Sicher(re)es Linux-Desktop OS

Übersicht

- Motherboard / BIOS / Secure Boot
- Sicheres Installationsmedium
- Verschlüssellung
- Auswahl von Laufwerken
- Betriebssystem und Software
- Sandboxing
- Fazit

Motherboard

- TPM 2.0 Modul
  - Ermöglicht Secure Boot
  - Kann Schlüssel generieren und speichern
  - Kann Daten verschlüsseln und signieren
  - Von aktuellen Kernels als `/dev/tpm0 eingebunden`
- `Intel SGX`
  - `Sichere Ausführungsumgebung im Prozessor`

BIOS

- Einstellung der Boot-Reihenfolge
- Weitere Sicherheitseinstellungen
  - „Internal Storage Tamper Detection“
  - Komische Microsoft Dienste
- Verschlüssellungspassworte für SED
- Passwortschutz des BIOS und des Bootvorgangs
- Unterstützt Secure Boot

BIOS Passwort

- „Supervisor Passwort“ sperrt Zugriff auf bestimmte BIOS Einstellungen
  - Optional alle Einstellungen / Auswahl von Booteinträgen / Booten selbst
- Online viele Umgehungsmöglichkeiten zu finden
  - 3 x Falsch → Masterpasswort (findet sich im Netz)
  - CMOS-Batterie vorübergehend entfernen
  - Security Chip auf Mainboard kurzschließen
- Lenovo Support: Kein Masterpasswort, CMOS beeinträchtigt Passworte im TPM-Chip nicht, Kurzschluss macht Mainboard kaputt

Secure Boot

- Vor dem Booten wird Signatur des Bootimages geprüft
- Ursprünglich nur für Windows eingeführt
  → Microsoft ist Certificate Authority
- Microsoft signiert auch fremde Images
- Bei Linux meißt kleiner Bootloader „shim“
  - Überprüft Kernel- und Kernelmodulsignaturen
  - Eigene Public-Keys können eingetragen werden
    - Explizite Bestätigung beim Booten, dann speichern in NVRAM

Sicheres Installationsmedium

- Vertrauen in alle beteiligten Geräte nötig
- Bisheriges OS kontaminiert → Installer kontaminiert
- Sicherungsidee: USB-Stick mit Schreibschutzschalter, Checksumme auf mehreren Rechnern

  → Es muss nur einer die Wahrheit sagen!

Bootloader

- Standardmäßig GRUB
  - Booteinträge modifizierbar & Terminalzugriff
  - Sollte durch Passwort geschützt werden

  → Dann aber mehrere Passworteingaben beim Booten
- Mit UEFI müsste es auch ohne Bootloader gehen
  - Keine Zeit :(

Verschlüssellung

- Moderne SSDs sind oft SED (AES-256)
- Schlüssel aus dem BIOS mit Passwort verschlüsselbar
  - Verfahren ist Hersteller*innenabhängig aber bekannt
    - → Keine zusätzliche Sicherheit aber Inkompatibilität
- In der Kritik wegen Backdoors und unsauberen Implementierungen

Sicheres? Wipen

- ATA-Anweisung an die SSD, neuen Schlüssel zu generieren
- `sedutil-cli --initialsetup debug <Drive>`
- `sedutil-cli --yesIreallywanttoERASEALLmydatausingthePSID <PSID> <Drive>`
- „Physical Security Identification“ auf SSD

Linux Unified Key Setup

- Nutzt dm-crypt Kernelmodul für transparente Verschlüssellungsschicht
- Header mit Passwortslots und Masterkey
  - Bei SSDs nicht zuverlässig überschreibbar!
    - → Komplett neuverschlüssel bei Kompomittierung
  - Besser: Header auf HDD auslagern
- Bei Debian-Installation auswählbar

Qubes OS

- Alle Applikationen in virtuellen Maschienen
  - Unterschiedliche OSs möglich (Debian, Windows 7, ...)
- „Disposable VMs“ für genau eine Laufzeit
- VPN- bzw. Tor-Tunnelling pro VM einstellbar
- USB-Geräte explizit VM zuzuweisen
- Umständlich und Ressourcenaufwändig

  → Zu faul ¯_(ツ)_/¯

Debian

- Eine der stabilsten Distros
- Alle Bugs sofort öffentlich gelistet und schnell gepatched
- „Security Audit“-Team sucht aktiv Sicherheitslücken
  - Kann nicht alle Pakete überprüfen
- „Debian Security Manual“ & „Secure Personal Computer“
  - Auswahl von Services, Software, Rechtevergabe, etc.
  - Firewall, Integritätschecks und Anti-Malware-Tools
  - Kernelparameter (Restriktivere Netzwerkkonfiguration)
  - Sandboxing (Firejail, Virtuelle Maschienen)
  - „debsecan“ listet installierte Programme mit bekannten Lücken

OS Konfiguration

- Erst mal alle Konfigurationsdateien rüberkopieren…
  - ~/.bashrc, ~/.xsessionrc, ~/.config/init/init.vim, ...

  → Kann bereits Schadcode enthalten
- Idee: Git-Repository mit Konfigurationsdateien
  - Schnelle Einrichtung, Änderungen nachvollziehbar
  - Enthält Script für automatisches verlinken

Sandboxing

- Firejail (konfigurierbare Sandbox für Programme)
- X11-Sandboxing via Xpra, OverlayFS für COW, AppArmor für Netzwerk- & Dateizugriffsrestriktionen
- Kommt mit Profilen für viele Anwendungen
  - Z. B. firefox darf nur ~/.mozilla und ~/Downloads
  - Auch root shell in OverlayFS möglich um Programminstallation zu testen
  - Mein Plan: PDF-Viewer und LibreOffice sandboxen
- AppArmor: Kernelsupport für Netzwerk-, Dateizugriffsbeschränkung etc.
  - Installierte Pakete platzieren Profile in /etc/apparmor.d/
- Auch komplette VM nutzbar

USB Guard

- White- / Blacklisting von USB-Geräten
- Verwendet USB Authorisierungsfunktion des Kernels
- GUI-Frontent mit Popups für einfache Bedienung
- Betrifft auch eingebaute Webcam
- Keine Anzeige / Beschränkung von Gerätearten
  - Plan: Feature Request stellen
- Nützlich fürs Aufladen & zur Sensibilisierung

Sicheres Backup

- Verschlüsselte Daten schnell komplett verloren
- Vollständiges Festplattenbackup
  - Vollständig aber dauert lange
- Partclone + GZip für komprimiertes Blocklevel-Backup
  - Bringt bei verschlüsselter Platte überhaupt nichts
  - → Backup der entschlüsselten Partitionen und der Partitionsstruktur
  - Danach entweder durch gpg pipen oder auf verschlüsselte Platte
  - Von separater Partition (z. B. USB-Stick) für „stehendes“ Backup
- Inkrementelles Backup des Home-Directories
  - Schlaues Script fürs automatische Löschen alter Backups

Sichere Authentifizierung

- Am wichtigsten: Verschlüssellung der Daten
  - Von Schraubenzieherbesitzer*in geklaut…
  - Multi-Faktor-Authentifizierung schwierig
- Sicheres Passwort (> 18 zufällige Zeichen)
  - Kann mit normaler Kamera einfach gestohlen werden
  - Sichtschutz für die Eingabe

Zusammenfassung

- BIOS-Passwort
  - Sollte ohne Hardwareeingriffe das Booten fremder Systeme verhindern können
- Secure Boot
  - Wer Schreibzugriff auf die SSD bekommt, kann trotzdem Kernel oder Bootloader infizieren
- Installationsmedium mit Schreibschutz
  - Sollte Integrität des Installationsimages sicherstellen

Zusammenfassung 2

- SSD mit Verschlüssellungsheader
  - Alte Passwörter dürfen nicht kompromittiert werden
- Qubes hat viele Sicherheitsfeatures
- Debian ist auch ganz OK
- Es gibt einiges an Sandboxingfeatures
  - Nicht sicher vor Ausbrüchen

Zusammenfassung 3

- USB Guard erlaubt Kontrolle über USB-Geräte
  - Wichtige features fehlen noch
- Sichere Authentifizierung
  - Passwörter abgucken verhindern

Fazit

- Wenn jemand das Gerät in der Hand hatte ist alles verloren
- Alles unübersichtlich und fehleranfällig wegen Altlasten
- Übersichtliche und zuverlässige Repräsentation des Systemzustands wäre schön
- Closed-Source Hardware sowieso bedenklich

Quellen

- https://www.cocosenor.com/articles/computer/3-ways-to-unlock-bios-password-on-lenovo-thinkpad-laptop.html
- https://www.howtogeek.com/116569/htg-explains-how-windows-8s-secure-boot-feature-works-what-it-means-for-linux/
- https://ubuntu.com/blog/how-to-sign-things-for-secure-boot
- http://cdimage.debian.org/debian-cd/current/amd64/iso-cd/
- https://jbeekman.nl/blog/2015/03/lenovo-thinkpad-hdd-password/
- https://threatpost.com/academics-find-critical-flaws-in-self-encrypting-hardware-drives/115103/
- https://gitlab.com/cryptsetup/cryptsetup
- https://en.wikipedia.org/wiki/Initial_ramdisk
- https://www.cvedetails.com/cve/CVE-2016-4484/
- https://www.youtube.com/watch?v=f4U8YbXKwog
- https://fossbytes.com/secure-linux-distros-privacy-anonymity/
- https://www.educba.com/centos-vs-debian/
- https://www.debian.org/security/audit/
- https://lists.debian.org/debian-security-announce/
- https://www.debian.org/doc/user-manuals#securing
- https://linuxsecurity.com/docs/QuickRefCard.pdf
- https://wiki.debian.org/SetupGuides/SecurePersonalComputer
- https://wiki.debian.org/AppArmor/HowToUse
- https://usbguard.github.io/