

App-Registrierungen und Consent-Governance

Es gibt einen Angriff auf deinen Microsoft-365-Tenant, bei dem dein Passwort keine Rolle spielt. MFA auch nicht. Conditional Access nur am Rande. Der Angreifer bekommt Zugriff auf Postfächer, Dateien und Kontakte, und die einzige Handlung, die dafür nötig war, ist ein Klick auf eine Schaltfläche mit der Aufschrift **Zustimmen**. Auf einer echten Microsoft-Seite. Mit gültigem Zertifikat. Ohne Tippfehler in der Domain.

Das ist Consent-Phishing, und es ist der unangenehmste Vertreter seiner Gattung, weil es sämtliche Reflexe deiner Organisation ins Leere laufen lässt. Passwort zurücksetzen? Ändert nichts. MFA neu registrieren? Ändert nichts. Der Zugriff hängt nämlich nicht am Konto, sondern an einer separaten Zustimmung, die im Verzeichnis liegt und geduldig darauf wartet, dass niemand hinschaut.

Dieser Beitrag gehört zum Handlungsfeld C der Serie [Microsoft 365 Security](#) und behandelt alles, was mit App-Registrierungen, Enterprise Applications und Zustimmungen zu tun hat: wie der Angriff funktioniert, welche drei Schalter du im Tenant umlegst, wie du ein belastbares Inventar aufbaust und warum ein guter Teil deiner App-Registrierungen vermutlich aus Karteileichen besteht, die niemand vermisst — außer dem nächsten Angreifer.

FAKTEN Worum es hier konkret geht

- › User Consent so einstellen, dass Anwender nicht mehr die Tür aufhalten können
- › Admin-Consent-Workflow als geordneten Ersatz, damit niemand am Prozess vorbeiarbeitet
- › Inventar über Enterprise Applications, Berechtigungen, Eigentümer und Anmeldeinformationen
- › Application Permissions von Delegated Permissions unterscheiden — und entsprechend behandeln
- › Zertifikate statt Client Secrets, App Management Policies, verwaiste Apps abräumen

Consent-Phishing: der Angriff, der dein Passwort ignoriert

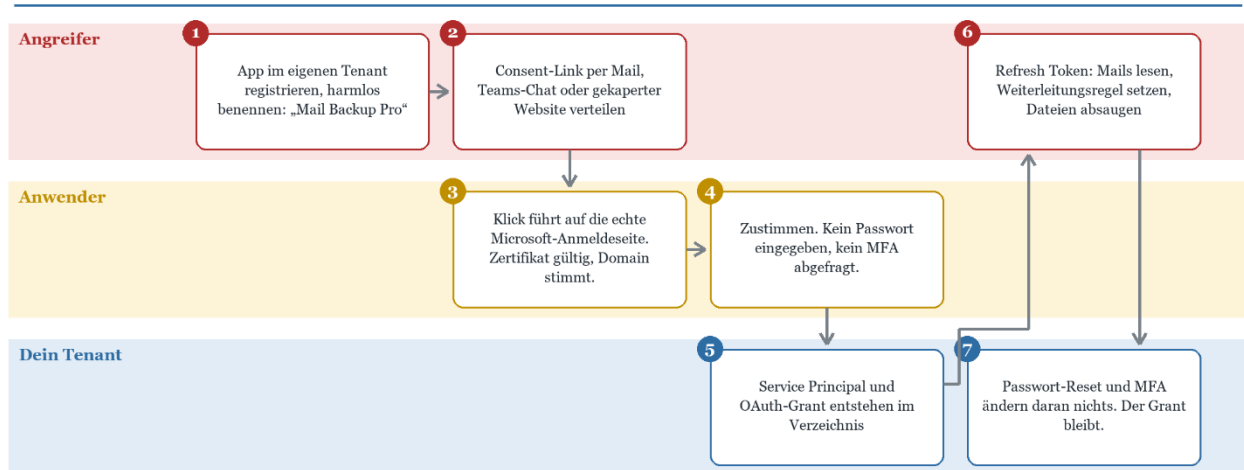
OAuth 2.0 ist die Technologie, die dafür sorgt, dass du einem Werkzeug Zugriff auf deine Daten geben kannst, ohne ihm dein Passwort zu geben. Das ist eine hervorragende Idee. Sie ist so hervorragend, dass Angreifer sie ebenfalls schätzen gelernt haben. Denn was für den gutartigen Kalender-Assistenten gilt, gilt genauso für die App namens *Mail Backup Pro*, die es seit zwei Tagen gibt und deren Herausgeber in einem Tenant sitzt, den es ebenfalls erst seit zwei Tagen gibt.

Der Ablauf in sieben Schritten

Der Angreifer registriert im eigenen Verzeichnis eine mehrmandantenfähige Anwendung. Er wählt einen Namen, der glaubwürdig klingt, hinterlegt eine Redirect-URI auf einen von ihm kontrollierten Server und fordert Berechtigungen an, die nach Produktivitätswerkzeug aussehen: Mail lesen, Dateien lesen, Kontakte lesen, dazu `offline_access` für ein Refresh Token. Dann baut er den Consent-Link und verteilt ihn auf dem üblichen Weg — Mail, Teams-Chat, kompromittierte Partnerseite, QR-Code auf einer angeblichen Rechnung.

Consent-Phishing: der Ablauf

Kein Passwort, kein MFA-Prompt — nur ein Klick auf „Zustimmen“



Zwischen Schritt 4 und Schritt 6 vergehen in echten Vorfällen Sekunden. Die Entdeckung braucht im Regelfall Tage.

Skizze 1: Der Ablauf eines Consent-Phishing-Angriffs. Die Anmeldeseite ist echt, das Zertifikat ist gültig, der Dialog stammt original von Microsoft. Nur die App ist es nicht.

Der entscheidende Punkt steht in Schritt 3: Die Seite, auf der dein Kollege landet, ist keine Fälschung. Es ist die Anmeldeseite von Microsoft, gehostet von Microsoft, mit einem Zertifikat von Microsoft. Alles, was du deiner Belegschaft über URL-Prüfung und Zertifikatswarnungen beigebracht hast, greift hier nicht. Der Nutzer sieht ein Berechtigungsfenster mit einer Liste, die er nicht liest, und einen Knopf, den er drückt, weil er weiterarbeiten will.

Ab Schritt 5 existiert im Verzeichnis ein Service Principal für die fremde Anwendung und ein OAuth-Grant, der ihre Rechte festhält. Die App bekommt einen Authorization Code, tauscht ihn gegen ein Access Token und ein Refresh Token — und ab da braucht sie den Menschen nicht mehr. Das Refresh Token hält sie am Leben, solange niemand den Grant entfernt.

WARNUNG Warum die Standardreaktion hier versagt

Passwort-Reset, MFA-Neuregistrierung und das Sperren des Kontos richten sich gegen die Anmeldung des Nutzers. Ein erteilter OAuth-Grant hängt nicht an der Anmeldung. Die App ist eine eigene Identität mit eigenen Tokens. Wenn du nach einem Vorfall nur das Konto aufräumst und den Grant stehen lässt, hast du den Angreifer nicht ausgesperrt, sondern nur höflich umgesetzt.

Reihenfolge im Ernstfall: App deaktivieren, OAuth-Grants entziehen, Refresh Tokens widerrufen, Postfachregeln prüfen — erst dann über Passwörter reden. Der vollständige Ablauf steht in [Kompromittiertes Konto erkennen und aufräumen](#).

Was Microsoft von sich aus tut — und wo es aufhört

Entra ID ist nicht untätig. Auffällige Anwendungen werden von Microsoft geprüft und bei bestätigtem Verstoß deaktiviert; die betroffenen Objekte tragen dann den Status `DisabledDueToViolationOfServicesAgreement` und lassen sich bewusst nicht mehr löschen, damit sie in keinem Tenant erneut auftauchen können. Hat jemand aus deiner Organisation der App vorher zugestimmt, bekommt ein Administrator eine Benachrichtigung. Dazu kommt die risikobasierte Zustimmungseskalation: Erkennt Entra ID eine riskante Consent-Anfrage — etwa

eine mehrmandantenfähige App ohne verifizierten Herausgeber — verweigert es die Nutzerzustimmung und verlangt eine Administratorfreigabe.

Das ist gut, aber es ist Nachlauf. Die Deaktivierung kommt, nachdem genügend Opfer gemeldet haben. Bereits ausgestellte Access Tokens bleiben bis zu ihrem Ablauf gültig. Und die risikobasierte Eskalation wirkt überhaupt nur dort, wo Nutzerzustimmung noch erlaubt ist — sie ist eine Bremse, kein Verbot. Deine eigene Konfiguration entscheidet, ob der Angriff eine Chance bekommt.

FAKTEN Ein Muster aus dem Frühjahr 2026

Anfang März 2026 beschrieb Microsoft eine Kampagne, die gar nicht auf die Zustimmung zielte, sondern auf die Umleitungsmechanik von OAuth selbst: Präparierte Autorisierungs-Links mit ungültigen Scopes und dem Parameter `prompt=none` provozierten einen Fehlerfall, woraufhin der Identitätsanbieter die Anwender brav auf die im `state`-Parameter hinterlegte Angreiferdomain weiterleitete. Von dort kamen ZIP-Archive mit LNK-Dateien, PowerShell und DLL-Sideloadung. Microsoft deaktivierte 18 beteiligte OAuth-Client-IDs; im Visier standen vor allem Behörden und öffentliche Einrichtungen.

Lehre daraus: Der Consent-Dialog ist nicht die einzige Angriffsfläche der App-Plattform. Aber er ist die mit dem größten Schaden pro Klick.

Delegated oder Application: der Unterschied, der über alles entscheidet

Wenn du in einer App-Registrierung auf *API-Berechtigungen* klickst, fragt Microsoft dich zuerst nach dem Typ. Diese Frage sieht harmlos aus und ist die wichtigste des ganzen Bildschirms. Denn `Mail.Read` als `Delegated Permission` und `Mail.Read` als `Application Permission` sind namensgleiche Zwillinge mit völlig verschiedenen Lebensläufen.

Delegated gegen Application Permissions

Derselbe Berechtigungsname, zwei völlig verschiedene Reichweiten

Delegated (App + angemeldeter Nutzer)	Application (App allein, ohne Nutzer)
Nutzer meldet sich an Interaktive Anmeldung, MFA und Conditional Access greifen.	Kein Mensch beteiligt Client Credentials Flow. MFA und Conditional Access für Nutzer greifen nicht.
Token trägt den Nutzerkontext Schnittmenge aus App-Berechtigung und dem, was der Nutzer selbst darf	Token trägt nur die App-Identität Die App bekommt exakt das, was in der App-Rolle steht — ungebremst
Reichweite Genau ein Postfach, genau ein OneDrive — das des angemeldeten Nutzers	Reichweite Standardmäßig ALLE Postfächer, ALLE Sites, ALLE Objekte im Tenant
Wer stimmt zu Der Nutzer selbst, sofern du es erlaubst. Sonst der Administrator.	Wer stimmt zu Ausschließlich ein Administrator. Immer. Ohne Ausnahme.

Merksatz: Delegated ist ein Schlüssel für eine Tür. Application ist der Generalschlüssel für das ganze Haus.

Skizze 2: Gleicher Berechtigungsname, andere Reichweite. Rechts fehlt der Mensch — und mit ihm alles, was du an nutzerbezogenen Schutzmaßnahmen aufgebaut hast.

Merkmal	Delegated Permission	Application Permission
Wer ist beteiligt	Die App handelt im Namen eines angemeldeten Nutzers	Die App handelt allein, ohne Nutzer (Dienst, Daemon, Skript)
Reichweite	Schnittmenge aus App-Berechtigung und den Rechten des Nutzers	Alles, was die App-Rolle hergibt — standardmäßig tenantweit
Wer darf zustimmen	Der Nutzer selbst (sofern erlaubt) oder ein Administrator	Ausschließlich ein Administrator, ohne Ausnahme
MFA und Conditional Access	Greifen bei der Anmeldung des Nutzers	Greifen nicht; nur Conditional Access für Workload Identities hilft
Anmeldeverfahren	Authorization Code Flow mit Refresh Token	Client Credentials Flow mit Secret, Zertifikat oder föderierter Anmeldeinformation
Typischer Missbrauch	Consent-Phishing gegen Endanwender in der Breite	Diebstahl eines Secrets aus Repository, Pipeline oder Skript
Sichtbarkeit im Protokoll	Interaktive Anmeldungen, der Nutzer ist erkennbar	Dienstprinzipal-Anmeldungen, leicht zu übersehen

Tabelle 1: Die beiden Berechtigungstypen im direkten Vergleich.

Die praktische Konsequenz: Delegated Permissions sind das Massengeschäft des Consent-Phishings, weil sie sich an jeden Endanwender richten. Application Permissions sind das Zielobjekt für Rechteauserweiterung, weil eine einzige gestohlene Anmeldeinformation genügt, um alle Postfächer zu lesen. Beide brauchen Governance, aber eine unterschiedliche.

Berechtigungen, die immer Alarm sind

WARNUNG Diese drei siehst du nie ohne triftigen Grund

Mail.ReadWrite (Application) — Vollzugriff auf jedes Postfach im Tenant, lesend und schreibend. Damit lassen sich Nachrichten nicht nur absaugen, sondern auch verändern und löschen. Eine gefälschte Zahlungsanweisung im Posteingang der Buchhaltung sieht dann echt aus, weil sie es technisch ist.

full_access_as_app — die EWS-Berechtigung, die einer Anwendung erlaubt, sich als jedes beliebige Postfach der Organisation auszugeben. Der Name ist ehrlicher als alles andere in diesem Bereich. Wenn du sie wirklich brauchst, dann ausschließlich zusammen mit einer Einschränkung über RBAC für Anwendungen in Exchange Online.

Directory.ReadWrite.All — Schreibzugriff auf das Verzeichnis. Damit kann eine App Gruppenmitgliedschaften ändern, Anmeldeinformationen an Dienstprinzipale hängen und sich Schritt für Schritt zur Verzeichnisherrschaft hocharbeiten. In derselben Liga spielen `Application.ReadWrite.All`, `RoleManagement.ReadWrite.Directory` und `User.ManageCreds.All`.

Dazu eine unbequeme Beobachtung aus der Incident-Response-Praxis von Microsoft: In rund 99 Prozent der Consent-Phishing-Fälle genügten den Angreifern die harmlos wirkenden delegierten Berechtigungen rund um Mail, Kontakte, Postfacheinstellungen, Personen, Dateien und Notizen. Niemand hält delegiertes `Files.Read` für gefährlich. Genau deshalb funktioniert es.

Berechtigung	Typ	Was damit möglich wird	Einstufung
User.Read	Delegated	Eigenes Profil lesen. Der Normalfall für die Anmeldung.	Unbedenklich
offline_access	Delegated	Refresh Token — der Zugriff überlebt die Sitzung	Allein unbedenklich, in Kombination kritisch

Mail.Read / Mail.ReadWrite	Delegated	Postfach des Opfers lesen, Regeln und Inhalte ändern	Hoch — das Standardziel
Files.ReadWrite.All	Delegated	Alle für den Nutzer erreichbaren Dateien, auch geteilte	Hoch
Mail.ReadWrite	Application	Jedes Postfach im Tenant, lesend und schreibend	Kritisch
full_access_as_app	Application	Identitätsübernahme jedes Postfachs über EWS	Kritisch
Directory.ReadWrite.All	Beide	Verzeichnisobjekte ändern, Rechte ausweiten	Kritisch
RoleManagement.ReadWrite.Directory	Beide	Rollenzuweisungen ändern — der direkte Weg zum globalen Administrator	Kritisch
Directory.AccessAsUser.All	Delegated	Handeln mit allen Rechten des angemeldeten Nutzers	Kritisch bei privilegierten Konten

Tabelle 2: Eine Arbeitsliste für das Inventar. Die rechte Spalte ist die, nach der du sortierst.

Diese Liste ist bewusst kurz. Es gibt tausende Berechtigungen in Microsoft Graph, und der Versuch, sie alle zu klassifizieren, endet in einer Tabelle, die niemand pflegt. Nimm stattdessen die Faustregel: Alles mit .All im Namen, alles Schreibende und jede Application Permission gehört auf den Tisch. Der Rest darf durchlaufen.

Die drei Schalter, die den Unterschied machen

Die gute Nachricht: Gegen Consent-Phishing gibt es eine Konfiguration, die wirkt, nichts extra kostet und in zwanzig Minuten steht. Die schlechte: Sie erzeugt Arbeit, und zwar bei dir. Genau deshalb steht sie in vielen Tenants noch auf Werkseinstellung, und die Werkseinstellung lautet sinngemäß: Jeder darf zustimmen, solange kein Administrator gefragt werden muss.

Schalter eins: die Einstellungen für die Benutzerzustimmung

Du findest die Einstellung im Microsoft-Entra-Admin-Center unter *Anwendungen > Unternehmensanwendungen > Zustimmung und Berechtigungen > Einstellungen für die Benutzerzustimmung*. Drei Optionen stehen zur Wahl, und jede von ihnen ist hinter den Kulissen die Zuweisung einer Berechtigungserteilungsrichtlinie an die Standardnutzerrolle.

Option	Richtlinien-ID	Wirkung	Passt zu
Benutzerzustimmung nicht zulassen	keine managePermissionGrantsForSelf-Richtlinie zugewiesen	Jede Zustimmung läuft über einen Administrator. Maximale Kontrolle, maximaler Aufwand.	Regulierte Umgebungen, kleine Tenants, Behörden
Zustimmung für Apps verifizierter Herausgeber, für ausgewählte Berechtigungen	microsoft-user-default-low	Nutzer dürfen nur Apps verifizierter Herausgeber oder aus dem eigenen Tenant zustimmen, und nur für Berechtigungen, die du als „low impact“ eingestuft hast.	Die Empfehlung für so gut wie jeden Mittelständler
Zustimmung für alle Apps zulassen	microsoft-user-default-legacy	Jeder Nutzer darf jeder App jede Berechtigung erteilen, die keine Administratorzustimmung erfordert. Der Auslieferungszustand.	Niemanden, den ich kenne

Tabelle 3: Die drei Zustimmungsstufen und ihre technische Entsprechung.

WICHTIG Der Stolperstein, über den alle fallen

Die mittlere Option wirkt erst, wenn du unter *Berechtigungsklassifizierungen* tatsächlich Berechtigungen als „low impact“ markiert hast. Tust du das nicht, ist die Liste der erlaubten Berechtigungen leer — und die Einstellung verhält sich exakt wie ein komplettes Verbot. Danach kommen die Tickets: *Für diese App wird eine Genehmigung des Administrators benötigt.*

Sinnvolle Startklassifizierung: User.Read, User.ReadBasic.All, openid, profile, email und offline_access. Das deckt die Anmeldung ab und sonst nichts. Alles Weitere nimmst du bewusst dazu, einzeln, mit Begründung.

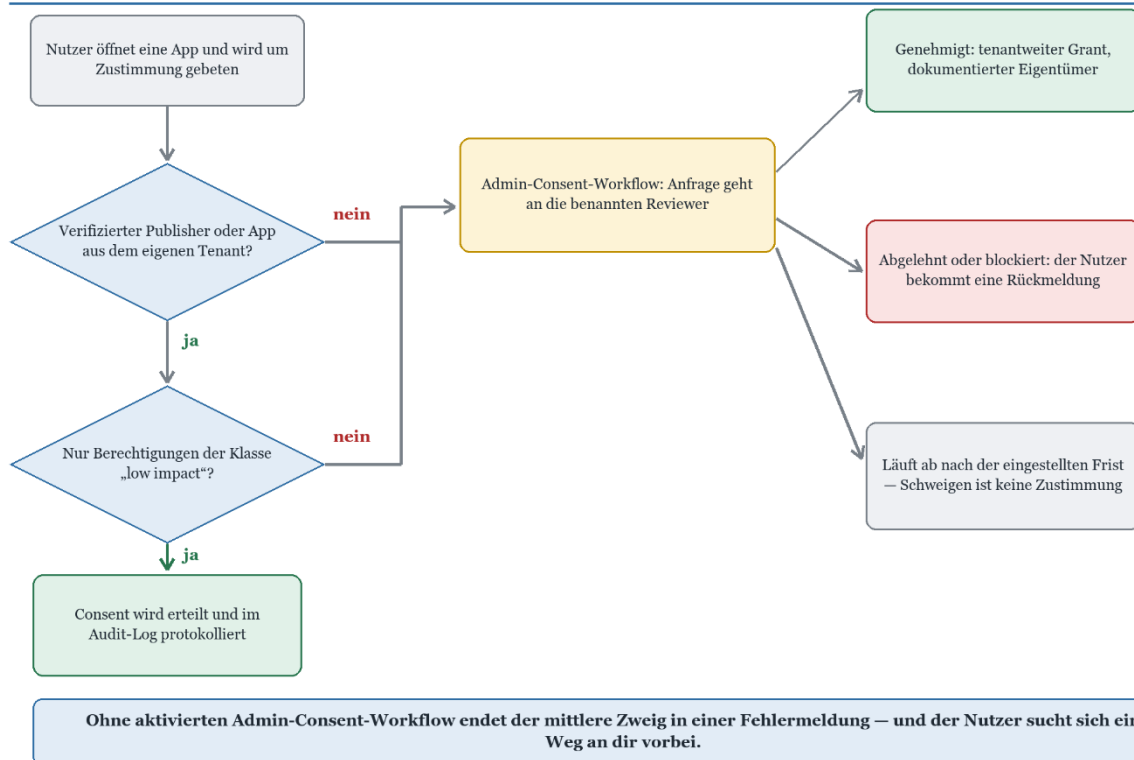
Und noch eine Unbequemlichkeit: Eine Änderung der Zustimmungseinstellungen gilt nur für künftige Vorgänge. Bereits erteilte Zustimmungen bleiben unverändert bestehen. Der Schalter schützt die Zukunft, nicht die Vergangenheit — die Vergangenheit räumst du im Inventar auf.

Schalter zwei: der Admin-Consent-Workflow

Wenn du Nutzern die Zustimmung nimmst, musst du ihnen einen Weg geben. Sonst passiert das, was immer passiert: Jemand legt sich einen privaten Tenant an, jemand exportiert die Daten auf einen Privatrechner, jemand findet ein Werkzeug, das ohne Anmeldung auskommt und dafür alles über die Zwischenablage macht. Schatten-IT ist keine Charakterschwäche, sondern eine Reaktion auf fehlende Wege.

Der Weg einer Consent-Anfrage

So sieht es aus, wenn User Consent auf „verifizierte Publisher, geringe Berechtigungen“ steht



Skizze 3: Der Entscheidungspfad einer Zustimmungsanfrage. Der Zweig in die Mitte braucht einen Empfänger — sonst endet er in einer Fehlermeldung.

Der Admin-Consent-Workflow wird unter *Zustimmung und Berechtigungen* > *Einstellungen für die Administratorzustimmung* aktiviert. Du benennst Prüfer — Nutzer, Gruppen oder Rollen —, legst fest, ob sie Benachrichtigungen und Ablaufferinnerungen bekommen, und bestimmst, nach wie vielen Tagen eine Anfrage verfällt. Bis die Aktivierung greift, kann eine Stunde vergehen.

TIPP Was du beim Workflow von Anfang an richtig machst

- › **Prüfer sind eine Gruppe, keine Person.** Urlaub, Krankheit und Kündigung sind keine Ausnahmefälle, sondern der Normalbetrieb.
- › **Prüfer zu sein verleiht keine Rechte.** Wer genehmigen will, braucht zusätzlich die Rolle Anwendungsadministrator oder Cloudanwendungsadministrator. Für Anwendungsberechtigungen auf Microsoft Graph bleibt es beim globalen Administrator.
- › **Ablauffrist bewusst wählen.** Sieben bis vierzehn Tage sind ein guter Kompromiss. Zu kurz erzeugt Frust, zu lang erzeugt eine Warteschlange, die niemand mehr aufarbeitet.
- › **Neue Prüfer erben keine alten Anfragen.** Wer nachträglich in die Liste kommt, sieht bestehende Anfragen nicht. Wer herausgenommen wird, bekommt für seine alten Anfragen weiterhin Erinnerungen. Plane Übergaben also von Hand.
- › **Antwortzeit zusagen und einhalten.** Ein Workflow, der drei Wochen schweigt, ist schlimmer als keiner. Dann glaubt niemand mehr daran.

Schalter drei: Berechtigungsklassifizierungen pflegen

Die Klassifizierung ist der Teil, der nicht einmalig ist. Neue Dienste bringen neue Scopes, Fachbereiche bringen neue Werkzeuge, und irgendwann steht die Frage im Raum, ob Calendars.ReadWrite nun geringes Risiko darstellt oder nicht. Meine Antwort: Es kommt darauf an, wer im Kalender steht. Bei einer Anwaltskanzlei ist ein Terminkalender ein Mandantenverzeichnis. Bei einem Handwerksbetrieb ist er eine Tourenplanung. Dieselbe Berechtigung, zwei Risikobewertungen.

Genau an dieser Stelle wird Consent-Governance zu einer Aufgabe, die nicht allein in der IT gelöst wird. Wenn du dabei Unterstützung brauchst, ist das der Kern unserer [Microsoft 365 Security Beratung](#): nicht die Schalter umlegen, sondern gemeinsam festlegen, was in deiner Organisation als geringes Risiko gilt und wer diese Frage künftig beantwortet.

Inventar: was in deinem Tenant tatsächlich lebt

Jetzt der unangenehme Teil. Du legst die Schalter um und bist ab morgen sicherer. Aber alles, was vorher passiert ist, steht weiterhin im Verzeichnis. Und in einem Tenant, der ein paar Jahre alt ist, steht dort einiges: Testregistrierungen aus einem Proof of Concept, ein Skript vom Praktikanten, die Integration eines Dienstleisters, dessen Vertrag 2023 endete, und ein Sicherungswerkzeug, das jemand evaluiert und nie wieder angefasst hat. Mit Mail.ReadWrite auf Anwendungsebene.

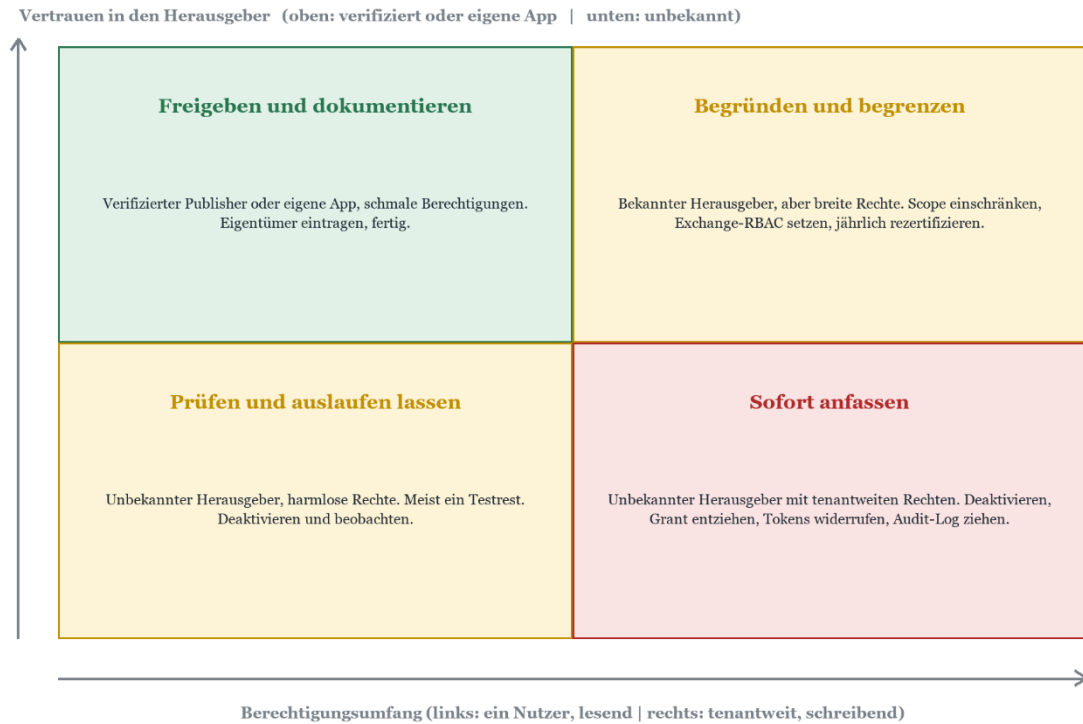
Die erste Bestandsaufnahme

Der Weg über das Portal ist gründlich, aber langsam, weil du pro Nutzer schauen musst; für Administratorzustimmungen zeigt dir das Portal zudem nur die letzten 90 Tage. Für eine vollständige Aufnahme führt der Weg über Microsoft Graph PowerShell und einen Export aller Zustimmungen und Dienstprinzipale in eine Tabelle. Rechne damit, dass der Lauf in einem großen Tenant Stunden dauert — starte ihn abends und schau am nächsten Morgen hinein.

In der Auswertung interessieren dich drei Spalten zuerst: der Zustimmungstyp, die Berechtigung und der Anzeigename. Beim Zustimmungstyp suchst du nach AllPrincipals — das bedeutet, ein Administrator hat für die gesamte Organisation zugestimmt. Bei Microsoft-eigenen Anwendungen ist das normal. Bei jeder anderen Anwendung ist es eine Frage, die beantwortet werden will.

Inventar-Matrix für Enterprise Applications

Jede App landet in genau einem Feld. Das Feld bestimmt die Handlung.



Application Permissions liegen fast immer in der rechten Spalte. Das ist kein Zufall, sondern ihre Bauart.

Skizze 4: Zwei Achsen genügen, um den Berg zu sortieren. Vertrauen in den Herausgeber gegen Berechtigungsumfang — der rote Quadrant kommt zuerst.

Was du prüfst	Die Frage dahinter	Rotes Tuch
Anzeigename	Klingt der Name nach einem Produkt, das es wirklich gibt?	Rechtschreibfehler, generische Namen wie „Test App“, bewusste Nachahmung bekannter Marken
Herausgeber	Verifizierter Herausgeber, eigener Tenant oder unbekannt?	Kein verifizierter Herausgeber bei einer mehrmandantenfähigen App
Zustimmungstyp	AllPrincipals oder Principal?	AllPrincipals bei allem, was nicht von Microsoft stammt
Berechtigungen	Passt der Umfang zum erkennbaren Zweck?	Application Permissions, Schreibrechte, alles mit .All
Redirect-URI	Wohin fließen die Tokens?	Frisch registrierte Domain, Wegwerf-Hoster, IP-Adresse statt Name, localhost in der Produktion
Eigentümer	Wer aus deiner Organisation steht dafür gerade?	Kein Eigentümer — oder ein Konto, das es nicht mehr gibt
Anmeldeinformationen	Secret oder Zertifikat, wie viele, wie alt?	Mehrere Secrets nebeneinander, Laufzeiten über zwei Jahre, nie benutzte Einträge
Letzte Tokenausstellung	Wird die App überhaupt noch benutzt?	90 Tage ohne Token — Entra ID meldet das selbst als Empfehlung

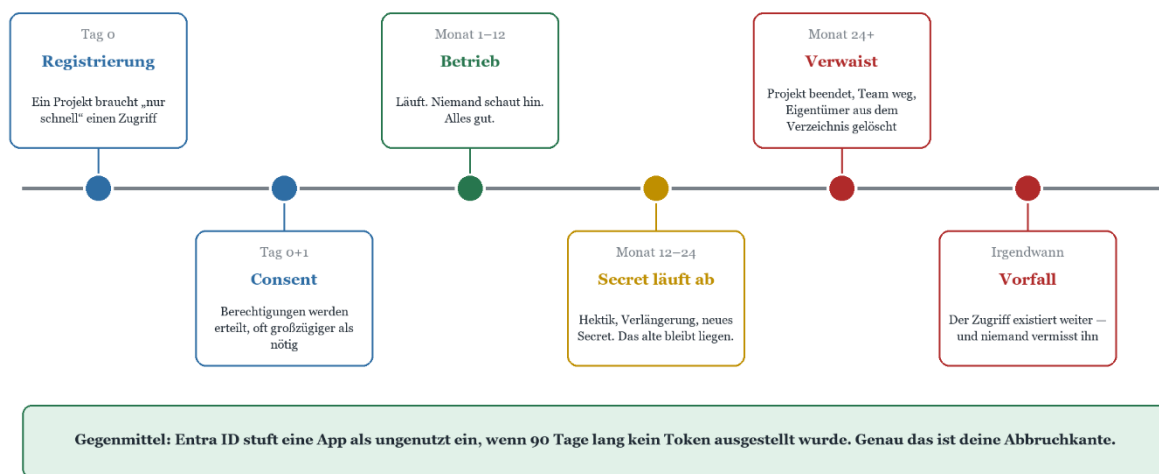
Tabelle 4: Acht Spalten, mit denen du jede Enterprise Application in wenigen Minuten einordnest.

Verwaiste Apps und tote Anmeldeinformationen

Entra ID liefert dir die Vorarbeit frei Haus. Unter *Übersicht* > *Empfehlungen* findest du zwei Einträge, die für dieses Thema gemacht sind: *Nicht verwendete Anwendungen entfernen* und *Nicht verwendete Anmeldeinformationen aus Apps entfernen*. Als ungenutzt gilt eine Clientanwendung, für die in den letzten 90 Tagen kein Token ausgestellt wurde, beziehungsweise eine Ressourcenanwendung, bei der in diesem Zeitraum kein Client ein Token angefordert hat. Du brauchst also nicht einmal eine eigene Auswertung — nur jemanden, der die Liste abarbeitet.

Lebenszyklus einer App-Registrierung

Die meisten Tenants kennen nur die linke Hälfte



Skizze 5: Der Lebenslauf einer App-Registrierung, wie er in der Praxis meistens verläuft. Die rechte Hälfte fehlt in den meisten Prozessbeschreibungen.

WICHTIG Deaktivieren, nicht löschen — jedenfalls nicht sofort

Bei einer verdächtigen Anwendung ist Deaktivieren die richtige erste Handlung, nicht Löschen. Eine gelöschte Anwendung kann zurückkommen, sobald ein anderer Nutzer erneut zustimmt. Eine deaktivierte kann das nicht: Sie bekommt keine neuen Tokens, und niemand kann sich mehr an ihr anmelden oder ihr zustimmen.

Für Karteteileichen ohne Verdachtsmoment gilt derselbe Ablauf aus einem anderen Grund: deaktivieren, 30 bis 60 Tage warten, ob sich jemand meldet, dann löschen. Das erspart dir den Anruf um 23 Uhr, weil die nächtliche Rechnungsverarbeitung seit heute nicht mehr läuft.

Zertifikate statt Secrets — und der Rest der Hygiene

Ein Client Secret ist ein Passwort mit besserer Presse. Es steht im Klartext in irgendeiner Konfigurationsdatei, in einer Pipeline-Variablen, in einem Wiki-Artikel oder — mein Favorit — in einem Teams-Chat mit dem Betreff „kurz für dich“. Es lässt sich kopieren, weitergeben und benutzen, ohne dass irgendjemand es merkt. Und es hat eine Laufzeit, die genau dann endet, wenn niemand Zeit hat.

Die Rangfolge der Anmeldeverfahren

Verfahren	Wo das Geheimnis liegt	Rotation	Bewertung
Client Secret	Im Klartext überall dort, wo die App läuft — und in jeder Kopie davon	Manuell, mit Ausfall bei Versäumnis	Letzte Wahl. Wenn schon, dann mit kurzer Laufzeit.
Zertifikat	Privater Schlüssel im Zertifikatspeicher oder in einem Key Vault	Manuell oder automatisiert, planbar	Deutlich besser: nicht durch Abschreiben kopierbar
Verwaltete Identität	Gar nicht — die Plattform stellt die Tokens aus	Entfällt vollständig	Erste Wahl, wenn die Last in Azure läuft
Föderierte Anmeldeinformation	Beim vertrauenswürdigen Identitätsanbieter, etwa der Build-Pipeline	Entfällt	Erste Wahl für CI/CD und Fremddclouds

Tabelle 5: Vier Wege, wie sich eine Anwendung anmeldet. Nach unten wird es besser.

Die Reihenfolge ist keine Geschmacksfrage. Ein Secret ist ein Wissensgeheimnis und wandert überall dorthin, wo Text hinwandert. Ein Zertifikat ist ein Besitzgeheimnis und bleibt dort, wo der private Schlüssel liegt. Eine verwaltete Identität hat gar kein Geheimnis mehr, das jemand stehlen könnte. Wer das Prinzip vertieft nachvollziehen will, findet dieselbe Logik in der klassischen Verzeichniswelt wieder — dort heißt sie [Microsoft Kerberos](#) und arbeitet seit Jahrzehnten nach demselben Muster.

App Management Policies: Regeln, die das Nachwachsen verhindern

Aufräumen allein reicht nicht, wenn morgen wieder jemand ein Secret mit 24 Monaten Laufzeit anlegt. Dafür gibt es Anwendungsverwaltungsrichtlinien: eine Standardrichtlinie für den ganzen Tenant und Einzelrichtlinien, die bestimmte Anwendungen ausnehmen oder strenger fassen. Beide greifen über das Erstellungsdatum des Objekts — du kannst also für alles Neue sofort scharf schalten und den Bestand in deinem eigenen Tempo nachziehen.

Restriktion	Was sie bewirkt	Sinnvolle Einstellung
passwordAddition	Verbietet neue Client Secrets auf Anwendungen komplett	Aktivieren, sobald deine Entwicklung auf Zertifikate umgestellt ist
passwordLifetime	Begrenzt die maximale Laufzeit neuer Secrets	90 bis 180 Tage. Erzwingt Rotation als Gewohnheit.
customPasswordAddition	Blockiert selbst erzeugte Secrets, die nicht von Entra ID stammen	Aktivieren. Blockiert nebenbei alte PowerShell-Module.
asymmetricKeyLifetime	Begrenzt die Laufzeit von Zertifikaten	12 bis 24 Monate, passend zu deiner PKI
nonDefaultUriAddition	Erlaubt nur Bezeichner-URIs im Standardformat	Aktivieren — verhindert eine ganze Klasse von Verwechslungen
uriAdditionWithoutUniqueTenantIdentifier	Verlangt eine eindeutige Tenantkennung in der Bezeichner-URI	Aktivieren

Tabelle 6: Auswahl aus den Restriktionen der Anwendungsverwaltungsrichtlinien.

FAKTEN Wichtig für die Erwartungshaltung

Weder die Standardrichtlinie noch eine Einzelrichtlinie blockiert die Tokenausstellung für bestehende Anwendungen. Eine App, die den Regeln nicht entspricht, läuft weiter. Blockiert wird nur der Erstellungs- oder Änderungsvorgang, der gegen die Richtlinie verstößt. Die Richtlinie ist also ein

Türsteher für Neuzugänge, kein Rausschmeißer für Stammgäste.

Die geringstprivilegierten Rollen für die Verwaltung dieser Richtlinien sind Anwendungsadministrator und Cloudanwendungsadministrator. Beides sind Rollen, die du wie privilegierte Rollen behandelst — wer Anwendungen verwalten darf, kommt über Umwege an sehr viel mehr.

Zwei Details, die selten jemand setzt

App Instance Property Lock. Wenn deine Organisation eine mehrmandantenfähige Anwendung veröffentlicht, kann ein Administrator im Kundenantrag an den dortigen Dienstprinzipal eigene Anmeldeinformationen hängen. Das ist selten gewollt und meistens gefährlich. Die Instanzsperrung verhindert genau das, indem sie sensible Eigenschaften — Anmeldeinformationen mit dem Verwendungstyp Sign und Verify sowie den Tokenverschlüsselungsschlüssel — nach der Bereitstellung gegen Änderung sperrt. Die Empfehlung lautet, alle verfügbaren sensiblen Eigenschaften zu sperren, nicht nur die, an die man zuerst denkt.

RBAC für Anwendungen in Exchange Online. Wenn eine Anwendung wirklich Postfachzugriff auf Anwendungsebene braucht — Archivierung, Signaturen, Journaling —, dann gib ihr nicht den gesamten Tenant. Über die rollenbasierte Zugriffssteuerung für Anwendungen weist du der Anwendung eine Rolle mit einem Ressourcenbereich zu, der sich auf bestimmte Postfächer oder eine Gruppe beschränkt. Die älteren Anwendungszugriffsrichtlinien erfüllen denselben Zweck, gelten aber als Vorgängermodell und sollten für Neueinrichtungen nicht mehr verwendet werden.

Betrieb: Erkennung, Reaktion und ein Kalender, der hält

Governance ohne Betrieb ist ein Dokument. Damit die Sache trägt, brauchst du zwei Dinge: eine Erkennung, die anschlägt, wenn etwas Ungewöhnliches zugestimmt wird, und einen Rhythmus, in dem jemand hinschaut, auch wenn nichts anschlägt.

Woran du einen Vorfall erkennst

Im einheitlichen Überwachungsprotokoll suchst du nach der Aktivität *Zustimmung zu einer Anwendung erteilt*. Interessant ist dort das Feld `IsAdminConsent`: Steht es auf `True`, wurde tenantweit zugestimmt. Wenn du das nicht erwartet hast, ist der Vorfall damit bereits bestätigt. Rechne mit einer Verzögerung von 30 Minuten bis 24 Stunden, bis der Eintrag durchsuchbar ist. Kommt eine Zustimmungsanfrage über die risikobasierte Eskalation nicht durch, findest du im Verzeichnisprotokoll die Kategorie `ApplicationManagement` mit dem Statusgrund `Risky application detected`.

Setz dir dafür eine eigene Benachrichtigung auf — wie das sauber geht, steht in [Alert Policies und Benachrichtigungen richtig aufsetzen](#). Wenn du [Defender for Cloud Apps im Mittelstand](#) im Einsatz hast, bekommst du zusätzlich die App-Governance: Richtlinien für OAuth-Anwendungen, Anomalieerkennung und eine laufende Bewertung des Verhaltens von Drittanbieteranwendungen. Für dieses Handlungsfeld ist das die mit Abstand nützlichste Erweiterung, die du kaufen kannst. Wenn Zustimmungsergebnisse zusätzlich in einer zentralen Korrelation landen sollen, lohnt der Blick auf [Microsoft Sentinel anbinden: wann es sich lohnt](#).

WARNUNG Was du im Ernstfall in welcher Reihenfolge tust

- › **1. Anwendung deaktivieren.** Keine neuen Tokens, keine weitere Zustimmung.
- › **2. Zustimmungen entziehen.** Alle OAuth-Grants der Anwendung entfernen, nutzerbezogen und tenantweit.
- › **3. Refresh Tokens widerrufen.** Für jedes betroffene Konto, sonst lebt der Zugriff weiter.
- › **4. Postfachregeln prüfen.** Weiterleitungen und Regeln, die Nachrichten mit bestimmten Stichwörtern in den Papierkorb schieben, sind die klassische Zweitspur.
- › **5. Umfang bestimmen.** Das Audit-Log auf den Zeitraum zwischen Zustimmung und Entdeckung auswerten. Ohne aktivierte Protokollierung geht das nicht — dann bleibt nur die Annahme des schlimmsten Falls.
- › **6. Melden.** Die verdächtige Anwendung an Microsoft melden, damit sie tenantübergreifend deaktiviert wird.

Der vollständige Ablauf mit Zuständigkeiten und Zeitfenstern steht in [Phishing-Vorfall: die ersten 60 Minuten](#) und im [Incident-Response-Plan für Microsoft 365](#).

Der Governance-Kalender

Rhythmus	Aufgabe	Zuständig
Laufend	Admin-Consent-Anfragen bearbeiten, zugesagte Antwortzeit einhalten	Prüfergruppe, mindestens zwei Personen
Wöchentlich	Neue Dienstprinzipale und neue Zustimmungen durchsehen	IT-Betrieb
Monatlich	Entra-Empfehlungen zu ungenutzten Anwendungen und Anmeldeinformationen abarbeiten	IT-Betrieb
Quartalsweise	Alle Anwendungen mit Application Permissions rezertifizieren — der Eigentümer bestätigt schriftlich den Bedarf	Anwendungseigentümer, IT-Security
Quartalsweise	Ablaufende Secrets und Zertifikate der nächsten 90 Tage vorziehen	IT-Betrieb
Jährlich	Berechtigungsklassifizierungen überprüfen, Richtlinien nachschärfen	IT-Security mit den Fachbereichen
Jährlich	Den Ablauf im Trockenen üben: verdächtige App abschalten, Grant entziehen, kommunizieren	IT-Security, Kommunikation

Tabelle 7: Ein Rhythmus, der sich in mittelständischen Organisationen bewährt hat.

Der letzte Punkt ist kein Beiwerk. Der Unterschied zwischen einem Vorfall, der zwei Stunden dauert, und einem, der zwei Wochen dauert, liegt fast nie in der Technik, sondern darin, ob die Beteiligten den Ablauf schon einmal gemacht haben. Wie du so etwas aufsetzt, beschreibt [Feuerwehübung: Security-Notfälle im Team trainieren](#). Und weil Consent-Phishing eben nicht an fehlender Technik scheitert, sondern an einem Klick, gehört das Thema in jede Awareness-Runde — passende Szenarien liefert [Attack Simulation Training: Phishing-Übungen richtig aufsetzen](#).

WEITER Passt thematisch dazu

- › [Tenant-Härtung: die 20 wichtigsten Schrauben](#) — die Consent-Einstellungen gehören dort zu den Schrauben mit der größten Wirkung
- › [Defender for Cloud Apps im Mittelstand](#) — App-Governance, OAuth-Richtlinien und Anomalieerkennung
- › [Kompromittiertes Konto erkennen und aufräumen](#) — warum der Passwort-Reset allein nicht genügt

- › [Conditional Access: Baselines, die nicht nerven](#) — inklusive der Regeln für Workload Identities
- › [Zero Trust im Mittelstand: pragmatisch statt dogmatisch](#) — Anwendungsidentitäten sind der am häufigsten vergessene Teil davon
- › [NIS2 aus Security-Sicht: technische Maßnahmen in M365](#) — Zugriffskontrolle und Lieferkette betreffen App-Registrierungen unmittelbar
- › [Microsoft 365 Security Schulung](#) — wenn dein Team den Consent-Dialog künftig lesen soll, statt ihn wegzuklicken

Häufige Fragen

Wenn ich die Benutzerzustimmung abschalte, bricht dann etwas weg?

Bestehende Zustimmungen bleiben unangetastet — die Einstellung wirkt nur auf künftige Vorgänge. Was sich ändert: Anwendungen, die ein Nutzer bisher selbst freischalten konnte, brauchen ab sofort eine Freigabe. Deshalb schaltest du niemals nur ab, sondern immer zusammen mit dem Admin-Consent-Workflow und einer zugesagten Antwortzeit. Rechne in den ersten zwei Wochen mit einem Anfragenüberhang, danach läuft es sich ein.

Reicht die mittlere Option wirklich, oder soll ich Zustimmung komplett verbieten?

Für die meisten mittelständischen Organisationen reicht die mittlere Option — verifizierte Herausgeber oder eigene Anwendungen, nur mit als gering eingestuften Berechtigungen. Sie fängt das Massengeschäft des Consent-Phishings ab und lässt gleichzeitig die alltäglichen Anmeldungen durch. Ein vollständiges Verbot lohnt dort, wo die Datenlage es verlangt: Gesundheitswesen, Rechtsberatung, Behörden, Rüstungszulieferer. Der Preis ist zusätzlicher Aufwand bei den Prüfern, und den musst du personell auch decken können.

Was ist der Unterschied zwischen App-Registrierung und Enterprise Application?

Die App-Registrierung ist die Definition der Anwendung in dem Tenant, in dem sie entwickelt wurde: Name, Berechtigungen, Redirect-URIs, Anmeldeinformationen. Die Enterprise Application ist die lokale Instanz dieser Anwendung in deinem Verzeichnis — technisch der Dienstprinzipal. Bei einer selbst entwickelten Anwendung hast du beides. Bei einem Fremdprodukt hast du nur den Dienstprinzipal, die Registrierung liegt beim Hersteller. Für die Governance ist die Enterprise Application der Ort, an dem du Zustimmungen siehst, Nutzer zuweist und die Anwendung deaktivierst.

Hilft Conditional Access gegen Consent-Phishing?

Nur mittelbar. Der Zustimmungsvorgang selbst läuft über eine reguläre Anmeldung, auf die deine Richtlinien greifen — der Nutzer muss sich also anmelden, mit MFA und allem, was du eingestellt hast. Aber er ist ja der berechtigte Nutzer, er besteht die Prüfung, und danach klickt er auf Zustimmung. Der anschließende Zugriff der Anwendung läuft nicht mehr über seine Anmeldung. Für Anwendungsidentitäten gibt es Conditional Access für Workload Identities, mit dem du beispielsweise die Herkunftsstandorte für Dienstprinzipale einschränken kannst. Das ist wirksam, aber ein eigenes Werkzeug mit eigener Lizenzfrage.

Wie erkenne ich, ob eine Anwendungsberechtigung tatsächlich gebraucht wird?

Über die Nutzung, nicht über die Beschreibung. Die Anmeldeprotokolle für Dienstprinzipale zeigen dir, welche Anwendung wann Tokens angefordert hat. Die Entra-Empfehlung zu ungenutzten Anwendungen setzt genau dort an und verwendet 90 Tage ohne Tokenausstellung als Schwelle. Wenn eine Anwendung mit kritischen Berechtigungen seit einem Quartal schweigt, brauchst du keine weitere Diskussion: deaktivieren, warten, ob sich jemand meldet, dann löschen.

Können wir Zertifikate erzwingen, oder ist das nur eine Empfehlung?

Erzwingen geht. Die Anwendungsverwaltungsrichtlinien kennen eine Restriktion, die das Hinzufügen neuer Kennwortgeheimnisse auf Anwendungen komplett blockiert, und eine zweite, die selbst erzeugte Geheimnisse verbietet. In Verbindung mit dem Erstellungsdatum kannst du das für alles Neue sofort aktivieren, ohne den Bestand lahmzulegen. Wichtig ist nur die Vorwarnung an die Entwicklung: Blockiert wird der Anlagevorgang, und der scheitert dann mit einer Fehlermeldung, die ohne Kontext wie ein Plattformfehler aussieht.

Wir haben einen Dienstleister mit weitreichendem Zugriff. Wie gehen wir damit um?

Erstens: ein dokumentierter Eigentümer auf eurer Seite, nicht auf der des Dienstleisters. Zweitens: Zugriff so eng wie möglich schneiden — bei Postfachzugriff über RBAC für Anwendungen in Exchange Online auf die tatsächlich benötigten Postfächer. Drittens: eigene Anmeldeinformationen je Dienstleister, niemals geteilte. Viertens: eine Rezertifizierung im Quartal, die schriftlich bestätigt, dass der Zugriff noch gebraucht wird. Und fünftens: ein Punkt im Vertrag, der die unverzügliche Meldung eines Sicherheitsvorfalls beim Dienstleister regelt. Wie schnell aus dem Zugang eines Dritten ein eigener Vorfall wird, zeigt das Dossier [TerminalFix und der Berlin-Hack](#).

Was hat Consent-Governance mit Datenklassifizierung zu tun?

Mehr, als man denkt. Die Frage, ob eine Berechtigung geringes Risiko darstellt, hängt daran, was hinter der Berechtigung liegt. Zugriff auf eine Ablage mit Kantinenplänen ist etwas anderes als Zugriff auf eine Ablage mit Personalakten — technisch aber dieselbe Berechtigung. Wenn deine Daten klassifiziert und mit Vertraulichkeitsbezeichnungen versehen sind, kannst du diese Frage begründet beantworten, statt zu raten. Der Werkzeugkasten dafür ist im [Kompetenzbereich Microsoft Purview](#) beschrieben.

Wie oft sollten wir das Inventar tatsächlich anfassen?

Neuzugänge wöchentlich, weil dort die Angriffe sichtbar werden. Die Entra-Empfehlungen monatlich, weil sie sich sonst zu einem Berg auftürmen, den niemand mehr anfasst. Anwendungen mit Application Permissions quartalsweise, weil dort der Schaden am größten wäre. Alles Übrige jährlich. Und wenn du nur eine dieser Frequenzen durchhalten kannst, nimm die quartalsweise Rezertifizierung der Anwendungsberechtigungen. Der Rest ist Komfort, dieser Punkt ist Substanz.

Fazit

Consent-Governance ist eine der wenigen Sicherheitsmaßnahmen, bei denen der Aufwand ehrlich klein und der Effekt ehrlich groß ist. Drei Einstellungen im Portal, eine Prüfergruppe, ein Inventar und ein Rhythmus — mehr braucht die Grundausstattung nicht. Dafür schließt du eine Tür, die

sonst offen steht und durch die Angreifer gehen, ohne auch nur an deinem Passwortschutz zu kratzen.

Die Reihenfolge, wenn du morgen anfängst: Erst den Admin-Consent-Workflow aktivieren und Prüfer benennen, damit ein Weg existiert. Dann die Berechtigungsklassifizierung mit den sechs Anmeldeberechtigungen füllen. Dann die Benutzerzustimmung auf verifizierte Herausgeber und geringe Berechtigungen umstellen. Dann das Inventar ziehen und den roten Quadranten der Matrix abarbeiten. Und erst danach die Feinarbeit: Zertifikate, Anwendungsverwaltungsrichtlinien, Rezertifizierung.

Was du dabei nicht vergessen darfst: Der Schalter schützt die Zukunft. Alles, was in den letzten Jahren zugestimmt wurde, steht weiterhin im Verzeichnis und wartet geduldig. Die App, die niemand mehr kennt, deren Eigentümer seit zwei Jahren aus dem Verzeichnis gelöscht ist und die Mail.ReadWrite auf Anwendungsebene besitzt — die ist nicht harmlos, weil sie alt ist. Sie ist gefährlich, weil sie alt ist. Alter schützt in dieser Disziplin vor gar nichts, außer vor Aufmerksamkeit.