File 20101027.0650: Notes from Security Reading Group:

Shamal Faily introduced the paper by Peter Naur, 'Programming as Theory Building' (Microprocessing and Microprogramming 15) from 1985. Dr Flechais, Dr Martin, Shamal, Joe, John, Wattana, Anbang, Ronald and one other person I could not recognise on the camera image were there. Audio quality was good but the video dropped out frequently. The major case study in this paper is of two compiler groups. Naur is a compiler expert; he developed the Algol 60 compiler and won a Turing Award. The question is how theories differ from documentation and from source code.

Dr Flechais noted that 'theory' is an interesting word for it; theories are supposed to be falsifiable. I asked whether theories can be transmitted; can they be taught, learnt, stored, retrieved, compared? Dr Flechais was of the opinion that theories cannot be concisely written down. Shamal told a story of an apprentice jade dealer, who heard from his master endless stories about jade, telling where this or that piece of jade came from, but the master never gave any formal lessons. How did the apprentice come to learn the master's theory?

I thought the paper was a nice philosophical walk for a change. To link the theory idea to security, what about reverse engineering? I wondered aloud whether the term 'reverse engineering' had been coined yet in 1985; that was right around the time when a startup called Compaq was duplicating the IBM PC BIOS by analysis of the object code and API docs. The author, in Section 6, seems to say that it is never the case that programmers receive a system without any documentation and have to develop a theory of it. That is not true; hackers do it all the time and an attacker sometimes can develop a better theory of operation of a piece of software than the designer had, as evidenced by the fact that the hacker manages to make the programme do things that the designer never intended. (I forgot about the example of the Enigma, which Dr Flechais pointed out.) There was a recent report in the media: an article by Seymour Hersch in this week's New Yorker about the Chinese reverse engineering of a U.S. Navy EP-3 from April, 2001. They reverse engineered the embedded OS from surveillance and crypto apparatus that had just undergone rapid declassification, and the evidence is that it took them less than the seven years between capture of the aircraft and when the disclosure came early in 2008. Someone else mentioned the Samba project as a successful example of reverse engineering, and of course there is is DeCSS.

Theories may be incorrect or incomplete. What about purposely misleading theories? A security system designer may choose to design portions of the system in such a way as to engender false theories in the attacker's mind; a discussion of honeypots followed. Codebreakers and consumers of intelligence in WWII did something similar; they modified or controlled their own behaviour in order to give false impressions to the Axis powers of the efficacy of their own security systems and those of the attacker. It may be possible to design security systems similarly. (See the example from the Guardian of estimating tank production from analysis of fragmentary data from serial numbers of captured hulls; the estimate of 256 tanks per month was within 0.5 percent of the actual value, established after the war. If the Germans had obfuscated their sequential numbering system, the British would not have been able to make that deduction.

John brought up theories developed by the Information Systems Security Officer. Are they required to be complete? No, the ISSO can made decisions with an incomplete or even incorrect theory, so long as the predictions yielded by the theory are useful. Programmers need a complete and accurate theory to make changes to source code. Dr Flechais asked about the impact of incorrect theories. The author makes the case that they lead to decay of the programme text. I brought up the idea of revival of dead programmes in Section 6. All this talk of incorrect theories suggests that storage and retrieval, in particular checkpointing of theories would be a fruitful line of research and development. Shamal mentioned knowledge management systems in that regard. Can theories be compared? If so, they could be improved, distributed, and debugged.

I mentioned a paper from an unrelated discipline, Yuri Lazebnik's paper from cancer biology, 'Can a Biologist Fix a Radio' (Biochemistry Moscow 69(12), pp. 1403–1406, 2004). That paper is all about theory building. It talks about incorrct theories, whether incorrect theories are useful, and how theories are improved.

References