Dateisysteme sicher freigeben mit gssproxy

Daniel Kobras (PITC)
Michael Weiser (s+c)

Chemnitzer Linux-Tage
2024-03-17

science + computing

Über uns

Daniel Kobras

Principal Architect (Puzzle ITC)

Kubernetes/Container, Digital Identity, Kerberos-Buch

kobras@puzzle-itc.de

Michael Weiser

Senior Solution Architect (science+computing ag)

Kubernetes/Container, Cloud, Security, HPC

michael.weiser@eviden.com

PUZZLE ITC

changing IT for the better

- IT-Dienstleister in CH (Bern, Zürich, Basel) und DE (Tübingen)
- 25 Jahre, 150 Members
- Schwerpunkte:
  - Open-Source-Technologien
  - Applikationsentwicklung
  - Container-Plattformen, CI/CD
  - Linux System Engineering
  - Mobility
- https://www.puzzle.ch/

science + computing

- IT-Dienstleister mit Standorten in Tübingen, München, Berlin und Düsseldorf
- Schwerpunkte:
  - HPC
  - Big Data
  - Security
- https://www.science-computing.de/

Agenda

NFS und SMB mit gssproxy

- Ausgangslage
- Überblick
- Umsetzung
- Demo

Worum geht es?

Ausgangslage

- Wir schreiben das Jahr 2024
- Wir verwenden HTTPS, SSH und vielleicht sogar SMTPS

Alle Daten im Netz sind verschlüsselt!

Alle Daten?

NFS

NFS in der Praxis

- Export-Protokoll der Unix-Welt für lokale Daten
- Multi-userfähige Integration in Linux-Dateisystem
- herkömmliche Alternative: NFS mit **AUTH_SYS**
  - keine User-Authentisierung
  - schwache Client-Authentisierung (IP-Adresse)
  - keine Integritätsprüfung
  - überträgt Daten unverschlüsselt im Netz

Sicherheitsniveau ähnlich `rlogin` *oder* `telnet`

NFS in sicher

- sichere Alternative: NFS mit **AUTH_GSS** (GSSAPI/Kerberos)
  - starke User-Authentisierung pro Zugriff
  - starke Client-Authentisierung
  - Integritätsprüfung und Transportverschlüsselung optional zuschaltbar
  - seit Jahrzehnten verfügbar, Pflicht mit NFSv4

In freier Wildbahn kaum anzutreffen

Warum eigentlich NFS?

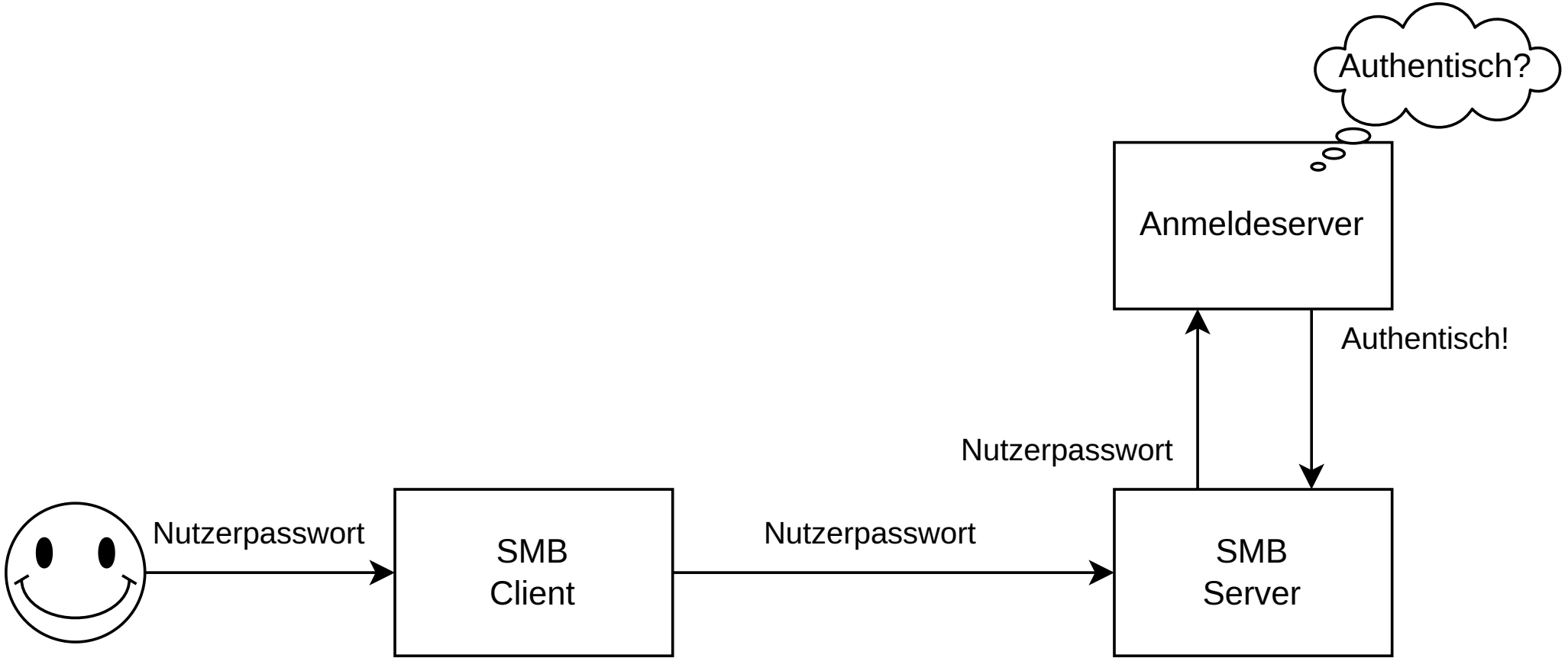
- NFS:
  - Server ist Linux/UNIX
  - Appliance mit interner POSIX-Semantik (z.B. NetApp UNIX security mode)
- SMB:
  - Server ist Windows oder Linux-Clients eine Nische
  - Appliance mit interner NTFS-Semantik (z.B. NetApp NTFS security mode)

SMB als Alternative?

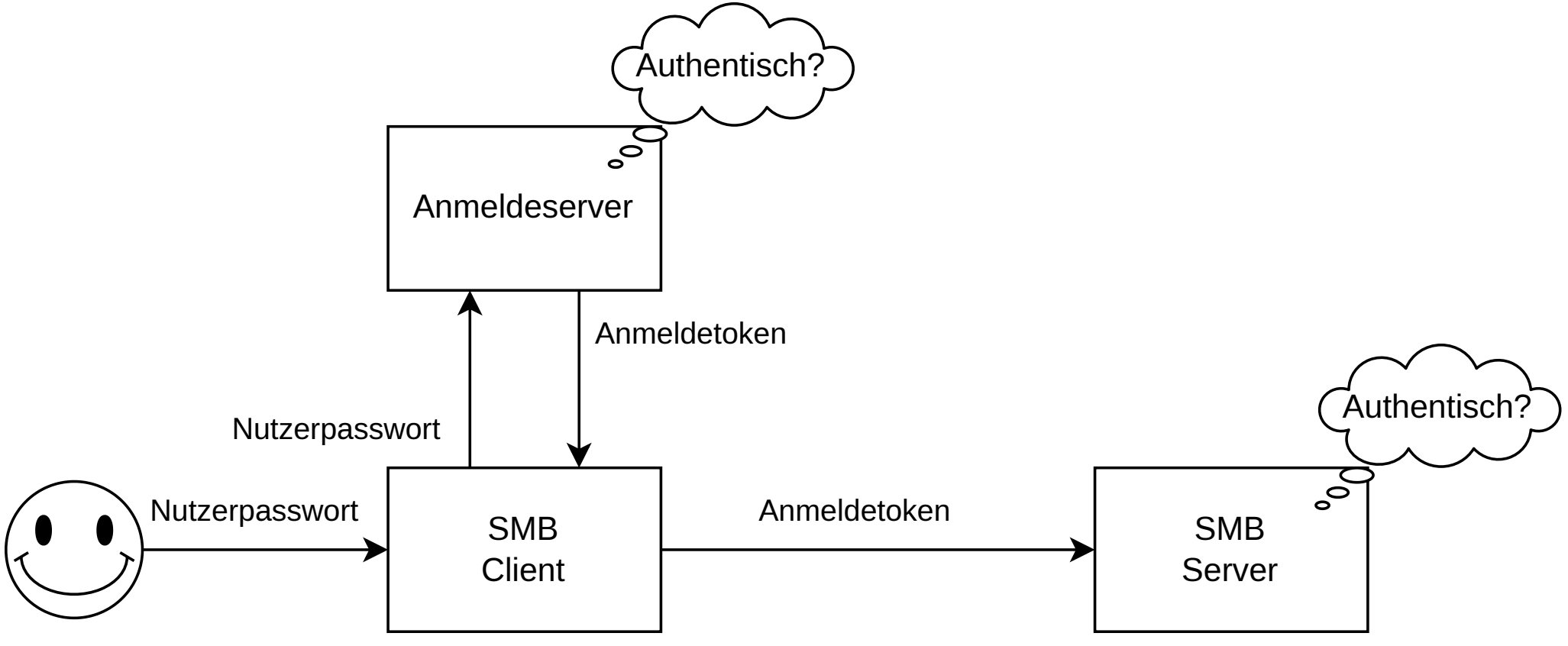
- Export-Protokoll der Windows-Welt für lokale Daten
- Integration in Linux-Dateisystem möglich (`mount.cifs`)
- klassisch User-Authentisierung pro Verbindung (Mount), damit nur bedingt multi-userfähig
- Mount-Option `multiuser` (aka multisession) authentisiert einzelne Zugriffe
  - für Multi-User-Einsatz unter Linux besser geeignet
  - benötigt entweder Passwörter (`cifscreds`/`pam_cifscreds`) oder Kerberos
  - Passwort-Varianten in größeren Umgebungen unpraktikabel
- Integritätsprüfung und Transportverschlüsselung optional zuschaltbar
- Abstriche bei POSIX-Semantik (Permission Bits, ACLs, Eigentümer)

Andere Ursache (Multiuser-Zugriff), gleiche Konsequenz (Kerberos)

Anmeldung in der guten alten Zeit



Anmeldung mit Kerberos



Zwischenfazit

- unverschlüsselter Datenverkehr (längst) nicht mehr zeitgemäß
- Verschlüsselung samt Multiuser-Fähigkeit in Standardprotokollen (NFS, SMB) gegenwärtig gekoppelt an starke User-Authentisierung mit Kerberos
- starke User-Authentisierung schafft zusätzliche Schwierigkeiten, vor allem bei nicht-interaktiven Zugriffen
- starke User-Authentisierung häufig gar keine unmittelbare Anforderung, sondern nur technische Folge von Multi-User-Fähigkeit und Verschlüsselung
- Alternativen zu NFS und SMB zwar möglich, aber weniger universell und interoperabel

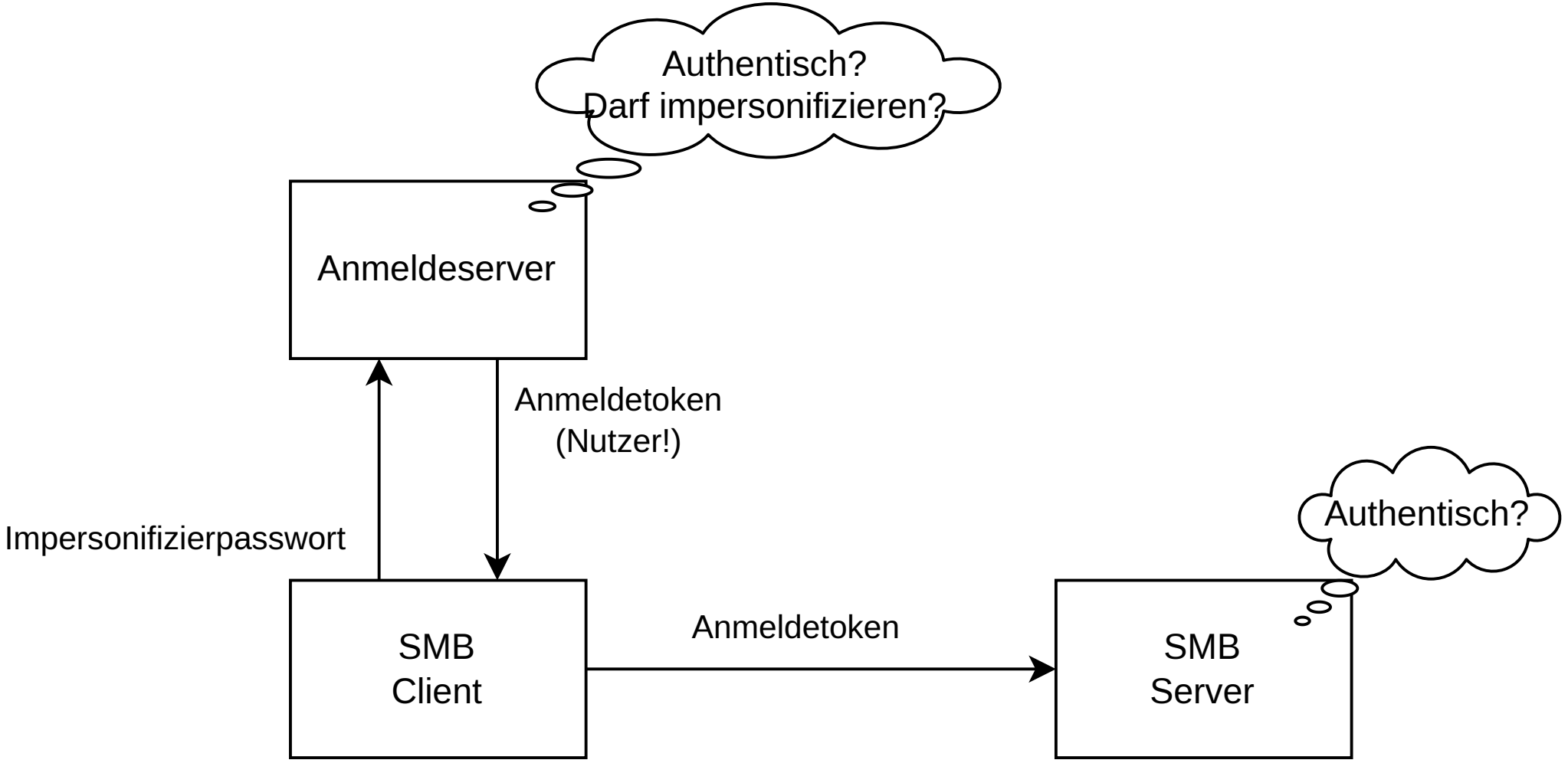
*Hinweis: NFS mit **AUTH_TLS** im Linux-Kernel v6.4 (Server) bzw. v6.5 (Client), aber noch nicht breit verfügbar (Distributionen, Appliances)*

Lösungswege

1. Industriestandard: klein beigeben und auch in 2024 noch unverschlüsselte Daten per NFS übers Netz schicken
2. Modernisieren: Cloud, Container, alles gut (Spoiler: Wer RWX-PVs in k8s benötigt, landet schnell wieder beim guten, alten NFS - und den alten Problemen)
3. Im Folgenden: Griff in die Kerberos-Trickkiste, um die starke User-Authentisierung gezielt aufzuweichen und dadurch praxistauglicher zu machen

Überblick

Suche den User



Eigenschaften

- Client identifiziert User, greift in dessen Namen auf Freigaben zu
- Client muss sich dazu selbst stark authentisieren
- Client benötigt Mitwirkung (Zustimmung) des Kerberos-Servers (KDC)
    - für jeden User
    - für jeden Fileserver
- Client kann zusätzliche Optionen nutzen (Integritätsprüfung, Verschlüsselung)

User-Authentisierung durch Client konzeptionell ähnlich zu NFS mit ***AUTH_SYS****, aber zusätzlich auditierbar und konfigurierbar (einschränkbar)*

Bestandteile

1. Infrastruktur: Kerberos-Erweiterungen für Delegation (bzw. Impersonifizierung)
    - breit unterstützt: Active Directory (Microsoft+Samba), FreeIPA
    - weniger breit bekannt (außer bei Pentestern)
    - nicht nur gut für Angriffe, sondern auch für sinnvolle Anwendungen :-)
2. Client: `gssproxy`
    - kann Kerberos-Erweiterungen (`S4U2Self`, `S4U2Proxy`) nutzen
    - arbeitet mit NFS- und SMB-Clients zusammen
    - für Linux verfügbar

Umsetzung

Impersonifizierung/Delegation mit Kerberos

- auf Protokoll-Ebene recht komplex
- Grundkonzept deutlich simpler und hier ausreichend
- NFS- bzw. SMB-Client soll selbsttätig Identität beliebiger Nutzer annehmen können
- Berechtigung idealerweise beschränkt auf bestimmte NFS- bzw. SMB-Server (Constrained Delegation)

Nachfolgend mit Active Directory beschrieben, aber nicht darauf beschränkt

Impersonifizierung Variante 1: Protocol Transition

- Konfiguration am Beispiel Active Directory:
  - Flag `TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` (`0x01000000`) in Attribut `userAccountControl` des Computerobjekts für den NFS/SMB-Client setzen
  - `nfs`- bzw. `cifs`-Serviceprincipalnames aller NFS/SMB-Server in Attribut `msDS-AllowedToDelegateTo` des NFS-Clients eintragen
- Eigenschaften:
  - Auch mit älteren Kerberos-Bibliotheken umsetzbar
  - Auf eine Domäne beschränkt
  - Änderungen am Computerobjekt für jeden NFS-Client nötig

Impersonifizierung Variante 2: Resource-based Constrained Delegation

- Konfiguration am Beispiel Active Directory:
  - NT Security Descriptor des NFS-Clients in Attribut `msDS-AllowedToActOnBehalfOfOtherIdentity` am AD-Objekt des NFS/SMB-Servers hinzufügen
- Eigenschaften:
  - Benötigt auf NFS/SMB-Client mindestens MIT Kerberos 1.19 (EL9, Ubuntu 22.04)
  - Über Domänengrenzen hinweg funktionsfähig (Vertrauensstellung nötig)
  - Nur Änderungen am AD-Objekt des NFS/SMB-Servers nötig
  - Tooling für binäre Datenstruktur des Security Descriptors nötig

Grundkonfiguration gssproxy für NFS

- Paket `gssproxy` auf NFS-Clients installieren
- `gssproxy` in NFS-Konfiguration (`/etc/nfs.conf`) aktivieren:

```
(...)
[gssd]
use-gss-proxy=1
```

- Alternative für Distributionen mit älteren `nfs-utils` ohne `nfs.conf`: Umgebungsvariable `GSS_USE_PROXY=yes` für Prozess `rpc.gssd` setzen.
- NFS-Client verwendet nun primär `gssproxy`, um User per Kerberos zu authentisieren
- bereits nutzbar z.B. mit Keytab-Authentisierung

Konfiguration gssproxy für SMB

- Pakete `gssproxy` und `cifs-utils` >= 7.0 auf SMB-Clients installieren
- statischer mount mit Maschinenidentität aus Keytab:

```
mount -o vers=3.11,sec=krb5i,seal,multiuser,user='FS1$' //fs1.example.com/share /mnt
```

- oder dynamisch mit dem Automounter und gleich der Nutzeridentität:

```
share -fstype=cifs,vers=3.11,sec=krb5i,seal,multiuser,cruid=${UID} ://fs1.example.com/share
```

- In `/etc/request-key.d/cifs.spnego.conf` Umgebungsvariable `GSS_USE_PROXY` setzen:

```
create  cifs.spnego    * * /bin/env GSS_USE_PROXY=yes /usr/sbin/cifs.upcall %k
```

Impersonifizierung mit gssproxy

- Impersonifizierung in `gssproxy`-Konfiguration für NFS-Clients erlauben:
  (alt: `/etc/gssproxy/99-nfs-client.conf`,
  neu: `99-network-fs-clients.conf`)

```
[service/network-fs-clients]
(...)
allow_any_uid = yes
trusted = yes
impersonate = true  # <-- Zeile hinzufügen
euid = 0
```

- Nur bei Variante 1 (Protocol Transition): In Kerberos-Konfiguration `/etc/krb5.conf` Tickets mit Forwardable-Flag anfordern:

```
[libdefaults]
(...)
forwardable = true
```

Demo

Konfiguration Tipps&Tricks

- Mitprotokollieren, für welche Principals `gssproxy` Tickets anfordert
  - Log-Level in `gssproxy`-Konfiguration (`/etc/gssproxy/gssproxy.conf`) erhöhen

```
[gssproxy]
debug_level = 1
```

- Impersonifikation nur, falls herkömmliche Kerberos-Authentisierung nicht erfolgreich:
  - `gssproxy`-Integration als Fallback konfigurieren
  - Für Prozess `rpc.gssd` bzw. `cifs.spnego`-Upcall die Umgebungsvariable `GSSPROXY_BEHAVIOR=LOCAL_FIRST` setzen

- Separate NFS/SMB-Konfiguration mit gssproxy
  - Abschnitt `service/network-fs-clients` wird in Standardform von NFS- und SMB-Clients angewendet
  - Falls unterschiedliche Konfiguration gewünscht, kann `program`-Direktive zwischen aufrufenden Programmen unterscheiden
  - `/etc/gssproxy/99-nfs-client.conf`:

```
[service/nfs-client]
impersonate = false
program = /usr/sbin/rpc.gssd
```

`/etc/gssproxy/99-cifs-client.conf`:

```
[service/smb-client]
impersonate = true
program = /usr/sbin/cifs.upcall
```

  - Alternativ auch Zugriff über verschiedene UNIX Domain Sockets möglich (Direktive `socket` und Env-Variable `GSSPROXY_SOCKET`)

Konfiguration Tipps&Tricks

- Vor Delegation geschützte Accounts in Active Directory (über Gruppe `Protected Users` oder UAC-Flag `0x100000 NOT_DELEGATED`)
  - nur interaktiv (mit vorhandenen, gültigen Kerberos-Credentials) verwenden
  - über Keytab (`/var/lib/gssproxy/clients/31337.keytab`) authentisieren mit Override-Eintrag `/etc/gssproxy/00-protecteduser.conf`:

```
[service/protecteduser]
mechs = krb5
cred_store = ccache:FILE:/var/lib/gssproxy/clients/krb5cc_%U
cred_store = client_keytab:/var/lib/gssproxy/clients/%U.keytab
cred_usage = initiate
# uid of protecteduser
euid = 31337
```

  - Stellt sicher, dass Impersonifizierung für diesen Account auf ausgewählte Clients mit vorhandener Keytab beschränkt ist

Konfiguration Tipps&Tricks

- Lokale Accounts ohne zugeordneten AD-User:
  - Explizites Mapping auf Principal in `gssproxy` konfigurieren
  - Override-Eintrag `/etc/gssproxy/00-localuser.conf`:

```
[service/localuser]
mechs = krb5
cred_store = keytab:/etc/krb5.keytab
cred_store = ccache:FILE:/var/lib/gssproxy/clients/krb5cc_%U
cred_usage = initiate
krb5_principal = aduser@EXAMPLE.COM
trusted = yes
impersonate = true
# uid of localuser
euid = 101
```

Zusammen-fassung

Fazit

- NFS klassisch mit geringem Sicherheitsniveau
- NFS mit Kerberos mit sehr hohem Sicherheitsniveau, das aber zu praktischen Problemen führt
- SMB als möglicher Ausweg, benötigt bei NFS-ähnlicher Konfiguration aber ebenfalls Kerberos und landet bei denselben Problemen
- `gssproxy` und Kerberos-Delegation/Impersonifizierung reduzieren kontrolliert die Sicherheit, aber machen kerberisiertes NFS und SMB dadurch praxistauglich
- benötigte Komponenten inzwischen breit verfügbar
- Konfigurationsaufwände für Impersonifizierung vergleichsweise gering

Vielen Dank!

Michael Weiser <michael.weiser@eviden.com>

Daniel Kobras <kobras@puzzle-itc.de>

@puzzleitc

@github.com/puzzle