Challenge 1: Covert channels

Implement a one-way inter-VM communication scheme that makes use of covert channels.

Experimental setup

Two Xen virtual machines running Linux reside on the same computer. Each VM has:

- one virtual CPU. The VMs VCPUs are pinned to the same physical CPU
- 256 MB or RAM
- an ssh server

The VMs' clocks are synchronized.

The VMs are not able to exchange packets with each other.

Your task

Your task is to implement a sender and a receiver that make use of a based covert channel to communicate. Each of them runs on a separate VM. We recommend using CPU load to send bits across. If you want to use a different covert channel, please discuss it with us first.

The two machines must not exchange any data via conventional means.

Your solution must be tolerant to noise from other VMs. You are expected to implement something like frame control sequences [1]. FCS mismatches are reported by the receiver, prompting the sender to resend the frames. For the sake of simplicity you can use a simple start-stop protocol with fixed-length frames.

The sender should read the data that must be sent from a file of arbitrary length; likewise, the receiver should write it to a file. The files' format should be either:

- a string of ASCII characters, each representing 8 bits or
- a string of '0' and '1' characters, each representing one bit.

It is OK if the received data is padded with zeros.

You will be graded primarily on how well the protocol was designed. Your solution does not need to be resilient to crashes.

Reproducibility

Because the hardware used to check your homework is very likely not to have the same specs as the one it was developed on, refrain from hardcoding anything related to performance (e.g. how many loops the CPU can execute per second).

Instead, write a program/script that measures these constants and writes them to a file. If the program/script needs to run in both VMs at the same time, write a script that runs in Dom0 and calls the former via SSH.

Hints

It helps to view the protocol as a two-layer protocol stack:

- Layer 1 is responsible for getting bits across.
- Layer 2 is responsible for reliably carrying packets across.
-

When designing the protocol, consider the following questions:

- Is the protocol prone to clock skew? Aim for a fixed bit rate.
-
- Is the FCS strong enough? Can it be made full-proof?
- Is the sender always made aware of losses? Can it ever mistake a loss for a properly received packet?
- Can retransmissions of properly received packets occur? If so, can the receiver tell the difference between a retransmission and a new packet?
- Use the clock to align both the sender's and the receiver's bit times.
-

Submission

You should submit an archive containing the following:

- all programs/scripts
- a document explaining:
    - your chosen solution and the motivation behind it
    - how to build your program(s)
    - how to run it (in detail)

Got an idea?

If you would like to pursue a covert channel not outlined here or would like to use a different experimental setup, drop us a line.

[1] http://en.wikipedia.org/wiki/Frame_check_sequence