Josef Ressel Center for User-friendly Secure Mobile Environments (u'smile)
University of Applied Sciences Upper Austria
FH OÖ Forschungs & Entwicklungs GmbH

Contact:
Dr. Michael Roland

Softwarepark 11
4232 Hagenberg/Austria

+43 (0)50804-27149
michael.roland@fh-hagenberg.at
www.usmile.at • www.fh-ooe.at

Software Firewall to Protect against (Bad)USB Devices

Motivation

Recently, the term "BadUSB" [1, 2] raised awareness to attacks that can be performed by simply attaching a USB device to a system. Plug-and-play USB devices can, for instance, emulate keyboard and mouse inputs to automatically launch commands upon attaching. In parallel such a USB device could act as a mass storage device or as a network card to inject additional data. An attacker could abuse this to automatically launch malware or to modify network settings in order to intercept network traffic.

BadUSB is not new [3]. Even before BadUSB became widely known, attack tools like the USB Rubber Duck [4] existed and allowed injection of keyboard inputs, etc. However, BadUSB adds an additional threat: The firmware running on genuine USB devices (e.g. mass storage devices) can be modified (directly over their USB interface) to perform an attack.

This class of attack capabilities seems to be an inherent problem of general-purpose plug-and-play busses like USB. By looking at the physical object, a user can only guess what functionality a certain USB device exposes. For instance, if a device that looks like a webcam, a user would typically expect that it acts like a webcam (and only like that). The only reliable method would be to inspect what functionality the device actually exposes upon attaching (actually a bit more complicated: whenever the device (re-)configures itself). However, upon attaching the device to a host, the device is automatically configured and made ready-to-use. As a consequence, the user has no chance to inspect the USB device prior to potentially triggering any malicious functionality.

Objective

In order to overcome such attack scenarios, a software firewall should be developed (cf. [2, 6]). By default, all (new) USB devices should be banned (effectively disabling the plug-and-play capability). Upon detection of a new device, the user should be prompted to accept the device (showing details on what type of device that is). Only after the user explicitly accepted/whitelisted the device, it should be activated on the system.

Questions that should be answered in this master's thesis are

- How can the interception/firewalling be implemented on Windows and Linux?
- What information can be presented to the user to help in making a decision? E.g.,
  - "this device exposes mass storage functionality"
  - "this device has be moved to a different port"
  - "you previously used this device"
  - "this is a keyboard but there's already another keyboard attached"
- How can this information be presented in a user-friendly form that's not simply ignored by users?
- How can whitelisting be implemented? E.g.,
  - Can devices be uniquely identified (or could this information potentially be forged)?
  - Can devices be pinned to specific physical ports? Even across intermediate USB hubs?

Literature

[1] K. Nohl, S. Krißler, and J. Lell: "BadUSB – On accessories that turn evil", Presented at Black Hat USA 2014, https://srlabs.de/blog/wp-content/uploads/2014/07/SRLabs-BadUSB-BlackHat-v1.pdf

[2] K. Nohl, S. Krißler, and J. Lell: "BadUSB – On accessories that turn evil", Presented at PacSec 2014, https://srlabs.de/blog/wp-content/uploads/2014/11/SRLabs-BadUSB-Pacsec-v2.pdf

[3] J. Edge: "BadUSB: Clever but not novel", in LWN.net, Aug. 2014, http://lwn.net/Articles/608503/

[4] USB Rubber Duck, https://hakshop.myshopify.com/collections/usb-rubber-ducky

[5] USBdriveby, http://samy.pl/usbdriveby/

[6] Comments on article "Plug-and-play sanitization of USB thumb drives", in LWN.net, Dec. 2014, http://lwn.net/Articles/626559/#CommAnchor626763