File-Service mit Sync&Share-Zugriff – ownCloud mit Kerberos und NFSv4 Integration

Gregor van den Boogaart und Michael Roth

Rechenzentrum der Universität Augsburg
Universitätsstraße 8
86159 Augsburg
{cfs|gregor.boogaart|michael.roth}@rz.uni-augsburg.de

Zusammenfassung. Es wird ein Ansatz vorgestellt Cloud-Zugriff auf Daten in einem bereits bestehen File-Service zu ermöglichen. Der Zugriff durch den Cloudspeicher-Server auf den File-Service erfolgt über kerberisiertes NFSv4. Als Cloud-Software-Stack wird ownCloud verwendet. Um den NFSv4-Zugriff zu ermöglichen wurden kleinere Anpassungen am ownCloud-Code vorgenommen. Der Ansatz lässt sich aber auch auf andere Cloud-Software-Stacks übertragen.

1 Vorhandene Infrastruktur: zentraler File-Service

Das Rechenzentrum der Universität Augsburg betreibt einen als "Campus File System" (CFS) bezeichneten zentralen File-Service. Er ist campusweit an allen Arbeitsplätzen verfügbar und außerhalb des Campus via VPN erreichbar. Im CFS steht jedem Mitarbeiter und jedem Student ein Homelaufwerk für persönliche Daten zu Verfügung. Für gemeinsam genutzten Daten einer organisatorischen Einheit (Institut, Lehrstuhl, zentrale Einrichtung) stehen Share-Laufwerke zur Verfügung. Alle Daten liegen im CFS in einem einheitlichen, globalen Namensraum.

Das CFS speichert die Daten im Backend auf einem GPFS (General Parallel File System) von IBM. Der Zugriff der Clients erfolgt nicht als GPFS-Clients, sondern über SMB bzw. NFSv4. Dafür werden auf GPFS-Servern Export-Dienste betrieben: für SMB-Export wird Samba verwendet, für NFSv4-Export der NFS-Server im AIX. Die Zugriffsrechte werden über NFSv4 ACLs für Gruppen und Nutzer vergeben, die nativ im GPFS gespeichert sind. SMB-seitig sind die ACLs als Windows/NTFS ACLs sichtbar, NFSv4-seitig sind sie als native NFSv4 ACLs sichtbar. Um via SMB bzw. NFSv4 (gemäß der ACLs) zugreifen zu könnten muss sich ein Nutzer Client-seitig am zentralen Kerberos authentifizieren.

Die Datenhaltung ist damit vollständig zentralisiert und transparent, unabhängig davon über welches Protokoll bzw. von welchem Arbeitsplatz zugegriffen wird.

2 Ziel des neuen Dienstes

2.1 Ziel

Der vorhandene File-Service eignet sich für die Nutzung mit stationären Clients (Desktops und überwiegend stationär genutzte Laptops), die mit typischen Desktop-Betriebssystemen betrieben werden. Die SMB-Shares und NFSv4-Exports sind nur über das Universitätsnetzwerk bzw. via VPN erreichbar. Ein Sync&Share Zugriff von mobilen Clients bzw. über Web-Schnittstellen fehlt.

Die Daten der Nutzer liegen aber bereits im File-Service. Ein separat aufgebauter Cloudspeicher-Dienst erfordert entweder die Daten doppelt zu pflegen oder sie ganz oder teilweise zu migrieren. Ein direkter Zugriff mit Cloud-Methoden auf die Daten im File-Service würden helfen dieses Dilemma aufzulösen. Die im Backend verwendeten, mächtigen ACLs erlauben es prinzipiell, die für Sharing benötigten Rechte transparent abzubilden.

Ziel ist es deswegen, dass der neue Cloudspeicher-Dienst durch `SSL/TLS` geschützt aus dem öffentlichen Internet erreichbar ist und einen Sync&Share-Zugriff auf Home- und Share-Verzeichnisse im File-Service von überall erlaubt. Zusätzlich soll das Teilen von Daten sowohl mit Nutzern, die als Mitglied der Universität eine Kennung besitzen, als auch mit externen Nutzern möglich sein.

2.2 Zugriffsweg

Für diesen Zweck muss der Cloudspeicher-Dienst als zusätzlicher Export-Dienst (im folgenden: Cloud-Exporter) an den File-Service angebunden werden. Aktuell wird dafür ownCloud eingesetzt. Der hier beschrittene Weg sollte sich aber analog für alle im Source-Code veränderbaren, Web-basierten Cloudspeicher-Technologien anwenden lassen.

In der Infrastruktur des CFS gibt es prinzipiell zwei Möglichkeiten einem Cloud-Exporter Zugriff auf die Daten zu verschaffen:

Direkter Zugriff: Der Cloud-Exporter läuft mit root-Rechten direkt auf einem oder mehreren GPFS-Cluster Membern. Er hat damit Zugriff über die POSIX-Schnittstellen auf das File-System und kann unabhängig von ACLs alle Daten lesen und schreiben. Die Autorisierung wird komplett vom Cloud-Exporter übernommen.

Vorteile:
- Sharen mit beliebigen Nutzern ist damit einfach möglich, da Rechteverwaltung bei Cloud-Zugriff allein beim Cloud-Exporter liegt.

Nachteile:
- Berechtigungen laut Cloud-Exporter und Berechtigungen laut File-Service ACLs sind völlig losgelöst und damit intransparent.
- Der File-Service-Betrieb und der Betrieb des Cloud-Exporters sind bei Wartungsarbeiten eng verzahnt.
- Fehler im Security-Design des Cloud-Exporters haben unbegrenzte Auswirkungen.

Zugriff über Export-Dienst: Der Webserver läuft auf einem eigenen System und greift jeweils mit den Rechten des angemeldeten Users auf den File-Service zu. Dies ist prinzipiell via SMB oder via NFSv4 möglich.

Vorteile:
- Die Security-Escalation ist begrenzt.
- Der Betrieb von File-Service und Cloud-Exporter ist klar getrennt.
- Prinzipiell ist es durch ausgefeilte ALCs möglich, die Rechte-Struktur des Cloud-Exporters transparent in den ACLs des Filesystems abzubilden.

Nachteile:
- Sharen mit beliebigen Nutzern, die keine Kennung besitzen und folglich nicht persönlich bzw. über Gruppenmitgliedschaft über ACLs berechtigt werden können erfordert Anpassungen im Cloud-Exporter.

Um durch Fehler im Cloud-Exporter nicht das komplette CFS freizugeben haben wir uns entschlossen ownCloud nicht direkt an das GPFS anzubinden. Die Anbindung über den in ownCloud integrierten SMB-Client hatte in unseren Tests eine schlechte Performance. Zusätzlich wird das Benutzerpasswort in der PHP-Session gespeichert und war ebenfalls über die Prozessliste des Servers auslesbar. Aus diesen Gründen haben wir uns für eine Anbindung über NFSv4 entschieden. Da ownCloud zur Zeit noch kein NFSv4 mit Kerberos unterstützt, mussten wir Änderungen am Code vornehmen.

3 Integration in ownCloud: Umsetzung mit BindFS und NFSv4

Der ownCloud-Server hängt das komplettes CFS via NFSv4 an den Mountpoint `/cfs` ein. Dazu wird die Kerberos-Identität des Server verwendet.

```
root@owncloud~# mount -t nfs4 -o sec=krb5p nfs.uaux.de:/ /cfs
```

Damit ist das CFS prinzipiell vom Server aus erreichbar. Da jeder Nutzer im CFS nur auf die für die eigene Kennung berechtigten Elemente Zugriff hat, kann ownCloud mit dem lokalen Webserver-Nutzer nicht auf Daten eines CFS-Benutzers zugreifen. Um den Zugriff für den Webserver-Nutzer zu ermöglichen muss sich dieser gegenüber dem CFS als ein anderer Benutzer ausgeben, d.h. Impersonation durchführen.

3.1 Änderungen bei Login und Logout für Impersonation

Der Webserver-Nutzer `www-data` soll mit den Rechten des Users `USER` auf das CFS zugreifen, d.h. sich als dieser Benutzer ausgeben (Impersonation). Dies geschieht beim Login-Vorgang am ownCloud-Dienst in zwei Schritten.

Das Passwort wird durch einen Login-Hook abgefangen. Mit dem Befehl

```
www-data@owncloud:~$ sudo -u <USER> kinit
```

wird für *USER* ein Kerberos-Ticket angelegt. Anschließend wird das Passwort nicht mehr benötigt und wird deswegen nicht gespeichert. Danach kann der User *www-data* solange das Ticket gültig ist mit Kommandos der Form `sudo -u <USER> <cmd>` mit den Rechten des Nutzers unter dem Pfad `/cfs` auf den File-Service zuzugreifen.

Um die Impersonation abzuschließen und Dateien lesen und schreiben zu können ist es allerdings nötig ohne den Umweg über eine Shell und `sudo` zuzugreifen. Der zweite Befehl

```
www-data@owncloud:~$ sudo -u <USER> bindfs -M www-data /cfs /mnt/<USER>
```

hängt deswegen das komplette CFS über FUSE (File System in Userspace) mittels Loop-Mount im Impersonation-Pfad `/mnt/<USER>` erneut ein. Als FUSE-Modul wird bindfs verwendet, das mit den Rechten des Nutzers *USER* ausgeführt wird. Deswegen kann danach solange das Kerberos-Ticket gültig ist unter dem persönlichen Impersonation-Pfad mit den Rechten des Nutzers *USER* auf das CFS zugegriffen werden. Die Option `-M www-data` gaukelt dem Nutzer *www-data* dabei vor, Besitzer aller Objekte zu sein. Dadurch kann ownCloud auf das CFS wie auf ein lokales Dateisystem zugreifen.

Meldet sich der Benutzer von ownCloud ab wird der Impersonation-Pfad ausgehängt und das Kerberos-Ticket zerstört.

```
www-data@owncloud:~$ sudo -u <USER> umount /cfs
www-data@owncloud:~$ sudo -u <USER> kdestroy
```

Diese Änderungen wurden in der neuen App **user_kerberos** implementiert. Dazu wurde die in ownCloud enthalten App **user_external** als Vorlage benutzt und angepasst.

3.2 Änderungen im Dateizugriff

Das Wurzelverzeichnis eines Benutzers nennt ownCloud *primary storage*. ownCloud erlaubt darüberhinaus externen Speicher (*external storage*) einzubinden. Der primary storage kann in der aktuellen Version (ownCloud 8) nicht auf einen externen Speicher zeigen, d.h. der primary storage liegt im Moment außerhalb des CFS und wird sonst in der Testphase nicht benutzt. Um den Impersonation-Pfad als externen Speicher zu verwenden, wird er in ein Unterverzeichnis des primary storage eingehängt.

Dafür haben wir die App **files_external** um die Klasse `CFS` erweitert. Sie erbt alle Methoden von der Klasse `Local` für den lokalen Dateizugriff. Die einzige Änderung wurde im Konstruktor implementiert. Dort wird aus dem Benutzername `USER` der individuelle Pfad zum Homeverzeichnis `/mnt/<USER>/home/<USER>` generiert. Er wird beim Login automatisch im primary storage in das Unterverzeichnisse `Home` gelegt.

4 ACLs und Sharing

4.1 ownCloud Rechte & Sharing: CRUDS

ownCloud speichert die Zugriffsrechte für Dateien und Ordner in einer Datenbank. Dabei wird das CRUDS-Model (*C*reate, *R*ead, *U*pdate, *D*elete, *S*hare) verwendet. Der Besitzer einer Datei hat Vollzugriff, solange der eingebundene externe Speicher dies unterstützt. Wenn Dateien oder Verzeichnisse geteilt werden, werden die Rechte für den neuen Anwender ebenfalls mit dem CRUDS-Model abgebildet. Tabelle 1 zeigt die einzelnen Rechte im Bezug auf geteilte Dateien und Verzeichnisse und welche Bedeutung sie besitzen.

Kürzel	Bezeichnung	Dateien		Verzeichnisse		Bemerkung
		setzbar	Bedeutung	setzbar	Bedeutung	
`R`	read					
`S`	can share	X	re-share	X	re-share	
`CUD`	can edit	X		X		Sammelbegriff
`C`	create			X	add file/subdir	
`U`	change	X	write	X	touch?	a.k.a. update
`D`	delete			X	del file/subdir	

Tabelle 1. CRUDS-Rechte für wie sie im Sharing-Dialog gesetzt werden können und ihre Bedeutung

4.2 Back to the roots: POSIX/Unix mode, a.k.a. `chmod`

Unix-basierte Dateisysteme benutzen *mode bits* um die Zugriffsrechte zu speichern. Eine Datei kennt neben den mode bits noch einen Besitzer und eine Gruppe.

Dabei existieren drei Kategorien von Benutzern:

owner Dies ist der Besitzer der Datei.
group Alle Benutzer, die Mitglied in der Gruppe der Datei sind, ohne den Besitzer.
other Alle Benutzer, die weder Besitzer noch Mitglied in der Gruppe der Datei sind.

Für jede dieser drei Benutzerkategorien können die Rechte separat gesetzt werden. Dabei gibt es Lese-(r), Schreib-(w)- und Ausführungs-Rechte(x) als Zugriffsrechte. Zusätzlich können noch die Sonderrechte s und t gesetzt werden.

Tabelle 2 zeigt die Bedeutung der Rechte für Dateien und Verzeichnisse. Wichtig ist hierbei, dass sich die Rechte auf Verzeichnisse auf den Inhalt des Verzeichnisses bezieht. Ein Benutzer mit Schreibrechte auf einem Verzeichnis kann beliebige

Dateien innerhalb des Verzeichnisses anlegen und löschen. Ob er das Verzeichnis verändern kann wird anhand der Zugriffsrechte des übergeordneten Verzeichnissees bestimmt.

Bit	Datei	Verzeichnis	Bemerkung
			Je `user`, `group`, `other`
`r`	read	read/list (`stat`)	
`w`	write	write (`touch`, `mv`, `rm`)	
`x`	execute	search	
„special bits“			
`s` (`setuid`)	execute „as“ `user`		
`s` (`setgid`)	execute „as“ `group`		
`t`	`sticky` bit		Veraltet
`s` (`setuid`)		inherit `user`	neue (Sub-)Objekte
`s` (`setgid`)		inherit `group`	neue (Sub-)Objekte
`t`		restricted deletion flag	Unter Linux
		`rm` nur `owner` & `root`	

Tabelle 2. Zugriffs-Rechte im POSIX/Unix mode-Modell für Datei bzw. Verzeichnis Objekte.

4.3 NFSv4 ACLs vs. NT(FS) ACLs

Mit den NFSv4 Access Controll Listen *ACL* werden die Rechte noch feiner aufgeteilt. Die Listen bestehen aus Access Control Entries *ACE*. Jeder Eintrag setzt die Rechte für eine Gruppe oder einen Benutzer. Es gibt auch die Möglichkeit Rechte explizit zu verweigern. Dabei sind die NFS Rechte ähnlich den in Windows verwendeten NT(fs) ACLs. Die Namen sind identisch, allerdings gibt es Unterschiede bei der Bedeutung.

Anstatt die Rechte nur für einen Benutzer, eine Gruppe und den Rest zu setzten unterstützen NFS ACLs das verknüpfen mehrere Benutzer und Gruppen mit unterschiedlichen Rechten.

Tabelle 3 listet die vorhanden Rechte auf.

Die Windowsrechte werden in Tabelle 4 mit den NFS-Rechten verglichen.

4.4 CRUDS mit NFSv4 ACLs

Tabelle 5 modelliert die von ownCloud verwendeten CRUDS-Rechte mit NFSv4 ACLs nach.

Bit	file	directory	Bemerkung
`r`	read	list contents	
`w`	write/modify	add file	meist `w=a`
`a`	append	add subdir	meist `w=a`
`D`	–	del file/subdir	„delete child“
`d`	delete	delete	
`x`	execute	traverse/search	
`t`	read basic attrs		`stat`
`T`	change basic attrs		`stat` + Win
`n`	read named attrs		
`N`	write named attrs		
`c`	read ACL		
`C`	write ACL		
`o`	change owner		take ownership
`y`	sync primitive		Nur zur Speicherung

Tabelle 3. NFSv4 ACLs

NFSv4				Win Std Rechte-Sets		
Bit	file	directory	Win	Read	Change	Full Controll
`r`	read	list contents	R	X	X	X
`w`	write/modify	add file	W		X	X
`a`	append	add subdir	A		X	X
`D`	–	del file/subdir	DC			X
`d`	delete	delete	D		X	X
`x`	execute	traverse/search	E	X	X	X
`t`	read basic attrs		RA	X	X	X
`T`	change basic attrs		WA		X	X
`n`	read named attrs		REA	X	X	X
`N`	write named attrs		WEA		X	X
`c`	read ACL		RC	X	X	X
`C`	write ACL		P			X
`o`	change owner		O			X
`y`	sync primitive		S	X	X	X

Tabelle 4. NFSv4 ACLs und NT(FS) ACLs in Gegenüberstellung

NFSv4			ownCloud Sharing				
			single	whole directory			
Bit	file	directory	file	dir	sub-dir	sub-file	Bemerkung
r	read	list contents	R	R	R	R	
w	write/modify	add file	U	C	C	U	
a	append	add subdir	U	C	C	U	
D	–	del file/subdir		D	D		entweder ...
d	delete	delete			D	D	... oder
x	execute	traverse/search		R	R		für Unix nur dirs
t	read basic attrs		R	R	R	R	
T	change basic attrs		U			U	
n	read named attrs		R	R	R	R	
N	write named attrs		U			U	
c	read ACL		R	R	R	R	
C	write ACL		S	S	S	S	entweder ...
o	change owner		S?	S?	S?	S?	... oder?
y	sync primitive		R!	R!	R!	R!	wg. Einheitlichkeit setzen

Tabelle 5. Heutige CRUDS-Rechte modeliert mit NFSv4 ACLs

4.5 Rechtverwaltung bei Teilen

Da ownCloud nicht direkt auf alle Daten im CFS zugreifen kann, müssen für das Teilen einer Datei die Rechte angepasst werden. Um das Teilen zu erlauben muss ein Benutzer angemeldet sein, der Zugriff auf die Datei besitzt und Rechte setzen darf. Da ownCloud über bindfs mit dem passenden Kerberos-Ticket sich als dieser Benutzer ausgeben kann, kann ownCloud beim teilen einer Datei die Rechte anpassen.

Teilen mit Benutzer der Universität Sollte eine Datei mit anderen Besitzern einer Universitätskennung geteilt werden, so wird der neue Benutzer einfach mit den entsprechenden Rechten eingetragen. Hierbei kann durch entsprechende Rechte-Flags bestimmt werden, ob nur lesend oder ein schreib-lesend auf die Datei zugegriffen werden kann. Auch die Rechte die Datei weiter zu teilen lassen sich durch das Recht zur Rechteänderung abbilden.

Teilen mit externen Benutzern Sollte eine Datei mit einem Benutzer geteilt werden, der nicht Teil der Universität ist, so wird die Rechteverwaltung von ownCloud übernommen. Dazu muss eine Kerberos-Identität für den ownCloud-Benutzer existierten. Dieser wird dann in die List der Zugriffsberechtigten eingetragen.

5 Fazit

Durch die Anbindung von ownCloud an den vorhandenen File-Service ist der Zugriff für Mitarbeiter und Studenten auf ihre persönliche Daten auch über mobile Geräte möglich. Dabei ist die Rechteverwaltung transparent über die Dateisystem ACLs realisiert und ein Fehler in ownCloud führt nicht zu einem unbeschränkten Zugriff auf alle gespeicherten Dateien.

Wir betreiben ownCloud seit Anfang des Jahres in einer Testinstallation in der hier vorgestellten Konfiguration. Das Testsystem läuft auf einer virtuellen Maschine mit einer vCPU und 1 GB Arbeitsspeicher. Bei zehn Benutzern und zirka 120.000 Dateien haben wir bisher keine Performanceprobleme durch bindfs oder die NFSv4-Anbindung feststellen können.