Workload Identity in Multi System Environments
Internet-Draft
Intended status: Informational
Expires: 5 July 2025

J. Salowey
Venafi
Y. Rosomakho
Zscaler
H. Tschofenig
1 January 2025

Workload Identity in a Multi System Environment (WIMSE) Architecture
draft-ietf-wimse-arch-03

Abstract

The increasing prevalence of cloud computing and micro service architectures has led to the rise of complex software functions being built and deployed as workloads, where a workload is defined as a running instance of software executing for a specific purpose. This document discusses an architecture for designing and standardizing protocols and payloads for conveying workload identity and security context information.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Workload Identity in Multi System Environments Working Group mailing list (wimse@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/wimse/.

Source for this draft and an issue tracker can be found at https://github.com/jsalowey/wimse-arch.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 July 2025.

Table of Contents

1. Introduction

The increasing prevalence of cloud computing and micro service architectures has led to the rise of complex software functions being built and deployed as systems composed of workloads, where a workload is defined as a running instance of software executing for a specific purpose.

Workloads need to be provisioned with an identity when they are started. Often, additional information needs to be provided, such as trust anchors and security context details. Workload identity credential is used to authenticate communications between workloads. Workloads make use of identity information and additional context information to perform authentication and authorization.

This architecture considers two ways to express identity information: X.509 certificates often used in the TLS layer and JSON Web Tokens (JWTs) used at the application layer. Collectively, these are referred to as WIMSE tokens. The applicability of given token format depends on application and security context and will be explored in later sections.

Once the workload is started and has obtained identity information, it can start performing its functions. Once the workload is invoked it may require interaction with other workloads. An example of such interaction is shown in [I-D.ietf-oauth-transaction-tokens] where an externally-facing endpoint is invoked using conventional authorization mechanism, such as an OAuth 2.0 access token. The interaction with other workload may require the security context associated with the authorization to be passed along the call chain.

In the rest of the document we describe terminology and use cases, discuss details of the architecture, and discuss threats.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the following terms:

* Workload

A workload is a running instance of software executing for a specific purpose. Workload typically interacts with other parts of a larger system. A workload may exist for a very short durations of time (fraction of a second) and run for a specific purpose such as to provide a response to an API request. Other kinds of workloads may execute for a very long duration, such as months or years. Examples include database services and machine learning training jobs.

* Security Context

A security context provides information needed for a workload to perform its function. This information is often used for authorization, accounting and auditing purposes and often contains information about the request being made. Some examples include user information, software and hardware information or information about what processing has already happened for the request. Different pieces of context information may originate from different sources.

* Identity Proxy

Identity proxy is an intermediary that can inspect, replace or augment workload identity and security context information. Identity proxy can be a capability of a transparent network service, such as a security gateway, or it can be implemented in a service performing explicit connection processing, such as an ingress gateway or a Content Delivery Network (CDN) service.

* Attestation

Attestation is the function through which a task verifies the identity of a separate Workload. (TBD: sync definition with reference RaTS architecture)

* Workload Identity Token

A token that contains a workload identifier used for service to service authentication. The token is bound to a cryptographic key and requires that the presenter provide proof of possession of the secret key material. This token is further defined in [I-D.ietf-wimse-s2s-protocol]

3. Architecture

3.1. Workload Identity

Workload identity construct consists of three basic building blocks: trust domain, workload identifier and workflow identity credentials. These components are sufficient for establishing authentication, authorization and accounting processes. More complex identity constructs can be created from these basic building blocks.

3.1.1. Trust Domain

A trust domain is a logical grouping of systems that share a common set of security controls and policies. Workload certificates and tokens are issued under the authority of a trust domain. Trust domains SHOULD be identified by a fully qualified domain name associated with the organization defining the trust domain. THe FQDN format of trust domain helps to ensure uniqueness of the trust domain identifier. A trust domain maps to one or more trust anchors for validating X.509 certificates and a mechanism to securely obtain a JWK Set [RFC7517] for validating WIMSE WIT tokens. This mapping MUST be obtained through a secure mechanism that ensures the authenticity and integrity of the mapping is fresh and not compromised. This secure mechanism is out of scope for this document.

A single organization may define multiple trust domains for different purpose such as different departments or environments. Each trust domain must have a unique identifier. Workload identifiers are scoped within a trust domain. If two identifiers differ only by trust domain they still refer to two different entities.

3.1.2. Workload Identifier

The WIMSE architecture defines a workload identifier as a URI [RFC3986]. This URI is used in the subject fields in the certificates and tokens defined later in this document. The URI MUST meet the criteria for the URI type of Subject Alternative Name defined in Section 4.2.1.6 of [RFC5280].

> The name MUST NOT be a relative URI, and it MUST follow the URI syntax and encoding rules specified in [RFC3986]. The name MUST include both a scheme and a scheme-specific-part.

In addition the URI MUST include an authority that identifies the trust domain within which the identifier is scoped. The trust domain SHOULD be a fully qualified domain name belonging to the organization defining the trust domain to help provide uniqueness for the trust domain identifier. The scheme and scheme specific part are not defined by this specification. An example of an identifier format that conforms to this definition is SPIFFE ID (https://github.com/spiffe/spiffe/blob/main/standards/SPIFFE-ID.md).

While the URI encoding rules allow host names to be specified as IP addresses, IP addresses MUT NOT be used to represent trust domains except in the case where they are needed for compatibility with existing naming schemes.

A workload identity only has meaning within the scope of a specific issuer. Two identities of the same value issued by different issuers may or may not refer to the same workload. In order to avoid collisions identity URIs SHOULD specify, in the URI's "authority" field, the trust domain associated with an issuer that is selected from a global name space such as host domains. However, the validator of an identity credential MUST make sure that they are using the correct issuer credential to verify the identity credential and that the issuer is trusted to issue tokens for the defined trust domain.

3.1.3. Workload Identity Credentials

The Agent provisions the identity credentials to the workload. These credentials are represented in form of JWT tokens and/or X.509 certificates.

JWT bearer tokens are presented to another party as a proof of identity. They are signed to prevent forgery, however since these credentials are often not bound to other information it is possible that they could be stolen and reused elsewhere. To mitigate these risks and make the token more generally useful the WIMSE architecture defines a workload identity token that binds a JWT to a cryptographic key.

Both X.509 certificate and workload identity token credentials consist of two parts:

* a certificate or WIT is a signed data structure that contains a public key and identity information

* a corresponding private key

The certificate or WIT is presented during authentication, however the private key is kept secret and only used in cryptographic computation to prove that the presenter has access to the private key corresponding to the public key.

3.2. Workload Identity System Scenarios

3.2.1. Basic Workload Identity Scenario

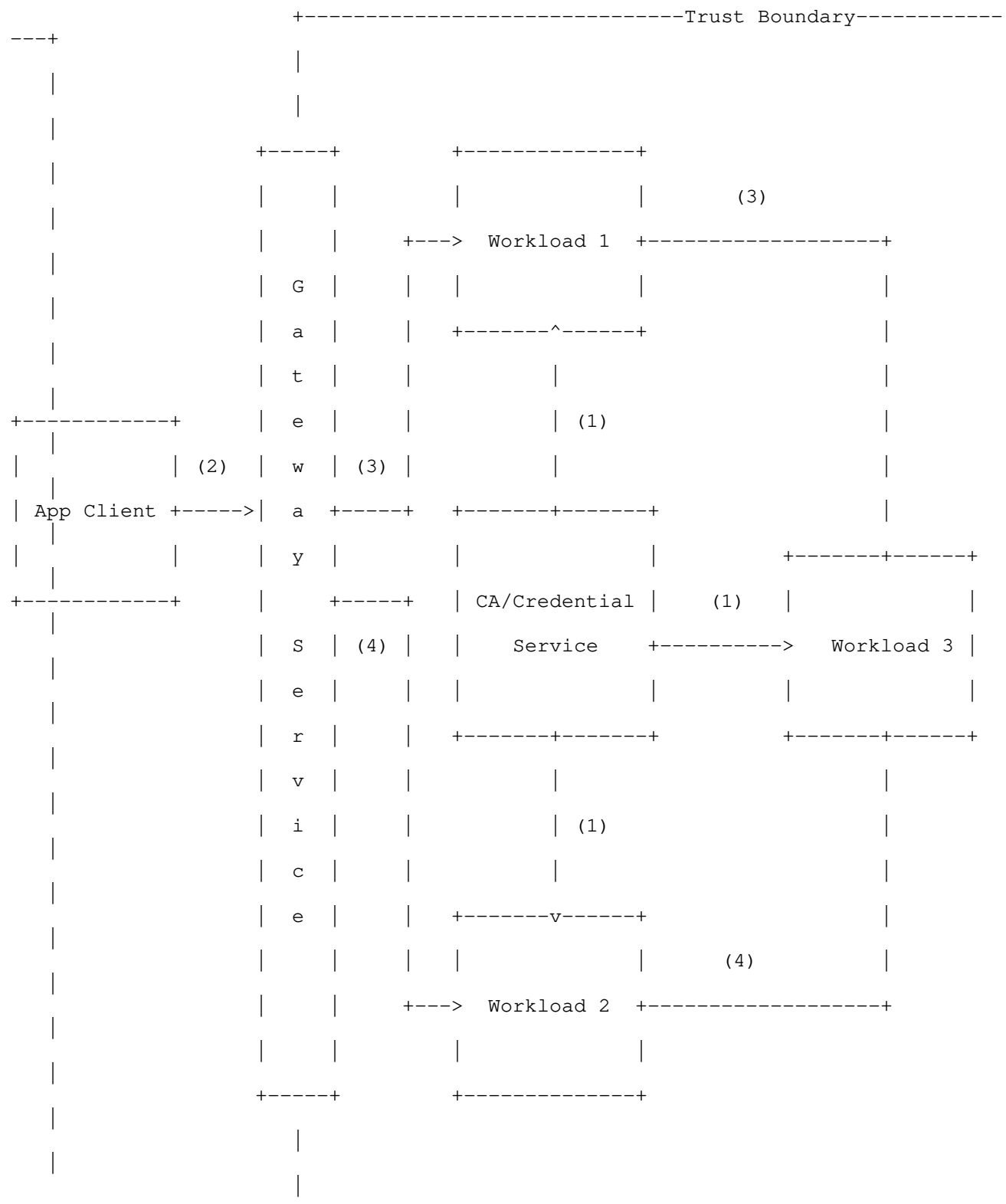

Figure 1: Basic example workload application system.

The above diagram presents a basic workload application system. The large box represents a trust domain within which the workload application is hosted. Within this example there are three workloads, a gateway, that accepts external clients and a CA/ credential service that issues workload identity credentials for the trust domain. External to the workload application system there is an application client that calls APIs on workloads.

Here is a brief summary of each component

* Trust Domain

The large box represents a trust domain of the application that is composed of several workloads. A trust domain may have a more complex internal structure with more workloads, multiple gateways, internal infrastructure, and other services.

* Workload

Three workloads are shown. Each workload is an instance of running software executing for a specific purpose. Workloads obtain their identity credentials from a Credentials Service (1) and use them to authenticate to other workloads and systems in the process of sending and receiving requests to and from external systems or other internal workloads.

* Gateway Service

A gateway service typically acts as an intermediary between the internal application trust domain and external systems. The gateway is responsible for ensuring appropriate isolation between external and internal domains. It also routes incoming requests to the correct workload. The gateway may also handle authentication, token exchange, and token transformation.

* CA/Credential Service

In this diagram the token/Credential service is a service responsible for issuing workload identities to workloads in the same trust domain. The credentials are often X.509 based or JWT based.

High level flows within the diagram

* (1) Workload Identity Credential Distribution

Workloads typically retrieve their workload identity credentials early in their lifecycle from a credentials service associated with their trust domain. The protocol interaction for obtaining credentials varies with deployment and is not detailed here.

* (2) Application client Requests

Clients send API requests to the application. In the example above, the gateway routes the request to the correct workload. In addition, the gateway may assist in authenticating the incoming request and provide information resulting from the authentication to the target

workload. The authentication exchange is not covered in detail in this example. The client request is typically made over HTTPS, but other protocols may be used in some systems. The gateway usually terminates the TLS session so it has visibility into the request in order to route it correctly.

* (3) API request to workload 1

The gateway is configured to forward requests to the correct workload. The gateway often modifies the request to include specific authentication information about the application client and to remove any information that should not be forwarded internally. The gateway authenticates the workload before forwarding the request. This authentication usually uses TLS. The target workload may authenticate the gateway using TLS or some other means. As part of servicing the request the workload must make a request to another workload in the system. In this scenario the workload is making a request to workload 3 over HTTPS. Workload 1 may be able to authenticate the identity of workload 3 through the TLS protocol to ensure it is making a request of the right party. Workload 3 will authenticate workload 1 using its workload identity credentials. If the workload identity credentials are X.509 certificates then this can happen through TLS client authentication (mutual TLS). Alternatively, the workloads can use a JWT based authentication mechanism to authenticate on another. Workload three can use the authenticated identity of workload 1 to determine which APIs workload 1 is authorized 2 and to associated the authenticated identity with logs and other audit information.

* (4) API request to workload 2

Similarly to the previous flow, the gateway may determine that for another API call, the application client’s request needs to be handled by workload 2. The case behaves the same as the previous flow except that the gateway may need to authenticate workload 2 before forwarding traffic to it. Workload 3 will then authorize and audit the request based on the authenticated identity of workload 2. Workload 2 and workload 1 may be authorized to use different APIs on workload 3. If workload 1 or 2 makes an API request that it is not authorized for, then workload 3 will reject the request.

3.2.2. Context and workload Identity

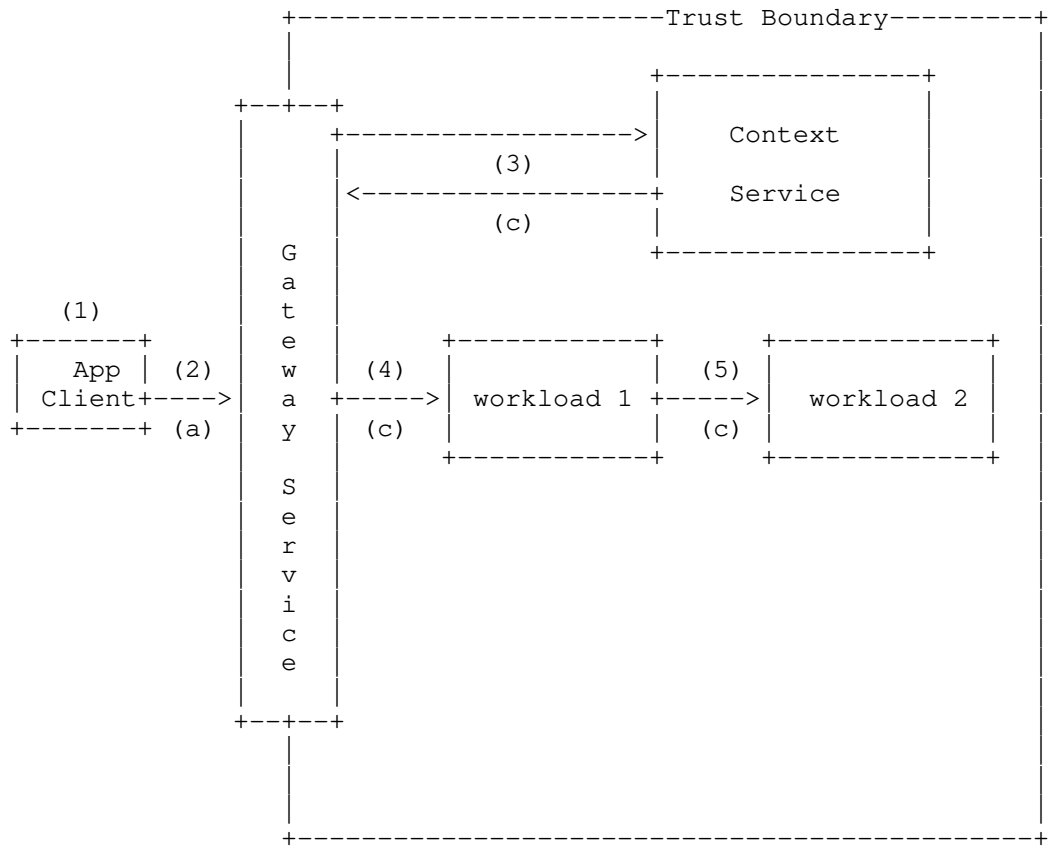

Figure 2: Context example workload application system.

In many cases the application system uses other security context information about the request during authorization and auditing. The following is a basic scenario that illustrates the propagation of security context in the workload system. Some of the components and interactions have been removed from the previous scenario for simplicity.

* Context Service This scenario adds a context service component which is responsible for creating security context based on authentication and other calculations. Context can be represented in many ways; it can be a plaintext data structure, a signed data structure such as a jwt or a pointer used to lookup the context as a data structure stored somewhere else. In one common example, creating the context may involve a token exchange converting an OAuth 2.0 access token into a different token format, such as a transaction token, that is understood by internal services.

* (1) Initial Authentication In the initial authentication the gateway service obtains credentials it can use with the gateway service. This authentication may involve several steps and may be performed by an external entity such as an identity provider. The authentication process will result in a credential that the gateway service can evaluate. For example, the credential could be an OAuth Access token. If the client already has an access token that it can use to authenticate to the gateway, such as an X.509 certificate, then it may skip this step.

* (2) Application Client Request The application client makes a request to the gateway over HTTPS. The client may be authenticated to the gateway through TLS client authentication (mutual TLS) or through a credential such as an access token obtained in step 1.

* (3) Establishing the request context The gateway service requests a security context token (c) from a token service. This process may entail sending an access token (a) along with other information to a token exchange endpoint to obtain the context token, which contains information about the entity accessing the system. This context is typically only relevant to the internal system and is not returned to the client. The gateway may use alternative mechanisms to get the internal security context information (c).

* (4) Forwarding Request to Workload The gateway forwards the request along with the context information (c) to the appropriate workload. A bearer token, such as an access token (a), is not usually forwarded as it is only meant for external access. The workload uses information in the context token in applying authorization policy to the application client's request. If the workload does not receive a context token, then it will deny requests that rely on information from the token.

* (5) Making Additional Workload Originated Requests The workload may need to make requests of other workloads. When making these requests, the workload includes the context information so Workload 2 can authorize and audit the request. Workload 2 may have a policy requiring Workload 1 to authenticate its service identity and provide valid context information (c) to access certain APIs.

3.2.3. Cross-Domain Communication

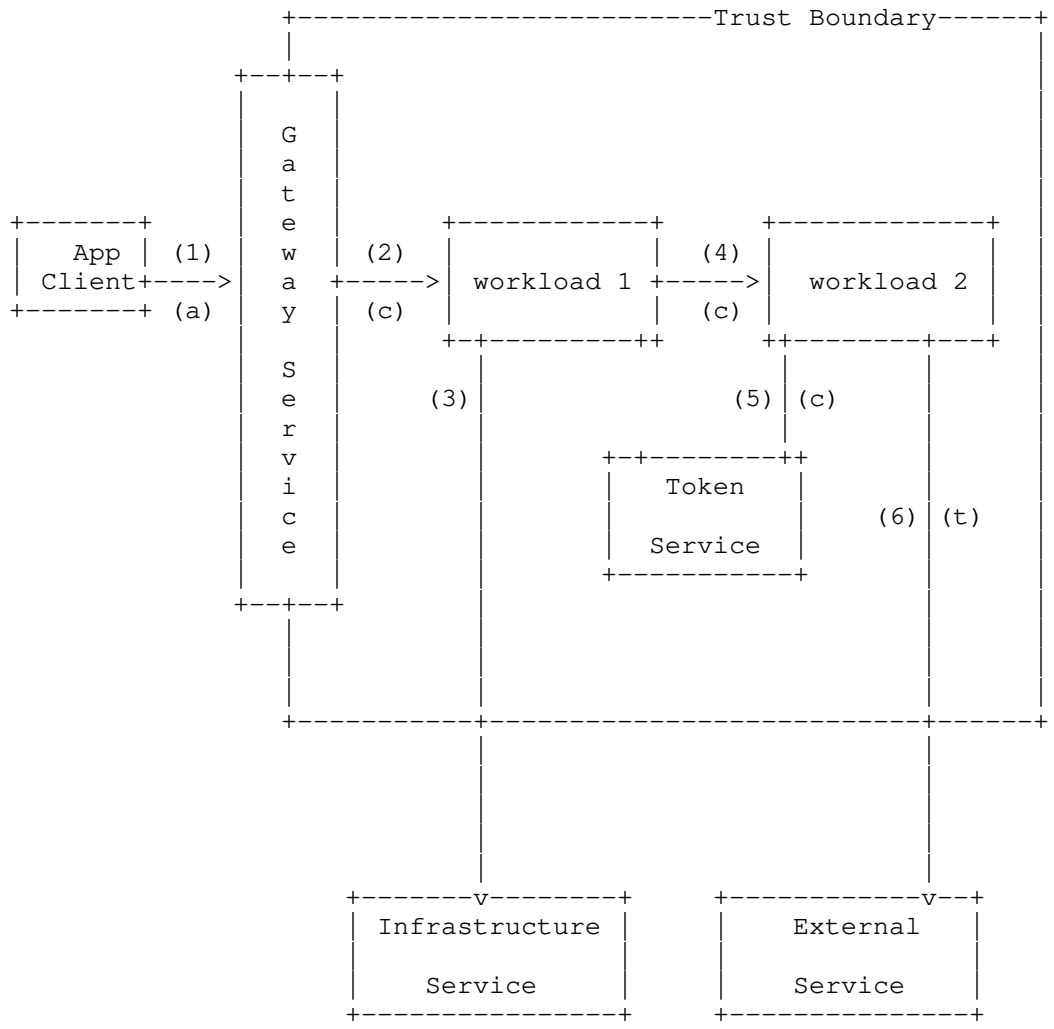

Figure 3: External request workload application system.

In many applications workloads must make requests of infrastructure that operates as a different trust domain or external services that are in a separate trust domain. Figure Figure 3 shows this scenario. The scenario shows some new components described below. Components and interactions from previous scenarios are still relevant to this example, but are omitted for simplicity.

* Token Service - the token service is responsible for exchanging information that is internal to the system such as service identity and/or security context information for a token that can

be presented to an external service to gain access to infrastructure or an external service. Note that in some cases the token service may reside in a different trust domain or there may be token services in multiple trust domains. In some cases the workload will need to contact token services in various domains in order to gain access to infrastructure or external service.

* Infrastructure Service - this service is often part of the application, but it is managed by an infrastructure provider and may require different information to access it.

* External Service - this service is distinct from the application and hosted in a separate trust domain. This trust domain often has different access requirements that workloads in the internal trust domain.

Some example interactions in this scenario:

* (1) The application client is making requests with authentication information as in the other scenarios

* (2) The gateway forwards the request to the appropriate workload with the security context information

* (3) The workload needs to access an infrastructure service and, because it is managed by the same organization, it authenticates to the service directly using its workload credentials.

* (4) Workload 1 contacts Workload 2 to perform an operation. This request is accompanied by a security context as in the other scenarios.

* (5) Workload 2 determines it needs to communicate with an external service. In order to gain access to this service it must first obtain a token/credential (t) that it can use externally. It authenticates to the token service using its workload identity and provides security context information. The token service determines what type of externally usable token to issue to the workload.

* (6) Workload 2 uses this new token/credential (t) to access the external service.

* Note that in (3) and (6) the workload may need to authenticate to an external token service to obtain an access token to access the system in the foreign trust domain.

3.3. Workload Identity Use Cases

3.3.1. Bootstrapping Workload Identity

[TODO: this section will need to be updated to discuss workload identifier as a concept as well]

A workload needs to obtain its identity early in its lifecycle. This identity is the base identity upon which further identity and security context are built.

Identity bootstrapping often utilizes identity information provisioned through mechanisms specific to hosting platforms and orchestration services. This initial bootstrapping information is used to obtain specific identity credentials for a workload. This process may use attestation to ensure the workload receives the correct identity credentials. An example of a bootstrapping process follows.

Figure 4 provides an example of software layering at a host running workloads. During startup, workloads bootstrap their identity with the help of an agent. The agent may be associated with one or more workloads to help ensure that workloads are provisioned with the correct identity. The agent provides attestation evidence and other relevant information to a server. The server validates this information and provides the agent with identity credentials for the workloads it is associated with. The server can use a variety of internal and external means to validate the request against policy. After obtaining identity credentials from the Server, the agent passes them to the workload.

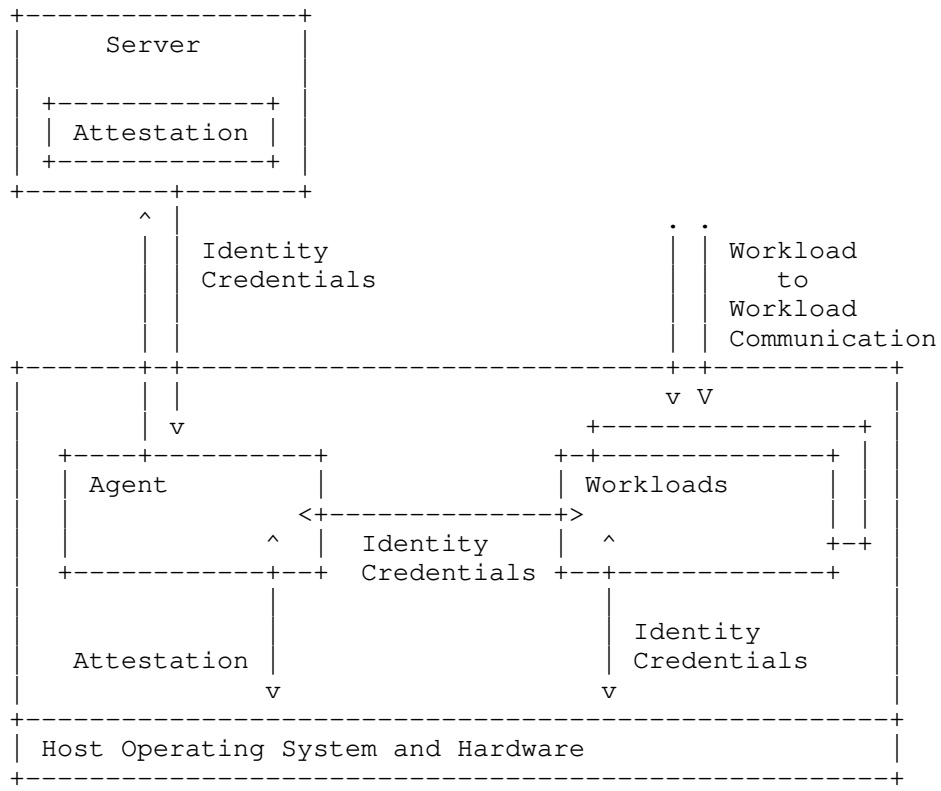

Figure 4: Host Software Layering in a Workload Identity Architecture.

How the workload obtains its identity credentials and interacts with the agent is subject to different implementations. Some common mechanisms for obtaining this initial identity include:

* File System - in this mechanism the identity credential is provisioned to the workload via the filesystem.

* Local API - the identity credential is provided through an API, such as a local domain socket (for example SPIFFE or QEMU guest agent) or network API (for example Cloud Provider Metadata Server).

* Environment Variables - identity credential may also be injected into workloads using operating system environment variables.

3.3.2. Service Authentication

One of the most basic use cases for workload identity is authentication of one workload to another, such as in the case where one service is making a request to another service as part of a larger, more complex application. Following authentication, the request to the service offered by the workload needs to be authorized. Even in this simple case the identity of a workload is often composed of many attributes such as:

* Trigger Information
* Service Name
* Instance Name
* Role
* Environment
* Trust Domain
* Software Attestation
* Hardware Attestation

These attributes are used for various purposes such as:

* ensuring the request is made to the correct service or service instance
* authorizing access to APIs and resources
* providing an audit trail of requests within a system
* providing context for other decisions made within a service

There are several methods defined to perform this authentication. Some of the most common include:

* TLS authentication of the server using X.509 certificates and client bearer token, encoded as JWTs.
* Mutual TLS authentication using X.509 certificate for both client and server.
* TLS authentication of the server and HTTP request signing using a secret key.

Figure 5 illustrates the communication between different workloads. Two aspects are important to highlight: First, there is a need to consider the interaction with workloads that are external to the trust domain (sometimes called cross-domain). Second, the interaction does not only occur between workloads that directly interact with each other but instead may also take place across intermediate workloads (in an end-to-end style).

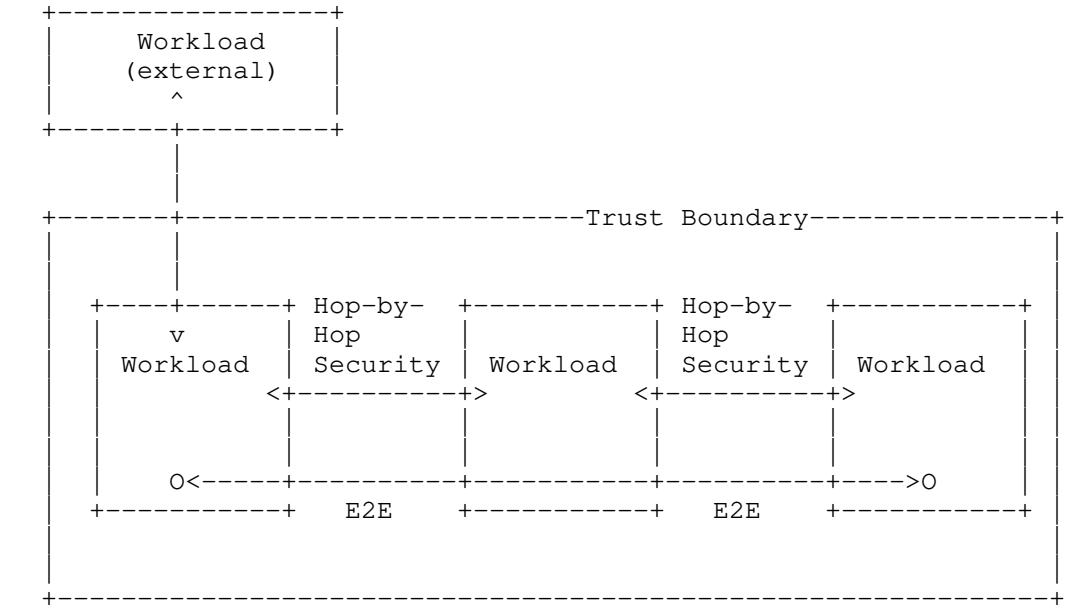

Figure 5: Workload-to-Workload Communication.

3.3.3. Security Context Establishment and Propagation

In a typical system of workloads additional information is needed in order for the workload to perform its function. For example, it is common for a workload to require information about a user or other entity that originated the request. Other types of information may include information about the hardware or software that the workload is running or information about what processing and validation has already been done to the request. This type of information is part of the security context that the workload uses during authorization, accounting and auditing. This context is propagated and possibly augmented from workload to workload using tokens. Workload identity comes into play to ensure that the information in the context can only be used by an authorized workload and that the context information originated from an authorized workload.

3.3.4. Service Authorization

After authentication of the peer, a workload can perform authorization by verifying that the authenticated identity has the appropriate permissions to access the requested resources and perform required actions. This process involves evaluating the security context described previously. The workload validates the security context, and checks the validity of permissions against its security policies to ensure that only authorized actions are allowed.

3.3.5. Delegation and Impersonation

When source workloads send authenticated requests to destination workloads, those destination workloads may rely on upstream dependencies to fulfill such requests. Such access patterns are increasingly common in a microservices architecture. While X.509 certificates can be used for point-to-point authentication, such services relying on upstream microservices for answers, may use delegation and/or impersonation semantics as described in RFC 8693 OAuth 2.0 Access Token Exchange.

WIMSE credentials constrain the subjects and actors identified in delegation and impersonation tokens to be bound by TrustDomains, and to follow their issuing authorities' trust configurations. Upstream workloads should consider the security context of delegation and/or impersonation tokens within and across Trust Domains, when arriving at authorization decisions.

3.3.6. Asynchronous and Batch Requests

Source workloads may send authenticated asynchronous and batch requests to destination workloads. A destination workload may need to fulfill such requests with requests to authorized upstream protected resources and workloads, after the source workload credentials have expired. Credentials identifying the original source workload as subject may need to be obtained from the credential issuing authority with appropriately-downscoped context needed access to upstream workloads. These credentials should identify the workload as the actor in the actor chain, but may also identify other principals that the action is taken on behalf. To mitigate risks associated with long-duration credentials, these credentials should be bound to the workload identity credential such as an X.509 certificate or Workload Identity Token (WIT) of the acting service performing asynchronous computation on the source workload's behalf.

3.3.7. Cross-boundary Workload Identity

As workloads often need to communicate across trust boundaries, extra care needs to be taken when it comes to identity communication to ensure scalability and privacy. (TODO: align with OAuth cross domain identity and authorization)

3.3.7.1. Egress Identity Generalization

A workload communicating with a service, or another workload located outside the trust boundary may need to provide modified identity information. Detailed identity of internal workload originating the communication is relevant inside the trust boundary but could be excessive for the outside world and expose potentially sensitive internal topology information.

A security gateway at the edge of a trust boundary can be used to validate identity information of the workload, perform context specific authorization of the transaction and replace workload specific identity with a generalized one for a given trust domain.

3.3.7.2. Inbound Gateway Identity Validation

Inbound security gateway is a common design pattern for service protection. This functionality is often found in CDN services, API gateways, load balancers, Web Application Firewalls (WAFs) and other security solutions. Workload identity verification of inbound requests should be performed as a part of these security services. After validation of workload identity, the gateway may either leave it unmodified or replace it with its own identity to be validated by the destination.

4. Security Considerations

4.1. Traffic Interception

Workloads communicating with applications may face different threats to traffic interception in different deployments. In many deployments security controls are deployed for internal communications at lower layers to reduce the risk of traffic observation and modification for network communications. When a security layer, such as TLS, is deployed in these environments. TLS may be terminated in various places, including the workload itself, and in various middleware devices, such as load balancers, gateways, proxies, and firewalls. Therefore, protection is provided only between each adjacent pair of TLS endpoints. There are no guarantees of confidentiality, integrity and correct identity passthrough in those middleware devices and services.

4.2. Information Disclosure

Observation and interception of network traffic is not the only means of disclosure in these systems. Other vectors of information leakage is through disclosure in log files and other observability and troubleshooting mechanisms. For example, an application may log the contents of HTTP headers containing JWT bearer tokens. The information in these logs may be made available to other systems with less stringent access controls, which may result in this token falling into an attackers hands who then uses it to compromise a system. This creates privacy risks and potential surface for reconnaissance attacks. If observed tokens can be reused, this also may allow attackers to impersonate workloads.

4.3. Workload Compromise

Even the most well-designed and implemented workloads may contain security flaws that allow an attacker to gain limited or full compromise. For example, a server side request forgery may result in the ability for an attacker to force the workload to make requests of other parts of a system even though the rest of the workload functionality may be unaffected. An attacker with this advantage may be able to utilize privileges of the compromised workload to attack other parts of the system. Therefore it is important that communicating workloads apply the principle of least privilege through security controls such as authorization.

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/rfc/rfc7517>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

6.2. Informative References

[I-D.ietf-oauth-transaction-tokens] Tulshibagwale, A., Fletcher, G., and P. Kasselman, "Transaction Tokens", Work in Progress, Internet-Draft, draft-ietf-oauth-transaction-tokens-04, 30 December 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-transaction-tokens-04>.

[I-D.ietf-wimse-s2s-protocol] Campbell, B., Feldman, D., Salowey, J., Schwenkschuster, A., and Y. Sheffer, "WIMSE Service to Service Authentication", Work in Progress, Internet-Draft, draft-ietf-wimse-s2s-protocol-01, 18 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-s2s-protocol-01>.

Acknowledgments

Todo: Add your name here.

Authors' Addresses

Joseph Salowey
Venafi
Email: joe@salowey.net

Yaroslav Rosomakho
Zscaler
Email: yaroslavros@gmail.com

Hannes Tschofenig

Email: hannes.tschofenig@gmx.net