Authentication for SaaS and Cloud Services: Workforce, Customers, Partners, AI Agents, and Machines on One Identity Plane

Industry Guides / Last updated 2026-06-11 / https://www.scrambleid.com/learn/authentication-for-saas-and-cloud

In one sentence: SaaS and cloud-services authentication is harder than enterprise authentication because the same platform must serve internal workforce, customer admins authenticating from their own IdPs, support staff impersonating customers under controlled conditions, partner integrations on machine-to-machine paths, and now AI agents acting on behalf of users, all without slowing engineering velocity or failing the next security questionnaire.

TL;DR (canonical)

- The SaaS authentication surface is broader than most other industries: workforce, customer-managed SSO, customer end-users, support impersonation, partner APIs, machine-to-machine, and AI agent / MCP server identity.
- The bar is now phishing-resistant by default. Customer security questionnaires, FedRAMP baselines, and SOC 2 auditor expectations have all moved.
- Customer-managed SSO via OIDC or SAML and SCIM provisioning are table stakes. The "SSO tax" is dying as a competitive practice.
- Service-account secrets are the biggest M2M failure mode. Cloud-native workload identity (IRSA, Workload Identity Federation, Managed Identity) plus sender-constrained tokens (mTLS, DPoP) replace long-lived secrets.
- AI agents and MCP servers are now first-class identities. Treating them as "whatever credential the user happened to have" is the next breach class.
- The architecture that scales: a single identity model across workforce, customer, partner, and machine, with phishing-resistant ceremonies on human paths and sender-constrained tokens on machine paths, plus an audit log that maps every action to a verifiable identity.

Why SaaS authentication is uniquely hard

A SaaS or cloud-services company is simultaneously:

- An **enterprise** (workforce SSO, internal tools, privileged access).
- A **product** (customer admins, customer end-users, customer-managed SSO).
- A **platform** (partner integrations, public APIs, webhooks).
- A **multi-tenant operator** (one breach affects every tenant).
- An **AI integrator** (agents acting on behalf of users, model providers, MCP servers).

These carry different threat models, regulatory regimes, and engineering constraints. Conflating them produces the seams that attackers exploit. The architectures that work treat these as distinct identity domains with explicit, audited boundaries between them.

The threat landscape

Workforce credential phishing. SaaS engineers and ops teams are heavily targeted; one compromised account can grant production access. Phishing-resistant MFA on workforce SSO is the highest-leverage control.

Support-staff impersonation abuse. "View as customer" is a feature in nearly every SaaS. It is also a frequent insider-misuse vector and a frequent finding in security questionnaires. The mitigation: explicit customer consent, time-bounded scope, signed audit, and dual control on the most sensitive impersonation actions. Lockstep (in development) is ScrambleID's design for enforcing that dual control once it ships.

Customer SSO misconfiguration. A customer pastes the wrong IdP metadata, forgets to enforce MFA, or delegates SSO to a poorly secured downstream IdP. The result is account takeover that looks legitimate from the SaaS's perspective. Mitigation: validate customer IdP configuration on setup, and treat customer authentication strength as something the SaaS itself can observe and act on.

Partner-integration credential theft. A partner's CI/CD pipeline leaks an API key. The key is good for months. The blast radius is whatever scope you granted. Mitigation: sender-constrained tokens, short-lived credentials, scope minimization.

Service-account credential dumps. A long-lived service-account password committed to a public repo, exposed in logs, or harvested from a stolen developer laptop. Mitigation: eliminate long-lived secrets in favor of cloud-native workload identity.

AI agent and MCP server abuse. An agent acting on behalf of a user reaches further than the user can or should. An MCP server with a single shared credential is reachable from any client that can find its endpoint. Mitigation: bind agent identity, user delegation, and tool scope at every action; authenticate MCP servers as first-class identities.

Software-supply-chain compromise. A compromised dependency, build pipeline, or signing key results in malicious code shipped to every customer. Authentication does not solve this directly, but signed builds, hardware-bound signing keys, and short-lived credentials in CI dramatically reduce the blast radius.

Compliance landscape

Framework	What it requires (authentication-relevant)	Practical implication
SOC 2 (Trust Services Criteria)	CC6.1 (logical access), CC6.2 (authentication and credentials), CC7.x (monitoring)	Auditors expect MFA on customer-data and admin paths; phishing-resistant is the default at high-trust customers
ISO/IEC 27001:2022	A.5.16 (identity management), A.5.17 (authentication information), A.8.5 (secure authentication)	Implementation control set built around your ISMS; authentication evidence is core
FedRAMP (Low/Moderate/High)	NIST 800-53 controls; IA-2 series; phishing-resistant MFA is the cleanest fit at Moderate/High	Phishing-resistant MFA on privileged paths; identity binding to NIST 800-63 AAL2/AAL3 properties
SEC Cybersecurity Disclosure (4-day)	Disclosure of material cybersecurity incidents within 4 business days	Incident-response readiness depends on identifiable, signed authentication events
GDPR	Article 32 (security of processing); right to erasure; data residency	Authentication scope and audit are part of the security-of-processing posture
CCPA/CPRA	Reasonable security; consumer rights workflows	Strong authentication on consumer-rights workflows is increasingly an enforcement focus
DORA (EU financial-services SaaS)	ICT risk management including identity; third-party concentration	SaaS providers to EU financial customers face new identity audit expectations
State data-breach laws (US)	Varying notification windows	Lower breach probability has compounding state-by-state benefit
Industry-specific frameworks	HITRUST (healthcare SaaS), PCI DSS (payments SaaS), CMMC (defense industrial base SaaS)	Each adds its own authentication expectations on top of SOC 2/ISO 27001

For a deeper compliance view, see Compliance Mapping: NIST and CISA.

The SaaS channel mix

Channel	Who authenticates	Typical legacy method	Phishing-resistant pattern
Internal workforce SSO	Employees	Password + push	Phishing-resistant SSO (passkeys, FIDO2)
Production access (privileged)	SREs, DBAs, on-call	SSO + push + bastion + sudo	Phishing-resistant SSO + JIT elevation + dual control on sensitive ops

Channel	Who authenticates	Typical legacy method	Phishing-resistant pattern
Customer admin console	Customer admins	Password + customer-managed MFA, or customer SSO	Customer-managed SSO via OIDC/SAML; passkeys for self-managed customers
Customer end-users (B2C product)	End users	Email/password + sometimes SMS	Passkeys; SMS removed from chain
Customer end-users (B2B product, customer SSO)	End users	Customer IdP via SAML/OIDC	Customer IdP, with the SaaS enforcing minimum assurance properties
Support team customer impersonation	Support reps	"Login as user" admin tool	Explicit customer consent, time-bounded scope, signed audit, dual control on highest-risk
Partner / OAuth integrations	Partner systems	OAuth client_secret_basic	Sender-constrained tokens (mTLS, DPoP); JWT client assertions
Service-to-service (internal)	Microservices	Long-lived service-account creds, often in vaults	Cloud-native workload identity (IRSA, Workload Identity, Managed Identity)
CI/CD pipelines	Build agents	Long-lived API keys	OIDC federation to cloud (GitHub Actions OIDC, GitLab OIDC), short-lived credentials
Webhooks (inbound and outbound)	External and internal	Shared HMAC secret	Mutual TLS; signed payloads with short-lived keys
AI agents acting on behalf of users	Agent + user delegation	Reuse user's bearer token	Short-lived, scoped credential bound to agent identity + user delegation + resource scope
MCP servers (tool brokers)	MCP server identity	Often shared API key	First-class identity for the MCP server with sender-constrained credential

The largest underinvestment patterns: long-lived service-account credentials inside the cloud footprint, "view as customer" tooling without consent and audit, and AI-agent paths that reuse a user's bearer token without scoping.

Authentication patterns by use case

Pattern 1, workforce SSO and privileged access

Goal: Eliminate credential phishing on workforce paths; enforce least privilege and JIT elevation on production.

Approach: Phishing-resistant SSO across the workforce. SCIM-driven lifecycle for joiners/movers/leavers. JIT elevation for production access with cryptographic step-up at the

moment of elevation. Dual control on the highest-risk operations (production database changes, IAM policy edits, customer-data exports). Recovery and break-glass paths at the same assurance level as the primary (see Recovery and Fallback Playbook).

Outcome: The class of breach that starts with "an engineer clicked a phishing link" stops being the dominant one. The narrative of "the attacker had production access for X days" becomes "the attacker had a one-hour JIT scope that triggered an alert."

Pattern 2, customer-managed SSO and SCIM

Goal: Make customer-managed SSO and SCIM friction-free at every plan tier.

Approach: Support OIDC and SAML for SSO. Support SCIM 2.0 for user lifecycle. Validate customer IdP configuration on setup (test login, SCIM round-trip, attribute mapping). Do not gate SSO behind enterprise-tier pricing; this practice is increasingly a competitive disadvantage and a security questionnaire flag.

Outcome: Customers' IdP-based MFA enforcement becomes the SaaS's MFA enforcement for their users. Customer onboarding accelerates. The SaaS inherits the customer's audit posture rather than maintaining a parallel one.

Pattern 3, customer end-user authentication for B2C and self-managed B2B

Goal: Eliminate password-based ATO on customer-facing accounts.

Approach: Passkeys as the default authenticator. SMS removed from the recovery chain. Phishing-resistant recovery (identity proofing, account recovery codes printed once and held offline, or device-to-device recovery for users with multiple devices). Treat the recovery flow as the most-attacked surface, not an afterthought.

Outcome: Customer ATO collapses on the customer-facing path. Support load drops because customer "lost MFA" tickets stop being a daily occurrence.

Pattern 4, support-staff impersonation

Goal: Allow support to impersonate customers when needed without becoming an insider-abuse vector.

Approach: Customer consent at the moment of impersonation (explicit acknowledgment, optional with break-glass for severity-1 customer-impacting incidents). Time-bounded scope. Signed audit log of every action taken in the impersonated session. Dual control (via Lockstep when it ships) on the highest-risk impersonation actions (writing to financial records, exporting customer data, deleting accounts).

Outcome: "View as customer" stops being a security questionnaire finding and starts being a control example. Insider-misuse cases become detectable and prosecutable rather than ambiguous.

Pattern 5, partner / OAuth integrations

Goal: Eliminate long-lived shared secrets on partner integrations.

Approach: OAuth 2.0 with sender-constrained tokens. Prefer JWT client assertions (RFC 7523) or mTLS client authentication (RFC 8705) over `client_secret_basic`. For first-class partner integrations, mandate DPoP (RFC 9449) on access tokens. Tight scopes, aggressive expiry, full audit trail. RFC 9700 (OAuth 2.0 Security BCP) is the canonical reference.

Outcome: A leaked partner credential cannot be replayed from a different network or workload. The blast radius of any partner-side incident is bounded.

Pattern 6, service-to-service inside the cloud footprint

Goal: Eliminate long-lived service-account credentials.

Approach: Cloud-native workload identity:

- **AWS:** IAM Roles for Service Accounts (IRSA) for EKS workloads; IAM Roles Anywhere for off-cloud workloads; AWS STS AssumeRole with web identity for federated workloads.
- **GCP:** Workload Identity Federation; Workload Identity for GKE.
- **Azure:** Managed Identity (system-assigned and user-assigned); Azure AD Workload Identity for AKS.
- **Cross-cloud and hybrid:** SPIFFE/SPIRE for workload attestation across heterogeneous environments.

For service-to-service paths that cross trust boundaries, layer sender-constrained tokens (mTLS or DPoP) on top of workload identity.

Outcome: No persistent secret in source control, in vaults longer than minutes, or held by workloads that have been decommissioned. Forensics narrows quickly when something goes wrong.

Pattern 7, CI/CD pipelines

Goal: Eliminate long-lived API keys and credentials in build environments.

Approach: OIDC federation between the CI provider and your cloud:

- GitHub Actions: OIDC token to AWS via IAM Roles, to GCP via Workload Identity Federation, to Azure via federated credentials.
- GitLab CI/CD: ID tokens with the same federation pattern.
- Jenkins, CircleCI, others: equivalent OIDC patterns.

The build pipeline gets a short-lived credential scoped to the specific job, not a persistent admin key.

Outcome: A leaked CI secret is no longer a 6-month attacker dwell time; it is a job-duration window.

Pattern 8, webhooks (in and out)

Goal: Eliminate shared HMAC secrets that go stale.

Approach: Mutual TLS where partner relationship supports it. Signed payloads using short-lived keys with key rotation built into the protocol. Validate timestamp + nonce to prevent replay. Audit every webhook invocation with the signing key identifier and the verification result.

Pattern 9, AI agents acting on behalf of users

Goal: Treat the agent as a first-class identity. Bind every action to (agent, user, resource scope, time).

Approach: When the agent acts, it presents:

1. Its own identity (the agent identity, attested at instantiation).
2. The user's delegation (a short-lived token with explicit scope).
3. The resource it intends to act on.

The authorization decision is made on all three, not just on the user's bearer token. For deeper coverage, see AI Agent Authentication and AI Agent Tool Access Playbook.

Outcome: An agent that drifts (prompt injection, tool misuse, scope creep) hits a hard authorization boundary rather than carrying the user's full privileges into a context the user never approved.

Pattern 10, MCP server identity

Goal: Authenticate MCP servers themselves, not just the clients that call them.

Approach: Treat each MCP server as a first-class identity with a sender-constrained credential. Verify the MCP server's identity at registration, not just at runtime. Apply the same circle-of-trust pattern that governs any other tool integration. See AI Agent Tool Access Playbook for the full architecture.

Outcome: A rogue or compromised MCP server cannot be silently inserted into the tool chain. The agent only routes through MCP servers that are explicitly allowed and verifiable.

Customer trust and the security questionnaire

Customer security questionnaires are the de facto SaaS audit. Common authentication-relevant questions and the answers a phishing-resistant omnichannel architecture provides:

Question	Answer pattern
"Do you support SSO via SAML/OIDC?"	Yes, on every plan tier; both protocols supported
"Do you support SCIM for user lifecycle?"	Yes; tested round-trip with customer IdPs
"Do you require MFA for admin and privileged access?"	Yes; phishing-resistant MFA on workforce SSO; phishing-resistant step-up on production access

Question	Answer pattern
"Do you support customer-managed MFA enforcement policies?"	Yes; we honor customer IdP MFA enforcement and observe minimum assurance properties
"How do support staff impersonate customers?"	Customer consent at the moment, time-bounded scope, signed audit, dual control on the highest-risk actions
"Do you store long-lived service-account secrets?"	No; cloud-native workload identity for in-cloud paths; sender-constrained tokens for cross-boundary paths
"How do you authenticate AI agents acting on behalf of users?"	Agent identity + user delegation + resource scope, bound at every action
"Where do you log authentication events and how long are they retained?"	[Specifics: SIEM, retention, customer-shippable logs]
"Do you have a documented break-glass and recovery procedure?"	Yes; phishing-resistant recovery; not a help-desk-driven password reset
"How do you handle a stolen credential incident?"	[Specifics: detection, revocation, customer notification, audit reconstruction]

A SaaS that can answer all of these with concrete, current architectures rather than aspirational language clears the high-trust customer bar. A SaaS that cannot is increasingly priced out of enterprise deals.

Anti-patterns to avoid

1. **The "SSO tax."** Charging extra for SSO on a plan that already includes user accounts. This is increasingly a competitive disadvantage and a security questionnaire flag. SSO should be table stakes at every tier.
2. **Customer-managed SSO with no minimum assurance enforcement.** A customer's IdP might allow password-only authentication. The SaaS should observe and enforce minimum assurance properties, not pass the buck.
3. **Long-lived API keys for partner integrations.** OAuth with sender-constrained tokens is the path. `client_secret_basic` for high-trust integrations is increasingly indefensible.
4. **"View as customer" with no consent and no audit.** This becomes a security questionnaire finding the moment a customer asks. Build consent and audit before the question lands.
5. **Service-account passwords in source control or in long-lived vault entries.** Cloud-native workload identity exists; use it.
6. **AI agents that reuse the user's bearer token at full scope.** This is the next breach class. Bind agent identity, user delegation, and resource scope at every action.
7. **MFA on customer-facing login, no MFA on admin console.** Inverted threat model; admin console is the higher-value target.

8. **CI/CD pipelines with persistent admin credentials.** OIDC federation patterns exist for every major CI; use them.
9. **Webhook signatures using a single shared HMAC secret with no rotation.** Build rotation into the protocol or move to mutual TLS.
10. **A single audit log that conflates internal workforce, customer admin, support impersonation, and machine identity.** Separate streams; correlate via signed identity claims rather than free-text fields.

How to evaluate a vendor for SaaS authentication

Beyond the standard passwordless evaluation criteria (see Enterprise Passwordless Vendors Compared), SaaS buyers should weight:

1. **Customer-managed SSO and SCIM at every plan tier.** Day-1 capability across major IdPs.
2. **Standards-native M2M.** JWT client assertions, mTLS, DPoP, and cloud-native workload identity (IRSA, Workload Identity Federation, Managed Identity) integration.
3. **AI-agent and MCP-server identity.** First-class support for agent identity, user delegation, scoped tool access, and MCP server authentication.
4. **Support-impersonation tooling.** Customer consent, time-bounded scope, signed audit, dual-control workflow.
5. **Recovery and break-glass posture.** Phishing-resistant recovery; documented break-glass procedure that does not become the new ATO surface.
6. **Audit and SIEM integration.** Shippable, signed events that map to SOC 2 CC6/CC7, ISO 27001 A.5/A.8, and FedRAMP IA-2/AU-2.
7. **FedRAMP roadmap.** If your roadmap includes federal customers, the vendor's FedRAMP-aligned posture is a critical evaluation dimension.
8. **AI workload velocity fit.** Does the vendor's identity model compose with the AI agent platforms your engineering team uses, or does it require parallel implementation?

Key Takeaway

Authentication for SaaS and cloud services has to satisfy multiple identity domains simultaneously: workforce, customer admins (with their own IdPs), customer end-users, support impersonation, partner APIs, machine-to-machine, and AI agents/MCP servers. The architecture that scales is a single identity model with phishing-resistant ceremonies on human paths and sender-constrained tokens on machine paths, customer-managed SSO and SCIM at every plan tier, cloud-native workload identity replacing long-lived service-account secrets, and first-class identity for AI agents and MCP servers. The biggest failure modes are the SSO tax, long-lived API keys, view-as-customer tooling without consent and audit, and AI agents reusing user bearer tokens at full scope.

FAQ

What does SOC 2 require for authentication?

SOC 2 is a trust-services criteria framework, not a prescriptive control catalog. The Common Criteria and the Security category require logical access controls (CC6.1), authentication and credential management (CC6.2), and monitoring/audit (CC7.x). Auditors expect MFA on customer-data and admin paths as a baseline; phishing-resistant MFA is increasingly the default for high-trust customers and for FedRAMP-aligned controls. The actual control set is derived from your risk assessment and your customer commitments.

Does FedRAMP require phishing-resistant MFA?

FedRAMP baselines incorporate NIST SP 800-53 controls including IA-2(1), IA-2(2), and IA-2(8) (replay-resistant authentication). Phishing-resistant MFA (FIDO2/WebAuthn) is the cleanest fit for the IA-2 series and aligns with the broader OMB M-22-09 federal Zero Trust direction, which requires phishing-resistant MFA for federal personnel. CSPs targeting FedRAMP Moderate or High should plan for phishing-resistant MFA on privileged and federal-facing access paths.

Should B2B SaaS support customer-managed SSO and SCIM?

Yes. Enterprise customers expect SSO via OIDC or SAML to their existing IdP (Okta, Entra ID, Ping, Google Workspace) and SCIM provisioning for user lifecycle. Charging a premium for SSO ("the SSO tax") is increasingly a competitive disadvantage; the SOC 2 audit and the security questionnaire both treat SSO as table stakes. SCIM is harder to standardize across IdPs but is the only sustainable path to lifecycle automation.

How should a SaaS handle authentication for AI agents and MCP servers?

Treat the agent and the MCP server as first-class identities, not as "whatever credential the user happened to have." Issue a short-lived, scoped credential to the agent at the moment of action; bind it to the agent identity, the user delegation, and the resource scope. For MCP servers (the protocol layer between the agent and your APIs), authenticate the MCP server itself with a sender-constrained credential separate from the agent. See AI Agent Authentication and the AI Agent Tool Access Playbook for the full pattern.

How do we eliminate long-lived service-account secrets in our cloud?

Use cloud-native workload identity: AWS IRSA (IAM Roles for Service Accounts) and Roles Anywhere; GCP Workload Identity Federation; Azure Managed Identity. For service-to-service auth across boundaries, use sender-constrained tokens (mTLS per RFC 8705 or DPoP per RFC 9449) and short-lived JWT client assertions (RFC 7523). Replace any long-lived API keys, passwords, or SSH keys held in vaults with broker-issued credentials whose lifetime is measured in minutes, not months.

What's the difference between authentication for the support team and authentication for customer admins?

Support team authentication is workforce SSO with phishing-resistant MFA, plus dual control on customer-impersonation actions (you should have a "view as customer" workflow that requires explicit consent, time-bounded scope, and complete audit). Customer admin authentication is on the customer's terms: their IdP via SSO, their MFA enforcement, their session policy. The two should never converge into "the same admin console for both," because that conflates the trust boundaries.

What authentication does a partner integration need?

OAuth 2.0 with sender-constrained access tokens (mTLS or DPoP) on every machine-to-machine path. Avoid `client_secret_basic` for high-trust integrations; prefer JWT client assertions or mTLS client authentication. Scope tokens tightly, expire them aggressively, and audit every issuance. RFC 9700 (OAuth 2.0 Security Best Current Practice) is the authoritative reference.

References (public)

- AICPA SOC 2 Trust Services Criteria: https://www.aicpa-cima.com/topic/audit-assurance/audit-and-assurance-greater-than-soc-2
- ISO/IEC 27001:2022: https://www.iso.org/standard/27001
- FedRAMP: https://www.fedramp.gov/
- OMB M-22-09 (Federal Zero Trust): https://www.whitehouse.gov/wp-content/uploads/2022/01/M-22-09.pdf
- NIST SP 800-53 Rev. 5: https://csrc.nist.gov/pubs/sp/800-53/r5/final
- NIST SP 800-63-4: https://csrc.nist.gov/pubs/sp/800/63/4/final
- RFC 7523 (JWT Profile for OAuth 2.0 Client Authentication): https://datatracker.ietf.org/doc/html/rfc7523
- RFC 8705 (OAuth 2.0 Mutual-TLS): https://datatracker.ietf.org/doc/html/rfc8705
- RFC 9449 (OAuth 2.0 DPoP): https://www.rfc-editor.org/rfc/rfc9449.html
- RFC 9700 (OAuth 2.0 Security Best Current Practice): https://www.rfc-editor.org/rfc/rfc9700.html
- SCIM 2.0 (RFC 7644): https://datatracker.ietf.org/doc/html/rfc7644
- CISA, Implementing Phishing-Resistant MFA: https://www.cisa.gov/sites/default/files/publications/fact-sheet-implementing-phishing-resistant-mfa-508c.pdf

---

Related reading

- Phishing-Resistant Web Authentication

- M2M Authentication Without Secrets
- Machine Identity (PoP, DPoP, mTLS)
- AI Agent Authentication
- AI Agent Tool Access Playbook
- Lockstep: Dual Control
- Recovery and Fallback Playbook
- Compliance Mapping: NIST and CISA
- Enterprise Passwordless Vendors Compared