CHECKLIST

Troubleshooting Checklist: Zero Trust Network Access Failures

A practical, vendor-neutral checklist for diagnosing Zero Trust Network Access failures by symptom, from authentication and policy checks to routing, DNS, device posture, and rollback-safe validation.

Troubleshooting Checklist: Zero Trust Network Access Failures

Use this checklist to isolate whether the failure is caused by identity, policy, posture, routing, DNS, transport, or application reachability. Work from the safest, highest-signal checks first.

Scope and assumptions

This checklist is for technical teams troubleshooting access to private applications through a Zero Trust access layer. It assumes you can inspect:

- Identity logs
- Policy evaluation logs
- Client status
- Network paths
- DNS resolution
- Device posture signals
- Connector or gateway health

It is vendor-neutral and applies whether the access path uses an agent, browser-based access, connectors, or application gateways.

---

1) First five checks

Before changing anything, confirm these five items:

- [ ] User and app scope: Is the failure isolated to one user, one device, one application, one network location, or one policy group?
- [ ] Identity state: Is the user authenticated successfully, and is the expected identity provider, token, or session being accepted?
- [ ] Policy decision: Was the request denied, challenged, redirected, or never evaluated?
- [ ] Device and posture state: Does the endpoint meet required checks for OS version, agent health, certificate, disk encryption, or EDR status?
- [ ] Path reachability: Can the access service reach the private application or connector, and can the endpoint resolve and reach the access entry point?

If any answer is unknown, gather evidence before touching routing, firewall rules, certificates, or policy exceptions.

---

2) Capture a known-good baseline

Compare one working user, one failing user, and one healthy application path.

Record:

- [ ] User identity and group membership
- [ ] Assigned policy path
- [ ] Device posture at the time of success and failure
- [ ] Client version or browser access mode
- [ ] DNS answer for the application name
- [ ] Gateway or connector health
- [ ] App target IP, port, and TLS expectation

- [ ] Policy decision and reason code

Also note the last thing that changed:

- [ ] Identity rule
- [ ] Posture requirement
- [ ] Connector certificate
- [ ] DNS record
- [ ] Outbound firewall policy
- [ ] Routing or proxy setting

---

3) Do not change these yet

During an active failure, avoid broad fixes that reduce security or obscure evidence.

- [ ] Do not disable Zero Trust policy globally to test access.
- [ ] Do not widen application access to all users before identifying the failure.
- [ ] Do not replace certificates, connectors, or agents before confirming the failure domain.
- [ ] Do not change multiple layers at once.
- [ ] Do not assume every timeout is a firewall block.

---

4) Diagnose by user-visible symptom

A. Repeated login, MFA, or SSO prompts

Possible causes:

- [ ] Clock skew between client, identity provider, or access service

- [ ] Expired or mis-scoped token, assertion, or session cookie
- [ ] Broken SSO redirect URI, IdP trust, or certificate chain
- [ ] Browser restrictions on third-party cookies or embedded auth flows
- [ ] Conditional access or MFA policy conflict

Fast checks:

- [ ] Confirm the same user can authenticate directly to the identity provider.
- [ ] Review identity provider sign-in logs and access service audit logs for the same timestamp.
- [ ] Validate system time and time sync on the client and any involved gateway or proxy.
- [ ] Test a clean browser profile or private session.

Safer fixes:

- [ ] Clear only the affected session.
- [ ] Correct time sync or certificate trust if drift is confirmed.
- [ ] Adjust SSO trust or redirect settings only after verifying the exact error.

Validate:

- [ ] Authentication succeeds once without looping.
- [ ] The IdP log shows successful assertion or token issuance.
- [ ] The access log shows a valid session and policy evaluation.

Rollback if:

- [ ] Login still loops after a clean session and clock check.
- [ ] A recent trust or cookie-related change made the issue worse.

B. Immediate deny after successful authentication

Possible causes:

- [ ] User not in the expected group or role
- [ ] Policy rule order causing a deny before an allow

- [ ] Device posture missing required signal
- [ ] Geographic, network, or risk-based condition blocking the session
- [ ] Application tag, hostname, or resource label mismatch

Fast checks:

- [ ] Inspect the policy decision record and reason code.
- [ ] Compare the user?s actual group membership against rule conditions.
- [ ] Verify device posture data is fresh and complete.
- [ ] Confirm the app matches by hostname, path, port, or resource ID exactly as configured.

Safer fixes:

- [ ] Correct the specific attribute mismatch rather than weakening the whole policy.
- [ ] Reorder policy rules only after verifying which rule matched first.
- [ ] Refresh posture collection or re-enroll the device if the signal is stale or missing.

Validate:

- [ ] Policy logs show the intended allow rule matched.
- [ ] Device posture updates within the expected collection window.
- [ ] The user reaches the application without a bypass or exception.

Rollback if:

- [ ] A change causes broader deny events.
- [ ] The policy decision becomes ambiguous.

C. Connection spins, then times out

Possible causes:

- [ ] Connector or gateway is down, overloaded, or unhealthy
- [ ] Application host is unreachable from the connector network
- [ ] TLS handshake failure between access layer and app
- [ ] Firewall or egress restriction blocks the connector path

- [ ] Split routing or proxy behavior breaks the flow

Fast checks:

- [ ] Check connector or gateway health status.
- [ ] Verify the connector can reach the private application host directly.
- [ ] Confirm the application is listening on the expected port.
- [ ] Inspect TLS expectations on both sides.
- [ ] Check whether the same app works from a different network or entry path.

Safer fixes:

- [ ] Restore connector health or capacity.
- [ ] Correct route, proxy, or allowlist entries.
- [ ] Fix app mapping, target host, or port if the definition is wrong.
- [ ] Address TLS trust only after confirming the handshake failure.

Validate:

- [ ] The connection completes without spinning or timing out.
- [ ] Connector health returns to normal.
- [ ] The app target is reachable from the access layer.

Rollback if:

- [ ] Timeouts broaden to other applications.
- [ ] The connector or route change affects unrelated services.

D. Some apps work, one app fails

Possible causes:

- [ ] App-specific routing or policy mismatch
- [ ] Incorrect hostname, port, or path mapping
- [ ] Missing application tag or resource label
- [ ] Backend service unavailable for that app only

- [ ] Per-app TLS or certificate mismatch

Fast checks:

- [ ] Compare the failing app definition with a working one.
- [ ] Verify target hostname, IP, port, and path.
- [ ] Confirm the access policy includes the app by the intended identifier.
- [ ] Check whether the backend application itself is healthy.

Safer fixes:

- [ ] Correct the app mapping or label.
- [ ] Restore the specific target endpoint.
- [ ] Adjust the app?s TLS expectation only after verifying backend behavior.

Validate:

- [ ] The failing app matches the intended rule or mapping.
- [ ] Only the affected app is restored.
- [ ] Other applications remain unchanged.

Rollback if:

- [ ] The change impacts unrelated applications.
- [ ] The app still fails after mapping correction.

E. Works on one network, fails on another

Possible causes:

- [ ] DNS response differs by network
- [ ] Egress filtering, proxy, or captive portal interference
- [ ] Split DNS or split-horizon resolution issues
- [ ] Local network blocks the access entry point
- [ ] Regional or network-based policy condition mismatch

Fast checks:

- [ ] Compare DNS answers from both networks.
- [ ] Compare outbound restrictions, proxy settings, and firewall rules.
- [ ] Verify the access entry point is reachable from the failing network.
- [ ] Confirm policy does not depend on network context you did not intend.

Safer fixes:

- [ ] Correct split DNS or resolver configuration.
- [ ] Add the required allowlist entry for the access service.
- [ ] Remove unintended network-location dependence from policy.

Validate:

- [ ] DNS answers are consistent where they should be.
- [ ] The access entry point is reachable from both networks.
- [ ] The app opens from the previously failing network.

Rollback if:

- [ ] DNS or egress changes affect unrelated sites or services.
- [ ] A policy relaxation expands access beyond the intended network scope.

F. Browser access works, agent access fails

Possible causes:

- [ ] Client version mismatch
- [ ] Broken local certificate chain or trust store
- [ ] Agent posture reporting problem
- [ ] Cached client state or stale session
- [ ] Endpoint-specific trust or registration issue

Fast checks:

- [ ] Compare client version and configuration with a known-good device.
- [ ] Review device certificate, trust chain, and posture data.

- [ ] Clear local client cache or test a fresh agent sign-in.
- [ ] Confirm browser and agent use the same identity and policy path.

Safer fixes:

- [ ] Repair or re-enroll the agent.
- [ ] Refresh certificate trust.
- [ ] Remove only the stale local client state.

Validate:

- [ ] Agent access succeeds on the same device.
- [ ] Posture data is current.
- [ ] Browser and agent now produce the same policy outcome.

Rollback if:

- [ ] Re-enrollment breaks other managed settings.
- [ ] The issue returns after cache cleanup.

---

5) Evidence to collect before escalation

Collect the following when the issue is not immediately clear:

- [ ] Exact timestamp of the failure
- [ ] User identity
- [ ] Device name and OS
- [ ] Client or browser version
- [ ] Authentication trace or IdP log entry
- [ ] Policy decision and reason code
- [ ] Posture status and collection time
- [ ] DNS answer for the application name
- [ ] Connector or gateway health state
- [ ] Application target host, port, and protocol

- [ ] Recent changes affecting identity, policy, routing, DNS, or certificates

---

6) Safe validation before production use

Before you declare the issue fixed, verify:

- [ ] The original symptom is gone.
- [ ] The fix applies only to the intended user, device, or app.
- [ ] The policy decision now matches the expected path.
- [ ] The posture signal is current and complete.
- [ ] The path from access layer to application is healthy.
- [ ] No unrelated applications or users were impacted.

---

7) Rollback-safe change habits

Use these habits to avoid creating a wider incident:

- [ ] Change one layer at a time.
- [ ] Keep a before-and-after record.
- [ ] Prefer targeted fixes over broad exceptions.
- [ ] Re-test with one known-good and one known-bad case.
- [ ] Revert immediately if blast radius increases or the decision trail becomes unclear.

---

8) Quick triage summary

Use this shortcut when you need a fast read:

- [ ] Login loop: check identity, cookies, time, and trust.
- [ ] Denied after login: check policy, group membership, posture, and app matching.
- [ ] Timeout: check connector health, app reachability, route, proxy, and TLS.
- [ ] One app only: check mapping, label, and backend health.
- [ ] Network-specific failure: check DNS, egress filtering, and split-horizon behavior.
- [ ] Agent-only failure: check client version, certificate, posture, and local trust.

---

9) Incident notes template

- [ ] Date and time:
- [ ] User:
- [ ] Device:
- [ ] Application:
- [ ] Symptom:
- [ ] Identity provider result:
- [ ] Policy decision and reason:
- [ ] Posture state:
- [ ] DNS result:
- [ ] Connector or gateway health:
- [ ] App target host and port:
- [ ] Change made:
- [ ] Validation result:
- [ ] Rollback needed?:

---

10) Final check

If you cannot identify the failure domain after the first five checks, stop and gather more evidence. In Zero Trust troubleshooting, the fastest path to a fix is usually the safest path: confirm identity, confirm policy, confirm posture, confirm reachability, then change only the layer that failed.