SECURITY ASSURANCE BRIEF

Single sign-on in Pytheus

How Pytheus authenticates your people against your own identity provider, which controls enforce that, and how each one is verified. Written for a security reviewer assessing whether to connect an enterprise directory.

PROTOCOLS	TESTED AGAINST	PROVISIONING	LAST VERIFIED
OIDC, SAML 2.0	**Okta, Microsoft Entra ID**	**Invite-only**	**17 September 2026**

The current version of this brief is published at pytheus.io/security/sso.

Scope and posture

Pytheus acts as a service provider. Your identity provider remains authoritative for who your people are, whether they are still employed, and what second factor they present. Pytheus never receives a directory password and never issues the credentials involved: the OIDC client secret is minted in your tenant, and the SAML signing certificate is yours.

A successful assertion does one thing: it binds an identity-provider subject to a Pytheus account that an administrator at your organization has already invited. It never creates an account. That is a deliberate decision, described under identity binding, and it is the single most common expectation mismatch in an enterprise rollout.

> **Presets and the generic path**
>
> Any compliant provider connects. The setup wizard offers Microsoft Entra ID as a named preset, and everything else, Okta, Google Workspace, Auth0, Ping, JumpCloud, ADFS, Keycloak and the rest, goes through the generic OpenID Connect or SAML 2.0 path. A preset is a shortcut that pre-fills known values and enables one extra check; it is never a requirement, and no provider is on an allowlist.
>
> Where the two paths differ is named explicitly in the control matrix rather than glossed over, because the Entra preset adds a `tid` tenant check that the generic path does not run.

> **How to read the status labels**
>
> **Verified** means an automated test exercises the control against a real signed assertion or a real signed token, and fails if the control is removed. **Operational** means the behaviour is implemented and reviewed but proven by configuration rather than by a test. **Not supported** means exactly that, stated deliberately.

OIDC sign-in, step by step

Authorization code flow with PKCE. Every value that must match later is generated and stored server-side before the user leaves, and consumed exactly once when they return.

1. **Connection resolved server-side.** The email domain selects the connection. A client cannot name the tenant or the organization it wants; the scope is derived from the request, never accepted from it.
2. **Handshake minted.** A random nonce, a PKCE verifier, and a signed state token, stored with a 15 minute expiry against the connection's own tenant.

3. **User authenticates at your IdP.** Pytheus is not involved. Your password policy, conditional access and MFA all apply.
4. **State consumed once.** The returned state is verified by HMAC and redeemed with a conditional update, so of two concurrent callbacks exactly one succeeds and the other is recorded as a replay.
5. **Code exchanged.** Over HTTPS to your token endpoint, authenticated with the client secret and carrying the PKCE verifier.
6. **Token validated.** Audience, expiry and nonce are checked against the handshake, and `iss` must equal the issuer discovered when the connection was saved, by exact string comparison. For a tenant-specific Microsoft authority that comparison is itself a tenant binding: a token minted in a different Entra tenant carries that tenant's identifier in `iss` and is refused. A connection created from the Microsoft preset adds a second, independent binding on top, where `id_token.tid` must equal the tenant your administrator declared. See the note below on which path you are on.
7. **Address accepted or refused.** The `email_verified` rule below decides, and which rule admitted the identity is written to the audit trail.
8. **Session issued through the standard path.** The same post-authentication step as password sign-in, so Pytheus-side MFA and account status behave identically.

> **Entra tenant binding, precisely**
>
> The `tid` check runs on connections created from the Microsoft preset, where the administrator declares a tenant identifier separately from the issuer. A connection created through the generic OpenID Connect path and pointed at Entra keeps the issuer binding and does not run the `tid` check. Stating this the other way round, as an unconditional property of every Entra connection, would be wrong.
>
> The practical difference is smaller than it first reads, because the issuer comparison already pins the tenant for any tenant-specific authority. What the preset adds is a second binding against an independently declared value, and validation of the issuer's shape at the moment the connection is saved rather than at first sign-in.
>
> The one Entra configuration that cannot work either way is a shared authority: `common`, `organizations` or `consumers` publish a discovery document whose issuer is a template rather than a real tenant, so no token can ever match it. Pytheus refuses those at save time with a message naming the fix, instead of accepting a connection that fails silently on first use.

> **On ID token signatures**
>
> The ID token arrives over a direct, TLS-secured, client-authenticated back channel from your token endpoint. Per OIDC Core 3.1.3.7, TLS server validation is what establishes the issuer in this flow, so the JWS signature is not additionally verified. That argument holds only while the channel really is TLS, so a test asserts that a plaintext `http://` issuer produces no working configuration at all. If anyone later relaxes transport security, that test fails before the change ships.

SAML sign-in, step by step

Service-provider initiated, HTTP-POST binding at the assertion consumer service. Assertion signing is expected by default.

1. **AuthnRequest issued.** Its ID is persisted alongside a signed RelayState, and that stored ID is what the response must later echo.

2. **User authenticates at your IdP.** Pytheus is not involved.
3. **RelayState consumed once.** Same one-time redemption as OIDC. A response arriving without it is refused, which is what makes replay detection work and is also why unsolicited sign-in is unsupported.
4. **Document schema-validated.** Against the SAML XSD before anything else is trusted.
5. **Signature verified.** Against the certificate stored on the connection. An unsigned assertion, one signed with a key we do not hold, or one altered after signing is refused.
6. **Assertion scoped.** `InResponseTo` must equal the stored AuthnRequest ID; `Audience` must name us and must be present; `Destination` must match our ACS URL and must be present; the conditions window must be current, with 120 seconds of clock tolerance.
7. **Subject bound.** The asserted address must sit inside the domain the connection is authoritative for.
8. **Session issued through the standard path.** As above.

> **Presence, not just agreement**
>
> The audience and destination checks require the elements to be **present**, rather than validating them only when supplied. An assertion that omits `Audience` names no service provider at all, which would otherwise make an assertion minted for a different relying party acceptable here. Both the wrong-value and the missing-value cases are covered by tests.

Control matrix

Each control names the test that fails if the control is removed.

CONTROL	STATUS	EVIDENCE
PKCE (S256) on every OIDC handshake	Verified	`sso-oidc-roundtrip`
Nonce bound per handshake and checked on return	Verified	`sso-oidc-roundtrip`
Issuer pinned to the one discovered at setup, plus audience and expiry	Verified	`sso-oidc-roundtrip`
Entra tenant binding via `tid`, on Microsoft preset connections	Verified	`sso-oidc-roundtrip`
Shared Microsoft authorities refused when the connection is saved	Verified	`sso-connection-issuer`
TLS required to the token endpoint	Verified	`sso-oidc-config`
SAML XSD schema validation before trust	Verified	`sso-saml-parse-guard`
XML signature verification	Verified	`sso-saml-roundtrip`
Signature-wrapping rejection	Verified	`sso-saml-roundtrip`
Audience and destination, presence required	Verified	`sso-saml-roundtrip`
`InResponseTo` bound to the stored request	Verified	`sso-saml-roundtrip`
Conditions window and clock tolerance	Verified	`sso-saml-roundtrip`

CONTROL	STATUS	EVIDENCE
One-time state redemption, race safe	Verified	`sso-state-replay`
Email domain binding on identity link	Verified	`sso-identity-link`
`email_verified` treated as three states	Verified	`sso-oidc-email-verification`
Platform staff cannot configure customer SSO	Verified	`sso-settings-rbac-matrix`
Cross-organization access refused	Verified	`sso-cross-org-isolation`
Lockout prevention on sign-in policy	Verified	`sso-policy-safety`
Break-glass restricted to the highest internal tier	Verified	`unlock/route`
Rate limiting on start and callback	Verified	`sso-callback-rate-limit`
Secrets encrypted at rest with authenticated AES-256-GCM	Verified	`field-encryption`
Secrets excluded from audit payloads	Verified	`connections/route`
Credential rotation in place, re-encrypted on write	Verified	`sso-connection-rotation`
Credential expiry recorded, surfaced and never enforced	Verified	`sso-credential-expiry`
SSRF hardening on metadata fetch	Verified	`sso-metadata-fetch`
Row-level security on both SSO tables	Operational	`check-sso-rls`
IdP-initiated sign-in	Not supported	–
Just-in-time provisioning	Not supported	–

Identity binding and provisioning

Provisioning is invite-only. A successful assertion never creates an account. To sign in, a person must already hold an active membership that an administrator at your organization created. Someone who authenticates successfully at your IdP but has no Pytheus account is told that plainly and pointed at their administrator, rather than shown a sign-in failure.

The email_verified rule

Pytheus treats the OIDC `email_verified` claim as three states rather than two, because an identity provider that says nothing and one that says no are different things.

CLAIM	OUTCOME	REASONING
Present and true	Accepted	Your IdP asserts the address. Okta sends this.
Present and false	Refused, always	An explicit statement that the address was not verified. No fallback overrides it.

CLAIM	OUTCOME	REASONING
Absent	Accepted only inside the connection's domain	Microsoft Entra omits this claim for organizational accounts. Without the fallback, every Entra tenant would fail its first sign-in.

The fallback is reachable only after the token's issuer and signature have been established, and on a Microsoft preset connection only after `tid` has matched. The address must then sit inside the domain the connection declares, and must already belong to an invited member. The audit record names which of the two rules admitted each identity.

Availability and lockout prevention

Requiring SSO turns off password sign-in and password reset together. If the identity provider connection does not actually work, that combination leaves nobody a way in, including the organization owner. Pytheus refuses to enter that state.

Before the sign-in policy can be raised, three conditions must hold: an enabled connection exists, it passes live validation, and **at least one person has completed a real SSO sign-in in the last 30 days**. The third is the one that matters. The first two describe configuration, and configuration can look healthy while sign-in does not work.

The same protection applies in reverse: the last enabled connection cannot be disabled or deleted while the policy requires it. Relaxing the policy is never blocked, since that is how an administrator recovers.

An organization that still strands itself can be unlocked by Pytheus staff. That action sits at our highest internal permission tier, requires a written reason, clears only the two policy flags, and touches no connection or credential, so your identity provider configuration survives the rescue intact. It is written to your own audit trail as well as ours, so you can see that we changed your sign-in policy even when you asked us to.

Credential expiry

Pytheus records when your client secret or signing certificate expires and shows an escalating warning on the connection, moving from advisory to urgent inside the final week. Where a certificate is supplied, the expiry is read from the certificate itself rather than typed in. Credentials can be replaced in place from the same screen, which matters because the alternative would be deleting and recreating the connection, and that collides with the protection above. The warning is in-product only today: no email is sent, so it is worth an administrator checking the screen ahead of a known renewal.

It **never disables a connection automatically**. A recorded date is not authoritative enough to justify cutting off access, and doing so would manufacture the very outage the controls above exist to prevent.

Tenant isolation

Every SSO connection is keyed to a tenant, and every query carries both the tenant and the organization. Holding an owner role in one organization confers nothing in another. A member of one organization requesting another receives a refusal that is identical whether or not the target exists in a given portfolio, so response codes cannot be used to enumerate customers.

Pytheus platform staff, **including the highest internal role**, cannot configure a customer's identity provider. A support engineer must not be able to point your sign-in at a provider of their choosing, which would convert

support access into the ability to authenticate as any of your people. This is asserted for every role in the system rather than spot-checked.

Row-level security policies sit underneath the application-layer scoping on both SSO tables as a second barrier. They are deliberately not the standard organization-scoped shape: sign-in start and login discovery both read these tables before any scope exists, so an organization-scoped policy would return nothing and prevent sign-in entirely.

Credential handling

Client secrets, identity provider certificates and cached metadata are encrypted at rest with AES-256-GCM under a key held outside the database. Each value is encrypted under a fresh initialisation vector and carries an authentication tag, so a modified ciphertext fails to decrypt rather than decrypting to something else. Decrypted values exist only for the duration of a request.

No credential is written to an audit record. Audit entries note whether a secret was supplied, never its value, and the query that lists connections for the administration screen does not select the encrypted columns at all.

Credentials can be rotated in place. This matters more than it sounds: without it, replacing an expiring certificate would mean deleting and recreating the connection, which collides with the lockout protection above and is the most likely way a customer strands themselves during routine certificate renewal.

Limitations, stated deliberately

These are design decisions and known gaps rather than oversights. A reviewer should weigh them as part of the assessment.

IdP-initiated sign-in

Not supported

A signed RelayState is mandatory, which is what makes replay detection work. The Entra My Apps tile and the Okta application chiclet both post without one, so those tiles will not sign a user in. Start from the Pytheus login page instead.

Just-in-time provisioning

Not supported

Accounts must be invited before first sign-in. Deliberate, and worth confirming against your onboarding expectations, since many enterprise buyers assume the opposite.

Expiry warnings are in-product only

No email

An expiring credential is flagged on the connection screen with escalating urgency, but no notification is sent. An administrator who does not open settings before a renewal date will not be told. Plan certificate renewals rather than relying on the warning.

Encryption key rotation

Manual

Stored ciphertext carries no key version, so rotating the encryption key requires a re-encryption migration rather than a rolling change.

Expired handshake records

Not reaped

Consumed and expired authorization states are retained indefinitely. This is a growth concern rather than a security one, since each record is single-use and time-bounded.

Metadata fetch and DNS rebinding

Partial

Fetching identity provider metadata by URL enforces HTTPS, blocks private and loopback address ranges, refuses redirects and caps the response size. It validates the hostname rather than the resolved address, so DNS rebinding is not addressed. Supplying metadata XML directly avoids this path entirely.

One domain per connection

By design

Domain matching is exact. An organization using both example.com and corp.example.com configures two connections, or designates a default.

How these claims were verified

Every control marked verified is exercised by an automated test that constructs real cryptographic material. The SAML tests mint assertions signed with a real X.509 key and run them through the same parser, schema validator and signature verification that production uses. The OIDC tests mint RS256 tokens and run a full authorization code exchange against an in-process issuer, with signature verification and every claim check on the real code path. Nothing on the protocol path is stubbed.

Each control also has negative coverage: a tampered signature, an untrusted signing key, an unsigned assertion, a wrong or absent audience, a wrong or absent destination, a mismatched request binding, an expired condition, a replayed handshake, and a foreign email domain are each asserted to be refused, and to be refused for their own reason rather than incidentally.

The brief itself is checked too. Its control matrix is structured data, and a test reads every row and fails if the named evidence file is not present in the repository. A control cannot claim a test that was deleted or renamed.

What this brief does not claim

Automated tests prove protocol correctness against material Pytheus generates. They cannot prove how a specific identity provider behaves in a specific tenant. Connection setup should be walked end to end against your own directory before rollout, and Pytheus will do that with you.

This brief also describes controls, not an independent audit. It is written to let your team assess the implementation directly and to name where the boundaries are.