c't magazin für computer technik

Linux komfortabel und sicher entsperren

| 16.05.2019 09:26 Uhr Jürgen Schmidt

Man kann recht einfach eine Passwortabfrage durch eine Gesichtserkennung ersetzen. Doch damit das mit der Bildschirmsperre funktioniert, muss man etwas basteln.

Unter Ubuntu und bei den meisten anderen Linux-Distributionen regiert die Gnome Shell die Bedienoberfläche. Sie bestimmt, wer Zugang erhält. Dazu authentifiziert sie den Anwender gemäß den Einstellungen der Pluggable Authentication Modules (PAM) in /etc/pam.d/gdm-password. Diese Vorgaben kommen sowohl beim initialen Login als auch beim Entsperren des Monitors zum Einsatz. Somit kann man keine Änderungen am Entsperrvorgang vornehmen, ohne dass diese auch den Login betreffen.

Genau das ist jedoch mein Anliegen in dieser Mini-Serie "Hallo Linux": Sicherer Login mit langem Passwort, Entsperren nach komfortableren Vorgaben, etwa mit Gesichtserkennung und PIN. Wie man das in PAM umsetzt, erklärten die Artikel **"Linux: Anmelden per Gesichtserkennung" [1]** und **"Authentifizierung per PIN, Token und mehr" [2]** . Hier geht es konkret darum, wie man die Bildschirmsperre des Ubuntu-Desktops richtig einrichtet.

Um die Entsperrfunktion vom Passwort zu entkoppeln, muss man einen separaten Screensaver installieren. Die naheliegende Option ist der nach wie vor in **Ubuntu [3]** enthaltene Gnome Screensaver. Doch das ist nur noch ein Dummy, der zwar noch eine PAM-Datei gleichen Namens installiert, doch die wird ignoriert. Das Entsperren geschieht weiterhin gemäß `gdm-password`.

XScreenSaver als Ersatz

Als Ersatz empfehle ich XScreenSaver. Dessen Autor Jamie Zawinski ist kein Anfänger; sein Projekt war über viele Jahre die Basis für den Gnome Screensaver. Außerdem war er einer der Gründer von Netscape und später Mozilla und pflegt sein Screensaver-Projekt. Daher rührt vielleicht auch der stellenweise etwas altbackene 90er-Jahre-

Charme. Aber dafür kann er mit einer unvergleichlichen Auswahl aufwarten. Erinnern Sie sich noch an die Flying Toasters? Pacman? Matrix? Alles da …

LINUX KOMFORTABEL ENTSPERREN

- **Linux mit dem Gesicht entsperren [4]**
- **PIN, Gesichtserkennung, zweiter Faktor: Linux-Authentifizierung [5]**
- **Linux mit komfortabler Bildschirmsperre [6]**

XScreenSaver funktioniert nur auf X-basierten Desktops; Systeme mit Wayland wie Fedora Workstation bleiben hier außen vor. Die Installation erfolgt einfach via `apt install xscreensaver` oder wenn man alle Module haben will `xscreensaver-gl`. Den Bildschirmschoner beziehungsweise dessen Sperre dann tatsächlich zu aktivieren, erfordert allerdings noch etwas Handarbeit. Am einfachsten geht man dazu über die Anwendungsübersicht und gibt dort "startp" ein, um die App für Startprogramme zu öffnen.

LOGIN VERSUS ENTSPERREN

Der Linux-Desktop Gnome betrachtet das initiale Anmelden am System und das Entsperren einer bereits aktiven Sitzung als funktional identisch und lässt keine getrennte Konfiguration dieser Aktionen zu. Doch die beiden Aktionen haben sehr verschiedene Anforderungen – sowohl aus Anwender- als auch aus Security-Sicht.

Für das Anmelden kann man durchaus hohe Hürden vorsehen; schließlich macht man das nur recht selten. Und man hat keinerlei Ahnung, wer da gerade die Kontrolle über das Gerät hat. Ähnliches gilt übrigens für das Aufwachen aus den Standby-Modus. Das Entsperren etwa eines Arbeitsplatz-PCs muss der Anwender aber ständig und oft auch unter Beobachtung erledigen. Deshalb muss es flott von der Hand gehen. Dafür ist es für Dritte deutlich schwieriger, in diesem Kontext die volle Kontrolle über das Gerät zu bekommen. Er kann in der Regel weder die Festplatte ausbauen noch "mal eben schnell" ein fremdes Betriebssystem vom USB-Stick starten.

Dass die Sicherheit von einer solchen Trennung profitiert, demonstrierte Apple mit dem iPhone sehr eindrücklich. Ursprünglich hatten über die Hälfte der Besitzer aus Bequemlichkeit gar keinen Sperrcode für ihr Gerät aktiviert; wer das Handy in die Hände bekam, hatte sofort vollen Zugriff auf alles. Nach der Einführung von TouchID wurde das Entsperren schnell und komfortabel – ein Auflegen des Fingers genügte. Im Gegenzug waren die Anwender bereit, PIN-Codes einzurichten, die man jedoch nur sehr selten, etwa nach einem Neustart eingeben musste. Die Zahl der mit einem Code geschützten iPhones verdoppelte sich innerhalb kurzer Zeit und liegt mittlerweile bei deutlich über 90 Prozent.

Auch die Sicherheit von Desktop-Systemen profitiert von einer komfortablen Entsperrfunktion, nicht zuletzt, weil sie extralange Passwörter für den Login praktikabel macht. Es wäre schön, wenn die Gnome-Entwickler da mehr Flexibilität ermöglichen würden, sodass der Umweg über XScreenSaver und die damit verbundenen Hacks überflüssig wären.

Dort legt man einen neuen Eintrag an, der via `xscreensaver` die Server-Komponente des Screensavers zu Beginn jeder Sitzung im Hintergrund startet. Wer das bei jedem Start kurz eingeblendete Logo nicht sehen möchte, hängt noch ein `-no-splash` an. Über das Programm `xscreensaver-command` kann man dem Server dann Befehle wie `-lock`, `-deactivate` oder `-restart` senden.

Dann gilt es, den Gnome Screen Lock auszumustern. Dazu stellt man in den Systemeinstellungen unter `Einstellungen/Energie` "Bildschirm abschalten" auf "Nie" und unter `Datenschutz/Bildschirmsperre` die "Automatische Bildschirmsperre" auf "Aus". Das klingt beunruhigend, aber diese Aufgaben fallen ja nicht weg – XScreenSaver übernimmt sie stattdessen. Außerdem bleibt Gnomes Bildschirmsperre über das Anwender-Menü erreichbar und sichert auch nach wie vor das Aufwachen aus dem Suspend-Modus ab.

Tastenkombination für Bildschirmsperre

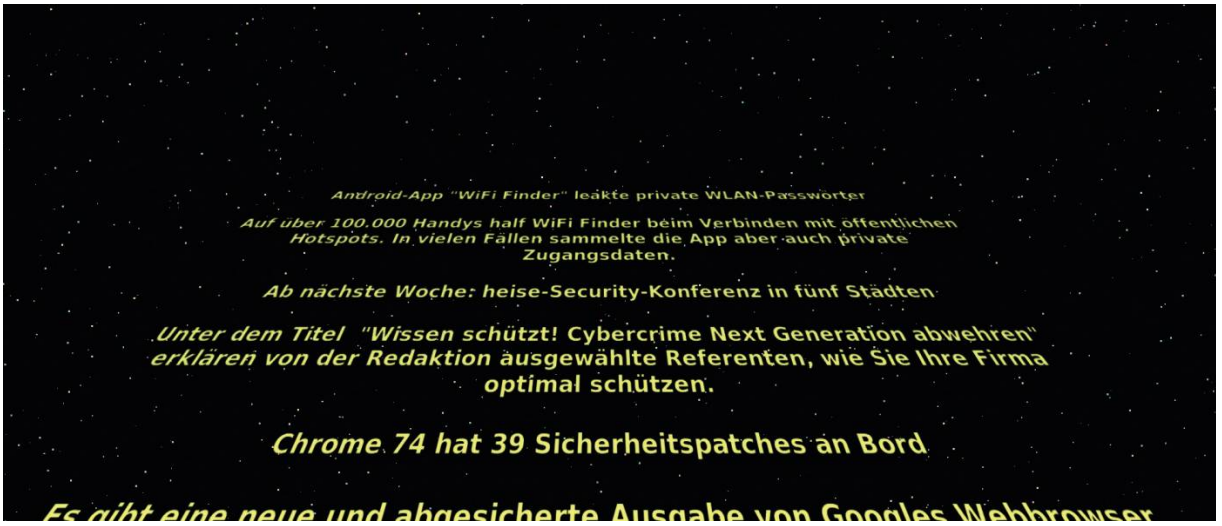
Wenn man schon dabei ist, sollte man unter `Geräte/Tastatur` den Hotkey Super-L (also Windows-Taste + L) freigeben, indem man für "Bildschirm sperren" eine andere Tastenkombination wie Super-I vergibt. Anschließend kann man über `+` ein neues Kommando definieren und Super-L auf `xscreensaver-command -lock` legen. Das geht aber erst, wenn man die Tastenkombination vorher auch wirklich freigegeben hat.

Die Tastenkombination "Super-L" löst die neue Bildschirmsperre aus.

Damit hat man den Gnome ScreenLock bereits durch einen Screensaver mit allen Schikanen ersetzt, der jedoch noch etwas Feinschliff benötigt. Als erstes sollte man mit dem Konfigurations-Programm `xscreensaver-demo` die gewünschte Zeit bis zum Aktivieren des Bildschirmschoners einstellen ("Schwärzen nach"). Kürzer ist besser; in meinem Alltag komme ich mit 3 Minuten gut zurecht. Das sperrt den Bildschirm aber noch nicht; jegliche Benutzeraktivität bringt die geöffneten Fenster wieder zum Vorschein.

Erst der Haken bei "Bildschirm sperren nach" aktiviert die Sperrfunktion. Die folgende Zeitdauer ermöglicht einen Versatz zum Aktivieren des Bildschirmschoners, den man auf 0 setzen kann. Ich finde es aber ganz angenehm, das automatische Sperren noch eine Minute lang durch einen Tastendruck vermeiden zu können.

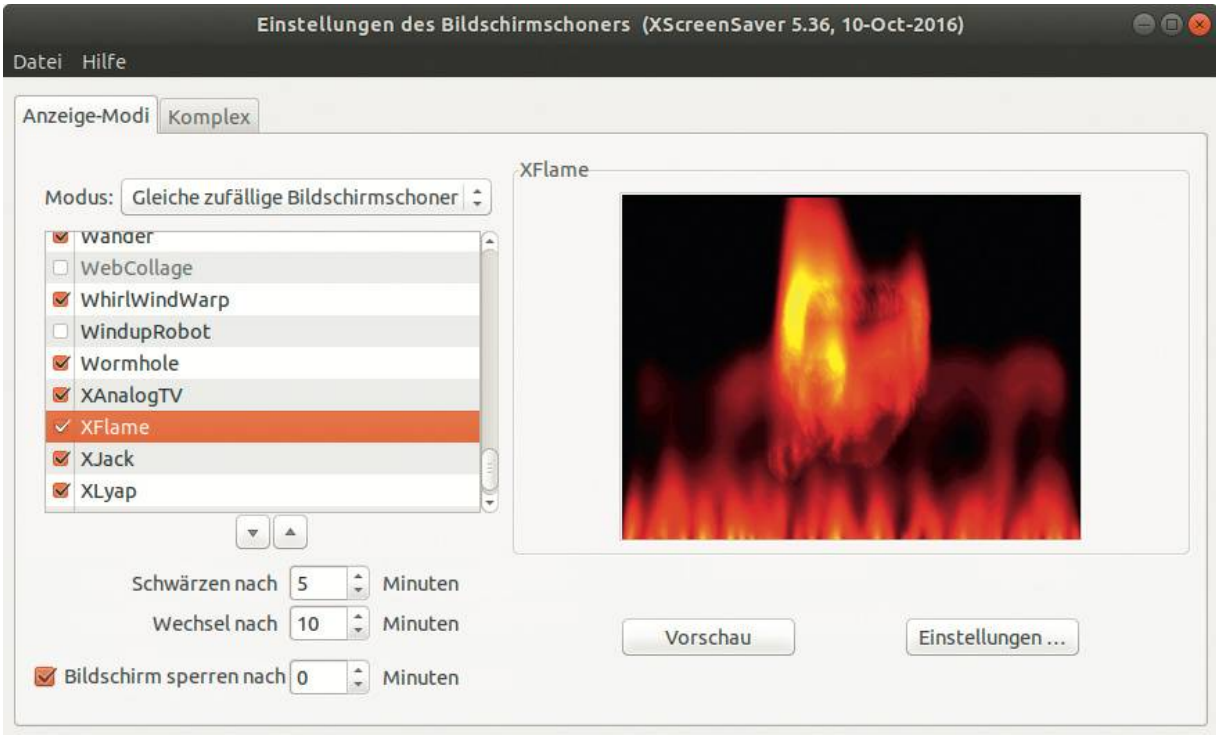
Darüber hinaus finden sich hier auch ein paar weitergehende Einstellungen. Mich machte es ganz wuschig, dass auf jedem meiner drei Monitore ein anderer Screensaver lief. Das beseitigte der Modus: "Gleiche zufällige Bildschirmschoner". Bei aktuellen Displays ist das "Schonen" des Bildschirms durch wechselnde Muster, die das Einbrennen verhindern, überflüssig. Wer die dafür verbrauchte Energie sparen will, stellt das lieber auf "Nur den Bildschirm schwärzen". Außerdem kann man unter "Komplex" auch gleich die Energieverwaltung aktivieren, damit der Monitor nach angemessener Zeit in den passenden Modus wechselt (auf Notebooks sollte man davon jedoch lieber die Finger lassen, weil das deren Energieverwaltung besser steuert). Man kann hier auch ein Verzeichnis für Bilder oder Texte angeben, die manche Module für die angezeigten Effekte nutzen. Den optionalen RSS-Feed lässt das Star-Wars-Modul im Stil des Intros nach hinten weglaufen.



Mit XScreenSaver kein Problem: Der RSS-Feed von heise Security im Stil des Star-Wars-Intros.

200 Meisterwerke

Ich kann nur jeder und jedem ans Herz legen, sich ein paar Minuten oder auch Stunden Zeit zu nehmen und durch die Vorschau-Darstellungen der weit über zweihundert Module zu streifen. Manche haben geradezu hypnotische Wirkung, andere demonstrieren verblüffende Effekte wie das langsame Zerfließen eines gerade mit der Webcam geschossenen Fotos. Man kann sich letztlich auf einen festlegen oder sich jedes Mal von einem zufällig ausgewählten Screensaver-Motiv überraschen lassen. Man sollte allerdings beachten, dass einige der Module mit Screenshots der aktuellen Desktop-Oberfläche arbeiten und somit vertrauliche Informationen präsentieren könnten.



XScreenSaver bietet weit über zweihundert Screensaver-Module zur Auswahl.

All diese Einstellungen gelten immer nur für den aktuell angemeldeten Benutzer. Man muss also im Zweifelsfall die gesamte Prozedur, angefangen vom Startup-Eintrag über Tastenkürzel bis zur XScreenSaver-Konfiguration, für alle Anwender wiederholen oder die jeweils relevanten Konfig-Dateien lokalisieren und kopieren. Auf anderen Systemen als **Ubuntu Desktop 18.04 [7]** sind die einzelnen Handgriffe eventuell anders, doch das Prinzip bleibt das gleiche. Der XScreenSaver-Autor gibt auch selbst diverse Tipps für andere Desktops wie KDE.

[8]

Entsperren über separate PAM-Konfiguration

Jenseits all der Spielereien bietet XScreenSaver einen ganz entscheidenden Vorteil: Er ermöglicht das Entsperren über eine separate PAM-Konfiguration und man kann jetzt via /etc/pam.d/xscreensaver komplett kontrollieren, wie das zu geschehen hat. So kann man **etwa eine Gesichtserkennung mit einem kurzen PIN-Code kombinieren [9]** , um nicht bei jeder kleinen Pause sein langes, sicheres Passwort eingeben zu müssen. **Es gibt außerdem diverse Beispielkonfigurationen [10]**, aus denen man den persönlichen Favoriten auswählen kann, den man einfach anstelle von `common-auth` einträgt. Für alle Konfigurationen gilt: Im Zweifelsfall funktioniert als Fallback immer noch das Login-Passwort.

Nur an einer Stelle gilt es aufzupassen: Der XScreenSaver-Prozess startet die Authentifizierung mit den reduzierten Rechten eines Anwenders. Damit hat das PAM-Modul `pam_userdb` aber keinen Zugriff auf die Liste der verschlüsselten PIN-Codes, die nur Root lesen darf. Man kann das umgehen, indem man die Datei für alle Benutzer des Systems lesbar macht

```
chmod a+r /etc/pinlist.db
```

Aber dann sollte man die PINs auf gar keinen Fall auch benutzen, um etwa via sudo den Zugang zu Root-Rechten zu gestatten. Denn die PIN-Codes sind zwar nach dem gleichen Verfahren wie die Systempasswörter verschlüsselt, aber ein Nutzer oder auch ein Trojaner, der die Hashes hat, kann die möglichen Codes leicht alle durchprobieren. Die eine Million Kombinationen einer sechsstelligen PIN sind da keine ernsthafte Hürde. Das ist ein Makel, für den ich noch keine wirklich befriedigende Lösung gefunden habe. Man kann jedoch damit leben, solange ein Angreifer durch das Knacken der Hashes keine höheren Rechte bekommt als die, die er bereits zum Lesen der Hashes benötigt. Das ist bei einer Konfiguration, in der die PIN nur in Kombination etwa mit einem U2F-Token den Bildschirm entsperrt, durchaus gegeben.

Ein weiteres, noch nicht ganz befriedigend gelöstes Problem ist die Anforderung, dass sich ein anderer Anwender am System anmelden möchte, während der Bildschirm gesperrt ist. XScreenSaver bietet dafür zwar einen Button namens "New Login" – doch die dahinterstehende Systemlogik existiert beim aktuellen Gnome nicht mehr. Behelfsmäßig kann man in die Datei ~/.Xresources etwas wie

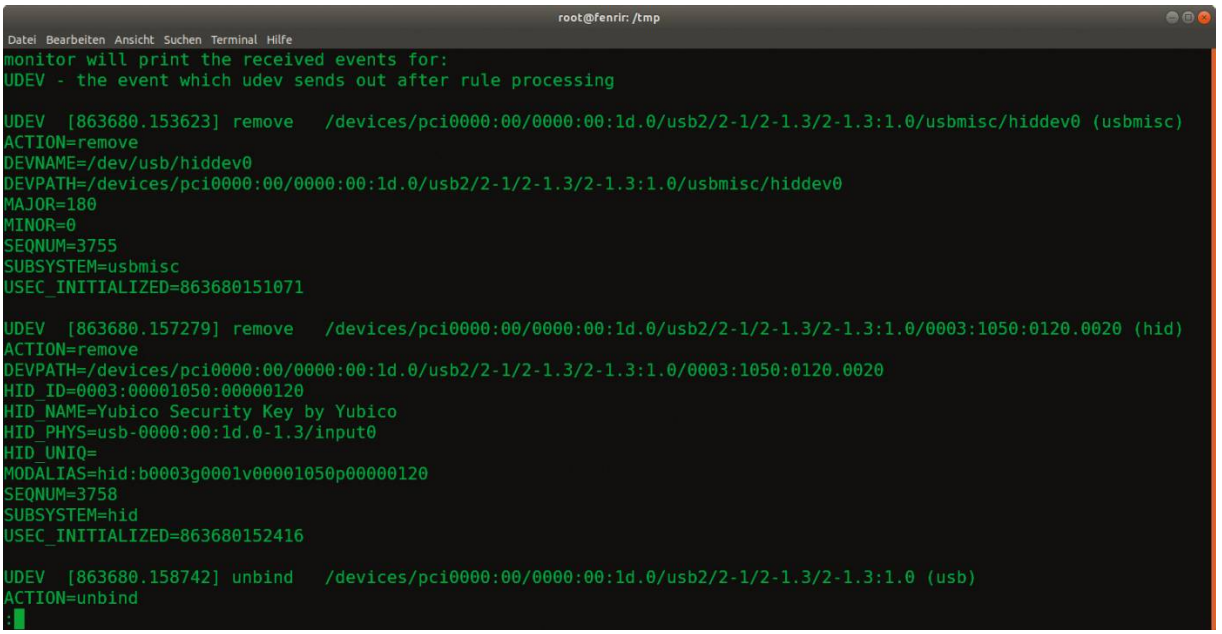
```
xscreensaver.newLoginCommand: gnome-session-quit --logout --force --no-prompt
```

schreiben. Damit beendet der Button ohne viel Federlesen die aktuelle Sitzung und präsentiert einen neuen Login-Screen. Dass jeder, der vorbeikommt, mich einfach komplett Ausloggen kann, ist nicht schön. Aber er könnte auch den Ausschalter betätigen – und dann lieber so. Wer das nicht will, belässt es bei der funktionslosen Default-Einstellung. Ich bin da durchaus offen für bessere Lösungen.

Entsperren mit Token

Am sichersten und einfachsten geht das Entsperren mit U2F-Token wie dem YubiKey. Ein kleiner Schubs an der Maus und schon blinkt das Token einladend und eine bestätigende Berührung des Tokens erledigt die Authentifizierung. Schön wäre es, wenn sich das noch weiter automatisieren ließe – also bereits das Anstecken das Entsperren auslöst und vor allem das Abziehen des Tokens den Rechner sofort sperrt.

Dank udev ist das alles kein Hexenwerk. Das flexible Geräte-Management erlaubt es, eigene Regeln zu definieren, die Aktionen beim Anstecken oder Entfernen beliebiger Hardware auslösen. Das Thema ist zwar komplex, denn zu udev könnte man mehrere eigene Artikel schreiben. Doch mit der folgenden, kompakten Anleitung sollten Sie in der Lage sein, einfache eigene Regeln zu erstellen.



Der Gerätemanager Udev generiert beim Anstecken und Entfernen von Devices Ereignisse, auf die man reagieren kann.

Das zentrale Verzeichnis dafür ist /etc/udev/rules.d/. Dort kann man eine Datei namens 80-yubikey.rules anlegen, die Events wie "U2F-Token wurde entfernt" filtert und daraufhin etwa ein Skript startet, das den Bildschirm sperrt. Natürlich kann man das sehr allgemein nutzen und sich auch Regeln basteln, die beim Erkennen eines ganz bestimmten USB-Sticks eine Fanfare abspielen und besonders wichtige Dateien sichern.

Das Problem dabei ist, dass die meisten Anleitungen im Internet, die beschreiben, wie man eigene udev-Regeln einrichtet, bei mir in der Praxis gescheitert sind. Die beschreiben im Wesentlichen, dass man sich zunächst mit `lsusb` einen Überblick über alle verfügbaren Devices verschafft

```
Bus 001 Device 022: ID 1050:0120 Yubikey Touch U2F Security Key
Bus 001 Device 005: ID 046d:c52b Logitech Unifying Receiver
```

und sich dann etwa mit

```
lsusb -vd 1050:0120
```

konkrete Infos zum gewünschten Gerät besorgt. Das ergibt eine lange Liste mit Daten wie

```
idProduct          0x0120 Yubikey Touch U2F Security Key
```

nach denen man prinzipiell suchen könnte. Allerdings schlugen meine diesbezüglichen Versuche fehl, obwohl die Regeln eigentlich richtig waren. Als Ursache stellte sich heraus, dass udev beim Entfernen eines Geräts nur einen kleinen Teil dieser Infos tatsächlich bereitstellt. Das Attribut `idProduct` etwa gehörte nicht dazu. Ähnliches gilt, wenn man sich mit dem Tool `udevadm` Infos zum noch aktiven Gerät anzeigen lässt: Man weiß nie, welche der vielen Datenfelder beim Entfernen tatsächlich sichtbar sind.

Deshalb habe ich eine etwas allgemeinere Vorgehensweise entwickelt, die sich auf alle Geräte übertragen lässt. Dabei protokolliert man kurzzeitig alle udev-Ereignisse mit

```
udevadm monitor -u -e | tee /tmp/udev-remove
```

und löst dann das gewünschte Event aus – zieht also beispielsweise das U2F-Token vom USB-Port ab. Aus der langen Ausgabe in /tmp/udev-remove sucht man sich dann ein Event aus, das ausreichende Informationen für einen passenden Filter liefert und nur einmal vorkommt. Bei mir war das

```
DEV  [588756.639688] remove   ...
ACTION=remove
HID_ID=0003:00001050:00000120
HID_NAME=Yubico Security Key by Yubico
SUBSYSTEM=hid
```

was ich zu folgender udev-Regel in /etc/udev/rules.d/80-Yubikey zusammengefasst habe

```
SUBSYSTEM=="hid",
ACTION=="remove",
ENV{HID_NAME}=="Yubico Security Key by Yubico",
RUN+="/usr/local/bin/dev_removed.sh"
```

Die startet das lokale Skript `dev_removed.sh` beim Entfernen des U2F-Tokens von Yubico. Analog habe ich auch ein `add-Event` mit `device_added.sh` verknüpft und beide Skripte via `chmod a+x` ausführbar gemacht. Die **Regel-Datei [11]** und auch die von mir eingesetzten **Skripte zum Entsperren [12]** und **zum Sperren [13]** können Sie von GitHub runterladen.

Benutzerkontext hinzufügen

Die Skripte müssen allerdings noch eine letzte Klippe umschiffen, bevor alles so läuft, wie man es sich vorstellt. Denn der Geräte-Manager ruft das Skript mit Root-Rechten ohne jeglichen Benutzerkontext auf. So kann es

jedoch nicht mit dem XScreenSaver reden. Dazu muss das Skript zunächst ermitteln, wem die aktuelle X-Sitzung gehört und dessen Identität übernehmen. Das erledigt die Befehlsfolge

```
user=`who | grep ':0' | awk '{print $1}'| head -1`
```

Damit kann dann

```
DISPLAY=:0 sudo -n -H -u "$user"/usr/bin/xscreensaver-command -lock
```

das Sperrkommando absetzen. Das funktioniert für meine Zwecke bereits sehr gut. Für den Produktionsbetrieb sollte man die Variable `$user` vor der Verwendung noch prüfen. In einem Szenario, in dem mehrere Anwender X-Sitzungen haben und Displays wie `:1` genutzt werden, müsste man noch etwas mehr Aufwand betreiben, den ich mir erspart habe.

Endlich fertig

Damit sind alle Komponenten für ein sicheres und doch komfortables Entsperren beisammen. Das vorgestellte Konzept könnte durchaus noch weiter gehen und etwa Sicherungen einbauen, die nach drei Fehlversuchen doch eine stärkere Authentifizierung, sprich das Login-Passwort einfordern. Auch das Einbeziehen von Security-Funktionen der Hardware (Trusted Platform Module, TPM), um Geheimnisse besser zu schützen, wäre überaus wünschenswert. Microsoft ist da mit "Windows Hello" deutlich weiter. Doch schon in der jetzigen Form ist das vorgestellte "Hallo Linux" ein deutlicher Schritt in die Richtung mehr Security durch mehr Komfort.

Mein bevorzugtes Authentifizierungsverfahren ist das U2F-Token, das sowohl für `sudo` als auch das Entsperren des XScreenSaver zum Einsatz kommt. Natürlich immer mit der Fallback-Option Passwort, wie es im ersten Teil der Mini-Serie beschrieben ist. Das U2F-Verfahren ist so sicher, dass man es im Prinzip allein einsetzen kann. Doch im Büro lasse ich den Schlüsselbund mit dem Token öfter mal am Rechner, wenn ich den Raum nur kurz verlasse. Um den Rechner auch dann nicht völlig offen zu lassen, erfordert das Entsperren zusätzlich mein Gesicht vor der Windows-Hello-kompatiblen Kamera. Das entspricht der PAM-Konfiguration `inc-u2f+howdy-pw` aus **der Sammlung der PAM-Skripte [14]**. Jetzt juckt es mich in den Fingern, auch das Entsperren der LUKS-verschlüsselten Datenträger und den initialen Login mit dem U2F-Token zu kombinieren. Vielleicht gibt es ja dazu irgendwann eine Fortsetzung. (**ju [15]**)

URL dieses Artikels:

https://www.heise.de/-4418407

Links in diesem Artikel:

[1] https://www.heise.de/ratgeber/Linux-mit-dem-Gesicht-entsperren-4404861.html

[2] https://www.heise.de/ratgeber/PIN-Gesichtserkennung-zweiter-Faktor-Komfortable-Linux-Authentifizierung-4404929.html

[3] https://www.heise.de/thema/Ubuntu

[4] https://www.heise.de/ratgeber/Linux-mit-dem-Gesicht-entsperren-4404861.html

[5] https://www.heise.de/ratgeber/PIN-Gesichtserkennung-zweiter-Faktor-Komfortable-Linux-Authentifizierung-4404929.html

[6] https://www.heise.de/ratgeber/Linux-komfortabel-und-sicher-entsperren-4418407.html

[7] https://www.heise.de/download/product/ubuntu-22040

[8] https://www.heise.de/ct/

[9] https://www.heise.de/ratgeber/Linux-mit-dem-Gesicht-entsperren-4404861.html

[10] https://github.com/ju916/pam-demos

[11] https://gist.github.com/ju916/8f1267b43edfbd7be5127eb24045546f

[12] https://gist.github.com/ju916/60990308b696007e4941f6a85d829e8f

[13] https://gist.github.com/ju916/e85629890d50b4e31f64f3620e930b45#file-device_removed-sh

[14] https://github.com/ju916/pam-demos

[15] mailto:ju@ct.de