Episode 28 (on malware analysis)

Reverse Engineering

Tanuki, this time it's malware analysis.
Before we begin, a few words.
Malware analysis, in other words, means reverse engineering.

Fox, what's malware?

The generic term for a variety of computer viruses is malware.
It refers to computer viruses in a broad sense.

I've never heard the term "reverse engineering" before, but what does it mean?

A simple way to describe reverse engineering is to disassemble.

The assembler language was mentioned briefly in the explanation of the CPU in episode 3, but disassembly, is that it?

Yes, that one.
We discussed in Episode 3 that converting a program written in an assembly language into a machine language is called assembling. Disassembly is the process of converting machine language expressed in hexadecimal or binary back into assembler language.

What does that disassembly (reverse engineering) have to do with malware analysis?

You know that malware, or computer viruses, are programs.
No hacker would be stupid enough to send you a virus in source file form. Of course, they would send it in executable form, or embed it in an existing file.
By executable, we mean machine language. Machine language is written in hexadecimal or binary, and it is impossible to analyze a program written in hexadecimal or binary. Therefore, we disassemble them and convert them back to an assembler language that is easy to analyze.

I'm not sure if I can get back to the assembler language, Ira, but I'm not sure. Do you understand fox?

I, too, am not good at reading assembler language. It takes time, it is tiring, and the language specification (architecture) differs depending on the OS. But since it is the language closest to the computer (machine language), I have no choice but to try.
About 40 years ago, when there were still only 8-bit PCs, Russia (then Soviet Union) and Ukraine, where computer resources were scarce, used assembler language like a limb to speed up the processing of programs, so there are many excellent computer engineers even today.

This is a typical example of how people stop being creative when they become too wealthy. Nowadays, with the abundance of computer resources, it is possible to achieve a reasonable processing speed with high-level languages and scripts for the Web without using assembler languages. I had never heard of assembly language either.

I don't think that Japan's information education system will nurture talented people in the information field. Of course, I think it will foster experts in Word, Excel, and smartphone apps. However, I don't think it will nurture people who can program the basis of deep learning for AI, or who can program to serve as a shield or a spear for hacking.

This is because they are making students learn how to program in Scratch from elementary school.

Scratch is a programming language, but the script itself is hidden in a graphic. The reason seems to be to make learning fun in order to eliminate aversion to information. Will students who are accustomed to Scratch learn assembler, C, or Java? It is obvious that they will find it too much of a hassle. Naturally, some bright students will see the limitations of Scratch, learn the basics of computers, realize the importance of assembler, and start learning on their own. But that's just a few.

I understand what Kitsune is trying to say. In other words, information science is based on understanding the five major devices in terms of the flow of data in a computer. After that, you're saying that by learning assembler language, then C language, then Java language, in that order, you'll become a person who can reverse engineer. You don't want to make young people stupid by suddenly making them scratch the surface.

But it can't be helped that Japan's government officials are idiots who haven't learned anything about information science. Better yet, teach them malware analysis. I'll get over it!

Translated at DeepL