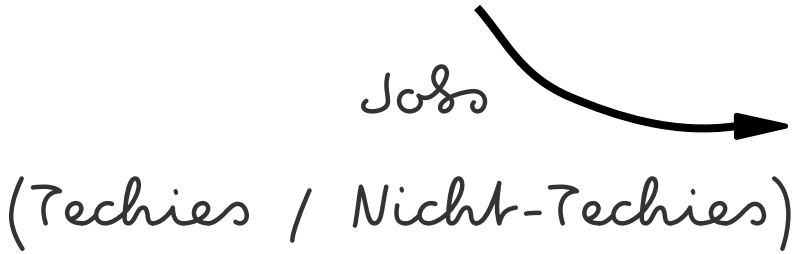
Troubleshooting für Systemadministrator*innen

Silke Meyer

Chemnitzer Linuxtage 2023

Herzlich willkommen

- Politikwissenschaftlerin und seit 15 Jahren Systemadministration für Linuxsysteme
- DMX Systems (CLT 2021) → Projekt linqa.eu
- DFN e.V.: Single Sign-on im Hochschulumfeld



Agenda

- Organisatorische Vorbereitung
- Tipps für eine systematische Vorgehensweise
- Nachbereitung
- Umgang mit Fehlermeldungen
- kurzes Beispiel

Organisatorische Vorbereitung

Arbeitsteilung im Team

- Rollenverteilung:
  - Wer löst das technische Problem?
  - Wer informiert Dritte?

Admin*s	Kommunikator*innen
Konzentration	Information der Betroffenen
Ständige Nachfragen stören	Proxy für Statusabfragen
Ehrliche Statusupdates geben	Vermittlung / Wording

Einzelkämpfer*innen

- Vorlage mit „Blanko-Vorfall“ vorbereiten
- nur noch Details zur Störung ergänzen
- Timeboxing grob planen: In welchen Intervallen unterbricht man die Arbeit, um Statusupdates zu verschicken?

Kommunikation

- Kommunikationskanäle geklärt haben
  - Mailing? Website? Statusseite?
  - Kund*innen-Interaktion über Social Media?
  - Wer bespielt die Kanäle?
- extern gehostete Kanäle vorhanden?
- gut formulierte Templates vorhanden?

Tipps für eine systematische Vorgehensweise

Überblick gewinnen

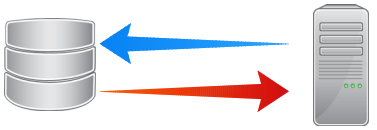
Don't panic! \o/

- Fehlermeldungen lesen, Fehler reproduzieren
- Blick ins Monitoring
- Ist klar, was das Problem ist? Auslöser klar?
- Klärung: Wer ist betroffen?
- Rollenklärung
- Störungsmeldung

Einstieg

- kaputten Zustand sichern für Analyse
- Doku von früherem Incident vorhanden?
- klären, ob / wann Backup eingespielt wird
- Notizen: unternommene Schritte, offene Fragen, ausstehende Aufgaben
- übliche Verdächtige durchgehen
- eventuell kleiner Testaufbau?

Tieferer Einstieg

- Loganalyse: Schritte nachvollziehen
- Zeichnung anfertigen 
- Vergleich mit funktionierendem Zustand
  - letzte Änderungen prüfen
  - Änderungen in der Infrastruktur außerhalb des Teams?
- Problem vereinfachen / Komplexität reduzieren

Recherche

- das Internet befragen
  - mitdenken und Tipps verifizieren!
  - `chmod 777` ist nie eine gute Idee
- Konfigurationsdateien lesen (inkl. conf.d)
- Dokumentation lesen

Andere hinzuziehen

- 4-Augen-Prinzip im Team
- Problem in Worte fassen → 
  - „Rubberduck“-Effekt
  - Hilfegesuch ausformulieren (inkl. Versionen, Logmeldungen, ausgeschlossene Ursachen)
  - „Was würde ich jetzt fragen?“
- externe Hilfe suchen (z.B. Foren, Mailinglisten)

Arbeitsweise

- Notizen abarbeiten √ uns ggf. ergänzen
- immer nur eine Sache auf einmal ändern
- fokussiert bleiben
- zum Debugging geeignete Tools identifizieren

→ Ach, uns Pause machen nicht vergessen!

Nachbereitung

Nachhaltigkeit

- zeitnahe Dokumentation
  - Man vergisst die Details sehr schnell!
- Wissen teilen, Kolleg*innen die Arbeit ersparen
- Ja, mehr / ausführlichere Kommentare helfen.
- Aber v.a. die relevanten Stufen des Prozesses aufschreiben…

Dokumentation

- … von Problemanalyse bis -behebung
- einheitliches Tool mit Suchfunktion
- Übersichtlichkeit, Vollständigkeit → ggf. Vorlage bereitstellen
- Schlagworte, Titel, Tags, komplette Fehlermeldungen → Durchsuchbarkeit

Umgang mit Fehlermeldungen

Klassische Logfiles

- Daemon syslog bzw. syslog-ng bzw. rsyslog schreibt nach `/var/log`
- Facilities und Priorities (Thema und Gewicht)
- alles konfigurierbar → ggf. Loglevel erhöhen
- `bei Anwendungen, die nicht aus Paketmanagement kommen, ggf. abweichend`

Logfiles durchsuchen

- zur Suche mit `grep` siehe CLT2022
- `rgrep -i -e error -e warn /var/log/* | less`
- Aktuelle vs. rotierte/komprimierte Logs:

`less datei.log`	`zless datei.3.gz`
`cat datei.log`	`zcat datei.3.gz`
`grep string datei.log`	`zgrep string datei.3.gz`

- `tail -f /var/log/apache2/*.log` → Logs mitverfolgen

Journald

- kein Logfile da? Systemd → zentrales Journal
- binäre Dateien, durchsuchbar mit `journalctl`
- standardmäßig nicht persistent
- `/etc/systemd/journald.conf`
- häufig standardmäßige Weiterleitung des Journals an den Syslog-Daemon
- `systemctl status UNIT` zeigt auch Fehler

Journal durchsuchen

- hilfreiche Schalter für journalctl:
  - `-r` neueste Meldungen oben
  - `-f` mitverfolgen
  - `-u` Filter auf Systemd Unit
  - `--since` bzw. `--until`
- `journalctl -u slapd --since 2023-03-08 08:00:00 --until yesterday`
- `strings /var/log/journal/UUID/system.journal | grep -i error`

Weitere Möglichkeiten

- Bei Webanwendungen: Browser-Konsole auf Fehlermeldungen prüfen
- Bei php-Anwendungen: Fehler in Seite zeigen lassen (`php.ini`)

```
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
display_errors = On
```

- Syntaxchecker nutzen: Bsp. `apache2ctl -t`
- `...`

Fehlermeldungen lesen

- verständliche Teile heraussuchen und zu verstehen versuchen, z.B.:
  - Connection refused → Verbindung
  - Invalid Credentials → Zugangsdaten
- Verweise auf Dateien / Zeilen ernst nehmen
- Abgleich mit letzten Änderungen

Fehlermeldungen lesen

```
2023-02-28 23:19:33 ERROR OpenSSL : error
code: 33558541 in bss_file.c, line 413
2023-02-28 23:19:33 ERROR OpenSSL : error
data:
fopen('/etc/ssl/private/privkey.pem','r')
2023-02-28 23:19:33 ERROR
XMLTooling.CredentialResolver.Chaining :
caught exception
processing embedded CredentialResolver
element: Unable to load private key from
file (/etc/ssl/private/privkey.pem).
```

Usual Suspect: Permissions

- hartnäckig wiederkehrende Fehlerquelle
- Prüfung der Dateirechte

```
# ls -la /etc/ssl/private/privkey.pem
-rw-r----- 1 root ssl-cert .. .. .. .. privkey.pem
```

- Vergleich mit Nutzerkennung / Gruppenzugeh.

```
# ps aux | grep prosody
prosody .. .. .. .. .. .. .. lua5.2 /usr/bin/prosody

# id prosody
uid=113(prosody) gid=121(prosody) .. .. ,120(ssl-cert)
```

ja, prosody darf privkey lesen

Fehlermeldungen lesen

- ERROR [net.shibboleth.idp.profile.impl.ResolveAttributes:317] - Profile Action ResolveAttributes: **Error resolving attributes** net.shibboleth. idp.attribute.resolver.ResolutionException: **Attribute Definition 'subjectHash':Script did not run successfully** at net.shibboleth .idp.attribute.resolver.ad.impl.ScriptedAttr ibuteDefinition$AttributeDefinitionScriptEvaluator.execute(ScriptedAttributeDefinition.java:228)Caused by: java.lang.RuntimeException: javax.script.ScriptException: **ReferenceError: "uid" is not defined** in <eval> at line number 2
- Wo werden in der Anwendung Attribute definiert?
- Was wurde dort geändert? Wo wird in der Änderung die uid referenziert?

Letzte Änderungen finden

- Konfigurationsmanagement einsetzen
- Versionierung relevanter Verzeichnisse (etckeeper, git)
- Dateivergleich:
- ~~(diff)~~ schwer zu lesen, besser: sdiff
- Ordnerinhalte rekursiv vergleichen: diff -r
- zum Editieren: vimdiff, meld (GUI)

Troubleshooting-Beispiel

Fehler bei Website-Aufruf

- Namensauflösung prüfen "It's always DNS!"
- ipv4 und ipv6 auf dem Schirm haben
- Routing nachvollziehen
- gezielte TCP-Verbindung zu Port aufbauen
- HTTP-Header abfragen, Statuscode prüfen
- TLS-Verbindung aufbauen, Zertifikat prüfen

Links und Quellen

… um nur einige zu nennen…

- Atlassian über Postmortems
- sadservers: Troubleshooting Übungssessions
- Julia Evans‘ Wizard Zines: Linux Debugging Tools ← süß!
- Übersicht über Configtest-Befehle
- Übersicht über Systemmonitoring-Tools
- Ubuntu-Wiki über journalctl

Danke für Ihr/Euer Interesse!

- Fragen? Anregungen?
- Kontakt: Silke Meyer
  silke@silkemeyer.net
  @freiefunken@mastodon.social
  LinkedIn: linkedin.com/in/freiefunken/