Keyboards and Covert Channels

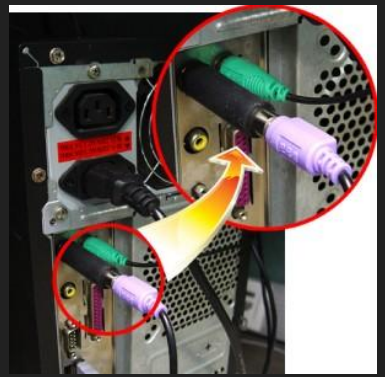
G. Shah, A. Molina, M. Blaze

USENIX 2006

Eric Hennenfent

The Humble Keylogger

- Malware
- Malicious Hardware
  - In-Cable Devices
  - Malicious Keyboards
    - Supply Chain Attacks
  - Malicious BIOS
  - Wireless Keyboard Sniffers
  - Evil Maids



Data Exfiltration

- Phone home to a server via malware
  - Possible to make this very subtle
    - Encryption + Steganography
    - Hide in social media posts
  - Doesn’t work for hardware keyloggers
- Break in and retrieve keylogger
- Connect to WiFi and phone home
- Broadcast keystrokes via radio
- Flicker the lights

The Keyboard *JitterBug*

- Hardware Keylogger - sits between the keyboard and computer (or inside a malicious keyboard)
- Exfiltrates data by introducing slight delays into keyboard event delivery
- Mistimed keystrokes can be statistically reconstructed from the TCP packet timings in an interactive connection
- Attacker only needs to intercept packets somewhere between the victim and a known host

Covert Channels

- Any means of data transfer that can’t be detected by hardware security mechanisms
  - Statistical analysis can sometimes reveal the use of covert channels
- Network Steganography Channels (Storage Channels)
  - All layers of the OSI model have been used to exfiltrate hidden data
    - TCP Sequence Numbers
    - DNS
    - TCP Headers
- Timing Channels
  - Data hidden in slight timing variations to legitimate data
- “Orange Book” requires some government systems to actively monitor network connections for covert channel activity

TCP Jitter

- Timing-based covert channel attacks generally require access to the TCP stack
- Many network events are caused directly because of events originating from a human interface device
- SSH and Telnet both send keystrokes to the remote machine as soon as they are entered
- Many web-based instant messaging platforms fire an ajax request each time a character is entered
- The keyboard Jitterbug causes TCP jitter by adding small delays to all keypresses

Encoding & Decoding

- Since the attacker doesn’t know the precise timing of the user’s keypresses, they can’t just insert a fixed delay
- Define a window size *w*
  - To encode a 1, add a delay such that the time between two keypresses mod *w* is exactly *w/2*
  - To encode a 0, add a delay such that the time between two keypresses mod *w* is exactly 0
- Calculate the time *t* between two packets at the listening end
  - Let *b* be *t* mod *w*
  - The closer *b* is to *w/2*, the more likely it is to encode a 1
  - Likewise, the further *b* is from *w/2*, the more likely it is to encode a 0
- In noisy network environments, it may be necessary to take the average of several values of t
- This scheme encodes one bit per keypress

Performance

Node	RTT	Hops	1s	500ms	100ms	20ms	15ms	10ms	5ms	2ms
ColumbiaU (NYC, NY)	6 ms	14	0	0	0	.007	.007	.010	.044	.089
UKansas (Lawrence, KS)	42 ms	14	0	0	0	.005	.007	.008	.067	.143
UUtah (Salt Lake City, UT)	73 ms	23	0	0	0	.005	.005	.005	.039	.092
UCSD (San Diego, CA)	84 ms	19	0	0	0	.010	.011	.011	.044	.102
ETHZ (ETH, Zurich)	112 ms	17	0	0	0	.005	.006	.009	.049	.092
NUS (Singapore)	236ms	18	0	0	0	.045	.047	.048	.228	.240

Table 1: **Measured Raw Bit Error Rate for different window sizes and network nodes (Levenshtein Distance Metric)**

Convolution

After the user presses a key, the following events must happen before the attacker can retrieve the exfiltrated data

- Keyboard event placed in buffer
- Interrupt fires; Operating system reads buffer
- Key code propagates to user space buffer
- User space code performs processing (encryption, web, etc)
- Packets delivered to TCP stack
- Packets sent over wire
- Packets routed between networks to destination

Each step introduces more variability, burying the deliberate time shifts in noise

Advantages

- Doesn’t require compromising the software on the machine
  - No antivirus to worry about!
- Almost impossible to detect
  - Even if it was detected, most users would explain the delays as faulty hardware
- Works even when the outgoing data is encrypted
- Given enough intercepted packets, the attacker can reconstruct exfiltrated data under any kind of noise conditions

Disadvantages

- Slow
  - On the order of bits per minute, not megabits per second
- Easy to confuse
  - The USB Stack, TCP Stack, and network infrastructure all introduce uncontrolled timing variances
  - Possible to introduce custom delay to reduce bandwidth of covert channel
- Substantially slows down as more confounding factors come into play
  - Exfiltrated bits must be statistically reconstructed from altered timings
  - If defenders artificially decrease the signal to noise ratio, the time to reconstruct grows
- Still requires attackers to intercept packets from the host
  - A very large number of intercepted packets are needed for noisy signals
- Requires an interactive application to be used*
- No way to know when a user has switched to a non-interactive application

Beyond Keyboards

- While the authors only implemented keyboard jittering, in theory, any USB device could be used to exfiltrate data in this way
- Streaming video and audio send packets at regular intervals, making them tempting targets for data exfiltration via jittering
- Unfortunately, such devices would not be able to provide access to typed passwords on their own
- A malicious USB hub could capture keypresses while taking advantage of high-bandwidth

Mitigation

- Detection
  - Dual-clock analysis - detects channels that use both timing and network steganography
  - High-level heuristics
  - Difficult because traffic timings are inherently irregular
- Confusion - the OS can insert its own arbitrary delays into incoming keypress handling and outgoing TCP packets
  - Can’t overcome statistical analysis forever
- Bandwidth limitations
  - Network pump - fixed-speed packet buffering
  - Fuzzy time & timing jammers - prevent processes from getting sufficiently precise time information

Discussion

- What are the key contributions of this paper?
- What are its limitations; what could it have done better?
- Is this a practical attack for real-world scenarios?