- ابزار فاز تست Regex
- ۱۱۲ ..... ۳-۷-سؤالات تشریحی

## ۱۱۳..... فصل چهارم: UML SEC

- ۱۱۳..... ۴-۱-مقدمه
- ۱۱۳ ..... ۴-۲- چرا مدل سازی UMLSEC؟
- ۱۱۴..... ۴-۲-۱- اهداف UMLSec
- ۱۱۵ ..... ۴-۲-۲- مزایای UMLSec
- ۱۱۵ ..... ۴-۲-۳- ویژگی های UMLSec
- ۱۱۶..... ۴-۲-۴- نمودارهای UMLSec و UML
- ۱۱۷ ..... ۴-۲-۵- نیازمندی های امنیتی در سیستم توزیعی
- ۱۱۸..... ۴-۳- مدل سازی رمزنگاری در UML
- ۱۱۹..... ۴-۴-چند مثال از مدلسازی با UML
- ۱۱۹ ..... ۴-۴-۱- نمودار موارد کاربری
- ۱۱۹ ..... ۴-۴-۲- نمودار کلاس
- ۱۱۹ ..... ۴-۴-۳- نمودار حالت
- ۱۲۰ ..... ۴-۴-۴- نمودار ترتیبی
- ۱۲۰ ..... ۴-۴-۵- نمودار فعالیت



- ۱۲۰ ..... ۴-۴-۶- نمودار استقرار
- ۱۲۱ ..... ۴-۴-۷- زیر سیستم‌ها
- ۱۲۲ ..... ۴-۵- مکانیزم‌های بسط UML
- ۱۲۳ ..... ۴-۶- ارزیابی امنیت نمودارهای UML
- ۱۲۴ ..... ۴-۷- بسط UMLSEC
- ۱۲۴ ..... ۴-۷-۱- کلیشه‌ها، مقادیر برچسب دار و محدودیت‌ها
- ۱۲۸ ..... ۴-۷-۲- مدل دشمن یا Adversary model
- ۱۲۹ ..... ۴-۷-۳- کلیشه issuer, POS device, smart card, wire, LAN, encrypted, Internet, node
- ۱۳۰ ..... ۴-۷-۴- کلیشه **guarded**
- ۱۳۰ ..... ۴-۷-۵- کلیشه «secure links» یا پیوندهای امن
- ۱۳۱ ..... ۴-۷-۶- کلیشه‌های «**high**»، «**integrity**»، «**secercy**»
- ۱۳۲ ..... ۴-۷-۷- کلیشه وابستگی امن یا «secure dependency»
- ۱۳۳ ..... ۴-۷-۸- کلیشه «Critical» یا بحرانی
- ۱۳۴ ..... ۴-۷-۹- کلیشه «no up-flow» و «no down-flow»
- ۱۳۵ ..... ۴-۷-۱۰- کلیشه «data security» یا امنیت داده
- ۱۳۸ ..... ۴-۷-۱۱- کلیشه «fair Exchange» یا تبادل منصفانه
- ۱۴۱ ..... ۴-۷-۱۲- کلیشه rbac
- ۱۴۲ ..... ۴-۸- نیازمندی‌های بسط UML برای ایجاد و توسعه سیستم‌های بحرانی امنیتی
- ۱۴۳ ..... ۴-۹- سؤالات تشریحی
- فصل پنجم: رهیافت STRIDE مایکروسافت برای مدل سازی تهدیدها... ۱۴۵**
- ۱۴۵ ..... ۵-۱- چرا مدل سازی تهدیدها؟

- ۲-۵-رسم نمودار ..... ۱۴۵
- ۳-۵-اجزای نمودار و مثال‌هایی از آن ..... ۱۴۶
- ۱-۳-۵-مرزهای اعتماد ..... ۱۴۷
- ۲-۳-۵-لایه‌های دیاگرام ..... ۱۴۷
- ۳-۳-۵-قوانین سرانگشتی برای اعتبارسنجی نمودار ..... ۱۴۹
- ۴-۵-مشخص کردن تهدیدها: چرخه STRIDE برای مدل‌سازی تهدید ..... ۱۵۰
- ۵-۵-اعمال تهدیدهای STRIDE روی هر المان ..... ۱۵۳
- ۶-۵-رسیدگی ..... ۱۵۴
- ۷-۵-مطالعه موردی: نرم‌افزار PET SHOP 4.0 EXTERNAL DEPENDENCIES ..... ۱۵۵
- ۱-۷-۵-ویژگی‌های سیستم ..... ۱۵۵
- ۲-۷-۵-ایجاد DFD ها ..... ۱۵۷
- ۳-۷-۵-تعیین تهدیدهای سیستم ..... ۱۵۹
- ۴-۷-۵-برنامه رسیدگی ..... ۱۶۳
- ۸-۵-معرفی ابزار مدل‌سازی تهدید ..... ۱۶۳
- ۹-۵-سؤالات تشریحی ..... ۱۶۵

## فصل ششم: مدلسازی UML و UMLSEC با ویژوال پارادایم ..... ۱۶۷

- ۱-۶-معرفی ویژوال پارادایم ..... ۱۶۷
- ۲-۶-نصب نرم‌افزار ..... ۱۶۷
- ۳-۶-نمای کلی رابط کاربر ..... ۱۶۸
- ۴-۶-باز کردن / بستن قاب‌ها ..... ۱۶۹
- ۵-۶-ایجاد پروژه جدید ..... ۱۶۹
- ۶-۶-ذخیره‌سازی پروژه ..... ۱۷۲

- ۱۷۲ ..... ۶-۷-تنظیمات فونت و رنگ
- ۱۷۳..... ۶-۸-تنظیم جعبه‌ابزار DIAGRAMS
- ۱۷۵ ..... ۶-۹-سوئیچ کردن بین دیاگرام‌های مختلف یک پروژه
- ۱۷۸ ..... ۶-۱۰-رسم نمودار موارد کاربری
- ۱۷۸ ..... ۶-۱۰-۱-روش اول
- ۱۸۱ ..... ۶-۱۰-۲-روش دوم
- ۱۸۲ ..... ۶-۱۱-ایجاد نمودار کلاس
- ۱۸۳ ..... ۶-۱۱-۱-نحوه ساخت کلاس
- ۱۸۳ ..... ۶-۱۱-۲-ساخت رابطه همکاری
- ۱۸۵ ..... ۶-۱۱-۳-تجمیع
- ۱۸۵ ..... ۶-۱۱-۴-تعیین نوع رابطه بین کلاس‌ها
- ۱۸۷ ..... ۶-۱۱-۵-ایجاد رابطه وراثت
- ۱۸۸ ..... ۶-۱۱-۶-رابطه وابستگی بین دو کلاس
- ۱۸۸ ..... ۶-۱۱-۷-ایجاد صفت کلاس
- ۱۸۹ ..... ۶-۱۱-۸-ایجاد عملکردهای کلاس
- ۱۹۰..... ۶-۱۱-۹-مرتب‌سازی اعضای کلاس
- ۱۹۱ ..... ۶-۱۲-ترسیم نمودار فعالیت
- ۱۹۱ ..... ۶-۱۲-۱-ایجاد Swimlane
- ۱۹۲..... ۶-۱۲-۲-الحاق پارتیشن به Swimlane
- ۱۹۳ ..... ۶-۱۲-۳-ایجاد گره‌های اولیه
- ۱۹۳ ..... ۶-۱۲-۴-ایجاد عملیات (کنش)
- ۱۹۶ ..... ۶-۱۲-۵-تفاوت Activity و Action

۱۹۶	..... ۱۳-۶-توسعه نمودار توالی
۱۹۹	..... ۱۴-۶-ایجاد نمودار بسته
۱۹۹	..... ۱-۱۴-۶-تعیین کلیشه
۲۰۱	..... ۱۵-۶-ایجاد نمودار مؤلفه
۲۰۳	..... ۱-۱۵-۶-تعیین کلیشه برای مؤلفه
۲۰۴	..... ۲-۱۵-۶-تعریف واسطها
۲۰۴	..... ۱-۲-۱۵-۶-تعریف واسط ارائه
۲۰۵	..... ۲-۲-۱۵-۶-تعریف واسط موردنیاز
۲۰۵	..... ۳-۱۵-۶-ایجاد وابستگی بین مؤلفهها
۲۰۶	..... ۱۶-۶-ایجاد نمودار استقرار
۲۰۶	..... ۱-۱۶-۶-ایجاد گرهها و ترسیم ارتباطات بین گرهها
۲۰۸	..... ۲-۱۶-۶-ایجاد نمودار حالت
۲۰۹	..... ۱-۲-۱۶-۶-ایجاد حالتها و انتقالها
۲۱۱	..... ۲-۲-۱۶-۶-اضافه کردن ناحیه به حالت
۲۱۲	..... ۳-۲-۱۶-۶-مدلسازی خصوصیات انتقال
۲۱۴	..... ۳-۱۶-۶-نمایش محتوای حالت در نمودار حالت
۲۱۸	..... ۱۷-۶-ترسیم نمودارهای UMLSEC در ویژوال پارادایم
۲۴۰	..... پروژه دروس مهندسی نرم افزار امن و امنیت بانکهای اطلاعاتی
۲۴۱	..... منابع:

## مقدمه:

امروزه نرم‌افزار به جزئی جدایی‌ناپذیر از زندگی بشر تبدیل شده 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 ساخته شدن آن! موضوعات بحرانی مرتبط با امنیت برنامه‌های کاربردی عبارتند از:

- ایزوله کردن کد در جعبه شنی یا Sandboxing (مثل ماشین مجازی جاوا)
- حفاظت در قبال کد مخرب
- قفل کردن 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 از نظر امنیتی مرور می‌کنیم تا برنامه فاقد آسیب‌پذیری باشد. به‌این ترتیب نرم‌افزاری امن و قابل اطمینان تولید خواهد شد. با توجه به اهمیت این موضوع، درس توسعه امن نرم‌افزار و مهندسی نرم‌افزار امن در سرفصل‌های مصوب وزارت علوم، تحقیقات و فناوری قرار گرفته است. علیرغم نیاز دانشجویان و اساتید به مرجعی مناسب برای این درس، تاکنون هیچ کتابی در این زمینه به چاپ نرسیده است. لذا تصمیم گرفتیم 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 داده شده است. امید است این اثر مورد توجه 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](mailto:m.a.torkamani@gmail.com) با مؤلف در میان بگذارند تا انشاءالله 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 نرم افزار همه جا هست و زندگی بدون نرم افزار تقریباً غیرممکن است. نرم افزار اتومبیل شمارا می راند. گوشی تلفن همراه شمارا کنترل می کند. چگونگی دستیابی شما به سرویس های مالی بانک شماست؛ چگونه برق، آب و گاز طبیعی را دریافت می نمایید؛ و چگونه از یک کرانه به کرانه ای دیگر پرواز می کنید. چه بدانیم یا ندانیم، همه ی ما بر سیستم های اطلاعاتی پیچیده، به هم مرتبط، و نرم افزاری متکی هستیم که از اینترنت به عنوان ابزاری برای مخابره و انتقال اطلاعات استفاده می کنند.

ساخت، استقرار، راه اندازی و استفاده از نرم افزاری که بر اساس و با ذهنیت امنیت تولید نشود، می تواند ریسکی بزرگ باشد. امنیت و قابلیت اطمینان نرم افزار بسیار مهم و در مواردی امری بحرانی است.

### ۲-۱- مسئله امنیت

به لحاظ تاریخی مسئله امنیت، مسئله تازه ای نیست. دفاع از قلعه ها و شهرها نمونه ای از مسئله امنیت در گذشته است. رویکرد دفاع پیرامونی (Defending perimeters) و دفاع در مرزهای سیستم (دیوار آتش) بازمانده مسائل امنیت فیزیکی است. بنابراین حمله کنندگان نیز از همان رویه های فریب دفاع پیرامونی تقلید می کنند. دفاع پیرامونی به تنهایی کافی نیست زیرا حتی نمی توانیم به درون سیستم ها 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.

نرم افزارهای مبتنی بر وب اهداف معمول می باشند که به راحتی قابل بهره برداری هستند. حفره های امنیتی در نرم افزار وجود دارد و نرخ رشد آسیب پذیریهایی نرم افزار رو به افزایش است. بنابراین تنها هزینه کردن برای تأمین امنیت شبکه مشکل امنیت را حل نمی کند بلکه باید نرم افزارهای بهتری ساخته شود. یک جنبه اصلی و بحرانی از امنیت کامپیوتر (Computer Security)، امنیت نرم افزار است.

برای امنیت نرم افزار چه کاری باید انجام دهیم؟ ایده مهندسی نرم افزار برای حل این مشکل این است که چنانچه نرم افزار تحت حملات مخرب قرار گرفت، بازهم بتواند به کار خود ادامه دهد. واضح است که برای طراحی و ساخت یک نرم افزار 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 ریسک های نرم افزاری و مدیریت آنها
  - تحلیل، طراحی، ساخت و تست نرم افزار برای ایمنی.
- مهندسی نرم افزار امن تلاش می کند نرم افزاری بسازد که بتواند با دفاع پیش از حمله به بقای خود ادامه دهد.
- قلب امنیت رایانه، شناسایی و حذف مشکلات نرم افزار است.
- برخی از رهیافت های امنیتی خوب به شرح ذیل است:
- تفکر به امنیت در تمامی مراحل چرخه حیات نرم افزار خصوصاً به امنیت در ابتدای چرخه حیات نرم افزار
  - طراحی، ساخت و آزمون نرم افزار برای امنیت
  - درک مشکلات عمومی (معایب مربوط به زبان برنامه نویسی)
  - همه فرآورده های نرم افزاری را با تحلیل ریسک و تست عمقی محک بزنییم (تحلیل ریسک و آزمون همه مصنوعات نرم افزاری)
  - آموزش درباره دخیل کردن امنیت (توسعه دهندگان، 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 نرم‌افزار امن، در نظر گرفتن امنیت در تمامی چرخه توسعه نرم‌افزار و تولید یک نرم‌افزار 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 ماشین مجازی جاوا)
- حفاظت در قبال کد مخرب
- قفل کردن برنامه‌های اجرایی
- رصد برنامه‌ها به هنگام اجرا (مخصوصاً ورودی برنامه‌ها)
- اعمال سیاست‌های استفاده از نرم‌افزار
- سیستم‌های قابل توسعه

در امنیت برنامه کاربردی، همان رویکرد شبکه‌گرا به امنیت را داریم. استفاده از رویکردهای استاندارد مثل نفوذ و وصله، پالایش ورودی و غیره وجود دارد. به‌طور کلی، 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 بودن” “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؟

## ۵-۱- 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:

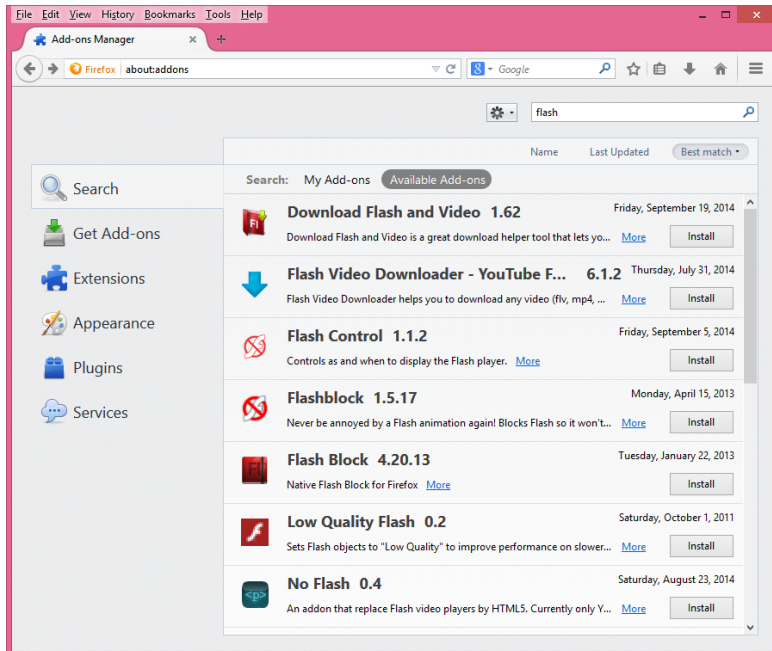


شکل ۱-۱: ابعاد سه‌گانه مسئله امنیت نرم‌افزار

(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 انتشار یافته‌اند.

۲) **قابلیت توسعه:** قابلیت توسعه یعنی درجه قابلیت گسترش سیستم‌ها. سیستم‌هایی که قابل بروز رسانی یا قابل توسعه‌اند، کدهای قابل حمل 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 که به‌صورت DLL ارائه می‌شوند، پشتیبانی می‌کنند. قابلیت توسعه برنامه‌های کاربردی از طریق اسکریپت نویسی، مؤلفه‌ها و اپلت‌ها که از نقطه‌نظر 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 به این حوزه عبارتند از:

- وابستگی ما به عملکرد مناسب Kernel! Kernel به تنهایی 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 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 در حال توسعه روزانه در حال افزایش می‌باشند.
- تعداد خطوط بیشتر یعنی خطاهای زیادتر.
- گاهی پایگاه کد رشد می‌کند در حالی که به نظر می‌رسد حجم آن کم است!
- استفاده از سکوی NET و J2EE به نظر کاهش اندازه کد پایه! در عمل افزایش آن!
- پیچیدگی بیشتر در نتیجه 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 نرم‌افزار عبارتند از:

- چرخه توسعه امن McGraw 2004
- چرخه توسعه امن میکروسافت

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

## ۷-۱-مشکلات امنیتی در نرم افزار

### ۷-۱-۱- نقص (Defect)

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

#### ۷-۱-۱-۱- اشکال یا خطا (Bug)

Bug یک مشکل نرم‌افزاری در سطح پیاده‌سازی است. Bug‌ها ممکن است در کد موجود باشد اما هیچ‌وقت اجرا نشوند. Bug‌ها به‌آسانی قابل کشف 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 چند 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. سرریز بافر استکی بسیار معمول است. استفاده از gets برای دریافت ورودی نامحدود:

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

#### ۷-۱-۱-۲- عیب (Flaw)

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

کشف عیوب در سطح طراحی وجود ندارد. تحلیل ریسک به صورت 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) رخ داده است، توجهی ندارند، اگرچه به سادگی 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<sup>۱</sup> نمی‌شوند. ریسک احتمالی را که یک اشکال یا عیب می‌تواند بر هدف نرم افزار ( به صورت منفی) اثرگذار باشد در نظر می‌گیرد.

$$\text{risk} = \text{probability} \times \text{impact}$$

probability احتمال و impact تأثیر است.

## ۳-۷-۱-محدوده نواقص (defects)

تعیین اشکال (bug) یا عیب (flaw) بودن یک نقص (defect) نرم افزاری مشکل است. انواع نقص‌های نرم افزاری 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.

---

<sup>1</sup> failures



## ۸-۱- سوالات تشریحی

- ۱- تفاوت امنیت نرم افزار با امنیت شبکه چیست؟
- ۲- با ترسیم شکل ابعاد سه گانه امنیت در نرم افزار 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؟