National Cyber Security Centre
a part of GCHQ

Zero Trust Network Access (ZTNA)

How to understand, apply and evolve Zero Trust to protect your organisation's systems, data and users.



This guidance provides an overview of Zero Trust Network Access (ZTNA). It is primarily aimed at architects, security practitioners, and technical decision makers responsible for designing or evolving access networks, and will help them to prepare for and design a ZTNA architecture.

It is intended to support informed decision-making rather than prescribe a single ZTNA solution or implementation approach. Organisations should make decisions based on their existing architecture, risk appetite, and objectives.

How to use this guidance

The sections within this guidance are organised to guide the reader from understanding to implementation of ZTNA.

Sections 1 (Introduction to ZTNA) and 2 (Signals and policy engines) explain the underlying concepts and terminology, to build a solid understanding. Sections 3 (What to do before building a ZTNA architecture) and 4 (How to implement ZTNA) translate these ideas into practical guidance and recommend implementation approaches, describing the preparations and architectural properties needed before adopting ZTNA. Sections 5 (ZTNA reference architecture) and 6 (ZTNA anti-patterns) provide reference materials for additional detail and clarification.

Organisations should ensure they have a clear understanding of the core ZTNA concepts before making design or deployment decisions. After that, readers may find it useful to move between the implementation-focused sections as needed, returning to conceptual material to inform and refine design choices.

Contents

- **Introduction to ZTNA** - introduces Zero Trust Network Access (ZTNA) and establishes a clear understanding of what it is intended to achieve. It defines the core concepts of ZTNA and addresses some common misconceptions.
- **Signals and policy engines** - explains how policy decisions are made within a ZTNA architecture. It introduces the concept of signals, how they can be categorised, and what constitutes a baseline set of signals. It also describes the role of policy engines and how they are used within a ZTNA architecture to make and enforce access decisions.
- **What to do before building a ZTNA architecture** - outlines the key prerequisites and preparatory steps organisations should consider before beginning the implementation of ZTNA.
- **How to implement ZTNA** - introduces 8 high-level design requirements essential to a modern ZTNA implementation. These are intended to help organisations assess whether their ZTNA architecture includes the necessary features and capabilities.
- **ZTNA reference architecture** - provides illustrative examples of using the design requirements to develop a ZTNA architecture, avoiding the identified anti-patterns. It applies the concepts introduced throughout the guidance to show how they could be implemented in practice. These examples are illustrative only and should be adapted to meet an organisation's specific use case and risk context.
- **ZTNA anti-patterns** - describes 4 anti-patterns identified by the NCSC, and aims to help organisations recognise them early to design them out of their ZTNA architectures.

Introduction to ZTNA

What is ZTNA?

Zero Trust Network Access (ZTNA) is an architectural approach for controlling how users and devices access applications over a network.

The perimeter-based security model

For many years, organisations have relied on a perimeter-based model (also described as a 'walled garden' or 'castle and moat') to manage their network security. In this model, strong controls such as firewalls protect the boundary of the network. Users are authenticated at the perimeter, and once inside, are typically granted broad access to internal systems. The model implicitly trusts users based on network location.

This approach was effective when users, devices, and services were largely contained within a well-defined network boundary.

Why the traditional approach is no longer enough

The risk environment in which organisations operate has changed significantly, which means network location alone is no longer a reliable basis for trust. For example:

- Attackers are increasingly capable of stealing credentials, compromising endpoints, and moving laterally within a network once they have established an initial foothold.
- Modern working practices – such as remote access, widespread use of Software as a Service (SaaS) and the adoption of cloud-hosted services – mean that applications, users, and devices are no longer confined to a single network perimeter.

How zero trust addresses this problem

The NCSC's zero trust architecture design principles address this shift in the risk environment by removing implicit trust from users, devices, services, and networks.

ZTNA applies zero trust principles specifically to access. In a ZTNA approach:

- network connectivity alone never grants access to a service
- each access request is explicitly authorised, based on defined policy and contextual information

What a well-designed ZTNA architecture should do

This guidance explains how the NCSC's zero trust principles may be translated into ZTNA design requirements, and how those requirements can be used to design access architectures that better reflect the realities of modern threats and working practices. This is achieved when access patterns, segmentation, and policy enforcement are deliberately designed to align with zero trust principles.

A well-designed ZTNA architecture should:

- Ensure access is explicitly authorised, based on a set of signals.
- Reduce the overall attack surface by limiting application exposure.
- Restrict lateral movement so that compromise of one device or account cannot easily spread across the network.
- Limit the blast radius of a security incident.
- Enforce continual verification, not one-time access decisions.
- Apply consistent access control across private applications, SaaS and cloud services.

For further background on zero trust and common misconceptions, see the NCSC's guidance on demystifying zero trust.

Where ZTNA applies

ZTNA architectures typically apply in 2 particular access scenarios:

- **Private application access**. This provides secure, brokered access to internally hosted applications (for example, on-premises or private cloud services) while minimising external connectivity.
- **SaaS application access**. This is an identity-centric access control for internet-hosted applications, typically enforced through single sign-on (SSO) and application-level policy enforcement.

The unifying principle for both scenarios is the same: access is explicitly authorised, narrowly scoped, continually enforced, and observable, regardless of where users, devices or applications are located.

Signals and Policy Engines

This section describes the role of signals and policy engines within a ZTNA architecture. A **policy engine evaluates signals about the identity, security posture, and behaviour of users and devices to determine whether an access request should be authorised**. The focus here is on how decisions are made, rather than on how they are enforced, often referred to as policy enforcement points.

This section should be read alongside the NCSC's zero trust architecture design principles and the wider ZTNA guidance. It should be adapted to reflect your organisation's architecture, threat model and risk appetite.

What are signals?

Signals are data points used by a policy engine to evaluate an access request. To support robust and reliable decisions, policy engines should consume signals from multiple, independent sources relating to both the user and the device. No individual signal is perfect, but combining signals increases assurance and reduces reliance on any single control, making bypass more difficult.

Selecting the most appropriate signals for an environment can be challenging as requirements will vary depending on your architecture. Requirements will also evolve over time as threats change or new capabilities become available, so you should seek to continually review and adapt your signals.

A practical starting point is to **define a baseline set of signals** aligned to the NCSC's zero trust architecture design principles about your user, service and device identities and user behaviour, service and device health.

In practice, this means prioritising signals relating to:

1. User identity
2. Device identity

3. Device health
4. User behaviour

The following are common examples of signals for each of these categories.

1. User identity signals

User identity signals are typically sourced from the organisation's Identity Provider (IdP) which provides single sign-on (SSO) for most corporate services. Integration between the IdP and the policy engine is critical.

Examples of user identity signals include:

- **User and group membership**
  Integrating user and group information from the IdP into the policy engine enables evaluation of user and group memberships as part of an access decision. Policies may allow access to specific users or groups while denying others. For example, users in an HR group may be granted access to HR systems, while other departments are denied.
- **Authentication strength**
  Authentication strength reflects the assurance level of the authentication mechanism used. This may range from a password and time-based one-time password (TOTP) to stronger methods such as phishing-resistant credentials backed by cryptographic hardware (for example, hardware security keys or passkeys). Policies can require stronger authentication for higher-risk or more sensitive applications.

2. Device identity signals

Device identity signals establish that a request originates from a device owned and managed by the organisation.

Device identity should be cryptographically bound to hardware where possible, for example using a Trusted Platform Module (TPM). Hardware-backed identity reduces the risk of device spoofing compared to software-only approaches.

Combining both user and device identity allows organisations to have greater confidence that a known user is accessing a resource from a managed corporate device.

Examples of device identity signals include:

- **MDM-managed device identity**
  Some mobile device management (MDM) platforms manage device identity and integrate directly with policy engines. The policy engine can query the MDM to confirm that a device is enrolled and recognised. Organisations should understand how device credentials are generated, stored and protected within the MDM platform.
- **Device certificates**
  Alternatively, organisations may issue hardware-backed device certificates, such as a TPM. Many policy engines can verify the presence and validity of such certificates before granting access. This model can provide greater control over how device identity is issued, stored, and protected. An MDM may still be used for deployment, but the organisation retains oversight of the credential's security properties.

3. Device health signals

After establishing trusted user and device identity, policy engines can incorporate signals describing device security posture. These signals assess whether a device is securely configured and not actively compromised.

Examples of device health signals include:

- **MDM-compliance status**
  An MDM can assess compliance against defined configuration baselines. The policy engine may query compliance status before authorising access to a resource. However, compliance checks may not operate in real time and can lag behind changes made on the device.
- **Endpoint Detection and Response (EDR) risk score**
  Some EDR platforms can provide near real-time device risk signals to a policy engine. Where supported, this enables access decisions to reflect the device's current security state. Combining compliance status and EDR risk scores provides assurance of both the device's configuration and its current security state.

4. User behaviour signals

User behaviour signals assess patterns of activity, rather than static identity attributes. These signals can help detect anomalous or high-risk activity.
Examples of user behaviour signals include:

- **User risk scores**
  User risk scores quantify behavioural risk in near real time, based on factors such as unusual login times, impossible travel, or other anomalous activity. This capability is often provided by an IdP or Secure Access Service Edge (SASE) platform.
- **Sign-in location controls**
  Policies may restrict access based on geographic location or known high-risk regions. Location-based signals are relatively weak when used alone, but can strengthen decisions when combined with stronger baseline signals such as identity, device trust, and authentication strength.

Policy Engines

A policy engine evaluates contextual signals about an access request and applies policy to decide whether the request should be authorised.

Unlike traditional 'walled garden' architectures, which typically trust traffic based primarily on network location (for example, a VPN IP address), a policy engine evaluates multiple signals to make more explicit, context-aware authorisation decisions. This directly adheres to the NCSC's zero trust architecture design principle of using policies to authorise requests.

The specific signals and policies implemented will vary according to environment, threat model, and organisational risk appetite.

When designing a ZTNA solution, it's important to understand where policy decisions are made within the architecture. Two common approaches are:

1. Network-level policy engines.
2. Application-level policy engines.

Organisations may deploy one or both approaches depending on their requirements.

1. Network-level policy engine

A network-level policy engine controls which network destinations a client can reach. These capabilities are commonly provided by cloud-based access platforms such as VPN-as-a-Service, Secure Service Edge (SSE) or SASE.

The policy engine evaluates signals associated with the request and determines whether traffic may reach a network segment. If access is authorised, the policy engine instructs a policy enforcement point (such as a proxy or connector) to forward the traffic.

This approach limits network exposure prior to application interaction.

2. Application-level policy engine

An application-level policy engine authorises access to an application or service.

Typically, a client attempting to access an application is redirected to a policy engine, often provided by an identity provider or SSO service. The policy engine evaluates signals associated with the request and, if authorised, issues a short-lived token. The client presents this token to the application to gain access.

While legacy federation standards such as Security Assertion Markup Language (SAML) may still be used for browser-based human access, zero trust architectures increasingly favour OAuth 2.0 and OpenID Connect which better support API access, non-human identities, granular authorisation, and continuous policy enforcement.

Applying policy at different architectural layers

Access policies may be applied at different layers in ZTNA architectures.

Applying policy at multiple layers

In some architectures, access control is applied at multiple layers. For example:

- a network-level policy engine first controls which network segments traffic can reach
- an application-level policy engine then authorises access to specific applications, typically as part of an SSO process.

This layered approach can reduce application exposure while enabling more granular access control.

Applying policy at identity and application layers

In other architectures, particularly where organisations don't manage the underlying application-hosting environment or network boundary, access decisions are primarily made at the identity and application layers. In these cases, applications rely on strong authentication, short-lived tokens, and secure transport mechanisms.

Wherever access policies are applied, access should be:

- explicitly authorised
- based on multiple signals
- centrally logged and auditable

Illustrative examples of these approaches are provided in the reference architecture section of this guidance.

What to do before building a ZTNA architecture

There are several important steps to complete before you begin to build your Zero Trust Network Access (ZTNA) architecture, as outlined below. ZTNA is more effective when introduced deliberately, with a clear understanding of the existing organisational architectural environment, the threats it faces and how ZTNA will integrate into a wider zero trust architecture.

1. Checking organisational readiness

Before beginning, organisations should ensure they have a solid understanding of their current architecture. This will make design decisions easier, and help set realistic and well-defined goals for ZTNA implementation, avoiding the risk of carrying existing weaknesses into the new access model.

You should:

- identify the organisation's users, devices, services and applications
- understand all data flows and user journeys
- recognise architectural constraints, legacy systems, and dependencies
- understand risks through threat modelling activities

1.1 Updating the threat model

ZTNA should either maintain or improve an organisation's security posture. Introducing ZTNA represents a change in how access is granted and enforced. As such, organisations should review and, where necessary, update their threat model before implementation.

This includes understanding:

- how existing threats are mitigated
- how those mitigations change when moving away from existing solutions
- any new risks which may be introduced

1.2 Implementing a zero trust strategy

ZTNA is not designed to operate in isolation. Before adopting ZTNA, organisations should have in place – or be developing – a wider zero trust strategy that includes the full range of architectural components: identity, devices, applications, services, data, and monitoring. See the NCSC's guidance on demystifying zero trust and zero trust architecture design principles to understand more about the wider zero trust concept.

2. Establishing the prerequisites for ZTNA

ZTNA depends on reliable signals about users, devices and activity. These signals allow policy decisions to be made and enforced consistently. Before ZTNA implementation, organisations should ensure the following prerequisites are in place.

2.1 Identity prerequisites

Strong identity foundations are essential for ZTNA because identity is the primary control used when making access decisions.
Organisations should have:

- **An authoritative identity provider:** an Identity Provider (IdP) acting as the single source of truth for user identities.
- **Strong authentication:** phishing-resistant authentication methods such as FIDO2 credentials.
- **Consistent authentication:** single sign-on (SSO) across applications so that all access is subject to the same authentication strength and policy.
- **Defined user roles:** clear understanding of what users need access to, and how many levels of access are required.
- **Identity signals:** contextual signals from the IdP that can inform access decisions.
- **Identity governance:** defined processes for creating, modifying, and removing user accounts.
- **Timely de-provisioning:** user accounts are removed promptly when access is no longer required.
- **Auditable activity:** identity activity is centrally logged and available for monitoring and investigation.

2.2 Device prerequisites

Strong device foundations are essential for ZTNA to reliably enforce risk-aware access controls. Organisations should have:

- **Strong device identity:** a cryptographically backed device identity uniquely bound to the physical device, using secure hardware such as a Trusted Platform Module (TPM).
- **Centralised device management:** authoritative oversight of organisational devices through a centralised device management platform, such as a mobile device management (MDM) system or equivalent.
- **Baseline device security:** enforced baseline security controls in line with the NCSC's device security guidance, including device hardening, automatic patching, secure configuration, and minimising unnecessary exposure.
- **Device posture visibility:** visibility of device health and configuration status, including signals such as patch level, configuration state, and compliance with security policies.
- **Endpoint monitoring:** monitoring of device behaviour to inform access decisions, integrating where appropriate with endpoint security tooling such as Endpoint Detection and Response (EDR).
- **Device posture signals:** the ability for devices to provide signals that can be used when making access decisions.
- **Device lifecycle management:** clear ownership and management responsibility for devices, including processes for provisioning, ongoing management, and secure decommissioning.
- **Auditable device activity:** device activity is centrally logged and available for monitoring and investigation. See the NCSC's logging and protective monitoring guidance for further information on what should be logged.

2.3 Internet access controls

Before protecting organisational resources using a zero trust approach, organisations should first establish a secure baseline understanding of how users and devices connect to the internet. Addressing common risks associated with general internet use helps ensure that broader zero trust controls are applied on top of a stable and well-managed foundation.
Organisations should have:

- **Malicious domain blocking:** controls that prevent users and systems from accessing domains associated with known malicious activity.
- **Internet traffic policy controls:** the ability to enforce policy over internet traffic, for example, by restricting or blocking high-risk content based on characteristics such as file type, malware scan results, traffic behaviour or website category.

- **Security logging for internet activity:** comprehensive logs of internet access and security events that can support monitoring and alerting. These logs should be integrated into a centralised logging and monitoring capability, such as a security information and event management (SIEM) tool.

Organisations may choose to implement additional controls to strengthen internet access security, including:

- **PDNS services:** PDNS uses threat intelligence to prevent users from connecting to known malicious domains before a connection is established. See the NCSC guidance on PDNS for public and private sectors to find out more information.
- **Client-side enforcement:** controls on user devices, such as EDR solutions, which can enforce internet access policies and monitor user activity.
- **Integrated internet filtering within ZTNA services:** some ZTNA platforms also provide filtering for general internet browsing. These capabilities can help organisations meet the baseline controls described above while using a single access platform.

2.3.1 Deciding whether to tunnel internet traffic

Some ZTNA and Secure Access Service Edge (SASE) products encourage organisations to route, or tunnel, user internet traffic through the service. In this context, traffic from a user's device is forwarded to a central policy enforcement point – such as a SASE, Secure Service Edge (SSE), or VPN service – before reaching its destination.

As most modern internet services use Transport Layer Security (TLS), which the NCSC believes provides appropriate levels of protection for data in transit, adding an additional encrypted tunnel for general web traffic may provide limited additional security benefit. In some cases, the performance, usability and complexity trade-offs may outweigh that benefit.

Where applications or services still rely on unencrypted protocols, organisations should prioritise removing or replacing them. Tunnelling may provide interim protection while this is addressed.

Organisations may choose to tunnel internet traffic where it enables a specific security outcome that cannot be achieved by other means. For example, tunnelling may support advanced content filtering, centralised monitoring, or consistent enforcement of organisational policy. However, tunnelling all traffic should not be treated as a default design choice. Decisions about whether to tunnel should be informed by the organisation's threat model, the capabilities of the products in use, and the organisational risk appetite.

2.3.2. Considerations for TLS inspection

Organisations sometimes consider using TLS inspection as part of their approach to controlling and monitoring internet-bound traffic. Introducing TLS inspection can require encrypted connections to be terminated and re-established by an intermediary which has significant architectural and security implications. TLS inspection concentrates trust in a small number of systems and increases the impact of compromise, misconfiguration, or operational failure. Components that can decrypt large volumes of user traffic should therefore be treated as critical to the security of the system as a whole.

In many cases, the outcomes organisations seek from TLS inspection, such as detecting malicious activity, enforcing acceptable use, or applying security policy, can be achieved through alternative controls. For example, EDR, device posture signals, DNS protections, and application-level logging can provide meaningful visibility and enforcement while reducing the need to introduce additional trust dependencies.

In some cases, inspection of traffic may be necessary; however, this should be done deliberately and transparently, with a clear understanding of the security and privacy implications. In these cases, it is important to understand how TLS inspection is benefiting the organisation and whether alternative controls can result in the same outcome.

Where TLS inspection is deployed, organisations should recognise that they are placing a significant amount of trust in the inspecting component and the organisation operating it, including how decrypted data is handled, protected, and secured in operation. As with other critical security services, this trust should be supported by appropriate assurance, transparency, and supply-chain risk management.

TLS inspection should not be treated as a default component of internet access, and the trade-offs involved should be explicitly understood and documented.

3. Assessing ZTNA services and providers

Adopting ZTNA commonly involves introducing a third-party service into a critical access path for users, devices, and applications. This creates a dependency on an external provider and moves part of the organisation's trust boundary outside its direct control. As a result, selecting a ZTNA service should be treated as a risk management and assurance decision rather than solely a procurement activity.

Many vendors offer broad platforms that combine several security capabilities and may market these as delivering 'zero trust'. However, zero trust is an architectural approach rather than a single product. Purchasing a product alone won't deliver zero trust outcomes without appropriate architectural design and integration.

When assessing ZTNA services, organisations should focus on how a product supports their intended architecture and security objectives. They should also understand how the introduction of the service changes the trust model as several decisions may now be handled by a cloud provider.

Organisations should ensure they understand the trust they are placing in the provider, how that trust is justified, and how it will be maintained over time. For further information, see the NCSC's guidance on evaluating and trusting a cloud supplier.

Before selecting or deploying a ZTNA product, organisations should undertake the following measures:

3.1 Understanding the trust relationship

Trust aspect	What to consider
Delegated functions	This may include identity enforcement, policy decision-making, traffic handling, logging, and availability of access to critical services.
Service criticality and dependency	If the service is unavailable, organisations should understand which users and services are affected, and what fallback or contingencies exist.
Visibility and control	Organisations should understand what control they retain over policy configuration, logging and monitoring – and what remains internal to the provider's platform.

3.2 Gaining confidence in a product

Confidence measure	What to consider
Evaluation before commitment	Organisations should seek opportunities to test products through trials or proof-of-concept deployments, using realistic scenarios that reflect their own users, devices and applications.
Alignment with architectural intent	Evaluation should focus on whether the product supports the organisation's intended zero trust design, rather than what the product advertises.
Integration and dependencies	Understanding how the product integrates with identity providers, device management, and monitoring systems helps avoid unexpected constraints later.

3.3 Assurance over security and operation

Assurance measure	What to consider
Secure design and operation	Vendors should be able to provide appropriate assurance about how their service is designed, operated and secured, proportionate to the risk and criticality of the deployment.
Logging, monitoring and incident support	Organisations should understand what security logs are available, how they can be accessed, and how incidents involving the service will be detected and handled. See the NCSC's cloud security principle on audit information and alerting for customers for further guidance.
Shared responsibility	It should be clear which security responsibilities sit with the vendor, and which remain with the customer, such as configuration, policy management, and response to compromise. See the NCSC's cloud security principle on secure use of the service for further guidance.

How to implement ZTNA

Well-designed ZTNA relies not only on adopting new technologies, but on architectures that consistently enforce zero trust principles in practice. This section sets out the **8 key design requirements for ZTNA implementations**. Meeting these requirements helps ensure that ZTNA deployments move beyond implicit network trust and provide meaningful, measurable security benefits.

The requirements are grounded in the zero trust architecture design principles and are intended to show how those principles can be realised in a practical ZTNA architecture. For each requirement, links are provided to the relevant associated zero trust principles.

1. Transport network traffic securely

Network traffic must be transported securely, in line with the NCSC zero trust principle of: don't trust any network, including your own. ZTNA architectures must ensure that all network traffic between users, devices and applications is protected against interception, tampering and impersonation. Traffic should be cryptographically secure throughout its journey, regardless of whether the underlying network is private, public or internal to an organisation. Organisations must ensure this protection is consistently enforced.

Secure transport mechanisms

Organisations should ensure that only strong cryptographic protocols, cipher suites, and key lengths are used and that insecure or deprecated options are disabled. Typically, the traffic should be protected using secure protocols such as TLS or IPsec. Where applications already support strong TLS directly between the client and the application, traffic can be protected without additional tunnelling. TLS or IPsec should be configured in line with the NCSC's guidance on using transport layer security to protect data and using IPsec to protect data.

Secure transport as a supporting control

Secure transport is a foundational requirement that enables ZTNA, but alone does not imply trust. Even when traffic is encrypted and authenticated, access decisions should still be driven by policy engines and signals. Secure transport must operate in support of broader zero trust controls rather than replace them.

2. Enforce policy before granting access

In a ZTNA architecture, access to resources must be explicitly authorised by policy, before any connection is established, in line with the zero trust principles to use policies to authorise requests and authenticate and authorise everywhere. Users and devices should not gain access simply because they are connected to a corporate network, using a VPN, or operating from a known location. Instead, every access request should be evaluated against defined policies that reflect the organisation's security requirements, risk appetite, and threat model.

Just enough access

Enforcing policy before granting access ensures resources are not exposed to unauthorised or non-compliant users or devices. ZTNA solutions that only provide secure connectivity without enforcing policy decisions do not meet the intent of zero trust.
Users should only be able to reach the resources they are explicitly authorised to use, and no others. This reduces the attack surface and limits the impact of compromised credentials or devices.

Policy enforcement points

Depending on the architecture, policy enforcement may occur at either network or application layer (see the section on signals and policy engines for details). They can be used independently or together, but the security outcome should be consistent: applications are not exposed until policy has been satisfied.

3. Use multiple signals to inform access decisions

Access decisions in a ZTNA architecture should be informed by multiple signals, rather than rely on any single attribute such as network location, user identity, or device ownership.

Why multiple signals are needed

Using multiple signals is consistent with the zero trust principle to use policies to authorise requests. No individual signal is sufficiently reliable on its own. Combining signals provides greater confidence in access decisions and makes it harder for attackers to bypass controls.

Signals should collectively describe:

- who is requesting access
- what device they are using
- whether the device is sufficiently secure

This allows organisations to make proportionate, risk-based decisions that reflect both the sensitivity of the application and the threat environment.

Core signals

As outlined in the signals and policy engine section of this guidance, at a minimum, organisations should establish a baseline set of signals that support strong assurance. In most environments this will include:

- user identity – to establish who is requesting access
- device identity – to confirm the request originates from a known or managed device
- device health – to assess whether the device is in an acceptable security state
- user behaviour – to help identify anomalous or high-risk behaviour

Selecting and managing signals

The exact combination of signals used will vary depending on the organisation's architecture, applications and risk appetite. Signals should be selected deliberately, with a clear understanding of what each signal contributes to the access decision and its limitations. The choice of signals should not be static. As threats evolve and new capabilities become available, organisations should review and adapt the signals used to inform access decisions.

Signals inform access policy decisions, but you should avoid excessive policy granularity, as it can quickly become difficult to manage and maintain. Where possible, access policies should be standardised into a small number of well-defined categories, such as guest access, standard business access, and access to critical applications.

4. Minimise exposure of application

ZTNA architectures should minimise the exposure of applications by ensuring they are not directly reachable from untrusted networks. Reducing application exposure limits attack surface, constrains attacker reconnaissance, and is in line with the zero trust concept of least privilege. Applications should only be reachable after access has been authorised, and exposure should be limited to the specific applications a user is permitted to access. Broad network access, even when encrypted, undermines ZTNA objectives and increases the potential impact of credential compromise.

Private applications

For privately-hosted applications (for example on-premises, private cloud, or private networking in a public cloud), access should be mediated using components designed to be exposed to an untrusted network such as an outbound-only connector, proxy or VPN endpoint.

Where possible, these components should be deployed close to the applications they protect and scoped narrowly, limiting which applications or services they can reach. This reduces the risk of lateral movement if a connector or user session is compromised.

SaaS applications

For SaaS applications, organisations typically don't control the underlying network or hosting environment. In these cases, exposure should be managed at the identity and application layer rather than the network layer.

Access to SaaS applications should be integrated with enterprise single sign-on (SSO) and protected by the application-level policy engine. This ensures that access decisions are made during authentication and users only obtain access when organisational policy is satisfied. Organisations should set permissions following the principle of least privilege and this should be a feature of the SaaS application, not the policy engine.

Even though SaaS applications are by design internet-facing, ZTNA principles still apply. Access should be mediated, explicitly authorised and limited to what is necessary.

5. Limit the impact of compromise

ZTNA architectures should be designed on the assumption that network compromise will occur and that the impact of such a compromise needs limiting. This is achieved by minimising blast radius – the scope of systems and data exposed if a user, device or component is compromised – and by restricting lateral movement between segments.

Network segmentation

Network segmentation is fundamental to minimising impact of compromise in ZTNA private access solutions. Large, flat networks without segmentation allow attackers to bypass security controls and are considered an anti-pattern. The appropriate segmentation model will vary depending on organisational structure, application architecture, and risk profile.

Each application should reside in an appropriately-sized network segment based on risk and business criticality. Access to each segment should be mediated through a connector, proxy or network security device (such as a VPN appliance, physical or virtual firewall), under the control of a network-level policy engine. This ensures that traffic can only reach a segment when access has been explicitly authorised. These components should be deployed in tightly-constrained environments, with defence-in-depth controls.

ZTNA controls don't remove the need for local network segmentation. ZTNA determines which segments a user can access, while traditional controls – such as firewalls, security groups, and network access control lists – restrict movement between segments after access is granted.

Segmentation model

In an ideal design, each application resides in its own isolated network segment, but this is often impractical and doesn't scale well. A more realistic and widely adopted approach is to group applications based on risk or business criticality, for example:

- **Low-risk, broadly accessible applications:** internal services, such as company news portals or restaurant menus, can share a segment since compromise or temporary unavailability would have limited impact.

- **Medium-risk, department-specific applications:** systems used by specific business functions (for example, HR, Finance, or Engineering) reside in smaller, dedicated segments to limit exposure.
- **Critical business applications:** services essential to business continuity or processing sensitive data, such as payroll systems, customer databases, or repositories of sensitive documents, are isolated in dedicated segments.

Organisations should consider using automation techniques, such as Infrastructure as Code (IaC), to define and manage network segments, access rules, and dependencies. This supports segmentation at scale and can reduce the operational burden of maintaining many small segments, making it easier to review, test, and evolve segmentation decisions over time.

6. Continuously re-evaluate access as context changes

ZTNA solutions should support contextual re-evaluation of a policy during a session. User behaviour, device posture, and risk conditions can change over time, and access decisions should reflect this. Where the conditions required for access are no longer met, access should be restricted or revoked. This reduces the window of opportunity for attackers and limits the impact of compromised credentials or devices.

Token lifetimes and policy re-evaluation

Using short-lived credentials or tokens is a common way to support continual policy enforcement without significantly affecting the user experience. When access tokens have a limited lifespan, policy engines are forced to re-evaluate policy before issuing a new token. Token lifetimes should be proportionate to the sensitivity of the application and the organisation's risk appetite.

Event-driven enforcement

Where available, organisations should use near real-time or event-driven enforcement. These allow accesses to be actively terminated if contextual conditions change, for example, if a device falls out of compliance, a user risk score increases, or suspicious activity is detected. They also reduce reliance on token expiry alone.

7. Ensure ZTNA architectures are observable

ZTNA architectures should be designed to be observable.

Importance of visibility

This visibility enables organisations to, for example:

- understand how access decisions are made
- detect suspicious activity
- support investigation and incident response

ZTNA solutions should generate high-quality, security-relevant logs for both successful and unsuccessful access attempts. These logs should include sufficient context to explain why an access decision was made, for example the identity involved, the application accessed, the time of the request, and the policy or authentication method used to make the decision.

ZTNA architectures are composed of multiple components, such as identity providers, policy engines, connectors, proxies, endpoint security tools, and SaaS platforms. Organisations should ensure they can collect and correlate logs from these different sources to build a coherent view of user activity and access decisions. Without this visibility, activity that spans multiple systems may be difficult to detect or investigate.

Key logging events

At a minimum, ZTNA architectures should log the following events:

- **Successful access requests**, including the identity involved, the application accessed, and the time and method of authorisation.
- **Denied or blocked access attempts**, including the reason for denial where available.
- **Changes in signals or context** that affect access decisions, such as a device becoming non-compliant or a change in user risk level.

Logs should be centralised into a monitoring capability such as a security information and event management (SIEM) platform to support correlation and analysis. Organisations should retain logs in line with operational and regulatory requirements, protect them from unauthorised access or tampering, and ensure they are actively monitored and reviewed.

Effective observability supports both security and operational assurance. Logging and monitoring enable organisations to validate that access policies are working as intended and identify overly-restrictive controls. This helps organisations to refine and improve their ZTNA implementation. Further information on developing logging use cases is available in the NCSC guidance for Security Operations Centres on onboarding systems and log sources.

8. Design for resilience and availability

ZTNA architectures should be designed to maintain service availability while enforcing strong access controls. If policies are overly restrictive or incorrectly configured, legitimate users may be blocked from the services they need. This can disrupt business operations and reduce confidence

in the access control system. Availability should therefore be treated as a core design consideration.

Testing and safe policy rollout

Organisations should design ZTNA policies with the expectation that failures will occur. Devices may temporarily fall out of compliance, identity attributes may change, or security signals may become unavailable. ZTNA architectures should handle these situations safely, allowing users to recover access without weakening security controls.

Where possible, organisations should test policies before enforcing them. Testing allows organisations to understand how policies affect legitimate access and identify unintended consequences before users are impacted. Phased or conditional rollout of new policies can further reduce the risk of widespread disruption.

Recovery mechanisms

ZTNA designs should also include clear recovery paths for common access failures. For example, organisations should consider:

- allowing non-compliant devices to access remediation services (such as update or patching systems), rather than blocking access entirely
- providing clear user or support processes to resolve access problems quickly
- ensuring support teams have sufficient visibility to diagnose policy-related access failures

Organisations should also plan for recovery from more serious failures, such as misconfiguration or service outage. Managing ZTNA infrastructure and policy configuration using IaC can help support this. Defining ZTNA components, access policies, segmentation rules and dependencies as code enables organisations to:

- restore environments quickly and consistently after failures
- reduce configuration drift across environments
- manage changes safely through version control, review and rollback

Emergency access

Organisations should implement a break glass or emergency access mechanism for use during incidents or recovery scenarios. Break glass access should be tightly restricted to designated accounts and devices, and used only when normal access mechanisms are unavailable.

Organisations should ensure that such access:

- bypasses standard policy enforcement only when necessary
- triggers heightened monitoring and alerting
- is tested regularly to ensure it functions as intended
- is subject to strict governance, auditing, and post-incident review

Designing for availability helps ensure ZTNA controls remain effective and trusted. By planning for failure, testing policies carefully, and providing safe recovery mechanisms, organisations can enforce zero trust principles without unnecessarily preventing legitimate access.

ZTNA reference architectures

The Zero Trust Network Access (ZTNA) reference architectures provide deliberately simplified illustrative examples of how to implement ZTNA. The examples are not prescriptive, but are an intended guide to inform architecture and design decisions. Organisations should adapt them to meet their particular environment, risk appetite, and security requirements.

The reference architectures detail the design requirements to connect users to:

1. Private applications
2. Software as a Service (SaaS) applications

Before beginning design, organisations should have read all sections of this guidance. As a minimum, organisations must have completed the user, device and internet foundation steps to help build the requisite signals for user and device identity, behaviour and health. In addition, organisations should ensure they are familiar with the common pitfalls to be avoided, described in the anti-patterns section.

1. Connecting to a private application

In the context of this guidance, a private application is an application or service that is not exposed to the public internet. These resources may be hosted on-premises or within customer-managed cloud environments.

Common examples of private applications include internal business systems such as HR platforms, finance systems, software development and test environments, and administrative or management interfaces.

In this section, a reference architecture is used to describe how access to private applications can be brokered using network-level components. This is just one example, and in practice, similar outcomes can also be achieved using application proxy approaches, where access is provided at the application layer.

Architecture overview

Figure 1 shows that the user's access begins on a managed end user device, which establishes a secure tunnel to a network policy engine. To reduce their attack surface, the private applications are not hosted on the internet, so a tunnel is required to ensure traffic can reach the policy engine and the application. This tunnel may be provided using a VPN, VPN-as-a-Service or a cloud platform such as Secure Service Edge (SSE) or Secure Access Service Edge (SASE).

At the edge of each network segment, a component designed to be exposed to the public internet such as a connector, proxy or a VPN endpoint is deployed and will establish outbound (reverse) tunnels to the policy engine. This allows access to be brokered without requiring inbound network connectivity to the application or hosting environment.

When a user attempts to access a private application, the network policy engine evaluates the request. It assesses signals such as user identity, device identity, device health, and user behaviour indicators, and compares them against defined access policies.

If the network-level policy requirements are satisfied, traffic is permitted through the secure tunnel to the network segment hosting the application. At this stage, the network policy engine grants access to the relevant network segment rather than directly to the application itself.

Access to an application is then controlled by a separate application policy engine. This typically takes place during the organisation's single sign-on (SSO) process. The application policy engine verifies that the authenticated user is authorised to access the specific application, evaluating signals such as user and device identity, device health, and behavioural indicators.

Private resources are hosted in network segments that are separated according to their business criticality. Local network controls – such as firewalls (virtual or physical) or network access control lists – prevent traffic flowing between segments, unless explicitly permitted. This helps to limit lateral movement and reduce the potential impact of compromise.

Network segments should be kept as small as is practical, balancing security benefits with operational manageability. Where possible, automation techniques such as infrastructure as code should be used to support the creation and management of smaller, more granular network segments.

All architectural components – including client software, policy engines, connectors and proxies – generate security and access logs. These logs should be collected centrally to provide visibility of

access decisions, user behaviour, and system health. Centralised logging supports monitoring, investigation, and audit activities.

Availability should be considered throughout the design. Policies should be monitored and tested regularly to reduce the risk of unintended access disruption. Controlled break-glass access mechanisms must be implemented to support recovery during exceptional circumstances and they must be tightly governed, monitored, and regularly reviewed. Configurations should be deployed using infrastructure as code as this supports automated restoration of the environment in the event of a disaster.

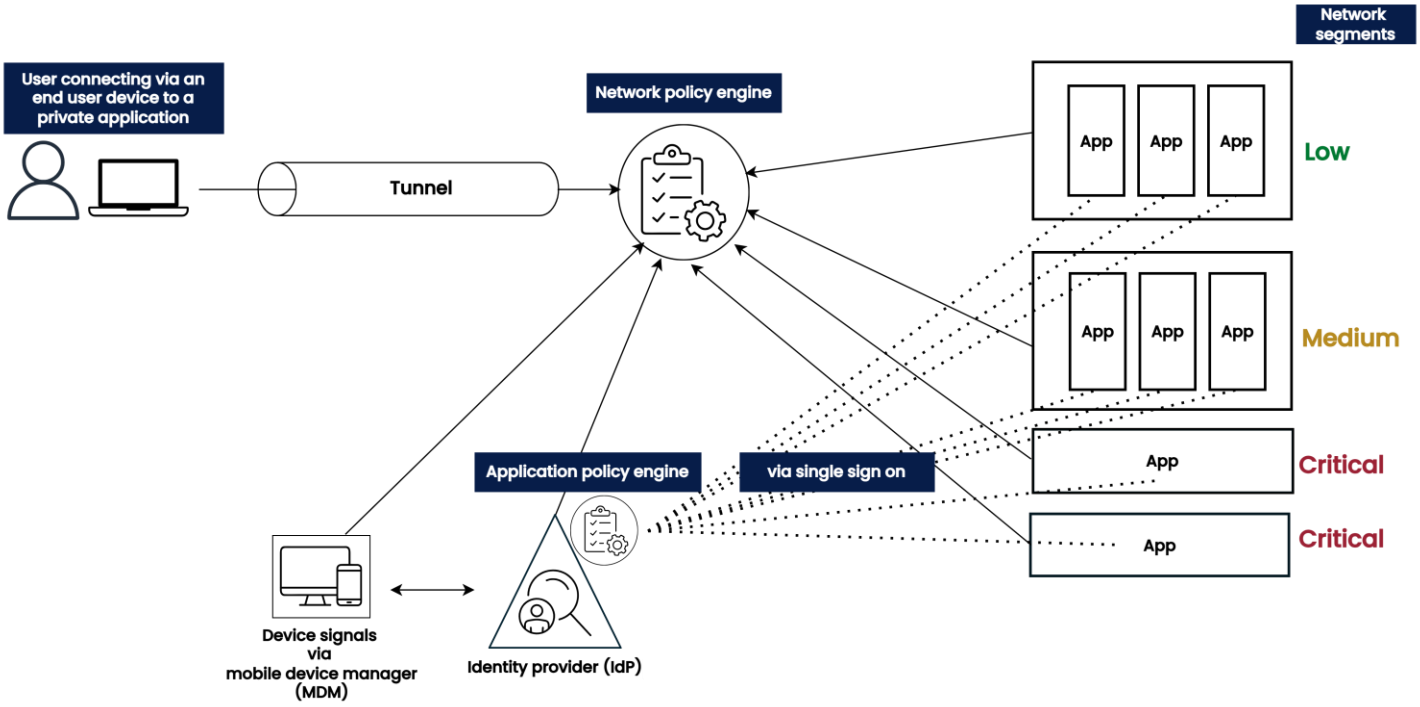


Figure 1: A visual representation of the steps taken by an end user device to connect to a private application, as described in the architecture overview.

2. Connecting to SaaS

Organisations increasingly rely on SaaS and productivity applications that are hosted in the cloud. These products are often used to store and process sensitive data.

Architecture overview

Figure 2 shows that a client device initiates a connection to a SaaS or productivity application using a secure transport mechanism, such as TLS. Rather than granting access immediately, the application redirects the user to the organisation's SSO process for authentication and authorisation.

During the SSO process, the access request is evaluated by an application policy engine. Centralising this decision point ensures that access policies are applied consistently, regardless of the user or device location or network.

The application policy engine evaluates the access request and assesses multiple signals such as user identity, device identity, device health, and user behaviour indicators, and compares them against defined access policies.

If the request satisfies the defined policy, the user is issued a short-lived access token that permits access to the application. Tokens are intentionally time-limited so that access is not assumed to remain valid indefinitely.

Ideally, before the access token expires, signals should be re-evaluated before the user session is extended. This ensures that continued access reflects the current risk associated with the user and device. Where supported, investigate using context-aware access features that can automatically block or revoke access if the user's context or device state changes.

In this approach, all routine user access to SaaS and productivity applications is routed through the application policy engine. A tightly controlled break-glass mechanism provides a limited exception for use during exceptional circumstances, such as service disruption or incident recovery.

All components involved in SaaS access generate security and access logs. These logs should be collected centrally to provide visibility of access decisions, user behaviour, and system health, supporting monitoring, investigation, and audit activities.

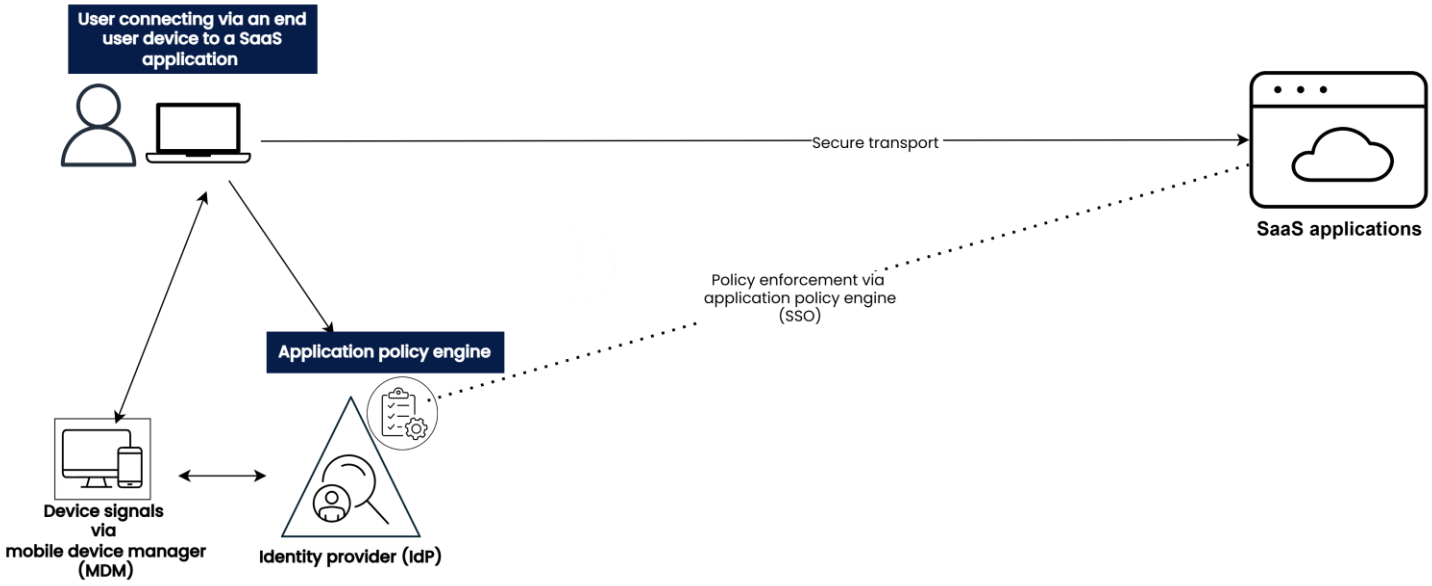


Figure 2: A visual representation of the steps taken by an end user device to connect to a SaaS application, as described in the connecting to SaaS architecture overview.

ZTNA anti-patterns

The security benefits of a ZTNA architecture can be undermined by poor design and implementation choices that conflict with zero trust principles. These designs are referred to here as 'anti-patterns'.

The following 4 anti-patterns highlight common architectural flaws and misconfigurations, outline the risks they introduce, and suggest alternative design approaches.

This section will help organisations design, deploy, and operate ZTNA solutions that deliver meaningful risk reduction, and align with our ZTNA guidance.

1. Anti-pattern: recreating the 'walled garden' trust model

As illustrated in Figure 1, this anti-pattern occurs when components such as Secure Service Edge (SSE) or Secure Access Service Edge (SASE) are deployed as a direct replacement for traditional VPNs in a 'walled garden' model, but the underlying access model remains unchanged. Users authenticate at connection time and are then granted broad network access.

This design has the following features:

- Access is based on network reachability rather than explicit per-application policy evaluation.
- Once connected, users can discover and access multiple services.
- Lateral movement is largely unconstrained.
- Network location is implicitly treated as a trust signal.

This recreates the 'walled garden' model that ZTNA is intended to replace.

Why this is a problem

If a user account or device is compromised, an attacker may inherit broad internal access. The resulting blast radius may be little different from a traditional 'walled garden' model.

Recommended approach

In a ZTNA architecture, access should be:

- explicitly authorised by policy
- scoped to the minimum set of resources required

> Delivering meaningful risk reduction typically requires architectural change, including segmentation and policy-driven access control, not just a change in remote access technology.

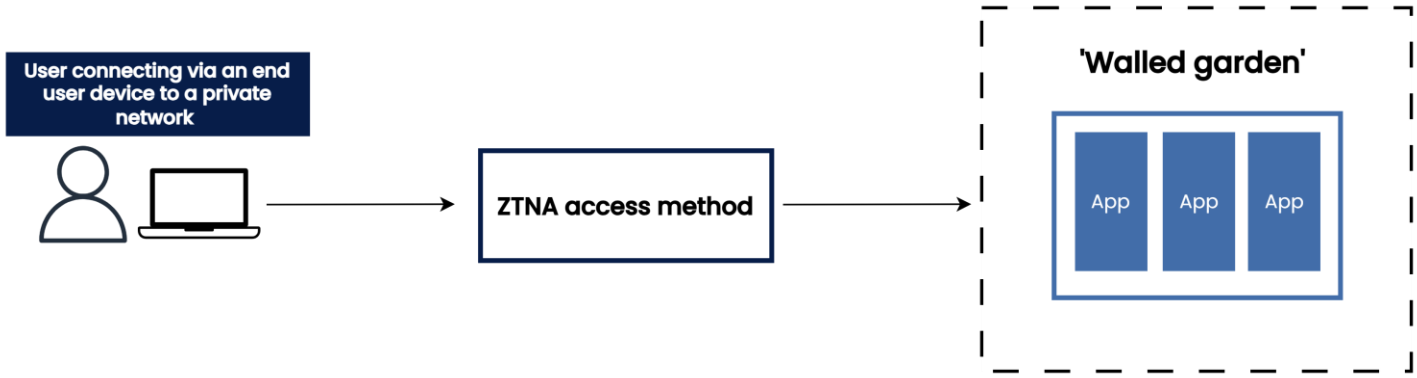


Figure 1: A visual representation of an end user device using a ZTNA access method (such as a Secure Service Edge (SSE) or Secure Access Service Edge (SASE)) to access a private network. Once inside the network, the user has access to all applications, thus recreating the 'walled garden' model.

2. Anti-pattern: trusting network location

As shown in Figure 2, this anti-pattern occurs when access to an application or service is granted solely because a request originates from a 'trusted' IP address or network location, rather than being evaluated by a policy engine using multiple security signals.

Examples include:

- implicitly trusting corporate office IP ranges and VPN address pools
- allowing internal cloud network ranges to bypass conditional access policies

Why this is a problem

An IP address does not provide assurance of:

- the user's identity
- authentication strength
- device compliance or health
- user risk

Relying on network location reintroduces implicit trust and recreates a perimeter-based model, which is inconsistent with zero trust principles.

If an attacker gains access to a trusted network location, they may be able to access resources without policy engine evaluation, increasing the risk of unauthorised access and lateral movement.

Recommended approach

All access requests should be evaluated by the policy engine regardless of their source IP address or location.

Network location may be used as a supporting signal to inform policy decisions, but it must not act as a standalone trust decision or provide a route to bypass policy enforcement.

> Access decisions should be based on multiple signals, including user identity, authentication strength, device posture, and user behaviour risk.

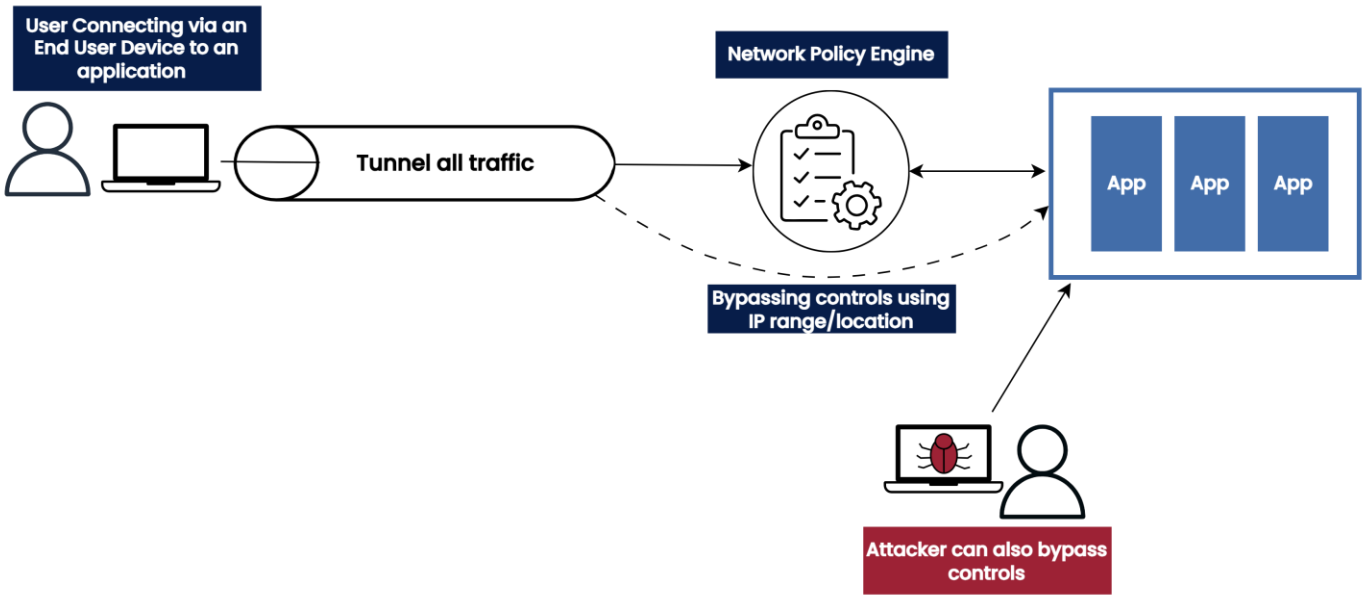


Figure 2: A visual representation of an end user device getting access to the private network after bypassing the policy engine. It has been permitted this bypass because it originates from a trusted IP address or network location.

3. Anti-pattern: exposing private applications to the internet

As shown in Figure 3, this anti-pattern occurs when internal services that are not designed to be internet-facing are made directly accessible from the public internet.

Examples include:

- publishing internal applications with public IP addresses
- exposing management interfaces to the internet
- making private APIs directly reachable from untrusted networks

Why this is a problem

Even where a policy engine controls access, vulnerabilities in the private service itself may be exploited before policy enforcement occurs – for example, through authentication bypass flaws. In these cases, the policy engine is effectively circumvented and the benefits of ZTNA are lost.

Recommended approach

Private services should not be directly exposed to the internet. Access should be brokered through a hardened internet-facing component, such as a connector or proxy, that is specifically designed to operate on untrusted networks. Only this component should be publicly exposed, not the application itself.

Such components should have the following features:

- Do not require services to have public IP addresses.
- Permit access to a service only when the access policy is satisfied.
- Minimise network-level exposure of private applications.

> The external attack surface should be reduced and all access explicitly authorised and policy-driven.

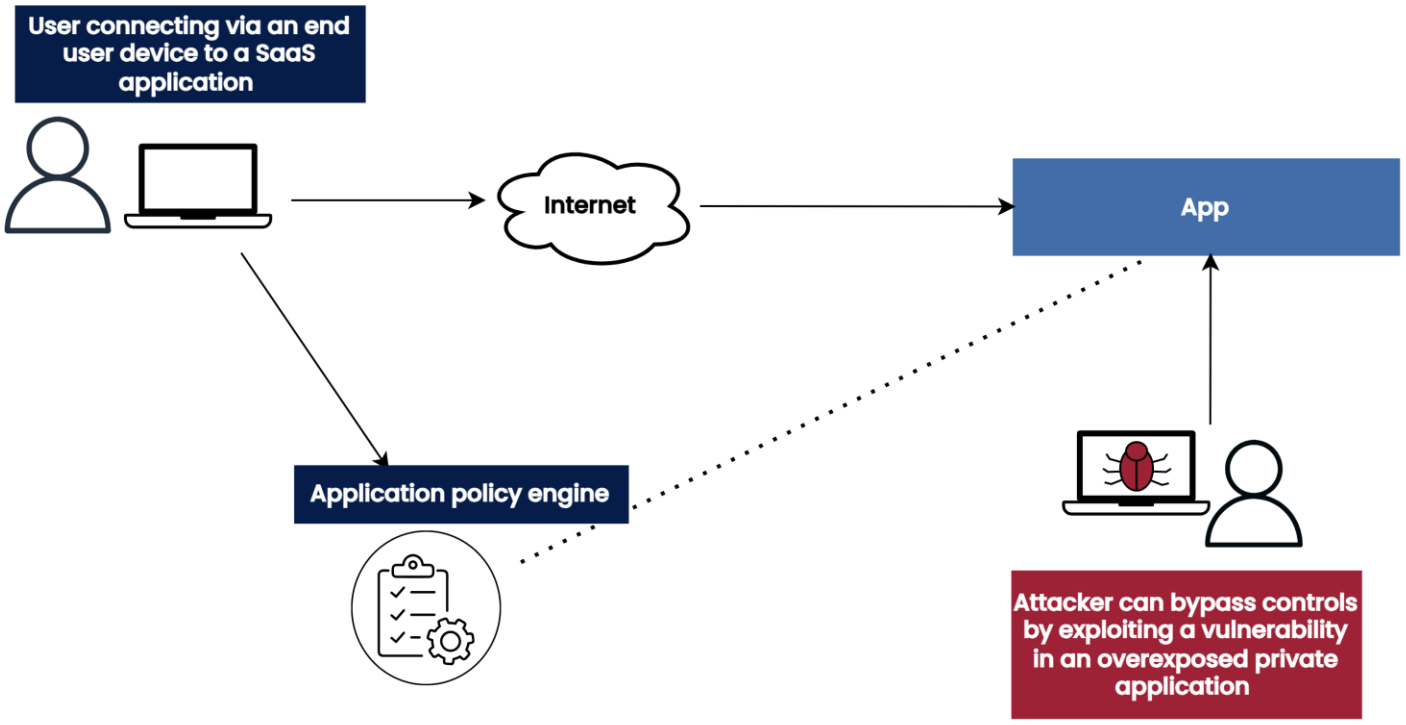


Figure 3: A visual representation of an attacker gaining access to a private application and bypassing the policy engine by exploiting a vulnerability in the app itself, due to it being exposed on the public internet

4. Maintaining legacy access routes that bypass the policy engine

As shown in Figure 4, this anti-pattern occurs when legacy access mechanisms are retained alongside a ZTNA solution, allowing users or systems to bypass the policy engine.

Examples include:

- VPN access that provides broad network connectivity
- direct network routing to internal services
- static firewall rules permitting access without policy evaluation

These routes are often maintained for operational convenience, to support legacy systems, or phased migration.

Why this is a problem

Legacy access routes reintroduce implicit trust and undermine the security benefits of ZTNA. Maintaining parallel access paths also increases architectural complexity, making it harder to understand, audit, and govern who can access what.

Recommended approach

All access to corporate resources should be mediated by the policy engine. Legacy access routes should be removed wherever possible. Where they must be retained, access should be tightly restricted to the minimum necessary.

> Access decisions should be consistent, centrally governed, and aligned with the access policy.

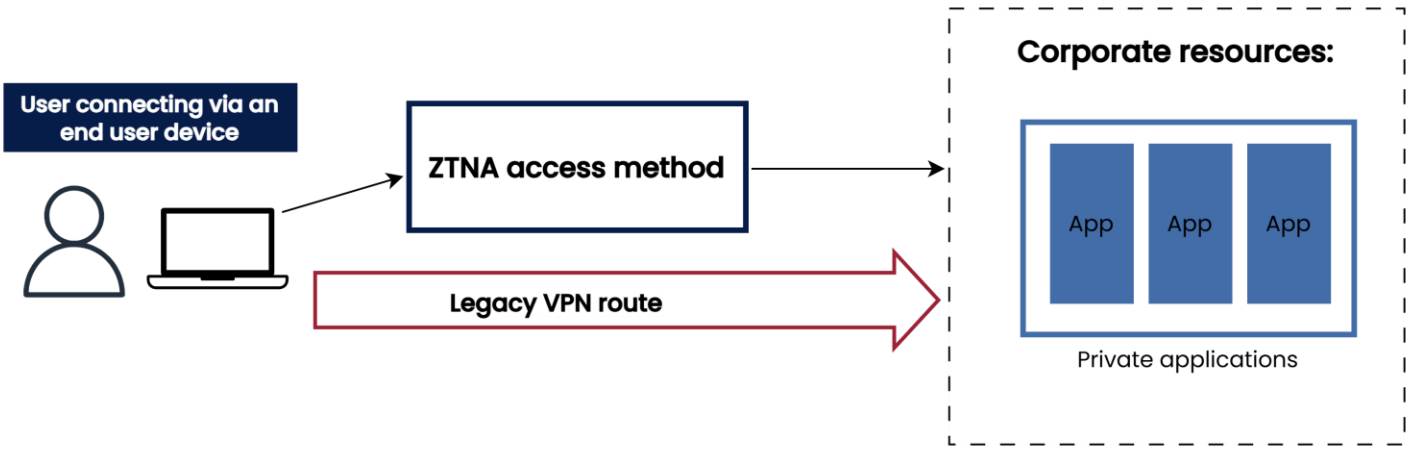


Figure 4: A visual representation of an end user device getting access to private applications in a network via two different routes. The first route is via the ZTNA access method; the second is via the legacy access route, such as a VPN.

© Crown copyright 2026. Photographs and infographics may include material under licence from third parties and are not available for re-use. Text content is licenced for re-use under the Open Government Licence v3.0.
(https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/)