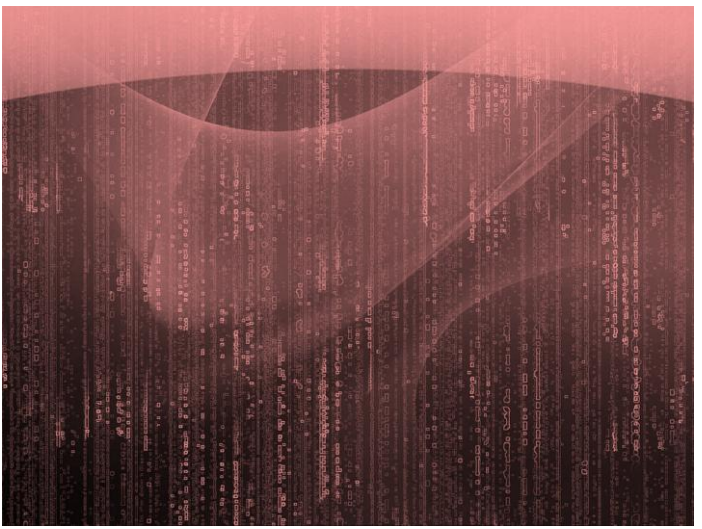
آشنایی با مفاهیم تست نفوذ در نرم افزارها



فهرست

آغاز

تست نفوذپذیری رویه‌ای است که برای ارزیابی امنیت نرم افزاری یک سازمان استفاده می‌شود. یک تیم برنامه‌نویس با استفاده از تکنیک های نفوذ[1]، یک حمله واقعی را شبیه سازی می‌کنند تا به این وسیله سطح امنیت یک سیستم (برنامه کاربردی) را مشخص کنند. تست نفوذپذیری به یک سازمان کمک می‌کند که ضعف‌های شبکه و ساختارهای اطلاعاتی خود را بهتر بشناسد و در صدد اصلاح آنها برآید. این امر به یک سازمان کمک می کند تا در مورد امنیت نیروها و شبکه خود یک ارزیابی واقعی داشته باشد.

عمومی‌ترین دلایل انجام ارزیابی امنیتی به شرح زیر می‌باشد:

۱. مشخص‌کردن خطرات و ریسک‌هایی که سرمایه‌های اطلاعاتی سازمان با آنها مواجه می‌شوند. در واقع می‌توان با ریسک‌های اطلاعاتی سازمان آشنا شد و سپس برای آنها به مقدار مورد نیاز هزینه کرد.
۲. با مشخص کردن نقاط ضعف و آسیب‌پذیری‌های سیستم‌های اطلاعاتی سازمان و بر طرف کردن آن‌ها به مقدار قابل توجهی از هزینه‌های صرف شده برای امنیت و هزینه‌های ایجاد شده به علت از دست رفتن اطلاعات کاسته می‌شود.
۳. یک ارزیابی دقیق و کامل امنیتی، آسودگی خاطر را برای سازمان به ارمغان می آورد.

---

[1] Hack

تامین امنیت در لایه‌ی نرم‌افزار به دلایل مطرح شده در زیریکی از پیچیده‌ترین حوزه‌های امنیت است:

۱. میلیون‌ها نرم‌افزار مختلف برای انجام امور مختلف وجود دارد که هیچ دوتای آنها از تمام ابعاد طراحی، پیاده‌سازی، پلتفورم، ظاهر و ... مشابه نیستند.
۲. تنوع و تعداد بسیار زیاد امکانات و قابلیت‌هایی که می‌توان در لایه‌ی نرم‌افزار داشت و طبعاً لزوم توجه به ملاحظات امنیتی هر یک.
۳. وجود پلتفورم‌های مختلف برای پیاده‌سازی نرم‌افزارها و طبعاً ملاحظات امنیتی خاص هر یک (انواع دیتابیس سرورها، انواع وب سرورها، انواع چارچوب‌های تولید نرم‌افزار و زبان‌ها)
۴. پیچیدگی و حجیم بودن بسیاری از نرم‌افزارها که مدیریت امنیت را در آن‌ها بسیار دشوار می‌سازد.
۵. کم‌تر ملموس بودن امنیت در لایه‌ی نرم‌افزار

در خصوص امنیت انواع نرم‌افزارها دو حوزه مطرح است:

۱. قابلیت‌ها و امکانات مربوط به امنیت هر نرم‌افزار(شناسایی کاربر، بررسی مجوزها، ثبت وقایع، رمزنگاری و...)
۲. طراحی، پیاده‌سازی و تست صحیح و امن اصل نرم‌افزار

در حوزه قابلیت ها و امکانات مربوط به امنیت هر نرم‌افزار، بحث حول اینمحور است که چه امکاناتی برای تامین امنیت نرم‌افزار لازم است و این امکانات باید چگونه باشند و چگونه عمل کنند و چگونه پیاده سازی شوند. برای مثال ابتدا تصمیم‌گیری در خصوص اینکه آیا نرم‌افزاری تصدیق هویت کاربران را لازم دارد یا خیر و سپس تصمیم‌گیری در خصوص اینکه با چه مکانیزمی تصدیق هویت صورت پذیرد (مبتنی بر نام کاربری و کلمه عبور یا مبتنی بر توکن و ....)

اما در حوزه طراحی، پیاده سازی و تست صحیح و امن نرم‌افزارها، بحث حول اینمحور است که یک نرم‌افزار و قابلیت‌های مختلف آن چه مربوط به اصل عملکرد نرم‌افزار باشد (مانند گزارش گیری)، چه مربوط به تامین امنیت نرم‌افزار باشند (مانند سیستم تصدیق هویت)، چگونه به صورت صحیح طراحی، پیاده سازی و تست گردند به نحوی که حداقل نقاط ضعف را برای سوءاستفاده بدخواهان داشته باشد. برای مثال چه نکاتی باید در خصوص طراحی و پیاده سازی قسمت چاپ اطلاعات نرم‌افزار در نظر گرفته شود تا یک فرد بدخواه

نتواند یک نسخه از اطلاعات چاپ شده را در یک محل دیگر ذخیرهکندیا پرینتر را دچار مشکل کند؟ و پس از لحاظ کردن نکات مربوط چگونه باید نرم‌افزار تست شود تا از درست کار کردن این تمهیدات اطمینان حاصل شود.

در پایان لازم به ذکر است، که اولاً حوزه امنیت نرم‌افزار یک حوزه بدیع و تازه است و تا قبل از آن، بیشتر تمرکز راه‌حل های امنیتی در لایه سیستم عامل (ضد ویروس ها و ...) و لایه شبکه (فایروال ها) بوده است و در واقع نسل سوم راه‌حل‌های امنیتی است و ثانیاً امنیت نرم‌افزاریک ویژگی جالب دارد که در سایر لایه‌ها مطرح نیست و آن ویژگی این است که می توان با ارائه راه‌حل هایی صحیح در حوزه امنیت نرم‌افزار بسیاری از نقاط ضعف لایه‌های دیگر را پوشاند و تهدیدهای ناظر به آنها را کاهش داد. برای مثال اگر یک نرم‌افزار یک ضد ویروس درونی داشته باشد ویروس‌های سیستم عامل نمی‌توانند تاثیر مخربی در آن ایجاد کنند و یا اگر در یک نرم‌افزار اطلاعات تبادلی بین مؤلفه های آن رمز شود نیازی به فعال‌سازی مکانیزم‌های رمزنگاری در لایه شبکه نیست.

متدولوژی اجرای پروژه

به طور کلی برای بررسی صحت و امنیت حوزه طراحی، پیاده سازی و تست دو روش جعبه سیاه و جعبه شفاف وجود دارد، در ادامه به بررسی آنها می‌پردازیم:

- ارزیابی جعبه‌ی سیاه

در روش جعبه‌ی سیاه، نرم‌افزار بدون اینکه اطلاعی از طراحی، پیاده‌سازی و سورس کد آن در اختیار باشد توسط تیم متخصص ارزیابی امنیت نرم‌افزار مورد بررسی قرار می گیرد و آنها بر اساس اصول علمی حوزه امنیت نرم‌افزار، تجربیات خود و تعدادی از ابزارهای ویژه این امر تلاش می‌کنند تا ملاحظات مربوط به امنیت نرم‌افزار استخراج شود.

- تکنیک‌های انجام تست جعبه‌ی سیاه

۱. انجام تست بر روی تمام واسط‌های بیرونی که به طور عادی برای استفاده از برنامه در اختیار کاربر قرار میگیرد مانند public classها. این تست شامل مراحل زیر است:

- اطمینان از اینکه واسط کاربری نسبت به تمام امکاناتی که در اختیار کاربر قرار می دهد تست شود. بدین معنی که Test Case ها باید به گونه ای باشند که بتوانند تمام کلاس‌های پیاده‌سازی شده در برنامه را استفاده کنند. در این مرحله، هم از ورودی تصادفی و هم از ورودی روتین استفاده می شود.
- انجام تست برای ورودی‌های مختلف: در این گام به بررسی درستی خروجی‌های مورد انتظار برای داده‌های مشخص می‌پردازیم، و از Handel کردن داده‌های غیرمجاز و یا داده‌هایی که باعث به وجود آمدنExceptionمی‌شوند اطمینان حاصل می‌کنیم. داده‌های ورودی می‌توانند به صورت تصادفی در محدوده مورد انتظار برنامه یا خارج از محدوده مورد انتظار برنامه تولید شوند. تست داده‌های خارج از محدوده مورد انتظار باعث می‌شود که قدرت برنامه نسبت به ورودی های غیر مجاز سنجیده شود.

۲. تست کارایی: در این قسمت کارایی برنامه نسبت به ورودی با حجم بالا سنجیده می شود. که دارای ۲ قسمت Stress Testing و Load Testing می‌باشد.

- Load Testing: این قسمت جهت تست کردن برنامه برای سنجش رفتار آن برای داده‌های نرمال و تست حداکثر ظرفیت استفاده می‌شود. این تست مشخص می‌کند که آیا برنامه توان نگه داشتن کارایی در حد قابل قبول را دارد و آیا می‌تواند مانعِ از بین رفتن داده‌ها شود.
- Stress Loading: در این قسمت رفتار برنامه در برخورد با باری بیش از حد توانش مورد بررسی قرار می گیرد هدف این قسمت پیدا کردن خطاهایی است که برنامه در مواجه شدن در این وضعیت‌ها از خود بروز می دهد. Synchronization issue (همگامی انتشار) ، Race Condition، Memory Leak. (تراوش حافظه) و... در

هنگام بر خورد با این مشکلات باید به منبع کد مراجعه کرده و اقدام به تصحیح کد شود.

۳. تست امنیتی[۱] به وسیله شبیه سازی محیط واقعی حمله برای برنامه اجرا می شود. این روش به شناسایی موارد آسیب پذیر برنامه با شبیه سازی تمام داده‌های ورودی مجاز می پردازد. هدف آنالیز پردازه[۲]های مختلف برنامه می‌باشد که ممکن است به خاطر نبود مهارت کافی در برنامه‌نویسی به خوبی سازماندهی نشده باشند. تست امنیت به شناسایی حفره‌هایی از برنامه که باعث نشت اطلاعات و دسترسی افراد غیر مجاز به سیستم می شود کمک می‌کند. انواع تست‌های امنیتی عبارتند از:

- معتبر بودن داده‌ها: اگر برنامه به صورت پیش فرض ورودی برنامه را معتبر بداند، می‌توان با تولید داده‌های متنوع غیرمعتبر باعث ایجاد خطا در برنامه شد.
- می توان با بدست آوردن کلید رمزنگاری داده‌ها، به داده‌های حساس پی برد.
- ایجاد Buffer overflow در برنامه.
- سعی در ایجاد نفوذ به برنامه با بدست آوردن حساب کاربری احراز هویت و تصدیق اصالت.
- تست نفوذ: در این مرحله تست کننده سعی می‌کند که به هر صورت ممکن به برنامه نفوذ کند در مرحله ممکن، تست کننده برای رسیدن به اهداف خود از برنامه هایی در این زمینه کمک بگیرد و یا از ترکیب رخنه‌های شناسایی شده در برنامه به هدف خود برسد.

---

[۱] Security Test

[۲] Process

- تست آسیب‌پذیری: در این قسمت تست کننده سعی می‌کند که برنامه را نسبت به نقاط آسیب‌پذیر معمول شناخته شده پویش کرده و نقاط آسیب‌پذیر را شناسایی کند، که این کار ممکن است توسط نرم‌افزارهایی که با این هدف توسعه یافته‌اند انجام شود.

- ارزیابی جعبه‌ی شفاف

در این تست، با دردست داشتن کدهای منبع[۱] برنامه، تست کننده به تمام اجزای درونی برنامه، منطق برنامه و ساختار کد برنامه بطور کامل سیطره دارد.

این تست از ۳ مرحله تشکیل شده است:

- Unit Testing: در این مرحله نرم افزارها به صورت مجزا بررسی می‌شوند منظور از unit در این‌جا زیر برنامه‌ای است که قابلیت شکسته شدن به زیر برنامه‌های کوچک‌تر را نداشته باشد. تست کننده در این مرحله شروع به نوشتن Test Caseهایی برای تست کد مزبور می‌کند. این مرحله به این دلیل اهمیت دارد که اگر کد ها به صورت unit تست نشوند، تست تمام برنامه‌ها به صورت یکباره بسیار سخت و زمان‌بر و در مواردی غیر ممکن است.
- Integration Testing: در این مرحله قسمت‌های مختلف نرم‌افزارو ارتباط آنها با یکدیگر مورد ارزیابی قرار می‌گیرد.
- Regression testing: در این مرحله تمام نرم‌افزار با تمام ابزارهای جانبی آن مورد بررسی قرار می‌گیرد تا اطمینان حاصل شود که تست های بالا باعث به وجود آمدن خطاهای نحوی ناخواسته نشده باشد و برنامه هنوز هم با نیازهای از پیش تعیین شده کامپایل[۲] می‌شود.

---

[۱] Source Code

[۲] Compile

تکنیک‌های انجام تست جعبه‌ی شفاف

۱. Crystal-Box Testing by Stubs and Drivers: در این قسمت بررسی می‌شود که آیا یک تکه کد با دریافت مقادیر مشخص به عنوان ورودی، خروجی مورد انتظار را تولید می‌کند یا خیر. در این قسمت از Stub[۱] وDriver[۲] استفاده می‌کنیم.

۲. Deriving Test Cases: در این قسمت شیوه‌ی انتخابtest case مناسب برای برنامه را بررسی می‌کنیم. با طراحی دقیق و منظم می‌توان بسیاری از bugهای برنامه مورد آزمایش را شناسایی کرد.

- Basis Path Testing: در این تکنیک فلوچارت برنامه‌ی مورد تست کشیده می‌شود و روابط توابع با یکدیگر مشخص می‌شود. در این مرحله test case های متنوع و مختلفی نوشته می‌شود تا تمامی حالات ممکن ارتباط بین توابع مورد بررسی قرار گیرد.
- Equivalence Partitioning/Boundary Value Analysis: در این قسمت به بررسی مقادیر ورودی حد مرزی توابع می‌پردازیم. بدون شک هنگامی که بر سر مقادیر ورودی تابع محدودیت هایی لحاظ می‌شود به Test Caseهای از این نوع نیازمندیم.

۳. Control-flow/Coverage Testing: Coverage Testing: این تست شامل سه مرحله می‌باشد که عبارتند از: پیدا کردن بخش‌هایی از برنامه که با Test Case های معمول آزمایش نشده است. تولیدTest Caseهای جدید

---

[۱] دستورات ساده برنامه نویسی است که به جای قست هایی از برنامه مورد تست که هنوزTest Case آن تولید نشده قرار می‌گیرد تا برنامه ساده شده و تست آن آسان‌تر شود.

[۲] ماژولی است که جهت فراخوانی برنامه مورد تست استفاده می شود. ماژول driver کار تولید ورودی، کنترل اجرای کد، گزارش نتیجه تست مزبور را بر عهده دارد .در یک تعریف ساده این ماژول وظیفه فراخوانی و مقدار دهی تابع را به عهده دارد.

برای ارزیابی نواحی تست نشده و تعیین میزانی برای اندازه گیری کمی Test Caseهای ایجاد شده برای برنامه.

- Method Coverage: درصد توابعی است که توسط Test Case ها فراخوانی می‌شود. از آن‌جا که تست‌های انجام شده باید بتواند تمام توابع را فراخوانی کند، لازم است بررسی فوق صورت پذیرد.
- Statement Coverage: درصد خط‌هایی از برنامه است که توسط Test Caseها اجرا می‌شود. تست‌ها باید به گونه‌ای طراحی شود که این مقدار به ۱۰۰٪ نزدیک شود. Decision/Branch Coverage: درصد شرط هایی از برنامه است که توسط Test Caseها بررسی می‌شود. تست‌های انتخاب شده باید تمام شرط های یک برنامه را بررسی کند.
- Condition Coverage: در این قسمت به طور عمیق‌تر به بررسی عبارات شرطی می‌پردازیم

۴. Data Flow Testing: در این قسمت سعی می‌شود که مشخص شود که متغیرهای برنامه چگونه تعریف شده‌اند و تخصیص‌دهی حافظه به آن‌ها چگونه می‌باشد.

۵. Failure (Dirty) Test Cases: در این قسمت به برنامه مانند جعبه سیاه نگاه کرده و در رابطه با راه هایی که یک شخص می‌تواند به برنامه نفوذ کند فکر می‌کنیم. ممکن است هنوز راه‌هایی باقی مانده باشد که با Test Case های مراحل قبل رفع نشده باشد.

خروجی‌ها

خروجی‌های زیر در پایان هر پروژه‌ی تست نفوذ باید تحویل کارفرما گردد:

- سند مشخصات عمومی نرم‌افزار
- سند نقاط ضعف ذاتی پلتفرم‌ها و ابزارهای مورد استفاده در تولید نرم‌افزار
- سند حملات و تهدید های ناظر به نرم‌افزار
- سند آسیب پذیری‌های نرم‌افزار بر اساس آزمون های جعبه‌ی سیاه و علت آن‌ها
- سند راه‌حل های برطرف کردن یا کاهش تهدیدات و حملات
- سند مشخصات Performance عمومی نرم‌افزار
- سند مقایسه‌ی عملکرد نرم‌افزار در Browserهای مختلف