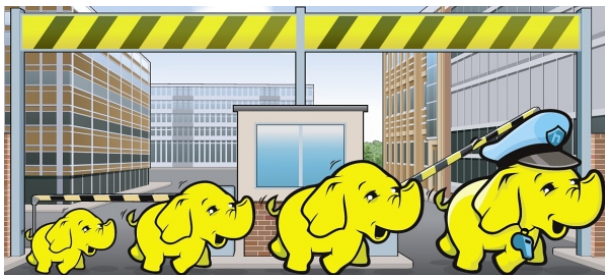


Hadoops *secure mode* – Fluch oder Segen?

Dr. Reiner Schlotte

r.schlotte@science-computing.de

TÜBIX 2016-06-11



About me

- ▶ Diplom-Physiker
- ▶ seit 18 Jahren bei science+computing ag
- ▶ Senior Solution Architect
- ▶ überwiegend CAE, HPC
- ▶ Hobby: Geocaching

Erfahrungshorizont und Blickwinkel

- ▶ Kundenprojekt seit etwa 2 Jahren
- ▶ Aufbau und Betrieb einer Vielzweck-Hadoop-Plattform
- ▶ Sichtweise eines Systemintegrators und Systemadministrators
– nicht eines Anwendungsentwicklers oder gar Hadoop-Entwicklers
- ▶ Erfahrungen beruhen größtenteils auf der *Pivotal Hadoop Distribution* PHD-3.0, die weitgehend identisch ist mit der *Hortonworks Data Platform* HDP-2.2 .

Hadoops Betriebsmodi

- ▶ standalone: 1 Maschine, keine laufenden Daemonen \Rightarrow nur für Spiel- und Entwicklungs-Zwecke interessant, sicherheitstechnisch kein Problem
- ▶ pseudodistributed: 1 Maschine, einige laufende Daemonen \Rightarrow nur für Entwicklungs-Zwecke interessant: Sandboxes
- ▶ fully distributed: 2 oder (viel) mehr Maschinen, viele laufende Daemonen \Rightarrow Produktion

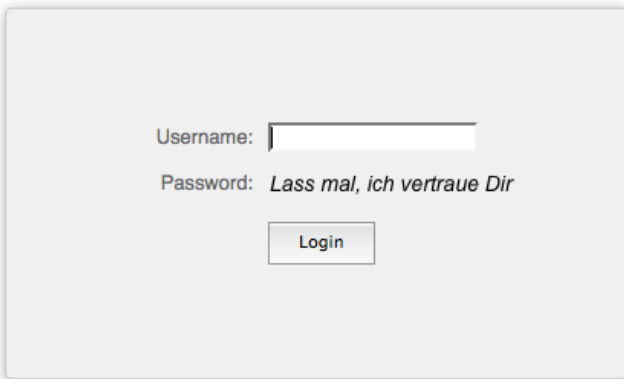
Hadoops Sicherheitsmodi

pseudodistributed und fully distributed modes
kennen wiederum 2 Sicherheits-Modi:

- ▶ simple Mode
- ▶ secure Mode

Simple Mode

KEINE Authentisierung: simple mode ist 100% unsicher!



A screenshot of the Hadoop Simple Mode login interface. It features a light gray rectangular box with a subtle drop shadow. Inside the box, the text "Username:" is followed by a white rectangular input field. Below this, the text "Password:" is followed by the pre-filled password *Lass mal, ich vertraue Dir*. At the bottom center of the box is a gray button with the text "Login".

Simple Mode

- ▶ Ich bin Donald Duck: Laß mal diesen Job laufen! \Rightarrow OK
- ▶ Ich bin Angela Merkel: Stoppe mal alle Jobs! \Rightarrow OK
- ▶ Ich bin Barack Obama: Zeig mir mal alle Daten! \Rightarrow OK
- ▶ Unix-Rechtekonzept (Gruppen, file permissions, umask ...) greift, *aber nur im HDFS!!!*
- ▶ Im simple mode laufen alle Hadoop-Worker-Prozesse (genauer: YARN-Container) als ein und der selbe User yarn: Ein Job kann trivial in die Daten des fremden laufenden Jobs schauen.
- ▶ Wer das Recht hat, eigenen Code einzuschleusen, kann beliebig Unfug treiben – und dieses Recht kann sich jeder beschaffen durch bloße Behauptung!

Simple Mode – Abwenden der Katastrophe

Wie bekommt man den simple mode (ein bißchen) sicher?

- ▶ den Hadoop-Cluster netzwerktechnisch abschotten
 - ▶ Firewalls
 - ▶ Verstecken im privaten Subnetz
- ▶ Zugriffe kanalisieren
 - ▶ authentifizierende Application Gateways bereitstellen: Hue, Knox
 - ▶ Autorisierung mit Ranger und Ranger-Plugins in Hue/Knox

Simple Mode – Application Gateways

- ▶ Hue (Hadoop User Experience): Web-GUI für klicki-bunti-Mausi
- ▶ Knox: REST-Interface – benötigt komplett andere Clients (wget, cURL ...) als Hadoop selbst
- ▶ Ranger: feingranulare Einstellung, welcher Benutzer was per Hue und Knox darf

Großer Nachteil dieser Gateways:

- ▶ Jede Zusatzkomponente im Hadoop-Ökosystem muß auch in Hue+Knox+Ranger integriert werden, ...
- ▶ ... und wenn's diese Integration nicht gibt \Rightarrow PROBLEM!

Simple Mode – Fazit

- ▶ Für den Unternehmenseinsatz ist der simple mode eigentlich vollkommen unbrauchbar, ...
- ▶ ... was die Unternehmen nicht davon abhält, ihn doch intensiv zu benutzen.

Secure Mode

Der secure mode wurde nachträglich in Hadoop eingebaut – ursprünglich war Security überhaupt kein Thema.

- ▶ Grundsatz 1: Jeder Hadoop-Worker-Prozess (genauer: YARN-Container) läuft als der Benutzer, der den Job abgeschickt hat!
 - ▶ Dafür müssen alle Benutzer auf OS-Ebene eingerichtet sein.
 - ▶ Unix-Rechtekonzept (Gruppen, file permissions, umask ...) greift überall.
 - ▶ YARNs container-executor ist setUID-root.
 - ▶ YARNs container-executor kennt eine blacklist von Usern: `hdfs`, `yarn`, `hive` ...

Secure Mode

- ▶ Grundsatz 2: Jeder Netzwerk-Zugriff muß authentifiziert werden!
 - ▶ Hadoop Tokens
 - ▶ Kerberos

Hadoop Tokens

- ▶ werden von einer zentralen Instanz (HDFS namenode) erteilt.
- ▶ haben einen begrenzten Gültigkeitszeitraum, können aber auch verlängert werden (renewal).
- ▶ müssen ggf. korrekt weitergereicht werden.
- ▶ können storniert werden.
- ▶ gibt's in verschiedenen Varianten: Delegation Token, Block Access Token, Job Token.
- ▶ sind im betrieblichen Alltag unkritisch:
 - ▶ Programmierer müssen sie aber sauber weiterreichen – alte Spark-Versionen erledigen das nicht richtig: Spark-Patch nötig.
 - ▶ In einem Oozie-Workflow muß bei den einzelnen Tasks angegeben werden, mit welchen Credentials (also Tokens) sie ausgeführt werden sollen. Wenn vergessen oder falsch ⇒ funktioniert nicht.

Hadoop Tokens – Praktisches Beispiel 1

auf der Kommandozeile:

```
> hdfs fetchdt --webservice http://srv1:50070 /tmp/r.token
Fetched token via http://srv1:50070 for 1.2.3.4:50070 into file:/tmp/r.token
> ls -la /tmp/r.token
-rw-r----- 1 reiner user 111 Jun 6 11:09 /tmp/r.token
> hdfs fetchdt --print /tmp/r.token
Token (HDFS_DELEGATION_TOKEN token 1825998 for reiner) for 1.2.3.4:50070
> hdfs fetchdt --webservice http://srv1:50070 /tmp/r.token
Fetched token via http://srv1:50070 for 1.2.3.4:50070 into file:/tmp/r.token
> hdfs fetchdt --print /tmp/r.token
Token (HDFS_DELEGATION_TOKEN token 1826004 for reiner) for 1.2.3.4:50070
```

Hadoop Tokens – Praktisches Beispiel 2

im YARN-Container:

```
> echo $HADOOP_TOKEN_FILE_LOCATION
/etc/hadoop/datadirs/fs3/yarn/local/usercache/reiner/appcache/\
application_1464968327601_3961/\
container_1464968327601_3961_01_000002/container_tokens
> ls -la $HADOOP_TOKEN_FILE_LOCATION
-rw----- 1 reiner hadoop 335 Jun 6 11:16 /etc/hadoop/datadirs/fs3/yarn/loc
> hdfs fetchdt --print $HADOOP_TOKEN_FILE_LOCATION
```

Hadoop Tokens – Praktisches Beispiel 2

im YARN-Container:

```
> echo $HADOOP_TOKEN_FILE_LOCATION
/etc/hadoop/datadirs/fs3/yarn/local/usercache/reiner/appcache/\
application_1464968327601_3961/\
container_1464968327601_3961_01_000002/container_tokens
> ls -la $HADOOP_TOKEN_FILE_LOCATION
-rw----- 1 reiner hadoop 335 Jun 6 11:16 /etc/hadoop/datadirs/fs3/yarn/loc
> hdfs fetchdt --print $HADOOP_TOKEN_FILE_LOCATION
Exception in thread main java.io.IOException:
Unknown version of delegation token 10
at org.apache.hadoop.security.token.delegation.AbstractDelegationTokenIdentifi
at org.apache.hadoop.hdfs.tools.DelegationTokenFetcher$1.run(DelegationToken
```

Da ist also irgendwas inkonsistent programmiert! ☹️

Hadoop Token Bootstrap: Kerberos

F: Wie bekommt der Benutzer auf *sichere* Weise übers Netzwerk sein (erstes) Hadoop Token?

A: Indem die erste Netzwerkverbindung per Kerberos authentisiert wird!



Kerberos ist klasse!

- ▶ Einmal mit `kinit` ein TGT (Ticket Granting Ticket) besorgen – passiert beim Einloggen/Screenunlock automatisch.
- ▶ Danach nie wieder (d.h. 10 Stunden lang) sein Paßwort eintippen müssen.
- ▶ Unter der Motorhaube besorgen sich die Clients unter Benutzung des TGT die nötigen Service Tickets.
- ▶ Kerberos ist ausgereift.
- ▶ Kerberos ist Kernkomponente von Active Directory – im Unternehmen leicht verfügbar.
- ▶ Kerberos ist ein Netzwerk-Authentisierungsmechanismus – keine Autorisierung!
- ▶ Kerberos-Authentisierung ist immer zweiseitig.

Voraussetzungen für Kerberos

- ▶ Uhren sind synchronisiert \Rightarrow kein Problem
- ▶ Namensauflösung stimmt vorwärts und rückwärts \Rightarrow kein Problem
- ▶ Verschlüsselungstypen passen zueinander \Rightarrow *oha*, siehe unten

Kerberos – ein paar Begriffe

- ▶ Realm: Verwaltungseinheit von Kerberos, z.B. `HADOOP.EXAMPLE.COM`
- ▶ KDC = Key Distribution Center: zentraler Authentisierungsdienst, jedes Realm hat ein KDC
- ▶ Principal: Benutzer oder Service, z.B. `reiner@EXAMPLE.COM` oder `HTTP/srv1.example.local@HADOOP.EXAMPLE.COM`
- ▶ Keytabs: eine Datei mit einem Paßwort(-äquivalent): Dienste brauchen sowas, Benutzer i.d.R. nicht. Kann der Admin mit dem KDC schreiben.
- ▶ Cross Realm Trust: gegenseitiges Vertrauen (eine oder beide Richtungen) zwischen 2 Realms

Kerberos Debugging

Debugging eines Kerberos-Setup ist nicht ganz einfach:

- ▶ Wenn irgendwas nicht funktioniert, heißt es immer nur 'Authentisierung fehlgeschlagen', aber der genaue Grund ist kaum ersichtlich.

Hadoop und Realms

Designentscheidung: eigenes Kerberos-Realm für Hadoop,
dazu Cross Realm Trust zum Unternehmens-Active-Directory

- ▶ wird für Hadoop so empfohlen
- ▶ entlastet die Active-Directory-Server
- ▶ entlastet das Active-Directory-Betriebsteam
- ▶ mehr Handlungsspielraum für das Hadoop-Betriebsteam
- ▶ kein Benutzer-Management im MIT-Kerberos-Realm nötig
- ▶ MIT-Kerberos wird bei Linux mitgeliefert

Herausforderung: Anlegen der Principals und der Keytabs

- ▶ veeeeele principals! – pro Maschine ca. 20 \Rightarrow von Hand aussichtslos
- ▶ Ambari (und andere Hadoop-Admin-Tools) helfen einem kaum dabei – eigene Skripte programmiert
- ▶ Viele Hadoop-Services benötigen nicht nur den Service-eigenen Principal
(`yarn/srv1.example.local@HADOOP.EXAMPLE.COM`), sondern auch den HTTP-Principal
(`HTTP/srv1.example.local@HADOOP.EXAMPLE.COM`)
- ▶ Bei alten Versionen von MIT-Kerberos bewirkt das Schreiben einer Keytab für einen Principal automatisch einen Paßwort-Reset, d.h. vorherige Keytabs werden ungültig.
- ▶ Skripte sind komplex und benutzen intensiv `ktutil`

Herausforderung: Einrichten des Cross Realm Trust

Eigentlich ist das Einrichten eines Cross Realm Trusts ganz einfach, denn die Schnittstelle ist ganz dünn:

- ▶ Client-Config passend schreiben: `/etc/krb5.conf`
- ▶ auf beiden KDCs einen ganz bestimmten Principal anlegen mit *identischen* Paßwörtern
- ▶ eine Hadoop-Config anpassen
- ▶ Aber es muß wenigstens einen gemeinsamen encryption type in beiden Realms geben! ⇒ beim Kunden AD zu fortschrittlich, Linux zu rückständig – einiges Hin und Her, schließlich gelöst

Herausforderung: Konfiguration der Hadoop-Services

- ▶ Da muß jedes i-Tüpfelchen stimmen; jeder Tippfehler führt zur Höchststrafe: `GSSAPI error`.
- ▶ Jeder Service liest nur 1 Keytab, auch wenn er 2 Principals nutzt (xyz und HTTP):
`dfs.datanode.keytab.file=/etc/security/keytabs/dn.service.keytab`
- ▶ HTTP-Principal steht in vielen Keytabs identisch drin – Problem siehe oben.
- ▶ Jeder Service ist auch Client von anderen Services:
 - ▶ Welchen Service-Principal soll ich selbst benutzen, d.h. aus der Keytab auslesen?
 - ▶ Für welchen Service-Principal muß ich mir ein Kerberos-Service-Ticket besorgen?
- ▶ Dafür gibt's aber nur einen gemeinsamen Config-Parameter pro Service, z.B. `dfs.datanode.kerberos.principal`

Hostspezifische Principals

Design-Entscheidung: Services nutzen hostspezifische Principals!

~~yarn@HADOOP.EXAMPLE.COM~~

yarn/srv1.example.local@HADOOP.EXAMPLE.COM

Damit das funktioniert, muß man im Wert von ...principal die Variable `_HOST` benutzen: `dn/_HOST@HADOOP.EXAMPLE.COM`

- ▶ ferner Host: `_HOST` ergibt sich aus der Config
- ▶ eigener Host: `_HOST=$(hostname)`
- ▶ daraus folgt: Das Netzwerk-Interface muß so heißen wie der hostname!
- ▶ Und was ist mit multi-homing???

Hostspezifische Principals – Lesson Learned

- ▶ secure mode und multi-homing vertragen sich nicht!
- ▶ nur eine IP-Adresse pro Maschine!
- ▶ Finger weg von Spezial-Hardware mit zusätzlichen Interconnects!
- ▶ höchstens: Channel Bonding

Herausforderung: Erziehung der Benutzer

- ▶ Beim Kunden ist ein kinit schon beim Login nicht möglich.
- ▶ Stattdessen: kleines Script adkinit
- ▶ Benutzer vergißt das adkinit
⇒ Höchststrafe: GSSAPI error
- ▶ Flut von Incidents ans User Helpdesk:
'Hadoop ist kaputt!'
- ▶ Abhilfe: tcsh-precmd

```
> alias precmd  
klist -s || echo 'WARNING: You have no valid Kerberos  
ticket! Use adkinit to obtain one.'
```

Herausforderung: secure mode *komplett* scharfschalten

- ▶ Ambari security wizard schaltet seltsamerweise nicht alle Dienste in den secure mode.
- ▶ Hue benutzt zum Killen eines YARN-Jobs eine andere Schnittstelle als zum Starten und Überwachen.
- ▶ Folge: Das Killen von YARN-Jobs funktioniert nicht aus Hue heraus; das Dialogfenster bleibt einfach stehen.
- ▶ Workaround: CLI: `yarn kill -applicationId`

Herausforderung: secure mode im privaten Subnetz

- ▶ privates Subnetz ist dank NAT und Portforwarding eigentlich kein Problem
- ▶ aber NAT und Kerberos vertragen sich ganz schlecht.
- ▶ Lösungsansatz: Hue und Knox

secure mode im privaten Subnetz – Lesson Learned

- ▶ secure mode Hadoop-Cluster ins Corporate Network stellen, nicht ins private Subnetz!
- ▶ Firewalls kann man zur Not öffnen.

Tidbits – Oracle Java

- ▶ Die Default-Installation von Oracle Java erlaubt nur schwache Verschlüsselung.
- ▶ \Rightarrow 2 Dateien nachinstallieren: JCE policy files
- ▶ Nicht irritieren lassen, wenn die Dateien scheinbar schon da sind – sind sie nicht!

Tidbits – Hive

Hive kennt 2 Betriebsmodi:

- ▶ SQL Standards Based Hive Authorization – analog zu klassischen Datenbanken
- ▶ Storage Based Authorization – Zugriffsrechte im HDFS greifen

Kunde wählt Storage Based Authorization:

- ▶ Alle Zugriffsrechtsprobleme schlagen voll durch.
- ▶ Hive braucht gelegentlich unerwartet Schreibrechte auch bei lesendem Zugriff.
- ▶ derzeit noch offene Baustellen und unverstandene Probleme

Tidbits – container-executor 1

- ▶ container-executor: kennt blacklist und whitelist
- ▶ aber in Ambari nur blacklist konfigurierbar

Tidbits – container-executor 2

container-executor:

- ▶ kennt 4 Betriebsmodi
- ▶ 2 sind direkt in C implementiert – unproblematisch
- ▶ 2 weichen auf Java aus
- ▶ bei einem von beiden wird Javas Heap Size Limit nicht gesetzt
- ▶ Debugging zog sich über Wochen hin, am Ende: bekannter Bug
- ▶ Bug zeigt sich nur unter bestimmten seltenen Randbedingungen – die bei uns vorlagen. ☹️

Tidbits – Snakebite

- ▶ alternativer HDFS-Client in Python
- ▶ *wesentlich* schneller als das Hadoop-hdfs, weil JVM-Initialisierung wegfällt
- ▶ funktionierte anfangs, dann mit neuer Hadoop-Version nicht mehr
- ▶ Ursache: Principals waren hardwired hdfs anstatt aus Config-Files ausgelesen nn, dn
- ▶ immer noch unvollständig implementiert

Tidbits – distcp

Hadoop-distcp (distributed copy):

Tool zum parallelen Kopieren von Daten zwischen 2 Clustern.

- ▶ nicht ausreichend konfigurierbar:
 - ▶ Die beiden Cluster müssen bezüglich Kerberos identisch konfiguriert sein.
 - ▶ Insbesondere: Die beiden Cluster müssen im gleichen Kerberos-Realm stehen.
- ▶ Workaround laut Handbuch: 1 Cluster in simple mode bringen – betrieblich nicht umsetzbar
- ▶ Bei Datenmanagement-Aufgaben muß distcp als superuser `hdfs` laufen (analog `rsync` als `root`), ...
- ▶ ... aber `hdfs` steht normalerweise in der blacklist des container-executors.

Tidbits – verschiedene Kaufprodukte

- ▶ sind laut Werbung Hadoop-kompatibel, ...
- ▶ ... aber nur mit dem simple mode.
- ▶ Das steht natürlich nicht in der Werbung!

Tidbits – Spark

- ▶ eigentlich unabhängig von Hadoop
- ▶ kann aber on top of Hadoop (YARN) laufen
- ▶ wesentlich performanter als MapReduce
- ▶ deutlich jünger
- ▶ noch nicht komplett an secure mode angepaßt
 - ▶ Weiterreichen von Tokens teilweise fehlerhaft
 - ▶ Integration in Hue+Knox unreif
 - ⇒ Spark-Benutzer brauchen CLI
 - ⇒ neue Sicherheitsfragen

Tidbits – Ambari

Ambari unterstützt in vielen Kleinigkeiten den secure mode nicht richtig:

- ▶ häufig: Hadoop-Parameter fehlen
- ▶ Einzelfall: Scheinbar frei konfigurierbarer Parameter ist im RPM-Paket festgenagelt.
- ▶ schneidet leading/trailing whitespace weg, der aber für YARN-ACLs gelegentlich notwendig ist.
- ▶ hält `hive-site.xml` für Client und Server identisch, was aber falsch ist, u.a. wegen Paßwörtern.

⇒ Ambari ist schön anzuschauen, aber für ernsthafte Arbeit kaum zu gebrauchen.

Tidbits – Verschlüsselung

secure mode = Authentisierung
keine Verschlüsselung der Nutzdaten!

Verschlüsselung der Nutzdaten

- ▶ beim Transport im Netzwerk
- ▶ auf der Platte

sind zusätzliche Schritte!

Hue und Knox sind HTTPS-tauglich.

Hadoops secure mode – Fluch oder Segen?

Ganz klar: beides!

Segen: löst (im Prinzip) die eklatanten Sicherheitsprobleme des simple mode

Fluch: Komplexität steigt *deutlich* an, Know How ist dünn gesät, *auch* bei den Distributoren

Fluch: unreif: schwierig zu konfigurieren, viele wünschenswerte Features fehlen noch

Segen: verschafft IT-Consultants (uns! ☺) Lohn und Brot

Zum Vertiefen

<http://carfield.com.hk/document/distributed/hadoop-security-design.pdf>
https://github.com/steveloughran/kerberos_and_hadoop/blob/master/sections/hadoop_tokens.md
<http://hortonworks.com/blog/the-role-of-delegation-tokens-in-apache-hadoop-security/>
<https://www.infoq.com/articles/HadoopSecurityModel>

Fragen?