usbguard

Schutz vor manipulierten USB-Sticks
Betriebssystem: Linux

PC-Treff-BB, Markus Näher

Worum geht's?

- Schutz vor manipulierten USB-Sticks

  Dabei handelt es sich um (scheinbare) Speicher-, WiFi-, ... Sticks, die sich insgeheim auch noch wie eine Tastatur verhalten, eine „Eingabeaufforderung" bzw. Terminal öffnen (z.B. per Hotkey) und dort Kommandos ausführen. Das geht so schnell, dass man es nicht mitkriegt. Solche Sticks wurden auch schon mit Linux als Zielsystem gesichtet.
- Wie immer: Sicherheit kostet ein bisschen Bequemlichkeit.
- Anmerkung: Alle folgenden Einrichtungs- und Konfigurations-Tätigkeiten müssen als root vorgenommen werden.

  Also vor alle Kommandos ein `sudo` setzen oder gleich mit `sudo su -` eine root-Shell öffnen.

Wie funktioniert’s (1)?

- USB-Geräte müssen sich am PC identifizieren und dabei auch angeben, was sie alles können:

  Massenspeicher (Stick, Festplatte), Drucker, Scanner, Kamera, Eingabegerät, ...

  Diese „Fähigkeiten“ werden „interface“ genannt. Multifunktionsgeräte geben mehrere an.
- Im Betrieb können sie dann nur das, was sie beim Einstecken angegeben haben.

Wie funktioniert’s (2)?

- Ein interface wird als 3 Byte (Hex)Zahl angegeben, getrennt durch Doppelpunkte.

  Beispiel: 08:06:50
- Interessant ist vor allem die Geräteklasse (der erste Teil):
  - 03: Eingabegerät
  - 08: Massenspeicher
  - ...
- D.h. wenn etwas sich z.B. als 03: und 08: anmeldet,

  ist Vorsicht geboten.
- Doku über USB-Geräteklassen:

  https://www.usb.org/defined-class-codes

Wie funktioniert’s (3)?

- Konfiguriert den Kernel so um, dass neu angeschlossene Geräte erst mal in Quarantäne kommen.
- Lauscht im USB-Subsystem nach Ereignissen.
- Bei USB-Ereignis „Neues Gerät“ werden die Gerätedaten mit den Freigabe-Regeln abgeglichen.
- Freigabe-Regeln stehen in eigener Konfigurationsdatei.
- Nur Geräte, die explizit freigegeben wurden, werden aus der Quarantäne entlassen.
- Erst daraufhin werden die Gerätetreiber vom Kernel geladen und an das Gerät gebunden.

Installation

- Debian, Ubuntu, Mint: `apt install usbguard`
- Arch, Manjaro: `pacman -S usbguard`
- Fedora: `dnf install usbguard`
- **Auf keinen Fall starten, ohne eine Grundkonfiguration erstellt zu haben !!!**

  Man würde sich selber aussperren, da auch bei Laptops die interne Tastatur und Maus oft über USB angeschlossen sind.
- Je nach Distribution kann der Installer evtl. schon eine Konfigurationsdatei erstellt haben.
- **Trotzdem überprüfen !!!**
- Konfigurations-Datei: `/etc/usbguard/rules.conf`

Einrichtung (1)

- Grund-Konfiguration selber erstellen:

  ```
  usbguard generate-policy >/etc/usbguard/rules.conf
  ```

  ggf. alles, was man so regelmäßig benutzt, vorher schon mal anschließen (Drucker, Scanner, Kamera, …)
- **Die erstellte Konfigurationsdatei prüfen !!!**
- usbguard starten:

  ```
  systemctl start usbguard.service
  ```
- Automatischen Start beim Booten einschalten:

  ```
  systemctl enable usbguard.service
  ```

  Wird ggf. auch schon vom Installer erledigt. Prüfen mit:

  ```
  systemctl status usbguard.service
  ```

Einrichtung (2)

- Später weitere Geräte hinzufügen:
  - In Konfigurations-Datei eintragen (Details im Praxis-Teil).
  - usbguard neu starten:

    ```
    systemctl restart usbguard.service
    ```

    Ohne Neustart werden die Änderungen nicht aktiv.

Feinheiten in der Konfiguration

- Kommentare und Leerzeilen zur Strukturierung sind möglich.
- USB-Hubs müssen auch eingetragen werden.
- Geräte werden sehr genau identifiziert, bis hin zur Seriennummer.
- Auch an welchem Anschluss sie eingesteckt sind.
- Geräte am Hub oder direkt sind „unterschiedliche“ Geräte.
- Man kann die Angaben nachbearbeiten und z.B. ungenauer machen (ich lösche immer „via-port“, d.h. den genauen Anschluss).

Wildcard-Freigaben (1)

- Man will natürlich nicht jeden einzelnen USB-Stick freigeben müssen. Dafür gibt man alles frei, was sich als „Massenspeicher und sonst nix“ beim PC identifiziert.

  Ausschnitt aus der Konfiguration:

```
# Generic Storage-only devices
# ============================
allow with-interface equals { 08:*:* }
```

- Das „equals“ ist wichtig! Ansonsten wäre die Regel „hat u.A.“
- Man kann damit USB-Sticks auch überprüfen:

  Wenn ein Stick direkt nach dem Einstecken ohne weitere Aktion auf dem Desktop sichtbar wird, ist er OK.

Wildcard-Freigaben (2)

- Manche externen Festplatten (vor allem Seagate) oder USB-Gehäuse geben mehrere interfaces an,

  z.B. { 08:06:50 08:06:62 }

  aber da alles mit „08:“ beginnt, ist es harmlos.

  Die kann man ggf. damit auch noch erlauben:

```
allow with-interface equals { 08:*:* 08:*:* }
```

Special: Arduinos (1)

- Es gibt ein paar Arduinos, die können auch als Eingabegerät fungieren, z.B. der Leonardo und mache „Pro Mini's“.
- Der UNO aber nicht. ESP8266 / ESP32 auch nicht.
- Faustregel: Wenn es eine separate USB-Bridge hat, dann nicht.
- Man könnte jetzt sagen „wenn jemand einen USB-Stick manipuliert, verpasst er ihm die Leonardo-Gerätedaten“.
- Da hilft dann nur Disziplin:
  - Den entsprechenden Arduino immer nur am selben Port anschließen ("via-port" in der Konfiguration drin lassen).
  - USB-Sticks nie am selben Port einstecken.

Special: Arduinos (2)

Warum diese Spezialbehandlung?

- Man gibt bewusst ein Gerät mit 03:… frei.
- Das ist ein verbreitetes Gerät, das leicht nachzuahmen ist.
- Man will trotzdem sicherstellen, dass nur das eigene Gerät freigegeben wird.

Sonstiges

- Es gibt auch ein Tray-Applet mit GUI, das habe ich aber noch nicht ausprobiert.

  Scheint auch eher für einmalige Freigaben gedacht zu sein.
- Weiterführende Infos:
  - https://www.privacy-handbuch.de/handbuch_91a.htm
  - https://reposit.haw-hamburg.de/bitstream/20.500.12738/7447/1/Bachelorarbeit_Moritz_Duge.pdf

Praxis: neue Geräte hinzufügen

- Was man **nicht** machen sollte:

```
# usbguard generate-policy >/etc/usbguard/rules.conf
```

  Das ist nur für die Ersteinrichtung!!! - daher auskommentiert.

- Stattdessen:

```
usbguard generate-policy >/tmp/rules_NEW.conf
```

- Der Vergleich mit `/etc/usbguard/rules.conf` zeigt, dass alle Geräte, die früher schon mal eingetragen wurden, aber jetzt nicht angeschlossen sind, in der neuen Datei fehlen. Vergleichen mit:

```
meld /tmp/rules_NEW.conf /etc/usbguard/rules.conf
```

  bzw. `kdiff` für Wikinger … äh … KDE-User.

Skript: identify_usb_device.sh (1)

- Dateivergleich selbst mit GUI-Tools teilweise unübersichtlich.
- Lösung:

  Ein Skript, das nur die Daten des neuen Gerätes anzeigt.
- Diese werden dann (per Texteditor) manuell in der Konfigurationsdatei eingetragen.
- **Vorher noch prüfen, ob die 03: drin ist**, ggf. „via-port“ entfernen, ...
- Funktionsweise:
  - Der usbguard-Regel-Generator wird vor und nach dem Einstecken aufgerufen und ein vorher/nachher-Vergleich ausgegeben.
  - Dafür wartet das Skript darauf, dass man die Eingabetaste drückt.

Skript: identify_usb_device.sh (2)

```
#!/bin/bash

readonly temp_dir=/tmp
readonly old_file=${temp_dir}/usbguard_rules_OLD.conf
readonly new_file=${temp_dir}/usbguard_rules_NEW.conf

( usbguard generate-policy; echo; echo ) >${old_file}
echo "connect device, then press ENTER"
read dummy
( usbguard generate-policy; echo; echo ) >${new_file}
diff ${old_file} ${new_file} | sed -e 's/^> //'
rm -f ${old_file} ${new_file}
```

Skript: identify_usb_device.sh (3)

- Erläuterungen zum Skript:

```
( usbguard generate-policy; echo; echo ) >${old_file}
```

- Die Klammern fassen mehrere Kommandos zusammen, deren Ausgabe wird dann gemeinsam mit `>` in die angegebene Datei umgeleitet.
- Das `; echo; echo` ist notwendig, weil `usbguard generate-policy` keinen Zeilenumbruch am Dateiende hinzufügt, und das bringt dann `diff` durcheinander.
- Das `sed -e 's/^> //'` entfernt Markierungen von `diff`.