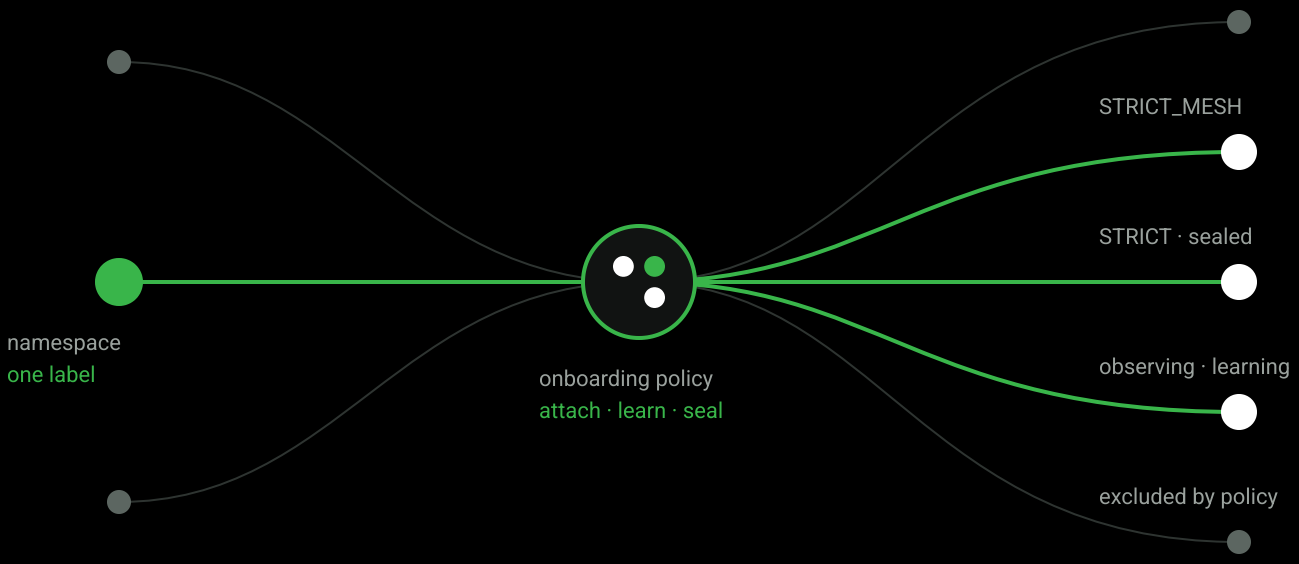
Greymatter.io

WHITE PAPER
GREYMATTER 2.5.0 · NOVEMBER 2026



Autonomous onboarding: label a namespace, get a zero trust mesh

How Greymatter enrolls, attaches, learns and hardens every workload on its own, under policy the platform team writes once.

© 2026 Greymatter.io, Inc.

Contents

		SECTION	PAGE
THE CASE	1	Executive summary	1
	2	Why meshes stall at partial coverage	3
	3	The architecture in one picture	4
ONBOARDING	4	One label: what Greymatter does on its own	5
	5	The Global Onboarding Policy	7
	6	The Tenant Onboarding Policy	9
	7	The posture climb: observe, learn, seal	10
	8	Learning the traffic	12
THE MESH	9	Encryption and identity on every hop	13
	10	Traffic control and resilience	14
	11	Visibility built in: one product, no assembly	16
	12	Forensics and control from the same evidence	19
PROOF	13	Posture export: from learned to written	20
	14	How Greymatter differs from a typical mesh	21
	15	Where it runs	25
	16	Operational and mission benefits	26
	17	Evidence for security frameworks	27
	18	Ask Greymatter	28

How to read the status marks

Every capability in this paper carries one mark. Nothing marked Fielded is aspirational.

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

A service mesh is only as strong as the workloads it covers. Greymatter puts every workload under zero trust on its own, from one label, under policy the platform team writes once.

IN BRIEF

Label a namespace. Greymatter enrolls it, attaches its data plane, learns its traffic, encrypts every hop, and hardens it to the posture your policy names.

WHO THIS IS FOR

Platform owners, security architects, program offices and assessors who must prove that every workload is in the mesh, not only the ones a team remembered.

Most meshes stall in the same place. The mesh itself installs in an afternoon. Then every team has to opt in, workload by workload. Each team annotates the deployment, writes the routing and writes the authorization policy. Then it rotates the certificates and keeps it all current. Coverage grows only as fast as the busiest team. The workloads nobody touched stay outside the mesh, unencrypted and unattributed.

Greymatter turns that around. A platform administrator writes one Global Onboarding Policy (GOP). It says which namespaces belong to the platform, who may write their policy, and how strict they must be. A team that needs something different writes a Tenant Onboarding Policy (TOP) inside the limits the GOP sets. Greymatter does the rest.

It attaches the data plane to every workload, including the ones that already run. It learns which ports each workload serves, which protocol each port speaks, and who calls it. It encrypts every hop with mutual TLS. It climbs from observing to strict on a schedule the policy sets. Before it refuses anyone, it names every caller the next step will refuse. It records every connection, and it keeps a posture record that nobody can edit by hand.

THE CENTRAL POINT

Greymatter makes zero trust the default state of a namespace, not a project each team must finish.

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

Six properties a program office can test

One label onboards

FIELDED

Labeling a namespace enrolls it and attaches every workload in it, existing ones included. No per-service configuration.

Policy written once

FIELDED

The GOP sets the rules for every namespace. A TOP refines them per team, never below the floor the GOP sets.

Strict by default

FIELDED

New namespaces refuse plaintext between workloads from the start. Every hop runs mutual TLS.

Learns before it seals

FIELDED

Greymatter observes callers, then narrows admission to them. It names who a seal will refuse before the seal lands.

Modern cryptography

FIELDED

TLS 1.3 with post-quantum key exchange by default, on FIPS 140-3 validated cryptography.

A record of every hop

FIELDED

Greymatter attributes every connection, and the posture record shows what it learned and when.

STATED PRECISELY, SO IT IS NOT OVER-READ

Greymatter onboards the workloads in namespaces a rule claims. A team can exclude a workload, a port or an address range, and an excluded path skips the data plane entirely. The policy shows every exclusion.

Strict postures need an identity plane. With Greymatter's built-in workload identity, or with certificates you supply, every rung enforces. Without either, Greymatter observes and reports, and refuses nothing.

INDEPENDENT RECOGNITION

GigaOm names Greymatter a Leader and an Outpacer in its Radar for Service Mesh, version 6, published in November 2026. Of the 15 products it scores, Greymatter has the highest average for governance, risk, compliance and security. See section 14.

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

2

Why meshes stall at partial coverage

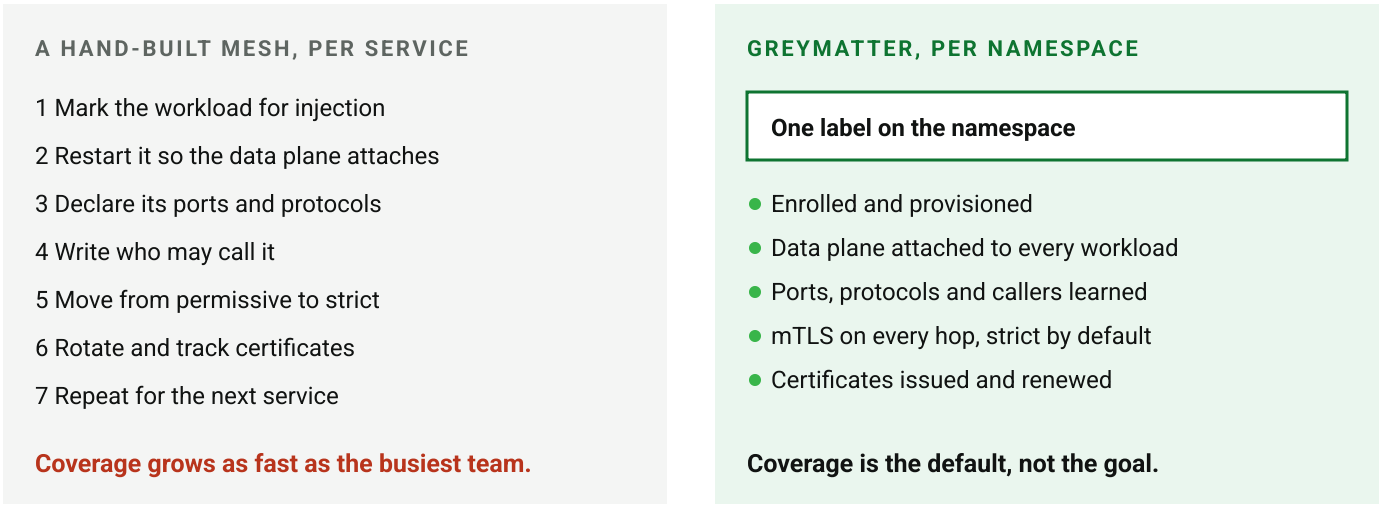
The hard part of a mesh is not encryption. It is getting every workload into it, and keeping it there.

WHY IT MATTERS

An assessor asks which workloads are in the mesh. A partial answer means the zero trust claim holds only for the services someone remembered.

1. **Opt-in by service.** Each team must mark, restart and configure each workload. Teams under deadline defer it.
2. **Policy by hand.** Someone must write who may call each service, from memory or from a diagram that is already out of date.
3. **Permissive forever.** A mesh that starts permissive so nothing breaks often stays permissive, because nobody knows what strict would refuse.
4. **Upkeep on people.** Certificates, exceptions and drift land on the one engineer who understands the scripts.

FIGURE 1 · PER-SERVICE WORK COMPARED WITH ONE LABEL

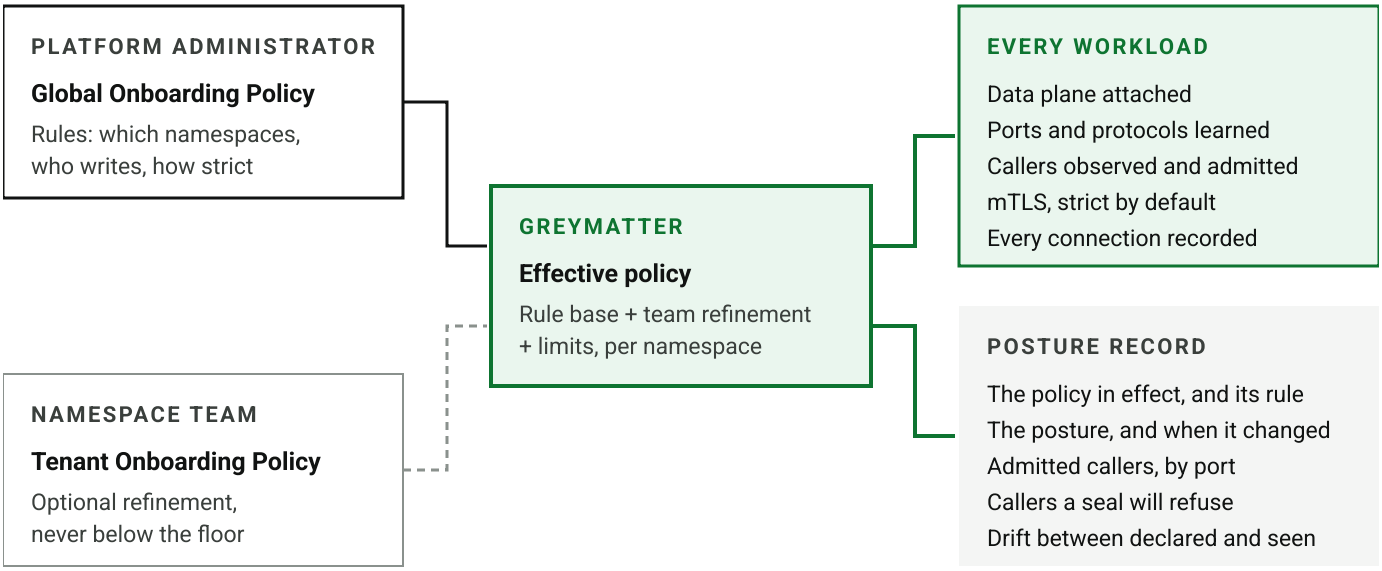


© 2026 Greymatter.io · Greymatter.io is a trademark of Greymatter.io, Inc.

3 The architecture in one picture

Two policies, one effective result per namespace, and one record of what Greymatter did. FIELDED

FIGURE 2 · FROM POLICY TO POSTURE



OBJECT	WHO WRITES IT	WHAT IT DECIDES
Global Onboarding Policy	Platform administrator	Which namespaces belong, who writes their policy, and the floor
Tenant Onboarding Policy	Namespace team, when allowed	How this namespace differs, within the rule's limits
Posture record	Greymatter alone	What is in effect, what Greymatter learned, and whom a seal will refuse
GSL or custom resources	Teams that declare a service	Named routes, egress and filters for declared services

A service declared in GSL or custom resources keeps its declared configuration. The onboarding policies govern every workload nobody declared.

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

4

One label: what Greymatter does on its own

The platform administrator writes policy once. Everything after the label is automatic. FIELDED

THE DEFAULT INSTALL

The install writes one rule. It claims every namespace that carries the platform label, at STRICT_MESH. A team can soften to OBSERVE, or tighten to Strict Identity, which matters more as AI agents call services on their own.

WHO ACTS

Nobody. On the default install, a team does not need to apply anything in its namespace.

On a default install, labeling a namespace is the whole onboarding procedure. The label enrolls the namespace and the default rule supplies its policy. A team that wants something different writes a TOP, within the limits the rule sets.

The table lists the work Greymatter does without a person. Each row is an action a platform team would otherwise script, schedule or ask application teams to do.

GREYMATTER DOES THIS	WHAT HAPPENS
Enroll and provision	The rule claims the namespace and Greymatter installs its tenant services and pull credentials.
Label by pattern	A rule that matches by name, such as `*-prod`, labels a new namespace the moment it exists, and records that it did.
Attach new workloads	Each new workload gets its data plane at creation, named after the controller that owns it.
Attach existing workloads	Greymatter recreates running workloads through their controllers so they come back with a data plane.
Redirect traffic	Every inbound and outbound connection goes through the data plane from the moment the workload starts.
Issue identity	Each workload gets its own workload identity and certificates that renew without restarts.
Learn ports and protocols	Greymatter classifies each port from what the workload declares, its probes and the traffic itself.

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

GREYMATTER DOES THIS	WHAT HAPPENS
Keep health probes working	Greymatter rewrites probes onto health listeners it creates, so strict ports do not break them.
Observe callers	Greymatter attributes every connection and adds each caller to the admitted set for its port.
Harden on schedule	The namespace climbs to its declared posture through the hardening window the policy sets.
Resilience by default	Callers retry failed connections, stop sending to failing pods, and balance on least request.
Report drift	Protocol drift and repeated refusals appear in the posture record, without any change to traffic.
Detach cleanly	When a rule stops claiming a namespace, Greymatter removes what it installed and leaves human labels in place.

Safe when it cannot act

NEVER HALF-ATTACHED

A workload whose owner Greymatter has not yet seen waits for its controller to retry. It never starts outside the mesh for its whole life.

- **Attach never blocks a deploy.** A failure while attaching creates the workload without a data plane, and the posture record counts it, so nobody mistakes it for covered.
- **Standalone pods stay up.** Greymatter never deletes a pod that no controller would recreate. It reports the pod, and the team replaces it.
- **A management plane with no single point.** It runs three copies by default, one elected leader at a time, and keeps a copy up through node maintenance. A program can run up to ten.
- **Configuration survives an outage.** Data planes keep the configuration they last received while the management plane is unreachable.
- **No silent downgrade.** A call that cannot reach mTLS does not fall back to plaintext between mesh workloads.

EVERY WORKLOAD, NOT EVERY WILLING TEAM

Coverage follows the namespace, not the team. Nobody declared or configured some workloads, and nobody knew about them. Those workloads get the same data plane, identity and record as a workload a team built for the mesh.

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

5

The Global Onboarding Policy

One object per platform settles three questions for every namespace: is it ours, who writes its policy, and what does it get. FIELDED

ORDERED RULES

The first rule whose target matches a namespace wins. Specific rules go first and the catch-all last.

BOUND TO ONE PLATFORM

The policy names the platform it governs. A copy applied to the wrong cluster governs nothing.

A rule targets namespaces by label, by whole-name pattern, or both, minus any exclusions. A pattern is a standing grant: a namespace that matches it joins the platform the moment someone creates it, with no ticket and no commit. Greymatter labels it, and records that it did.

The cluster's own system namespaces and Greymatter's own namespaces never match a rule. An administrator can also pin a namespace to a named rule, skipping the ordered match.

Who writes the namespace's policy

SOURCE	THE RULE SUPPLIES	THE TEAM'S TOP	WORKLOADS ATTACH
`admin`	The whole policy	Refused, with a message naming the rule	At once, existing ones included
`shared`	The base policy	Optional. Refines the base within limits	At once, at the rule's posture
`tenant`	Nothing	Required. It is the whole policy, within limits	Only after the team applies one

The default rule uses `shared`: workloads attach with no team action, and a team can still soften, exclude or grant within the floor.

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

Bound what a team may write

REFUSED AT WRITE TIME

The cluster refuses a team policy that crosses a limit, with the rule and the limit named in the message.

Each rule can set limits on the teams it governs. The floor is the most important: a team may pin its namespace at or above the floor, never below. Other limits reserve the settings that would let traffic skip the data plane, hand-written grants, re-learning, the mesh port and the data plane size.

LIMIT	WHAT IT RESERVES
Minimum posture	The floor. A team may pin at or above it, never below
Bypass	Ports and address ranges that would skip the data plane
Grants	Hand-written admissions that skip observed evidence
Learn	Re-learning, which reopens the namespace to observation
Data plane size	The resources of automatically attached data planes

Preview a rule before it goes live

DRY RUN

A dry rule claims nothing. Namespaces fall through to the live rules as if it were not there.

Greymatter evaluates and reports a rule marked as a dry run, but the rule changes nothing. Its report lists the namespaces it would newly enroll and the namespaces whose existing workloads it would attach. It also lists the namespaces it would take over from another rule, with that rule's name and source. An administrator reads the report, then removes the flag.

Offboarding works the same way in reverse. When no rule claims a namespace, Greymatter removes what it installed there. It keeps any label a person or a GitOps controller wrote.

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

6

The Tenant Onboarding Policy

A namespace team's declaration for the workloads Greymatter onboards on its own. Everything in it is optional on the default install. FIELDED

ONE PER NAMESPACE

The policy has one allowed name. Greymatter stamps the default posture into it, so the stored object always says what it does.

CHANGES WITHOUT RESTARTS

Most workload settings reach running data planes without restarting pods. Open connections drain first.

The TOP is how a team says what is different about its namespace. It sets how strictly the workloads refuse callers and who may call them. It also sets which traffic skips the data plane, how large the data plane is, and settings for individual workloads. Greymatter combines it with the rule's base and limits into the effective policy, and reports the result field by field.

SETTING	WHAT THE TEAM CONTROLS
Posture	OBSERVE, STRICT_MESH or STRICT, at or above the floor
Identity seal and window	Whether admission narrows to observed callers, and how long to observe first
Grants	Named callers admitted to named ports, across namespaces
Exclusions	Workloads, ports and address ranges that skip the data plane
Ingress admissions	Ports that also accept plaintext from callers outside the mesh
Caller settings	Retries, outlier detection, connection limits and balancing for every caller
Timeouts and limits	Request deadlines, idle timeouts and header size, per workload
Service card	The owner and capability the Console shows for each service
Data plane size	CPU and memory for the namespace or one workload
Guardrails and DLP	Inspection on selected workloads, inbound and outbound

A workload entry names the workload by its full mesh identity, so a cross-namespace caller or grant is always unambiguous.

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

7

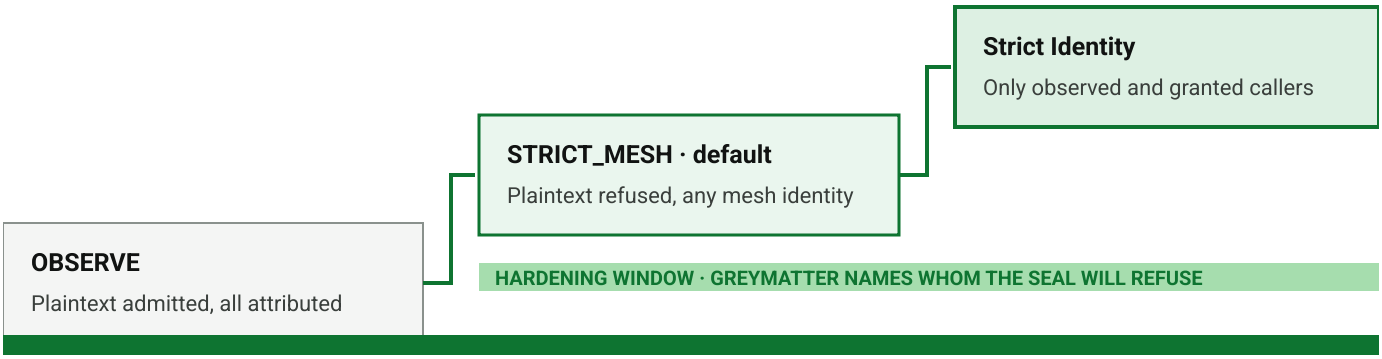
The posture climb: observe, learn, seal

Greymatter learns who calls each workload before it refuses anyone. It names the callers a seal will refuse before the seal lands. FIELDED

FIGURE 3 · THREE RUNGS AND ONE SEAL

THE ADMITTED SET GROWS AT EVERY RUNG

Greymatter adds callers as it sees them. Only a re-learn, a deleted policy or offboarding removes them.



TRANSPORT AND IDENTITY

Posture decides whether Greymatter refuses plaintext. The identity seal decides which identities it admits. They are separate switches.

NEEDS STRICT

The identity seal requires STRICT, and either grants or a hardening window, so there is always a caller list to seal from.

OBSERVE admits plaintext from anyone, prefers mTLS, and attributes every connection. **STRICT_MESH**, the default, refuses all plaintext and admits any caller with a valid mesh identity. **STRICT** refuses plaintext from the moment a team declares it. With the identity seal, which the Console calls Strict Identity, admission narrows to the callers Greymatter observed and the callers a grant names.

A hardening window holds the namespace at STRICT_MESH for a period the policy sets before a seal that will refuse someone. The window gives Greymatter time to see every caller.

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

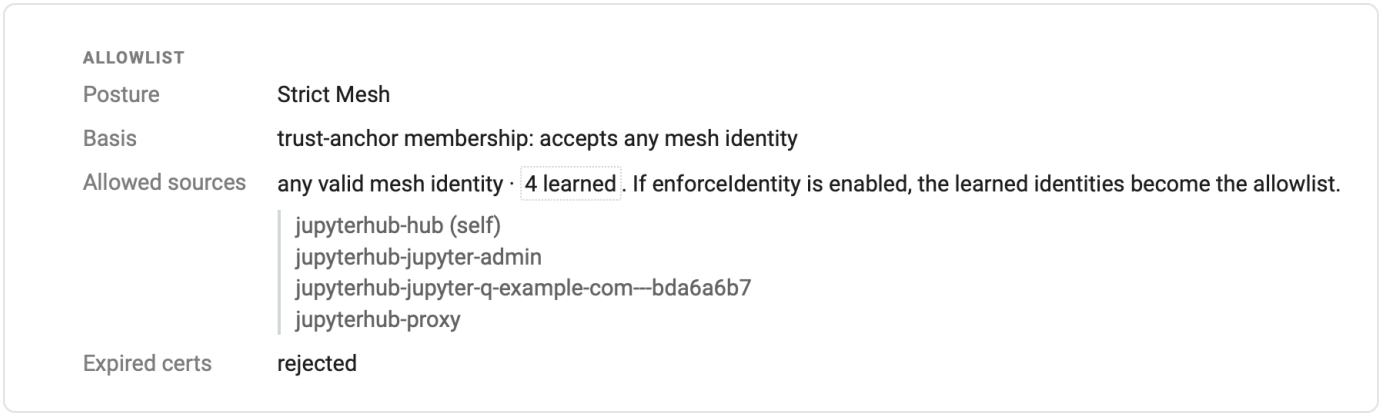
No surprise refusals

TWO KINDS NAMED

Plaintext callers that STRICT already refuses, and mesh callers outside the admitted set that the seal will refuse.

Until the seal lands on a STRICT declaration, the posture record lists the callers that the seal will refuse, by workload and port. It also states when the seal lands. A team reads the list, grants the callers it wants, and lets the seal land. Greymatter refuses nothing the team could not see coming.

FIGURE 4 · LEARNED IDENTITIES ON A LISTENER, IN THE CONSOLE



Grants, re-learning and impersonation

ONLY GROWS

Greymatter never drops a caller for going quiet. Admission is evidence, and time does not delete evidence.

- **Grants.** After the seal, a grant is the only way to admit a caller Greymatter did not observe. Removing the grant revokes it, and nothing else reopens.
- **Re-learning.** A new trigger value drops the namespace back to observing once, forgets the admitted callers, and climbs again through the window. It is the only way to revoke an observed caller, short of deleting the policy.
- **Impersonation.** Greymatter records callers that act for a person before the seal. After the seal, the gate admits those callers and the ones the team names, spelled exactly as presented.
- **Never backwards.** A restart never lowers a posture already reached.

WHY THIS MATTERS FOR A PROGRAM

Most programs never reach strict because nobody can say what strict will break. Greymatter settles that question from evidence, in advance, per caller and per port. Hardening becomes a scheduled event instead of an outage risk.

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

8

Learning the traffic

Greymatter works out what each workload serves and calls, so nobody has to describe it first. FIELDED

SOURCES, IN ORDER

A declared service, the port's declared protocol, its name, the container's port name, its probes, the traffic itself, then the history of what Greymatter saw.

REPORTED, NOT ACTED ON

Drift is evidence for a person to read. Greymatter does not change traffic because of it.

Ports and protocols. Greymatter treats each port as HTTP, HTTP/2, gRPC, TLS or TCP, and records what established that class. A port nothing has classified yet stays out of the record rather than guessed.

Health probes. Greymatter rewrites a workload's network probes onto health listeners it creates, so a port that refuses plaintext still responds to its health checks.

Outbound calls. The data plane captures every outbound connection, whatever its port. A call to another mesh workload goes over mTLS. A call to anything else reaches the destination the application dialed, and the data plane records it, by name where Greymatter can resolve one. Protocols where the server speaks first, such as databases and mail, keep working.

Non-web traffic. TCP services get the same identity and mTLS as web services. Greymatter also supports Kafka traffic.

DRIFT GREYMATTER REPORTS	WHAT IT MEANS
Protocol drift	A port's traffic contradicts the protocol a team declared or Greymatter classified
Refusal pattern	Greymatter keeps refusing one caller at the same port
Path drift	A port declared as HTTP keeps receiving traffic that is not HTTP

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

9

Encryption and identity on every hop

Mutual TLS between every pair of workloads, with modern cryptography switched on before anyone configures it. FIELDED

VALIDATED MODULE

The FIPS 140-3 cryptography is NIST CMVP certificate 5191, the Greymatter.io Cryptographic Module. It validates the module, not the platform.

TWO IDENTITY PLANES

Built-in workload identity is the default. A program can instead supply certificates from its own authority.

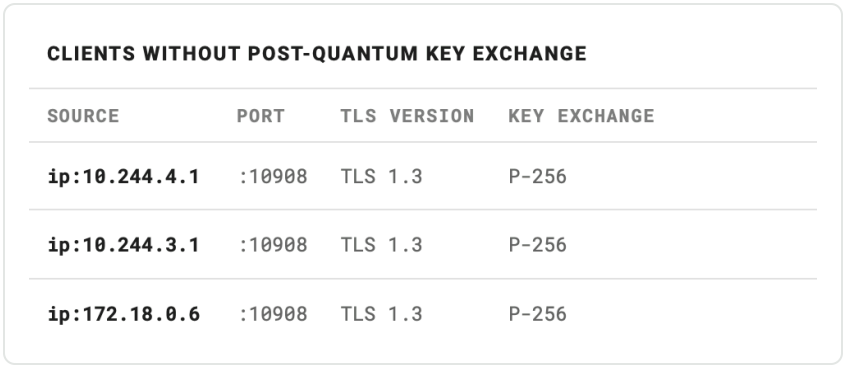
Workload identity by default. Each data plane receives a SPIFFE identity that names its namespace and service. Certificates arrive over the data plane's secret API and renew without anyone touching a secret or restarting a pod.

Your own authority when required. The program's certificate authority can be the trust anchor instead. The platform then mounts certificates you supply, copies them into each namespace a rule names, and keeps them synced. Rotation is then yours to schedule.

TLS 1.3 and post-quantum key exchange. Mesh hops negotiate TLS 1.3 and the hybrid post-quantum group X25519MLKEM768 by default. A peer that supports only TLS 1.2 still connects.

FIPS 140-3, always. Greymatter builds every data plane on FIPS 140-3 validated cryptography, and no setting turns it off. The Console reports the TLS versions, cipher suites and key exchange groups in use, and the share of handshakes that used post-quantum key exchange.

FIGURE 5 · CLIENTS WITHOUT POST-QUANTUM KEY EXCHANGE, IN THE CONSOLE



DEFAULT	VALUE	A USER CAN CHANGE IT
Namespace posture	STRICT_MESH	Yes, within the rule's floor
Key exchange	X25519MLKEM768 first	Yes, per listener
Data plane cryptography	FIPS 140-3	No. Always on

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

10

Traffic control and resilience

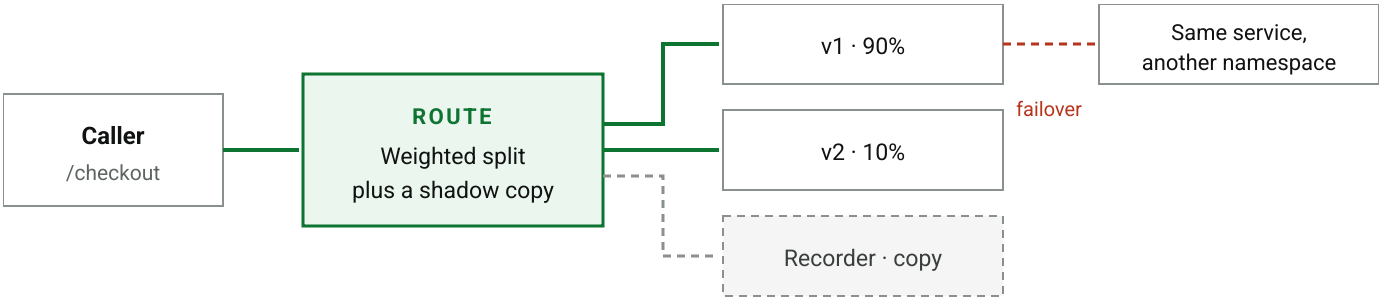
Automatically onboarded workloads get sensible resilience with no configuration. Declared services get full control of every route. FIELDED

SET BY THE CALLEE

The called workload declares how its callers behave, and every caller in the mesh applies it without restarting.

For every onboarded workload. Callers retry failed connections within a shared retry budget, stop sending to pods that keep failing, and choose among pods by least request. Each of these is a default the called workload's team can tune in its TOP.

FIGURE 6 · A WEIGHTED RELEASE WITH A SHADOW COPY AND FAILOVER



RELEASE SAFELY

A new version takes a small share first. One change moves everyone back.

Progressive delivery. A route sends each upstream its weighted share of requests. A shadow upstream receives a copy of a share of requests, and the data plane discards its responses. A new version can then see production traffic without serving it.

Failover. An upstream lists ordered fallbacks, up to 16. Each fallback receives traffic only while every earlier one has no healthy endpoint, and the same health checks and outlier detection check every target.

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

CONTROL	WHAT A TEAM SETS
Weighted routing	Up to 32 upstreams per rule, each with a weight, served or shadowed
Failover	Ordered fallbacks across namespaces, judged by health checks
Health checks	Active checks of each endpoint and each failover target
Outlier detection	Remove endpoints that fail in a row or fall below the success rate
Circuit breakers	Connections, pending requests, requests and retries per upstream
Retries and timeouts	Conditions, attempts, back-off, per-try and whole-request deadlines
Load balancing	Round robin, least request, ring hash, random or Maglev, zone aware
High priority	A separate connection pool and limits for one rule's requests

SAME PIPELINE

Identity, mTLS, routing, inspection and audit run on the same data plane at the edge and between workloads.

Declared or discovered, one data plane

A team that needs named routes, egress or filters declares a service in GSL or as custom resources. A declared service always wins over discovery for its own pods. Every other workload in the namespace stays onboarded automatically. Teams move a workload from discovered to declared when they need more control, and nothing else in the namespace changes.

The same data plane serves as the north-south edge, with TLS to outside clients, OAuth2, WAF and App Guard. No separate gateway product with its own identity model sits in the path.

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

11

Visibility built in: one product, no assembly

Greymatter shows the mesh, the traffic and the infrastructure under it, out of the box. There is no third-party stack to install, wire together or keep current. FIELDED

SHIPS WITH THE PLATFORM

The platform always deploys its own log store, metrics store and datastore. A program adds nothing to see what this paper describes.

ONE VIEW OF THE TRUTH

The Greymatter API decides health once. Every Console page shows the same word, with the reasons that chose it.

A typical mesh hands its telemetry to other products, one each for metrics, traces, logs and topology. Each product has its own install, upgrade, schema and access model. Visibility then depends on how well a team assembled and maintains that stack.

Greymatter is one product. Its data planes, management plane and node agents feed stores the platform runs itself. The Console reads them through one API, which applies each person's access before a response leaves it. Every request the Console sends carries the signed-in person's own credentials, so the audit names who looked.

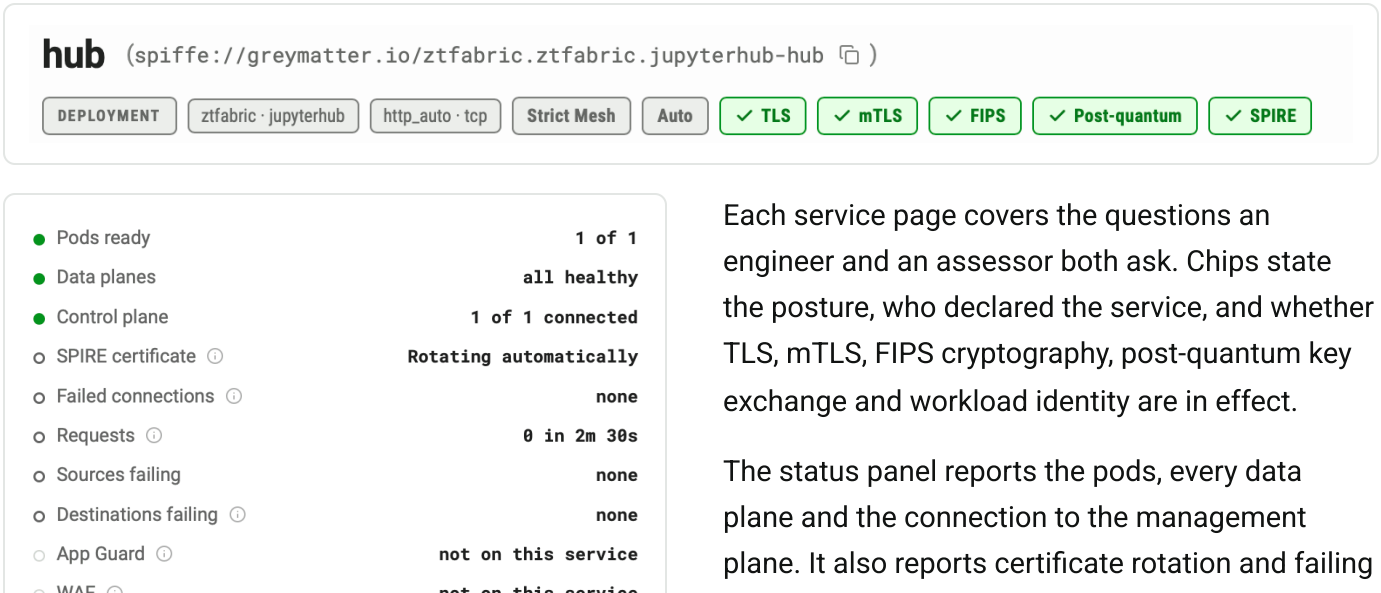
LAYER	WHAT GREYMATTER SHOWS, WITH NO ADDED TOOLING
The mesh itself	The management plane, every data plane and its connection, node agents judged per node, configuration writes, and certificate rotation and expiry
The traffic	Every caller of every service: transport, TLS version, post-quantum key exchange, request counts, failures and the exact cause of each
The infrastructure	Namespaces, workloads and pods with one health word each, the node each pod runs on, and callers that bypassed mTLS by node
The policy	Posture, learned identities, per-port allowlists, exclusions and what a seal will refuse

NOTHING TO ASSEMBLE

- No metrics, tracing or log pipeline to install before the mesh is visible.
- No dashboards to build, and no collectors to upgrade in step with the mesh.
- No second access model to keep in sync with the platform's roles.

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

FIGURE 7 · A SERVICE PAGE IN THE CONSOLE

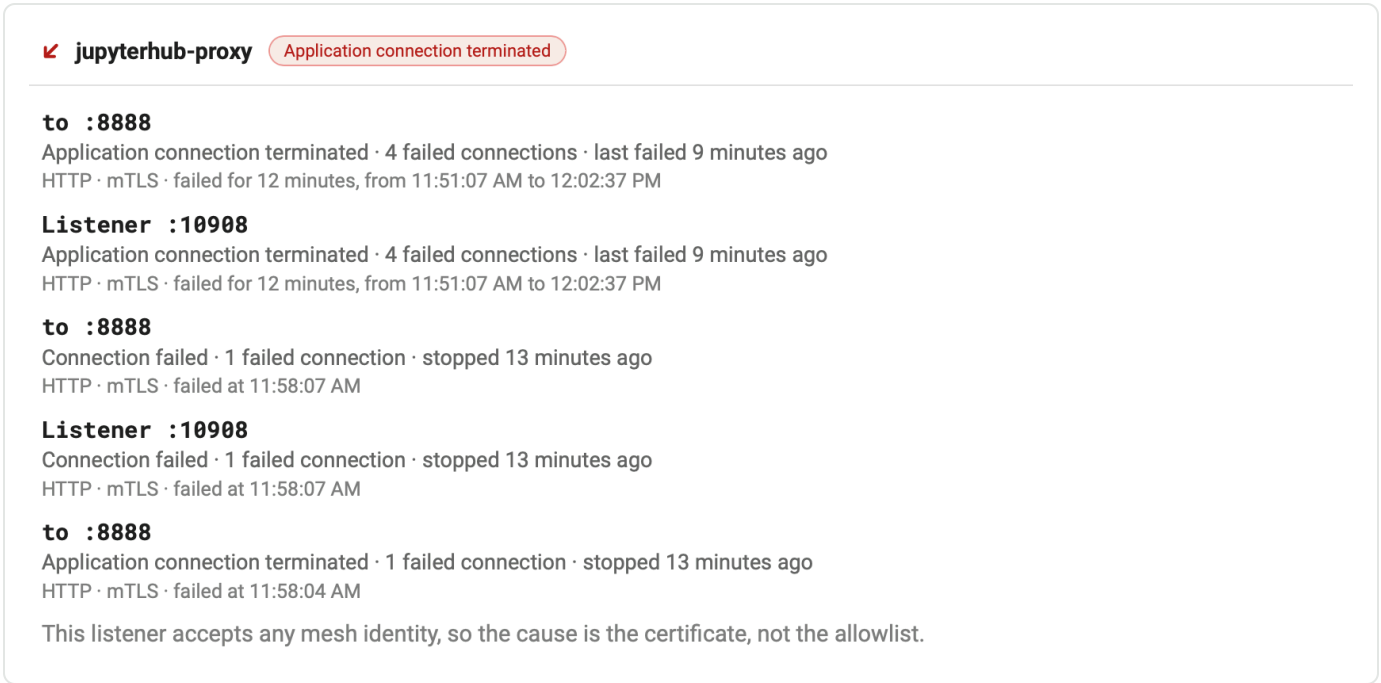


Each service page covers the questions an engineer and an assessor both ask. Chips state the posture, who declared the service, and whether TLS, mTLS, FIPS cryptography, post-quantum key exchange and workload identity are in effect.

The status panel reports the pods, every data plane and the connection to the management plane. It also reports certificate rotation and failing sources and destinations, with the reason behind each word.

When the platform could not read a source, the Console says so. It leaves out a count it did not read, and never shows it as zero.

FIGURE 8 · INBOUND FAILURES, DOWN TO THE CAUSE



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

The posture record and the map

ONE WRITER

Only Greymatter writes the posture record. Nobody can edit or erase the admitted callers or promotions by hand.

Each namespace has a posture record. It holds the claiming rule, the effective policy, and the posture in force with the time it began. It also holds each workload's stage on the climb, its ports, protocols and pods, and the admitted callers and promotions per port. The Console shows the same truth as chips on every card, including the countdown while a namespace moves to Strict Identity.

WORKLOAD STAGE	WHAT IT MEANS
Observed	The namespace is at OBSERVE, a floor that never climbs
Observing	Inside the observation window
Automatic mTLS	Window elapsed, mTLS built, callers admitted as Greymatter observes them
Enforced	At least one port promoted to mTLS only

FIGURE 9 · READING THE NETWORK GRAPH

LINE · TRANSPORT

- **mTLS** · encrypted, both ends proved who they are
- **TLS** · encrypted, the caller has no workload certificate
- **Plaintext** · no encryption seen
- **Mixed** · some calls of each
- **No handshake** · TLS started and did not finish

LIGHT AND RING

- A live path, seen recently
- Calls on this path failed
- Not managed by Greymatter
- Posture and source chips on every card

Across the estate

SEVERAL PLATFORMS

Fleet lists every connected platform by cluster, with the state of its namespaces and services.

The Platform view turns the same scrutiny on Greymatter itself. It shows the health of every platform service and flags any check it could not complete. Fleet shows every connected platform in one place. Ask Greymatter answers plain-language questions from the same data, as the person who asked. Discovery reads workloads from the cluster, from service catalogs and from virtual machine inventories, so the record covers more than containers.

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

12

Forensics and control from the same evidence

The record that drives the posture climb is the record an investigator reads. There is no second system to reconcile. FIELDED

CREDENTIALS SCRUBBED

The audit records every hop with credentials already removed. A keyed fingerprint stands in where an investigator must match a credential.

SIGNED HISTORY

Greymatter keeps a signed history of configuration changes, so anyone can show that a change is unaltered.

Every connection attributed. Even at OBSERVE, Greymatter attributes each connection to the workload that made it. Each data plane writes a request audit at every hop, where it happens, with the verified person when there is one.

Admission is evidence. Each admitted caller in the posture record came from an observed call, or from a grant a person wrote. An assessor can see which, per caller and per port, and when.

Exceptions are explicit. Every path that skips the data plane is a line in a policy. That line names an excluded workload, port or address range, or a port that admits outside plaintext. The posture record echoes the exclusions in effect, and a rule can reserve them for administrators.

Control stays in policy. Posture, floors, grants and exclusions live in two declarative objects. A change is a reviewed commit. A tenant cannot write past the rule, and the cluster refuses the attempt with the rule's name.

QUESTIONS AN ASSESSOR CAN ANSWER IN MINUTES

- Which workloads in this namespace are in the mesh, and which are not, and why?
- Which callers may reach this port, and how did Greymatter admit each one?
- Who would a stricter posture refuse, before anyone turns it on?
- Which hops ran mTLS, with which TLS version and key exchange?

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

13

Posture export: from learned to written

What Greymatter learned becomes a file the team owns, ready to review, commit and reuse. FIELDED

TWO OUTCOMES

Move it to a new cluster, secured from its first pod. Pin it, so the learned posture lives in a file.

READ-ONLY TO BUILD

Building an export changes nothing on the platform. The person downloads it and decides what to apply.

Learning is useful only if a team can keep it. Posture export turns what Greymatter learned about an automatically onboarded project into a TOP. The TOP holds the project's services and the grants that reproduce who may call each one.

Move it. Stand the project up in a cluster that has never seen these services. It comes up locked down from its first pod, without observing the environment again.

Pin it. Apply the file back to the project, so the posture lives in a reviewed file instead of only in the platform. The project can then restart, clear its history or rebuild, and its rules stay in place.

The export opens from Settings in the Console, or from a namespace's posture panel, for administrators of that namespace or the platform.

THE LOOP THIS CLOSES

Observe in one environment, prove what Greymatter learned, write it down, and promote it. Each environment after the first starts sealed instead of starting over.

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

14

How Greymatter differs from a typical mesh

Most meshes give a team tools to secure its services. Greymatter secures the namespace and gives the team tools to differ.

ABOUT THIS COMPARISON

The left column describes common patterns in the service mesh market, as the GigaOm report on the next page also describes them. It does not describe any one product.

The difference is the operating model. A typical mesh is one tool among several: a gateway, a certificate manager, an observability stack and an audit pipeline. A stretched operations team must run each one and the glue between them. Every seam adds cost and risk, and the risk grows when the people who understand the glue leave.

Greymatter is one full product. Onboarding, identity, encryption, traffic control, visibility and evidence ship and upgrade together, under one policy model. The platform delivers coverage, strictness and evidence on its own.

AREA	A TYPICAL SERVICE MESH	GREYMATTER
Operations	Several tools to integrate, upgrade and keep working together	One product to install, upgrade and support
Onboarding	Teams opt in each workload, then restart it	One label onboards the namespace, existing workloads included
Default posture	Often permissive inbound until someone tightens it	STRICT_MESH from the first workload, plaintext refused
Authorization	Written by hand, per service, from memory	Learned from observed callers, then sealed, with grants for the rest
Rollout safety	Turn on strict and watch what breaks	Greymatter names the callers a seal will refuse before it lands
Policy governance	Per-namespace objects with no platform floor	A platform rule with a floor that a team cannot write below
Forensics	Mesh logs and access logs, joined later	An audit at every hop, attributed, credentials scrubbed
Visibility	Topology from configuration and metrics	Transport per line from what the wire recorded, unmanaged workloads shown
Cryptography	Post-quantum and FIPS often extra work	TLS 1.3 and X25519MLKEM768 by default, on FIPS 140-3 cryptography that is always on
Drift	Found during an incident	Protocol, refusal and path drift reported in the posture record

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

AI moved in. Enforcement moved out.

THE OTHER MODEL

Some meshes move the proxy out of the pod. A shared proxy on each node handles encryption and L4 policy, and a separate waypoint proxy adds L7 only where a team deploys one.

WHERE WE STAND

At the workload, on every hop, from the first pod. The seam is where AI traffic goes wrong, so that is where Greymatter enforces.

Node-level meshes moved enforcement out of the pod to cut cost. They did it at the moment AI agents arrived. Agents borrow a person's authority and call on that person's behalf. They carry prompts, responses and data across boundaries. They reach tools directly, at machine volume, from inside the cluster.

That traffic needs judgement at the workload, and a node-level mesh no longer stands there. Without a waypoint, a call gets encryption and L4 policy only: no request-level authorization, no inspection, no carried person and no forensic chain.

A waypoint does not close the gap. It inspects traffic away from the workload, as a separate proxy in front of the destination. The caller's side and the last hop into the pod still carry L4 only. Nothing inspects a call that does not route through the waypoint. Each of those is an open seam. The enforcement point AI needs is the one the industry has vacated.

WHAT AI TRAFFIC NEEDS	NODE-LEVEL L4 WITH WAYPOINTS	GREYMATTER, AT THE WORKLOAD
Nothing reachable by default	L4 policy at the node, with caller sets written by hand	Callers learned over a window, then sealed, with grants for the rest
The person behind the agent	Lost between hops unless each application passes it on	The verified person carried hop to hop, in the same record as the agent
Prompts and responses judged	Only where a waypoint exists	Guardrails for the OWASP LLM Top 10 at the workload, detect then enforce
Data leaving, not only arriving	Not inspected where the call starts	DLP on the workload's own outbound calls, classification markings included
Tool calls governed	A separate AI gateway product	MCP Router: tools granted by role, with call limits (Preview)
A contained blast radius	One node proxy acts for every pod on its node	Each data plane holds one workload's identity

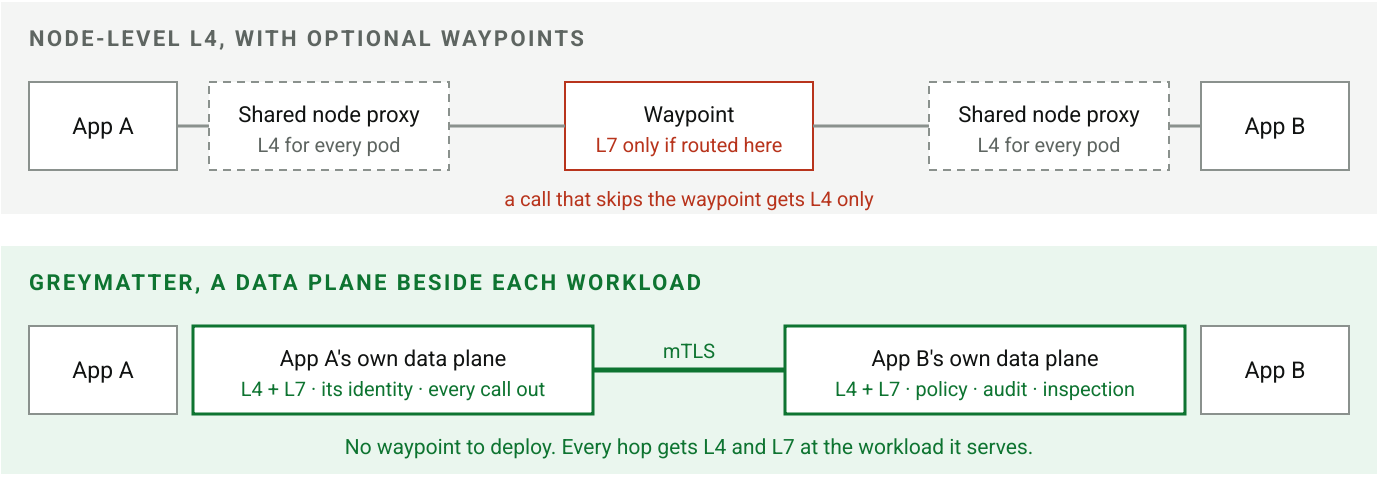
THE POINT

Moving the mesh out of the workload saves resources and gives up the place where zero trust has to hold for AI. Greymatter keeps enforcement, judgement and evidence in that place, with no SDK and no application change.

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

What Greymatter keeps, hop by hop

FIGURE 10 · WHERE L4 AND L7 RUN



CONCERN	NODE-LEVEL L4 WITH WAYPOINTS	GREYMATTER
L4 and L7 coverage	L4 at the node, and L7 only for traffic routed through a waypoint	L4 on every connection and L7 on every HTTP and gRPC port, at the workload, AI workloads included
Identity custody	A shared node proxy acts for every pod on its node	Each data plane holds only its own workload's identity
Outbound control	Seen at the node, shared by every pod there	The caller's own data plane captures every outbound call
Person carried	Lost between hops unless each application passes it on	The verified person carried from hop to hop by the data plane
Forensic trail	Fragments from waypoints, where they exist, joined in other tools	A request audit at every workload, linked into one chain per request
Extra tooling	Separate products for request audit, tracing and AI governance	Audit, forensic chains, posture and the map in the platform
What a team must add	A waypoint per namespace or service that needs L7	Nothing. Onboarding attaches the data plane

The trade. A data plane per workload costs more than a shared node proxy. Greymatter sizes it per workload.

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

What the analysts measured

GigaOm scores 15 service mesh products across four lenses. Governance, risk, compliance and security (GRCS) is the CISO's lens. It covers mTLS enforcement, workload identity, enterprise PKI, FIPS cryptography, L7 authorization, audit, supply chain and policy rollout safety. Greymatter's GRCS average is 4.9. The next highest is 4.3.

The criteria below map to the four things this paper is about: forensics, control, security management and visibility.

LEADER AND OUTPACER

GigaOm places Greymatter in the inner Leader ring, in the Disruptive Pioneers quadrant, with the Outpacer rating for market velocity.

THE SOURCE

GigaOm Radar for Service Mesh, version 6, by Ivan McPhee, published 2 November 2026. Scores run from 1 to 5.

GIGAOM CRITERION	GREYMATTER	NEXT BEST	STANDING
Audit Logging and Compliance Reporting	5.0	4.5	Only product at 5.0
mTLS Enforcement Posture	5.0	4.5	Only product at 5.0
Progressive Delivery and Advanced Traffic Control	5.0	4.5	Only product at 5.0
Multiprotocol Support	5.0	4.5	Only product at 5.0
Runtime Security and Observability Convergence	5.0	4.5	Only product at 5.0
FIPS and Regulated-Environment Cryptography	5.0	5.0	Highest score
Policy Rollout Safety	5.0	5.0	Highest score
Post-Quantum Cryptography	5.0	5.0	Highest score
GRCS lens average	4.9	4.3	Highest of 15

Next best is the highest score among the other 14 products. GigaOm also notes trade-offs. Greymatter keeps a data plane beside each meshed workload. Its east-west routing uses GSL and custom resources rather than Gateway API.

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

15

Where it runs

The same onboarding model in every environment the mission uses, connected or not. FIELDED

SAME POLICY EVERYWHERE

The GOP and TOP mean the same thing in every cluster, so a posture export moves between them.

DISCONNECTED

The platform installs and pulls from a mirror registry you control, for offline and air-gapped clusters.

Public and sovereign clouds. Greymatter runs in public clouds, in sovereign and government clouds, and on premises. The identity model, the onboarding policies and the posture record do not change between them.

Several platforms in one cluster. One installation serves every platform it can see. Each platform has its own name, its own namespace and its own onboarding policy, and they share the cluster's identity decision.

Enterprise distributions. Greymatter installs from the OpenShift console or with Helm, and fits GitOps workflows such as Argo CD and Flux.

Sized per workload. The data plane size for automatically onboarded workloads comes from the platform default, the rule, the namespace or the workload. A rule can reserve the setting.

NEED	HOW GREYMATTER MEETS IT
Offline install	Mirror registry and pull credentials copied into every enrolled namespace
Existing CA	Supplied certificates as the trust anchor, synced per rule
New namespaces at scale	Name patterns that enroll a namespace the moment it exists
Separate programs	Several platforms in one cluster, each with its own policy

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

16

Operational and mission benefits

Each outcome, what produces it, and its status.

OUTCOME	WHAT PRODUCES IT	STATUS
Full coverage	One label onboards a namespace, existing workloads included	FIELDED
Strict from the start	STRICT_MESH by default, refusing plaintext between workloads	FIELDED
Policy written once	A GOP with ordered rules, sources and a floor	FIELDED
Teams keep autonomy	A TOP refines the rule within limits the cluster enforces	FIELDED
Least privilege without guesswork	Observed callers sealed into the admitted set, plus grants	FIELDED
No surprise outages	Greymatter names the callers a seal will refuse, in advance	FIELDED
Safe rule changes	Dry-run rules report what they would claim and attach	FIELDED
Modern cryptography	TLS 1.3 and post-quantum key exchange by default, FIPS 140-3 always	FIELDED
Resilience without tuning	Retries, outlier detection and balancing for every caller	FIELDED
Controlled releases	Weighted splits, shadow copies and ordered failover	FIELDED
Evidence an assessor can read	Posture record, per-hop audit and signed change history	FIELDED
Visible edges	Transport per connection and unmanaged workloads on the map	FIELDED
Portable posture	Export what Greymatter learned and start the next cluster sealed	FIELDED

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

17

Evidence for security frameworks

Zero trust frameworks require encryption, segmentation and continuous monitoring on every path. Greymatter makes them the default state of a namespace. FIELDED

POLICY WRITTEN ONCE

One onboarding policy carries the control to every workload a rule claims, including the ones nobody remembered.

International and national frameworks agree on the core of zero trust. Every connection is encrypted and authenticated, access is narrowed to what a workload needs, and the posture is watched and recorded. Greymatter turns each of those into a policy setting, applied by label.

FRAMEWORK	WHAT IT REQUIRES	WHAT GREYMATTER SUPPLIES
ISO/IEC 27001:2022	Annex A 8.20 networks security, 8.22 segregation of networks, 8.24 use of cryptography	Mutual TLS on every hop, admission narrowed to observed and granted callers
ISO/IEC 27001:2022	Annex A 8.15 logging, 8.16 monitoring activities	Every connection attributed, and a posture record nobody edits by hand
EU NIS2 Directive	Article 21: cryptography and encryption, access control, and asset management	TLS 1.3 with post-quantum key exchange, Strict Identity, and an inventory of every workload
NIST SP 800-207	All communication secured regardless of location, and access granted per session	Mutual TLS between every pair of workloads, and identity checked on every connection
NIST SP 800-207	Monitor the posture of every asset and collect data to improve it	The posture record, protocol drift reports, and the network map
United States FIPS 140-3	Validated cryptography	A data plane built on FIPS 140-3 validated cryptography, always on
United States CISA Zero Trust Maturity Model	Networks pillar: encrypted traffic and isolation, with automation	Strict by default, one-label onboarding and scheduled hardening
United States DoD Zero Trust Strategy	Network and environment, automation and orchestration, visibility and analytics pillars	Micro-segmentation by caller and port, automatic onboarding, and the Console views

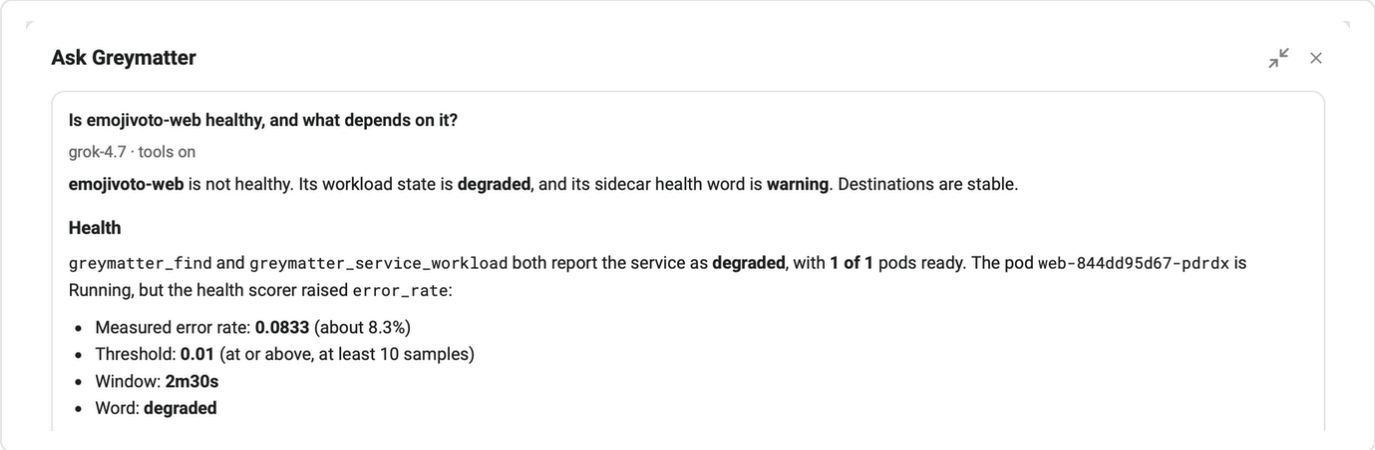
Using Greymatter does not make a system compliant with any framework. It supplies controls and evidence an assessor can test. National rows come last.

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

18 Ask Greymatter

Ask Greymatter answers questions about posture, traffic and dependencies, from the same data the Console shows. FIELDED

FIGURE 11 · 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 the mesh, it reads each service's posture, the flow ledger, the posture export worklist and what depends on each service.

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

“Which services are strict, and which are only observed?”
`Security` · each service's live posture

“If the orders service goes down, what breaks?”
`Traffic` · what depends on one service

“Which edges use post-quantum TLS, and which clients do not?”
`Security` · post-quantum use at each edge

“Which connections were denied across the mesh in the last hour, and from where?”
`Traffic` · the flow ledger

“Which projects are still waiting to be exported?”
`Changes` · the posture export worklist

“How do I let a service admit a new caller?”
`Docs` · the Greymatter documentation

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

Key takeaways

1

One label

Labeling a namespace enrolls it and onboards every workload in it, existing ones included.

2

Policy written once

A Global Onboarding Policy sets the rules and the floor. A Tenant Onboarding Policy refines them, never below.

3

Strict by default

STRICT_MESH refuses plaintext from the first workload. Every hop runs mutual TLS.

4

Learn, then seal

Greymatter admits the callers it observed and names who the seal will refuse before it lands.

5

Modern cryptography

TLS 1.3 and X25519MLKEM768 by default, on FIPS 140-3 cryptography that is always on.

6

Evidence, not claims

A posture record nobody can edit, an audit at every hop, and a map of what the wire did.

IN ONE SENTENCE

Greymatter makes zero trust the default state of every namespace: labeled once, learned automatically, sealed on schedule, and proven by its own record.

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

Greymatter.io

Label once. Learn automatically. Seal on schedule. Prove it.

LEARN MORE

greymatter.io

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

© 2026 Greymatter.io, Inc.