

فهرست مطالب

۱ تست نرم افزار

۱-۱ خلاصه فصل.....	۲۴
۱-۲ تست نرم افزار چیست.....	۲۴
۳-۱ وابستگی تست و کیفیت.....	۲۶
۴-۱ دلایل اهمیت تست نرم افزار و پیامدهای عدم وجود آن.....	۲۷
۵-۱ چه موقع باید عمل تست انجام شود.....	۲۸
۶-۱ تست نرم افزار در تئوری و عمل.....	۲۸
۱-۶-۱ تئوری تست.....	۲۸
۲-۶-۱ تست در عمل.....	۲۹
۷-۱ تفاوت تست و اشکالزدایی.....	۲۹
۸-۱ مفاهیم بنیادی در تست نرم افزار.....	۳۰
۱-۸-۱ واریسی.....	۳۰
۱-۸-۲ اعتبارسنجی.....	۳۰
۱-۸-۳ تفاوت واریسی و اعتبار سنجی.....	۳۱
۱-۸-۴ ارزیابی.....	۳۱
۱-۸-۴-۱ قابلیت استفاده.....	۳۱
۲-۸-۴-۱ قابلیت تحمل.....	۳۱
۳-۸-۴-۱ قابلیت نگهداری.....	۳۱
۹-۱ چرخه حیات تست نرم افزار.....	۳۱
۱-۹-۱ فاز ۱: تحلیل نیازمندی ها.....	۳۲
۱-۹-۲ فاز ۲: برنامه ریزی تست.....	۳۳
۱-۹-۳ فاز ۳: توسعه نمونه تست.....	۳۳
۱-۹-۴ فاز ۴: آماده سازی محیط تست.....	۳۴

۳۴.....	۵-۹-۱ فاز ۵: اجرای تست
۳۵.....	۶-۹-۱ فاز ۶: پایان یافتن چرخه آزمون
۳۶.....	۱-۱۰ زمان شروع تست
۳۶.....	۱-۱۱ زمان پایان تست
۳۷.....	۱-۱۲ اصول تست نرم افزار
۳۸.....	۱-۱۳ روش های تست نرم افزار
۳۸.....	۱-۱۳-۱ تست ایستا
۳۸.....	۲-۱۳-۱ تست پویا
۳۸.....	۱-۱۳-۲-۱ تست تابعی
۳۸.....	۲-۱۳-۲-۲ تست منطقی
۳۹.....	۳-۱۳-۱ مقایسه تست ایستا و پویا
۳۹.....	۱-۱۴ استراتژی های تست
۳۹.....	۱-۱۴-۱ استراتژی تست منطق
۳۹.....	۱-۱۴-۱-۱ تست جعبه سیاه
۴۰.....	۱-۱۴-۱-۱-۱ افزایش هم ارزی
۴۰.....	۲-۱۴-۱-۱-۲ تحلیل مقدار مرزی
۴۰.....	۳-۱۴-۱-۱-۳ حدس خطا
۴۰.....	۴-۱۴-۱-۱-۴ نمودار علت و معلول
۴۱.....	۲-۱۴-۱-۲ تست جعبه سفید
۴۱.....	۱-۱۴-۱-۲-۱ تست منطقی
۴۱.....	۲-۱۴-۱-۲-۲ تست های اثبات ریاضی
۴۱.....	۳-۱۴-۱-۲-۳ تست های اتاق تمیز
۴۱.....	۳-۱۴-۱-۳ تست جعبه خاکستری
۴۲.....	۴-۱۴-۱-۴ مقایسه رویکرد تست جعبه سفید، خاکستری و سیاه
۴۳.....	۲-۱۴-۱ استراتژی چگونگی اجرای تست

۴۳	 تست بالا به پایین. ۱-۱۴-۲-۱
۴۳	 تست پایین با بالا. ۱-۱۴-۲-۲
۴۳	 سطوح تست. ۱۵-۱
۴۴	 تست واحد. ۱-۱۵-۱
۴۴	 تست یکپارچگی. ۱-۱۵-۲
۴۴	 تست رابط مؤلفه. ۱-۱۵-۳
۴۵	 تست سیستم. ۱-۱۵-۴
۴۵	 تست پذیرش. ۱-۱۵-۵
۴۶	 انواع تست. ۱۶-۱
۴۶	 تست نصب و راه اندازی. ۱-۱۶-۱
۴۶	 تست سازگاری. ۱-۱۶-۲
۴۶	 تست دود. ۱-۱۶-۳
۴۶	 تست رگرسیون. ۱-۱۶-۴
۴۶	 تست آلفا. ۱-۱۶-۵
۴۷	 تست بتا. ۱-۱۶-۶
۴۷	 تست عملکرد. ۱-۱۶-۷
۴۷	 تست غیر عملکرد. ۱-۱۶-۸
۴۷	 تست مخرب. ۱-۱۶-۹
۴۷	 تست کارایی نرم افزار. ۱-۱۶-۱۰
۴۸	 تست قابلیت استفاده. ۱-۱۶-۱۱
۴۸	 تست امنیت. ۱-۱۶-۱۲
۴۸	 متریک های تست. ۱۷-۱
۴۸	 متریک های نمونه تست. ۱-۱۷-۱
۴۸	 متریک های پوشش. ۱-۱۷-۲
۴۸	 متریک های خرابی. ۱-۱۷-۳

۴۸.....	۱-۱۷-۳-۱ متوسط زمان برای خطا.....
۴۹.....	۱-۱۷-۳-۲ قابلیت اطمینان.....
۴۹.....	۱-۱۷-۳-۳ در دسترس بودن.....
۴۹.....	۱-۱۷-۴ متریک های قابلیت استفاده.....
۴۹.....	۱-۱۷-۴-۱ چگالی مشکلات قابلیت استفاده.....
۴۹.....	۱-۱۷-۴-۲ آسانی استفاده.....
۵۰.....	۱-۱۷-۴-۳ متریک سرعت کار.....
۵۰.....	۱-۱۷-۴-۴ بهره وری.....
۵۰.....	۱۸-۱ گواهینامه تست نرم افزار.....
۵۰.....	۱-۱۸-۱ انواع گواهینامه های تست نرم افزار.....
۵۰.....	۱-۱۸-۲ معرفی برخی از گواهینامه های تست.....
۵۱.....	۱۹-۱ استانداردهای تست نرم افزار.....
۵۱.....	۱-۱۹-۱ استاندارد ISO/IEC ۹۱۲۶.....
۵۲.....	۱-۱۹-۲ استاندارد ISO/IEC ۹۲۴۱-۱۱.....
۵۲.....	۱-۱۹-۳ استاندارد ISO/IEC ۲۵۰۰:۲۰۰۵.....
۵۲.....	۱-۱۹-۴ استاندارد ISO/IEC ۱۲۱۱۹.....
۵۳.....	۱-۱۹-۵ دیگر استانداردهای تست نرم افزار.....
۵۳.....	۲۰-۱ نتیجه گیری.....
۵۴.....	۲۱-۱ سوالات متداول.....
۵۴.....	۲۲-۱ منابعی برای مطالعات بیشتر.....

۲ تست امنیت نرم افزار

۵۶.....	۱-۱ خلاصه فصل.....
۵۶.....	۲-۱ امنیت نرم افزار چیست؟.....
۵۶.....	۳-۱ مهمترین عوامل برقراری امنیت.....
۵۷.....	۱-۳-۱ محرمانگی.....

۵۷.....	۱-۳-۲ یکپارچگی.....
۵۷.....	۱-۳-۳ دسترس پذیری.....
۵۸.....	۴-۱ معرفی ۲۵ مورد از خطرناک ترین خطاهای نرم افزاری.....
۶۰.....	۱-۴-۱ نگاه طبقه بندی شده به خطاها.....
۶۱.....	۱-۴-۱-۱ تعامل نا امن بین مؤلفه ها.....
۶۱.....	۱-۴-۱-۲ مدیریت پر مخاطره منابع.....
۶۲.....	۱-۴-۱-۳ دفاع نفوذ پذیر.....
۶۲.....	۱-۴-۲ تسکین دهنده و راه حل های عمومی جهت کاهش اثر خطاها.....
۶۲.....	۱-۴-۲-۱ ایجاد و حفظ کنترل بر روی همه ورودی ها - M۱.....
۶۳.....	۱-۴-۲-۲ ایجاد و حفظ کنترل بر همه خروجی ها - M۲.....
۶۳.....	۱-۴-۲-۳ قفل کردن محیط - M۳.....
۶۳.....	۱-۴-۲-۴ فرض اینکه مؤلفه های خارجی می توانند.....
۶۴.....	تخریب و کدهای شما توسط هرکسی خوانده شوند - M۴
۶۴.....	۱-۴-۲-۵ استفاده از ویژگی های امنیتی پذیرفته شده.....
	در صنعت به جای نوع آوری - M۵
۶۴.....	۱-۴-۲-۶ استفاده از کتابخانه ها و چارچوب هایی.....
	که اجتناب از ضعف های معرفی شده را آسان می سازند - GP۱
۶۵.....	۱-۴-۲-۷ یکپارچه کردن امنیت به سرتاسر.....
	چرخه حیات توسعه نرم افزار - GP۲
۶۵.....	۱-۴-۲-۸ استفاده از ترکیب گسترده ای.....
	از روش ها برای پیدا کردن و جلوگیری ضعف ه به طور جامع - GP۳
۶۵.....	۱-۴-۲-۹ اجازه به کلاینت های قفل شده برای تعامل با نرم افزار - GP۴.....
۶۶.....	۱-۴-۳ نگاشت تسکین دهنده ها به ۲۵ خطای مهم نرم افزاری.....
۶۷.....	۱-۴-۴ مقایسه ۲۵ خطای CWE با ده آسیب پذیری مهم OWASP.....
۶۸.....	۵-۱ تست امنیت نرم افزار چیست؟.....
۶۸.....	۶-۱ دلایل تست امنیت نرم افزار.....
۶۹.....	۷-۱ تست امنیت در مقابل تست متعارف نرم افزار.....
۶۹.....	۸-۱ چه زمانی تست امنیت نرم افزار استفاده می شود.....
۶۹.....	۹-۱ برنامه های نیازمند به تست امنیت.....

۶۹.....	۱۰-۱ اصول و قواعد انجام تست امنیت.....
۷۲.....	۱۱-۱ تکنیک های تست امنیت.....
۷۲.....	۱-۱۱-۱ بازبینی و بازرسی دستی.....
۷۳.....	۱-۱۱-۲ مدل سازی تهدید.....
۷۳.....	۱-۱۱-۳ بازبینی کد.....
۷۳.....	۱-۱۱-۴ تست نفوذ.....
۷۴.....	۱-۱۱-۵ مقایسه ۴ تکنیک تست امنیت.....
۷۵.....	۱۲-۱ روش های اصلی تست امنیت.....
۷۵.....	۱-۱۲-۱ تست امنیت رسمی.....
۷۵.....	۱-۱۲-۲ تست امنیت مبتنی بر مدل.....
۷۶.....	۱۳-۱ تست امنیت مبتنی بر تزریق خطا.....
۷۶.....	۱-۱۳-۱ تست فازی.....
۷۶.....	۱-۱۳-۲ تست بررسی آسیب پذیری.....
۷۷.....	۱-۱۳-۳ تست مبتنی بر صفت خاص.....
۷۷.....	۱-۱۳-۴ تست امنیت مبتنی بر جعبه سفید.....
۷۷.....	۱-۱۳-۵ تست امنیت مبتنی بر ریسک.....
۷۷.....	۱۴-۱ رویکرد تست امنیت.....
۷۹.....	۱۵-۱ تست امنیت در بحث سیستم مدیریت امنیت اطلاعات.....
۷۹.....	۱۶-۱ استاندارد ISO/IEC ۲۷۰۳۴ - راهنمای امنیت برنامه کاربردی.....
۸۰.....	۱-۱۶-۱ هدف.....
۸۰.....	۱-۱۶-۲ فرآیندها.....
۸۱.....	۱-۱۶-۳ کنترل امنیت برنامه کاربردی.....
۸۱.....	۱۷-۱ استاندارد ISO/IEC ۱۵۴۰۸ - معیارهای ارزیابی برای امنیت محصولات IT.....
۸۱.....	۱۸-۱ نتیجه گیری.....
۸۲.....	۱۹-۱ سوالات متداول.....

۲۰-۱ منابعی برای مطالعات بیشتر..... ۸۳

۳ استاندارد تست نرم افزار: OWASP

۱-۱ خلاصه فصل..... ۸۶

۲-۱ آشنایی با پروژه امنیت نرم افزار تحت وب باز..... ۸۶

۳-۱ مخاطرات امنیتی برنامه های کاربردی..... ۸۷

۱-۳-۱ رتبه بندی مخاطرات بر اساس روش OWASP..... ۸۷

۱-۳-۲ بررسی ده مخاطره مهم وب سایت ها در سال ۲۰۱۳ و راهکار آن ها از نگاه OWASP..... ۸۷

۱-۳-۲-۱ تزریق (A۱)..... ۸۸

۱-۳-۲-۲ احراز هویت و مدیریت نشست نقض شده (A۲)..... ۸۹

۱-۳-۲-۳ نفوذ به سایت ها از طریق اسکریپ نویسی (A۳)..... ۹۰

۱-۳-۲-۴ ارجاعات شیء مستقیم نا امن (A۴)..... ۹۱

۱-۳-۲-۵ پیکربندی نادرست امنیت (A۵)..... ۹۲

۱-۳-۲-۶ در معرض قرار گرفتن اطلاعات حساس (A۶)..... ۹۴

۱-۳-۲-۷ از دست رفتن کنترل دسترسی سطح عملکرد (A۷)..... ۹۵

۱-۳-۲-۸ جعل درخواست خارج از وب سایت (A۸)..... ۹۷

۱-۳-۲-۹ استفاده از مؤلفه ها با آسیب پذیری های شناخته شده (A۹)..... ۹۸

۱-۳-۲-۱۰ تغییرمسیردهی و انتقال های نامعتبر (A۱۰)..... ۹۹

۴-۱ پروژه فرآیند تضمین امنیت نرم افزار OWASP..... ۱۰۰

۱-۴-۱ تضمین نرم افزار واقعا چیست؟..... ۱۰۰

۱-۴-۲ معرفی فرآیندهای تضمین امنیت نرم افزار..... ۱۰۰

۱-۴-۲-۱ فرآیند های یکپارچه کردن امنیت نرم افزار در SDLC..... ۱۰۱

۱-۴-۲-۲ فرآیند شناسایی نیازمندی های امنیت..... ۱۰۲

۱-۴-۲-۳ فرآیند بازبینی امنیت معماری و طراحی..... ۱۰۲

۱-۴-۲-۴ فرآیند بازبینی کد امن..... ۱۰۳

۱-۴-۲-۵ فرآیند تست امنیت..... ۱۰۳

- ۱-۴-۲-۶ فرآیند بازبینی امنیت استقرار..... ۱۰۳
- ۱-۴-۲-۷ فرآیند بازبینی امنیت انتشار..... ۱۰۳
- ۵-۱ نتیجه گیری..... ۱۰۳
- ۶-۱ سوالات متداول..... ۱۰۴
- ۷-۱ منابعی برای مطالعات بیشتر..... ۱۰۴

۴ روش تست نرم افزار: ASVS

- ۱-۱ خلاصه فصل..... ۱۰۶
- ۲-۱ نحوه استفاده از استاندارد ASVS..... ۱۰۶
- ۳-۱ چگونگی رویکرد ASVS..... ۱۰۸
- ۴-۱ به کارگیری ASVS در چرخه حیات توسعه نرم افزار..... ۱۰۹
- ۵-۱ بررسی سطوح چهارگانه ASVS..... ۱۰۹
- ۱-۵-۱ سطح ۱ - واریسی اتوماتیک..... ۱۱۰
- ۱-۵-۱-۱ سطح ۱A - اسکن پویا (واریسی اتوماتیک جزئی)..... ۱۱۰
- ۱-۵-۱-۲ سطح ۱B - اسکن کد منبع (واریسی اتوماتیک جزئی)..... ۱۱۱
- ۱-۵-۲ سطح ۲ - واریسی دستی..... ۱۱۱
- ۱-۵-۲-۱ سطح ۲A - تست امنیت (واریسی دستی جزئی)..... ۱۱۲
- ۱-۵-۲-۲ سطح ۲B واریسی کد (واریسی دستی جزئی)..... ۱۱۲
- ۱-۵-۳ سطح ۳ - واریسی طرح..... ۱۱۲
- ۱-۵-۴ سطح ۴ - واریسی داخلی..... ۱۱۲
- ۱-۵-۵ حداقل نیازهای سطح بالا برای سطوح مختلف ASVS..... ۱۱۳
- ۱-۵-۶ نیازهای تفضیلی واریسی در ASVS..... ۱۱۴
- ۱-۵-۶-۱ واریسی ۱. نیازهای مستندسازی معماری امنیتی..... ۱۱۴
- ۱-۵-۶-۲ واریسی ۲. نیازمندی های واریسی احراز هویت..... ۱۱۵
- ۱-۵-۶-۳ واریسی ۳. نیازمندی های واریسی مدیری نشست..... ۱۱۷
- ۱-۵-۶-۴ واریسی ۴. نیازمندی های واریسی مکانیزم دستیابی..... ۱۱۸

- ۵-۶-۱ واری ۵. نیازمندی های واری صحت ورودی.....۱۲۰
- ۶-۶-۱ واری ۶. نیازمندی های واری کدگذاری/ اسکپ خروجی.....۱۲۱
- ۷-۶-۱ واری ۷. نیازمندی های واری رمزنگاری.....۱۲۲
- ۸-۶-۱ واری ۸. نیازمندی های واری مدیریت یا کنترل خطا و گزارش گیری.....۱۲۴
- ۹-۶-۱ واری ۹. نیازمندی های واری حفاظت داده.....۱۲۶
- ۱۰-۶-۱ واری ۱۰. نیازمندی واری امنیت اطلاعات.....۱۲۶
- ۱۱-۶-۱ واری ۱۱. نیازمندی های واری امنیت HTTP.....۱۲۸
- ۱۲-۶-۱ واری ۱۲. نیازمندی های واری تنظیم امنیت.....۱۲۹
- ۱۳-۶-۱ واری ۱۳. نیازمندی های جستجوی کد مخرب.....۱۳۰
- ۱۴-۶-۱ واری ۱۴. نیازمندی های واری امنیت داخلی.....۱۳۰
- ۶-۱ نتیجه گیری.....۱۳۱
- ۷-۱ سوالات متداول.....۱۳۱
- ۸-۱ منابعی برای مطالعات بیشتر.....۱۳۱

۵ تست برنامه‌های کاربردی وب بر اساس OWASP

- ۱-۱ خلاصه فصل.....۱۳۴
- ۲-۱ راه اندازی چرخه حیات توسعه نرم افزار با فعالیت های واری ASVS.....۱۳۴
- ۱-۲-۱ گام اول.....۱۳۴
- ۱-۲-۲ گام دوم.....۱۳۵
- ۱-۲-۳ گام سوم.....۱۳۵
- ۱-۲-۴ گام چهارم.....۱۳۵
- ۳-۱ چارچوب تست بر اساس OWASP.....۱۳۶
- ۱-۳-۱ فاز اول - پیش از شروع توسعه.....۱۳۶
- ۱-۳-۱-۱ فاز ۱A: بازیابی سیاست ها و استانداردها.....۱۳۷
- ۱-۳-۱-۲ فاز ۱B: توسعه اندازه گیری و معیار متریک ها (تضمین قابلیت ردیابی).....۱۳۷
- ۱-۳-۲ فاز دوم- تست در زمانتعریف و طراحی.....۱۳۷

۱۳۸.....	۱-۳-۳ فاز سوم- تست در طی توسعه.
۱۳۸.....	۱-۳-۳-۱ فاز ۳A: جلسه بررسی کد.
۱۳۸.....	۱-۳-۳-۲ فاز ۳B: بازبینی های کد.
۱۳۸.....	۱-۳-۴ فاز چهارم - تست در طی مستقر کردن.
۱۳۸.....	۱-۳-۴-۱ فاز ۴A: تست نفوذ برنامه کاربردی.
۱۳۹.....	۱-۳-۴-۲ فاز ۴B: تست مدیریت پیکر بندی.
۱۳۹.....	۱-۳-۵ فاز پنجم- تعمیر و نگهداری و بهره برداری.
۱۳۹.....	۱-۳-۵-۱ فاز ۵A: نگهداری و عملیات.
۱۳۹.....	۱-۳-۵-۲ فاز ۵B: انجام بررسی های وضعیت سلامت به صورت دوره ای.
۱۳۹.....	۱-۳-۵-۳ فاز ۵C: تضمین تأیید تغییر.
۱۳۹.....	۱-۳-۵-۴ یک نمونه جریان کاری تست SDLC.
۱۴۰.....	۴-۱ تست های نفوذ برنامه های کاربردی تحت وب بر اساس OWASP.
۱۴۰.....	۱-۴-۱ تست نفوذ برنامه های کاربردی تحت وب چیست؟
۱۴۰.....	۱-۴-۲ آسیب پذیری، تهدید، تست.
۱۴۱.....	۱-۴-۳ معرفی متدولوژی تست OWASP.
۱۴۱.....	۱-۴-۳-۱ حالت انفعالی.
۱۴۱.....	۱-۴-۳-۲ حالت فعال.
۱۴۱.....	۱-۴-۴ معرفی کنترل های تست OWASP.
۱۴۴.....	۱-۴-۴-۲ جمع آوری اطلاعات.
۱۴۴.....	۱-۴-۴-۲-۱ OWASP-IG-001 - روبات ها و خزنده ها
۱۴۵.....	۱-۴-۴-۲-۲ OWASP-IG-002 - کشف/شناسایی موتور جستجو
۱۴۵.....	۱-۴-۴-۲-۳ OWASP-IG-003 - شناسایی نقاط ورود به برنامه کاربردی
۱۴۵.....	۱-۴-۴-۲-۴ OWASP-IG-004 - تست اثرانگشت برنامه کاربردی تحت وب
۱۴۶.....	۱-۴-۴-۲-۵ OWASP-IG-005 - کشف برنامه کاربردی
۱۴۶.....	۱-۴-۴-۲-۶ OWASP-IG-006 - تحلیل کدهای خطا

۱۴۶	تست مدیریت پیکر بندی	۱-۴-۴-۱
۱۴۶	OWASP-CM-001 - SSL/TLS تست	۱-۴-۴-۱-۱
۱۴۶	OWASP-CM-002 - تست شنود کننده پایگاه داده	۱-۴-۴-۱-۲
۱۴۷	OWASP-CM-003 - تست مدیریت پیکربندی زیرساخت	۱-۴-۴-۱-۳
۱۴۷	OWASP-CM-004 - تست مدیریت پیکربندی برنامه کاربردی	۱-۴-۴-۱-۴
۱۴۷	OWASP-CM-005 - تست مدیریت فرمت های فایل	۱-۴-۴-۱-۵
۱۴۸	OWASP-CM-006 - فایل های قدیمی، پشتیبان و ارجاع داده نشده	۱-۴-۴-۱-۶
۱۴۸	OWASP-CM-007 - رابط های ادمین برنامه زیرساخت و کاربردی	۱-۴-۴-۱-۷
۱۴۸	OWASP-CM-008 -XST و HTTP تست متدهای	۱-۴-۴-۱-۸
۱۴۹	تست تصدیق	۱-۴-۴-۲
۱۴۹	OWASP-AT-001 - انتقال گواهی نامه ها بر روی یک کانال رمز شده	۱-۴-۴-۲-۱
۱۴۹	OWASP-AT-002 - تست جهت شمارش کاربر	۱-۴-۴-۲-۲
۱۴۹	OWASP-AT-002 - تست حساب کاربری قابل حدس (دیکشنری)	۱-۴-۴-۲-۳
۱۴۹	OWASP-AT-004 -Brute Force تست	۱-۴-۴-۲-۴
۱۵۰	OWASP-AT-005 - تست دور زدن طرح های احراز هویت	۱-۴-۴-۲-۵
۱۵۰	OWASP-AT-006 - تست آسیب پذیر بودن به یادآوری پسورد و بازیابی آن	۱-۴-۴-۲-۶
۱۵۰	OWASP-AT-007 - تست مدیریت خروج و حافظه نهان مرورگر	۱-۴-۴-۲-۷
۱۵۰	OWASP-AT-008 -CAPTCHA تست	۱-۴-۴-۲-۸
۱۵۰	OWASP-AT-009 - تست احراز هویت عوامل چندگانه	۱-۴-۴-۲-۹
۱۵۱	OWASP-AT-010 - تست شرایط رقابت	۱-۴-۴-۲-۱۰
۱۵۱	مدیریت نشست	۱-۴-۴-۳
۱۵۱	OWASP-SM-001 - تست طرح مدیریت نشست	۱-۴-۴-۳-۱
۱۵۱	OWASP-SM-002 - تست ویژگی های کوکی ها	۱-۴-۴-۳-۲
۱۵۲	OWASP-SM-003 - تست تثبیت نشست	۱-۴-۴-۳-۳
۱۵۲	OWASP-SM-004 - تست متغیرهای افشاء شده نشست	۱-۴-۴-۳-۴

۱۵۲.....	OWASP-SM-005 - CSRF تست ۱-۴-۴-۳-۵
۱۵۲.....	تست مجوز ۱-۴-۴-۴
۱۵۳.....	OWASP-AZ-001 - تست پیمایش مسیر ۱-۴-۴-۴-۱
۱۵۳.....	OWASP-AZ-002 - تست دور زدن طرح مجوز قانونی ۱-۴-۴-۴-۲
۱۵۳.....	OWASP-AZ-003 - تست بالا بردن حق ویژه ۱-۴-۴-۴-۳
۱۵۳.....	OWASP-BL-001 - تست منطق کسب و کار ۱-۴-۴-۵
۱۵۳.....	تست اعتبارسنجی داده ها ۱-۴-۴-۶
۱۵۴.....	OWASP-BL-001 - XSS منعکس شده ۱-۴-۴-۶-۱
۱۵۴.....	OWASP-BL-002 - XSS ذخیره شده ۱-۴-۴-۶-۲
۱۵۴.....	OWASP-BL-003 - XSS بر پایه DOM تست ۱-۴-۴-۶-۳
۱۵۴.....	OWASP-BL-004 - Cross Site Flashing تست ۱-۴-۴-۶-۴
۱۵۴.....	OWASP-BL-005 - SQL تزریق ۱-۴-۴-۶-۵
۱۵۵.....	OWASP-BL-006 - LDAP تزریق ۱-۴-۴-۶-۶
۱۵۵.....	OWASP-BL-007 - ORM تزریق ۱-۴-۴-۶-۷
۱۵۵.....	OWASP-BL-008 - XML تزریق ۱-۴-۴-۶-۸
۱۵۵.....	OWASP-BL-009 - SSI تزریق ۱-۴-۴-۶-۹
۱۵۶.....	OWASP-BL-010 - XPath تزریق ۱-۴-۴-۶-۱۰
۱۵۶.....	OWASP-BL-011 - IMAP/SMTP تزریق ۱-۴-۴-۶-۱۱
۱۵۶.....	OWASP-BL-012 - تزریق کد ۱-۴-۴-۶-۱۲
۱۵۶.....	OWASP-BL-013 - فرمان دادن به سیستم عامل ۱-۴-۴-۶-۱۳
۱۵۶.....	OWASP-BL-014 - سر ریز بافر ۱-۴-۴-۶-۱۴
۱۵۶.....	OWASP-BL-015 - تست آسیب پذیری تکوین یافته ۱-۴-۴-۶-۱۵
۱۵۷.....	OWASP-BL-016 - HTTP تست تکه کردن/قاچاق ۱-۴-۴-۶-۱۶
۱۵۷.....	تست انکار سرویس ۱-۴-۴-۷
۱۵۷.....	OWASP-DS-001 - SQL Wildcard تست حملات ۱-۴-۴-۷-۱

۱۵۷.....	OWASP-DS-002 - قفل کردن حساب مشتریان
۱۵۷.....	OWASP-DS-003 - سر ریزهای بافر
۱۵۷.....	OWASP-DS-004 - تخصیص شیء تعیین شده توسط کاربر
۱۵۷.....	OWASP-DS-005 - ورودی کاربر به عنوان شمارش گر حلقه
۱۵۸.....	OWASP-DS-006 - نوشتن داده های فراهم شده توسط کاربر بر روی دیسک
۱۵۸.....	OWASP-DS-007 - شکست در آزاد کردن منابع
۱۵۸.....	OWASP-DS-008 - ذخیره بیش از حد داده ها در نشست
۱۵۸.....	۱-۴-۴-۸ تست سرویس های وب
۱۵۸.....	OWASP-WS-001 - جمع آوری اطلاعات سرویس وب
۱۵۸.....	OWASP-WS-002 - WSDL تست
۱۵۸.....	OWASP-WS-003 - XML تست ساختار
۱۵۹.....	OWASP-WS-004 - XML تست سطح محتوای
۱۵۹.....	OWASP-WS-005 - HTTP در REST/GET تست پارامترهای
۱۵۹.....	OWASP-WS-006 - SOAP ضمیمه های ناشایسته
۱۵۹.....	OWASP-WS-007 - تست بازپخش
۱۵۹.....	۱-۴-۴-۹ Ajax تست
۱۵۹.....	OWASP-AJ-001 - Ajax آسب پذیری های
۱۶۰.....	OWASP-AJ-002 - Ajax تست چگونگی
۱۶۰.....	۵-۱ نتیجه گیری
۱۶۰.....	۶-۱ سوالات متداول
۱۶۰.....	۷-۱ منابعی برای مطالعات بیشتر

۶ آزمایشگاه تست امنیت نرم افزار

۱۶۲.....	۱-۱ خلاصه فصل
۱۶۲.....	۲-۱ معرفی آزمایشگاه های تست امنیت نرم افزار
۱۶۲.....	۱-۲-۱ آزمایشگاه CCTl

- ۱۶۲.....Ounce آزمایشگاه ۱-۲-۲
- ۱۶۳.....Bug Huntress آزمایشگاه ۱-۲-۳
- ۱۶۳.....وظایف آزمایشگاه امنیت نرم افزار ۳-۱
- ۱۶۳.....متدولوژی ایجاد آزمایشگاه امنیت نرم افزار ۴-۱
- ۱۶۳.....طراحی آزمایشگاه امنیت نرم افزار ۱-۴-۱
- ۱۶۳.....بخش فنی ۱-۴-۱-۱
- ۱۶۴.....بخش مدیریتی ۱-۴-۱-۲
- ۱۶۴.....پیاده سازی آزمایشگاه امنیت نرم افزار ۱-۴-۲
- ۱۶۵.....فرآیندهای ایجاد آزمایشگاه امنیت نرم افزار ۵-۱
- ۱۶۵.....اندازه گیری و تحلیل امنیت نرم افزار ۱-۵-۱
- ۱۶۶.....شناسایی و تست متداول ترین کنترل های امنیتی ۱-۵-۲
- ۱۶۶.....مدل سازی تهدیدات ۱-۵-۳
- ۱۶۷.....رویکرد مهاجم-محور ۱-۵-۳-۱
- ۱۶۷.....رویکرد نرم افزار-محور ۱-۵-۳-۲
- ۱۶۷.....رویکرد دارایی-محور ۱-۵-۳-۳
- ۱۶۷.....کاهش خطاهای رایج برنامه نویسی ۱-۵-۴
- ۱۶۷.....ارائه خطاها به منظور شناسایی الگوی رفتاری ناشناخته و جلوگیری از ایجاد آن ۱-۵-۵
- ۱۶۸.....Cybox: بیان قابل مشاهده سایبری ۱-۵-۵-۱
- ۱۶۸.....CAPEC: شمارش و دسته بندی الگوهای متداول حمله ۱-۵-۵-۲
- ۱۶۸.....انجام تست های امنیتی عملی با استفاده از ابزارهای موجود ۱-۵-۶
- ۱۶۸.....۶-۱ ابزارهای تست امنیت نرم افزار
- ۱۶۸.....دسته بندی ابزارهای تست امنیت ۱-۶-۱
- ۱۶۸.....۱-۶-۱-۱ ابزارهای منبع باز تست جعبه سیاه
- ۱۶۹.....۱-۶-۱-۲ ابزارهای تجاری تست جعبه سیاه
- ۱۶۹.....۱-۶-۱-۳ ابزارهای منبع باز/ایگان تحلیلگر کد منبع

۱۶۹.....	۱-۶-۱-۴ ابزارهای تجاری تحلیلگر کد منبع.....
۱۷۰.....	۱-۶-۱-۵ ابزارهای منبع باز تست پذیرش.....
۱۷۰.....	۱-۶-۱-۶ ابزارهای تست آسیب پذیری های خاص.....
۱۷۱.....	۱-۶-۲ معرفی برخی از اسکنرهای امنیتی.....
۱۷۱.....	۱-۶-۲-۱ Netsparker نرم افزار.....
۱۷۱.....	۱-۶-۲-۱-۱ امکانات نرم افزار.....
۱۷۲.....	۱-۶-۲-۱-۲ معرفی برخی خروجی های اصلی برنامه.....
۱۷۲.....	۱-۶-۲-۲ Acunetix Web Vulnerability Scanner نرم افزار.....
۱۷۳.....	۱-۶-۲-۲-۱ امکانات نرم افزار.....
۱۷۳.....	۱-۶-۲-۲-۲ معرفی برخی خروجی های اصلی برنامه.....
۱۷۴.....	۱-۶-۲-۳ Subgraph Vega نرم افزار.....
۱۷۴.....	۱-۶-۲-۳-۱ امکانات نرم افزار.....
۱۷۵.....	۱-۶-۲-۳-۲ معرفی برخی خروجی های اصلی برنامه.....
۱۷۵.....	۱-۶-۲-۴ Owasp Zed Attack Proxy نرم افزار.....
۱۷۶.....	۱-۶-۲-۴-۱ امکانات نرم افزار ZAP.....
۱۷۶.....	۱-۶-۲-۵ W3af نرم افزار.....
۱۷۷.....	۱-۶-۲-۶ AppScan – IBM نرم افزار.....
۱۷۷.....	۱-۶-۲-۶-۱ ویرایش ها.....
۱۷۷.....	۱-۶-۲-۷ ParosPro – MileSCAN Technologies نرم افزار.....
۱۷۷.....	۱-۶-۳ توزیع Kali لینوکس، ابزاری برای تست امنیت نرم افزار.....
۱۷۸.....	۱-۶-۳-۱ مجموعه ابزارهای تست نفوذ در توزیع Kali.....
۱۷۹.....	۱-۶-۴ الویت به کار گیری ابزارها.....
۱۸۱.....	۷-۱ پدافند غیر عامل.....
۱۸۱.....	۱-۷-۱ سطوح فعالیت ها.....
۱۸۱.....	۱-۷-۱-۱ سطح حیاتی.....

- ۱۸۱.....۱-۷-۱-۲ سطح حساس
- ۱۸۱.....۱-۷-۱-۳ سطح مهم
- ۱۸۱.....۱-۷-۲ برخی ملاحظات پدافند غیرعامل در ارتباط با آزمایشگاه تست امنیت نرم افزار
- ۱۸۲.....۸-۱ نتیجه گیری
- ۱۸۲.....۹-۱ سوالات متداول
- ۱۸۲.....۱۰-۱ منابعی برای مطالعات بیشتر
- ۱۸۳.....ضمیمه ۱ چک لیست های واری و اعتبار سنجی مرتبط با سیاست های امنیتی در فازهای SDLC
- ۱۹۱.....ضمیمه ۲ راه حل ها و ارائه دهندگان تست ۲۰ کنترل مهم امنیتی
- ۱۹۹.....ضمیمه ۳ واژه نامه انگلیسی به فارسی

فصل اول

۱

تست نرم افزار

تست نرم افزار

1-1 خلاصه فصل

«کیفیت، هدف اصلی تمامی افراد، شرکت ها و بصورت کلی سازمان هایی است که در جهت تولید و توسعه نرم افزار گام بر می دارند. در واقع کیفیت نرم افزار یک اصل مهم رقابتی در زمینه تولید یک محصول است، یکی از شاخه های اصلی برای رسیدن به این هدف، تست یا آزمون نرم افزار است. در واقع تست نرم افزار به دنبال خطایابی و عیب یابی محصول، قبل از تحویل به مشتری است. اینکه هم توسعه دهندگان و هم کاربران نهایی بر روی یک نرم افزار کارآمد و قابل بکارگیری که پاسخگوی نیازمندی های تعریف شده باشد هم نظر باشند بسیار اهمیت دارد.

آمارهایی که از سازمان های رسمی^۱ در زمینه مشکلات نرم افزارها و پیامدهای ناشی از آن منتشر می شود، مشکلاتی که در برخی موارد غیر قابل جبران می باشند، نیاز و توجه ویژه به مقوله تست نرم افزار را نشان می دهد، فعالیت هایی که باید در فرآیندهای چرخه حیات نرم افزار در نظر گرفته شود و با دیدی جامع و توجه به تمام جوانب، آزمون های متفاوتی را انجام داد تا بتوان ریسک های موجود در محصول را به حداقل برسانیم.

اهداف کلی این بخش:

- آشنایی با مفهوم تست نرم افزار و بررسی چرخه حیات آن
- بررسی روش ها و سطوح تست نرم افزار
- آشنایی با متریک های تست نرم افزار
- آشنایی با دوره های تست و استانداردهای مرتبط

1-2 تست نرم افزار چیست

تست نرم افزار، فرآیند اجرای یک برنامه یا سیستم با هدف پیدا کردن خطاهاست، یا هر فعالیتی با هدف ارزیابی ویژگی یا قابلیت یک برنامه یا سیستم مشخص و برآورده کردن نتایج مورد نیاز را، دربر می گیرد. تست نرم افزار، برخلاف دیگر فرآیندهای فیزیکی که ورودی ها دریافت و خروجی ها تولید می شوند نیست. تفاوت های نرم افزار در شیوه ای است که در آن با شکست مواجه شده است. اغلب سیستم های فیزیکی در یک مجموعه از راه های ثابت شکست می خورند. در مقابل، نرم افزار می تواند از راه های بسیار متفاوتی و ناشناخته ای دچار شکست شود. به طور کلی کشف تمام حالت های مختلف شکست برای نرم افزار، کاری غیر ممکن است. بر خلاف اکثر سیستم های فیزیکی، اغلب کاستی ها در نرم افزار، خطاهای طراحی هستند نه عیب های ساخت. نرم افزار از فرسایش، فرسودگی و کهنگی رنج نمی برد، به طور کلی تا زمان به روز رسانی ها یا تا زمان منسوخ شدن، تغییر نمی کند. بنابراین یک بار که نرم افزار تولید شود، عیوب طراحی (یا اشکال ها)^۲ فراموش می شوند و تا زمان فعال سازی، پنهان باقی می ماند.

اشکالات نرم افزاری تقریباً همیشه در هر ماژول نرم افزاری با اندازه متوسط وجود دارند؛ نه به خاطر اینکه برنامه نویسان بی دقت یا بی مسئولیت هستند، بلکه به خاطر اینکه به طور کلی پیچیدگی نرم افزار سخت و

دیر مهار است و بشر تنها توانایی محدودی برای مدیریت پیچیدگی دارد. همچنین این مطلب برای هر سیستم پیچیده ای صحیح است که کاستی‌های طراحی، هرگز نمی‌توانند به طور کامل نفی شوند.

به دلیلی مشابه پیچیدگی، کشف عیوب طراحی در نرم افزار، به همان اندازه مشکل است. به این علت که نرم افزار و هر سیستم دیجیتالی، پیوسته نیستند، تست مقادیر مرزی برای تضمین صحت و درستی، کافی نمی‌باشند. تمام مقادیر ممکن نیاز دارند که تست و تأیید شوند اما تست کامل غیر عملی است. تست جامع یک برنامه ساده که تنها دو ورودی عدد صحیح ۳۲ بیتی را با هم جمع می‌کند، موجب 2^{64} نمونه تست مجزا می‌شود که انجام آن صدها سال زمان خواهد برد، حتی اگر تست‌ها با نرخ هزاران تست در ثانیه انجام شود. بدیهی است برای یک مازول نرم افزاری واقعی، نتیجه فراتر از مثال نام برده می‌باشد. اگر ورودی‌ها، از دنیای واقعی باشند، مشکل بدتر خواهد شد. زیرا زمان بندی و تأثیرات محیطی غیرقابل پیش بینی و اثرات متقابل انسان، همگی پارامترهای ورودی ممکن تحت بررسی، خواهند بود.

مشکلات بیشتر در رابطه با طبیعت پویای برنامه‌ها می‌باشد. اگر در طی تست اولیه شکستی رخ دهد و کد تغییر یابد، نرم افزار ممکن است حالا برای یک نمونه تستی که قبلاً کار نمی‌کرده، کار کند و جواب دهد. اما رفتار آن بر روی نمونه تست‌های پیش از رخ دادن خطا، که قبلاً آن‌ها را سپری کرده، دیگر نمی‌تواند تضمین شده باشد. برای به حساب آوردن این امکان، عملیات تست باید از ابتدا شروع شود که هزینه انجام این کار اغلب گران و بازدارنده است.

عملیات تست معمولاً برای اهدافی انجام می‌گیرد از جمله:

• بهبود کیفیت

از آنجایی که کامپیوترها و نرم افزار در برنامه‌های کاربردی حیاتی مورد استفاده قرار می‌گیرند، پیامد یک اشکال می‌تواند بسیار شدید و جدی باشد. اشکالات می‌توانند باعث خسارت‌های بزرگی شوند. اشکال‌ها در سیستم‌های بحرانی باعث سقوط هواپیما، انحراف مأموریت شاتل فضایی، توقف معاملات در بازار بورس و موارد بدتری شده‌اند. اشکالات می‌توانند باعث کشتن انسان‌ها و باعث فجایع شوند. در دنیای احاطه شده با کامپیوتر، کیفیت و قابلیت اطمینان نرم‌افزار، موضوع مرگ و زندگی است.

کیفیت به معنای منطبق بودن با الزامات طراحی مشخص و تعیین شده است. وجود صحت، حداقل نیازمندی کیفیت، به معنای انجام آنچه که نیاز است می‌باشد و البته تحت شرایط مشخص شده. اشکال زدایی، دیدگاه محدودی از تست نرم‌افزار است که به شدت به وسیله برنامه‌نویس برای پیدا کردن نواقص طراحی انجام می‌شود. نقص ناشی از طبیعت انسان تقریباً این امر را غیر ممکن می‌سازد که یک برنامه نسبتاً پیچیده در اولین بار درست باشد. پیدا کردن مشکلات و رفع آن‌ها، هدف اشکال زدایی در فاز برنامه نویسی است.

همچنین می‌توانیم کیفیت در میان محصولات مختلف را تحت خصوصیات مشابه بر اساس نتایج یک تست یکسان مقایسه کنیم. ما نمی‌توانیم کیفیت را به صورت مستقیم تست کنیم اما این کار را می‌توانیم به وسیله فاکتورهای مربوطه برای قابل مشاهده کردن کیفیت، انجام دهیم. این فاکتورها می‌توانند به عوامل و جزئیات بیشتری در سطوح پایین تر شکسته شوند. جدول ۱۱ متداول ترین ملاحظات و فاکتورهای کیفیت را نمایش می‌دهد.

جدول ۱-۱ نمونه فاکتورهای کیفیت نرم افزار

کیفیت های بیرونی ^۱	کیفیت های درونی ^۲	کیفیت های بعدی ^۳
صحت	کارآیی	انعطاف پذیری
قابلیت اطمینان	تست پذیری	قابلیت استفاده مجدد
قابلیت استفاده	مستندات	قابلیت نگهداری
تمامیت و یکپارچگی	ساختار	

یک تست خوب، سنجش ها برای همه فاکتورهای مرتبط را فراهم می کند. اهمیت هر فاکتور خاص، از یک برنامه کاربردی به برنامه دیگر متفاوت است. هر سیستمی که با زندگی انسان رابطه دارد، در خطر است و باید تأکید شدید بر قابلیت اطمینان و یکپارچگی این سیستم ها داشت. در سیستم های تجاری قابلیت های استفاده و نگهداری فاکتورهای کلیدی هستند در حالی که برای یک برنامه علمی ممکن است هیچ یک قابل توجه نباشند.

• واریسی و اعتبارسنجی

یکی دیگر از اهداف مهم تست، بحث واریسی و اعتبارسنجی می باشد. تست می تواند به عنوان معیار به کار رود. تست به شدت به عنوان یک ابزار در فرایند V&V مورد استفاده قرار می گیرد. تست کنندگان می توانند بر اساس تفسیری از نتایج تست، ادعا کنند که محصول تحت شرایط خاص کار می کند، یا کار نمی کند. تست با هدف اعتبارسنجی آثار و نتایج محصول، تست تمیز یا تست مثبت نامیده می شود. تعداد محدودی تست نمی تواند کار کردن نرم افزار در تمام موقعیت ها را تأیید کند. در مقابل، تنها یک تست شکست خورده برای نشان دادن این که نرم افزار کار نمی کند کافی است. تست کثیف یا تست منفی به تست هایی گفته می شود که هدفشان شکست نرم افزار می باشد و نمایش اینکه نرم افزار کار نمی کند.

• تخمین قابلیت اطمینان

قابلیت اطمینان نرم افزار ارتباطات مهمی با بسیاری از جنبه های نرم افزار از جمله ساختار و میزان تست آن دارد. بر اساس پروفایل عملیاتی (تخمینی از دفعات استفاده از ورودی های گوناگون به برنامه)، عملیات تست می تواند به عنوان یک متد نمونه آماری به منظور به دست آوردن داده های ناموفق برای تخمین قابلیت اطمینان به کار گرفته شود.

1-3 وابستگی تست و کیفیت

در بسیاری از شرکت ها حدود ۳۰ تا ۵۰ درصد هزینه تولید نرم افزار صرف تست آن می شود. هنوز هم خیلی از افراد اعتقاد دارند که نرم افزارها قبل از انتشار، به درستی تست نمی شوند. این شرایط به دو دلیل به وجود می آید. اول این که تست نرم افزار امری بسیار مشکل است. دوم این که تست معمولاً بدون متدولوژی مشخص و ابزار لازم انجام می گیرد. آنچه که باعث می شود تا شرکت ها هزینه زیادی صرف تست نرم افزارها کنند، چیزی جز دستیابی به کیفیت مطلوب نیست.

تست نرم افزار ۱ فصل اول

کیفیت چیزی است که ما در تولیدات، فرایندها و خدمات به دنبال آن هستیم. کیفیت یک ویژگی منحصر به فرد نیست، بلکه یک مشخصه چند بعدی است و می تواند در یک فرایند یا محصول وجود داشته باشد. کیفیت به مشخصه ای اطلاق می شود که :

- تعدادی از نیازمندی های توافق شده را برآورده کند.
- به وسیله معیارهای سنجش و اندازه گیری توافق شده ارزیابی شود.
- به وسیله فرآیند مورد قبولی تولید شود.

کیفیت نرم افزار را می توان به دو دسته تقسیم کرد:

کیفیت محصول: کیفیت محصول در مورد کیفیت محصولی که به وسیله فرآیندها تولید می شود، بحث و نتیجه گیری می کند.

کیفیت فرآیند: کیفیت فرآیند به میزان مقبولیت و کیفیت یک فرآیند که برای تولید یک محصول اجرا می شود، اشاره می کند. اگر ما یک فرایند با کیفیت داشته باشیم، ریسک تولید محصول با کیفیت پایین بسیار کم می شود، در حالی که خلاف این مورد معمولاً "درست نیست. یعنی داشتن یک محصول نهایی با کیفیت بالا، دلیل بر وجود یک فرآیند با کیفیت بالا نیست.

1-4 دلایل اهمیت تست نرم افزار و پیامدهای عدم وجود آن

صرف نظر از محدودیت ها، تست بخش جدایی ناپذیر در توسعه نرم افزار است و باید به طور گسترده در هر فاز در چرخه توسعه نرم افزار مستقر شود. به طور معمول، بیش از ۵۰ درصد از زمان توسعه در تست به سر می برد. نرم افزاری که به درستی کار نکند، می تواند تأثیر مخرب زیادی بر روی یک سازمان داشته باشد. بدون انجام ارزیابی بر روی نرم افزار، نمی توان از مشکلات عملیاتی، کیفیتی و در نتیجه امنیتی آن مطلع شد که این مسئله منجر به بروز مشکلات زیادی از جمله این موارد می شود:

- ❖ از دست دادن سرمایه: این مورد می تواند شامل از بین رفتن حقوق مشتری ها باشد که به دلیل عدم رعایت الزامات قانونی منجر به مجازات های مالی می گردد.
- ❖ از دست دادن زمان: علت این مورد می تواند معاملاتی باشد که زمان زیادی برای پردازش می برد اما می تواند شامل کارکنی شود که قادر نیست به خاطر یک خطا یا کاستی کار کند.
- ❖ خسارت به شهرت کسب و کار: اگر یک سازمان با توجه به مشکلات نرم افزاری، قادر به ارائه خدمات به مشتریان خود نباشد، مشتریان اعتماد و ایمان خود به این سازمان را از دست می دهند. (و احتمالاً کسب و کار خود را در جایی دیگر انجام می دهند)
- ❖ آسیب یا مرگ: برخی از سیستم های بحرانی در بحث ایمنی، اگر به درستی کار نکنند می توانند باعث آسیب و یا مرگ و میر شوند. (به عنوان مثال نرم افزار کنترل ترافیک پرواز)

1-5 چه موقع باید عمل تست انجام شود

برای اینکه تست هزینه موثر داشته باشد باید بتوان خطاها را تا آنجا که امکان دارد به محض بروزشان

از منبع حذف کند. بنابراین تست باید یک فرایند مستمر و پیوسته در طول چرخه حیات نرم افزار باشد. تست متدیک می تواند هرگاه که یک محصول به یک وضعیت جدید از پالایش می رسد، انجام شود. اما مهندسی نرم افزار صحیح به بیش از این نیاز دارد. مهندس نرم افزار باید تست غیر رسمی را همزمان با توسعه محصول انجام دهد. فرضیات باید حذف شوند و با قوانین بررسی شده جایگزین شوند. وقتی یک خطا آشکار می شود باید پیگیری شود تا منبع آن یافت شود، سپس روالها باید تغییر داده شده و اصلاح شوند. تا از وقوع مجدد خطاها جلوگیری شود. با ترکیب تست غیر رسمی و متدیک هدفمان ایجاد یک روش کار است که فقط منحصر به کشف زود هنگام خطاها نباشد بلکه از وقوعشان جلوگیری شود. گذشته از همه آنچه که ذکر شد موثر ترین روش تولید محصول از لحاظ هزینه، دریافت محصول سالم از همان ابتدا است که هزینه ناشی از درمان اشتباهات را از بین می برد.

1-6 تست نرم افزار در تنوري و عمل

یکی از مشکلات مهم در کامپیوتر توسعه نرم افزار قابل اعتماد است. متد اصلی برای رسیدن به یک محصول نرم افزاری مطمئن تست می باشد. استراتژی های تست محصول، آن را روی زیر مجموعه کوچکی از ورودی هایش اجرا می کنند و مشکل اساسی از همین جا آغاز می شود که موارد تست را چگونه انتخاب کنیم تا تنها با تکیه بر نمونه کوچکی از ورودی ها اطمینان کافی در مورد محصول کسب کنیم. استراتژی های فراوانی در زمینه تست بیان شده است. اغلب استراتژی ها نوعی دانش و آگاهی در مورد برنامه را به کار می گیرند که ممکن است ساختار برنامه، مشخصات تابعی برنامه اطلاعات در مورد انواع خطاهایی که غالباً روی می دهند یا ترکیبات گوناگونی از آنها باشد. اتوماسیون فرایند تست عامل مهمی در بکارگیری یک متد است که شامل ابزارهایی برای تولید موارد تست است تا یک معیار تست را برآورده سازد. ابزارهایی که موارد تست را اجرا کرده و نظارت می کنند و ابزارهایی که صحت خروجی ها را بررسی می کنند. مساله مهم دیگر نوع نتایجی است که از حاصل تست می توان به آنها (درباره قابلیت اعتماد برنامه) رسید.

1-6-1 تنوري تست

در طول سال ها استراتژی های تست زیادی پیشنهاد شده اند که هر کدام بر یک قضیه منطقی استوار است. اگر شما عمل تست را چنان و چنین انجام ندهید چگونه می توانید مطمئن باشید که برنامه کار می کند؟ یا "اگر ما عمل X را انجام دهیم ممکن است Y را از دست بدهیم لذا بهتر است Z را انجام دهیم" حتی طرحهای تستی وجود دارند که معادل اثبات های برنامه هستند. متأسفانه این طرح ها مجبورند به طور جدی انواع خطاهایی را که می توانند کنترل شوند یا آنهایی را که در عمل نمی توانند پیاده سازی شوند را محدود کنند.

ارزیابی های زیادی از استراتژیهای گوناگون صورت گرفته و مقایسه های زیادی بین آنها انجام شده است. اما چندین عامل، ارزش آنها را در عمل محدود می کند. و واضح است که هیچ برنده آشکاری از نظر استراتژی وجود ندارد. به عنوان مثال معنی اینکه تست مسیر از تست شاخه بهتر است چیست؟ درست است که تست مسیر برنامه را کاملتر تست می کند ولی هزینه اش نیز بالاتر است و هنوز این امکان وجود دارد که مجموعه ای از n مورد تست برای تست شاخه در نهایت به خطایی دست یابد که $n*m$ مورد تست مسیر نتواند. آیا یک مقایسه عادلانه، تست شاخه را m برابر تکرار می کند؟ از هر روش به چه نتایجی درباره قابلیت اعتماد برنامه می توان رسید؟ تست تصادفی دانش حداقلی را از برنامه برای طراحی موارد تست به کار می برد. به نظر می رسد که هیچ شانس برای رقابت با استراتژی هایی که دانش برنامه را برای طراحی موارد تست به کار می برند وجود نداشته باشد. اما برخی از مطالعات انجام شده نشان می دهند که تست تصادفی نزدیک به تست پارتیشن عمل می کند. از طرف دیگر موقعیت هایی وجود دارند که تست تصادفی در آنها ناموثر است.

1-6-2 تست در عمل

تعداد کمی از استراتژی های تست فراوانی که پیشنهاد شده اند در عمل به کار رفته اند. عوامل زیادی نظیر هزینه، کمبود ابزار اتوماسیون برای پشتیبانی آنها، مشکلات عملی و تئوری در پیاده سازی آنها، کمبود ارزیابی های واقعی کارایی و ... به این امر کمک می کنند. یک متدولوژی تست نرم افزار به صورت زیر است

- یک دور بازرسی دستی کد برنامه
- یک مقدار ۹۵ درصدی از پوشش سراسری شاخه
- یک مقدار ۹۰ درصدی از پوشش سراسری call – pair
- ۹۸ درصد تست بازگشتی اتوماتیک
- ۵۰ درصد یا بیشتر پوشش مسیر برای مازول های بحرانی.

چنین متدولوژی ای یک تعداد از استراتژی های تست را بدون تأمین نیازمندی های هر یک به طور کامل ترکیب می کند. مقایسه ها و ارزیابی های تئوری استراتژی های گوناگون، چگونه می توانند در داخل چنین ترکیبی از استراتژی ها به کار گرفته شوند؟ در حوزه قابلیت اعتماد نرم افزار سوالات از پاسخ ها فراترند. یک چیز که واضح به نظر می رسد این است که از تقابل بیشتر بین تئوری و عمل نتیجه بهتری حاصل می شود.

1-7 تفاوت تست و اشکال زدایی

تست اشکال زدایی نیست. اشکال زدایی در رابطه با خطاهای ویژه مستقر در اجزای محصول است اما تست با اجرای یک برنامه به قصد یافتن خطاها توسط ابزار منطقی یا فیزیکی صورت می گیرد. یک برنامه تست شده برنامه ای است که هنوز شرایطی که آن را وادار به شکست می کند در آن یافت نشده اند یعنی اگر نتیجه تست با نتیجه مورد نظر سازگار است، فرض می شود که تست قطعه با موفقیت روبرو شده و صحیح است. در تست یک نرم افزار دو هدف عمده مورد نظر است:

- رسیدن به کیفیت مناسب (Debug Testing): تست به عنوان وسیله ای برای رسیدن به قابلیت اطمینان به کار می رود که در اینجا هدف بررسی نرم افزار برای کشف باگهای قابل اصلاح است که نرم افزار را از نظر قابلیت اطمینان بهبود می بخشد.
- ارزیابی کیفیت موجود (Cope rational Testing): تست به عنوان وسیله ای برای کسب اطمینان از اینکه نرم افزار قابلیت اطمینان کافی برای به انجام رساندن مقاصد را که به منظور آنها طراحی شده است "دارا است" به کار می رود. در اینجا هدف ارزیابی قابلیت اطمینان است.

متدهای Debug انتخاب تصادفی داده تست از یک پروفایل عملیاتی را در نظر نمی گیرند در حالی که این امر برای متدهای عملیاتی بسیار حائز اهمیت است. متدهای Debug بدون هیچ مدرک واقعی ادعای موثر بودن در آشکار سازی نواقص و معایب و اصلاح و رفع آنها را دارند، اما یک ارزیابی قابل دفاع از قابلیت اطمینان نتایج ارائه نمی دهند.

از طرف دیگر متدهای عملیاتی که ارزیابی صحیح و دقیق را عرضه می کنند ممکن است برای رسیدن به قابلیت اطمینان مفید نباشند. بررسی ارتباط این دو هدف با به کارگیری تحلیل و آنالیز احتمالی امکان پذیر است. یک متد تست سیستماتیک معیاری برای انتخاب موارد تست دارد و یک معیار برای تصمیم گیری اینکه

چه موقع تست را پایان دهد. اغلب تست های سیستماتیک یا بر پایه یافتن اشکالات با نمونه برداری از همه موقعیت هایی که احتمال دارد منجر به شکست شود تکیه دارند، یا با تمرکز روی موقعیت هایی که بیشترین احتمال را در این امر دارند.

1-8 مفاهیم بنیادی در تست نرم افزار

در فرآیند توسعه، هدف آن است که یک سیستم با مشخصات خواسته شده تولید شود. هدف این فرآیند از یک سو برآورده ساختن نیازهای کاربران و از سوی دیگر تضمین کیفیت مناسب عملکرد سیستم است. بنابراین بایستی حاوی مکانیزم هایی برای اعتبارسنجی و واریسی باشد. پیدا کردن نقص در اواخر چرخه حیات توسعه می تواند به طور قابل توجهی در زمان و سرمایه پر هزینه باشد. فرآیندهای خوب V&V می توانند قبل از اینکه سیستم وارد تست شود، به کاهش نقص در توانمندی های عملیاتی کمک کنند. ارزیابی نرم افزار، یک عنصر از عنوان گسترده تری است که اغلب با واریسی و اعتبارسنجی شناخته می شود.

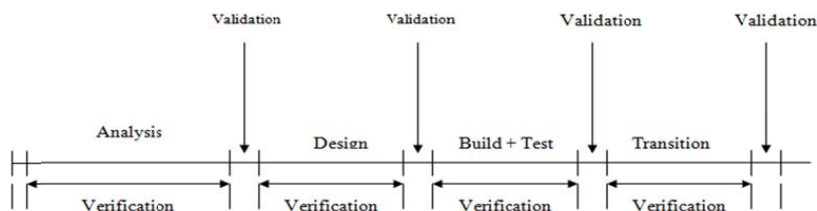
1-8-1 واریسی

مطابق استاندارد ISO 9000 واریسی عبارت است از فراهم آوردن شواهد عینی در مورد اینکه الزامات و یا خواسته های مشخص شده برآورده شده اند. اصطلاح "واریسی شده" به منظور مشخص کردن وضعیت مربوطه (پس از واریسی) بکار می رود. واریسی می تواند شامل فعالیت هایی باشد از قبیل:

- ✓ انجام محاسبات به روش های دیگر
- ✓ مقایسه مشخصات یک طراحی جدید با مشخصات طراحی های مشابه که درستی آنها به اثبات رسیده است
- ✓ انجام آزمون ها و اثبات از طریق نشان دادن
- ✓ بازنگری مدارک پیش از صدور

1-8-2 اعتبارسنجی

مطابق استاندارد ISO 9000، اعتبارسنجی عبارت است از تأیید از طریق فراهم آوردن شواهد عینی در مورد اینکه الزامات و یا خواسته ها برای استفاده مورد نظر یا کاربرد خاص برآورده شده اند. اصطلاح "اعتبارسنجی شده" به منظور مشخص کردن وضعیت مربوطه (پس از اعتبارسنجی) به کار می رود. شرایط استفاده می تواند واقعی یا شبیه سازی شده باشد. شکل I چگونگی ارتباط V&V در چرخه حیات نرم افزار را نمایش می دهد.



شکل I جایگاه V&V در SDLC