P4

Interviewer: Ok. Then the first question is, could you describe your practice in code auditing? For example, after you receive the project code, how do you conduct code auditing to identify its vulnerabilities? It would be best if you could share your screen and explain the entire process in detail by combining specific examples or projects.

P4: When getting a project, first look at its project background, white paper, etc. First, understand the background of a project, what it does, which libraries it uses, and its preconditions. After knowing the project, then look at its code, and you will have a corresponding understanding. Then you can sort out the general framework, including what its main components are and what it interfaces with externally. All these can be learned from the white paper and project introduction. Then, when looking at the code, actually compare it. First, look at the main description of its code implementation, and check whether the comments are consistent with the precondition description, and whether they are consistent with the introduction or white paper.Then take a look at the process from the entry point of the entire project, that is, how to start and play the project, how to enter from that project, and where the data goes. The entire flowchart here needs to be sorted out. After sorting out the flowchart, you can refer to the overall framework to understand what this project is doing, and whether it is consistent at the code level. From the data passed in by the user, the modified state, where this framework is passed, where it is modified, and where it is stored. After having a general understanding, you can apply some original risk items, describe some historical experiences, or summarize some security bugs. You can check which aspects may have which problems. After a general review, you can go deeper. Going deeper means that after reviewing the basic security information, you can identify these basic issues. Going even deeper can help you understand the operating principle of the entire project thoroughly, because the pre-information is already clear, and there won't be some other relatively low-level issues. Then you can delve into some of its code rules and logic, and judge where it may be the weakest. For example, for some data passed in by the user, which ones must have permission control but don't, you should judge whether it has permission control, which interfaces are accessible to users, and which are data exposure interfaces. It is in these places that you may read abnormal data, and they can use this data to do things, or you can directly modify some data. This is roughly delving into it.

Then, the mathematical aspects are not quite the same as normal vulnerabilities. You need to consider some precision issues, some manual input issues, as well as some other calculation methods, and it may also involve some aspects related to the company. This set of logic is quite different from that of vulnerabilities, just at these levels. As for the general overall framework, that is, the thinking of an audit, first judge the project information, then based on the project information, analyze its code logic, and then under the judgment of the code structure, check whether there are some basic issues. If not, you can delve deeper to identify some weak points of the project, then try to address some corresponding issues, and finally look at the mathematical aspects to see if there are any issues with formulas or other aspects. Roughly speaking, it is like this.

Interviewer: Okay, then could you briefly describe how you used ChatGPT to assist you throughout the entire process?

P4: When using GPT, generally you first ask it to describe something, then paste a code snippet to it. You may check whether GPT's understanding of the code is consistent with yours, or you can directly ask ChatGPT to understand it and interpret based on its understanding. First, you understand the information GPT provides about the code snippet you gave it, then ask it for some security issues. After getting the security issues, you can verify them, or you can directly ask GPT to provide some security verifications. Then check whether your own stored security knowledge is consistent with GPT's, or if there are aspects GPT considered that you didn't, you can conduct some verifications. Or, if GPT's considerations are all wrong, they can still give you some hints, and then you can look for relevant information based on them.

Another way is guided questioning. That is, you may want to conduct some tests, but you don't have any ideas about how to do them. You can tell GPT what problems you have at this time, and guide it to help you describe how to identify the problem. Then you can make some corresponding test preparations. When you quickly review some code in the future, you can adjust it to GPT, and it can directly help you identify the problems you previously provided. That's roughly how it works.

Interviewer: Just now we were asking about the entire process of using GPT for assistance, right? Can you describe the differences at different stages between the entire process with ChatGPT assistance and without ChatGPT assistance? For example, did it improve your efficiency, or did it enhance your auditing capabilities?

P4: Actually, both efficiency and breadth have improved, but efficiency is not guaranteed; different projects may not necessarily see improvements. Some are very broad and concise, some are relatively basic contracts, and GPT can provide relatively high-level evaluations, and it can point out those security issues. However, when dealing with some complex projects, the issues reported by GPT require you to gradually check whether the issues it presents actually exist and whether they are reasonable. These validations are necessary.

Interviewer: That is to say, after the complexity of this project increases, specifically how, that is, how the complexity increases, it will lead to some problems. What problems will arise when using ChatGPT?

P4: Regarding the code, it should be the roles existing within the code. If we introduce Oracle, introduce elements like users, as well as some other pools, and also things like staking, the more roles and functions it has, the more its accuracy will decline. This is because it has to consider more factors. Since all you provide is just the code, but you may not have provided all the preconditions. If you don't feed these to GPT, it will make some guesses based on its current environment, which you can be sure are invalid, but GPT will still give them to you, and you still need to spend time to verify them. At this level, the efficiency will be relatively low, and it is also quite troublesome to verify, because some preconditions you think are possible, but GPT may have some other justifications, and you still need to confirm these.

Interviewer: So, based on what you just described about the entire process, it can actually be divided into understanding, finding vulnerabilities, and then verifying vulnerabilities, right? So, when the task complexity increases, does ChatGPT cause a decrease in efficiency in the verification phase, or does it not necessarily improve efficiency in the early stage?

P4: It mainly enhances in that aspect, primarily helping you understand how the listed projects operate. It has a relatively clear flowchart or function capabilities, clearly listing what specific functions are being implemented and with whom it interacts. However, the relationships between them may introduce some security issues. Since ChatGPT is equivalent to having a database, it stores more information than normal auditors, and it may also introduce security issues beyond the website. It may not be homogeneous, but it will take them into account. When it presents you with corresponding issues and you need to verify them, you will also need other knowledge, which is where efficiency is somewhat reduced. If your knowledge is insufficient, it will be quite difficult to verify the issues presented by GPT. However, if it provides some understanding, it will make your understanding faster.

Interviewer: You just mentioned your Knowledge Base, right? Your own Knowledge.

P4: Right.

Interviewer: Since you have approximately two and a half years of experience in code auditing, you have a relatively systematic and mature knowledge system of your own. Then, when you conduct manual comparisons, for example, you can directly refer to this knowledge system to make a comparison.

P4: Regarding auditing, apart from those scattered security knowledge, when you talk about precision, such as reentrancy, but in fact, specifically, you still need to combine it with the code. For which projects, which reentrancy might lead to some stop-loss issues, and which reentrancy might lead to some calculation issues. For example, read-only reentrancy, its reentrancy logic is different, and the issues are also different. When these accumulate, if you are very familiar with the code and then see similar code snippets, you may consider the corresponding issues. These can be regarded as personal accumulation. However, GPT can think of more code snippets, but it is not very effective when you try to verify them.

Interviewer: Do you make some comparisons? For example, if you have a list, when you look at these codes, do you search for some comparisons, taking some vulnerabilities as your priority to look for? Or do you still rely on your intuition?

P4: With basic code knowledge, first you need to understand that some characteristics of a project, such as price manipulation in a recent reported project, exist. Based on the project, there are some vulnerabilities in its characteristics, as well as some common vulnerabilities and some permission-related vulnerabilities. These actually fall into several levels, including common code-level, no, language-level vulnerabilities, such as the simplest overflow issues, overflow issues before 0.8. You may need to take these into account, based on the version of the language it uses and other factors. There are language-level, logical-level, and common permission-level vulnerabilities, which can be divided into several major categories. When auditing a project, you can apply these accordingly. It can be understood as being similar to the self-study course mentioned earlier, which is actually based on your existing knowledge to see which items need to be applied to which projects. However, when it comes to the actual audit, you still need to understand the project and what modifications it has made. If these modifications are at a new level, you need to understand this and then consider some other issues.

Interviewer: And the next question, when you use ChatGPT to assist you in code auditing, what do you think are some of its biggest challenges? You just mentioned a challenge in verification, which is the biggest one for you. Are there any others?

P4: Actually, the biggest problem is that the accuracy is not high enough. Some of the issues you need to verify may exist, but if you want to verify them, the efficiency is low. Or, some of the content provided by ChatGPT may lead you astray. You were originally thinking about one thing and focused on this aspect, but GPT gives you other ideas, which may not be correct. When you try to verify these ideas, you may end up going in other directions, which will waste some time.

Interviewer: Then there's this question, which is that when you usually use other software, apart from ChatGPT, what do you think are the most critical features that other software has but GPT doesn't, yet you still consider very important?

P4: I haven't used other tools much; it seems like I haven't used them at all. In fact, I also use GPT relatively rarely because GPT is generally only used when working on complex projects. Some of its functions may take you a long time to understand and require a lot of effort. Then you can ask ChatGPT to help you interpret them so that you can quickly get started. However, when it comes to security issues, currently, unless you use a personal GPT to ask some security knowledge, which you can trust to some extent, for ordinary GPT, the information it provides is not very useful, mainly at the understanding level. And it is used relatively rarely in terms of assisting with security. As for other tools, their testing results are not very good, and I haven't used them much either.

Interviewer: Well, we're now planning to introduce a tool. This tool is equivalent to being mainly based on ChatGPT, that is, developing a tool based on ChatGPT to assist auditors in conducting audits. If you had such a tool, what do you think would be the biggest purpose for using it, as well as the future you envision?

P4: The purpose is definitely some of his safety tips. Just like when you do something, if you directly feed it a project, it can give you some safety effectiveness issues related to the project. An assistant, because ChatGPT is like you can feed it something, and if the things you feed it are only in one domain, it will definitely be more accurate. Because during human moderation, you actually won't think of every vulnerability. If the database it stores is large enough, and it can give a reminder when you encounter a similar problem, this is actually the most preferred, because humans may forget, but what is stored in the database won't. If it can give a reminder when there are issues like something to be fixed, it can better ensure the efficiency and effectiveness of one's own audit during the audit process.

Interviewer: It's about efficiency and effectiveness, which is equivalent to these two points.

P4: Right.

Interviewer: During the audit process, what exactly might it look like? That is to say, if GPT is used, what other possible uses are there? For example, not based on the Knowledge Base, because you know it's very difficult to collect the Knowledge Base. Let alone tacit knowledge, even the explicit knowledge pile, right, is quite a handful.

P4: Aren't you still doing fuzz testing? If you're still doing fuzz testing in that form.

Interviewer: Right, both exist. That's the traditional approach, but there are also areas of combination. What I mean is, in the idea of using GPT to assist auditing, what other ways are there that do not need to rely on the Knowledge Base?

P4: Can GPT assist, like in your testing, to assist with formal verification and such?

Interviewer: It's about writing some specific rules for fuzz testing and formal verification. Currently, there are two overall approaches using ChatGPT. One is the Knowledge Base approach, and the other is pure GPT. ChatGPT has been trained with sufficient knowledge over the past few years. As long as I have an effective prompt to ask, for example, if I ask 10,000 times, I can definitely collect all the results.

P4: You're like some beginners. Beginners surely need to get started, and when getting started, they may have no ideas for practical courses. Your GPT can provide them with some ideas. If you're cultivating someone, or if they're just beginners who want to do auditing but don't know some key points, ChatGPT can provide them. Regarding the auxiliary auditing just mentioned, the auxiliary tools you've developed for beginners will surely have an impact. Also, for development, it actually requires some security foundation, and your tool can provide support for development. You don't have to target only security personnel; you can also target developers. Developers can provide feedback on some issues and what problems have been fixed.

Interviewer: Does it mean that, for example, during the entire process of your auditing, GPT monitors your actions or mental activities in real time, and when it detects that you're stuck, it gives you a prompt?

P4: Almost now there's [unclear voice] that can help you write code, which can assist you and provide prompts while you're writing.

Interviewer: Copilot, right? Copilot is more of a result provider; it gives you more results and no reasons. If it could also tell you the reasons, it would actually be a great teaching tool. But I still can't quite understand how the beginner's tutorial works. It doesn't know to what extent you understand things.

P4: Right, it doesn't need to worry about how far you go; it just needs to provide the current standard. For example, when many people are developing, they often copy some courses and then modify some code. If your modifications are incorrect, it can point out the problems. What I envision more is that for beginners writing their own code, when they encounter some logical issues or language problems, it can prompt you that there is a problem. Also, on this platform, many people participate, and you can collect all the issues they write about.

Interviewer: I think the scene just now was really interesting. Well, I think today's interview will end here for now.