Feedback Report (Week 01)

Student ID 2021315385

Name 이건

Title: Practical Software Diversity

Speaker: Professor Hyungjoon Koo

After attending this week's seminar, please write a brief summary of the content along with you r comments. Your response should be approximately half a page in length.

이번 주 세미나를 듣고, 세미나의 내용을 간단히 요약한 뒤, 이에 대한 본인의 생각이나 의견을 작성하시기 바랍니다. 전체 분량은 약 0.5페이지 내외로 작성하여 제출하시기 바랍니다.

1. **Summary**

Briefly summarize the main content of this week's seminar.이번 세미나의 주요 내용을 간단히 요약하세요.

Practical software diversity is a part of the moving target defense (MTD) approach, which alters software data to complicate attacks. Techniques such as canaries, non-executable (NX) memory, data execution prevention (DEP), and address space layout randomization (ASLR) have defended code from several attacks. However, attackers have found ways to bypass these defenses through code reuse attacks. This problem has highlighted the need for a more adaptive defense mechanism, which has led to the development of compiler-assisted code randomization—a form of practical software diversity.

Compiler-assisted code randomization actively modifies the layout of the code by rearranging functions and basic blocks. This technique relies on compiler-rewriter cooperation. They consolidate and embed metadata, such as basic blocks and layouts, into a master binary, which software vendors then distribute through legacy distribution channels. Once end users receive the master binary, they generate variants locally. This approach improves the reliability, transparency, compatibility, and cost-effectiveness of the distribution process.

In summary, compiler-assisted code randomization represents a practical application of software diversity as it makes attacks more complex.

2. **Comments**

Please include your comments, noting what you found interesting, what you would like to learn more

about, and your personal thoughts on the seminar.이번 세미나에서 흥미로웠던 부분, 더 알고 싶은 주제, 그리고 본인의 생각을 자유롭게 작성하세요.

Practical software diversity is an excellent icebreaker for software security. It was interesting to learn about the practical application of how current-day programmers defend their code against various attacks. This topic has enlightened me to the importance of code randomization, as current-day AI and tools may help attackers manipulate code that does not utilize the MTD paradigm. If I pursue this field in the future, I would like to know the specifics on how to apply the compiler-assisted code randomization, as well as different variations of possible randomizations.