IIS-Eigenentwicklungen und http.sys-Dienste veröffentlichen



WISSEN

Alles zum Kemp LoadMaster mit Microsoft-Brille: Exchange, SharePoint, ADFS, RDS, Eigenentwicklungen und Kerberos an einem Ort.

› **Kemp LoadMaster**

BERATUNG

Hausgemachte Webanwendung oder Hersteller-API hinter den LoadMaster bringen, ohne dass Kerberos heimlich zu NTLM wird? Wir schauen gemeinsam drauf.

› **Beratung zum Kemp LoadMaster**

SCHULUNG

Virtual Services, Health Checks und Kerberos hinter einer VIP im Labor durchspielen, bevor die Buchhaltung es am Monatsende für dich tut.

› **Kemp-LoadMaster-Schulung**

Exchange, SharePoint und ADFS haben einen unschätzbaren Vorteil: Irgendjemand hat schon vor dir darüber nachgedacht, wie sie hinter einem Load Balancer funktionieren. Es gibt Templates, Deployment-Guides und Foren voller Menschen, die denselben Fehler schon 2019 gemacht haben. Deine Eigenentwicklung hat nichts davon. Sie hat einen Entwickler, der inzwischen in einer anderen Abteilung arbeitet, eine web.config mit auskommentierten Abschnitten aus drei Jahrzehnten und eine Dokumentation, die aus dem Satz „läuft auf dem IIS“ besteht. Und die ERP-Schnittstelle des Herstellers? Die ist eine Blackbox mit Port 8443, einem Windows-Dienst und einem Supportvertrag, der genau dann endet, wenn du eine Frage hast.

Trotzdem landen genau diese Anwendungen früher oder später hinter dem LoadMaster. Meistens dann, wenn der eine Server, auf dem alles läuft, zum dritten Mal am Patchdienstag nicht wieder hochgekommen ist. Der zweite Server ist schnell installiert, die VIP ist schnell angelegt, und der erste Test im Browser klappt sogar. Die Probleme kommen danach: Anmeldefenster, die vorher nie da waren, Benutzer, die nach dem Mittagessen plötzlich abgemeldet sind, und ein Health Check, der grün leuchtet, während die Anwendung seit einer Stunde nur noch Fehlerseiten ausliefert. Der LoadMaster ist an keinem dieser Probleme schuld. Er macht sie nur sichtbar, und zwar zuverlässig.

Dieser Beitrag gehört zur Serie rund um den Kemp LoadMaster und kümmert sich um alles, was nicht aus der Microsoft-Produktpalette stammt, aber auf Microsoft-Technik läuft: IIS-Anwendungen aus dem eigenen Haus, Hersteller-APIs als Windows-Dienst auf http.sys, ASP.NET-Core-Dienste mit Kestrel. Es geht um vier Fragen, die du vor dem Go-live beantworten solltest: Wer entschlüsselt das Kerberos-Ticket, wenn drei Server unter einem Namen antworten? Unter welchem Konto läuft der Dienst, und warum ist „LocalSystem, weil es ging“ keine Antwort? Woran

erkennt der LoadMaster, dass die Anwendung wirklich lebt? Und wo merkt sich die Anwendung, wer du bist? Am Ende steht ein anonymisierter Praxisfall: eine ERP-API mit Kerberos, die vorher nur an guten Tagen funktioniert hat.

FAKTEN · Das Wichtigste in sechs Sätzen

Ein Kerberos-Ticket ist mit dem Schlüssel des Kontos verschlüsselt, das den SPN zum angefragten Namen trägt; jeder Server hinter der VIP muss genau diesen Schlüssel haben.

Dienste unter LocalSystem, NetworkService oder ApplicationPoolIdentity treten im Netz als Computerkonto auf, und jedes Computerkonto hat einen eigenen Schlüssel. Für eine Farm unter einem gemeinsamen Namen taugt das nicht.

IIS entschlüsselt Tickets standardmäßig im Kernelmodus mit dem Computerkonto; in Farmen mit Dienstkonto brauchst du useAppPoolCredentials="true" in der Windows-Authentifizierung.

Selbst gehostete http.sys-Dienste brauchen eine URL-Reservierung und eine Zertifikatsbindung per netsh; welches Konto dort Tickets entschlüsselt, legt die jeweilige Anwendung fest.

Ein Health-Endpunkt sollte anonym erreichbar sein, wenige Sekunden brauchen, die Pflicht-Abhängigkeiten prüfen und einen Drain-Schalter für die Wartung kennen.

Session-State im Arbeitsspeicher eines Servers ist mit Load Balancing unvereinbar; Persistenz am LoadMaster lindert das, heilt es aber nicht.

Was hinter der VIP eigentlich läuft: IIS, http.sys und Kestrel

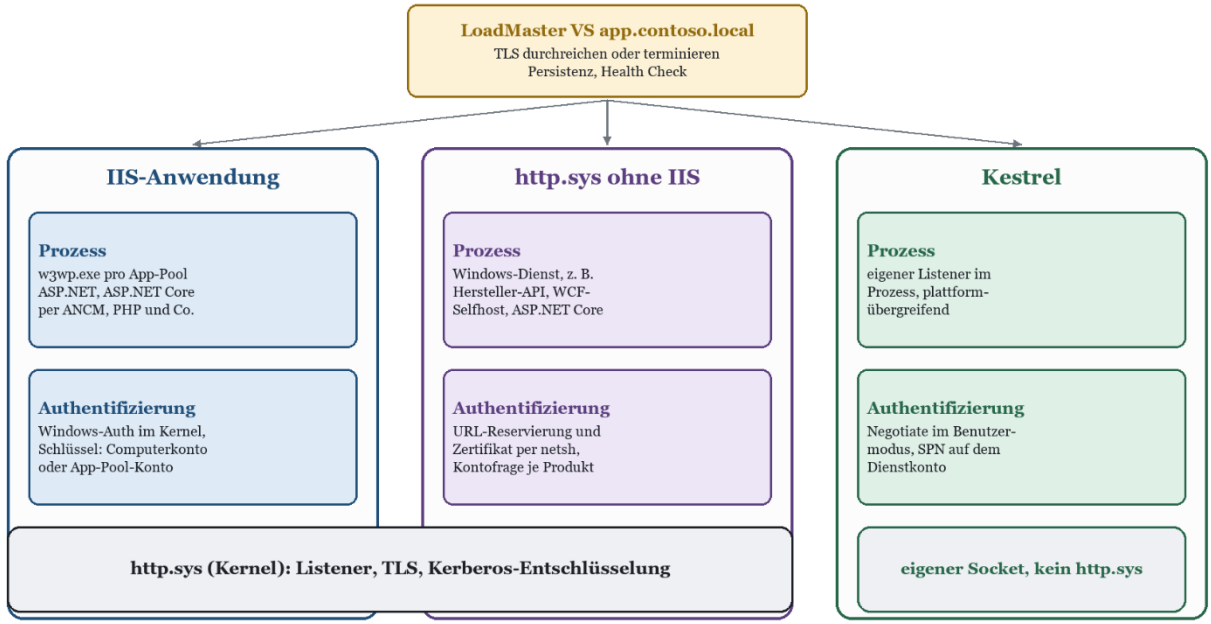
Bevor du am LoadMaster auch nur einen Virtual Service anlegst, lohnt sich eine unscheinbare Frage an die Anwendung: Wer nimmt eigentlich die HTTP-Anfrage entgegen? Die Antwort ist nicht immer „der IIS“, auch wenn sie in Besprechungen gern so gegeben wird. Unter Windows gibt es drei verbreitete Modelle, und sie unterscheiden sich genau in den Punkten, die hinter einem Load Balancer wehtun: Wo die Authentifizierung passiert, mit welchem Schlüssel Kerberos-Tickets entschlüsselt werden und wo das Zertifikat hängt.

Drei Hosting-Modelle mit eigenen Regeln

Das erste Modell ist die klassische IIS-Anwendung. Ein Arbeitsprozess pro Anwendungspool, Bindungen im IIS-Manager, Authentifizierung über die bekannten Schalter. Darunter liegt http.sys, der HTTP-Treiber im Windows-Kernel, der Verbindungen annimmt, TLS erledigt und Anfragen an den richtigen Prozess weiterreicht. Das zweite Modell nutzt http.sys direkt, ganz ohne IIS. Das machen mehr Produkte, als man denkt: SQL Server Reporting Services, ADFS, WinRM, viele WCF-Dienste im Selfhosting und eine erstaunliche Zahl von Hersteller-Schnittstellen, die als Windows-Dienst daherkommen. Dazu kommt ASP.NET Core mit dem HTTP.sys-Server. Das dritte Modell ist Kestrel, der plattformübergreifende Webserver von ASP.NET Core, der seinen eigenen Socket öffnet und mit http.sys nichts zu tun hat, sofern er nicht ohnehin hinter dem IIS läuft.

Drei Hosting-Modelle, ein gemeinsamer Unterbau

IIS und selbst gehostete Dienste teilen sich http.sys; Kestrel geht eigene Wege



Merke: Wer im Kernel entschlüsselt, braucht den Schlüssel des Kontos, das den SPN trägt.

Skizze: vereinfacht; Namen fiktiv, Beispieldomäne contoso.local

Merkmal	IIS-Anwendung	http.sys ohne IIS	Kestrel eigenständig
Listener	http.sys, verwaltet durch IIS	http.sys, verwaltet durch den Dienst	eigener Socket im Prozess
Zertifikat	IIS-Bindung, intern ebenfalls http.sys	netsh http add sslcert	Konfiguration der Anwendung
Windows-Auth	Kernelmodus (Standard) oder Benutzermodus	Kernelmodus über http.sys	Negotiate-Handler im Benutzermodus
Kerberos-Schlüssel	Computerkonto oder App-Pool-Konto	je Produkt, laut Microsoft bei ASP.NET Core: Computerkonto	Dienstkonto des Prozesses
URL-Rechte	durch IIS erledigt	netsh http add urlacl, sonst Zugriff verweigert	Portfreigabe genügt
Typische Fehlerquelle	Kernelmodus und Dienstkonto beißen sich	Bindung, Reservierung, Konto unklar	SPN auf falschem Konto

Hersteller-APIs: die Blackbox mit Port 8443

Bei Eigenentwicklungen kannst du den Entwickler wenigstens noch in der Kantine abfangen. Bei Hersteller-APIs bist du auf das angewiesen, was in der Installationsanleitung steht, und das ist erfahrungsgemäß wenig. Typisch ist ein Windows-Dienst, der sich unter LocalSystem installiert, auf einem krummen Port lauscht und in einer Konfigurationsdatei einen Schalter für Windows-Authentifizierung hat. Ob das Produkt für mehrere Knoten gedacht ist, steht meistens nirgends. Das musst du herausfinden, bevor der LoadMaster davor steht, nicht danach.

AUS DER PRAXIS · Fragen, die du dem Hersteller vor dem Projekt stellst

Ist der Dienst für den Betrieb auf mehreren Knoten hinter einem Load Balancer freigegeben, und hält er Zustand im Speicher, etwa Sitzungen, Jobs, Caches oder Sperren?

Unter welchem Konto darf der Dienst laufen, und werden gruppenverwaltete Dienstkonten unterstützt?

Mit welchem Konto werden Kerberos-Tickets entschlüsselt, und wo muss der SPN liegen?

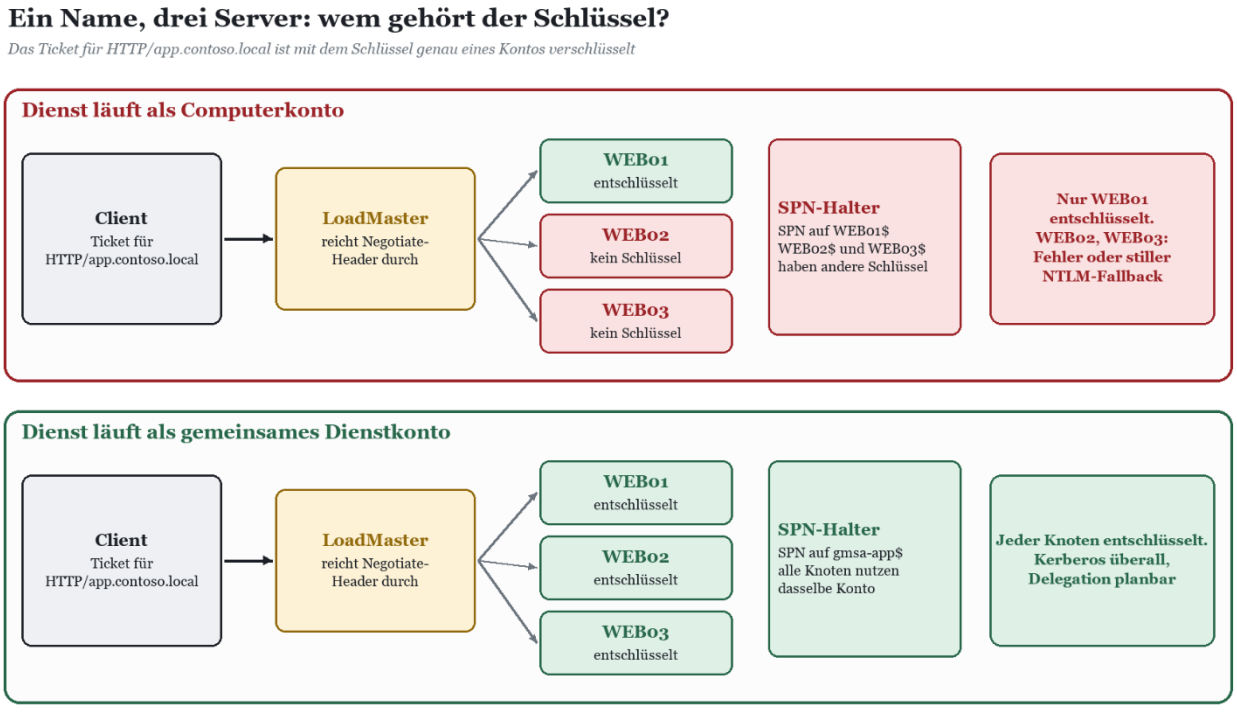
Gibt es einen anonymen Endpunkt, der den Zustand des Dienstes inklusive Datenbank meldet? Wenn nein: Wann gibt es einen?

Kommt der Dienst mit TLS-Terminierung am Load Balancer zurecht, und wertet er X-Forwarded-For aus?

Windows-Authentifizierung hinter einer VIP: wer entschlüsselt das Ticket?

Kerberos funktioniert im Kern wie ein versiegelter Brief. Der Client bittet den Domänencontroller um ein Ticket für einen Dienst, zum Beispiel HTTP/app.contoso.local. Der Domänencontroller sucht im Active Directory das Konto, das diesen Service Principal Name trägt, und verschlüsselt das Ticket mit dessen Schlüssel. Öffnen kann den Brief nur, wer diesen Schlüssel kennt. Solange genau ein Server hinter dem Namen steht, fällt das niemandem auf, denn der Server läuft als er selbst, und der SPN liegt auf seinem Computerkonto. Mit dem LoadMaster ändert sich das Bild schlagartig: Jetzt antworten drei Server unter einem Namen, und alle drei müssen denselben Brief öffnen können.

Der LoadMaster selbst ist dabei meist nur Briefträger. Solange er keine eigene Vorauthentifizierung macht, reicht er den Authorization-Header mit dem Negotiate-Token einfach durch, egal ob er TLS terminiert oder nicht. Was er nicht kann: einen Server, der den Schlüssel nicht hat, nachträglich zum Mitwisser machen. Die Grundlagen zu Tickets, Schlüsseln und Verschlüsselungstypen findest du auf der Seite Microsoft Kerberos; die typischen Namensprobleme behandelt der Beitrag SPN-Probleme hinter dem Load Balancer: ein Name, mehrere Server.



Skizze: Prinzip; IIS mit useAppPoolCredentials oder Dienst unter gMSA

Computerkonto oder Dienstkonto: die eigentliche Architekturentscheidung

Unter Windows treten mehrere Identitäten im Netz als Computerkonto auf, auch wenn sie lokal ganz unterschiedlich heißen. LocalSystem, NetworkService und die virtuelle ApplicationPoolIdentity des IIS melden sich am Domänencontroller allesamt als SERVER$ an. Das ist bequem, weil sich niemand um Kennwörter kümmern muss, und es ist für einen einzelnen Server völlig in Ordnung. Für eine Farm ist es eine Sackgasse, denn WEB01$, WEB02$ und WEB03$ haben drei verschiedene Schlüssel, und ein SPN darf nur an einem einzigen Konto hängen. Hängt er an mehreren, findet der Domänencontroller ein Duplikat und stellt überhaupt kein Ticket mehr aus.

Identität	Auftritt im Netz	SPN für den VIP-Namen	Eignung hinter dem LoadMaster
LocalSystem	Computerkonto SERVER$	nur auf einem Knoten möglich	nur Einzelserver, für Farm ungeeignet
NetworkService	Computerkonto SERVER$	nur auf einem Knoten möglich	nur Einzelserver, für Farm ungeeignet
ApplicationPoolIdentity	Computerkonto SERVER$	nur auf einem Knoten möglich	nur Einzelserver oder ohne Kerberos
Domänenbenutzer als Dienstkonto	eigenes Konto, auf allen Knoten gleich	einmal am Dienstkonto	geeignet, Kennwortpflege per Hand
Gruppenverwaltetes Dienstkonto (gMSA)	eigenes Konto, Kennwort vom AD verwaltet	einmal am gMSA	erste Wahl, wenn die Anwendung es unterstützt

Die Faustregel ist deshalb schlicht: Sobald mehr als ein Server unter einem Namen Kerberos sprechen soll, läuft die Anwendung unter einem gemeinsamen Dienstkonto, und der SPN des VIP-Namens liegt genau dort. Gruppenverwaltete Dienstkonten sind dafür ideal: Das Kennwort rotiert automatisch, niemand kennt es, und AES ist von Haus aus aktiviert. Das erspart dir nebenbei die Sorte Ärger, die der Beitrag Die RC4-Falle: KCD-Keytab trifft AES-only-Domäne beschreibt. Du legst fest, welche Server das Kennwort abrufen dürfen, installierst das Konto auf jedem Knoten und trägst es im Dienst oder im Anwendungspool mit einem Dollarzeichen am Ende und leerem Kennwort ein.

TIPP · SPN-Hygiene in drei Befehlen

setspn -Q HTTP/app.contoso.local zeigt, an welchem Konto der Name gerade hängt. setspn -S HTTP/app.contoso.local CONTOSO\gmsa-app$ setzt ihn und prüft dabei vorher auf Duplikate. setspn -X listet alle doppelten SPNs der Domäne. Den Port lässt du in aller Regel weg: Browser hängen ihn standardmäßig nicht an, und auch .NET verzichtet seit Version 6 beim HttpClient darauf.

IIS: Kernelmodus und useAppPoolCredentials

Jetzt kommt die Stelle, an der IIS-Farmen reihenweise scheitern, obwohl Konto und SPN stimmen. Der IIS entschlüsselt Kerberos-Tickets seit Version 7 standardmäßig im Kernelmodus, also in http.sys, und zwar mit dem Computerkonto. Der Anwendungspool kann noch so schön unter gmsa-app$ laufen: Das Ticket für HTTP/app.contoso.local ist mit dem Schlüssel des gMSA verschlüsselt, http.sys versucht es mit dem Schlüssel von WEB02$, und das geht schief. Der Browser bekommt einen 401, versucht es noch einmal, und wenn NTLM erlaubt ist, klappt es dann eben damit. Niemand merkt etwas, bis eine Delegation ins Spiel kommt oder NTLM abgeschaltet wird.

Microsoft beschreibt genau dieses Farmszenario und empfiehlt, den Kernelmodus eingeschaltet zu lassen und zusätzlich in der Windows-Authentifizierung das Attribut useAppPoolCredentials auf true zu setzen. Dann entschlüsselt http.sys das Ticket mit der Identität des Anwendungspools. Den Kernelmodus komplett abzuschalten funktioniert auch, ist aber nicht die empfohlene Variante. Du findest den Schalter im Konfigurations-Editor des IIS-Managers unter system.webServer/security/authentication/windowsAuthentication, oder du setzt ihn per appcmd beziehungsweise PowerShell, idealerweise identisch auf allen Knoten.

WARNUNG · Ein Name, zwei Anwendungen, zwei Konten: geht nicht

Der SPN enthält keinen Port und keinen Pfad. Laufen unter app.contoso.local zwei Anwendungen mit unterschiedlichen Konten, etwa eine Webanwendung auf 443 und eine API auf 8443, kann nur eine von beiden Kerberos. Die andere fällt auf NTLM zurück oder gar nicht. Die saubere Lösung ist ein eigener DNS-Name pro Dienstkonto, am LoadMaster gern auf derselben VIP und per Host-Header getrennt.

http.sys ohne IIS: Reservierung, Bindung und die Kontofrage

Selbst gehostete http.sys-Dienste nehmen dir die Arbeit ab, die der IIS sonst im Hintergrund erledigt, nämlich überhaupt nicht. Ein Prozess, der nicht als Administrator oder LocalSystem läuft, darf eine URL erst belegen, wenn sie für sein Konto reserviert ist. Das erledigst du mit netsh http add urlacl, einem URL-Präfix aus Protokoll https, dem Platzhalter + für alle Hostnamen, Port 8443 und Pfad /erp/ sowie dem Parameter user=CONTOSO\gmsa-erpapi$. Das Zertifikat bindest du mit netsh http add sslcert, entweder an IP und Port oder, bei mehreren Namen auf einem Port, per hostnameport an den Namen mit SNI. Mit netsh http show urlacl, netsh http show sslcert und netsh http show servicestate siehst du, was tatsächlich konfiguriert ist und welcher Prozess welche URL hält.

Die Kontofrage ist bei http.sys-Diensten heikler als beim IIS, weil es keinen einheitlichen Schalter wie useAppPoolCredentials gibt. Welches Konto die Tickets entschlüsselt, legt die Anwendung fest, wenn sie die Authentifizierung bei http.sys einrichtet. ADFS etwa arbeitet seit jeher mit dem SPN am Dienstkonto und läuft problemlos als Farm. Für ASP.NET Core mit dem HTTP.sys-Server sagt Microsoft dagegen ausdrücklich, dass der SPN am Host und nicht am Benutzer der Anwendung zu registrieren ist; der Benutzermodus mit Kerberos wird dort nicht unterstützt. Wer so eine Anwendung als Farm unter einem Namen mit Kerberos betreiben will, landet folglich beim IIS mit useAppPoolCredentials oder bei Kestrel mit dem Negotiate-Handler, bei dem der SPN am Dienstkonto liegt. Bei Hersteller-Diensten hilft nur Fragen und Testen.

FAKTEN · So testest du, wer entschlüsselt

Rufe die Anwendung von einem Client über den VIP-Namen auf und prüfe mit klist, ob ein Ticket für HTTP/app.contoso.local vorhanden ist. Im Sicherheitsprotokoll des Servers zeigt das Anmeldeereignis 4624 im Feld zum Authentifizierungspaket, ob Kerberos oder NTLM gewonnen hat. Schalte zum Test einen Real Server nach dem anderen am LoadMaster ab, bis jeder Knoten einmal allein bedient hat. Kerberos auf einem Knoten beweist gar nichts.

Stolperfallen: CNAME, Extended Protection und TLS-Offload

Drei Klassiker sorgen dafür, dass Kerberos trotz korrektem SPN nicht zustande kommt. Erstens der CNAME: Windows-Clients und Chromium-Browser bilden den SPN standardmäßig aus dem kanonischen Namen, den der DNS nach Auflösung des Alias liefert. Zeigt app.contoso.local per CNAME auf lm-vip01.contoso.local, fragt der Client nach einem Ticket für den VIP-Hostnamen,

nicht für die Anwendung. Lege für Anwendungsnamen deshalb A-Einträge an. Zweitens die Intranetzone: Schickt der Browser den Namen nicht in die lokale Intranetzone, versucht er erst gar nicht, sich automatisch anzumelden, und der Benutzer sieht ein Anmeldefenster.

Drittens, und das ist die gemeinste Falle: Extended Protection for Authentication, also die Kanalbindung. Sie koppelt die Windows-Anmeldung an den TLS-Kanal zwischen Client und Server. Terminiert der LoadMaster TLS und baut zum Server einen neuen Kanal auf, passt die Bindung nicht mehr, und die Anmeldung scheitert. Bei http.sys ist die Kanalbindung standardmäßig aus, im IIS kannst du sie unter den erweiterten Einstellungen der Windows-Authentifizierung auf „Akzeptieren“ oder „Erforderlich“ stellen, und Sicherheitsrichtlinien verlangen genau das immer öfter. Wenn ja, reichst du TLS am LoadMaster durch, statt es zu terminieren. Mehr zu den Varianten steht im Beitrag TLS-Offload, Re-Encryption und Zertifikate: PKI trifft LoadMaster.

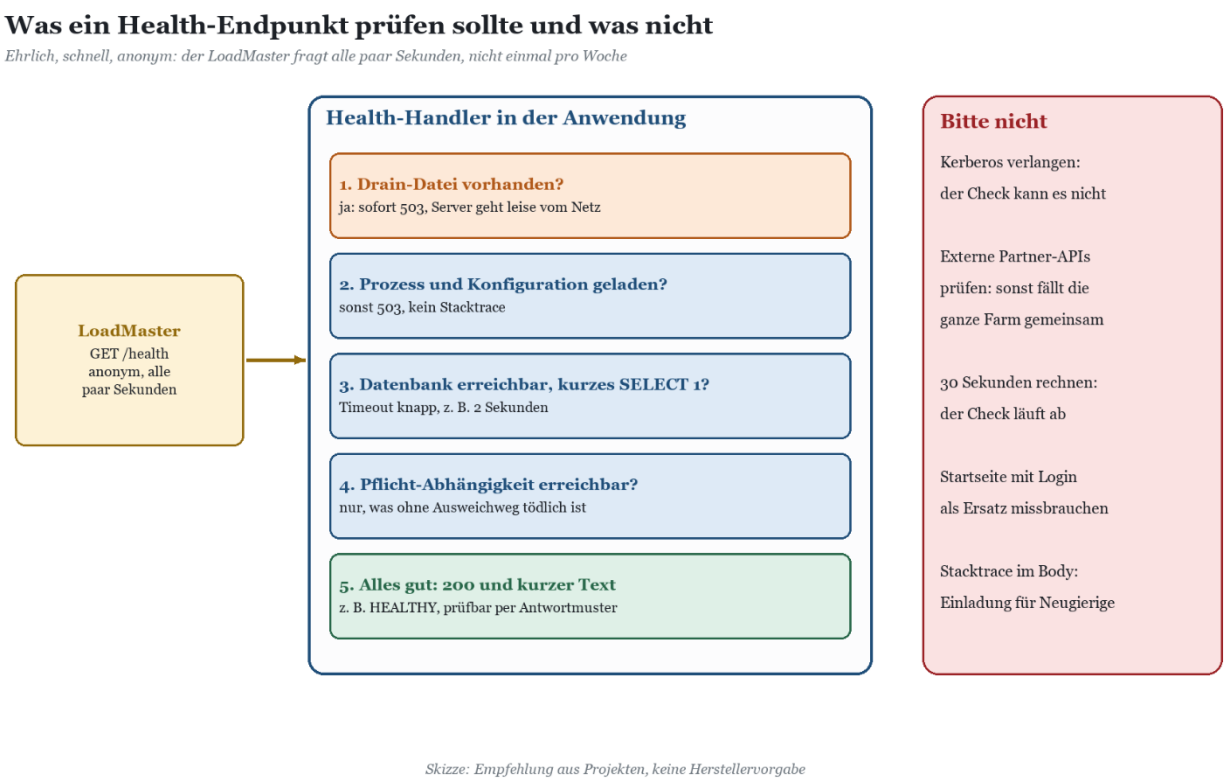
WICHTIG · Der stille NTLM-Fallback

Fast jede der hier beschriebenen Fehlkonfigurationen führt nicht zu einem sichtbaren Fehler, sondern zu NTLM. Die Anwendung funktioniert, nur mit dem schwächeren Protokoll, ohne Delegation und mit verbindungsgebundener Anmeldung, die sich mit Verbindungswiederverwendung schlecht verträgt. Wie du das erkennst und am LoadMaster behandelst, beschreibt der Beitrag NTLM, NTLM-Proxy und der Kerberos-Fallback, den keiner bemerkt.

Health-Endpunkt: lass ihn dir bauen

Der beliebteste Health Check für Eigenentwicklungen ist ein TCP-Check auf Port 443. Er prüft zuverlässig, ob http.sys läuft. Das tut es übrigens auch dann noch, wenn der Anwendungspool längst gestoppt ist, weil der Rapid-Fail-Schutz nach fünf Abstürzen in fünf Minuten die Reißleine gezogen hat. Der Client bekommt dann einen 503 von http.sys, der LoadMaster sieht einen offenen Port und ist zufrieden. Der zweitbeliebteste Check fragt die Startseite ab, die bei aktivierter Windows-Authentifizierung mit 401 antwortet. Je nach Konfiguration gilt das als gesund oder als tot, und beides ist gleich wenig aussagekräftig.

Die richtige Lösung kostet die Entwicklung einen Nachmittag: einen eigenen Endpunkt, der ehrlich Auskunft gibt. In ASP.NET Core ist das mit der eingebauten Health-Checks-Middleware und MapHealthChecks fast geschenkt, in klassischem ASP.NET ist es ein kleiner Handler. Bei Hersteller-APIs ist es eine Anforderung, die du schriftlich stellst. Wie du den Check am LoadMaster dann mit URL, Methode und Host-Header einrichtest und worauf du bei Intervallen und Schwellwerten achtest, erklärt der Beitrag Health Checks, die wirklich prüfen: warum „Port 443 offen“ nicht reicht.



Skizze: Empfehlung aus Projekten, keine Herstellervorgabe

Was ein guter Endpunkt prüft

Prüfung	Gehört in den Endpunkt?	Begründung
Prozess antwortet, Konfiguration geladen	Ja	Minimum; ohne das ist alles andere egal
Datenbankverbindung mit kurzem Abfrage-Timeout	Ja	Ohne Datenbank liefert die Anwendung nur Fehlerseiten aus
Drain-Schalter, etwa eine Datei oder ein Registry-Wert	Ja	Erlaubt Wartung, ohne am LoadMaster zu klicken
Interne Pflichtdienste, etwa ein Dateishare für Belege	Mit Augenmaß	Nur, wenn die Anwendung ohne sie wirklich nicht arbeiten kann
Externe Partner-APIs, Mailserver, Lizenzserver	Nein	Fallen für alle Knoten gleichzeitig aus und nehmen dann die ganze Farm vom Netz
Aufwendige Berichte oder Selbsttests	Nein	Der Check läuft alle paar Sekunden, auf jedem Knoten

Anonym, schnell und ehrlich

Der Endpunkt muss anonym erreichbar sein, denn der Health Check des LoadMasters spricht kein Kerberos. Im IIS erreichst du das mit einem location-Abschnitt in der web.config, der für den Pfad /health die anonyme Authentifizierung zulässt, während der Rest der Anwendung bei Windows-Authentifizierung bleibt. Damit der anonyme Pfad nichts verrät, gibt er nur einen Statuscode und ein kurzes Wort zurück, keine Versionsnummern, keine Servernamen, keine Verbindungszeichenfolgen. Wenn du ganz vorsichtig bist, blockierst du den Pfad zusätzlich per Content Rule am LoadMaster für externe Clients.

Schnell heißt: Die Prüfungen haben eigene, knappe Timeouts, und der Endpunkt antwortet innerhalb einer Zeit, die deutlich unter dem Timeout des Health Checks liegt. Ehrlich heißt: Ist die Datenbank weg, kommt ein 503 und kein freundliches 200 mit dem Text „Database: unavailable“

im Body. Den LoadMaster interessiert in erster Linie der Statuscode. Wer zusätzlich ein Antwortmuster prüft, sollte sich auf ein eindeutiges Wort einigen, das nur im Erfolgsfall erscheint.

Der Drain-Schalter für die Wartung

Der eleganteste Teil eines Health-Endpunkts ist der Drain-Schalter. Liegt auf dem Server eine bestimmte Datei, etwa C:\App\maintenance.flag, antwortet der Endpunkt mit 503. Der LoadMaster nimmt den Server nach den eingestellten Fehlversuchen aus der Verteilung, bestehende Verbindungen laufen aus, und das Patchteam kann in Ruhe arbeiten, ohne WUI-Zugang zum LoadMaster zu brauchen. Nach dem Neustart löscht ein Skript die Datei, und der Server kommt von allein zurück. Das passt hervorragend in Wartungsskripte, Ansible-Läufe oder eine geplante Aufgabe nach dem Patchdienstag.

TIPP · Anwendungspool warmhalten

Ein Health Check, der alle paar Sekunden anfragt, hat eine nette Nebenwirkung: Der Anwendungspool fällt nicht mehr in den Leerlauf-Timeout, der beim IIS standardmäßig nach 20 Minuten zuschlägt. Für den ersten Aufruf nach einem Recycling hilft zusätzlich das Modul Application Initialization, damit nicht der erste Benutzer am Morgen die Kompilierung bezahlt.

Session-State: wenn die Anwendung sich merkt, wer du bist

Eigenentwicklungen haben ein Gedächtnis, und es sitzt erstaunlich oft im Arbeitsspeicher. Der Warenkorb, der halb ausgefüllte Antrag, der Filter in der Liste, der Status eines Exportjobs: Alles liegt in der Sitzung, und die Sitzung liegt im Prozess. Auf einem einzelnen Server ist das schnell und bequem. Hinter dem LoadMaster ist es ein Glücksspiel, bei dem du mit jedem weiteren Knoten schlechtere Karten hast. Landet die nächste Anfrage woanders, ist der Antrag weg, und die Kollegin aus dem Bürgerbüro tippt ihn zum zweiten Mal ab. Beim dritten Mal ruft sie an.

Wohin mit dem Zustand

Variante	Technik	Hinter dem LoadMaster	Haken
InProc	Sitzung im Arbeitsprozess, Standard in ASP.NET	nur mit Persistenz	Recycling oder Failover löschen alles
StateServer	ASP.NET-Zustandsdienst auf einem Server	funktioniert	neuer Single Point of Failure
SQL Server	Sitzungen in einer Datenbank	funktioniert gut	etwas Latenz, Datenbank muss hochverfügbar sein
Verteilter Cache	ASP.NET Core IDistributedCache mit SQL oder Redis	funktioniert gut	zusätzliche Komponente im Betrieb
Zustandslos	alles im Token, in der Datenbank oder beim Client	ideal	erfordert Umbau, wenn nicht von Anfang an geplant

Machine Key und Data Protection: das unsichtbare Gemeinschaftsgut

Selbst eine Anwendung ganz ohne Sitzung kann an mehreren Knoten scheitern, und zwar an Schlüsseln, die niemand je bewusst angelegt hat. Klassisches ASP.NET verschlüsselt und signiert Formular-Authentifizierungscookies und ViewState mit dem Machine Key. Ist der auf jedem Server automatisch generiert, kann WEB02 das Cookie von WEB01 nicht lesen, und der Benutzer landet wieder auf der Anmeldeseite. Die Lösung ist ein fest eingetragener, auf allen Knoten identischer Machine Key in der Konfiguration.

ASP.NET Core hat dafür das Data-Protection-System mit einem Schlüsselbund, der standardmäßig pro Server und Benutzerprofil abgelegt wird. Für eine Farm muss dieser Schlüsselbund an einem gemeinsamen Ort liegen, etwa auf einem Dateishare, in einer Datenbank oder in einem Cache, und alle Knoten müssen denselben Anwendungsnamen verwenden. Sonst scheitern Anmeldecookies, Antiforgery-Token und alles andere, was die Anwendung verschlüsselt, sobald der LoadMaster einmal wechselt. Der Fehler zeigt sich typischerweise als Anmeldeschleife oder als 400 beim Absenden eines Formulars, und zwar nur manchmal.

WARNUNG · „Läuft doch“ heißt bei Persistenz gar nichts

Mit Persistenz am LoadMaster bleiben Benutzer meist am selben Server kleben, und Machine-Key- oder Session-Probleme bleiben unsichtbar. Sie tauchen erst auf, wenn ein Server ausfällt, gepatcht wird oder die Persistenz abläuft. Teste die Farm deshalb bewusst einmal ohne Persistenz oder mit einem gezielt abgeschalteten Knoten, bevor du sie freigibst.

Persistenz als Krücke, nicht als Fundament

Natürlich kannst du Anwendungen mit InProc-Sitzungen hinter den LoadMaster stellen. Persistenz per Cookie, wenn der LoadMaster TLS terminiert, oder per Quell-IP, wenn er durchreicht, sorgt dafür, dass ein Benutzer bei seinem Server bleibt. Das ist für viele Altanwendungen der pragmatische Weg und völlig legitim. Sei dir nur bewusst, was du damit kaufst: Lastverteilung nach dem Zufallsprinzip der Anmeldung, Datenverlust beim Ausfall eines Knotens und Benutzer hinter einem gemeinsamen NAT, die alle auf demselben Server landen. Welche Persistenzverfahren der LoadMaster anbietet und wann welches passt, steht im Beitrag Persistence und Scheduling: wer hängt an welchem Server und warum.

Praxisfall: eine ERP-API hinter dem LoadMaster mit Kerberos

Ein mittelständisches Unternehmen betreibt ein ERP-System, dessen Hersteller eine REST-Schnittstelle als Windows-Dienst mitliefert. Der Dienst nutzt http.sys, lauscht auf Port 8443 und unterstützt Windows-Authentifizierung. Angebunden sind ein Excel-Add-in in der Controlling-Abteilung, ein halbes Dutzend PowerShell-Skripte der IT und ein internes Webportal auf dem IIS, das im Namen der angemeldeten Benutzer Bestellungen abfragt. Nach einem Ausfall des einzigen API-Servers mitten im Quartalsabschluss wurden zwei weitere Knoten aufgebaut und ein Virtual Service erp-api.contoso.local am LoadMaster angelegt. Seitdem funktionierte es manchmal.

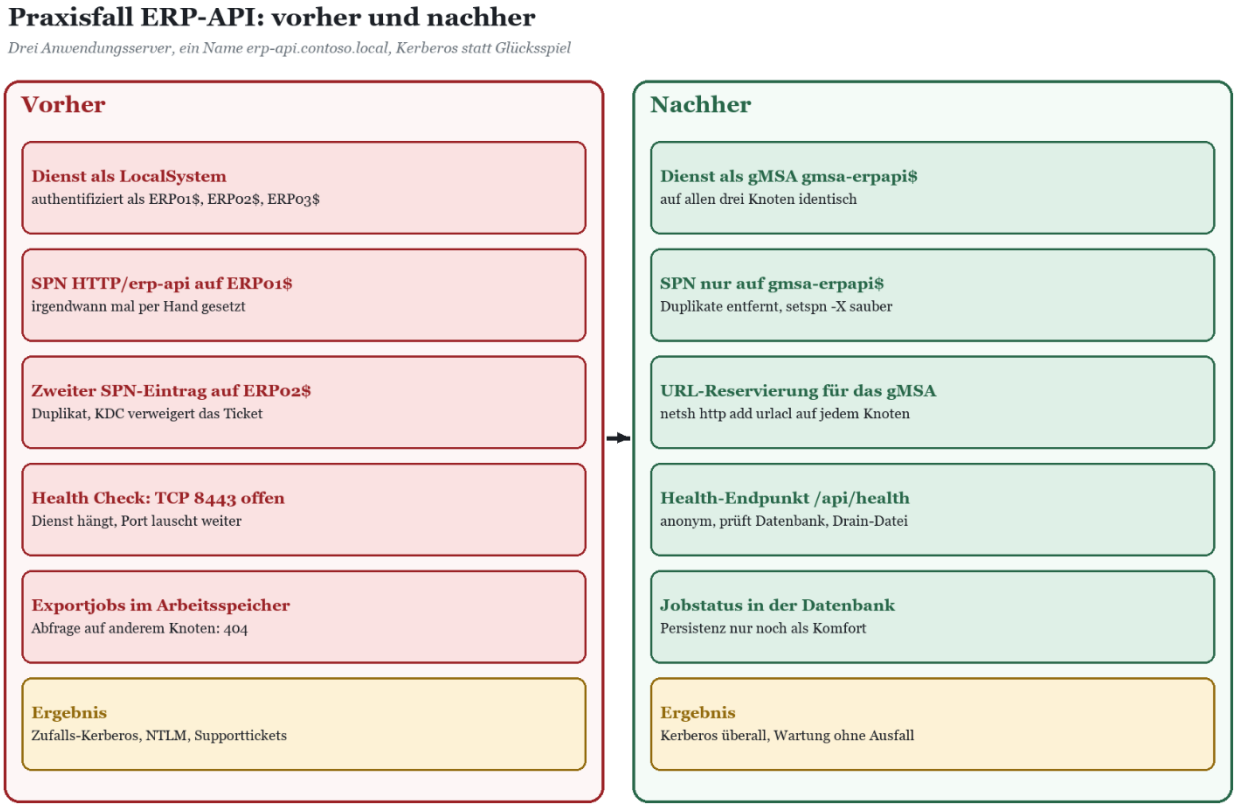
Die Symptome

- **Excel-Add-in:** sporadische Anmeldefenster, vor allem morgens, wenn viele Benutzer gleichzeitig starten.
- **PowerShell-Skripte:** Invoke-RestMethod mit Standardanmeldeinformationen lief mal durch, mal kam ein 401.
- **Webportal:** Bestellabfragen im Namen der Benutzer scheiterten regelmäßig, weil die Delegation ein Kerberos-Ticket für die API braucht und kein NTLM akzeptiert.
- **Exporte:** große Exportjobs liefen an, die anschließende Statusabfrage endete häufig mit 404.
- **Monitoring:** der TCP-Check auf 8443 blieb grün, auch als ein Knoten nach einem Update nur noch Fehler 500 auslieferte.

Die Ursachen

Die Analyse dauerte einen Vormittag, die Erklärung passt auf einen Bierdeckel. Der Dienst lief, wie vom Installer vorgeschlagen, auf allen drei Knoten als LocalSystem. Der SPN HTTP/erp-api.contoso.local war irgendwann am Computerkonto des ersten Servers eingetragen worden, und ein hilfsbereiter Kollege hatte ihn beim Aufbau des zweiten Knotens dort ebenfalls registriert.

setspn -X förderte das Duplikat sofort zutage. Mit doppeltem SPN stellte der Domänencontroller gar kein Ticket mehr aus, und alles, was funktionierte, funktionierte per NTLM. Die Statusabfrage der Exporte scheiterte, weil der Dienst Jobs im Arbeitsspeicher verwaltete und die Persistenz am LoadMaster für die Skripte mit wechselnden Quellports und kurzen Verbindungen nicht griff. Und der Health Check prüfte, ob der Port offen ist, was er natürlich war.



Skizze: anonymisierter Praxisfall, Namen fiktiv

Die Lösung Schritt für Schritt

- **Konto:** Der Hersteller bestätigte auf Nachfrage die Unterstützung von gruppenverwalteten Dienstkonten. Angelegt wurde gmsa-erpapi$, abrufbar für eine Gruppe mit den drei API-Servern, installiert auf jedem Knoten.
- **SPN:** Duplikate an den Computerkonten entfernt, HTTP/erp-api.contoso.local einmalig per setspn -S am gMSA registriert. Der Name ist ein A-Eintrag auf die VIP, kein CNAME.
- **Dienst:** Anmeldekonto des Windows-Dienstes auf allen Knoten auf das gMSA umgestellt, URL-Reservierung per netsh http add urlacl für das gMSA ergänzt, Zertifikatsbindung kontrolliert. Der Test mit klist und Ereignis 4624 zeigte Kerberos auf jedem Knoten einzeln.
- **Delegation:** Das Webportal darf jetzt per ressourcenbasierter eingeschränkter Delegation an den SPN der API delegieren. Weil der SPN an einem einzigen Konto hängt, reicht ein Eintrag für die ganze Farm.
- **Health-Endpunkt:** Der Hersteller lieferte mit dem nächsten Update einen anonymen Endpunkt /api/health, der Datenbankverbindung und eine Drain-Datei prüft. Der LoadMaster fragt ihn per HTTPS ab und wertet den Statuscode aus.
- **Zustand:** Ein Konfigurationsschalter des Herstellers verlegt den Jobstatus in die ERP-Datenbank. Persistenz nach Quell-IP blieb als Komfort aktiv, ist aber nicht mehr tragend.

Die Delegation ist das Kapitel, an dem die meisten Projekte länger hängen als am Rest zusammen. Wie KCD und RBCD für Dienste hinter dem LoadMaster funktionieren und warum der eine

gemeinsame SPN dabei so viel Arbeit spart, beschreibt der Beitrag Kerberos Constrained Delegation am LoadMaster: KCD und RBCD sauber einrichten.

Kriterium	Vorher	Nachher
Authentifizierungsprotokoll	überwiegend NTLM, Kerberos nur zufällig	Kerberos auf allen Knoten
Delegation aus dem Webportal	scheitert regelmäßig	funktioniert per RBCD
Kontopflege	keine, weil LocalSystem	keine, weil gMSA
Erkennung eines kranken Knotens	gar nicht	binnen weniger Health-Check-Intervalle
Wartung eines Knotens	Absprache, Klick am LoadMaster, Hoffnung	Drain-Datei, Patch, Datei löschen
Exportjobs bei Knotenwechsel	404	Status aus der Datenbank

AUS DER PRAXIS · Was wir aus dem Fall mitnehmen

Der LoadMaster war in diesem Projekt der einzige Teil, an dem nichts geändert werden musste, abgesehen vom Health Check. Alle Probleme lagen in Identität, SPN und Zustand der Anwendung. Das ist bei Eigenentwicklungen und Hersteller-APIs die Regel, nicht die Ausnahme, und genau deshalb lohnt sich der Blick auf die Windows-Seite, bevor du am Virtual Service schraubst.

FAQ: Eigenentwicklungen und http.sys-Dienste hinter dem LoadMaster

Brauche ich für Kerberos hinter dem LoadMaster immer ein Dienstkonto?

Sobald mehrere Server unter einem gemeinsamen Namen Kerberos sprechen sollen, ja. Die Alternative wäre, dass jeder Server unter eigenem Namen erreichbar ist, und das ist dann kein Load Balancing mehr. Ein gruppenverwaltetes Dienstkonto ist die bequemste Variante.

Kann der LoadMaster das Kerberos-Problem für mich lösen?

Nur, wenn er selbst die Authentifizierung übernimmt, etwa per Vorauthentifizierung mit Edge Security Pack und eingeschränkter Delegation zum Server. Dann braucht der Server aber trotzdem einen sauberen SPN. Als reiner Durchreicher ändert der LoadMaster nichts am Ticket.

Was mache ich, wenn der Hersteller kein gMSA unterstützt?

Dann nimmst du ein klassisches Domänenkonto als Dienstkonto, setzt ein langes Kennwort, aktivierst AES für das Konto und planst die Kennwortänderung als wiederkehrende Aufgabe ein. Unschön, aber deutlich besser als LocalSystem mit Zufalls-Kerberos.

Muss ich useAppPoolCredentials auch setzen, wenn der App-Pool als ApplicationPoolIdentity läuft?

Nein. In dem Fall tritt der Pool als Computerkonto auf, und der Kernelmodus entschlüsselt ohnehin mit dem Computerkonto. Für eine Farm unter gemeinsamem Namen taugt diese Kombination aber nicht, weil der SPN dann nur an einem Server hängen kann.

Warum funktioniert Kerberos über den Servernamen, aber nicht über den VIP-Namen?

Weil für den Servernamen automatisch der Host-SPN des Computerkontos greift, für den VIP-Namen aber ein eigener SPN am richtigen Konto fehlt oder doppelt vorhanden ist. setspn -Q für den VIP-Namen bringt meistens sofort Klarheit.

Darf der LoadMaster TLS terminieren, wenn die Anwendung Windows-Authentifizierung nutzt?

Ja, solange Extended Protection auf dem Server nicht erzwungen wird. Ist die Kanalbindung auf „Erforderlich“ gestellt, reichst du TLS durch oder verzichtest am Server darauf.

Wie prüfe ich, welches Konto eine URL bei http.sys belegt hat?

Mit netsh http show urlacl siehst du die Reservierungen, mit netsh http show servicestate die aktiven Anforderungswarteschlangen und Prozesse. Die Fehlerprotokolle von http.sys unter LogFiles\HTTPERR im System32-Verzeichnis zeigen zusätzlich, welche Anfragen gar nicht erst bei der Anwendung angekommen sind.

Reicht es, wenn der Health Check die Startseite abruft?

Selten. Bei Windows-Authentifizierung antwortet sie mit 401, ohne Datenbank oft trotzdem mit 200. Ein eigener anonymer Endpunkt, der die Pflicht-Abhängigkeiten prüft, ist die einzige Variante, die wirklich etwas aussagt.

Fazit: erst die Identität, dann der Virtual Service

Eigenentwicklungen und Hersteller-APIs scheitern hinter dem LoadMaster fast nie am LoadMaster. Sie scheitern an Annahmen, die auf einem einzelnen Server nie geprüft wurden: dass das Computerkonto schon passt, dass der SPN schon irgendwo hängt, dass ein offener Port ein gesunder Dienst ist und dass der Arbeitsspeicher ein guter Ort für Sitzungen ist. Wer vor dem Go-live die vier Fragen dieses Beitrags beantwortet, spart sich die Wochen, in denen „manchmal funktioniert es“ zum Betriebszustand wird.

Die Reihenfolge ist dabei immer dieselbe: ein gemeinsames Dienstkonto, am besten als gMSA, ein einziger SPN für den VIP-Namen, beim IIS useAppPoolCredentials, bei http.sys URL-Reservierung und eine geklärte Kontofrage. Dann ein ehrlicher, anonymer Health-Endpunkt mit Drain-Schalter. Dann ein Zustand, der nicht im Prozess wohnt. Erst danach kommt der Virtual Service, und der ist dann erfreulich langweilig. Langweilig ist im Betrieb das höchste Lob. Spannend wird es ohnehin wieder, sobald jemand die nächste Schnittstelle „nur mal schnell“ auf einem Server installiert.

TIPP · Unterstützung bei Planung und Umsetzung

Wenn du eine Eigenentwicklung oder Hersteller-API mit Kerberos hinter den LoadMaster bringen willst oder herausfinden musst, warum sie heimlich NTLM spricht, hilft dir die Beratung zum Kemp LoadMaster. Wer Virtual Services, Health Checks und Kerberos hinter einer VIP mit dem eigenen Team im Labor durchspielen möchte, findet das in der Kemp-LoadMaster-Schulung.

WEITERLESEN · Weiter im Thema

› SPN-Probleme hinter dem Load Balancer: ein Name, mehrere Server
› Kerberos Constrained Delegation am LoadMaster: KCD und RBCD sauber einrichten
› Persistence und Scheduling: wer hängt an welchem Server und warum
› Health Checks, die wirklich prüfen: warum „Port 443 offen“ nicht reicht
› NTLM, NTLM-Proxy und der Kerberos-Fallback, den keiner bemerkt

› Migration von Windows NLB, IIS ARR oder TMG-Erbe auf den LoadMaster