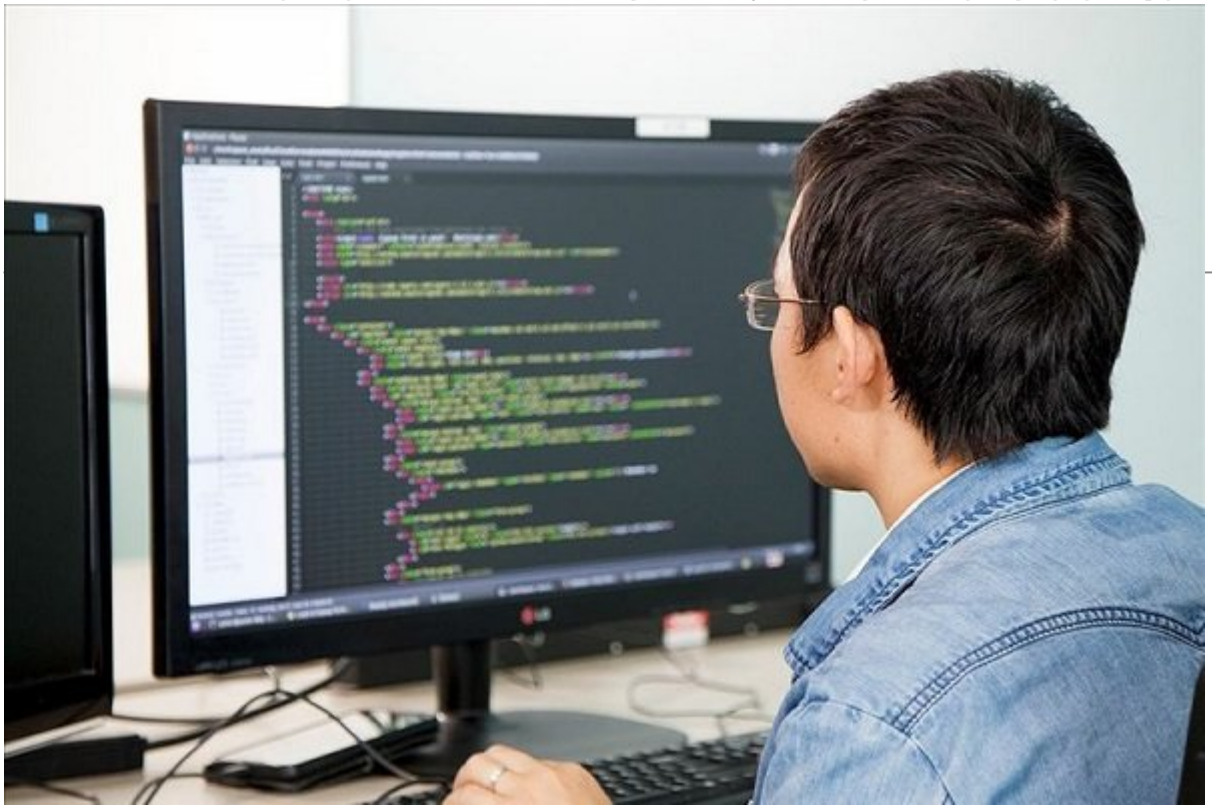
دو مفهوم در ظاهر یکسان، اما در باطن متفاوت

مهندسی نرم‌افزار و برنامه‌نویسی چه تفاوت‌هایی با یکدیگر دارند؟



در این مقاله قصد داریم به واکاوی این موضوع بپردازیم که مهندسان نرم‌افزار چه افرادی هستند، برنامه‌نویسان چه کسانی هستند و این دو گروه از متخصصان چه تفاوت‌هایی با یکدیگر دارند. جمله معروفی در این ارتباط وجود دارد که می‌گوید: «همه مهندسان نرم‌افزار می‌توانند برنامه‌نویسی کنند، اما همه برنامه‌نویسان نمی‌توانند یک نرم‌افزار را مهندسی کنند.»

واژه مهندس نرم‌افزار در ارتباط با چه افرادی مورد استفاده قرار می‌گیرد؟

اصطلاح **مهندس نرم‌افزار** در ارتباط با متخصصانی به کار گرفته می‌شود که قادر هستند نرم‌افزارهای با کیفیت بالا بنویسند. این گروه از متخصصان تنها به‌منظور کسب درآمد و امرار معاش به این حرفه نگاه نمی‌کنند. این حرف به معنای آن است که آگاهی شما از نحوه کدنویسی باعث نمی‌شود تا شما خود را یک **مهندس نرم‌افزار** توصیف کنید.

با توجه به این‌که برنامه‌نویسی فرآیندی پیچیده و دشوار نیست، یادگیری برنامه‌نویسی به گروه خاصی از افراد جامعه محدود نمی‌شود. در نتیجه هر شخصی می‌تواند برنامه‌نویسی را یاد بگیرد یا برای انجام برخی کارهای ساده برنامه‌های موردنیاز خود را بنویسد، اما زمانی‌که صحبت از نرم‌افزارهای جدی به میان می‌آید، به‌ویژه نرم‌افزارهایی که قرار است روی دستگاه‌های مختلفی به کار گرفته شود، موضوع فرق خواهد کرد. برای روشن شدن این موضوع اجازه دهید به شکل دقیق‌تری این موضوع را تحلیل کنیم. ما ریاضی و اصول نگارش را در مدرسه می‌آموزیم، اما این یادگیری باعث نمی‌شود تا همه ما ریاضیدان یا نویسنده شویم. بسیاری از ما به‌راحتی می‌توانیم به کلاس رفته و آشپزی یاد بگیریم، اما زمانی که مهمانی بزرگی می‌گیریم و در نظر داریم برای افراد زیادی غذا آماده کنیم، تصمیم می‌گیریم به سراغ یک آشپز حرفه‌ای برویم. شما هیچ‌گاه از یک کارگر ساده درخواست نمی‌کنید تا برای شما خانه‌ای را بسازد. تمثیل‌هایی که به آن‌ها اشاره شد، در مورد برنامه‌نویسی و مهندسی نرم‌افزار به‌خوبی صدق می‌کنند.

کدنویسی چیست؟

در این مقاله به شما می‌گوییم که نوشتن برنامه‌های ساده در مقایسه با نوشتن برنامه‌های مهندسی‌شده دو مفهوم کاملا متفاوت از یکدیگر هستند. خود مفهوم برنامه‌نویسی یا در اصطلاح رایج کدنویسی در ساده‌ترین تعریف خود به مجموعه دستورالعمل‌های کامپیوتری اشاره دارند که به کامپیوتر اعلام می‌دارند بر مبنای ورودی‌های مشخصی که دریافت می‌کند کاری را انجام داده و خروجی‌هایی را که مدنظر ما قرار دارد، ارائه کند. حرفه مهندسی نرم‌افزار در ارتباط با طراحی، نوشتن، آزمایش و نگهداری از برنامه‌های کامپیوتری به‌منظور حل مشکلاتی است که پیش روی کاربران قرار گرفته است. به عبارت دقیق‌تر، پیاده‌سازی راه‌حل‌هایی قدرتمند و ایمن که در شرایط مختلف بارها و بارها آزمایش‌شده و در نهایت تاییدیه مربوط را اخذ کرده‌اند. راه‌حل‌هایی که حتی برای حل برخی مشکلات ناشناخته دنیای واقعی نیز می‌توان از آن‌ها استفاده کرد. مهندسان نرم‌افزار اشراف کاملی روی مسائل دارند، جزییات مربوط

به یک مشکل را به‌درستی می‌دانند، به‌خوبی آگاه هستند که چه محدودیت‌هایی پیرامون راه‌حل‌ها وجود دارد، چالش‌های مربوط به حفظ حریم خصوصی و پیامدهای امنیتی عدم توجه به این چالش‌ها را به‌خوبی درک کرده‌اند. اگر شخصی نمی‌تواند نکته‌هایی را که به آن‌ها اشاره شد، به‌درستی درک کند، نباید به او اجازه داده شود تا برای حل یک مشکل راه‌حلی ارائه کرده و آن‌را پیاده‌سازی کند.

مطلب پیشنهادی

برنامه‌نویسان از شنیدن چه جملاتی بیزار هستند؟
کدنویسان با شنیدن این هفت جمله به سرعت عصبانی می‌شوند

داشتن مهارت ذهنی در حل مشکلات و ارائه راه‌حل‌ها

مهندسان نرم‌افزار به مهارت خود در قالب یک کار عادی همچون برنامه‌نویسی نگاه نمی‌کنند. برای یک **مهندس نرم‌افزار** هیچ‌چیز مهم‌تر از آن نیست که با ارائه راهکاری برای حل یک مشکل باعث خشنودی مردم شود. این طرز نگرش برای آن‌ها مهم است، به‌واسطه آن‌که هر مشکلی از طریق یک راهکار برنامه‌نویسی حل نمی‌شود. برخی از مشکلات می‌توانند از طریق به‌کارگیری برنامه‌های موجود یا از کنار هم قرار دادن و ترکیب چند برنامه مختلف حل شوند. حتی این ظرفیت وجود دارد تا از طریق انجام یکسری کارهای ساده، به‌طورکلی از بروز یکسری مسائل پیشگیری کرد. در نتیجه، طراحی یک برنامه خوب به معنای برنامه‌ریزی کردن به‌منظور پیشگیری از بروز مشکلاتی است که ممکن است در آینده رخ دهند. جمله معروفی از آلبرت انیشتین وجود دارد که می‌گوید افراد باهوش مشکلات را حل می‌کنند، درحالی‌که نوابغ مانع از آن می‌شوند تا مشکلات به وجود آیند.
در حالت کلی شما برای حل مشکلات پیچیده مجبور هستید برنامه‌های متعددی را بنویسید. برخی از مشکلات به برنامه‌هایی نیاز دارند که به شکل موازی اجرا شوند، درحالی‌که برخی دیگر از مشکلات به برنامه‌هایی نیاز دارند که به شکل پیوسته اجرا شوند. فراموش نکنید که برخی مشکلات از طریق آموزش کاربران یا کارمندان قابل‌حل بوده و نیازی نیست برنامه‌ای برای آن‌ها بنویسید. پیش از آن‌که فرآیند نوشتن یک برنامه را آغاز کنید، به‌عنوان یک **مهندس نرم‌افزار** این سوالات را از خود بپرسید:

- چه مشکلی وجود دارد که در نظر دارم آن‌را حل کنم؟
- چه کاری به غیر از کد نوشتن می‌تواند به حل مشکل کمک کند؟
- از چه راهکاری می‌توانم برای حل ساده‌تر مشکل استفاده کنم؟ به‌طوری‌که با ساده‌ترین کدنویسی مشکل حل شود؟

کیفیت کدها

برنامه‌های بزرگ و عالی از کدهایی کاملا خوانا برخوردار هستند. برنامه‌هایی که فرآیند توسعه آن‌ها به‌سادگی امکان‌پذیر است. این مدل برنامه‌ها به‌خوبی می‌توانند با سایر برنامه‌ها کار کرده و همچنین فرآیند نگهداری سورس‌کدها نیز به‌سادگی امکان‌پذیر است. به‌واسطه آن‌که مهندسان نرم‌افزار مجبور نیستند برای بازبینی و بررسی کدها ساعت‌های متمادی وقت صرف کنند. در نتیجه کیفیت کدها قابل‌مذاکره نبوده و نباید آن‌را با پول مقایسه کرد. فراموش نکنید در کدهای باکیفیت هیچ‌گاه از میان‌برها استفاده نمی‌شود و کدهای درهم‌برهم جایی در یک برنامه خوب ندارند؛ اما متاسفانه در برخی موارد زمانی که تیم‌های نرم‌افزاری احساس می‌کنند مهلت زمانی آن‌ها به پایان رسیده از چنین رویکردهایی استفاده می‌کنند. یکی از مهم‌ترین جنبه‌های مهندسی نرم‌افزار، طراحی نرم‌افزار از ابتدا تا زمانی است که نرم‌افزار آماده استقرار می‌شود. پس از استقرار نرم‌افزار طبیعی است که یکسری امکانات یا قابلیت‌ها باید تغییر پیدا کنند، به‌واسطه آن‌که کاربران به ویژگی‌های بیشتری نیاز داشته و به راه‌های ساده‌ای برای برقراری ارتباط با نرم‌افزاری که از آن استفاده می‌کنند، نیاز دارند. به‌طورمعمول، یک نرم‌افزار با حداقل امکانات خیلی مفید واقع نمی‌شود. یک نرم‌افزار زمانی که به ویژگی‌ها یا به عبارت دقیق‌تر به مولفه‌های مختلفی تجهیز شود و این مولفه‌ها به‌خوبی بتوانند با یکدیگر در ارتباط بوده و به کار گرفته شوند و فرآیند تبادل داده‌ها میان بخش‌های مختلف به شکل درستی انجام پذیرد و در نهایت از رابط کاربری ساده‌ای برخوردار باشد، به‌خوبی قابل‌استفاده خواهد بود.

مطلب پیشنهادی

تقابل علاقه و اجبار: بر اساس یافته‌های سایت استک‌اورفلو
10 زبانی که برنامه‌نویسان علاقه دارند و 15 زبانی که بیزارند

پرسش‌هایی که در زمان طراحی نرم‌افزار باید به آن‌ها پاسخ داد

در حالت کلی زمانی که تصمیم می‌گیرید برنامه‌ای را طراحی کنید باید بتوانید به پرسش‌های زیر به‌درستی پاسخ دهید:

- برنامه‌ای که در حال ساخت آن هستید، چه پیام‌هایی را دریافت خواهد کرد؟
- برنامه‌ای که در حال طراحی آن هستید، چه رویدادهایی را تحت نظارت قرار خواهد داد؟
- از چه پیام‌هایی باید صرف‌نظر شود؟
- چگونه می‌توانیم ارتباطات را تایید کنیم؟
- برنامه طراحی‌شده چه پیامی را به‌عنوان خروجی منتشر خواهد کرد؟

یکی دیگر از جنبه‌های حائز اهمیت برنامه‌های بزرگ در ارتباط با وضوح و شفافیت کدها است. در نتیجه تعداد آزمایش‌های انجام‌شده یا گزارش‌هایی که این آزمایش‌ها را پوشش داده‌اند، ملاک خوب بودن یک برنامه نیست. پرسش ساده‌ای که باید به آن پاسخ داده شود این است که آیا کدهای نوشته‌شده برای شخص دیگری نیز قابل فهم است؟ آیا کدنویسی که امروز این کدها را نوشته در چند هفته آینده نیز خواهد توانست منطق کدهایی را که خود آن‌ها را نوشته است، درک کند؟ فیل کارلتون می‌گوید: «در دنیای علوم کامپیوتر دو کار سخت وجود دارد. نام‌گذاری اشیا و متغیرها و دوم معتبر بودن مقدار کش شده.»
خوانا بودن کدها موضوعی است که بیش از آنچه تصور می‌کنید حائز اهمیت است؛ اما متاسفانه سنجه‌های درستی برای بررسی وضوح و خوانایی کدها وجود ندارد. در این میان به خاطر سپردن الگوهای نرم‌افزاری (Software patterns) و همچنین تمرین‌های عملی تا حدودی کمک می‌کنند تا چشم به دیدن کدهای شفاف عادت کند. مهندسان نرم‌افزار خبره، نوشتن کدهایی را که خوانایی بالایی دارند، به شکل تجربی و شهودی یاد گرفته‌اند. زمانی که در حال نوشتن برنامه‌ای هستید، ناخودآگاه مرتکب اشتباه‌هایی می‌شوید، اما ویژگی یک نرم‌افزار خوب در این است که اجازه می‌دهد به‌سرعت اشتباه‌ها را پیدا کرده و آن‌ها را برطرف کنید. خطاهایی که در برنامه‌ها به وجود می‌آید باید همراه با پیام‌های روشنی باشند و گزارش مربوط به خطاها نیز به شکل متمرکز در مکانی در سیستم ثبت شود که در ادامه تیم نگهداری و پشتیبانی بتوانند آن‌ها را موردبررسی و تحلیل قرار داده و خطاها را برطرف کنند.

محیط به‌کارگیری و آزمایش نرم‌افزار

زمانی که یک **مهندس نرم‌افزار** برنامه‌ای را می‌نویسد، باید از این موضوع اطمینان حاصل کند که برنامه او در محیط‌ها، دستگاه‌ها، ماشین‌هایی با منابع محدود یا ایده‌آل و در شرایط مختلف به‌خوبی کار خواهد کرد. نرم‌افزار نوشته‌شده باید این پتانسیل را داشته باشد تا روی دستگاه‌هایی با صفحه‌نمایش‌ها و جهات افقی و عمودی به‌خوبی کار کند. همچنین در صورت کمبود حافظه یا توان پردازشی بازهم بتواند همچون گذشته کار کند. در این حالت برنامه به‌طور خودکار نیازمند بهینه‌سازی فرآیندها و به حداقل رساندن یکسری عملیات است. به‌طور مثال، زمانی که برنامه‌ای را برای یک مرورگر وب می‌نویسید، این برنامه باید بتواند روی مرورگرهای مطرح و مدرن بدون مشکل اجرا شود. زمانی که یک برنامه دسکتاپ می‌نویسید، این برنامه باید بتواند در شرایط مختلف در اختیار کاربران ویندوز یا مک قرار داشته باشد. زمانی که در حال ساخت برنامه‌ای هستند که با داده‌ها سروکار دارد، این برنامه باید بتواند به روش‌های مختلف به سرور یا بانک اطلاعاتی متصل شده و فرآیند واکشی داده‌ها را انجام دهد. همچنین در زمان قطع بودن ارتباط باید بتواند از راهکارهای جایگزین استفاده کند. در زمان نوشتن یک قطعه نرم‌افزاری، مهندسان نرم‌افزار به همه سناریوهای احتمالی فکر کرده و در ادامه به آزمایش سناریوهای فرضی می‌پردازند. در این

مرحله مهندسان نرم‌افزار به بررسی مشکلات احتمالی می‌پردازند که ممکن است رخ دهد. در ادامه این مسائل فرضی را مکتوب کرده و برای هر کدام‌یک راه‌حل می‌نویسند. برخی از مهندسان نرم‌افزار این‌کار را با نوشتن کدها آغاز می‌کنند، رویکردی که در اصطلاح تخصصی به آن Test Case گفته می‌شود. در این روش آن‌ها به شبیه‌سازی سناریوهای مختلف می‌پردازند. این تکنیک به مهندسان نرم‌افزار اجازه می‌دهد، سناریوهای مختلف را آزمایش کرده و سناریویی را که نتیجه مطلوب از آن به‌دست‌آمده است، پیاده‌سازی عملیاتی کنند. مهندسان نرم‌افزار به‌خوبی می‌توانند ملزومات و کاستی‌های یک نرم‌افزار را درک کنند؛ اما توجه داشته باشید که مهارت منحصربه‌فرد یک **مهندس نرم‌افزار** در ارتباط با نوشتن یک راه‌حل نیست، بلکه در ارتباط با شناسایی ملزوماتی است که باید درون یک راه‌حل قرار گیرند.

هزینه‌ها و عملکرد

مهندسان نرم‌افزار در اغلب موارد می‌توانند مشکلات را به‌سرعت حل کنند. اگر جزء آن گروه از افرادی هستید که تصور می‌کنید استخدام برنامه‌نویسان متبحر به معنای افزایش هزینه‌ها است، بهتر است در فکر خود تجدیدنظر کنید. زمانی‌که یک توسعه‌دهنده مجرب را استخدام می‌کنید، این فرد قادر است در مدت‌زمان کوتاهی راهکاری قدرتمند، قابل‌اطمینان و پایدار را ارائه کند. رویکردی که در طولانی‌مدت باعث کاهش هزینه‌های جانبی شرکت می‌شود. نکته قابل توجه دیگر در ارتباط با هزینه اجرای نرم‌افزار است. هر برنامه‌ای برای اجرا به منابع سیستمی نیاز دارد که این منابع در اغلب موارد آزاد نبوده و در اختیار برخی از برنامه‌ها قرار دارد. مهندسان نرم‌افزار می‌توانند نرم‌افزارها را به شکلی کارآمد پیاده‌سازی کنند که از منابع سیستمی بیهوده استفاده نکنند. به‌طور مثال، ذخیره‌سازی داده‌ها در حافظه کش از جمله استراتژی‌هایی است که در چنین شرایطی عالی عمل می‌کند؛ اما توجه داشته باشید که استراتژی فوق یکی از هزاران راهکاری است که برای اجرای سریع‌تر و کارآمدتر برنامه‌ها در اختیار شما قرار دارد. یک توسعه‌دهنده تازه‌کار ممکن است راه‌حلی کم‌ارزش به شما ارائه کند، راه‌حلی که ممکن است هزینه‌های زیادی را هم به شما و هم به مشتریان شرکت شما تحمیل کند. درحالی‌که یک توسعه‌دهنده مجرب ممکن است مقرون به‌صرفه‌ترین راهکار را به شما پیشنهاد دهد.

قابلیت استفاده

برنامه‌های خوب با در نظر گرفتن تجربه کاربری طراحی و پیاده‌سازی می‌شوند. شیوه تعامل انسان با سامانه‌های کامپیوتری از جمله مباحثی است که تاکنون پژوهش‌های متعددی در ارتباط با آن انجام‌شده و دستاوردهای جالب‌توجهی نیز در این ارتباط به‌دست‌آمده است. این دستاوردها باعث بهبود عملکرد نرم‌افزارهای امروزی شده است. اجازه دهید برای روشن‌تر شدن مطلب به مثال‌هایی در این زمینه اشاره کنیم.
زمانی که فرم‌های ورودی را به‌منظور دریافت ورودی‌ها از کاربران، همچون آدرس‌های ایمیل طراحی می‌کنید باید به چند نکته دقت کنید: اول آن‌که ورودی‌های مربوط به آدرس‌های ایمیل نباید به حروف کوچک و بزرگ حساس باشند. اگر برنامه‌ای آدرس ایمیل جدید کاربر را قبول کرده و فرآیند اعتبارسنجی را انجام داد باید پیام روشنی در ارتباط با معتبر بودن یا اشتباه بودن آدرس ایمیل به کاربر نشان دهد. به‌طور مثال کاربر ممکن است علامت @ را فراموش کرده باشد یا پسوند مربوط به آدرس ایمیل را به‌اشتباه ocm. وارد کرده باشد.
زمانی که کاربر روی پیوندی کلیک کرده و فرآیند انتقال (هدایت) کاربر به صفحه دیگری انجام می‌شود، یک برنامه خوب به‌طور خودکار پیوند اصلی کاربر را ذخیره می‌کند تا زمانی که کار وی به سرانجام برسد، دومرتبه به مکان اولیه بازمی‌گردد. یک برنامه ایده‌آل به‌خوبی می‌تواند اطلاعات و تعاملات گذشته انجام شده میان برنامه و کاربر را در خود ذخیره کرده باشد تا در صورت نیاز کاربر بتواند به وضعیت قبلی خود باز گردد.
از واژه خوب تنها زمانی می‌توانیم در ارتباط با یک برنامه استفاده کنیم که در زمان طراحی، مهندس نرم‌افزار خود را به‌جای کاربر قرار داده باشد. در چنین شرایطی مهندس نرم‌افزار سعی کرده حالات مختلف و نحوه تعامل کاربر با برنامه را از دید کاربر آزمایش کند. در نتیجه مهندس نرم‌افزار تنها به دنبال اضافه کردن یکسری ویژگی‌ها به برنامه کاربردی نیست.

قابلیت اطمینان، امنیت و ایمنی

این سه فاکتور باعث می‌شوند تا مهندسان حرفه‌ای از مهندسان تازه‌کار متمایز شوند. مهندسان نرم‌افزار در زمان

ارائه راه‌حل‌های خود از ایمن‌بودن آن‌ها آگاهی کامل دارند. به‌طور مثال، یک برنامه باید در ارتباط با ورودی‌های مخرب واکنش درستی از خود نشان دهد. توجه به چنین فاکتوری به‌دقت و تجربه بالایی نیاز دارد. عدم توجه به این فاکتور مهم باعث شده به‌طور مرتب در ارتباط با نقض حریم خصوصی یا افشای اطلاعات اخبار مختلفی در رسانه‌ها بخوانیم. برخی از کاربران سعی می‌کنند یک نرم‌افزار را با ورودی‌های اشتباه یا بد آزمایش کنند. برخی از این کاربران سعی می‌کنند از طریق ورودی‌های مخرب بانک‌های اطلاعاتی یک نرم‌افزار را خراب کنند. ضروری است چنین سناریوهایی مورد توجه قرار گرفته باشند تا ورودی‌های مخرب به داده‌ها آسیب وارد نکنند. البته این سناریو تنها به دستورات مخرب یا هکرها محدود نیست. فراموش کردن گذرواژه‌ها از سوی کاربران، تعداد دفعاتی که می‌توانند دومرتبه گذرواژه را وارد کنند، قفل شدن فرآیند وارد کردن گذرواژه پس از چند مرتبه اشتباه وارد کردن گذرواژه‌ها، هک شدن یک حساب کاربری از سوی یک هکر، به‌کارگیری گذرواژه‌ها روی یک ارتباط غیر ایمن، ورود به یک حساب کاربری از یک موقعیت مکانی که متفاوت از موقعیت قبلی بوده است، لاگین کردن خودکار، محافظت از کاربران در برابر یک حمله منع سرویس توزیع‌شده، مقابله با حملات XSS، حمله مرد میانی، فیشینگ اجتماعی، بازیابی داده‌ها در زمان بروز یک حمله و... تنها چند سناریو ساده در این ارتباط هستند. برنامه‌های ایمن هیچ‌گاه اطلاعات حساس را در قالب یک متن ساده ذخیره‌سازی نمی‌کنند، بلکه از مکانیزم‌های رمزگذاری داده‌ها با بهترین الگوریتم‌های موجود استفاده می‌کنند. راهبرد پشتیبان‌گیری از داده‌ها باعث می‌شود تا خطر از دست رفتن داده‌ها به حداقل برسد.
فراموش نکنید که همواره ممکن است مشکلات پیش‌بینی‌نشده‌ای برای برنامه‌ها به وجود آید. اگر فردی هستید که از وجود چنین مشکلاتی آگاه نبوده‌اید، باید بدانید که یک مهندس نرم‌افزار نیستید، بلکه تنها یک برنامه‌نویس هستید که یک برنامه غیرایمن را تولید می‌کند. نقض‌های امنیتی موضوعی نیست که به‌راحتی بتوان از آن گذشت. به همین دلیل است که مهندسان نرم‌افزار از سرویس‌ها و ابزارهای خوبی که در این زمینه وجود دارد برای شناخت بهتر مسائل امنیتی استفاده می‌کنند.

به‌کارگیری ابزارهای مختلف

جای هیچ‌گونه تردیدی وجود ندارد که مهندسان نرم‌افزار برای حل درست مسائل به ابزارهای بهتر و بیشتری نیاز دارند. به‌کارگیری ابزارهای مختلف باعث می‌شود تا بسیاری از مشکلات شناسایی شوند. ابزارهایی شبیه به Chrome DevTools به درک بهتر مشکلات کمک فراوانی می‌کنند. هرچه با ابزارهای تحلیلی بیشتری آشنا شوید به همان نسبت خروجی کار شما دقیق‌تر خواهد بود. نکته دیگری که باید به آن دقت کنید انتخاب درست یک زبان برنامه‌نویسی است.

مطلب پیشنهادی

یادگیری الگوی طراحی، انتخاب یا ضرورت
چند راهکار برای حل مشکلات رایج برنامه‌نویسی

تبدیل‌شدن به یک مهندس نرم‌افزار مجرب به زمان نیاز دارد

هیچ‌کس نمی‌تواند با گذشت شش ماه یا حتی یک سال به یک مهندس نرم‌افزار واقعی تبدیل شود. حتی حضور در کلاس‌های آموزشی برنامه‌نویسی به شیوه فیزیکی یا آنلاین نیز باعث نمی‌شوند شما یک مهندس نرم‌افزار قدرتمند شوید. یک مهندس نرم‌افزار واقعی فردی است که سال‌ها در حوزه طراحی و ساخت نرم‌افزارها تجربه اندوخته است. باوجود این، این فرد حتی با گذشت سال‌های متمادی بازهم به دنبال یادگیری است. شما تنها از طریق یادگیری، پیاده‌سازی و تحلیل برنامه‌هایی که از سوی طیف گسترده‌ای از کاربران به کار گرفته‌شده است، قادر خواهید بود به یک توسعه‌دهنده حرفه‌ای تبدیل شوید. در این میان نرم‌افزارهای متن‌باز به پیشرفت کاری شما کمک فراوانی می‌کنند. واقعیت این است که مسائل روزبه‌روز برای مهندسان نرم‌افزار پیچیده‌تر می‌شوند. دورنمای آینده نشان می‌دهد که اغلب کاربران عادی در سال‌های آتی خواهند توانست برای حل مشکلات ساده خود راهکارهایی را ارائه کرده و از کامپیوترهای شخصی خود به نحو مطلوب استفاده کنند. بدون آن‌که به سال‌ها تحقیق و آموزش نیازی داشته باشند. این کاربران از طریق به‌کارگیری ابزارهای کوچک قادر خواهند بود بر مشکلات خود چیره شوند. در چنین شرایطی این وظیفه مهندسان نرم‌افزار خواهد بود که ابزارهای بهتری را برای حل مشکلات تولید کنند. مهندسانی که سعی خواهند کرد مانع به وجود آمدن مشکلات ناخواسته شوند.

تاریخ انتشار:

16 مرداد 1397

نشانی منبع:
https://www.shabakeh-mag.com/workshop/13609/%D9%85%D9%87%D9%86%D8%AF%D8%B3%DB%8C-%D9%86%D8%B1%D9%85%E2%80%8C%D8%A7%D9%81%D8%B2%D8%A7%D8%B1-%D9%88-%D8%A8%D8%B1%D9%86%D8%A7%D9%85%D9%87%E2%80%8C%D9%86%D9%88%DB%8C%D8%B3%DB%8C-%DA%86%D9%87-%D8%AA%D9%81%D8%A7%D9%88%D8%AA%E2%80%8C%D9%87%D8%A7%DB%8C%DB%8C-%D8%A8%D8%A7-%DB%8C%DA%A9%D8%AF%DB%8C%DA%AF%D8%B1-%D8%AF%D8%A7%D8%B1%D9%86%D8%AF%D8%9F