Dateidienste – sicher und alltagstauglich

Tübix 2019

Daniel Kobras
Puzzle ITC Deutschland GmbH

PUZZLE ITC

changing IT for the better

- IT-Dienstleister mit Hauptsitz in Bern, CH
- gegründet 1999
- ca. 125 Mitarbeiter
- https://www.puzzle.ch/
- Schwerpunkte:
  - Applikationsentwicklung, bevorzugt Open Source
  - Continuous Integration, Continuous Delivery
  - Cloud-Infrastruktur (APPUiO)
  - Linux System Engineering
  - Consulting

PUZZLE ITC
Deutschland

- deutsche Tochtergesellschaft von Puzzle ITC mit Sitz in Tübingen
- gegründet 2018
- Schwerpunkte:
  - Single Sign-On und starke Authentisierung
  - Verzeichnisdienste und Benutzerverwaltung
  - Hochverfügbarkeit, Storage und Dateidienste
  - Konfigurations- und Lifecycle-Management

Sichere Dateidienste

Aspekte

- Zugriffe über eventuell unsicheres Netzwerk
- Authentisierung (Nutzer, System)
- Integritätsprüfung
- Verschlüsselung

Schnittstellen

- Posix/VFS, systemweit, globaler Namensraum, konkurrierender Zugriff durch mehrere Nutzer
- applikationsspezifisch (z.B. WebDAV, SMB), ggf. per fuse oder klassisches `mount.cifs` in Namensraum eingebunden, ein Nutzer pro Session

Authentisierung bei konkurrierendem Zugriff

- Problemstellung:
  - ein systemweiter Namensraum
  - mehrere Nutzer
  - vorgegebene Schnittstelle (`open()`, `read()`, `write()`, ...)
  - nur `fsuid`, `fsgid` als Nutzerinformation
- Lösungsmöglichkeiten:
  - nur Systemauthentisierung, Server traut Nutzerinformation des Clients (NFS mit `AUTH_SYS`, Ceph, GPFS, ...)
  - Authentisierung beim Einbinden, alle Zugriffe serverseitig als ein Nutzer, ggf. clientseitig Zugriffrechte beschränken
  - Dateisystemschnittstelle um Authentisierungsinformation erweitern (AFS, `aklog`, ephemerische gids)
  - Heuristik für automatischen Zugriff auf hinterlegte Authentisierungsinformationen (NFS, Lustre, `mount.cifs+multiuser` mit Kerberos/GSSAPI)

Dateidienste und Kerberos

- GSSAPI/SSPI Basis für System- und Nutzerauthentisierung multiuser-fähiger Netzdateisysteme
- generische Schnittstellen, unterstützen beliebige Authentisierungssysteme
- für Dateisysteme praxisrelevanter Mechanismus: Kerberos
- bietet neben starker Authentisierung optional auch Integritätsprüfung und Verschlüsselung

> *Wer Dateidienste sicher in Mehrbenutzerumgebungen einbetten möchte, muss sich zwangsläufig mit Kerberos auseinandersetzen.*

tl;dr Kerberos

- Authentisierungssystem aus den 80ern
- Standardisiertes Protokoll, über die Jahre diverse Überarbeitungen und Erweiterungen
- quelloffene und proprietäre Implementierungen
- hoher Verbreitungsgrad in Enterprise-Umgebungen, integraler Bestandteil von Active Directory
- verwaltet langlebige (Passwörter, Schlüssel, Zertifikate) und kurzlebige (Tickets, Session Keys) Authentisierungsinformationen von Principals (Nutzer, Dienste)

Kerberos und die Konsequenzen

- alle Nutzer benötigen Kerberos-Principal
- nur noch Zugriff für Principals mit gültigem Ticket (Ausnahme: `nobody`, `system:anyuser`, `Jeder`)
- Tickets begrenzt gültig (typisch: 8-10h)
- Nutzer erhalten nach Passworteingabe neue Tickets

sicher ✔
alltagstauglich ✘

Alltägliche Hürden mit Kerberos

- Passworteingabe einmal täglich kein Problem für interaktive Nutzer
- aber was machen
  - lokale Accounts ohne Eintrag in der zentralen Nutzerverwaltung (Systemdienste)?
  - nicht-interaktive Nutzerprozesse (`cron`, `at`, Batch-Jobs)?
  - langlebige Nutzerprozesse?
- erschwerend: Art der Zugriffe in der Praxis häufig nur unvollständig bekannt

> *Wer kerberisierte Dateidienste einsetzen möchte, muss sich zwangsläufig mit nicht-interaktiven Zugriffen auseinandersetzen.*

Lösungswege für nicht-interaktive Zugriffe

1. Klein beigeben

- zurück zu AUTH_SYS (falls unterstützt, z.B. NFS, Lustre)
- `chmod -R 0777`

alltagstauglich ***?***
sicher ✘

2. Sicherheit per Definition

- Kerberos-Authentisierung im Hausnetz, AUTH_SYS im Datacenter
- "alle Knoten im HPC-Cluster sind sicher und vertrauenswürdig"
- falls unterstützt (z.B. NFS, Lustre)

alltagstauglich ✔
sicher ***?***

3. Client-Keytabs allerorten

- Keytab speichert Kerberos-Langzeitschlüssel im Dateisystem
- (fast) äquivalent zu Klartextpasswort in Datei
- etabliertes Standardverfahren
- für Service-Principals meist alternativlos
- auch sinnvoll, wenn Service-Accounts im eigenen Namen auf weitere kerberisierte Dienste zugreifen sollen
- Verwaltungsaufwand: Wo überall liegen Keytabs? Wie sind sie geschützt? Wer kümmert sich um Key-Rollover ("Passwortwechsel")? Braucht apache Zugriffsrechte auf Heimatverzeichnis jedes Nutzers?
- problematisch für interaktive Nutzer (Angriffsvektor auf Nutzerpasswort; Wechsel des Nutzerpassworts macht alle Keytabs ungültig)

alltagstauglich ***?***
sicher ***?***

4. Mit dem Kopf durch die Wand

- Passworteingabe abfangen
- Passwort in Truststore zwischenspeichern
- bei Bedarf damit Tickets anfordern
- nur tauglich für interaktive Nutzer
- Beispiele:
  - Windows
  - AUKS

alltagstauglich ✔
sicher **?**

5. Dem Höllenhund ins Maul geschaut

- Aktuelles Kerberos kann Principals berechtigen, stellvertretend im Namen anderer Identitäten tätig zu werden:
  - Mapping (Zuordnung Authentisierungsmerkmal zu Kerberos-Principal)
  - Delegation (eigene Kerberos-Identität aktiv weiterreichen)
  - Impersonifizierung (Kerberos-Principal darf andere Identitäten annehmen)
- Funktionsumfang implementierungsabhängig
- nur in Teilen standardisiert

> *Wer kerberisierte Dateidienste sicher und alltagstauglich einsetzen möchte, muss sich zwangsläufig mit den Untiefen von Kerberos-Erweiterungen auseinandersetzen.*

Public-Key-Erweiterung (PKINIT)

- Standardisierte Kerberos-Erweiterung (RFC 4556)
- ermöglicht Kerberos-Clients, Zertifikate als Langzeitschlüssel zu verwenden
- lässt sich grundsätzlich parallel zu Authentisierung mit Passwort benutzen
- Zertifikat bleibt auch bei Wechsel des Nutzerpassworts gültig
- PKINIT benötigt Abbildung von Identität in X.509-Zertifikat zu Kerberos-Principal
- 1:1-Abbildung laut RFC, aber auch andere Abbildungen zulässig
- *Certificate Mapping*: Zertifikat kann ein oder mehrere Kerberos-Identitäten annehmen
  - Active Directory: `altSecurityIdentities`
  - MIT Kerberos: `pkinit_cert_match`
- praktikable Lösung, wenn Dienst im Namen anderer Nutzer auf beliebige weitere Dienste zugreifen soll (Nutzerliste muss vorab bekannt sein)

alltagstauglich (✔)
sicher (✔)

Delegation

- Varianten: *Ticket Forwarding*, *Constrained Delegation*
- Nutzer reicht seine Kerberos-Identität aktiv an andere Dienste/Systeme weiter
- Dienst muss Delegation unterstützen
- Client (und ggf. KDC-Konfiguration) muss Delegation erlauben (*forwardable* Ticket)
- Zugriff im Namen des Nutzers auf beliebige (Ticket Forwarding) oder ausgewählte (Constrained Delegation) Dienste möglich
- mit Refresh-Mechanismus begrenzt auch für nicht-interaktive Zugriffe nutzbar
- Constrained Delegation beruht auf Microsoft-Erweiterung (`S4U2Proxy`), auf anderen Plattformen wenig unterstützt

alltagstauglich (✔)
sicher ✔

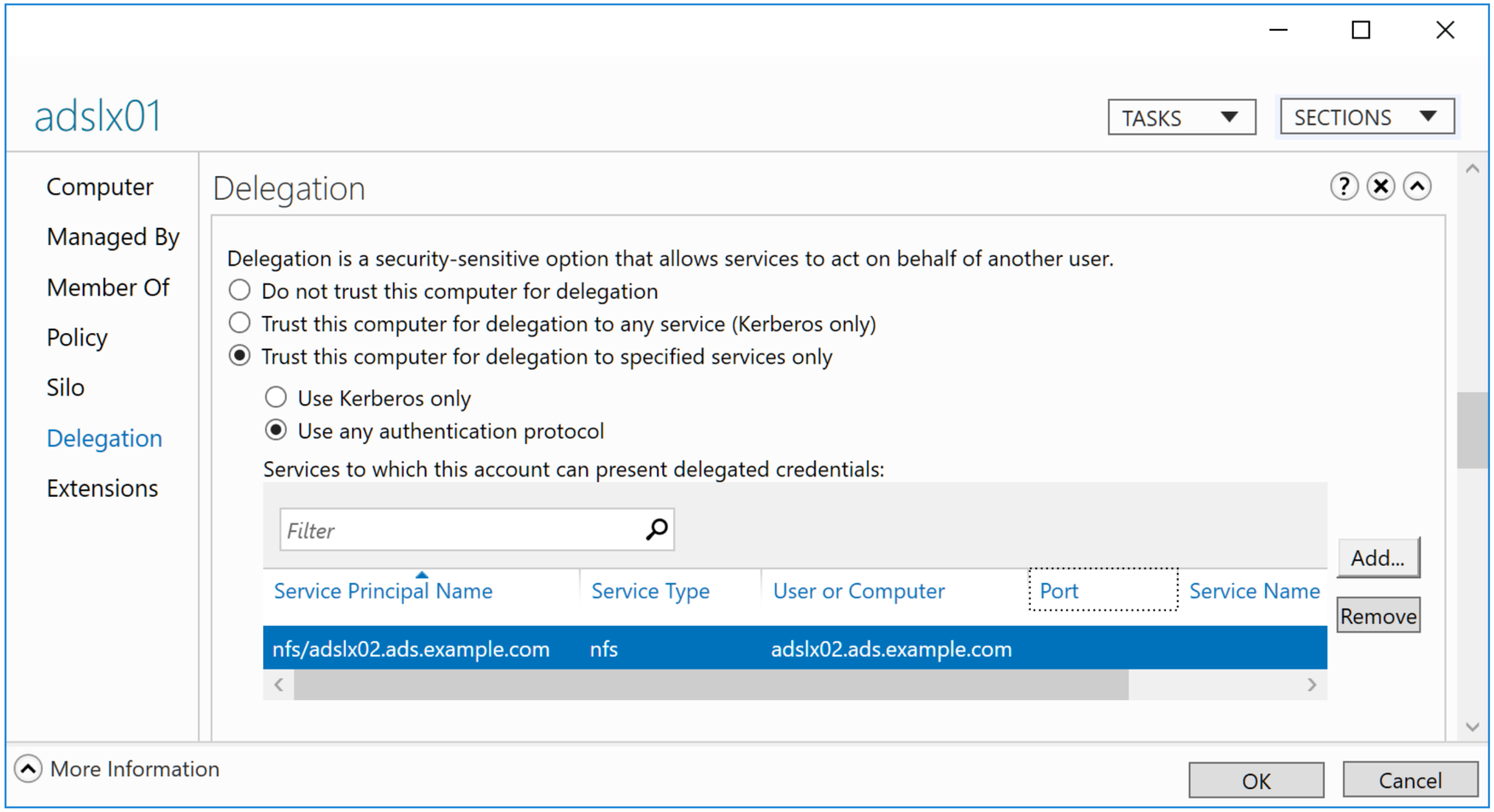
Impersonifizierung

- authentisierter Kerberos-Principal darf im Namen beliebiger anderer Principals auf ausgewählte Dienste zugreifen (*Protocol Transition*)
- Ausnahmen für besonders geschützte Principals möglich
- Protocol Transition für Maschinen-Principals ersetzt effektiv Kerberos-Authentisierung für Nutzer durch Kerberos-Authentisierung für Systeme
- aber: auditierbar!
- beruht auf Microsoft-Erweiterungen (`S4U2Self` und `S4U2Proxy`)
- auf KDC-Seite unterstützt von Active Directory, FreeIPA, MIT Kerberos (mit LDAP-Backend), Samba AD (nur RC4)
- Client-Unterstützung für Linux durch Plugin in `libgssapi`, keine Unterstützung für nicht-GSSAPI-Systeme (`mount.cifs`, AFS)

alltagstauglich ✔
sicher (✔)

gssproxy und sichere Dateisysteme

- KDC muss Maschinen-Principal eines Dateisystem-Clients Protocol Transition für Zugriffe auf Service-Principals des Dateisystems erlauben
- Beispiel:



gssproxy und sichere Dateisysteme

- gssproxy-Plugin für `libgss` aktivieren (`/etc/gss/mech.d/gssproxy.conf`):

```
# GSS-API mechanism plugins
#
# Mechanism Name  Object Identifier           Shared Library Path                                  Other Opt
gssproxy_v1       2.16.840.1.113730.3.8.15.1  /usr/lib/x86_64-linux-gnu/gssproxy/proxymech.so  <interpos
```

- Impersonifizierung in gssproxy aktivieren (`/etc/gssproxy/99-nfs-client.conf`):

```
[service/nfs-client]
  mechs  =  krb5
  cred_store  =  keytab:/etc/krb5.keytab
  cred_store  =  ccache:FILE:/var/lib/gssproxy/clients/krb5cc_%U
  cred_store  =  client_keytab:/var/lib/gssproxy/clients/%U.keytab
  cred_usage  =  initiate
  allow_any_uid  =  yes
  trusted  =  yes
  impersonate  =  true
  euid  =  0
```

- Umgebungsvariable GSS_USE_PROXY=yes setzen für Credential-Acquisition des Dateisystemclients (z.B. NFS: `rpc.gssd`, Lustre: `lgss_keyring`)

Fazit

- Kerberos bildet in Mehrbenutzersystemen üblicherweise die Basis für sichere Dateidienste
- Kerberos gewährleistet grundsätzlich starke Authentisierung von Nutzern und Systemen, Integritätsprüfung und Verschlüsselung
- Randbedingungen des Alltagsbetriebs erfordern Kompromisse in punkto Sicherheit
- Kerberos-Erweiterungen schaffen Flexibilität bei der Ausgestaltung der Kompromisse
- Unter Linux stellt `gssproxy` die nötigen Funktionen zur Impersonifizierung zentral für alle GSSAPI-basierten Dienste und Applikationen bereit

Vielen Dank

und

Auf Wiedersehen!

Daniel Kobras (kobras@puzzle-itc.de)
Puzzle ITC Deutschland GmbH (info@puzzle-itc.de)