Covert Channels

- bottom line: covert communication of information using typical, valid channels of communication
  - overt communication on such channels is typical and normal
    - e.g., web sites, FTP sites, etc
  - what are overt communication channels?
    - talking, phone conversations, chats, Facebook newsfeed, TCP/IP packets, …, ?
  - can you think of ways that such channels could be used to communicate covertly?
    - code words, preset conversation systems, puzzles and codes, …, ?

- technically: an attack that allows the transfer of information between entities that aren't supposed to be allowed to communicate as set by some access control policy
  - this establishes the idea of “covert”
  - hidden from access control mechanisms
  - hard to detect
  - but hard to setup (requires administrative access to machines)
  - low bandwidth (takes a long time to exfiltrate information)

  - note that staganography (hiding data within data) is not a covert channel
    - we're simply using a channel to covertly transmit data

- storage covert channels
  - communicate by modifying some storage location
  - one process writes to some resource
  - another process reads from it

  - e.g., abuse the print queue
    - sender either fills up the queue (signals a 1)
    - or leaves it alone (signals a 0)
    - receiver polls the queue to receive the message
  - e.g., abuse web site log files
    - request web page A (signals a 0)
    - request web page B (signals a 1)
    - receiver reads the log file for the message
  - e.g., abuse FTP site file privileges and permissions
    - `drwxrwxrwx` (10 bits of information!)
    - order files alphabetically, set permissions appropriately
    - concatenate bits for the message
    - can be done on an anonymous FTP with incoming permitted

- timing covert channels
  - communicate by affecting/modifying some observed response time (of a receiver)
    - e.g., modulating usage of system resources (e.g., CPU time) that a receiver can monitor
  - time (the clock) is the shared resource

  - e.g., pattern of opening and closing a file (timing)
  - e.g., using port knocking on different ports using some timing mechanism
  - e.g., using the hard drive head
    - sender has access to the entire hard drive (administrative access)

- receiver has access to some portion of the hard drive
- sender makes a file request far away from the receiver's hard drive area (signals a 1)
- sender does nothing (signals a 0)
- receiver makes a request within its hard drive area
- receiver uses the time it takes for the head to travel to its section and finish the request
    - long time → 1
    - short time → 0

network covert channels

- information is placed in packet headers (not in the payload – that's steganography)
    - e.g., IP, offset, options, TCP checksum, sequence numbers
- or conveyed through action/reaction
    - send a covert packet, some number of legitimate packets, another covert packet, etc

combination of storage and network (with a patsy)!

- sender encodes a covert message in the sequence number field of a packet (or many packets)
- sender forges the source IP address with the IP address of the intended receiver
- sender sends the packet(s) as part of the TCP handshake to a patsy
    - TCP handshake: **SYN**, SYN-ACK, ACK
        - or **SYN**, SYN-RST
    - patsy: an unknown “man-in-the-middle”
    - this makes the covert channel harder to detect
- the patsy receives the packets(s) and either:
    - (1) responds to the receiver (via the forged IP address) with SYN-ACK
    - (2) responds to the receiver with SYN-RST
    - increments the sequence number of the packet(s) by 1 (standard procedure)
- receiver receives the SYN-ACK or SYN-RST packet(s)
- receiver decrements the sequence number of the packets(s)
    - and decodes the covert message
- in either case (i.e., SYN-ACK or SYN-RST), receiver doesn't respond to the patsy

tutorials

- ftp (storage)
- chat (timing)