Epic List of Interview Questions

< Prev. Next >

June 14, 2013 interviewing 8 Comments

When it comes to interviewing having a good list of questions to share with your team can be invaluable. It helps everyone prepare a little easier; plus being aware of what other people are asking is a great way to ensure that you cover a diverse set of areas (giving you a better view into an imperfect process, and the candidate a better chance to get things right).

This is the list I used to use (largely for software engineers and web developers, but many are also good for other roles like PMs, etc.).

There is also a link to a google doc of the questions here – that way you can make a copy and tailor it to your team. Suggestions, comments, links, etc. are welcome!

Enjoy!

Epic List of Software Interview Questions

Below is a list of software engineer skills or areas that can be tested and evaluated in an interview context. Use the list below to help you get started, and you'll soon be developing your own favorite questions and competencies for interviews. There are so many creative ways to test and evaluate skills when building your team – so let's get started!

- **Abstraction & Design**
  - Have you ever seen OO go bad? What happened? What elements of OO design are most prone to abuse and misuse? What are some ways to prevent these mishaps?
  - Implement a game of tic-tac-toe. How do you represent the game board? What interfaces do you expose?
  - Implement a stock ticker. How do you handle displaying all of the data quickly to the end user? // Since most people are familiar with stocks I generally have them articulate the requirements and functionality first. The crux of this problem is thinking about the different granularities in which they can view a stock price – real time, hourly, daily, yearly, etc. and modeling their data to support fast querying for those graphs.
  - You are tasked with designing software that runs and controls elevators. What interfaces and class objects would you use? What configuration options would you need for the software to work in skyscrapers, buildings with only one elevator, and buildings with banks of elevators? How would these use cases change the objects and interfaces in your design?
  - Imagine you were tasked with designing a text editor (or instant messaging program). What are the primary functions? What are the various interfaces, classes, etc. that you would need to provide those functions? // and feel free to replace text editor with any common small program of your choosing (mobile apps make great examples)

my greatest hits
Looking for a place to start? Here are some of my most popular posts:

My Swan Story - a journey of self improvement, discovery, and success

The Paradox of Autonomy and Recognition

Lean Software Development – build v1s and v2s

Distributed Systems Basics – Handling Failure: Fault Tolerance and Monitoring

What Every Programmer Should Know About SEO

topics

being awesome
business
career
communication
entrepreneur
good stuff
personal
productivity
rants
self improvement

- **Algorithms (different approaches and performance)**
  - Describe the algorithm for a depth-first graph traversal. // It is also useful to have the candidate tell you about the data structure they are using to represent the graph.
  - Write a function that takes two strings as arguments and returns a string containing only the characters found in both strings. Have them write 2 versions – one that is O(n) and one that is O(n^2).
  - You are given a sorted array and you want to find the number N. How do you do the search as quickly as possible (not just traversing each element)?
    - How would the performance of your algorithm change if there were lots of duplicates in the array?

  - Given a list of numbers that is a circular list, such that iterating through the list would return the first number after you reached the end. How might you find the minimum number in the list? // It is easier if you can assume all the numbers are increasing, until it reaches the end of the list.Implement a function that will divide two numbers without using the division operator. (solution for dividing a number by 3)
    - Now find any given number in the list. What is the upper bound on for the running time of this function?

  - Given two different lists of objects, come up with an efficient solution to find the intersection of the two lists. (solution)
  - How do you find all the permutations of a string? What is the running time?
    - Now imagine that the string has repeating characters. How could you modify your solution so it would only find unique permutations as efficiently as possible?

  - Given a list of numbers, assume each number represents the amount of time it takes to execute a task. How would you dive the tasks across two different servers to they finished in the same amount of time?Write a program that will find the 10 most popular words (popularity is determined by how often they occur) in a file. How can you do this efficiently in terms of space? In terms of time? // Sometimes it helps to tell them it is a really big file that can't fit in memory if they get hung up on timing
    - How would you divide them across N servers?
    - Now assume you want the jobs to finish at the same time, what is the optimal number of servers and how would you distribute the jobs across them to achieve this goal?

  - Given an array what is the longest contiguous increasing subsequence of elements? (solution) What is the longest increasing subsequence? (solution) (This is a classic dynamic programming problem)
  - Imagine that you want to return a random number from a really long linked list of numbers. You do not know the length of it, but it is big. Write a function which will return a random number from the list. (solution)
  - Create an algorithm that will output the results of rolling a die (1-6) using a function that simulates a coin toss (1 or 2). All 6 outcomes should be equally likely. (solution)
  - Given an array of integers write a program that will determine if any two numbers add up to a specified number N. Do this without using hash tables. (solution, although some use hash tables)

- **Problem Solving (like algorithms, only more general)**
  - If there are N different teams, and over time they are all going to merge into one giant team. How many different ways can the teams merge?
  - Given a dollar value come up with all the different ways to make change given the standard coin denominations (solution).
  - Count the 2's between 0 and N. And every 2 digit counts as a 2, so if N was 7 the answer would be 1 (just the number 2), whereas if n was 23 there would be 7 2's {2, 12, 20, 21, 22 (this counts for 2), 23}. // This is actually pretty challenging and one of my favorite questions to ask candidates that are doing really well in with lots of other questions. Getting the brute force answer is easy, coming up with an elegant solution takes a little more effort.
  - How many degrees are there in the angle between the hour and minute hands of a clock when the time 4:15? // You can also use the current time if there is a clock in the room. You aren't looking for a wild

technology

search this site

Search

follow along via RSS

RSS - Posts

guess but for someone who can rationalize out the answer. For candidates rusty on geometry I will give them the number of degrees in a circle via hints

- Imagine that you have a large dataset on disk (or in the cloud like S3), and you have limited memory on your servers but you want to sort and manipulate the data. What is the best algorithm/sort to use in this situation? (solution)
- We want to open up another office in a different state. How would you pick the state? How would you go about deciding what salaries to pay the employees? // This is a great question for a leadership role, if you don't like the salary version you can modify the question to be about opening a data center and partitioning services.

Here are some other problem solving questions too, if you need additional examples.

- **Data Structures: Trees & Graphs**
  - Given a tree write breadth first search and depth first search, and explain the run time and space requirements.Convert a binary search tree into an ordered array. How do you do this as efficiently as possible?
    - Now modify your solutions to handle trees with weighted edges and/or loops and print out the path from start to finish.
  -
  - How do you find the 7th element in a binary search tree? How do you generalize the function to find the Nth element? (solution)
  - Given two binary trees write a function that will compare the two and see if they are equal – both in terms of data and structure. (solution) // You can also do just data or just structure to mix it up. Just make sure you know how to explain the differences.
  - Write a function to determine if a graph contains a cycle. (solution)
  - Give at least two ways to represent a graph in memory. What are the pros and cons of each solution? When would you choose one over another? (solution)

- **Data Structures: Queues & Stacks**
  - Design and implement a stack. Implement the different methods: push, pop, retrieve the minimum element in constant time. (solution)
  - Design a queue using stacks as the underlying data structure (solution). Implement a stack using queues as the underlying data structure (solution).

- **Data Structures: Hash Tables**
  - How do hash tables work?What are some examples of real life hash tables? When is a hash table a poor data choice?
    - What are different ways of managing collisions?
    - Implement a hash table.
  - Find the first non-repeated character in a string.
  - Implement a spell checker. What are the interfaces you would expose?
    - What are some ways you could come up with alternative words to suggest?
    - Write the code that would do the checking on the document.

- **Bits & Bytes**
  - How much space would you need to store 1 billion phone numbers?
  - Convert a binary (or hex, or any other base of your choosing) string to an integer. Convert an integer to binary (or hex, or any other base of your choosing).
  - If you had a product catalog of 1 million items how much space would you need to store all of them? Assume each item has a title, a description and price. // feel free to set limits on the fields or your choosing, or ask the candidate to pick something reasonable.

    - (There are a ton more of these types of questions online if you want to find more of them. Most of the time I am looking to verify the candidate understands the different sizes of data and how it would be impact their programs. Other interviewers go deeper here, though, so it is up to you.)

- **Command Line & Scripting**
    - Design an API to provide product data from a supplier to an ecommerce website. What functions do you support? What do the interfaces look like?
    - Write a regular expression which matches a email address // You can use url, phone number, etc. instead of email address
    - How would you find all the *.plist files in a directory via the command line?
    - What is a shell What kind of shell do you use? Have you customized it all? If yes, what are some of your customizations?
    - How do you copy a file from one system to another?
    - How do you view all the processes running on a Linux system? When might you want to do this?

- **Database Administration**
    - How can you prevent your DBA (or anyone else really) from obtaining a list of your customers' passwords?
    - Imagine your database is having performance issues, what are some things you might consider to speed it up? // Also pay attention to what questions the candidate asks you about the database – hopefully the questions will help clarify assumptions and problems and will likely give you more insight than just the potential solutions.

- **Data Modeling & Data Schemas**
    - Design a data schema for [insert example here]. // I like to use one of the products or features at my company, but you could also use familiar scenarios like: course catalog for students, a rental car database, a flight database for an airline, inventory for an ecommerce site, etc.
    - Have you ever had to design a data model from scratch? What project was it? Was there ever any issues? What do you think you got really right, and what could have been improved?

- **SQL Queries (querying for data)**
    - What is a primary key?
    - You know that data is in a particular table but you don't know what the schema is for that table. How could you figure it out?
    - What are some different kinds of joins? How do they work? Can you give examples?
    - What is the difference between GROUP BY and ORDER BY?
    - Write a query to delete duplicate rows from a table.
    - Describe a schema and ask the candidate to query the table in lots of different ways. Try to pick a schema from one of your databases if possible – it is always nice to use real world data. // If you can't come up with one an employee or student database are easy to grok examples.
    - For more SQL questions, check out Jitbit's guide here.

- **Networking**
    - What are some common networking protocols and what makes them special? Any of the following will work: TCP, UDP, HTTP, DNS, but there are way more. // In the event the candidate starts to explain one you are less familiar with take notes and ask questions – you can verify their knowledge later and test their communication now.
    - A customer complains your website is slow. How do you troubleshoot the issue?
    - Describe what happens when I type "google.com" into a browser and hit return. Be as detailed as possible.
    - How does TCP handle congestion? How can this impact the performance of applications that communicate across the network?
    - A member of your team comes running up and says that your website is under a Denial of Service (DoS) attack. How would you verify if this is the case? What could you do to stop it or mitigate the issue?

- **System Design & Thinking**
  - What is the difference between a 2-tier system and 3-tier system?
  - What are some alternate ways to store data besides using a relational database? Do you have any experience with other data stores? What was the use case? Why might you consider an alternate solution (to an relational DB) and what would be the downsides of doing so?
  - What is the difference between stateless and stateful systems? How does this impact scaling?
  - What is a cache? In your past projects, what types of caches were present?How would you decide if you should buy servers with more memory or disk space? How would you develop a cost model to help you make the decision? // This question is probably best for someone with some experience managing or having exposure to servers. You can still answer it without that knowledge of course, but there may be better questions.
    - If you had to implement a cache (feel free to pick a type of cache, such as an image cache, or type of data) how would you do it? How do you know which items need to be refreshed? What happens when the cache is full; how do you decide which items to evict? // BTW this is one of my favorite interview questions for phone screens. Mostly because you can go really deep with candidates that really understand this concept and you don't need a diagram or whiteboard to communicate the question and answer.
    - What are different cache eviction strategies? // such as LRU, LFU, FIFO etc.
  - A website with 3 app servers and one database is slow, what are some ways to troubleshoot this system?
  - What are some ways to scale a read-heavy application? How about a write-heavy application?

- **APIs**
  - Why are interfaces important to software and systems? What are some examples of "interfaces" you have built or used in a previous project?
  - Describe a situation where you had to use a RESTful web service? What were the languages and technologies that you used? Did you learn anything from the experience?
  - What are some qualities of a well designed API? How about a poorly designed one? // Feel free to replace the example with anything that involves data, hopefully something that relates to your product and business.
  - How do you keep APIs secure? What are some considerations with API security?

- **Web Development**
  - JavaScript
    - What JavaScript libraries (or frameworks if you would prefer) have you used?
    - Explain AJAX in as much detail as possible. How does it work? What have you used it for in the past?
    - You want to get a query string parameter from the browser's URL, how would you do it?
    - What is the difference between document load and document ready events?
    - What are ways to write object oriented JavaScript? For example, explain how inheritance works.
  - HTML
    - What are some of the building blocks of HTML5?
    - What is the difference between cookies, sessionStorage and localStorage?
  - CSS
    - How do you organize CSS files? What are the pros and cons of this approach? Have you ever tried other ways?
    - How do you avoid duplicating colors or fonts in CSS, when those colors or fonts are applied to multiple elements? What are the pros and cons of that approach?
    - What is the CSS box model? (A: width/padding/border/margin)
    - What are some clearing techniques and when is it appropriate to use them?

        - What is the difference between "visibility:hidden" and "display:none"? What are the pros and cons of using "display:none"?
        - How does the browser determine where to place positioned elements?

    - Web BrowsersWhat is responsive design? What is the difference between fixed and fluid layouts? What are some of the pros and cons with these designs?
        - How do you do browser compatibility testing?
        - What are some ways to prevent web browser caching?
        - What is your favorite browser? What sort of tools do you use to debug websites?

    - What are some considerations in selecting font sizes? // This question is focused on accessibility
    - How does the DOM work? Explain in as much detail as possible.
    - What do you think of "hacks"? When should they be used in your code and when should they be avoided?
    - What is MVC and why is it useful? When would MVC not be an appropriate design pattern choice?
    - What are the advantages of client side rendering vs. server side rendering? If you were building our site which would you use and why?
    - What does minification do?
    - What are some ways to make websites faster? Name as many different techniques as you can.
    - How do you test the performance of your code and/or web pages?
    - What are some common security issues with web applications and how do you avoid them?
    - What is the difference between Canvas and SVG? Do you have experience with either?

- **Reliability & Operations**
    - Have you ever had to be on-call? How did it work? Did you ever miss an alert? How often did you get paged when on-call?
    - Have you ever had someone let you down at work? What happened? How did you handle it? What did you learn and did it change the way you did things?
    - Can you describe a stressful situation from a previous role. What you did to create a positive outcome? How do you manage stress in your daily work?
    - Have you ever had a bug in your code that showed up in production? What happened? What did you learn from the experience?
    - Tell me about a time when you had to go above and beyond the call of duty to complete your work.
    - Tell me about a situation when you aren't able to complete you work. What happened? What did you learn from the situation and experience?
    - operations and best practices
    - Tell me a story about the "best" outage you have ever been a part of. What made it the best and what was your role?
    - What is the purpose of post mortems, and why are they useful? What are the attributes that make a great post mortem?

- **Software Engineering**
    - How would you describe the software lifecycle at your last position? What did you like about it? What did you wish you could change?
    - What is an example of a sandbox? Have you ever used one? What is the purpose of one and what are some potential alternatives?
    - What does refactoring mean to you? Why is it important and when have you done it? (Some candidates can even talk about different refactorings and design patterns here, but mostly I am looking for someone who wants to improve the code they write – as very few people ever get it all right the first time)
    - Given the following variables: time, budget, customer happiness and best practices which are most important in a project? Give them an order and explain why.
    - What is the advantages of best practices like continuous integration, automated testing, and code reviews? What are the disadvantages of these practices? (Feel free to insert your own best practices

too, these are just examples.)
  - Are you familiar with the concept of convention over configuration? What is an example?
  - How do you design, develop and debug applications? What tools do you like to use best? Have you tried others before? What were the reasons you use the ones you do? // There is really no right answer here, it is mostly just about learning how they do their work, and how it fits into past projects. Good answers usually involve thoughtful responses on trade-offs, technology, and experience.

- **Teamwork & Collaboration**
  - Give examples of project that were completed as a team. Were there any that went better than others? Why? What was different?
  - What is the best way to collaborate on a coding project?
  - Have you ever had to deal with features that involved multiple people working in the same areas? How did it go? Was there anything that could have been done to improve it?
  - Do you do your best work alone or in a group? Does the type of work matter?
  - What does it mean to be a good teammate? Have you ever had any bad teammates? If so, did you tell them and give the feedback?
  - Have you ever had to work with someone else that didn't pull their weight on a project? How did you handle it? Did things ever improve? If you had to do it all over again would you change anything?
  - We are in the middle of a development cycle (or sprint if you use agile) and there is a major change in the functionality of a feature you have been working on. How do you respond? What questions do you ask?
  - Have you ever worked on any open source projects? If so, what were some of the issues within the project? If you haven't worked on any projects what do you foresee as potential issues?

- **Product Sense & Judgement**
  - What do you dislike about our website/product/service? How would you improve it? Which of those changes would likely have the biggest customer benefit?
  - What is a really well designed website that you use? What makes it great? // This is also a great question to ask if you have a web browser and can pull it up and have the candidate walk you through it
  - Describe the structure and contents of a design document. What do you consider the minimum amount of information for development to start building something? Give an example of when you didn't have enough information, or when you had too much/unnecessary information.
  - What do you think makes [insert product here – ideally one they would have heard of] successful? What is the appeal? What made it special?
  - Create a scenario and design a product. // I have heard everything from an alarm clock for the blind, a parking lot, internal reporting tools, a slot machine, replacement for Google Maps, and more; the possibilities are endless. If I ask this question though, I always try to pick a feature or potential product we might be building (or have recently built but not released) – ideally the candidate might offer some new ideas. Although it is really about the process and how they arrive at their ideas.
  - (For more questions on this note, there is an amazing post here on hiring a product manager but a lot of the questions could easily be adapted here as well)

- **Customer Focus**
  - You join a team and discover most of the people are spending their time handling customer issues. What kind of process would you suggest to help the team be more proactive about addressing what needs to get fixed?
  - Tell me about a time when you were wowed by the service you received. What made it special? How could you apply those lessons to our product/service? What are some ways we could wow our customers?
  - Have you ever worked on a "customer first" team? What was it like? What policies or processes could a software team employ to be a "customer first" team?
  - A customer is complaining about a particular issue [it is great if you can pick a particular issue or problem with your site that could (or did) happen] but you can't reproduce it. How would you handle this situation?

- **Productivity & Ability to Get Things Done**
  - Tell me about a time you did something that you were really proud of but no one else knew about. // Ideally they should answer with some sort of cleanup work, like refactoring, or fixing other issues, or maybe even something as simple as cleaning up the kitchen. You want a sign that they care about work that is more than just the tasks they are assigned.
  - How do you manage your tasks and stay organized?
  - Tell me about a time when you were the most productive. What made that time special? Did you change anything about your day to day work to take advantage of those same attributes?
  - You have been assigned to a project in a new technology you haven't worked with before. How do you get started? Have you ever done this before? How did it go?

- **Focus on Quality**
  - When do you consider a product to be finished? // My favorite response is 'it's never finished' but anything about some level of testing or verification is generally acceptable.
  - When do you know your code is ready for prime time (shipping to production)?
  - What is the role of QA in previous roles? How do you like to work best with quality teams?
  - Explain a feature or product. How would you test this feature (or product)? You can also ask how to test a specific function after a coding problem.
  - You have just been put in charge of a big legacy piece of software with some serious maintenance issues. What do you do first? // This question works best if you have a product or example with some issues in mind – otherwise it can be a pretty abstract question.
  - Have you ever done TDD? Do you like it? Why or why not?
  - There are all different types of tests – unit tests, smoke tests, acceptance tests, integration tests, etc. What sort of tests have you written? What's the difference between [insert test here] and [insert different test here]?

- **Curiosity**
  - How do you stay on top of current trends and innovations?
  - What sort of websites or blogs do you read regularly? What do you like about them?
  - What was the last new technology or tool you learned? Where did you learn about it? Have you used it since?
  - What is the last programming (or technical) book you read?
  - When was the last time you got something wrong? How did you know? What did you learn from it?

- **Communication**
  - Teach me about something for the next 10 minutes. // You are looking for them to select a topic they know (I generally don't care if they are technical or not) and how well they communicate and break things down.
  - How did you communicate progress in your previous role? Did that process always work? What could have been done differently to keep everyone on the same page?
  - Tell me about a time when you had a miscommunication at work. What happened? If you could do it all over again would you alter your actions? Why or why not?
  - How would you explain the Internet to a child?
  - Have you ever disagreed with your boss or manager? What did you do? If you haven't had this happen, imagine that it did, how would you handle this situation?
  - Explain the concept of cloud computing to my older (not-very-technical) mother. // I actually had to do this in real life and it was harder than I expected. Another example is explain a database to someone's grandparents.
  - Give me an example of a time when you were able to communicate and work with another person even when they may not have personally liked you (or vice versa).

- **Passion**
  - Do you have any personal projects? Tell me about them. // This is probably one of my favorite and most important questions. When I hire I like to hire candidates who are very passionate about what they do,

    and so programming isn't something they just do at their job. They are actually passionate about it and pursue their own projects outside of work.
  - If money weren't an issue and you had to work on a project for 3 months, what would you create?
  - What are 3 big contributors to your success?
  - Give me an example of a goal you set for yourself and how you achieved it.
  - What are some of your hobbies or passions outside of work? How did you first get into them?

- **Culture Fit**

- Tell me about 3 times you have failed. // Almost anyone can come up with 1 or 2, but it can be hard to think of 3. Be sure to wait and be quiet while the candidate thinks and ponders the answer – I often feel like the most insightful answers come towards the end of this question.
- Do you still write code in your job? Do you love it? // This is more targeted at managers or leaders, since many do not need to be writing code to do their jobs. Hopefully this question could lead to an interesting conversation on role, etc.
- What do you want to do in 5 years? How would this job fit into your plan and help you toward your goals?
- Why do you want to work at [company name]? Have you used our products? Is there a particular area or feature that got you excited?
- Give an example of when you completed a task without being asked. Can you give me another example? Another?
- Tell me about a time you improved a tool, task, or project you were working on. What was the circumstances? Why did you do it? Do you have any other examples?
- If you were hiring someone in this role, what would you look for? What sort of interview questions would you ask? Do you have a favorite question?
- How did you prepare for this interview?
- Do you consider yourself weird? Why or why not?
- Are there any questions that we didn't ask you that we should have?

Update 6/17: This next set of questions came from Jan Schaumann, who reached out to me via Twitter noticing the lack of security questions from the list. Then he took it to the next level and compiled this awesome list of security interview questions on his blog, which I will also share here. Awesome work, Jan!

Security Related Interview Questions

Abstraction & Design

- You are tasked with designing software that runs and controls elevators. Explain the defaults for all options available in your implementation. What safety precautions do you have to build in?
- Imagine you were tasked with designing an instant messaging program / mobile app. Explain the access model and privacy implications of your design.
- Explain the user interface and process flow decisions of an ATM. How would you redesign/improve the UI?
- Your login process requires a password. Talk about the UI decisions around this. What tradeoffs do you make between password complexity requirements and usability? How do you design the user feedback for repeatedly entered wrong passwords?
- What feedback can you give a user to assure them that their data/connection/… is "secure"?

Algorithms (different approaches and performance)

- Write a function which will return a random number from the list. How do you know the "random" numbers you generate are actually random? (Not a question about algorithms, but a good starting point to talk about randomness in general.)
- Take a number lock with N digits. For what value of N is brute-forcing the lock a reasonable option? Assume you can parallelize your brute-forcing attempts – how does that change your answer?

Problem Solving (like algorithms, only more general)

- Write a password cracker. What input could it take to become more efficient?
- Using your environment's user data, how long would it take for a normal desktop/laptop to brute force all passwords? How would you speed this up? (Can you slow this down in some way?)
- Design an anonymous question / answer board with the ability to 'accept' answers or delete your own questions.

Command Line & Scripting

- Explain how SSH tunneling / port forwarding works. What are some pitfalls or risks?
- How does `sudo(1)` work? What are common related pitfalls?
- What is the difference in permissions on a directory that is mode `0777` and one that is mode `1777`?
- Why should or shouldn't you have `.` (dot) in your `PATH`?
- Write a simple script that requires the user to enter a password and pass it to another program. Talk me through the various considerations, assumptions and decisions you need to make.

Database Administration

- Where does your database live on your network? Explain the access controls around it.
- How do you restrict access to your database by different accounts/users?
- Your automated tools cannot prompt the user for a password to run commands against the database — how do you handle this?

Data Modeling & Data Schemas

- Your application allows users to purchase items using credit cards. Describe the data model and schemas used to protect both the users' privacy (shipping address, purchase history, …) as well as CC data.

SQL Queries (querying for data)

- Who is Little Bobby Tables? (How does SQL Injection work? How do you avoid it?)

Networking

- Why is `ping(8)` setuid? (Then talk about what you can do with raw sockets, move on to `tcpdump(1)`.)
- What's the difference between HTTP and HTTPS?
- Why can't you do Virtual Named Servers when using HTTPS?
- Describe what happens when I type https://google.com into a browser and hit return. Be as detailed as possible.

System Design & Thinking

- How do you deploy your code?
- How would you store passwords in a web application? Your mobile app?
- Are there alternatives to storing passwords?
- How do you handle SSH host keys?
- How do you bootstrap trust amongst hosts?
- At what point in the development cycle do you involve your security team?
- What is the difference between "fail-open" and "fail-closed"? Give examples of when either is more appropriate than the other.
- Discuss the concept of anonymity in a common website / application / protocol.

APIs

- How do you authenticate to the API?
- What kind of data do you accept from clients?

- What kind of functionality do you expose in the API? How do you decide which functionality requires authentication and which can be accessed anonymously?

Web Development

- Your application allows users to upload photos. How do you verify that the data you received is in fact a photo?
- What library functions do you use for processing users' passwords?
- How do you process data uploaded or entered by a user?
- What are CSP, CSRF and XSS? What can you do to protect your users?
- We all love jquery, google-analytics, etc. How do you source third-party toolkits like these?
- When/how would you integrate with Twitter/Facebook/Google login? What are the repercussions / advantages?

Reliability & Operations

- Tell me about a security incident you've taken part in.
- How do you determine whether or not a security update needs to be applied? Should you always install the latest version?
- Who in your organization has 'root'? How is 'root' access granted?
- How do you handle service accounts (headless accounts, automation accounts, …) and the access they have?
- How do you manage / utilize shared jumpboxes in your environment? What are some of the implications and pitfalls of having them?

Software Engineering

- What is "Trustworthy Computing"?
- Is closed or open source software more secure?
- How does a cryptographic hash function differ from "encryption"?
- How do collisions in such hash functions affect security (in theory and in practice)?

Teamwork & Collaboration

- How do you handle the case if a senior member (with lots of subject matter expertise) of your team continues to take shortcuts or engage in unsafe coding practices?
- Do you have experience with teams / organizations with a generally security-aware culture? How did / would you establish this such a culture?

Product Sense & Judgment

- How would you attack $SomeProduct / $SomeWebsite?
- Comment on the privacy trade-offs in today's social networks. Compare mobile app defaults and settings.

Customer Focus

- Your customers / users complain about a policy or restriction dictated by your organization's security team. You do not agree with the policy, either — how do you defend it? How do you circle back feedback?
- How do you communicate breaches of your security with your customers?

Productivity & Ability to Get Things Done

- Tell me about how you keep track of the hundreds of passwords you have.
- How does your organization handle collaborative editing? What are some advantages and what are the trade-offs made?

Focus on Quality

- Have you ever done or requested a penetration test of your product/network? If so, how was this done, what was your role? (If not, why not?)
- What is the difference between input validation and input sanitization?

Curiosity

- Have you ever participated in a "Capture the Flag" competition/game?
- Almost every developer/engineer sooner or later ends up designing (perhaps unwittily) a protocol involving some form of "authentication". Tell me about yours.

Communication

- How do you communicate breaches of your security with the press/twitter/internet at large? (Ie, can you improve on "We take security seriously…"?)
- How do you communicate breaches of your security within your organization?

Passion

- What is more important: usability or security? (A trick question indeed.) How much does your answer depend on the context?

Culture Fit

- How do the principles exhibited by the organization reflect your own?
- Review the data privacy model of the organization you're applying with — what improvements or changes would you suggest?
- "The Net interprets censorship as damage and routes around it." — is this true?

General / Other

- What are common pitfalls in string handling?
- What is a buffer overflow?
- In the programming language of your choice, describe how you execute a system command with certain parameters provided by the user.
- How does SSL certificate verification work?
- `ssh(1)` warns you about a hostkey fingerprint mismatch. What do you do? (No, be honest.) How do you solve this problem?
- How much can you trust DNS?
- What kind of data do you currently make available to third parties? What services do you outsource? What are the tradeoffs you make?
- How do you communicate / store / handle passwords or other "secrets" with your peers?

Need more interview questions?

Here are a few lists of questions to check out. You can also just try searching for specific questions or skills that you are looking for – there is no shortage of questions to ask.

- http://kundansingh.com/interview/
- https://sites.google.com/site/steveyegge2/five-essential-phone-screen-questions
- http://www.noop.nl/2009/01/100-interview-questions-for-software-developers.html
- http://www.joelonsoftware.com/articles/fog0000000073.html
- http://www.impactinterview.com/2009/10/140-google-interview-questions/
- http://www.hanselman.com/blog/NewInterviewQuestionsForSeniorSoftwareEngineers.aspx
- Web Development focused: http://darcyclarke.me/development/front-end-job-interview-questions/

Related Posts:

- Shred Your Resume: How To Get a Job at a Startup
- Why You Shouldn't Put Perks in a Job Description, and How to Prove Me Wrong
- Creating Good SDE Interview Questions – Effective Interviewing
- Questions for Candidates to Ask in an Interview
- Systems Engineers Interview Questions (also for System Adminstrators and Network Engineers)

Comments

kjellkod 14th June 2013 at 08:53

That is a vast array of questions to choose from Kate.

I recently went through a period of study-for-interviews as well as undertaking lots of interviews and I recognize many of your questions.

How interviews are done in the US seems very, very different from how interviews are done in Sweden. A lot of tech US companies have an almost extreme focus on BFS, DFS and different tree traversals,. maybe because this is the *big* thing the manager remember from his/her university time? Here the non-US candidate need to study up if these questions are not common in his home country.

As far as doing the job later on : knowing BFS/DFS/ways-of-traversal means nothing! It only means that you have studied up for these types of questions. In my opinion this is a waste of interview time.

Some companies had another approach for the technical aspect. An approach where interpersonal skills are also tested:

1. Language + software skills compared to number-of-years experience
Best tested by having the candidate do a programming task and return it to you once its done. Bonus points of course if there are unit tests etc to verify the code. If the candidate later comes in for an interview you can go through the code : discussing the solution, potential problems with it, maybe coming up with a change of requirement and see how the candidate is reasoning of what changes needs to be made.

2. whiteboard: reasoning about OO principles. It is okay not to know them by heart but if they are explained, can the candidate understand them?

3. whiteboard: discussion about software design. pro/con. The important thing is NOT to have the candidate coming up with a brilliant solution in 1 minute but to see whether or not you can have a meaningful cooperation between with the candidate. Can the candidate understand a new, complex scenario and reason about it.

4. most important: Would I like to have this person as a colleague. Using a *best buddy guarantee* so that any team member that is involved in the interview process can veto out the candidate. No one wants a jackass on the team.

kjellkod 14th June 2013 at 08:58

Oh yea. And like you mentioned above about passion. This is absolutely vital.

I prefer to hire someone if they show they are passionate about creating software. A person that never reads blogs/books/ and studies up in his/her free time or has never made any software on the side will have a much harder time getting the job.

Srikar Doddi (@SrikarDoddi) 20th June 2013 at 07:07

These is a very good list of questions but do I want to sound a bit of caution on the Algorithms, Problem Solving & Data Structures sections. I don't think they are necessary and in fact in my real life hiring and watching engineers that I hired perform over time, I did not see a big difference in terms engineers who knew those answers vs. those who did not (but got hired because they did well in other areas) in terms of productivity/resourcefulness. Similarly, I also did not see any correlation between engineers productivity and their academic grades. My advice is to stick to candidate's previous projects and dig deep into how he built those components and what made him choose one over the other.

kate 21st June 2013 at 02:28

That's a great point, Srikar. It is definitely most valuable to dig deep into past projects and learning experiences to get a true, complete sense of a candidate's background. But it's also important to assess technical knowledge, so I like to have a few technical topics at the ready to quiz candidates with as well. They are good for getting a sense of their baseline level of skill. A really good interview covers lots of topics, in my opinion though: the very technical (questions with right/wrong answers), technical questions with many answers (and you're looking for how they problem-solve/find the best answer), plus past experiences, culture, team-fit, plans/goals… Thanks so much for your thoughtful comment!

Shane Mac 1st August 2013 at 16:02

This is just great. Thanks for such a comprehensive list.

Anshuman Mishra 29th October 2013 at 05:57

Epic!

Prashanth 25th November 2013 at 02:58

I think apart from knowing algo and DS,its important to know their practical applications.
I hope this link helps
http://cstheory.stackexchange.com/questions/19759/core-algorithms-deployed/

Frank Traylor 4th February 2014 at 21:55

Fantastic list. Thanks! Here's a hackpad with a few contributions from various sources on questions to test for cultural fit: https://hackpad.com/Interview-questions-to-test-cultural-fit-PS3ytaage0O

about me

katemats
Kate is known as one of the top technology leaders and CTOs. Her technical background is in creating and operating large-scale web applications. Her focus has primarily rested on SaaS applications and big data. She has extensive experience building and managing high-performance teams, and considers herself a fan of agile development practices and the lean startup movement. She is currently founding her own startup, popforms, but has held roles as developer, project manager, product manager, and people manager at great companies including Amazon and Microsoft. The last seven years she has been a VP of Engineering/CTO for companies like Moz, Decide (acquired by eBay), and Delve Networks (acquired by Limelight). Kate is a keynote speaker, and she is also the curator of the Technology and Leadership Newsletter (TLN - www.techleadershipnews.com) and has a personal blog at katemats.com.

Personal Links

personal website

Tech Leadership Newsletter

topics

being awesome

business

career

communication

entrepreneur

good stuff

personal

productivity

rants

self improvement

technology

top posts & pages

Epic List of Interview Questions

A surefire formula to deal with difficult employees

Systems Engineers Interview Questions (also for System Adminstrators and Network Engineers)

Managing Difficult People - Your Pesky Peers (part 2)

Interview Questions for Software Development Managers and Leaders

Conversations with Your Team, Peers and Boss - New Tech Leader Series

The Ultimate Guide to Note-Taking

Performance Management - Addressing Bad Attitudes

Distributed Systems Basics - Handling Failure: Fault Tolerance and Monitoring

What Every Programmer Should Know About SEO popforms, my company

View Full Profile →