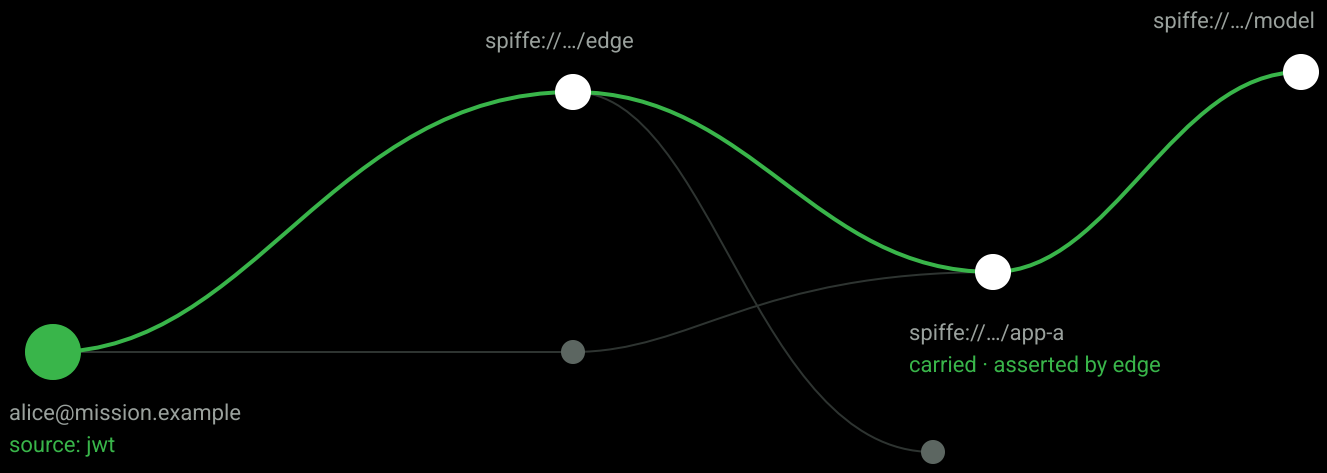
Greymatter.io

WHITE PAPER
EDITION 2 · NOVEMBER 2026



Zero trust through verifiable identity, for people and workloads

How Greymatter attests every workload, verifies every person, carries that person across every hop, and records it all without keeping one replayable credential.

Supersedes “Zero Trust Through Verifiable Workload Identity.”

© 2026 Greymatter.io, Inc.

Contents

	SECTION		PAGE
THE PROBLEM	1	Executive summary	1
	2	The two halves of the identity problem	3
	3	Three kinds of identity	4
WORKLOADS	4	Workload identity with SPIFFE and SPIRE	5
	5	Two questions on every connection	7
PEOPLE	6	Human identity at the edge	8
	7	Claims are not identity	11
THE CHAIN BETWEEN THEM	8	Identity context: the person on every hop	13
	9	Credentials as evidence, never as data	17
	10	The forensic record	19
OPERATING IT	11	Reaching enforcement by observation	21
	12	One model in every environment	23
	13	Why open standards matter	23
SUMMARY	14	Relationship to zero trust	24
	15	Benefits and key takeaways	25
	16	Evidence for security frameworks	26
	17	Ask Greymatter	27

How to read the status marks

Every capability in this paper carries one mark. Nothing marked Fielded is aspirational. Greymatter 2.5.0 adds identity context, the carried user, credential fingerprinting and the three identity classes.

FIELDED

Ships in Greymatter 2.5.0 today.

PREVIEW

Ships in 2.5.0 as a technology preview.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

1

Executive summary

Zero trust starts with one question about every request: who is this? The answer has two halves, and most programs only solve one.

IN BRIEF

Greymatter gives every workload an attested identity, verifies every person at the edge, and carries that person across every hop the mesh controls.

WHO THIS IS FOR

Program offices, ICAM leads, security architects and assessors in DoD and IC programs adopting zero trust at the application layer.

The first half is the **workload**. A service, job or agent must prove what it is before it talks to anything. The second half is the **person**. A request that starts with an analyst passes through a gateway, an application, a model and a tool. Each of those workloads acts for that analyst. If only the first hop holds the analyst's identity, every later decision and every audit record names a service, not a person.

Most programs solve the two halves separately. An identity provider signs the person in at the front door. Network rules or shared secrets protect the services behind it. Headers such as `x-user-id` fill the gap between them, and any caller can set those headers. That gap is where attribution fails and where impersonation succeeds.

Greymatter closes the gap. SPIFFE and SPIRE give every mesh workload an identity that the platform attests and rotates. Greymatter verifies the person once, at the edge, with the program's own identity provider or smart-card infrastructure. Then the data planes carry that person from hop to hop over mutual TLS. No application copies a header, and no application can name a person of its choosing.

Every hop then writes a record that names the verified person, the workload that vouched for them and the workload that connected. A fixed rule decides whether the record keeps, fingerprints or withholds each credential. An auditor can follow one session from sign-in to last use without ever seeing a value that someone could replay.

THE CENTRAL POINT

Greymatter establishes trust through cryptographically verifiable identity rather than network location, for workloads and for people, and it keeps that identity intact on every hop.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

Six properties a program office can test

Applications do not change

FIELDED

Teams do not implement mutual TLS, manage certificates, validate tokens or pass identity headers. The platform applies all of it to the application.

The person survives the hop

FIELDED

The receiving data plane learns the person from the calling data plane, not from the calling application. Greymatter writes that value in its own header, and no application can supply it.

Claims never become identity

FIELDED

Greymatter records a header that a client or an application sets as an unverified claim. Authorization never admits a request on it.

Evidence without secrets

FIELDED

Records keep who, what, when and how. Bearer tokens become keyed fingerprints. Records omit passwords and one-time codes entirely.

Identity belongs to the program

FIELDED

Workloads carry SPIFFE identities. People come from the program's own identity provider or PKI. Neither is proprietary to Greymatter.

Enforcement by observation

FIELDED

The mesh learns which verified identities call each port, then seals to exactly that set. Nobody writes the allowlist by hand.

STATED PRECISELY, SO IT IS NOT OVER-READ

Greymatter is not an identity provider and does not decide entitlements. It verifies the person a program's provider or PKI vouches for, carries that person, and enforces and records against them.

Device posture and data-layer controls belong to other parts of an enterprise architecture. Greymatter does not establish trust between separate trust domains. It does not issue workloads tokens for systems outside the mesh.

What's changed in the new release

ADDITION	WHAT IT DOES
Identity context	Decides the person, voucher and connecting service on every hop.
The carried user	Data planes carry the person to the next hop over mutual TLS.
Credential fingerprinting	No replayable credential in any record or log.
Three identity classes	People, non-human identities and SPIFFE workloads.
Work beyond the request	Background work and single-user workloads keep their person.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

2 The two halves of the identity problem

Network location fails as an identity for workloads. A header fails as an identity for people.

WHY NOW

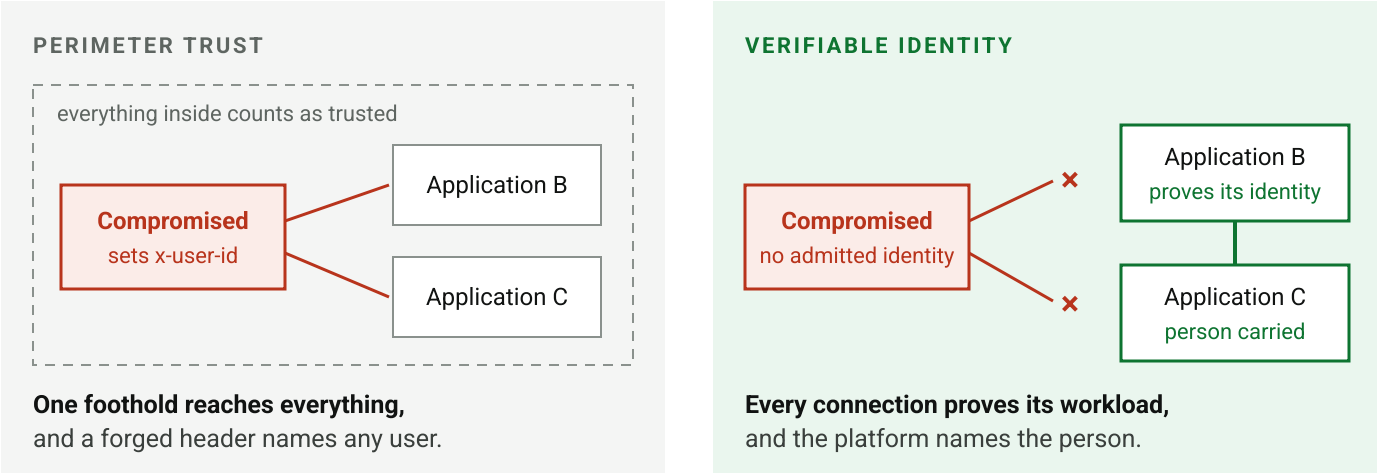
AI agents turn one person's question into a chain of model, tool and data calls. Each call is a workload acting for that person.

Mission systems span public and sovereign clouds, data centers and disconnected sites, and addresses change within minutes. Four assumptions fail.

1. **The network can identify the caller.** An IP address identifies a location, not a workload.
2. **Inside the boundary is safe.** After an adversary gets a foothold, implicit trust becomes the path for lateral movement.
3. **Each team can secure its own service.** When every team must implement mutual authentication, certificates and authorization, the results are inconsistent.
4. **A header can identify a person.** Behind the front door, services commonly learn the user from a header the previous hop wrote. Any caller that reaches the service can write the same header with another name in it.

The fourth assumption is the least examined. Without a trustworthy way to carry the person, a program cannot apply per-person limits or attribute a data-loss finding. It also cannot tell an auditor who read a record.

FIGURE 1 · PERIMETER TRUST VERSUS VERIFIABLE IDENTITY



© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

3 Three kinds of identity

Greymatter places every identity it records in exactly one class, because each class proves itself differently. FIELDED

People

An analyst signed in with OIDC, or an operator with a CAC or PIV card. Proven to the program's identity provider or PKI, then carried by the platform.

Non-human identities

An external system, a partner gateway or a batch job that connects from outside the mesh. Proven with its own certificate.

SPIFFE workloads

Mesh services, data planes, platform components and in-mesh AI agents. Attested by SPIRE and proven on every connection.

HOW IT WORKS

Greymatter decides the class from entry records, the first hop of each request, across the whole mesh.

ONE RECORD IS ENOUGH

A single entry record marks an identity as non-human for all of its records.

How Greymatter decides the class

CLASS	THE RULE
SPIFFE workload	The identity is a SPIFFE ID: it starts with `spiffe://`.
Non-human identity	An entry record names the identity in `x-forwarded-sys-id`. Or the client presented a certificate whose subject is the identity and which carries a DNS subject alternative name.
Person	Every other identity. A PIV or CAC certificate carries a URI SAN and no DNS SAN, so a smart-card holder stays a person.

An interior hop never decides the class. Every interior hop presents the calling data plane's own certificate, and counting that would mark every caller as non-human.

The rest of this paper takes each class in turn, then explains how they join on one request.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

4

Workload identity with SPIFFE and SPIRE

Attestation comes first, and issuance follows. A workload never asserts its own identity. FIELDED

WHY IT HOLDS

A compromised workload cannot request another workload's identity. It never makes the claim.

LIFETIME

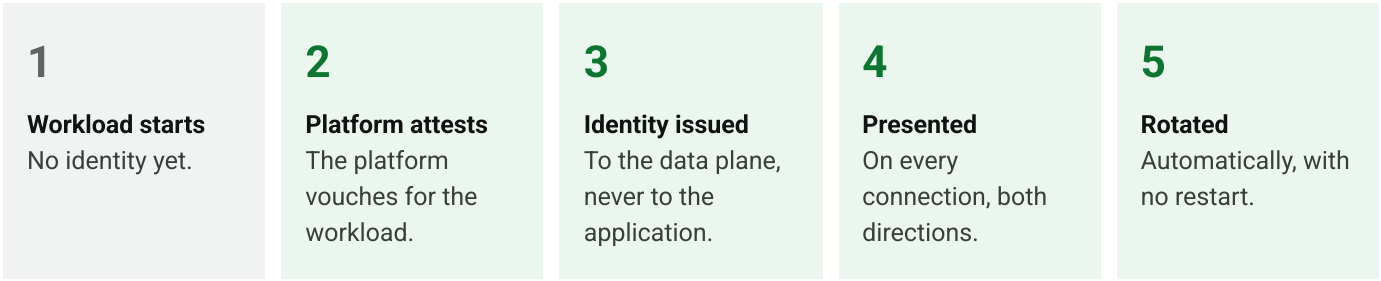
Hours, not months. Greymatter rotates every identity automatically.

Greymatter deploys and operates SPIRE as part of the platform. Every workload that Greymatter manages receives a SPIFFE identity, scoped so that SPIRE never issues one mesh an identity that belongs to another.

A Greymatter data plane runs beside each application. The data plane, not the application, obtains the identity, presents it, verifies the peer and enforces policy. The application sends and receives ordinary traffic. It never handles mutual TLS itself.

Before SPIRE issues anything, it queries the platform to identify the workload. The platform returns a projected service account token. Only then does SPIRE issue an identity, and it issues that identity to the data plane, never to the application.

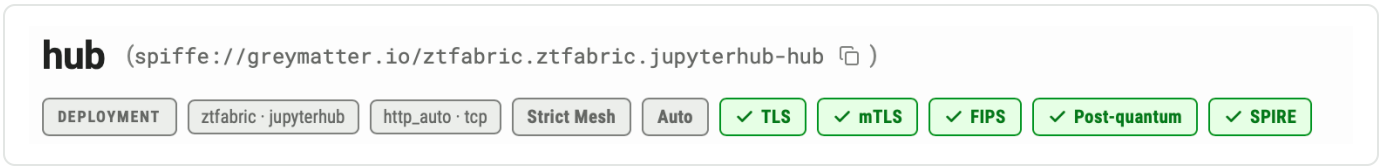
FIGURE 2 · THE WORKLOAD IDENTITY LIFECYCLE



Short lifetimes are the control. A stolen copy has a bounded window of use. Replacement is automatic and needs no restart, so there is no renewal calendar, no expiry outage and no distribution problem.

SPIRE delivers identity locally. An agent runs on every node, so a workload never reaches across the network for its credential. That property matters in constrained or disconnected environments, as section 12 explains.

FIGURE 3 · A WORKLOAD'S IDENTITY IN THE CONSOLE



© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

What Greymatter adds beyond the identity standard

SPIFFE and SPIRE deliver a verifiable identity to a workload. They do not encrypt a connection, decide who may call whom, produce evidence, or govern how an application joins the environment. Greymatter supplies that layer once, uniformly, with no application changes.

IDENTITY FOUNDATION

SPIFFE + SPIRE

Defines the identity
Attests the workload
Issues the identity
Rotates it
Open standard
CNCF graduated

Secure communications
Mutual TLS on every connection. The receiver requires a client certificate.

Policy by identity
Callers matched exactly against an admitted set per service and port.

The person, carried
Human identity carried across every hop. New in 2.5.0.

Evidence and audit
Every hop attributed to the authenticated peer, with credentials scrubbed.

Onboarding by policy
Enroll a namespace, and its workloads join with identity applied.

Same everywhere
One identity and policy model in public and sovereign clouds, data centers and disconnected sites.

New in 2.5.0 for workload identity

HOW IT WORKS

Greymatter delivers each secret only to the data plane or edge whose configuration uses it.

- **SPIRE your way.** Greymatter deploys a SPIRE that several meshes can share. A program can also bring its own SPIRE, its own registry and a new trust domain. FIELDED
- **Identity health on every pod.** The Console shows that each pod received its identity. It flags a missing identity, an expired certificate or a failing rotation. FIELDED
- **Secrets only to their owner.** Greymatter delivers a secret only to the data plane whose configuration uses it. Secrets never appear in a configuration dump. FIELDED
- **Identity for the platform itself.** Greymatter delivers configuration to each data plane over mutual TLS. It reaches each data plane's interfaces through the mesh by SPIFFE identity, with no port forwarding. FIELDED

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

5 Two questions on every connection

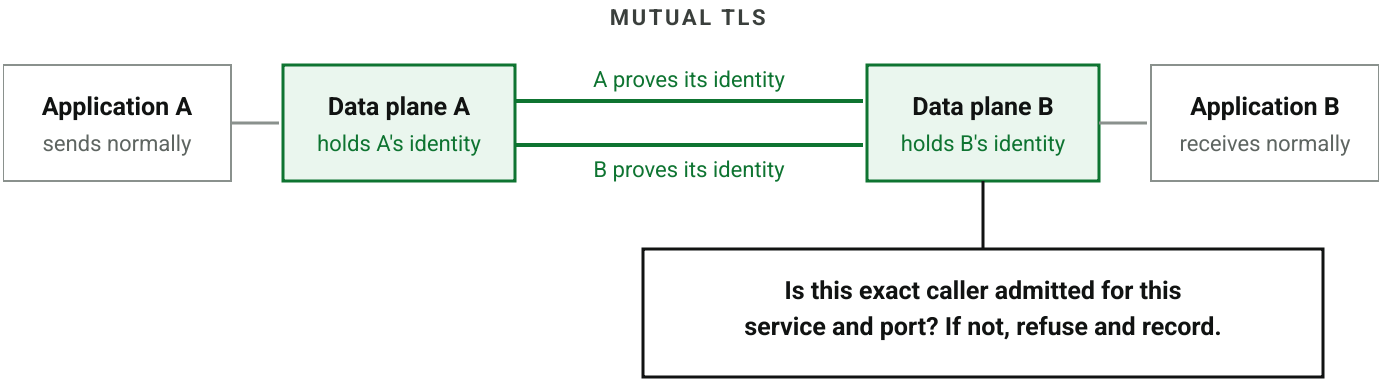
Who are you, and may you be here? The data planes decide both before any traffic reaches the application. FIELDED

WHY EXACT MATCHING

A valid certificate proves only that the caller is some workload in this environment. On a shared platform, that is not a control.

When one mission application calls another, neither application takes part in establishing trust. Their data planes do. Each presents its own identity and verifies the other's. Then the receiving data plane checks whether **that exact caller** may reach **that exact service and port**. If the caller is not in the admitted set, the data plane refuses the request and records the refusal.

FIGURE 4 · SERVICE-TO-SERVICE MUTUAL TLS WITH AN IDENTITY CHECK



Mutual authentication settles who. The admitted set settles whether. The decision uses an identity, not an address, so it stays correct when a workload moves, scales or restarts.

The enforcement point sits at the workload, not at a network boundary. Every request crosses one, including requests between two workloads on the same host. No interior position confers trust.

A DESIGN PRINCIPLE FOR SECURITY REVIEWERS

Greymatter refuses to arm identity-based authorization while it still admits unauthenticated traffic. We coupled the two settings on purpose. A gate on identities that still accepts anonymous traffic would enforce nothing. Assessors often find controls that an operator can set to look secure. An operator cannot set this one that way.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

6

Human identity at the edge

Greymatter is not an identity provider. It verifies the person that the program's provider or PKI vouches for, and it refuses the rest. FIELDED

WHERE IT HAPPENS

A Greymatter edge, where requests from people enter the mesh. It verifies once, at the first hop.

NOTHING TO REPLACE

Greymatter works with the identity provider and PKI the program already runs.

People reach the mesh through an edge, which verifies them with the program's existing identity authority. Greymatter supports three ways to sign in and records which one the person used.

SIGN-IN	WHAT THE EDGE VERIFIES	RECORDED AS
OIDC and OAuth2	The edge runs the OAuth2 login with the program's provider. Greymatter checks the token's signature against the provider's published keys. The person comes from the verified token, never from an unchecked cookie.	`jwt` arrival: OIDC
PKI, CAC and PIV	The edge requires a client certificate and validates it against the program's trust bundle. With OCSP turned on, the edge queries the responder during the handshake. A revoked response always fails the handshake, and the program chooses whether a missing response fails it too.	`cert` arrival: PKI
OpenShift sign-in	After the OAuth login, Greymatter queries its API for the owner of the bearer token. It records the person, their roles and the namespaces they may reach. It never blocks or changes the request on its own.	`openshift`

REVOCATION THAT A LATER RESPONSE CANNOT UNDO

Each edge keeps a cache of OCSP responses. A good response expires at the responder's next update or the program's cache limit, whichever is first. The edge keeps a revoked response and never replaces it with a later good or unknown response. A forged or later good response cannot reinstate a card.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

One header the application can trust

HOW IT WORKS

Greymatter removes any incoming `x-greymatter-verified-user` before it processes the request.

ONE LIMIT

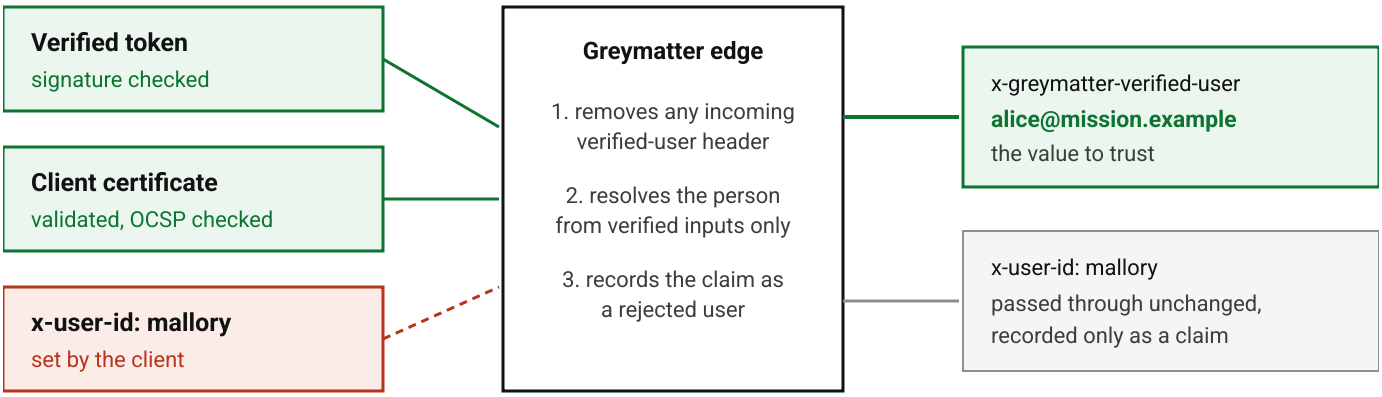
A TCP pass-through port parses no HTTP and removes nothing. An application behind such a port must not trust the header.

Applications behind the edge often need the person's identity. Greymatter gives them one value they can rely on: `x-greymatter-verified-user`.

Because this header is Greymatter's own, Greymatter discards any copy a request arrives with, before it processes the request. It then sets it only from an identity that Greymatter verified: a peer certificate, a verified token claim or a verified OpenShift sign-in. A client cannot supply it, and a stale value cannot survive.

Greymatter leaves the customer's own identity headers alone. A gateway or load balancer in front of the edge may pass `x-user-id`, `from` or `ssl-client-s-dn`. Those reach the application exactly as they arrived. Greymatter fills them in only when they are absent, and it treats every one of them as a claim.

FIGURE 5 · WHAT REACHES THE APPLICATION, AND FROM WHOM



The application reads one header for identity. Greymatter never overwrites the customer's own headers.

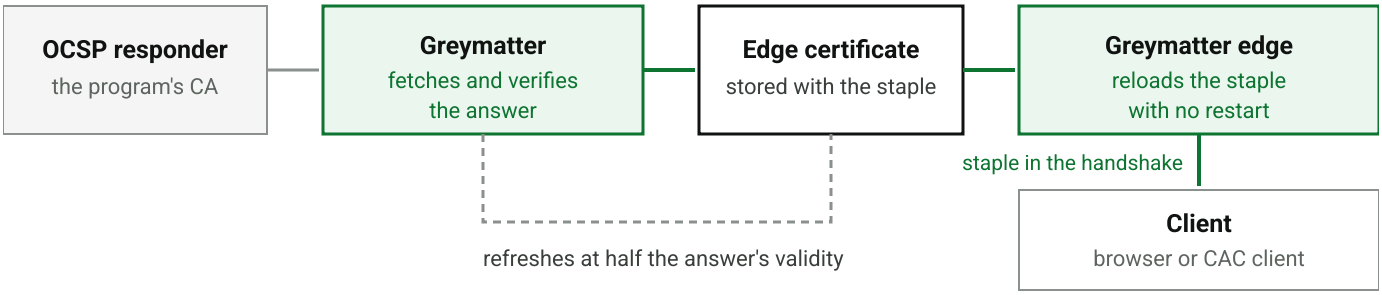
© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

OCSP stapling: the edge proves its certificate is good

The client certificate check protects the edge from a revoked card. Stapling protects the person from a revoked edge. In each TLS handshake, the edge hands the client a fresh, signed OCSP response about its own certificate.

FIELDED

FIGURE 6 · HOW A STAPLE REACHES THE CLIENT



WHY IT MATTERS

A client learns that the edge certificate is still good without contacting the responder. That works at sites that cannot reach the responder. Clients never contact the responder, so it cannot track who connects.

REFRESH

Greymatter replaces each response well before it expires.

What Greymatter checks before it staples

- **The right certificate.** The response names this exact certificate, so a rotation never reuses the old certificate's response.
- **The right signer.** The issuer, or a delegated OCSP responder that the issuer signed. Greymatter refuses SHA-1 signatures.
- **A current response.** Its next update must lie ahead. Greymatter never staples a response with no next update.
- **A safe fetch.** Greymatter refuses to dial loopback, link-local, cluster and cloud metadata addresses, and it checks the address of the connection it opens. It allows private ranges, where enterprise and air-gapped responders run.

STAPLE POLICY	WHAT THE EDGE DOES
Lenient	Serves a valid staple. When the staple is missing or expired, it completes the handshake without one.
Strict	Serves a valid staple and refuses the handshake when the staple has expired. A certificate with no staple still serves.
Must staple	Refuses every handshake without a valid staple. Greymatter allows this policy only when a staple already exists.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

7

Claims are not identity

A value that a caller can choose never admits a request and never counts as a verified person. FIELDED

THE FAILURE IT PREVENTS

A service trusts `x-user-id` because the edge usually sets it. An attacker who reaches the service sets it too.

VERIFIED SOURCES

Only three sources count as verified on a hop: `jwt`, `openshift` and `cert`. A gate that acts on who the user is accepts nothing else.

The most common identity failure behind the front door is a believed header. Greymatter closes this gap with five rules, and it enforces each one on every hop.

1. **A verified source beats a claim.** When Greymatter has verified one person and a header names another, Greymatter takes the verified person. It records the header's claim as rejected, with who sent it. The record keeps the claim when it is an email address or a SPIFFE ID, because those identify someone without carrying a credential. For anything else, such as a certificate subject or a pasted token, it shows `[withheld]`, which still shows the attempt.
2. **Greymatter does not guess.** Two vouched identities that name different people are a conflict. The record shows the conflict, and Greymatter takes neither person.
3. **Authorization reads only verified users.** Greymatter's impersonation check never reads the user from a request header, and admits a listed person only when this hop verified them. A client-set `x-user-id` on the allow list no longer admits anyone. A system that acts for another user must itself be on the allow list.
4. **A carried person does not open a gate.** Another Greymatter data plane vouched for a carried person, but that person may have started as a trusted application's forward. A gate that admits by user therefore does not accept a carried person on its own.
5. **Inspection reads the resolved person.** Guardrails, DLP, App Guard and the WAF take the user from the identity the mesh resolved. They ignore an unverified header. A header that an application sets cannot satisfy a rule about a user.

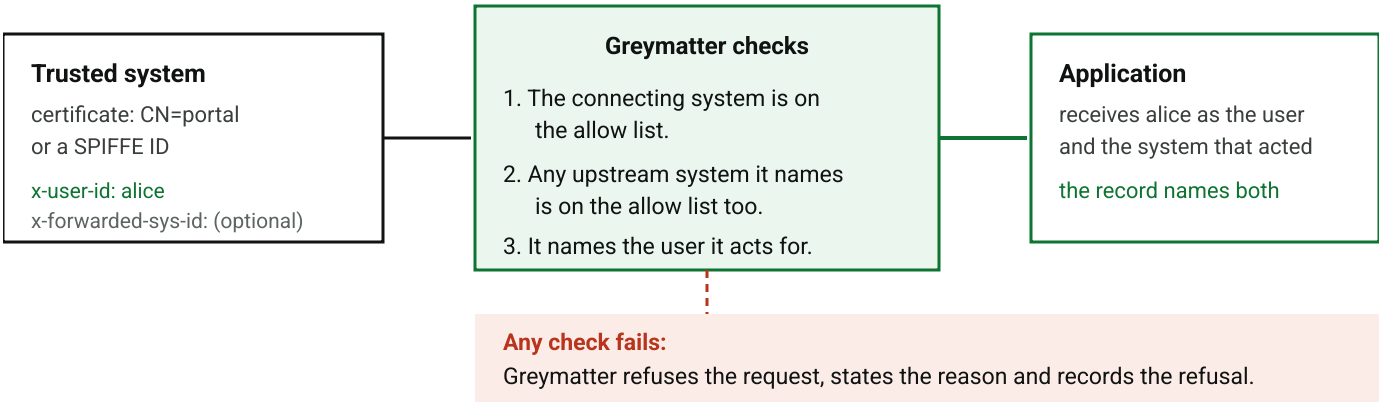
   Greymatter also validates every user value before it records it, and it discards malformed values.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

Acting for a person: impersonation

Mission systems often reach a service on behalf of a user: a portal, a gateway, a cross-domain solution. Greymatter lets a trusted system act for a person, and it refuses every other system that tries. FIELDED

FIGURE 7 · A TRUSTED SYSTEM ACTS FOR A USER



Two ways a request gets in

- **A trusted system acts for a user.** The certificate of the connecting system proves which system it is. An upstream system that it names can only cause a refusal and never admits a request on its own.
- **A listed person arrives directly.** Greymatter verified the person on this hop, and the person is on the allow list. A user header that names anyone else fails this path.

Impersonation in the forensic record

The Console lists every system that acted for users: the service it reached, the users it acted for, and its failures and policy denials. It counts a request only when the system's certificate proved the system. It marks a tenant's own edge as routine, so the exceptions stand out.

ALLOW LIST ENTRIES

SPIFFE IDs or certificate subjects. Greymatter compares subjects after it normalizes their format.

A PROGRAM'S CHOICE

A program can require an allow-listed system on every request, even for a person Greymatter verified.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

8

Identity context: the person on every hop

The data plane that receives a call learns the person from the calling data plane, not from the calling application. FIELDED

HOW IT WORKS

Built into every Greymatter data plane and edge. There is nothing to install.

TO TURN IT ON

One platform setting. A tenant has nothing to configure, and identity context never blocks a request.

Verifying a person at the edge solves only the first hop. In most meshes, the person survives the next call only if the application copies a header and the next service believes it.

Identity context removes that dependency on the application. It runs in three roles on each data plane:

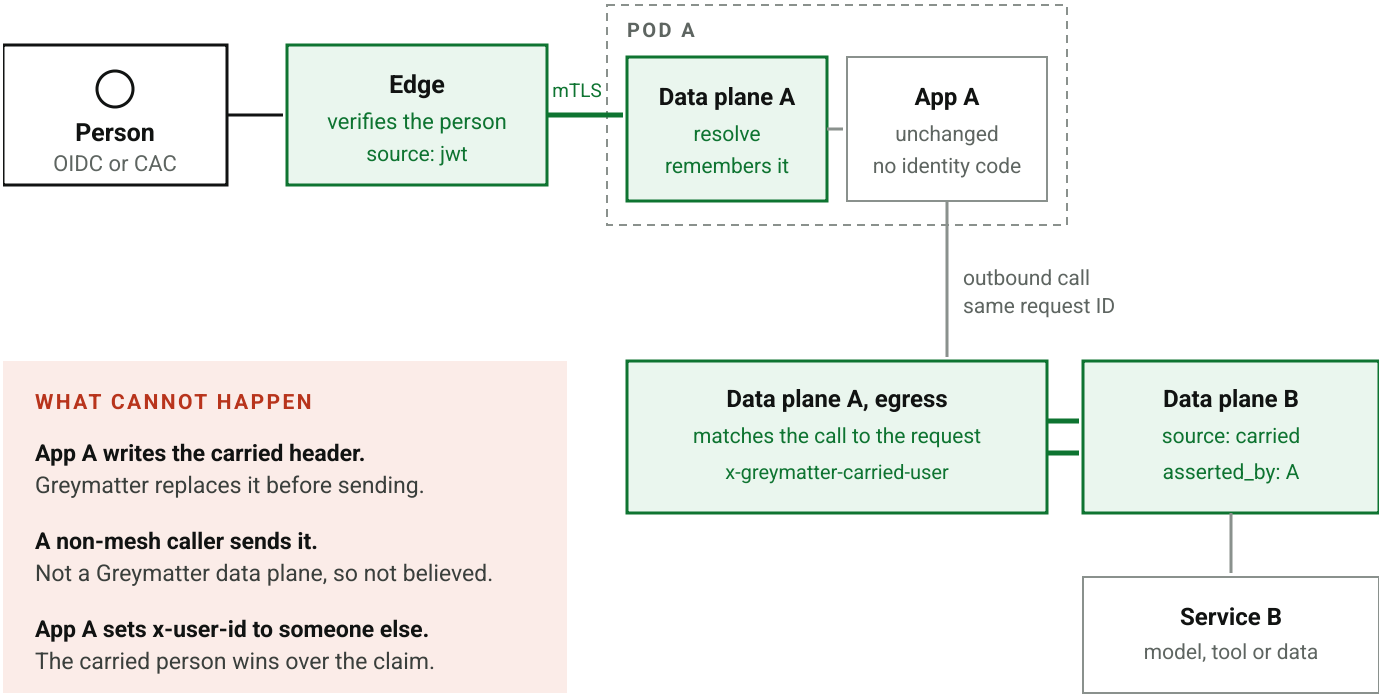
`observe`	Runs first, before any other processing. It records what the peer sent, unchanged.
`resolve`	Decides the person, the source, the voucher and the connecting service for each inbound request. It remembers the request while it is in flight and briefly after.
`egress`	Sits on the application's outbound path. It matches each call to the inbound request it serves, by request ID first and trace ID second.

The calling data plane's route then writes the person into `x-greymatter-carried-user`, and only on routes to mesh services over mutual TLS. Greymatter replaces any copy of its own header that a request arrives with. It never removes the customer's identity headers, and records those as unverified claims. A call with no verified person carries the caller's SPIFFE ID, which the receiver records as the service.

The receiving data plane believes the header only from **another Greymatter data plane or edge**: a peer whose certificate the handshake validated and that Greymatter lists as its own. It records the person with the source `carried` and names the peer that vouched for them.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

FIGURE 8 · CARRYING THE PERSON ACROSS HOPS



The application does nothing.
It does not copy, sign or forward identity. The person reaches Service B because two data planes agreed on it over mutual TLS, and the record at B names the data plane that vouched for the person.

The sources of a person, in the order resolve applies them

SOURCE	WHERE THE PERSON CAME FROM	OPENS A GATE
`jwt`	A token Greymatter verified on this hop	Yes
`openshift`	A person Greymatter verified through OpenShift sign-in	Yes
`cert`	A validated client certificate, such as a PKI client	Yes
`proxy`	A value Greymatter set on this same hop	No
`carried`	The carried user from another Greymatter data plane, which the record names	No
`forwarded`	A trusted mesh application's forward header, such as from oauth2-proxy	No
`unverified`	A header as it arrived. Recorded, never trusted	No

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

What carrying the person makes possible

These capabilities depend on the platform knowing the person several hops away from the edge. Each one works without a token reaching the downstream service and without a token exchange.

Per-person AI token budgets

FIELDED

Each model call counts against the person who asked, with an optional daily budget by group. The data planes carry the person's groups beside the person, so the model egress needs no token from the person.

Ask Greymatter acts as the person

FIELDED

The Console assistant reads the Greymatter API as the person who asked. An answer shows only what that person's roles let them see in the Console.

MCP tools see the verified user

PREVIEW

The MCP Router forwards the carried person to each mesh MCP server. A role lists and calls only its granted tools, prompts and resources, with per-person call limits.

Findings name a person

FIELDED

A Guardrails or DLP finding on a model or tool call names the person the call served, not only the service that made it.

PROVEN LINKS

Greymatter links a hop to its caller only when the caller's verified identity proves the link.

FOR THE INVESTIGATOR

One request opens as one chain, from the edge to the last service it reached, with the person on every hop.

Greymatter builds the forensic chain itself

Most tracing depends on every application passing the right header along. One application that drops it breaks the trail. Greymatter does not depend on that. Each data plane records what it verified and observed. That includes the person and how Greymatter learned it, who vouched, and which service connected. It also includes every outbound call the application made for that request. When an investigator opens a request, Greymatter joins those records into one chain.

Applications need no Greymatter code and pass on no Greymatter header. A request ID or trace context, which most applications already pass on, is enough to keep a hop in the chain. When two requests share an ID, Greymatter records the call for both and the chain flags the ambiguity instead of guessing.

A client cannot attach its calls to someone else's request by guessing an ID. Greymatter never uses an ID that a client chose to link a call.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

Work that outlives the request

Some common patterns can break the link between a call and the request it serves. Greymatter handles two of the most important for AI workloads: agents that keep working after they respond, and workloads that run for one person. FIELDED

WORK AFTER THE RESPONSE

The person stays with the trace

Many applications respond first and keep working: an agent with a background task, or a worker that drains a queue.

Greymatter holds each completed request's person under its trace ID for a short, fixed window. Any later call that carries the same trace still names the person. The window bounds how long any application can act for someone after their request ends.

Applies to every workload whose outbound calls pass through Greymatter and pass on trace context.

WORKLOADS THAT RUN FOR ONE PERSON

The workload names its owner

Some workloads run for one person and call out on their own, with no request or trace ID. A personal agent and a per-user workspace are examples.

The tenant policy names the pod annotation that holds the owner, for any workload it matches. Greymatter then names that owner on calls that match no request, with the source `workload_owner`. A gate that acts only on a verified user does not accept an owner.

Applies to every workload a tenant policy matches. A pod without the annotation names no one.

WHY IT MATTERS FOR AGENTS

Agents do most of their work after the person's request returns. Without a bounded hold, that work is anonymous.

Precision about what "carried" means

A carried person is a statement by another Greymatter data plane, made over mutual TLS. It says which person the calling data plane matched the call to. It is strong evidence for attribution, limits and inspection.

It is not a fresh verification. The first hop may have learned the person from a trusted application's forward header rather than from a token. For that reason, a carried person does not open an authorization gate on its own. The record also always names both the source and the voucher.

Where a program needs the downstream service itself to decide on a verified person, the service directly behind the verifying hop receives that person in `x-greymatter-verified-user`.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

9 Credentials as evidence, never as data

A forensic record must say who, what, when and how. It must never hold something an attacker could replay. FIELDED

THE RULE, BUILT INTO EVERY DATA PLANE AND EDGE

Keep anything that identifies who, what, when or how. Fingerprint only the part that would let someone replay a credential. Never store a password or a secret, raw or fingerprinted.

NO SWITCH

The transform is always on. It has no setting, because no setting may make it write a raw credential.

WHERE IT APPLIES

Forensic audit records, WAF events, Guardrails and DLP events, and Greymatter's own logs, error text and crash dumps.

Most audit pipelines copy request headers as they arrive. A bearer token in an audit store is a live credential in a second system, often with weaker controls than the first. Greymatter treats every credential it finds as one of three classes.

CLASS	STORED AS	EXAMPLES
Keep	The value, unchanged	User names, key IDs, OAuth client IDs, JWT header fields and selected claims, cookie names, cloud access key IDs
Fingerprint	`[fp:k1:9f2c41ab…]`	Bearer tokens, session cookies, API keys with no public ID, request signatures, CSRF tokens, OAuth codes
Withhold	`[withheld]`	Passwords, PINs, one-time and MFA codes, client secrets, Digest and NTLM responses, private keys

Withheld values get no fingerprint, for three reasons. Anyone with the key could enumerate six-digit codes or a password dictionary in milliseconds. Equal fingerprints would reveal password reuse. And password-derived responses fall to offline cracking, so storing them amounts to storing the password.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

FIGURE 9 · ONE REQUEST, BEFORE AND AFTER THE TRANSFORM

AS IT ARRIVED

```
authorization: Bearer eyJhbGciOiJSUzI1…
cookie: session=8f3b2c9e1d7a40f6b…
x-api-key: sk_live_51Hx9…Q2v
authorization: Basic YWxpY2U6aHVudGVyMg==
x-otp: 482913
```

AS THE RECORD SHOWS IT

```
Bearer [fp:k1:9f2c41ab0e7d5c11]
session=[fp:k1:2d81e07a9c4b3f55]
[fp:k1:c7a90b15d2e34f60 len=32
prefix=sk_live_]
user alice · password [withheld]
[withheld:short len=6]
```

Illustrative values. The JWT's issuer, subject, audience, scopes, times and authentication methods go into a separate structured section. A label next to them states whether Greymatter verified the token.

HOW IT WORKS

A keyed, one-way hash. Nobody can reverse a fingerprint, and without the key nobody can compute one.

THE KEY

Greymatter delivers the key to each data plane as a secret. It never comes from an SVID or certificate.

Properties a reviewer can check

- **Same credential, same fingerprint.** A token gives one fingerprint whether it arrived in a header, a URL or a JSON body. Records link without anything replayable.
- **Session lineage.** The `Set-Cookie` that issues a session and every later cookie that carries it share one fingerprint. An auditor can follow a session from sign-in to last use.
- **Replays show up.** A per-request signature seen twice has the same fingerprint twice, which is itself evidence of replay.
- **Short values get no fingerprint.** The record withholds short values and keeps only their length, because a key holder could enumerate them.
- **It fails closed.** Without a usable key, Greymatter writes `[withheld:no-key]`. It never falls back to the raw value.
- **Nobody can forge a marker.** The transform escapes text that already looks like a marker, so a caller cannot plant a victim's fingerprint and tie a request to that session.
- **A label marks token claims.** A token's claims say what the token says, not who the caller is. An unsigned token keeps no claims, and conflict checks use only verified identities.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

10

The forensic record

Every hop writes a record where it happens. Together they show who did what, through which workloads, and how each identity arrived. FIELDED

ON EACH RECORD

An identity object: person, source, voucher, connecting service, request ID, and the calls the application made for the request.

RETENTION VIEW

The Console's Forensic Audit page reads 24 hours, 7 days or 30 days. The same data is available from the Greymatter API.

Greymatter writes one record per hop, after it scrubs credentials. Records keep how the connection carried the hop: mutual TLS inside the mesh or TLS to the outside, and the TLS version negotiated. Greymatter joins the records into chains and rolls them up by hour.

How identity arrived

For the first hop of every request, the record says how its identity entered the mesh. Each first hop lands in exactly one bucket, and the buckets add up to the arrivals.

OIDC A verified token	**PKI** A client certificate, such as a CAC	**OpenShift** A verified OpenShift sign-in	**External system** Named in `x-forwarded-sys-id`
Service to service A SPIFFE workload	**Forwarded** From a trusted application	**Propagated** Any other source	**None** No identity at all

Everything one person did

Because the data planes carry the person, you can read the record by person. The per-person view shows the person's activity by hour and every hop of each request in order. For each hop, it shows how the connection carried it and the identity at that hop, with how it arrived. It also shows the person's totals of requests, blocks and denials. You can also read Guardrails findings person by person.

The change ledger records every GitOps push and every edit to a Greymatter resource, with its author. The ledger signs its checkpoints, so a reader can verify the chain.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

Forensic review: from a question to the evidence

An investigator starts with a question, not a log query. Greymatter gives direct answers to the common questions, from records that hold no replayable credential. FIELDED

What happened to this request?

The chain view lists every hop of the request in the order it traveled, with every branch. It flags any hop it cannot show, such as one past retention.

Who is new?

The Console lists the identities that first appeared in the last day, as people, non-human identities and SPIFFE workloads.

Who acted for whom?

The impersonation view lists each system that acted for users, the services it reached and the users it served.

Who came in unverified?

The sessions view groups requests that reached an edge with no verified person, by a shared cookie or token fingerprint or connection. It labels each session as an inference.

Did this credential appear here?

An investigator submits a suspect token to the Greymatter API. Greymatter fingerprints it under every key the mesh has held and reports where and when it appeared. No record ever holds the value.

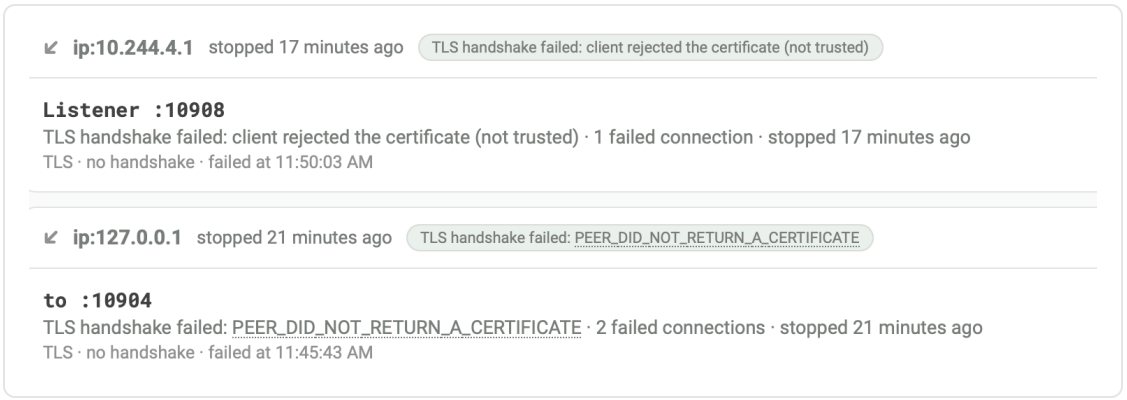
Is someone probing?

The Greymatter API lists endpoints where policy denied an identity and later served it, with the first denial and the last success.

REVIEW STAYS WITHIN THE READER'S OWN ACCESS

Every view reads only the namespaces the reader's grant reaches. A hop outside that reach appears in a chain as present but hidden, so the reader knows it exists without seeing its contents.

FIGURE 10 · FAILED CALLERS, NAMED WITH THE IDENTITY CAUSE



© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

11

Reaching enforcement by observation

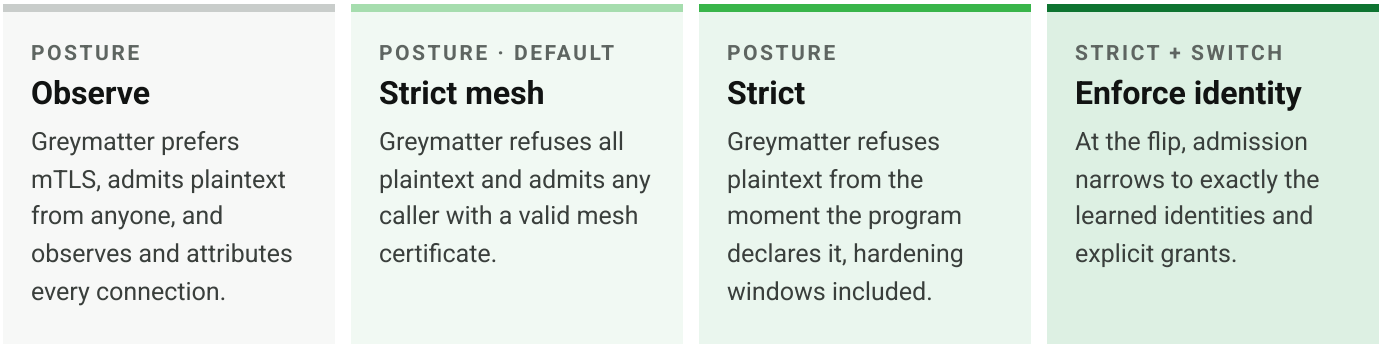
Greymatter learns who calls each port, then climbs a posture ladder on a clock the program sets. Nobody writes the allowlist by hand. FIELDED

HOW IT WORKS

The Console shows the learned identities on each port, with the note that they become the allowlist once identity enforcement is on.

The hardest part of zero trust is the transition. Nobody has a current inventory of who calls which service, on which port. Greymatter records every verified identity that calls each port and shows that learned list before it enforces anything.

FIGURE 11 · THE POSTURE LADDER



Enforce identity is a switch, not a rung. Greymatter accepts it only at Strict. On SPIRE, it narrows the handshake to the admitted set. With the program's own PKI, the impersonation check carries identity enforcement instead.

The hardening clock

DEFAULTS

Observation: 30 minutes.
Verified mTLS: 3 minutes.
Zero passthrough: 4 minutes.
A program can set each one.

Each namespace climbs through three timed windows, back to back. During observation, traffic passes through while Greymatter learns the call graph. During verified mTLS, Greymatter builds the mesh paths and traffic moves to them. When the zero-passthrough window closes, plaintext ends and the declared posture takes full effect.

The windows are timers on purpose. A step that waited for evidence could stall forever on a quiet service. A peer that never appeared during the windows meets a refusal at the flip. The Console lists that peer, so the program can admit it with a grant.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

Policy sets the posture: global and tenant

Two policy objects decide where each namespace sits on the ladder. The mesh owner writes one global policy. Each tenant can refine it, within the limits the owner sets. FIELDED

GLOBAL ONBOARDING POLICY

One per mesh, written by the owner

Ordered rules, first match wins. Each rule claims namespaces and decides who writes their policy. A dry run shows what a rule would claim. A minimum posture sets a floor for tenants.

TENANT ONBOARDING POLICY

One per namespace, written by the team

The team declares its posture, hardening windows, identity enforcement and grants. Greymatter reports the effective posture and any reason it differs.

RULE SOURCE	WHO WRITES THE NAMESPACE'S POLICY
Admin	The global rule is the whole policy. Greymatter rejects a tenant policy at admission, and claimed workloads attach at once.
Shared	The global rule is the base, and the tenant refines it within the owner's limits. If the tenant deletes its refinement, the namespace returns to the base.
Tenant	The tenant's policy is the whole policy, within the owner's limits. Nothing attaches until the team writes one, which makes it the safe way to roll out a first rule.

HOW IT WORKS

Greymatter refuses a floor stricter than the rule's own posture, and it never lets a floor drop a sealed policy below Strict.

Changing admission

- **A grant admits one caller.** It names a client identity and a port, and removing it revokes the admission. Traffic after the flip never widens admission.
- **A re-learn starts over.** It returns the namespace to Observe and runs the windows again. It is the only way to drop an admitted peer short of offboarding.
- **A failed read never revokes access.** If Greymatter cannot read what it observed, it keeps what it last read.
- **No identity plane, no false claim.** With neither SPIRE nor mTLS certificates, the strict postures refuse nothing, so Greymatter reports the namespace as Observe and says why.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

12

One model in every environment

The same identity model, policy model and record in public clouds, sovereign clouds, data centers and disconnected sites. FIELDED

STATED PRECISELY

Greymatter fields one trust domain that reaches across environments. It does not establish trust between separate trust domains.

- **SPIRE issues identity locally.** The SPIRE agent runs on the workload's own node. Getting a credential never depends on reaching a central service at that moment.
- **The platform installs disconnected.** A program can point every image at a local mirror and pin it by digest.
- **Each edge verifies people where they enter.** Each edge caches OCSP responses, and stapling lets clients check the edge without reaching the responder.

13

Why open standards matter

Programs write policy, audit and accreditation against identity. That foundation must not depend on one vendor.

OUR POSITION

Identity belongs to the program, not to any platform that consumes it.

The workload identity is a SPIFFE identity, defined by a public specification and issued by open-source software that is a graduated CNCF project. Other standards-based meshes, proxies and platforms understand it. The human identity comes from the program's own OIDC provider or PKI. Greymatter consumes it through the same open standards: OIDC, OAuth2, X.509 and OCSP.

The consequences are concrete. Infrastructure stays a choice, not a lock-in. Two programs that both adopt SPIFFE share a vocabulary for identity even where they share nothing else. The people in the record are the program's own people, named the way the program's ICAM names them. So the evidence keeps its meaning after any change elsewhere in the architecture.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

14

Relationship to zero trust

Mapped at the architectural level, not as a compliance checklist.

REFERENCES

DoD Zero Trust Reference Architecture. NIST SP 800-207. Both describe the same shift in different vocabularies.

Both frameworks require the same things. Stop inferring trust from position, and decide each access on its own merits. Enforce as close to the resource as possible. This architecture applies that at the application and workload layer, and now carries the user through it.

ZERO TRUST PRINCIPLE	HOW THIS ARCHITECTURE MEETS IT	PILLAR
No implicit trust from location	Position on the network confers nothing. Every connection proves its workload.	Network, Applications and workloads
Strong identity for non-person entities	SPIRE attests workloads. No workload asserts its own identity. Greymatter classes and records external systems as non-human identities.	User (NPE), Applications and workloads
The user on every access decision	The edge verifies the person, and the data planes carry that person to each hop. Claims never count as verified.	User
Enforcement at the resource	The enforcement point sits in front of each workload, including same-host traffic.	Applications and workloads
Per-request decisions	Authorization runs on every connection, against an exact admitted set.	Applications and workloads
Evidence that stands up	Every hop names a verified identity, and no record holds a replayable credential.	Visibility and analytics
Automated posture	Onboarding by policy. Enforcement learned from traffic, then sealed.	Automation and orchestration

WHAT THIS PAPER DOES NOT CLAIM

Greymatter does not issue human identities or decide entitlements. It does not assess device posture, and it does not classify or protect data at rest. Those belong to other parts of an enterprise architecture. Greymatter feeds them: identity to ICAM, records to the SIEM, findings to data governance.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

15

Operational and mission benefits

Each outcome, what produces it, and its status.

OUTCOME	WHAT PRODUCES IT	STATUS
Reduced lateral movement	Every connection authorized against an exact admitted set, including on the same host	FIELDED
Verifiable workload identity	Identity issued only after the platform attests the workload	FIELDED
No certificate burden	Short-lived identities replaced automatically, with no restart	FIELDED
People verified once, at the edge	OIDC, PKI with OCSP, or OpenShift sign-in, from the program's own authority	FIELDED
Proof the edge certificate is good	OCSP stapling, verified and refreshed by the platform, reloaded with no restart	FIELDED
Impersonation closed	Client-set identity headers recorded as claims, and gates that admit only verified users	FIELDED
Attribution to a person, hops away	Identity context and the carried user, data plane to data plane over mutual TLS	FIELDED
Per-person AI governance	Token budgets, Guardrails findings and Ask Greymatter answers tied to the person	FIELDED
Governed MCP tool access	The MCP Router forwards the verified user and grants tools by role	PREVIEW
Audit records hold no secrets	One keyed fingerprint per credential, or none, in every record and log	FIELDED
Enforcement without a manual inventory	Admission derived from observed traffic, then sealed	FIELDED
Disconnected operation	Local identity issuance and a mirrored, digest-pinned installation	FIELDED
Agents keep their person	Greymatter keeps the person on work that continues after the request ends	FIELDED
Single-user workloads name their owner	Calls a workload makes on its own still name the person it runs for	FIELDED

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

16

Evidence for security frameworks

Identity frameworks require that every access is authenticated, every identity is managed, and every action is traceable to who took it. FIELDED

USE WHAT YOU HAVE

Greymatter works with the identity provider and certificate authority an organization already runs.

Frameworks that govern access agree that a network location is not an identity. They require that each request is authenticated, that access follows policy, and that a record links each action to a person or workload. Greymatter verifies identity on every hop and keeps that link in the record.

FRAMEWORK	WHAT IT REQUIRES	WHAT GREYMATTER SUPPLIES
ISO/IEC 27001:2022	Annex A 5.15 access control, 5.16 identity management, 8.5 secure authentication	A verified identity for every workload and person, on every hop
ISO/IEC 27001:2022	Annex A 8.15 logging	Each record names the verified caller and the person it acts for
EU NIS2 Directive	Article 21: access control policies, and multi-factor or continuous authentication	Identity checked on every request. Sign-in and multi-factor stay with the identity provider
NIST SP 800-207	Access granted per session, by dynamic policy, with authentication enforced before access	Identity checked on every connection and request, with headers from clients treated as claims
United States CISA Zero Trust Maturity Model	Identity pillar: phishing-resistant authentication, continuous validation, visibility	Workload identity from certificates, verified people from tokens, and the identity records
United States DoD Zero Trust Strategy	Users pillar, with visibility and analytics	Person and workload on every record, and impersonation limited to what policy allows

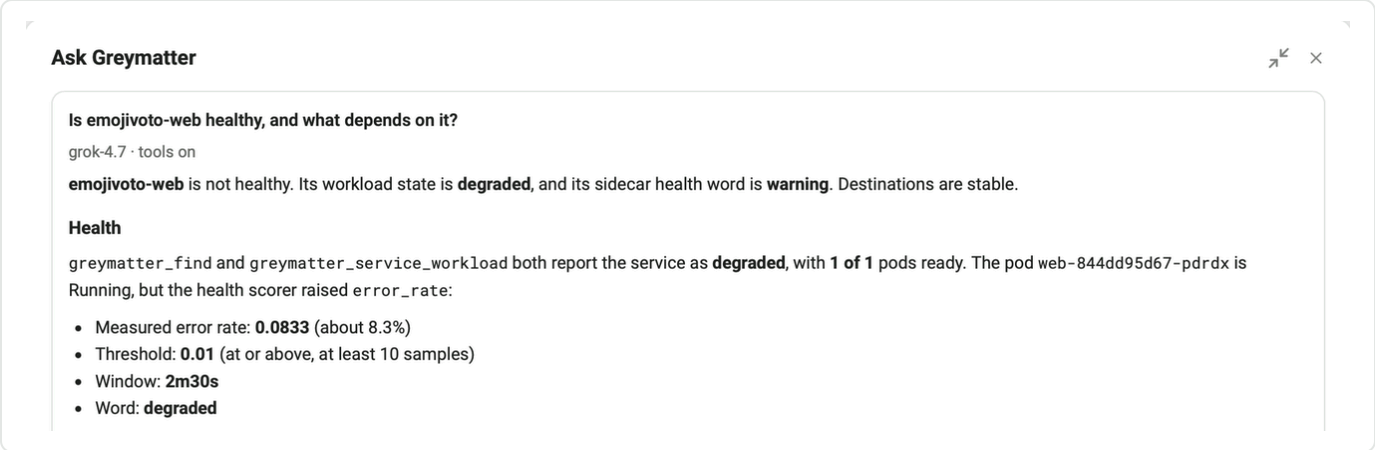
Using Greymatter does not make a system compliant with any framework. It supplies controls and evidence an assessor can test. National rows come last.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

17 Ask Greymatter

Ask Greymatter answers questions about who called what, and for whom, from the same records this paper describes. FIELDED

FIGURE 12 · ASK GREYMATTER IN THE CONSOLE



READS AS YOU

Each tool reads the platform as the person who asked, and only reads. An answer holds only what that role allows.

GOVERNED LIKE ANY AI USE

Guardrails and DLP check each model call. The audit trail names the person and the agent.

Ask Greymatter is on every page of the Console. A person types a question in plain language. A model the organization chooses answers it from the platform's own data, through read-only tools.

For identity, it reads the request records and the callers each service admits.

Tools come in groups: Health, Security, Traffic, Changes and Docs. Health and Docs are on by default. A person turns on the others in the Tools menu.

Questions a team can ask

“Is any system acting for a user it should not?”
`Security` · the records that carry a system and a user

“Show me the requests alice@example.com made in the last hour.”
`Security` · the request records for one identity

“Which callers does the orders service admit?”
`Security` · each listener's admitted callers

“Which identities were seen for the first time recently?”
`Security` · identity activity in the request records

“Who reached the payments namespace today, and how?”
`Security` · who reached one namespace

“Which identity header should my application trust?”
`Docs` · the Greymatter documentation

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

Key takeaways

1

Trust comes from identity, not location

The data planes mutually authenticate every connection and authorize it against an exact admitted set.

2

The edge verifies people once, then the mesh carries them

The edge verifies the person. The data planes carry that person to every hop, and no application can substitute another.

3

Claims never become identity

Greymatter records the headers a caller sets and never trusts them. Gates admit only a person this hop verified.

4

The record is evidence without secrets

Records keep who, what, when and how. They fingerprint or withhold every replayable credential.

5

Teams do not modify applications

No certificates, token checks or identity headers in mission software. The platform applies all of it.

6

Identity belongs to the program

SPIFFE for workloads, the program's own ICAM for people. Both travel with the program.

IN ONE SENTENCE

Greymatter establishes trust through verifiable identity rather than network location. SPIFFE and SPIRE identify every workload, and the program's own authority identifies every person. A record of both, on every hop, holds no secret.

© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

Greymatter.io

See every call. Inspect every prompt. Prove every control.

LEARN MORE

greymatter.io

Greymatter.io is a trademark of Greymatter.io, Inc.

© 2026 Greymatter.io, Inc.