Using USBGuard vs. UDev rules for USB device authorization

Daniel Kopeček dkopecek@redhat.com

What are the drawbacks when using UDev rules to implement USB device authorization?

- **Custom scripts needed for advanced features**. Users are forced to write custom scripts to handle advanced features like USB descriptor parsing, which is needed for analyzing USB interfaces of the device before they are allowed to interact with the system.
- **Maintenance and support complexity**. Maintenance of the scripts is complex, each user creates his own solution to the same problem.
- **Static Configuration**. Depending on the environment, the USB authorization policy may need to be dynamic. Using UDev, to be able to extend the policy means to have root access to the UDev configuration, because only then one can write a new rule and use the *udevadm* tool to apply it.
- **Security**. Most of the input on which a policy decision can be based is controlled by the USB devices and it cannot be fully trusted. Using custom script means that one usually has to make a compromise between the discriminative capabilities and the complexity of the script.

How is USBGuard solving these issues?

There's no need for custom scripts when using USBGuard because it implements a rule language which covers device matching and actions that need to be performed. However, when there's still something that should be performed using an external script, the rule language allows to specify an external script to be run.

Maintenance of customs scripts is reduced to the minimum if one still wants to use them. The rule set which defines the authorization policy is contained in one text file and can be modified either directly, or using CLI tools provided by

the USBGuard framework. Using the CLI tools, which are based on a public IPC interface to the daemon, one can implement dynamic policies.

USBGuard uses a best effort method for identifying USB devices from one another. The ability to uniquely identify a USB device depends also on the attribute values a device exports and USBGuard always uses all of the available attributes. The list of USB interfaces that a device provides cannot be faked in way that it would pass a properly written USBGuard policy restricting USB interfaces and then interact with the operating system using a forbidden interface. It handles USB device controlled inputs as untrusted, minimizing the risk of incorrectly authorizing a device or being controlled by a rogue USB device in unexpected ways.

For mitigation of successful exploitation of security bugs in the software, USBGuard uses a seccomp syscall whitelist and reduces it's process capabilities to a minimum.

It's possible to write third-party C++ applications that can interact with the daemon by using the USBGuard API provided in a shared library. The USBGuard Qt Applet is based on this library.

Example: Whitelisting a Yubikey device

UDev rules based solution

Example based on https://gist.github.com/grawity/52aad7d648735a236b0d.

`50-usb-deauthorize.rules` udev rules file:

```
ACTION!="add", GOTO="deauthorize_end"
SUBSYSTEM!="usb", GOTO="deauthorize_end"
TEST!="authorized", GOTO="deauthorize_end"

## make hubs deauthorize all devices by default
TEST=="authorized_default", ATTR{authorized_default}="0", GOTO="deauthorize_end"

## whitelist specific devices
ENV{ID_VENDOR}=="Yubico", ENV{ID_MODEL}=="Yubikey_NEO*", ENV{valid}="1"

## authorize matched devices, warn about the rest
ENV{valid}=="1", ENV{valid}="", ATTR{authorized}="1", GOTO="deauthorize_end"
RUN+="/usr/local/bin/usb-unauthorized $devpath"
LABEL="deauthorize_end"
```

The *usb-unauthorized* shell script can be viewed here: https://gist.github.com/grawity/52aad7d648735a236b0d#file-usb-unauthorized

USBGuard based solution

`/etc/usbguard/rules.conf`:

```
allow 1050:0010 serial "0001234567" name "Yubico Yubikey II" with-interface "03:01:01"
```

Note that the above rule also checks that the Yubikey interacts with the system by using only one USB HID (keyboard) interface.

The `authorized_default` flag state is handled automatically by the USBGuard daemon and it's configurable in `/etc/usbguard/usbguard-daemon.conf`.

NOTE: Logging actions or executing of scripts is currently not implemented. However, it's a feature planned to be completed in the first stable release, usbguard-1.0. Notification to desktop users can be displayed by the usbguard-applet-qt.