LINUX HOST SECURITY

GRUNDLAGEN UND LESSONS LEARNED

// WHOAMI

> Christian Stankowic

> Berater und Trainer @ SVA

> Linux, Virtualization, IaC, Automation

> moderiert den FOCUS ON: Linux Podcast 🎙️

> https://cstan.io 📖

// AGENDA

- Motivation
- Grundlagen
- Weitere Tools
- Dev-Sec

// MOTIVATION

MOTIVATION

- Linux-Systeme werden im Server-Bereich immer **populärer***
  - 80% der Webserver werden unter Linux betrieben
  - mehr als 50% der Azure-Workloads sind Linux-Workloads
  - **IoT**-Technologie ist i.d.R. Linux-basiert

MOTIVATION

- Linux-Systeme werden im Server-Bereich immer **populärer***
  - 80% der Webserver werden unter Linux betrieben
  - mehr als 50% der Azure-Workloads sind Linux-Workloads
  - **IoT**-Technologie ist i.d.R. Linux-basiert
- Dadurch steigt das Interesse für Angreifende
- Standard-Installationen sind leider i.d.R. **nicht sicher**
  - **Trade-off**: Sicherheit oder Komfort

* das Jahr des Linux-Desktops kommt sicher noch

MOTIVATION

- Glücklicherweise gibt es zahlreiche Möglichkeiten Linux zu härten, u.a.:
  - Sinnvolle Grundeinstellungen
  - **SELinux** und AppArmor
  - Automatisierte **Härtung** nach Best Practices
  - Automatisches Sperren von IPs
- Ausgiebiges Testen ist **unabdingbar**

// GRUNDLAGEN

SECURITY 101 - PAKETAUSWAHL

> *weniger ist mehr*
  > minimale Paketauswahl
  > keine **GUI** auf Servern installieren
  > nicht benötigte Pakete deinstallieren

> keine Pakete aus nicht **vertrauenswürdigen** Quellen

> Regelmäßig **Updates** installieren
  > `apt unattended-upgrades`
  > `yum-cron` / `dnf-automatic`

SECURITY 101

FIREWALL

> Firewall **nicht** deaktivieren

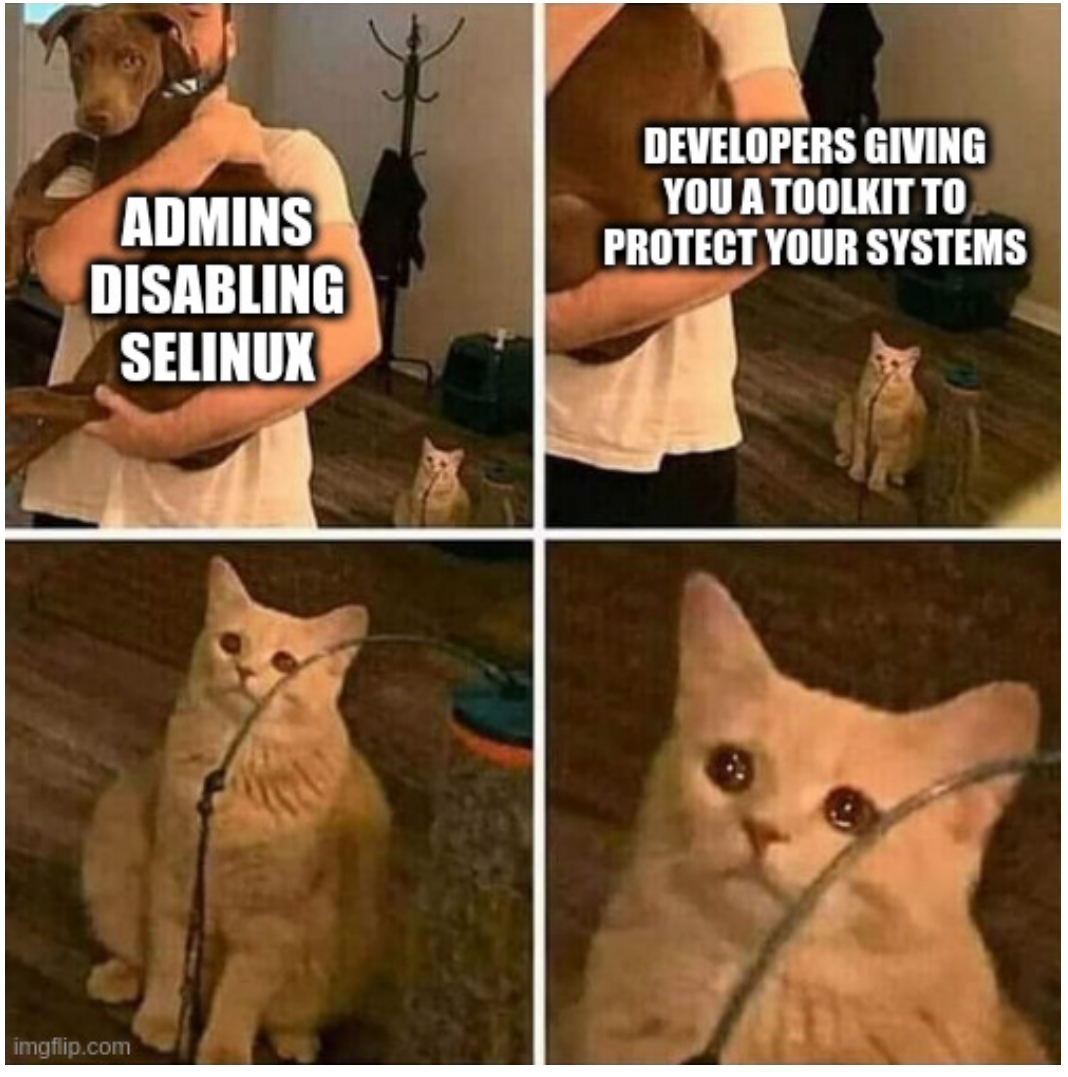
> sinnhaftige Konfiguration vornehmen

> Zonen-Konfiugration überprüfen

SELINUX

> **Nicht** deaktivieren

> Im Fehlerfall in **Permissive**-Modus versetzen und Fehler untersuchen



SECURITY 101

DATEISYSTEM

- niemals `chmod 0777` definieren
- **ACLs** einsetzen wo klassische Berechtigungen nicht ausreichen
- Einsatz von **setuid** und **setgid** vermeiden
- Dedizierte Partitionen einsetzen (*LVM oder Btrfs*)*
  - `/var`
  - `/home`
  - `/opt`

* verhindert Fehler durch erschöpften Speicherplatz

SECURITY 101

PASSWÖRTER

- **Aging** und **Länge** konfigurieren (`/etc/login.defs`)
  - `PASS_MIN_DAYS`, `PASS_MAX_DAYS`
  - `PASS_MIN_LEN`, `PASS_MAX_LEN`
- **Komplexität** konfigurieren (`/etc/security/pwquality.conf`)
- LDAPS statt LDAP einsetzen

SECURITY 101

SSH

> Root-Zugang deaktivieren (`PermitRootLogin no`)

> alte Chiffren deaktivieren

> Public Key-Authentifizierung aktivieren (`PubkeyAuthentication yes`)

> Passwort-Authentifizierung deaktivieren (`PasswordAuthentication no`)

> Leere Passwörter deaktivieren (`PermitEmptyPasswords no`)

> TCP-Forwarding/-Tunneling deaktivieren (`AllowTcpForwarding no`, `PermitTunnel no`)

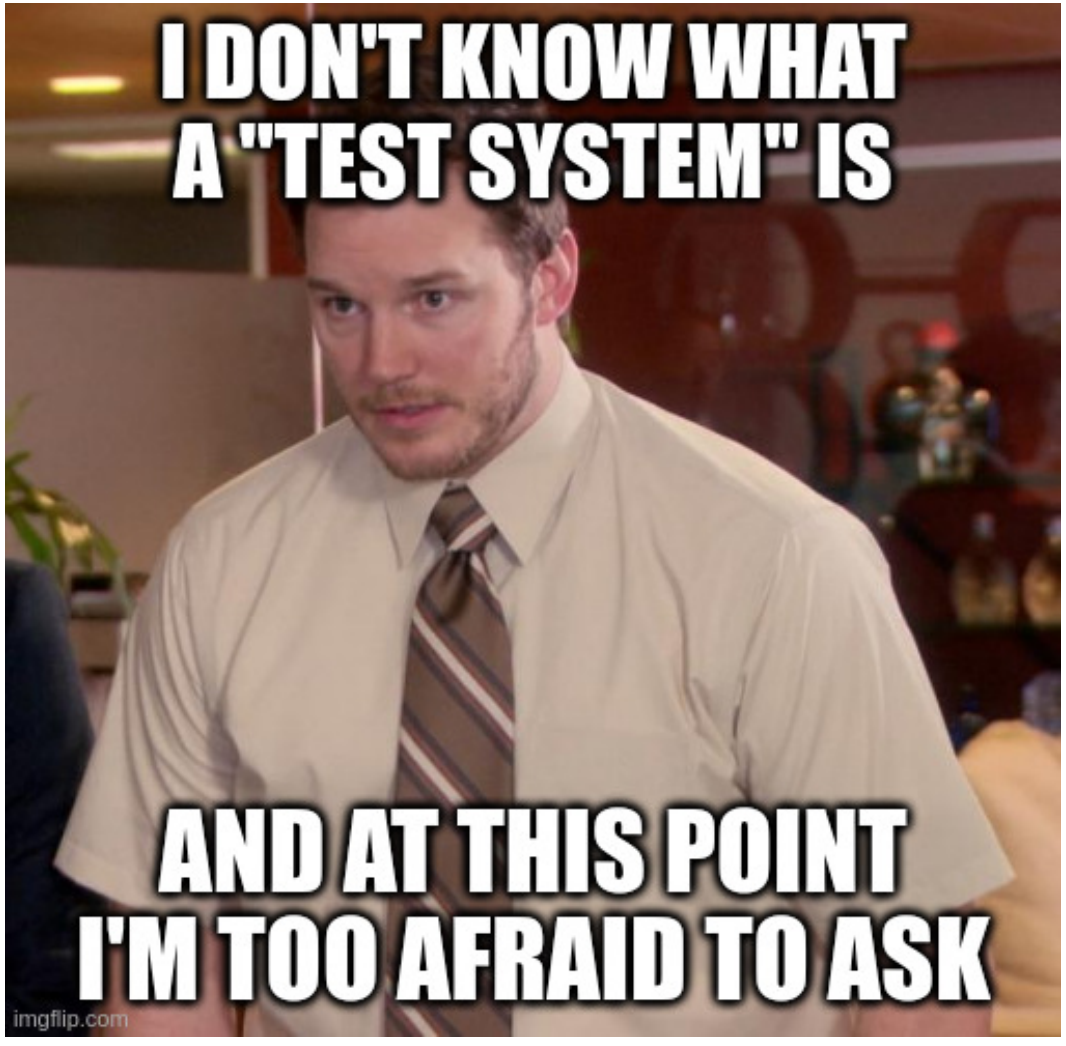
SECURITY 101

SERVERDIENSTE

> OS-/Anwendungsversionen verstecken (*`Banner`*, *`ServerTokens`*)

> Regelmäßige Backups konfigurieren und **Restores** testen

> **Testsysteme** bereitstellen



EXKURS: EXPLOIT-DATENBANKEN

- Server-Dienste geben i.d.R. **mehr** Informationen preis **als notwendig**
  - z.B. benutzte Linux-Distribution und Server-Software
  - **Debug**-Ausgaben von Web-Anwendungen
- In öffentlichen Umgebungen ergibt es Sinn, dieses Verhalten **abzustellen**

EXKURS: EXPLOIT-DATENBANKEN

- Server-Dienste geben i.d.R. **mehr** Informationen preis **als notwendig**
  - z.B. benutzte Linux-Distribution und Server-Software
  - **Debug**-Ausgaben von Web-Anwendungen
- In öffentlichen Umgebungen ergibt es Sinn, dieses Verhalten **abzustellen**
- Im Internet kursieren mehrere Datenbanken mit funktionalen **Exploits**:
  - Offensive Security Exploit Database
  - Rapid7 Vulnerability and Exploit Database
  - VulDB - dokumentiert **seit 1970** Verwundbarkeiten
  - CXSecurity

// WEITERE TOOLS

AIDE

> **A**dvanced **I**ntrusion **D**etection **E**nvironment

> überprüft die **Integrität** von Dateien und Verzeichnissen

> für **Rootkit**-Analyse geeignet

AIDE

- **A**dvanced **I**ntrusion **D**etection **E**nvironment
- überprüft die **Integrität** von Dateien und Verzeichnissen
- für **Rootkit**-Analyse geeignet
- erstellt Datenbanken mit kryptografischen **Prüfsummen**
- Konfigurationsdatei definiert, welche Dateien/Verzeichnisse überprüft werden
  - **Flags** definieren Berechtigungen, Datei-Änderungen, Benutzer/Gruppe,...
  - **Templates** erleichtern die Definition neuer Regeln

AIDE

- **A**dvanced **I**ntrusion **D**etection **E**nvironment
- überprüft die **Integrität** von Dateien und Verzeichnissen
- für **Rootkit**-Analyse geeignet
- erstellt Datenbanken mit kryptografischen **Prüfsummen**
- Konfigurationsdatei definiert, welche Dateien/Verzeichnisse überprüft werden
  - **Flags** definieren Berechtigungen, Datei-Änderungen, Benutzer/Gruppe,...
  - **Templates** erleichtern die Definition neuer Regeln
- regelmäßige Ausführung stellt Integrität sicher
- **Abweichungen** werden aufgelistet, Mail-Report möglich

FLAGS UND TEMPLATES (AUSSCHNITT)

Flag	p	i	u	g	s	m / a
Bedeutung	Berechtigungen	inodes	Benutzer	Gruppe	Größe	Zeitstempel Anpassung/Zugriff

Flag	acl	selinux	xattrs	md5 / sha1 / sha256 / …
Bedeutung	ACLs	SELinux-Kontext	Erweiterte Datei-Attribute	Prüfsumme

```
NORMAL = p+i+n+u+g+s+m+c+acl+selinux+xattrs+sha512
# For directories, don't bother doing hashes
DIR = p+i+n+u+g+acl+selinux+xattrs
# Access control only
PERMS = p+u+g+acl+selinux+xattrs
# Logfile are special, in that they often change
LOG = p+u+g+n+S+acl+selinux+xattrs
# Content + file type.
CONTENT = sha512+ftype
# Extended content + file type + access.
CONTENT_EX = sha512+ftype+p+u+g+n+acl+selinux+xattrs
```

BEISPIELE

```
/boot        CONTENT_EX
/opt         CONTENT

# Pkg manager
/etc/dnf CONTENT_EX
/etc/yum.conf$ CONTENT_EX
/etc/yum CONTENT_EX
/etc/yum.repos.d CONTENT_EX

# web server
/etc/httpd CONTENT_EX
```

FAIL2BAN

> Kombiniertes **IDS** / **IPS*** zum Schutz vor Bruteforcing

> in Python geschriebenes Framework, auf **POSIX**-Systemen lauffähig

> Integriert sich in:

- > Firewalls
- > Mail-Server
- > Webserver, Datenbanken

* **I**ntrustion **D**etection/**P**revention **S**ystem

FAIL2BAN

- **Filter** überwachen **Log-Dateien** aktivierter Dienste anhand
  - **Sucheinträgen** und Schlagwörtern
  - regulärer Ausdrücke

FAIL2BAN

- **Filter** überwachen **Log-Dateien** aktivierter Dienste anhand
  - **Sucheinträgen** und Schlagwörtern
  - regulärer Ausdrücke
- **Schwellwerte** steuern, ab wann IP-Adressen gesperrt werden
  - z.B. nach 3 Versuchen
- **Actions** (*hinterlegte Skripte*) können IP-Adressen auf mehrere Arten sperren
  - z.B. via `nftables`

FAIL2BAN

- **Filter** überwachen **Log-Dateien** aktivierter Dienste anhand
  - **Sucheinträgen** und Schlagwörtern
  - regulärer Ausdrücke
- **Schwellwerte** steuern, ab wann IP-Adressen gesperrt werden
  - z.B. nach 3 Versuchen
- **Actions** (*hinterlegte Skripte*) können IP-Adressen auf mehrere Arten sperren
  - z.B. via `nftables`
- Die Kombination eines *Filters* und einer *Action* wird **Jail** genannt
  - z.B. nach 3 fehlerhaften SSH-Loginversuchen über `nftables` sperren

JAILS

- Es gibt zahlreiche Konfigurationsbeispiele - u.a.:
  - `sshd`, dropbear, Guacamole, Webmin, phpMyAdmin
  - Apache, NGINX, `lighttpd`, Roundcube, Horde, Drupal
  - `proftpd`, `vsftpd`
  - Courier, Dovecot, Postfix, Sendmail, Exim, Cyrus
  - MySQL, MongoDB
  - GitLab

JAILS

- Es gibt zahlreiche Konfigurationsbeispiele - u.a.:
  - `sshd`, dropbear, Guacamole, Webmin, phpMyAdmin
  - Apache, NGINX, `lighttpd`, Roundcube, Horde, Drupal
  - `proftpd`, `vsftpd`
  - Courier, Dovecot, Postfix, Sendmail, Exim, Cyrus
  - MySQL, MongoDB
  - GitLab
- Konfiguration unterhalb `/etc/fail2ban/jail.d`
- Beispiele unter `/etc/fail2ban/jail.conf`

NBDE

- **N**etwork-**b**ound **D**isk **E**ncryption
- Erweiterung für **LUKS**-Geräte, um das Eingeben der **Passphrase** zu vermeiden
  - z.B. für verschlüsselte Fileserver oder **Cloud-VMs** relevant
- Passphrase als **Fallback**
  - Keys sollten regelmäßig **rotiert** werden

NBDE

> **N**etwork-**b**ound **D**isk **E**ncryption

> Erweiterung für **LUKS**-Geräte, um das Eingeben der **Passphrase** zu vermeiden
  > z.B. für verschlüsselte Fileserver oder **Cloud-VMs** relevant

> Passphrase als **Fallback**
  > Keys sollten regelmäßig **rotiert** werden

> Komponenten
  > **Clevis** - entschlüsselt LUKS-Geräte sofern Bedingungen erfüllt sind

  > **Tang** - Webserver-Komponenten auf weiteren Systemen; stellt Schlüssel zur Verfügung

  > **JOSE**-Framework für Ver-/Entschlüsselung von Token

EXKURS: LUKS

> **L**inux **U**nified **K**ey **S**etup

> Spezifikation zur Verschlüsselung von **block devices** als Container

> benutzt das `dm-crypt` Kernel-Modul
  > kann AES-Beschleunigung der CPU nutzen

> unabhängig vom darauf verwendeten **Dateisystem**

EXKURS: LUKS

> **L**inux **U**nified **K**ey **S**etup

> Spezifikation zur Verschlüsselung von **block devices** als Container

> benutzt das `dm-crypt` Kernel-Modul
>> kann AES-Beschleunigung der CPU nutzen

> unabhängig vom darauf verwendeten **Dateisystem**

```
# cryptsetup luksFormat /dev/vdb1
# cryptsetup luksOpen /dev/vdb1 mydisk
# cryptsetup luksClose mydisk
```

Formatieren, Öffnen und Schließen eines Containers via `cryptsetup`

NBDE

Auf **weiteren** Systemen werden **Tang**-Server installiert:

```
# dnf install tang
# systemctl enable --now tangd.socket
# firewall-cmd --add-service http --permanent
# firewall-cmd --full-reload
```

Installation von **Clevis** auf dem Server mit dem zu entschlüsselnden LUKS-Gerät:

```
# dnf install clevis{,-luks,-dracut}
```

NBDE

Es werden verschiedene **Policies** unterstützt, die gebräuchlichste ist jedoch **SSS***, bei der einige oder alle Tang-Server **verfügbar** und **vertrauenswürdig** sein müssen:

```
# cfg='{"t":2,"pins":{"tang":[{"url":"http://srv01"}, {"url":"http://srv02"}]}}'

# clevis luks bind -d /dev/vdb1 sss "$cfg"
# systemctl enable --now clevis-luks-askpass.path
```

`t` = **threshold**, Minimum verfügbarer Server

`/etc/crypttab` und `/etc/fstab` müssen noch angepasst werden, damit ein Einhängen beim Boot funktioniert (siehe auch RHEL-Dokumentation)

* **S**hamir's **S**ecret **S**haring

// DEV-SEC

MOTIVATION

> Eine regelmäßige Auditierung der Systeme ist **unabdingbar**
  > Ständige Updates bringen **neue** Komponenten und **Fehler**

  > Kritische Lücken erfordern **außerplanmäßiges** Handeln (*Heartbleed, Spectre, Meltdown*)

MOTIVATION

> Eine regelmäßige Auditierung der Systeme ist **unabdingbar**
  > Ständige Updates bringen **neue** Komponenten und **Fehler**

  > Kritische Lücken erfordern **außerplanmäßiges** Handeln (*Heartbleed, Spectre, Meltdown*)

> System muss schon bei der Bereitstellung **möglichst sicher** sein
  > **Nachträgliches** Absichern erfolgt erfahrungsgemäß nicht

> Hardening muss weitestgehend automatisiert werden > **praktikabel** und effizient

OPENSCAP

> Freie **SCAP***-Implementation

> Protokoll zur automatisierten Vulnerability- und Compliance-Messung und Auswertung

> **Ziel**: System-Verwundbarkeit darstellen

* **S**ecurity **C**ontent **A**utomation **P**rotocol

OPENSCAP

> Freie **SCAP***-Implementation

> Protokoll zur automatisierten Vulnerability- und Compliance-Messung und Auswertung

> **Ziel**: System-Verwundbarkeit darstellen

> Katalog-Quellen:
  > OpenSCAP-Projekt
  > GitHub-Projekt "ComplianceAsCode"

> Vor allem für internationale Standards interessant (*z.B.* ***PCI-DSS***)

* **S**ecurity **C**ontent **A**utomation **P**rotocol

OPENSCAP

> Anpassung **nur bedingt** möglich
  > Deaktivieren einzelner Checks über **Tailoring-File**

  > SCAP Workbench benötigt

> Erstellen eigener Inhalte **extrem aufwändig**
  > Pflege mehrerer XML-Dokumente

  > Alternativen, wie z.B. InSpec oder cnspec sinnvoller

DEV-SEC

> Nicht nur die sichere Bereitstellung neuer Systeme ist wichtig

> Auch **bestehende** Systeme müssen **regelmäßig** überprüft und abgesichert werden

> Je nach Systemlandschaft hoher Aufwand

DEV-SEC

> Nicht nur die sichere Bereitstellung neuer Systeme ist wichtig

> Auch **bestehende** Systeme müssen **regelmäßig** überprüft und abgesichert werden

  > Je nach Systemlandschaft hoher Aufwand

> Dev-Sec auditiert und sichert Server und Applikationen ab

> **Baselines** u.a. für

  > Microsoft Windows / Linux

  > SSH / SSL

  > Apache / NGINX

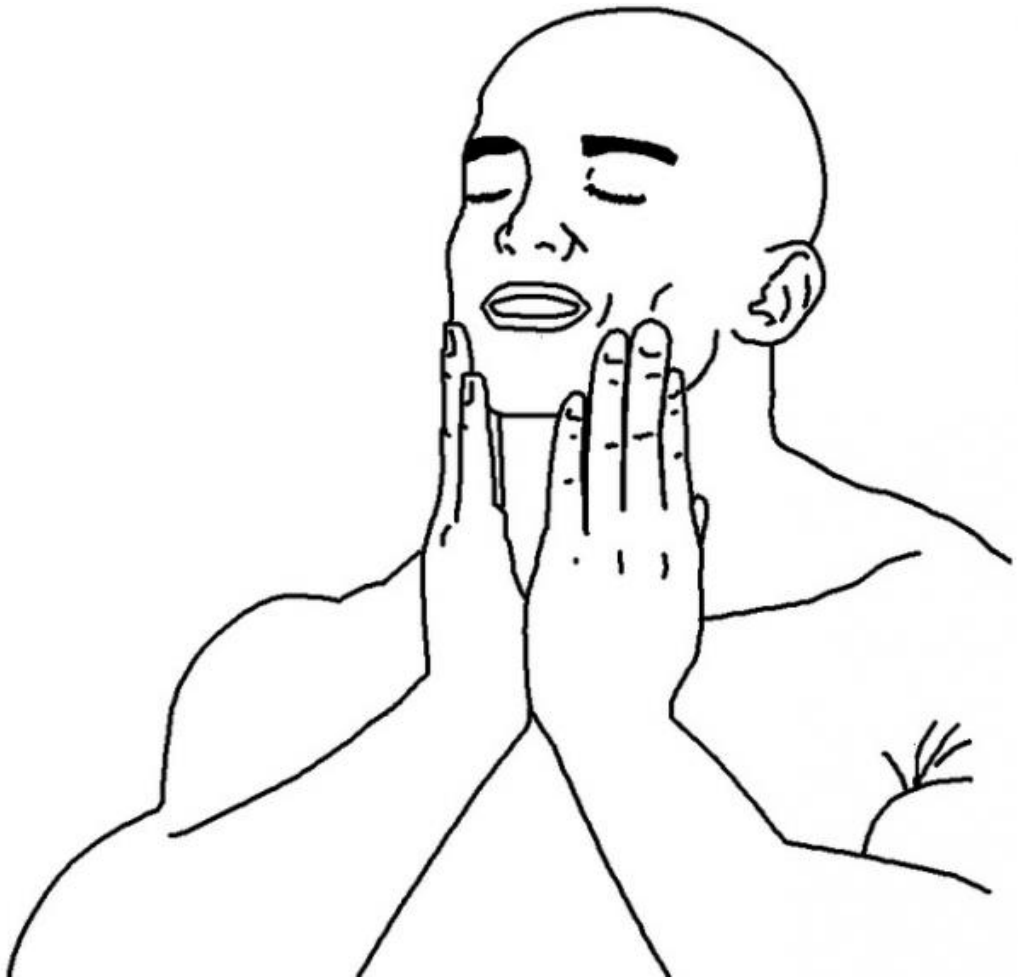
DEV-SEC

- Grundlagen für Dev-Sec-Kataloge:
  - **CIS**-Benchmarks (*Center for Internet Security*)
  - NSA Hardening Guides
  - Hersteller Best-Practices und Security Hardening Guides
- Überprüfung der Systeme via **InSpec** (*bald* `cnspec`)
- Open-Source: [klick!]

AUTOMATIC REMEDIATION

> Für die meisten* Baselines gibt es vordefinierte Automatismen für:
  > Ansible (Rolle)
  > Chef (Cookbook)
  > Puppet (Modul)
> Download über:
  > Dev-Sec-Webseite / GitHub
  > jeweilige Tool-Community

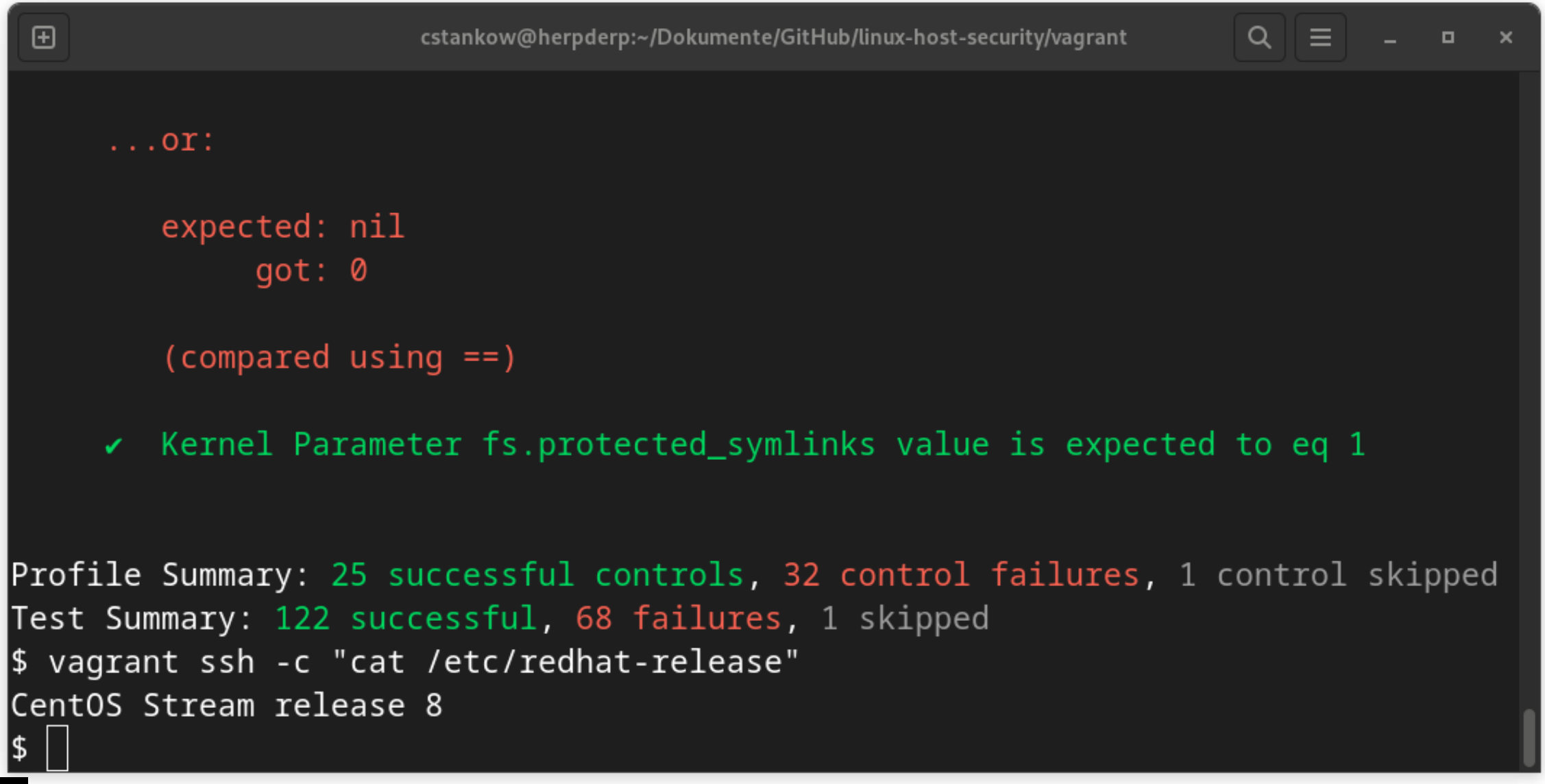
* ausgenommen Docker und Kubernetes



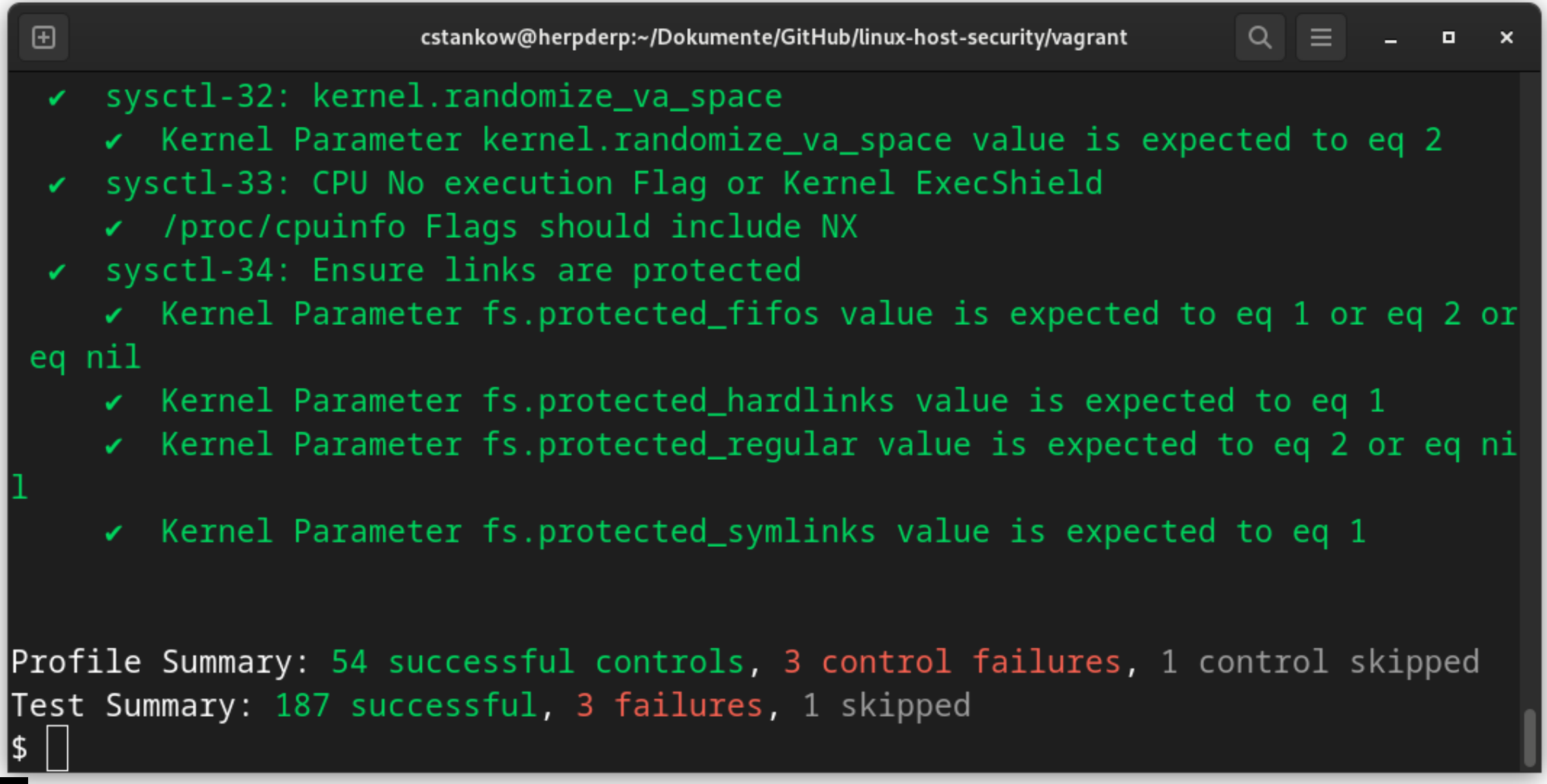
REMEDIATION

Beispielhaftes Playbook:

```
- name: Harden servers
  hosts: weakservers
  become: true
  roles:
    - name: devsec.hardening.os_hardening
      sysctl_overwrite:
        # Enable IPv4 forwarding (needed for containers)
        net.ipv4.ip_forward: 1
```


cstankow@herpderp:~/Dokumente/GitHub/linux-host-security/vagrant
...or:
expected: nil
got: 0
(compared using ==)
✔ Kernel Parameter fs.protected_symlinks value is expected to eq 1
Profile Summary: 25 successful controls, 32 control failures, 1 control skipped
Test Summary: 122 successful, 68 failures, 1 skipped
$ vagrant ssh -c "cat /etc/redhat-release"
CentOS Stream release 8
$


cstankow@herpderp:~/Dokumente/GitHub/linux-host-security/vagrant
TASK [Initialize database] **
changed: [default]
TASK [Overwrite database] ***
changed: [default]
TASK [Create cronjob] ***
changed: [default]
PLAY RECAP **
default : ok=116 changed=38 unreachable=0 failed=0 s
kipped=49 rescued=0 ignored=1
$ inspec exec linux-baseline -t ssh://vagrant@192.168.121.148 -i .vagrant/machin
es/default/libvirt/private_key --sudo


cstankow@herpderp:~/Dokumente/GitHub/linux-host-security/vagrant
✔ sysctl-32: kernel.randomize_va_space
✔ Kernel Parameter kernel.randomize_va_space value is expected to eq 2
✔ sysctl-33: CPU No execution Flag or Kernel ExecShield
✔ /proc/cpuinfo Flags should include NX
✔ sysctl-34: Ensure links are protected
✔ Kernel Parameter fs.protected_fifos value is expected to eq 1 or eq 2 or eq nil
✔ Kernel Parameter fs.protected_hardlinks value is expected to eq 1
✔ Kernel Parameter fs.protected_regular value is expected to eq 2 or eq nil
✔ Kernel Parameter fs.protected_symlinks value is expected to eq 1
Profile Summary: 54 successful controls, 3 control failures, 1 control skipped
Test Summary: 187 successful, 3 failures, 1 skipped
$

BEST PRACTICES

> Rollen zuerst auf Test-Systemen ausführen und **ausgiebig testen**
  > Anwendungen könnten nach Härtung nicht mehr funktionieren

> Dokumentation und insbesondere Parameter/Schalter konsultieren
  > diese stellen i.d.R. einzelne Härtungen ab

> MITRE heimdall2 zur **Visualisierung** benutzen

BEST PRACTICES

> Rollen zuerst auf Test-Systemen ausführen und **ausgiebig testen**
  > Anwendungen könnten nach Härtung nicht mehr funktionieren

> Dokumentation und insbesondere Parameter/Schalter konsultieren
  > diese stellen i.d.R. einzelne Härtungen ab

> MITRE heimdall2 zur **Visualisierung** benutzen

> **Version-Pinning** - nie ungetestet neue Rollen verwenden
  > `requirements.yml` nutzen!

  > neue Härtungen könnten wieder Fehler nach sich ziehen

  > bei neueren Versionen wieder zuerst auf Test-Systemen testen

// LINKLISTE

Beispiel-Projekt mit Ansible-Konfiguration und DevSec-Härtung:

https://github.com/stdevel/linux-host-security

Feedback und PRs sind willkommen!

FOCUS ON: LINUX

Themen wie diese könnt ihr alle 2 Wochen hören:

- News des Monats
- Tooltipps
- Thematische Sonderfolgen
  - Konferenzen, NixOS,...

Verfügbar via:

- RSS / fyyd
- Apple Podcasts
- Spotify

DANKE FÜR DIE AUFMERKSAMKEIT!