1- در حوزه ی امنیت نرم افزار، مرور کردن کد به چه معناست؟

در چارچوب باگ‌ها و رخنه‌ها، مرور کردن کد به معنی شناسایی و درست کردن باگ‌ها میباشد.

2- ابزارهای آنالیز ایستا چیست و چگونه کار میکند؟

ابزارهای آنالیز ایستا (که با نام آنالیز کد منبع هم شناخته میشوند) متن یک برنامه را به صورت ایستا بررسی میکند بدون این که آن را اجرا کند. به صورت تئوری شما میتوانید کد منبع یک برنامه و یا مدل کامپایل شده ی برنامه را بررسی کند، اگرچه کدگشایی کردن مورد دوم میتواند سخت باشد.

3- به چه دلیل ابزارهای حمله نمیتوانند برای بررسی سلامت کد کافی باشند ؟

به دلیل اینکه حمله کننده ها نیاز دارند تنها یک مشکل را پیدا کنند، (و یک سوء استفاده برای آن بنویسند) و مدافعان نیاز دارند که همه ی مشکلات را پیدا کنند (و آن ها را برطرف کننده و یا در مقابل حملات از آنها دفاع کنند)، ابزارهای حمله کننده میتواند از دقت کافی برخوردار نباشد ولی همچنان مفید است.

4- در ابزارهای آنالیز ایستا و در کل ابزارهای آنالیز امنیتی، نتایج استنتاج شده به 4 گروه دسته بندی میشوند. کدام یک از آنها خطرناک تر هستند و چرا؟

ابزارهای آنالیز ایستا از false negative رنج میبرند (که در آن برنامه دارای باگ‌هایی است که ابزار آن‌ها را گزارش نمیکند) و همچنین false positive (که در آن ابزار باگ‌هایی را گزارش میکند که واقعا درون برنامه وجود ندارد). false positive باعث شکایت سریع هر آنالیزکننده‌ای میشود که باید آنها را وارسی کند ولی false negative ها خطرناک‌تر هستند زیرا منجر به برداشت غلطی از امنیت میشوند.

5- اولین جستجوگر کد که برای پیدا کردن مشکلات امنیتی درون کد ها ساخته شده بود چه بود و چگونه کار میکرد؟

ITS4 بود که در واقع مخفف عبارت "It's The Software Stupid Security Scanner" میباشد. ITS4 و همتاهای آن RATS و FlawFinder به شدت ساده هستند. این ابزارها از روی یک فایل به صورت کلمه به کلمه عملیات جستجو را انجام میدهند و به دنبال تطبیق قواعدی میگردند که بر اساس تعدادی قانون ساده عمل میکنند که میتوانند آسیب پذیری‌های امنیتی ممکن را مشخص کنند.

6- کدام یک از ابزارهای لینوکس و یونیکس میتوانند به عنوان ابزاری برای آنالیز ایستای کد استفاده بشوند؟

احتمالا ساده ترین و سر راست ترین دیدگاه برای آنالیزهای ایستا، ابزار grep موجود در یونیکس میباشد. عملکردی مشابه ابزارهایی که قبلا در ابزارهایی همانند ITS4 پیاده سازی شده بود. با مجهز شدن توسط رشته های جستجوی خوب، grep میتواند اطلاعات مفیدی از کد نمایش دهد.

7- نقطه ضعف ابزارهایی مثل grep و در کل آنالیزهای واژه ای چیست؟

نقطه ضعف grep و در کل آنالیزهای واژه‌ای این است که تعداد زیادی false positives تولید میکنند زیرا تلاشی برای در نظر گرفتن کد هدف از نظر معناشناسی صورت نمیدهند. توضیحات، لفظ‌های رشته‌ای، تعریفات و توابع فراخوانی شده تماما جریانی از کاراکترها هستند که میتوانند برای تطبیق مورد مقایسه قرار گیرند. برای مثال موارد زیر همگی توسط grep به عنوان آسیب پذیری شناخته میشود:

- یک فراخوانی از تابعی آسیب‌پذیر: `gets(&buf);`
- یک توضیح: `/* never ever call gets */`
- و یک متغیر بی گناه و بی ربط: `int begetsNextChild = 0;`

8- برای افزایش دقت ابزارهای آنالیز ایستا به چه چیز نیاز میباشد ؟

برای افزایش دقت، یک ابزار تحلیل ایستا نیازمند استفاده ی بیشتری از تکنولوژی‌های کامپایلر میباشد. با ایجاد یک abstract syntax tree (AST) از کد منبع، چنین ابزاری میتواند معانی پایه‌ی برنامه‌ای که مورد بررسی قرار گرفته است را در نظر بگیرد.

9- انواع آنالیزها را از نظر حوزه‌ی مورد بررسی نام ببرید ؟

با مجهز شدن توسط AST تصمیم بعدی که باید گرفته شود مربوط به حوزه‌ی آنالیز میباشد. آنالیزهای محلی برنامه را به صورت یک تابع در هر زمان مورد بررسی قرار میدهند و رابطه ی بین توابع را در نظر نمیگیرند. آنالیز های سطح ماژول، در هر زمان یک کلاس یا واحد کامپایل را در نظر میگیرند بنابراین روابط بین توابع در یک ماژول یکسان و خصوصیاتی که به کلاس ها اعمال میشود را مد نظر قرار میدهند ولی فراخوانی بین ماژول ها را آنالیز نمیکنند. آنالیز های سراسری شامل آنالیز کل برنامه میباشد و بنابراین روابط بین همه ی توابع را در نظر میگیرد.

10- برای اینکه یک ابزار آنالیز کد منبع، مفید و مؤثر باشد باید چه ویژگی‌هایی داشته باشد ؟

باید دارای 6 ویژگی باشد:

- به منظور امنیت طراحی شده باشد.
- از چندین زبان و پلتفرم پشتیبانی کند.
- توسعه پذیر باشد.
- هم برای آنالیزکننده های امنیتی و هم برای توسعه دهندگاه به طور یکسان مفید باشد.
- از فرایندهای توسعه و IDE های موجود پشتیبانی کند.
- برای ذینفعان و سازمان‌ها قابل درک و محسوس باشد.