20 Years of Covert Channel Modeling and Analysis

Jonathan Millen
SRI International
Computer Science Laboratory
Menlo Park, CA 94025

Abstract

Covert channels emerged in mystery and departed in confusion.

1. The Paradox

My own initiation into the existence of covert channels came in a meeting at MITRE with the development team of the PDP-11/45 Brassboard Kernel, an experiment in the design of a multilevel security kernel. This was during the era when secure UNIX projects were going on, as well as secure Multics. Lee Schiller, the kernel designer, was explaining how a high-level process could signal information by using up the disk quota, which was detectable by a lower-level process. I had just finished proving with mathematical exactitude that the kernel satisfied the Bell-LaPadula model. How could this happen? It was profoundly disturbing to an impressionable youngster.

2. Storage and Timing Channels

Covert channels became known to the community largely due to Lampson's "Note on the Confinement Problem," which introduced the term "covert channels" but restricted its use to a subclass of leakage channels that excluded storage channels and "legitimate" channels. Nowadays, we call storage channels and timing channels covert channels, and call legitimate channels examples of information hiding. This usage reflects our advanced modern understanding of the problem, or it would if we could figure out the difference between storage and timing channels. That would have been easier if Schaefer hadn't come up with the disk-arm channel, which Wray described as a timing channel with a storage exploitation. Or maybe it was a storage channel with a timing exploitation.

Incidentally, I won't attempt to give references for the work I'm misrepresenting, since there is not enough space here. But you can find most of them in Virgil Gligor's Covert Channel Analysis guideline. [1]

3. Information Hiding

Covert channels are a means of communication between two processes that are not permitted to communicate, but do so anyway, a few bits at a time, by affecting shared resources. Information hiding is slightly different: the two communicating parties are allowed to talk, but the content is censored and restricted to certain subjects. The trick is to "piggyback" some contraband data invisibly on the legitimate content. The canonical example of this is to use the low-order two bits of each pixel in a picture for your secret message, since no one would notice if they were changed. When a similar idea was applied to smuggle information in network headers, we called it a network covert channel, mostly because the term "information hiding" hadn't been invented yet.

Then the crypto crowd came along with subliminal channels. For example, signalling a bit by choosing one of two keys to sign a permitted message. These channels share all the worst features of covert channels and information hiding: you get only a few bits at a time, and if the censors find out you're doing it, they cancel your service.

4. Modeling

Research in covert channels split up into four disciplines: explaining them, finding them, measuring them, and mitigating them. Explaining them meant coming up with a model that, unlike access control models, recognized their existence. Information flow models, like Dorothy Denning's lattice model, were among the first, but because they operated at the high-level language level, they overestimated flows. They found covert channels that weren't there. These were informally called "formal flows."

[1] National Computer Security Center, "A Guide to Understanding Covert Channel Analysis of Trusted Systems," NCSC-TG-030, November 1993, available at http://www.radium.ncsc.mil/tpep/library/rainbow/

Noninterference models conveyed a better approximation to the actual set of covert channels. Their main drawback was that they were defined only on abstract automata. In order to apply them to real systems, we needed "unwinding theorems," which were characterized by conditions on state transitions. The similarity of a state transition diagram to a game of "cat's cradle" must have suggested this terminology. The problem with using unwinding theorems is that you had to identify a suitable "view" function, and if you couldn't find the best one, you ended up with formal flows again. Despite this practical problem, the theory of noninterference was very popular, and spawned versions for all kinds of computational models in countries all over the world, leading to international cooperation and workshops in scenic places, where we had all the views we needed.

5. Searching

Information flow analysis was fairly successful, overestimates notwithstanding, as a way to search for covert channels in secure operating systems. Tools like the MITRE flow analyzer, The Gypsy flow analyzer, Ina Flo/MLS, and the SRI HDM flow analyzer were developed. Unfortunately, they analyzed formal specifications rather than code, and no real programmer liked to write formal specifications, so they were used only a few times. The Shared Resource Matrix method was better liked, because it was possible (though not recommended) to exercise it with nothing more than a command summary.

6. Measuring

Once some covert channels were identified, their seriousness could be determined, in part, by estimating their bandwidth in bits per second. The fact that bandwidth is a property of analog communication, measured in hertz, gives some indication of the rigor of this process. One reason for estimating covert channel rates was that the National Computer Security Center had evaluation requirements on them. Actually, there were never any requirements, just guidelines, and they were constantly being expanded and revised. The Orange Book suggested that a leakage rate of 100 bits per second should be considered high, since "many terminals" ran at that rate. They were thinking of ASR-33 teletypes, of which there aren't many nowadays. It is also hard to think of 100 bits per second as high, now that we know that there are hardware-based channels (for example, bus-contention channels) of thousands of bits per second, which are nearly unavoidable. On the other hand, your most valuable information is probably your 512-bit encryption key; how long is that going to be kept secret even at one bit per second?

7. Mitigating

There have been efforts to design or redesign computer systems to reduce or eliminate covert channels. One technique is called "fuzzy time." The idea is to make the system lie a little, randomly, about what time it is, so that the real-time clock (which should now perhaps be called the unreal-time clock) would be less useful for implementing timing channels. It also has the advantage that with 50% probability it delays the Y2K problem by several microseconds. Another technique is the NRL Data Pump, which supports reliable communication from low-level to high-level processes. The covert channel implicit in high-to-low acknowledgements is reduced by inserting random delays. I would have called this "fuzzy delivery," but no one asked me. One of the simplest solutions I've seen implemented is an adjustable system parameter that increases the execution time of every system call by a given number of milliseconds. Crank it up high enough, and you can bring your covert channel rate down as low as you like. There was an idea by Proctor and Neumann to build a system for efficient multilevel data sharing from a disk server, with zero covert channels. It was never built, probably because no potential sponsor wanted to give up the comforting cliche that it couldn't be done.

8. Ontology

The one question that everyone asks about covert channels is whether they are a real threat. They are difficult to implement and exploit. It is necessary to implant a Trojan horse program which then has to locate sensitive data, encode it, and leak it out over a long period of time. And to do so, it must run concurrently with a lower-level cooperating receiver program. All this without triggering alarms on audit data. They can be demonstrated "in captivity," but do they occur "in the wild?" Has any malicious party actually tried to exploit them? Because of the sensitivity of this subject, it went unanswered for many years. But at a workshop not too long ago, Bob Morris gave this question its final and complete answer: "Yes."