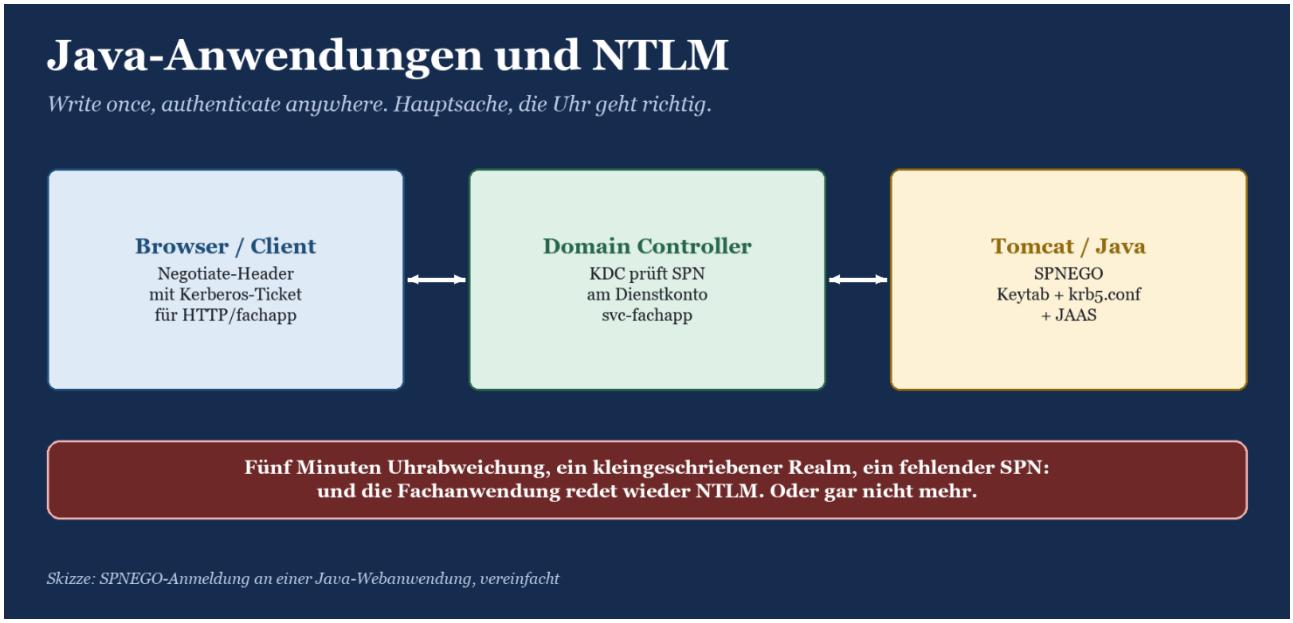
Java-Anwendungen und NTLM: Kerberos mit SPNEGO



WISSEN
Was der NTLM Diagnostiker erfasst, wie er arbeitet und was er dir zeigt.
› **NTLM Diagnostiker**

BERATUNG
Fachanwendung auf Kerberos umstellen, Hersteller einbinden: Wir planen das mit dir.
› **Beratung zur NTLM-Ablösung**

SCHULUNG
Ereignisse lesen, Verursacher finden, Ausnahmen begründen: praxisnah für dein Team.
› **Schulung zum NTLM Diagnostiker**

Jede Organisation hat sie: die Fachanwendung auf Java-Basis. Im Krankenhaus heißt sie Laborsystem, in der Verwaltung Fachverfahren, im Maschinenbau Produktionsplanung. Sie läuft auf einem Tomcat, den 2016 ein freundlicher Herstellertechniker installiert hat, sie meldet die Anwenderinnen und Anwender „ganz bequem mit Windows-Anmeldung“ an, und niemand hat je gefragt, was „Windows-Anmeldung“ technisch bedeutet. Die Antwort lautet erstaunlich oft: NTLM. Über eine Bibliothek, deren letzter Commit älter ist als dein Dienstwagen.

Genau darum geht es in diesem Beitrag zu java ntlm kerberos: Wo NTLM in Java-Anwendungen herkommt, wie du eine Java-Webanwendung per SPNEGO mit Keytab und Dienstkonto sauber auf Kerberos bringst, wie eine krb5.conf aussehen muss, und an welchen drei Klassikern es fast immer scheitert: Uhrzeit, SPN und die Groß- und Kleinschreibung des Realms. Und weil sich manches nicht per Konfiguration lösen lässt, geht es am Ende um den Teil, den keiner gern macht: den Brief an den Hersteller.

Der Zeitdruck ist bekannt: Seit dem Windows-Update vom 13.10.2026 blockieren Windows 11 ab 24H2 und Windows Server 2025 NTLMv1 für Single Sign-on standardmäßig, und mit der nächsten Windows-Server-Version ist NTLM standardmäßig abgeschaltet. Den Fahrplan findest du in NTLM-Abschaltung: Microsofts Fahrplan und was er für dich bedeutet. Für deine Java-Fachanwendung heißt das: Was heute „irgendwie mit Windows-Anmeldung“ funktioniert, funktioniert dann womöglich gar nicht mehr. Und die Anwenderinnen und Anwender werden nicht den Hersteller anrufen, sondern dich.

FAKTEN · Das Wichtigste in Kürze

Java bringt Kerberos seit Langem mit (JGSS, Krb5LoginModule). NTLM kommt meist aus Zusatzbibliotheken wie JCIFS, Waffle im NTLM-Modus oder dem NTLM-Schema von HTTP-Clients.

Für eine Java-Webanwendung heißt der Kerberos-Weg SPNEGO: Dienstkonto mit SPN HTTP/<FQDN>, Keytab mit AES-Schlüsseln, krb5.conf mit Realm in Großbuchstaben, JAAS-Eintrag für die Annahme.

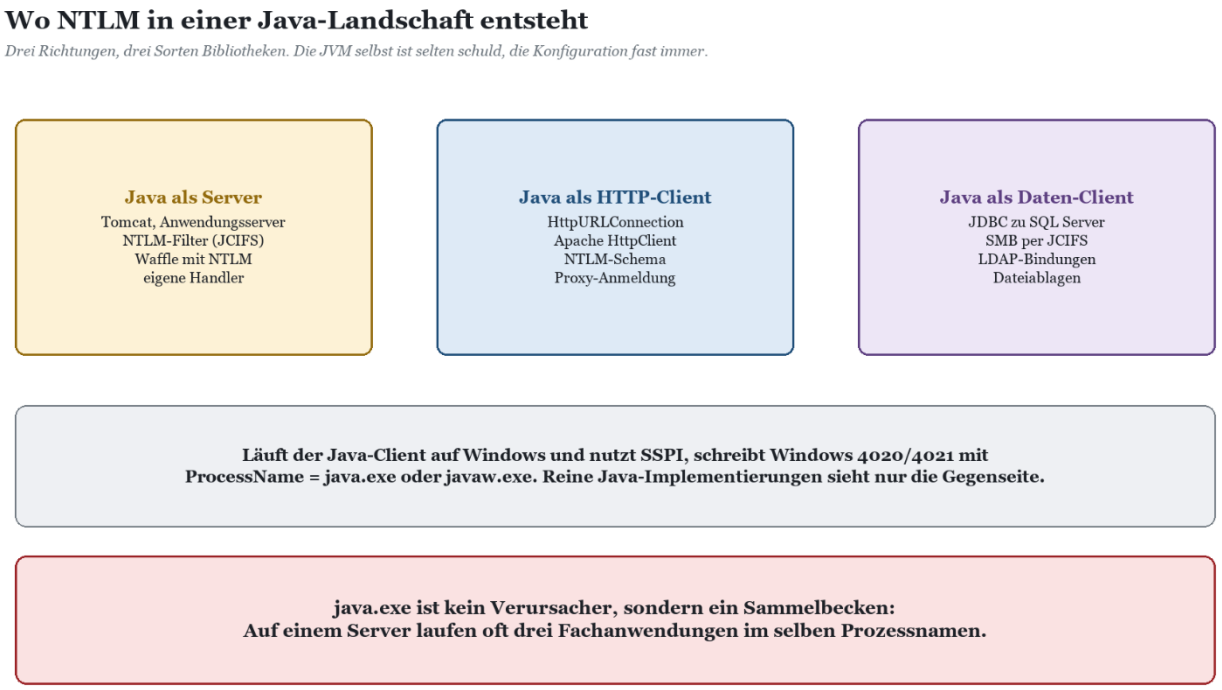
Ein reiner Kerberos-Authenticator macht kein NTLM. Schickt der Browser trotzdem ein NTLM-Token, siehst du im Java-Log Meldungen wie „Defective token detected“, im Browser eine Fehlerseite.

Die drei Klassiker: Uhrzeit weicht mehr als fünf Minuten ab, der SPN fehlt oder ist doppelt, der Realm ist klein geschrieben.

java.exe als ProcessName in 4020/4021 sagt dir nur, dass Java beteiligt ist, nicht welche Anwendung. Die Zuordnung braucht Ziel, Dienst und Kommandozeile.

Wo NTLM in Java-Anwendungen herkommt

Die gute Nachricht zuerst: Die Java-Laufzeit kann Kerberos. Das Paket für GSS-API und das Login-Modul com.sun.security.auth.module.Krb5LoginModule gehören seit Jahren zum Lieferumfang, SPNEGO ebenfalls. Wenn eine Java-Anwendung NTLM spricht, liegt es daher fast nie an der JVM, sondern an einer Bibliothek, die jemand bewusst eingebaut hat, oder an einer Konfiguration, bei der Kerberos still scheitert und der Rückfall greift. Welche Variante du vor dir hast, hängt davon ab, in welcher Richtung Java gerade unterwegs ist.



Skizze: typische NTLM-Quellen in Java-Umgebungen, beispielhaft für die Musterwerk GmbH

Java als Server: Tomcat und Anwendungsserver

Auf der Serverseite nimmt die Java-Anwendung eine Anmeldung entgegen. Ältere Fachanwendungen verwenden dafür gern den NTLM-HTTP-Filter aus JCIFS. Der beantwortet die Anfrage des Browsers mit einer NTLM-Aufforderung und reicht die Antwort zur Prüfung an einen Domain Controller durch. Das Projekt hat die NTLM-HTTP-Unterstützung schon vor Jahren als überholt eingestuft, in Fachanwendungen lebt sie trotzdem weiter. Unter Windows läuft häufig

Waffle, das die Windows-Schnittstelle SSPI nutzt. Waffle kann Negotiate und damit Kerberos, nimmt aber auch NTLM an, wenn der Browser es anbietet oder die Konfiguration nur NTLM vorsieht. Dazu kommen eigene Handler der Hersteller, deren Innenleben du nur aus der Dokumentation oder dem Log kennst.

Woran du die Seite erkennst: Bei JCIFS auf einem Linux-Tomcat entsteht auf Windows-Seite nur das, was der Domain Controller bei der Prüfung protokolliert, also 8004, 4030 bis 4033 und das Sicherheitsereignis 4776, wie in Ereignis 4776 auf dem Domain Controller: NTLM-Anmeldungen im Sicherheitsprotokoll beschrieben. Bei Waffle auf einem Windows-Server schreibt der Server eingehende Ereignisse 4022/4023 beziehungsweise 8002/8003, weil die Anmeldung über SSPI läuft.

Java als Client: HTTP, JDBC und SMB

Auf der Clientseite meldet sich die Java-Anwendung selbst irgendwo an. Klassiker sind HttpURLConnection, die unter Windows NTLM über die Systemanmeldung abwickeln kann, das NTLM-Schema im Apache HttpClient, die Anmeldung an einem Proxy und der Zugriff auf SMB-Freigaben über JCIFS. Für SQL Server unterscheidet der Microsoft-JDBC-Treiber über die Eigenschaft authenticationScheme ausdrücklich zwischen JavaKerberos, NativeAuthentication und NTLM. Mit integratedSecurity=true und authenticationScheme=JavaKerberos verwendet der Treiber die Java-eigene Kerberos-Implementierung; die Details beschreibt Microsoft unter Kerberos-integrierte Authentifizierung mit dem JDBC-Treiber für SQL Server. Die SQL-Server-Seite des Themas, also SPNs für MSSQLSvc und Dienstkonten, steht in SQL Server und NTLM: Windows-Anmeldung auf Kerberos umstellen.

Baustein	Richtung	NTLM-Verhalten	Kerberos-Weg
JCIFS NTLM-HTTP-Filter	Server	nur NTLM, Durchreichung an den DC	ersetzen durch SPNEGO (Tomcat, Spring Security Kerberos o. ä.)
Waffle	Server (Windows)	Negotiate oder NTLM über SSPI, je nach Konfiguration	Negotiate mit SPN am Dienstkonto, NTLM-Anbieter entfernen
Tomcat SPNEGO-Authenticator	Server	kein NTLM, nur Kerberos über SPNEGO	Keytab, krb5.conf, jaas.conf
HttpURLConnection	Client	kann unter Windows NTLM mit der Systemanmeldung nutzen	Negotiate mit Ticket, Zielname als FQDN
Apache HttpClient	Client	eigenes NTLM-Schema, oft fest konfiguriert	SPNEGO-Schema bzw. Kerberos-Konfiguration prüfen
Microsoft JDBC-Treiber	Client	authenticationScheme=NTLM möglich	integratedSecurity=true, authenticationScheme=JavaKerberos
JCIFS (SMB)	Client	NTLM mit Kennwort aus Konfiguration	Kerberos nur bei neueren Abspaltungen und passender Konfiguration

WARNUNG · Kennwort in der Konfigurationsdatei

Viele Java-Clients, die NTLM sprechen, holen Benutzername und Kennwort aus einer properties-, XML- oder YAML-Datei. Das ist nicht nur NTLM, das ist ein Kennwort im Klartext auf einem Server. Wenn du ohnehin umstellst, nimm Keytab oder Ticket-Cache und lösch die alte Zeile. Wirklich löschen, nicht auskommentieren. Auskommentierte Kennwörter sind auch Kennwörter.

java.exe ist kein Verursacher, sondern eine Ausrede

Läuft ein Java-Client auf Windows und nutzt SSPI, schreibt Windows ein ausgehendes Ereignis 4020 oder 4021 mit ProcessName = java.exe oder javaw.exe. Das klingt nach einem Treffer, ist aber nur eine halbe Spur. Auf einem Anwendungsserver laufen oft mehrere Java-Dienste im selben Prozessnamen: die Fachanwendung, der Tomcat des Druckdienstes, das Monitoring. Die andere Hälfte der Spur steckt in TargetMachine, TargetService und dem Begründungstext in NtlmUsageReason. Maßgeblich ist dieser Text, die Reason-ID ist nur ein Hinweis, denn Nummer und Text passen in der Praxis nicht immer zusammen. Was die Begründungen bedeuten, steht in NTLM-Reason-IDs erklärt: warum Windows nicht Kerberos benutzt hat.

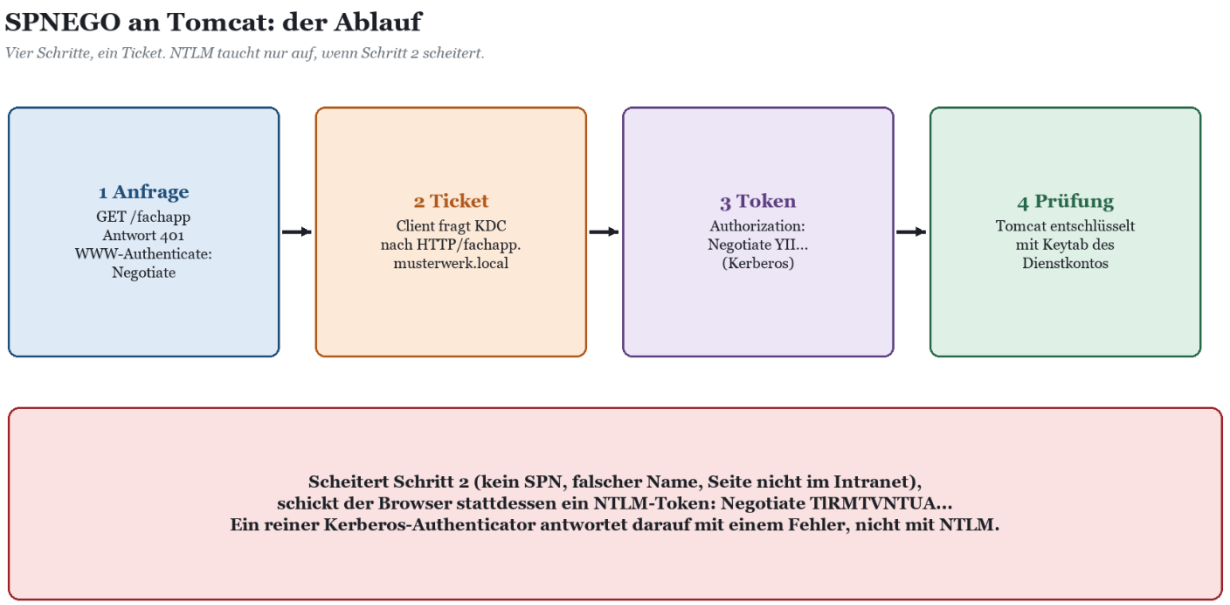
```
# Ausgehende NTLM-Ereignisse von Java-Prozessen der letzten 7 Tage, gruppiert nach Ziel und
Grund
Get-WinEvent -FilterHashtable @{ LogName = 'Microsoft-Windows-NTLM/Operational';
    Id = 4020,4021; StartTime = (Get-Date).AddDays(-7) } |
  ForEach-Object { $x = [xml]$_.ToXml(); $f = @{}
    $x.Event.EventData.Data | ForEach-Object { $f[$_.Name] = $_.'#text' }
    if ($f['ProcessName'] -match 'javaw?\.exe$') {
      [pscustomobject]@{ Prozess = $f['ProcessName']; Ziel = $f['TargetMachine'];
        Dienst = $f['TargetService']; Grund = $f['NtlmUsageReason'] } } } |
  Group-Object Prozess, Ziel, Dienst, Grund | Sort-Object Count -Descending

# Welche Java-Prozesse laufen hier, und was steht in der Kommandozeile?
Get-CimInstance Win32_Process -Filter "Name='java.exe' OR Name='javaw.exe'" |
  Select-Object ProcessId, @{n='Dienst';e={ (Get-CimInstance Win32_Service -Filter
    "ProcessId=$($_.ProcessId)").Name }}, CommandLine | Format-List
```

Die Kommandozeile verrät dir meist das Installationsverzeichnis, den Dienstnamen und manchmal die Systemeigenschaften wie -Djava.security.krb5.conf. Werte die Ereignisse grundsätzlich über die EventData-Felder aus, nicht über den Meldungstext, denn bei weitergeleiteten Ereignissen kann der Meldungstext leer sein. Mehr Rezepte dafür findest du in NTLM-Ereignisse mit PowerShell auswerten: Get-WinEvent-Rezepte.

SPNEGO mit Keytab und Dienstkonto: der saubere Weg

Für eine Java-Webanwendung ist SPNEGO das Gegenstück zu „Negotiate“ im IIS. Der Browser bekommt auf seine erste Anfrage eine 401-Antwort mit dem Kopf WWW-Authenticate: Negotiate, holt sich beim KDC ein Dienstticket für HTTP/<Servername> und schickt es im Authorization-Kopf zurück. Die Java-Anwendung entschlüsselt das Ticket mit dem Schlüssel aus einer Keytab und weiß damit, wer vor ihr steht. Kein Kennwort fliegt durchs Netz, kein Domain Controller muss für jede Anmeldung befragt werden. Wie das Gleiche auf IIS-Seite aussieht, steht in NTLM in IIS und Webanwendungen: Negotiate statt NTLM.



Skizze: SPNEGO-Ablauf über HTTP, vereinfacht; Beispielnamen der Musterwerk GmbH

Schritt 1: Dienstkonto und SPN

Die Anwendung braucht eine Identität in der Domäne, an der der SPN hängt. Läuft der Tomcat auf Linux, ist das ein eigenes Benutzerkonto als Dienstkonto, etwa svc-fachapp. Läuft er auf Windows, kann das auch ein vorhandenes Dienstkonto sein. Wichtig: Der SPN HTTP/<FQDN> darf in der gesamten Gesamtstruktur nur an genau einem Konto stehen. Spricht die Anwendung unter einem Aliasnamen, braucht auch der einen SPN, siehe CNAME, Aliasnamen und hosts-Datei: warum Kerberos den Namen nicht findet. Wie du Dienstkonten grundsätzlich aufräumst, steht in Dienstkonten als NTLM-Quelle: finden, prüfen, umstellen.

```
# SPN prüfen: Gibt es ihn schon, und an welchem Konto?
setspn -Q HTTP/fachapp.musterwerk.local

# SPN setzen (prüft vorher auf Duplikate)
setspn -S HTTP/fachapp.musterwerk.local MUSTERWERK\svc-fachapp

# AES für das Dienstkonto aktivieren (ActiveDirectory-Modul)
Set-ADUser svc-fachapp -KerberosEncryptionType AES128,AES256

# Gesamtstruktur auf doppelte SPNs durchsuchen
setspn -X -F
```

TIPP · Hostname, nicht Dienstname

Der Browser baut den SPN aus dem Namen, den er in der Adresszeile sieht. Steht dort fachapp.musterwerk.local, fragt er nach HTTP/fachapp.musterwerk.local, nicht nach dem Namen des Servers dahinter. Auch für https heißt die Dienstklasse HTTP. Und ein angehängter Port gehört bei den gängigen Browsern standardmäßig nicht in den SPN.

Schritt 2: Keytab mit ktpass erzeugen

Die Keytab enthält die Schlüssel des Dienstkontos, abgeleitet aus Kennwort, Principal-Name und Verschlüsselungstyp. Unter Windows erzeugst du sie mit ktpass. Microsoft empfiehlt, /crypto immer ausdrücklich anzugeben, weil die Standardwerte auf älteren MIT-Versionen beruhen, und

als Principal-Typ KRB5_NT_PRINCIPAL. Und Microsoft warnt ausdrücklich: Der Parameter /princ unterscheidet Groß- und Kleinschreibung und wird nicht geprüft.

```
ktpass /out C:\Sicher\fachapp.keytab ^
       /princ HTTP/fachapp.musterwerk.local@MUSTERWERK.LOCAL ^
       /mapuser MUSTERWERK\svc-fachapp ^
       /crypto AES256-SHA1 /ptype KRB5_NT_PRINCIPAL ^
       /pass *

# Auf dem Java-Server: Inhalt prüfen (JDK-Werkzeug bzw. MIT-klist)
klist -k -t -e /opt/fachapp/conf/fachapp.keytab
```

WARNUNG · ktpass setzt das Kennwort

Mit /pass setzt ktpass das Kennwort des Dienstkontos neu, und die Schlüsselversion (kvno) zählt hoch. Jede ältere Keytab für dieses Konto ist danach wertlos. Läuft die Anwendung auf zwei Knoten, brauchen beide dieselbe, neue Keytab. Und eine Keytab ist ein Kennwort in Dateiform: Dateirechte nur für das Dienstkonto des Tomcat, keine Kopie im Projektlaufwerk, kein Anhang im Ticketsystem.

Schritt 3: krb5.conf

Die Java-Laufzeit liest ihre Kerberos-Konfiguration aus einer krb5.conf, deren Pfad du mit -Djava.security.krb5.conf angibst. Ohne diese Angabe sucht Java an Standardorten, die je nach Betriebssystem und JDK verschieden sind. Gib den Pfad deshalb ausdrücklich an, dann weißt du, welche Datei gilt.

```
[libdefaults]
  default_realm = MUSTERWERK.LOCAL
  dns_lookup_kdc = true
  dns_lookup_realm = false
  forwardable = true
  default_tkt_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96
  default_tgs_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96

[realms]
  MUSTERWERK.LOCAL = {
    kdc = dc01.musterwerk.local
    kdc = dc02.musterwerk.local
  }

[domain_realm]
  .musterwerk.local = MUSTERWERK.LOCAL
  musterwerk.local = MUSTERWERK.LOCAL
```

Der Realm steht überall in Großbuchstaben, der DNS-Domänenname in [domain_realm] links klein. Fest eingetragene KDCs sind bequem, bis ein Domain Controller ersetzt wird. Mit dns_lookup_kdc = true findet Java die KDCs über die SRV-Einträge der Domäne. Die Enctypes-Zeilen sind optional; sie helfen, wenn alte Konfigurationen noch RC4 oder DES erzwingen wollen.

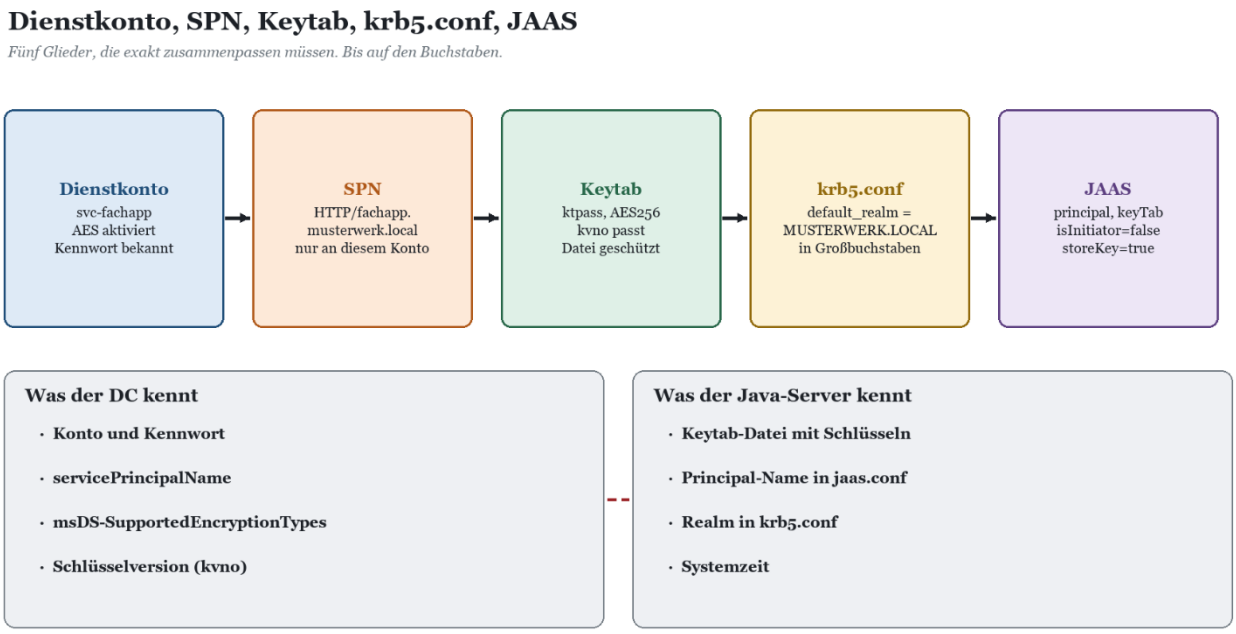
Schritt 4: JAAS und Tomcat

Tomcat bringt einen SPNEGO-Authenticator mit. In der web.xml der Anwendung setzt du die Anmeldemethode auf SPNEGO, in der jaas.conf beschreibst du, wie Tomcat an seine Schlüssel kommt. Der Eintrag für die Annahme heißt com.sun.security.jgss.krb5.accept. Die Pfade übergibst du dem Tomcat über -Djava.security.krb5.conf und -Djava.security.auth.login.config, zum Beispiel in setenv.sh oder in den Java-Optionen des Windows-Dienstes.

```
com.sun.security.jgss.krb5.accept {
  com.sun.security.auth.module.Krb5LoginModule required
    principal="HTTP/fachapp.musterwerk.local@MUSTERWERK.LOCAL"
```

```
    useKeyTab=true
    keyTab="/opt/fachapp/conf/fachapp.keytab"
    storeKey=true
    isInitiator=false
    doNotPrompt=true;
};

<!-- web.xml der Anwendung -->
<login-config>
  <auth-method>SPNEGO</auth-method>
</login-config>
```



Skizze: Bausteine einer SPNEGO-Konfiguration für eine Java-Webanwendung; links und rechts müssen übereinstimmen

Bei Fachanwendungen wirst du die web.xml selten selbst anfassen dürfen. Viele Hersteller haben eine eigene Einstellung „Kerberos/SPNEGO“ in ihrer Administrationsoberfläche, die intern genau diese Bausteine setzt. Das Prinzip bleibt gleich: Dienstkonto, SPN, Keytab, krb5.conf, JAAS. Kennst du die fünf Glieder, kannst du auch eine fremde Oberfläche lesen. Auf der Browserseite muss die Adresse außerdem als Intranet gelten, sonst schicken Edge, Chrome und Firefox gar kein Kerberos-Ticket. Wie du das per Gruppenrichtlinie einstellst, steht in Edge, Chrome und Firefox: Browser für Kerberos im Intranet einstellen.

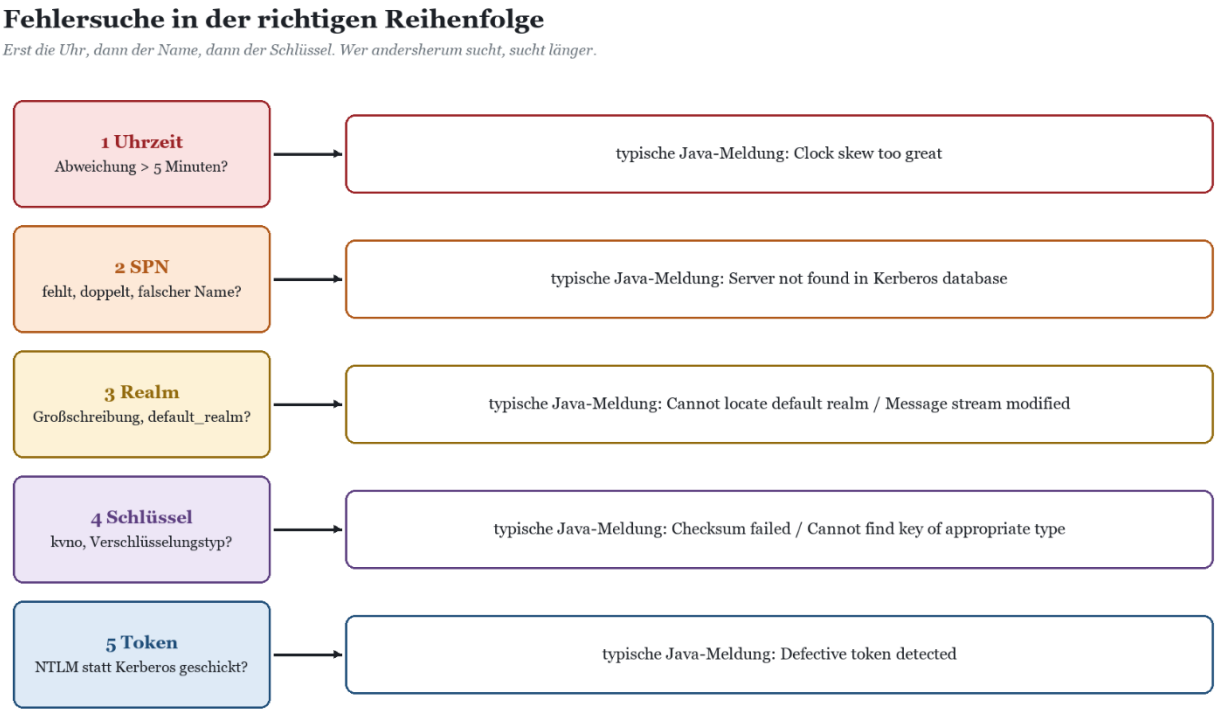
AUS DER PRAXIS · Musterwerk GmbH: das Laborsystem

Bei der Musterwerk GmbH meldet sich das Labor-Frontend über einen JCIFS-Filter an. Nach der Umstellung auf SPNEGO funktioniert alles, außer auf den drei Rechnern im Prüflabor. Dort steht die Adresse als http://10.20.4.17:8080 im Lesezeichen. IP-Adresse heißt: kein SPN, kein Ticket, Browser bietet NTLM an, Tomcat lehnt ab. Die Lösung bestand aus einer neuen Verknüpfung mit dem FQDN und einem verlegenen Lächeln.

Die üblichen Verdächtigen: Uhrzeit, SPN und Realm

Kerberos ist ein Protokoll mit wenig Humor. Was bei NTLM irgendwie durchgeht, quittiert Kerberos mit einer Fehlermeldung, deren Text du erst beim dritten Lesen verstehst. Die gute Nachricht: Fast alle SPNEGO-Probleme in Java-Umgebungen lassen sich auf eine Handvoll

Ursachen zurückführen. Prüfe sie in einer festen Reihenfolge, dann bist du schneller fertig als der Herstellersupport mit seiner Rückfrage nach der Versionsnummer.



Skizze: Prüfreihenfolge bei SPNEGO-Fehlern; Meldungstexte gekürzt, abhängig von JDK-Version

Java-Meldung (gekürzt)	Wahrscheinliche Ursache	Prüfschritt
Clock skew too great	Systemzeit weicht mehr als die erlaubte Toleranz ab	Zeitquelle prüfen, gegen DC synchronisieren
Server not found in Kerberos database	SPN fehlt oder Name passt nicht	setspn -Q HTTP/<FQDN>, Adresse im Browser prüfen
Cannot locate default realm	krb5.conf nicht gefunden oder ohne default_realm	Pfad in -Djava.security.krb5.conf prüfen
Message stream modified	häufig Realm-Schreibweise oder Principal-Name	Realm in Großbuchstaben, Principal exakt wie bei ktpass
Checksum failed / Specified version of key is not available	Keytab passt nicht zum aktuellen Kennwort (kvno)	Keytab neu erzeugen und verteilen
Cannot find key of appropriate type	Keytab enthält nicht den Verschlüsselungstyp des Tickets	AES am Konto und in der Keytab, klist -k -e
Defective token detected	Browser schickt NTLM-Token statt Kerberos	SPN, FQDN und Intranet-Zone prüfen

Die Texte unterscheiden sich je nach JDK-Version und Bibliothek leicht. Mehr Details siehst du, wenn du die JVM vorübergehend mit -Dsun.security.krb5.debug=true startest. Das Debug-Log ist ausführlich und landet meist in catalina.out oder der Dienstausgabe; schalte es nach der Fehlersuche wieder ab.

Uhrzeit: fünf Minuten bis zum Ausfall

Kerberos-Tickets enthalten Zeitstempel. Die Windows-Richtlinie „Maximale Toleranz für die Synchronisierung der Computeruhr“ steht standardmäßig auf fünf Minuten. Ein Linux-Tomcat, der seine Zeit aus einer anderen Quelle bezieht als die Domain Controller, oder eine virtuelle

Maschine nach einem Snapshot-Rücksprung driftet schneller aus dieser Spanne, als dir lieb ist. Stell die Zeitquelle des Java-Servers auf die Domain Controller oder auf dieselbe Quelle, die auch der PDC-Emulator nutzt. Auf Windows prüfst du mit w32tm /query /status, auf Linux mit timedatectl oder chronyc tracking.

SPN: fehlt, doppelt oder auf dem falschen Namen

Fehlt der SPN, bekommt der Browser kein Ticket und bietet NTLM an. Ist er doppelt, also an zwei Konten registriert, stellt der KDC ein Ticket aus, das mit dem Schlüssel des falschen Kontos verschlüsselt sein kann, und die Java-Anwendung kann es nicht lesen. Beides ist häufig, besonders nach einem Serverumzug, bei dem der alte SPN am Computerkonto stehen geblieben ist. Steht der SPN noch am Computerkonto des Servers, weil früher ein IIS dort lief, und gleichzeitig am neuen Dienstkonto: Glückwunsch, das ist der Klassiker. Hintergrund zum fehlenden SPN als NTLM-Ursache liefert Fehlender SPN: wenn Kerberos den Dienst nicht findet.

Realm: MUSTERWERK.LOCAL ist nicht musterwerk.local

Active Directory nimmt es bei Namen nicht so genau, Kerberos-Implementierungen außerhalb von Windows schon. Der Realm einer AD-Domäne ist der DNS-Domänenname in Großbuchstaben. Steht in der krb5.conf default_realm = musterwerk.local oder im Principal der jaas.conf HTTP/fachapp.musterwerk.local@musterwerk.local, passen Ticket und Erwartung nicht zusammen. Microsoft weist in der Dokumentation zu ktpass ausdrücklich darauf hin, dass Kerberos-Implementierungen, die Groß- und Kleinschreibung unterscheiden, bei nicht exakt passender Schreibweise Probleme haben können, bis hin zum Scheitern der Vorauthentifizierung.

WICHTIG · Verschlüsselung: RC4 ist kein Rettungsanker

Neuere JDK-Versionen lehnen schwache Verfahren wie DES und RC4 standardmäßig ab, und auch in Active Directory ist RC4 auf dem Rückzug. Wenn eine alte Anleitung allow_weak_crypto = true empfiehlt, ist das kein Fix, sondern eine Verschiebung des nächsten Ausfalls. Aktiviere AES am Dienstkonto, erzeuge die Keytab mit AES256-SHA1 und prüfe mit klist -k -e, was wirklich drinsteht.

Prüfen, ob Kerberos wirklich greift

Auf dem Windows-Client zeigt klist nach dem Aufruf der Anwendung ein Ticket für HTTP/fachapp.musterwerk.local. Fehlt es, hat der Browser es gar nicht erst versucht. Im Netzwerkmitschnitt oder in den Entwicklerwerkzeugen des Browsers erkennst du am Authorization-Kopf, was tatsächlich geschickt wurde: Beginnt der Wert nach Negotiate mit YII, ist es Kerberos in einer SPNEGO-Hülle; beginnt er mit TlRMTVNTUA, ist es NTLM. Ausführlich zum Werkzeug: Mit klist prüfen, ob Kerberos wirklich greift. Auf Windows-Seite gilt außerdem: Für dieses Ziel sollten keine neuen Ereignisse 4020/4021 mehr auftauchen.

Wenn der Hersteller ran muss

Nicht jede Java-Anwendung lässt sich per Konfiguration umstellen. Ist der NTLM-Filter fest in die Anwendung eingebaut, spricht der HTTP-Client der Schnittstelle nur NTLM oder verlangt der Hersteller für Kerberos eine neue Hauptversion, endet deine Zuständigkeit an der Stelle, an der die des Herstellers beginnt. Dann brauchst du eine Herstelleranfrage, und zwar eine, die nicht nach drei Wochen mit „Kerberos wird unterstützt, siehe Handbuch Kapitel 7“ zurückkommt.

Was in eine gute Herstelleranfrage gehört

Frage an den Hersteller	Warum du sie stellst
Unterstützt die eingesetzte Version Kerberos über SPNEGO, ohne NTLM-Rückfall?	„Windows-Anmeldung“ heißt nicht Kerberos. Du brauchst eine eindeutige Aussage.
Welche Bibliothek übernimmt die Anmeldung	JCIFS-NTLM ist ein anderes Gespräch als Waffle oder

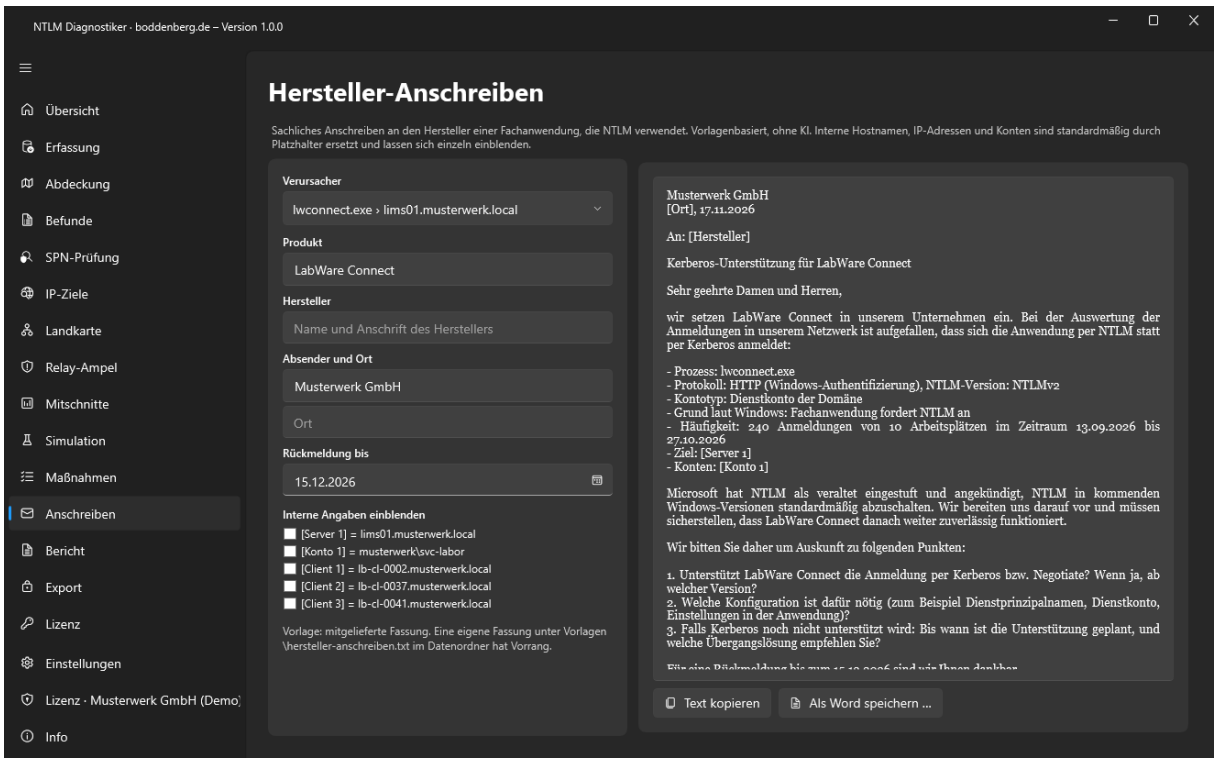
(Name, Version)?	ein eigener Handler.
Welche Version bzw. welches Update ist dafür nötig, und ab wann ist es verfügbar?	Du planst gegen einen festen Stichtag, nicht gegen „demnächst“.
Welcher SPN, welches Dienstkonto und welche Keytab-Anforderungen gelten?	Dienstklasse, Hostname, Verschlüsselungstypen, Pfade für krb5.conf und Keytab.
Spricht die Anwendung ausgehend weitere Systeme per NTLM an (Datenbank, Freigaben, Schnittstellen)?	Die Webanmeldung ist oft nur die Hälfte.
Wird NTLMv1 an irgendeiner Stelle verwendet?	NTLMv1-SSO wird bereits blockiert; das hat Vorrang.

Was die Anfrage überzeugend macht, ist der Beleg: welcher Prozess, welches Ziel, welcher Dienst, wie oft, mit welcher NTLM-Version und mit welchem Begründungstext. Mit Bordmitteln baust du dir das aus Get-WinEvent, Gruppierung und einer Excel-Tabelle zusammen. Für einen Server geht das. Für zwanzig Java-Anwendungen auf vierzig Servern, deren Ereignisse teils auf den Clients, teils auf dem DC und teils gar nicht entstehen, wird es zur Nebenbeschäftigung. Dazu kommt, dass du beim Kopieren aus dem Ereignisprotokoll gern mehr Interna an den Hersteller schickst, als dir lieb ist: Kontonamen, interne Hostnamen, IP-Adressen.

Genau an dieser Stelle hilft der NTLM Diagnostiker. Er sammelt die NTLM-Ereignisse vom Domain Controller, über einen Windows Event Collector, per Direktabfrage oder als Datei-Import, fasst gleichartige Verbindungen nach Ursache, Prozess und Ziel zusammen und macht aus einem Verursacher wie einem Java-Dienst mit wenigen Klicks ein Anschreiben an den Hersteller, inklusive Beleg. Er arbeitet dabei ausschließlich lesend: Er ändert nichts an deiner Domäne, nichts an deinen Servern und nichts an der Fachanwendung.

SO GEHT'S MIT DEM NTLM DIAGNOSTIKER · Hersteller-Anschreiben mit Beleg

1. Wähl in der Liste der Verursacher den Prozess aus, um den es geht, etwa den Java-Dienst deiner Fachanwendung. Der Diagnostiker übernimmt Verursacher, Ziel und Häufigkeit aus den erfassten Ereignissen.
2. Trag Produkt, Hersteller, Absender, Ort und eine Rückmeldefrist ein.
3. Lege fest, welche internen Angaben im Schreiben erscheinen dürfen. Interne Details sind standardmäßig ausgeblendet; du blendest nur ein, was der Hersteller wirklich braucht.
4. Prüf die Textvorschau. Das Schreiben fragt nach Kerberos-Unterstützung und den Voraussetzungen dafür und legt den Beleg bei: Prozess, Ziel, Häufigkeit.
5. Kopier den Text in deine Mail oder speichere ihn als Word-Datei für den Postweg oder das Ticketsystem des Herstellers.



So erstellst du ein Anschreiben an den Hersteller deiner Fachanwendung, mit Beleg zu Prozess, Ziel und Häufigkeit (Demodaten)

Für die interne Abstimmung, etwa mit der Fachabteilung oder dem Einkauf, hilft der Word-Bericht. Er ordnet Fachanwendungen als eigene Kategorie ein und nennt Verursacher, Ursache und Anzahl der Anmeldungen. Damit ist schnell geklärt, warum gerade diese Anwendung nicht bis „irgendwann“ warten kann.

Kategorie	Verursacher	Anmeldungen	davon NTLMv1 (Verursacher)
Fachanwendung	4	1.338	1
Sonstige	3	906	0
Keine Sichtverbindung zum DC	1	680	0
SQL Server	2	600	0
Inventarscanner	2	580	1
Multifunktionsgerät	2	400	2
NAS	1	180	0
Zielname fehlt	1	150	0
Anonyme Anmeldung	1	120	0
Cloud-Konto	1	60	0

4 Analysebefunde

Gleichartige Verbindungen sind zu Verursachern zusammengefasst (Ursache, Prozess, Ziel). Kritisch: NTLMv1 oder bereits blockierte Anmeldungen; mittel: hohes Risiko beim Abschalten oder fehlende Integritätsprüfung; niedrig: alles Übrige.

4.1 Befunde kritisch

Verursacher	Ursache	Kategorie	Quellen	Anmeldungen
scanagent.exe › scan01.musterwerk.local	Inventarscanner	Inventarscanner	12	480
lager.exe › lager-app01.musterwerk.local	Fachanwendung fordert NTLM an	Fachanwendung	6	408
scan-eg-01 › fs01.musterwerk.local	Gerät ohne Windows (Drucker, Scanner)	Multifunktionsgerät	1	260
mfp-og2 › fs01.musterwerk.local	Multifunktionsgerät (Scan-to-Folder)	Multifunktionsgerät	1	140

Inventarscanner (scanagent.exe › scan01.musterwerk.local)

Inventarisierungs- und Netzwerkscanner melden sich mit einem Sammelkonto an vielen Rechnern an, häufig per IP-Adresse oder mit einem lokalen Konto. Kerberos scheitert dann, Windows fällt auf NTLM zurück.

Mehr dazu: https://www.boddenberg.de/ntlm-dienstkonten/

Fachanwendung fordert NTLM an (lager.exe › lager-app01.musterwerk.local)

Die Anwendung ruft NTLM ausdrücklich auf statt Negotiate. Das lässt sich nur im Produkt selbst ändern.

Mehr dazu: https://www.boddenberg.de/ntlm-anwendung-hersteller/

Gerät ohne Windows (Drucker, Scanner) (scan-eg-01 › fs01.musterwerk.local)

Ein Netzwerkgerät ohne Windows-Ereignisprotokoll meldet sich per NTLM an. Sichtbar wird das nur im Mitschnitt oder auf der Serverseite.

Mehr dazu: https://www.boddenberg.de/ntlm-nas-drucker-scanner/

Multifunktionsgerät (Scan-to-Folder) (mfp-og2 › fs01.musterwerk.local)

Multifunktionsgeräte legen Scans per SMB in einer Freigabe ab und können oft nur NTLM, ältere Geräte sogar nur NTLMv1.

Mehr dazu: https://www.boddenberg.de/ntlm-nas-drucker-scanner/

So liest du im Bericht die kritischen Befunde, darunter Fachanwendungen mit Ursache und Anzahl der Anmeldungen (Demodaten)

Antwortet der Hersteller mit einem Termin, trägst du die Maßnahme im Maßnahmen-Board ein und prüfst nach dem Update über die Erfolgskontrolle, ob die NTLM-Anmeldungen dieses

Verursachers wirklich verschwinden. Antwortet er nicht, hast du immerhin einen datierten Beleg dafür, dass du gefragt hast. Das ist bei einer begründeten Ausnahme in den Restrict-NTLM-Richtlinien Gold wert, siehe Restrict-NTLM-Richtlinien erklärt: Überwachen, Ausnahmen, Verweigern. Die kostenlose Express-Edition liefert dir den Fünf-Minuten-Befund mit Readiness-Ampel und Top-10-Verursachern direkt vom DC; das Hersteller-Anschreiben kannst du sieben Tage lang je Domäne im Testzeitraum ausprobieren. Wenn du mehrere Fachverfahren und Hersteller parallel steuern musst, unterstützt dich unsere Beratung zur NTLM-Ablösung. Wie das in Verwaltungen mit Rechenzentrumsbetrieb aussieht, beschreibt NTLM-Ablösung in der Kommunalverwaltung: Fachverfahren und Rechenzentren.

FAQ zu Java, NTLM und Kerberos

Kann Java überhaupt Kerberos?

Ja. Die Java-Laufzeit enthält eine eigene Kerberos-Implementierung mit GSS-API, dem Krb5LoginModule und SPNEGO. NTLM kommt in Java-Anwendungen meist aus Zusatzbibliotheken.

Fällt der Tomcat-SPNEGO-Authenticator auf NTLM zurück, wenn Kerberos scheitert?

Nein. Er verarbeitet Kerberos über SPNEGO. Schickt der Browser ein NTLM-Token, scheitert die Anmeldung, typischerweise mit einer Meldung wie „Defective token detected“ im Log.

Brauche ich für SPNEGO ein Benutzerkonto oder reicht das Computerkonto?

Läuft die Anwendung auf Linux, nimmst du in der Regel ein eigenes Dienstkonto mit SPN und Keytab. Wichtig ist nur, dass der SPN HTTP/<FQDN> an genau einem Konto steht und die Keytab zu dessen aktuellem Kennwort passt.

Warum klappt die Anmeldung per Name, aber nicht per IP-Adresse?

Weil der Browser den SPN aus dem Namen in der Adresszeile bildet. Für eine IP-Adresse gibt es standardmäßig keinen SPN, also kein Ticket, also bietet der Browser NTLM an.

Muss der Realm wirklich großgeschrieben sein?

Ja. Der Realm einer AD-Domäne ist der DNS-Name in Großbuchstaben. Java unterscheidet Groß- und Kleinschreibung, und ein abweichend geschriebener Realm oder Principal führt zu Fehlern.

Wie viel Uhrabweichung verträgt Kerberos?

Standardmäßig fünf Minuten, festgelegt in der Kerberos-Richtlinie der Domäne. Synchronisiere Java-Server deshalb gegen die Domain Controller.

Der JDBC-Treiber für SQL Server nutzt NTLM. Was stelle ich um?

Setz integratedSecurity=true und authenticationScheme=JavaKerberos und sorge für eine passende krb5.conf und einen gültigen SPN für MSSQLSvc. Ob Kerberos greift, zeigt dir auf dem SQL Server die Spalte auth_scheme in sys.dm_exec_connections.

Was tue ich, wenn der Hersteller kein Kerberos liefert?

Belegen, nachfragen, Frist setzen und bis dahin eine eng gefasste, dokumentierte Ausnahme mit Ablaufdatum einrichten. Und die Frage nach der Zukunft dieser Anwendung offen stellen.

Fazit: Java kann Kerberos, man muss es nur lassen

Java-Fachanwendungen sind bei der NTLM-Ablösung selten technisch hoffnungslos. Die Laufzeit bringt Kerberos mit, Tomcat bringt SPNEGO mit, und der Weg über Dienstkonto, SPN, Keytab, krb5.conf und JAAS ist gut beschritten. Was scheitert, sind die Details: eine Uhr, die driftet, ein SPN, der an zwei Konten hängt, ein Realm in Kleinbuchstaben, eine Keytab von vor dem letzten

Kennwortwechsel. Prüf sie in dieser Reihenfolge, und die meisten Probleme sind nach einer Stunde erledigt statt nach einer Woche.

Bleibt der harte Kern: Anwendungen, die NTLM fest eingebaut haben. Dort hilft keine krb5.conf, sondern nur eine Herstelleranfrage mit Beleg und Frist. Je früher sie rausgeht, desto besser, denn der Hersteller hat noch ein paar hundert andere Kunden mit derselben Frage. Wie du die Umstellung in Etappen planst, steht in NTLM schrittweise abschalten, ohne dass etwas ausfällt, und wie du danach erkennst, ob jemand heimlich zurückfällt, in Nach der Abschaltung: Rückfälle erkennen und NTLM-frei bleiben. Wenn dein Team Kerberos-Fehler und NTLM-Ereignisse künftig selbst sicher lesen will, schau in die Schulung zum NTLM Diagnostiker.

WEITERLESEN · Weiter in dieser Serie

› NTLM-Nutzung ermitteln: der komplette Leitfaden
› NTLM-Reason-IDs erklärt: warum Windows nicht Kerberos benutzt hat
› Fehlender SPN: wenn Kerberos den Dienst nicht findet
› NTLM in IIS und Webanwendungen: Negotiate statt NTLM
› Dienstkonten als NTLM-Quelle: finden, prüfen, umstellen
› SQL Server und NTLM: Windows-Anmeldung auf Kerberos umstellen
› Edge, Chrome und Firefox: Browser für Kerberos im Intranet einstellen
› CNAME, Aliasnamen und hosts-Datei: warum Kerberos den Namen nicht findet
› Mit klist prüfen, ob Kerberos wirklich greift
› Linux und Samba in der Domäne: Kerberos statt NTLM
› Microsoft Kerberos