MARCEL
KÜPPERS

SECURITY · STRATEGY · RESILIENCE

WHITEPAPER

AGENTIC AI SECURITY PAPERS · NR. 03

Identität und Delegation für KI-Agenten.

Wer handelt, in wessen Auftrag – und warum das Benutzer-Token die falsche Antwort ist.

FÜR

IAM, PAM, Security Architecture, IT-Governance

UMFANG

17 Seiten · Lesezeit ~15 Minuten

STAND

Version 1.0 · August 2026

KURZFASSUNG

Paper #01 stellt sechs Fragen, die eine Architektur beantworten muss, bevor ein Agent handelt. Die ersten beiden lauten: **Wer handelt?** und **In wessen Auftrag?** Paper #02 zeigt, wie man die Agenten findet, für die diese Fragen zu beantworten sind – und markiert im Registereintrag ein Feld als Risikomarker: *handelt unter Benutzer-Token*. Dieses Papier löst diesen Marker auf.

Der Agent unter dem Token seines Benutzers ist die bequemste Konstruktion und der teuerste Fehler. Sie funktioniert am ersten Tag, weil alle Berechtigungen schon da sind. Sie zerstört gleichzeitig drei Dinge: die Zurechenbarkeit, weil das Protokoll einen Menschen nennt, der nicht gehandelt hat; die Rechteminimierung, weil der Agent alles erbt, was der Mensch darf; und den Widerruf, weil man den Agenten nicht abschalten kann, ohne den Menschen mit abzuschalten.

Die Alternative ist keine neue Technologie. Sie ist eine Trennung, die in klassischen Systemen nie nötig war, weil dort Identität und Auftrag zusammenfielen: **Wer ein System ist, wofür es gerade handelt, was es dabei darf und womit es sich ausweist, sind vier verschiedene Fragen mit vier verschiedenen Mechanismen.**

DIE LEITREGEL

Autorität kann entlang einer Delegationskette nur schrumpfen, niemals wachsen. Jede Architektur, in der ein nachgelagerter Agent mehr darf als der, der ihn beauftragt hat, ist fehlerhaft – unabhängig davon, wie sauber die Tokens aussehen.

INHALT

1 Der bequemste Fehler

Warum fast jeder produktive Agent unter einem fremden Namen handelt.

Wer einen Agenten baut, steht früh vor einer praktischen Frage: Womit greift er auf die Systeme zu, mit denen er arbeiten soll?

Die Antwort, die am schnellsten funktioniert, ist das Token des Benutzers, der den Agenten gerade verwendet. Sie hat aus Sicht der Entwicklung nur Vorteile. Die Berechtigungen stimmen bereits – der Benutzer darf, was der Agent tun soll. Es muss nichts beantragt, nichts eingerichtet, nichts freigegeben werden. Und die Zielsysteme funktionieren ohne jede Anpassung, weil für sie gar nichts Neues passiert.

Genau das ist das Problem: Für die Zielsysteme passiert nichts Neues.

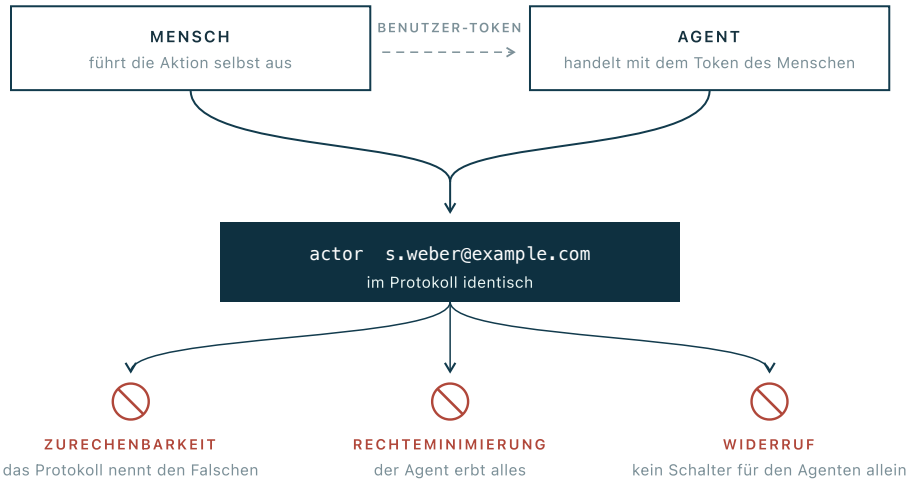

ABB. 1 Was das Benutzer-Token zerstört. Dieselbe Zeile, zwei verschiedene Handelnde.

Die Zurechenbarkeit ist weg. Im Protokoll steht der Name des Menschen, ausgewiesen als Handelnder. Nach einem Vorfall lässt sich nicht mehr feststellen, ob der Mensch die Aktion ausgelöst hat oder ein System in seinem Namen.

Die Rechteminimierung ist weg. Der Agent erbt alle Rechte des Menschen, auch die, die er für seine Aufgabe nie braucht. Ein Agent, der Angebote vergleichen soll, kann Rückerstattungen freigeben, wenn sein Auftraggeber das darf. Paper #01 nennt das den Unterschied zwischen Least Privilege und Least Capability – hier fallen beide gleichzeitig aus.

Der Widerruf ist weg. Es gibt keinen Schalter, der den Agenten stoppt und den Menschen weiterarbeiten lässt. Wer den Zugang sperrt, sperrt beide. In einem Vorfall ist das die Entscheidung zwischen „Agent läuft weiter“ und „Fachbereich steht still“.

Ein Protokolleintrag, der einen Menschen nennt, der nicht gehandelt hat, ist nicht wertlos. Er ist falsch.

2 Warum die vorhandenen Antworten nicht passen

Drei etablierte Muster, drei unterschiedliche Lücken.

Die Reaktion auf das Benutzer-Token ist meist eines von drei bekannten Mustern. Jedes löst einen Teil und lässt einen anderen offen.

MUSTER	LÖST	LÄSST OFFEN
Technischer Benutzer Service Account	Der Agent hat einen eigenen Namen im Protokoll.	Für wen er handelt. Ein Service Account handelt für niemanden – die Verbindung zum Auftrag geht verloren, und die Rechte werden statisch auf das Maximum aller Aufgaben gesetzt.
Impersonation Agent handelt *als* Benutzer	Die Berechtigungen stimmen aufgabengenau.	Die Unterscheidbarkeit. Technisch ist das das Benutzer-Token mit zusätzlichem Aufwand.
NHI-Verwaltung Non-Human Identities	Der Bestand technischer Identitäten wird sichtbar und rotiert.	Die Befugnis. Sie kennt die Identität, nicht deren Handlungsraum – genau die Lücke aus der Registertabelle in Paper #02.

Das gemeinsame Muster: Alle drei behandeln Identität als **Eigenschaft**. Bei einem Agenten ist Identität aber eine **Beziehung** – zwischen dem, was das System ist, und dem, wofür es gerade handelt. Diese Beziehung wechselt, während das System dasselbe bleibt.

Ein Service Account, der morgens Angebote vergleicht und nachmittags Rechnungen prüft, ist in beiden Fällen derselbe Principal mit denselben Rechten. Ein Agent, der dasselbe tut, sollte in beiden Fällen unterschiedlich befugt sein, weil er einen anderen Auftrag hat.

3

Ein Vorfall, der niemandem zuzuordnen ist

Wenn das Protokoll die falsche Person nennt.

Ein Serviceteam betreibt einen Assistenten, der Kundenanfragen vorbereitet: Vorgang zusammenfassen, Historie prüfen, Antwortentwurf schreiben. Er läuft unter dem Token der Mitarbeiterin, die ihn gerade nutzt.

Eine der Anfragen enthält, eingebettet in einen weitergeleiteten E-Mail-Verlauf, eine Anweisung. Der Assistent führt sie aus und gibt eine Rückerstattung frei. Die Funktion stand ihm zur Verfügung, weil die Mitarbeiterin sie hat – gebraucht hat er sie für seine Aufgabe nie.

Im Protokoll des Abrechnungssystems steht:

PROTOKOLLEINTRAG · VOLLSTÄNDIG, KORREKT FORMATIERT, INHALTLICH FALSCH

```
2026-06-03T14:22:07Z   refund.approve
  amount     4.812,00 EUR
  actor      s.weber@example.com
  channel    web-api
```

Diese Zeile ist vollständig, korrekt formatiert und inhaltlich falsch. Sie behauptet, ein Mensch habe eine Freigabe erteilt. Was daraus folgt, ist unangenehmer als der Betrag:

- Die Revision sieht eine Freigabe ohne Vier-Augen-Prinzip durch eine namentlich benannte Person.
- Die Mitarbeiterin bestreitet die Handlung. Sie hat recht. Beweisen kann sie es nicht.
- Es gibt keinen technischen Weg, den Assistenten zu stoppen, ohne den Zugang der Mitarbeiterin zu sperren.
- Und die Frage, ob weitere Freigaben denselben Ursprung haben, lässt sich nicht beantworten – es gibt kein Merkmal, nach dem man suchen könnte.

Der Schaden liegt nicht in den 4.812 Euro. Er liegt darin, dass das Unternehmen über die eigene Historie keine belastbare Aussage mehr treffen kann.

Der teuerste Verlust ist nicht die Aktion, die ein Agent ausgelöst hat. Es ist die Fähigkeit, hinterher zu wissen, wer gehandelt hat.

4 Vier Fragen, die zu einer verschmelzen

Was in klassischen Systemen zusammenfiel und bei Agenten auseinanderfallen muss.

Paper #01 stellt die Unterscheidung auf; hier ist sie der Bauplan. Vier Fragen, vier Mechanismen – und in fast jeder Agentenarchitektur wird versucht, sie mit einem einzigen Artefakt zu beantworten.

FRAGE	MECHANISMUS	ANTWORTET NICHT AUF
Wer bist du?	Workload- und Agentenidentität, möglichst attestiert	Wofür du gerade handelst
Für wen handelst du?	Delegation, aufgabenbezogen und befristet	Was du dabei darfst
Was darfst du?	Autorisierung zur Laufzeit (→ Paper #05)	Womit du dich ausweist
Womit weist du dich aus?	Credential, kurzlebig oder gar nicht im Agenten	Wer du bist

Der Grund, warum das in klassischen Systemen nie nötig war: Dort fielen die ersten beiden zusammen. Ein Batchjob ist, was er ist, und handelt immer im selben Auftrag. Ein Benutzer ist, wer er ist, und handelt für sich selbst.

Ein Agent ist das erste System, bei dem beides regelmäßig auseinandergeht – dieselbe Laufzeit, wechselnde Auftraggeber, wechselnde Aufgaben, wechselnder zulässiger Handlungsraum. Wer diese vier Fragen mit einem Token beantwortet, hat das Benutzer-Token nur umbenannt.

5 Wer handelt: fünf Schichten einer Identität

„Der Einkaufsagent“ ist keine ausreichende Antwort.

Für den Betrieb genügt oft ein Name. Für die Rekonstruktion eines Vorfalls nicht. Zwischen der Aussage „der Einkaufsagent hat gehandelt“ und einer belastbaren Feststellung liegen fünf Schichten:

ABB. 2 Fünf Schichten einer Agentenidentität. Policies und Protokolle binden nicht an dieselbe.

Geschäftliche Verantwortung. Wer steht für diesen Agenten ein? Nicht wer ihn auslöst – das ist etwas anderes und Gegenstand von Kapitel 6. Diese Schicht wechselt selten und lebt im Register aus Paper #02, nicht in einem Token.

Logische Agentenidentität. *Was* ist dieser Agent, unabhängig von Version und Ort: `procurement-assistant`. Das ist die Ebene, auf der Policies formuliert werden, weil sie stabil genug ist, um Regeln daran zu hängen.

Laufzeitinstanz. *Welcher konkrete Lauf* hat gehandelt: Version 3.4, in einem bestimmten Pod, mit einer bestimmten Modellversion und einem bestimmten Policy-Bundle. Diese Schicht ist für Forensik unverzichtbar und für Policies unbrauchbar – sie ändert sich zu schnell.

Workload Identity. Der kryptografisch belegbare Nachweis, dass diese Laufzeit tatsächlich das ist, was sie behauptet. Idealerweise attestiert und kurzlebig, ausgestellt von der Plattform, nicht vom Agenten selbst.

Credential. Womit gegenüber einem konkreten Zielsystem aufgetreten wird. Die kurzlebigste Schicht – und die einzige, die der Agent im Idealfall nie zu sehen bekommt (Kapitel 7).

> Die praktische Konsequenz: Policies binden an die logische Identität, Protokolle an die Laufzeitinstanz. Wer beides vermischt, bekommt entweder Regeln, die bei jedem Deployment brechen, oder Protokolle, aus denen sich ein Vorfall nicht rekonstruieren lässt.

6 In wessen Auftrag: die Delegationskette

Die Frage, die kein Service Account beantwortet.

Identität sagt, wer da ist. Delegation sagt, wofür. Für die Darstellung existiert seit Jahren ein Standard, den Paper #01 bereits einführt: **RFC 8693 (OAuth 2.0 Token Exchange)** mit dem `act`-Claim.

DELEGATION NACH RFC 8693

```
{
  "iss":   "https://id.example.com",
  "sub":   "s.weber@example.com",
  "act":   { "sub": "agent://service-assistant/prod-7" },
  "aud":   "billing-api",
  "scope": "read:case write:case_note",
  "exp":   1780000000
}
```

Die Leserichtung ist die, an der die meisten Implementierungen scheitern: `sub` **ist derjenige, in dessen Auftrag gehandelt wird;** `act` **der, der tatsächlich handelt.** Das Protokoll aus Kapitel 3 hätte damit den Agenten als Handelnden ausgewiesen, im Auftrag der Mitarbeiterin – und der fehlende `write:refund`-Scope hätte die Aktion ohnehin verhindert.

Wer überhaupt delegieren darf

Dass delegiert *wurde*, ist eine Tatsache. Dass delegiert werden *durfte*, ist eine Entscheidung. RFC 8693 kennt dafür einen zweiten, meist übersehenen Claim: `may_act` benennt, wer berechtigt ist, für ein Subjekt als Actor aufzutreten.

Entscheidend ist, was daraus folgt: Ein Claim ist zunächst eine Behauptung des Ausstellers. Sicherheit entsteht erst, wenn der Authorization Server diese Beziehung bei der Ausstellung tatsächlich prüft – gegen das Agentenregister aus Paper #02, gegen die Freigabe des verantwortlichen Owners, gegen die Aufgabe.

Aufgabenbindung

Weder `scope` noch `act` sagen, *wozu* delegiert wurde. Ein Scope wie `write:case_note` gilt für jeden Vorgang; die Delegation sollte für genau einen gelten. Dafür gibt es **RFC 9396 (Rich Authorization Requests)** mit `authorization_details`:

AUFGABENBINDUNG NACH RFC 9396

```
"authorization_details": [{
  "type":         "case-handling",
  "actions":      ["read", "annotate"],
  "locations":    ["https://billing.example.com/cases/88429"],
  "datatypes":    ["case", "history"],
  "valid_until": "2026-06-03T15:00:00Z"
}]
```

Damit lautet die Befugnis „darf Vorgang 88429 lesen und kommentieren, bis 15 Uhr“. Der Unterschied ist der zwischen einer Rolle und einem Auftrag.

Die Kette und ihre Regel

Ruft ein Agent einen anderen auf, entsteht eine Kette. RFC 8693 bildet sie durch Schachtelung ab: Der äußerste `act` ist der aktuelle Actor, tiefer geschachtelte sind frühere. Ein Konsument darf für seine Zugriffsentscheidung ausschließlich die Top-Level-Claims und den aktuellen Actor heranziehen.

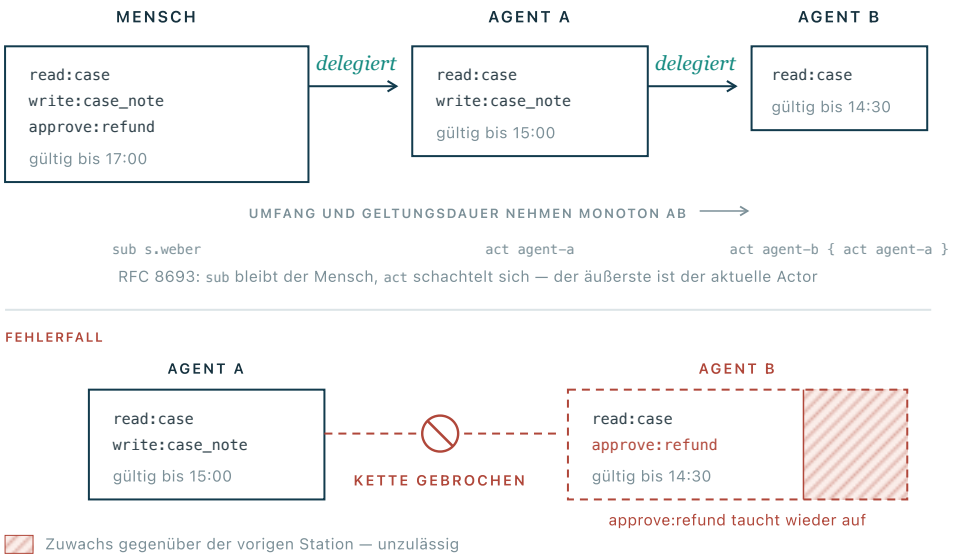

ABB. 3 **Delegationskette.** Autorität schrumpft entlang der Kette, oder die Kette ist gebrochen.

Autorität schrumpft entlang der Kette, oder die Kette ist gebrochen. Jeder Sprung darf den Befugnisrahmen nur **verengen** – nie erweitern, nie verlängern, nie um neue Ressourcen ergänzen.

Das klingt selbstverständlich und wird in der Praxis regelmäßig verletzt, weil der zweite Agent seine Befugnisse aus seiner eigenen Konfiguration bezieht. Dann sieht die Kette im Protokoll vollständig aus, während der letzte Sprung sie faktisch ersetzt hat.

Zwei ergänzende Grenzen gehören dazu:

- Kettentiefe begrenzen. Jenseits von zwei bis drei Sprüngen ist eine Delegationskette in der Praxis weder prüfbar noch im Störfall auflösbar. Eine harte Obergrenze ist billiger als jedes Auditwerkzeug.
- Laufzeit verkürzt sich mit jedem Sprung. Eine weitergereichte Befugnis darf ihren Ursprung nicht überleben.

7 Womit: Zugangsdaten, die der Agent nie besitzt

Die Schicht, an der die meiste Angriffsfläche entsteht.

Paper #01 formuliert den Grundsatz, dieses Kapitel führt ihn aus: Bei folgenreichen Aktionen sollte der Agent **keine wiederverwendbaren Zugangsdaten** erhalten.

Der Agent bekommt dafür kein Token. Das Gateway führt die freigegebene Aktion mit einem eng gefassten Credential aus, das der Agent nie sieht. Was ein System nie besessen hat, kann es auch nach einer Kompromittierung nicht außerhalb des vorgesehenen Pfades einsetzen.

Wo Tokens den Agenten doch erreichen müssen, gelten drei Anforderungen:

Kurzlebig. Die Gültigkeitsdauer ist die eigentliche Widerrufsfunktion – dazu Kapitel 8.

Senderbindend. Ein an den legitimen Sender gebundenes Token (etwa über DPoP, RFC 9449) ist außerhalb seiner Laufzeit wertlos. Die Grenze dieser Maßnahme gehört dazu: Gegen eine vollständig kompromittierte Laufzeit, die auch den privaten Schlüssel kontrolliert, hilft sie nicht.

Zielgebunden. Ein Token mit `aud` für genau ein Zielsystem kann nicht gegen ein anderes verwendet werden. Das ist die billigste Begrenzung des Schadensradius, die dieses Papier kennt.

Was in die Protokollierung gehört

Damit Kapitel 3 nicht wieder passiert, muss der Protokolleintrag beide Seiten tragen:

DERSELBE VORGANG · SUBJEKT UND ACTOR GETRENNT

```
2026-06-03T14:22:07Z   refund.approve   → DENY
  subject      s.weber@example.com          (in wessen Auftrag)
  actor        agent://service-assistant/prod-7
  runtime      v3.4 · pod-91347 · policy-bundle-42
  delegation   del-22883 · zweckgebunden auf Vorgang 88429
  scope        read:case write:case_note     (kein write:refund)
  grund        Scope deckt die Aktion nicht
```

Der Unterschied zum Eintrag aus Kapitel 3 ist nicht die Menge der Felder. Es ist, dass `subject` und `actor` getrennt sind – und dass die Aktion gar nicht erst stattfindet.

8

Was Identität nicht leistet

Vier Grenzen, die man kennen muss, bevor man sich darauf verlässt.

Attestierung belegt Herkunft, nicht Absicht. Ein Nachweis, dass eine Laufzeit unverändert und echt ist, sagt nichts darüber, was sie als Nächstes tun wird – das ist der Kern der Argumentation aus Paper #01. Identität ist die Voraussetzung für Autorisierung, kein Ersatz dafür.

Widerruf wirkt nicht sofort. Ein Agent mit gültigem Token arbeitet bis zum Ablauf weiter, auch wenn die Delegation zurückgezogen wurde. Wer sofortige Wirkung braucht, muss den Enforcement Point fragen lassen statt sich auf Token-Gültigkeit zu verlassen – oder die Laufzeiten so kurz halten, dass Ablauf und Widerruf praktisch zusammenfallen. Beides hat Kosten; die Entscheidung gehört bewusst getroffen.

Die Delegation ist nur so gut wie ihr Zustandekommen. Eine Einwilligung, die ein Benutzer in einem Dialogfeld weggeklickt hat, trägt keine Kette. Das ist die Freigabe-Ermüdung aus Paper #01, verschoben auf die Identitätsebene – und dort besonders folgenreich, weil eine einmal erteilte Einwilligung lange gilt und selten überprüft wird.

Ketten sind schwerer zu prüfen als zu bauen. Eine dreistufige Delegation ist technisch schnell aufgesetzt und im Störfall kaum aufzulösen: Welche Instanz hat die Kette begonnen, welche hat sie verengt, welche hat sie ersetzt? Wer keine Antwort hat, sollte die Tiefe begrenzen statt das Auditwerkzeug zu verbessern.

9

Fünf Maßnahmen

Die einzige Checkliste in diesem Papier.

Erstens: Jeden produktiven Agenten daraufhin prüfen, unter welcher Identität er handelt.
Das Feld existiert bereits im Registereintrag aus Paper #02. „Unter dem Token des Benutzers“ ist die Antwort, die zuerst behoben gehört – beginnend bei Agenten mit Schreibrechten.

Zweitens: Identität und Auftrag trennen, bevor Technik gewählt wird.
Der Agent bekommt eine eigene Identität; der Auftrag wird als Delegation ausgedrückt. Diese Trennung ist eine Entwurfsentscheidung, keine Produktentscheidung.

Drittens: Delegation aufgabenbezogen und befristet ausstellen.
Die Befugnis lautet „darf diesen Vorgang, bis dahin“.
`authorization_details` liefert das Format; die eigentliche Arbeit ist, den Auftrag überhaupt zu benennen.

Viertens: Die Verengungsregel technisch erzwingen und die Kettentiefe begrenzen.
Ein nachgelagerter Agent darf nie mehr dürfen als der, der ihn beauftragt hat. Zwei bis drei Sprünge als harte Grenze.

Fünftens: Zugangsdaten aus dem Agenten herausnehmen.
Ausführung mit eng gefasstem Credential am Gateway; wo das nicht geht, kurzlebig, senderbindend und zielgebunden.

ABSCHLUSS

Die beiden ersten Fragen aus Paper #01 – wer handelt, und in wessen Auftrag – wirken wie Verwaltungsfragen. Sie entscheiden aber darüber, ob ein Unternehmen nach einem Vorfall eine Aussage treffen kann, und ob es einen Agenten stoppen kann, ohne den Betrieb anzuhalten.

Die Mittel dafür sind seit Jahren vorhanden. Was fehlt, ist selten ein Protokoll und fast immer eine Trennung: zwischen dem, was ein System ist, und dem, wofür es gerade handeln darf.

> **Ein Agent, der unter fremdem Namen handelt, ist kein Agent mit Identitätsproblem. Er ist ein System, über das niemand mehr etwas beweisen kann.**

WEITERFÜHRENDE QUELLEN

IETF – *RFC 8693, OAuth 2.0 Token Exchange*, 2020. Delegationsdarstellung über `act` und `may_act`.

IETF – *RFC 9396, OAuth 2.0 Rich Authorization Requests*, 2023. Feingranulare Autorisierungsinformationen über `authorization_details`.

IETF – *RFC 9449, OAuth 2.0 Demonstrating Proof of Possession (DPoP)*, 2023. Senderbindung von Tokens.

NIST – *SP 800-207 Zero Trust Architecture*, 2020, sowie *SP 800-207A* zu granularen Policies auf Basis von Anwendungs- und Dienstidentitäten.

OWASP GenAI Security Project – *Top 10 for Agentic Applications*. Insbesondere Identity and Privilege Abuse.

Marcel Küppers – *Agentic AI Security Paper #01* und *#02*, August 2026.

Abbildungsverzeichnis

Die Serie

01	Von Model Security zu Action Security	ERSCHIENEN
02	Agent Inventory: Was läuft eigentlich bei uns?	ERSCHIENEN
03	Identität und Delegation für KI-Agenten	DIESES PAPIER
04	Least Capability	IN VORBEREITUNG
05	Runtime Authorization für autonome Systeme	IN VORBEREITUNG
06	MCP und Tool Security	IN VORBEREITUNG
07	Prompt Injection ist ein Aktionsproblem	IN VORBEREITUNG
08	Der Agent Action Ledger	IN VORBEREITUNG

Alle Papers erscheinen unter marcel-kueppers.de/agentic-ai-security-papers. Verwandt, außerhalb der Serie: *NIS2 trifft Agentic AI* (August 2026).

NÄCHSTER SCHRITT

Eine eigene Identität geben.

Im zweitägigen Training **„AI Agents für Security Management“** bauen Sie handelnde Agenten selbst – vollständig lokal, ohne eine Zeile Code. Sie richten dabei die Trennung ein, um die es in diesem Papier geht: eigene Identität für den Agenten, Delegation für den Auftrag, Zugangsdaten, die er nie zu sehen bekommt. Sie gehen mit lauffähigen Werkzeugen und einem 90-Tage-Plan nach Hause.

MEHR ERFAHREN UND PLATZ SICHERN

marcel-kueppers.de/ai-agents-security-management

Marcel Küppers · Security.Strategy.Resilience. – Agentic AI Security Papers, Nr. 03 „Identität und Delegation für KI-Agenten“, Version 1.0, Stand August 2026.

Quellen (Auswahl): IETF RFC 8693, RFC 9396, RFC 9449 · NIST SP 800-207/-207A · OWASP GenAI Security Project, Top 10 for Agentic Applications · Marcel Küppers, Agentic AI Security Papers #01 und #02 (2026). Framework-Stände bitte zum Zeitpunkt der Anwendung prüfen.

MARCEL KÜPPERS

SECURITY • STRATEGY • RESILIENCE