

AI agent egress control:

constrain the request and the role

13 October 2026

An agent that reads public websites should have a defined network boundary. Its tools should reach only the destinations needed for that work, and its execution identity should authorize only the operations the workload requires. Review both. A restricted role cannot prevent every unwanted connection; a destination filter cannot correct excessive permissions.

This guide separates the AgentCorruption research from current provider guidance, checks three MCP security records, and sets out a practical review for platform and security teams. It builds on [where an agent's tools live](#) by asking what those tools can reach.

What AgentCorruption establishes

Zenity's 8 October report describes credential retrieval through an exposed AgentCore agent, broad access to other agents and resources within the same account and region, and later platform hardening. These are researcher findings. Zenity says the described issues were addressed; the report is not evidence of a current customer breach. [1]

AWS's statement in the report calls the behaviour “documented and expected.” Its independent documentation explains that AgentCore uses the MicroVM Metadata Service (MMDS) to supply execution-role credentials inside the microVM. Any code or actor there can access those credentials, independently of the configured network mode. [1, 2]

The terminology matters. Zenity calls the path IMDS; AWS documents AgentCore's implementation as MMDS. EC2's Instance Metadata Service (IMDS) has related concepts, but its configuration guidance should not be copied uncritically to a managed runtime.

AWS also documents that, from 30 June 2026, runtime invocations require MMDSv2. Runtimes missing that configuration are rejected; existing sessions are unaffected. That clarifies the current requirement without proving that every historical runtime or execution role was automatically repaired. [3]

The defensible lesson is about reachable credentials and the permissions behind them. The evidence does not justify a claim of a microVM escape, universal cross-account compromise, or a collapse of the entire cloud security model.

Where an unwanted request can originate

Server-side request forgery (SSRF) occurs when an application is induced to contact an unintended destination. A model-selected URL is one input to that problem. Direct user instructions or malicious material in retrieved content may influence tool use; the latter is indirect prompt injection. Zenity's basic demonstration used a direct chat request. [4, 5]

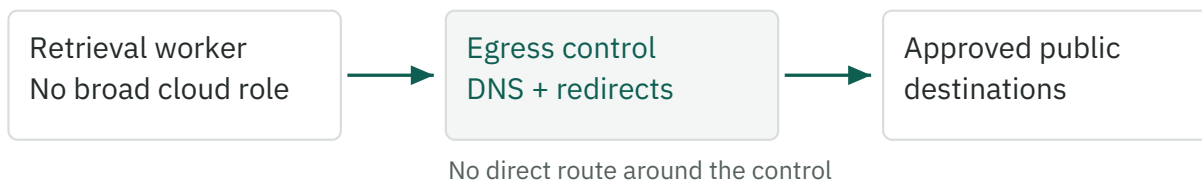
Model Context Protocol (MCP) integrations add more than one request origin. A fetch server connects from its own host. An MCP client may fetch OAuth metadata supplied by a remote server during authentication. That client path does not require prompt injection or a tool call. [6, 7]

Architecture recommendation: maintain a request-path inventory that names the initiating process, destination classes, redirect behaviour, DNS resolver, proxy, credentials and control owner. Include browser tools, HTTP-source connectors, authentication discovery and shell environments.

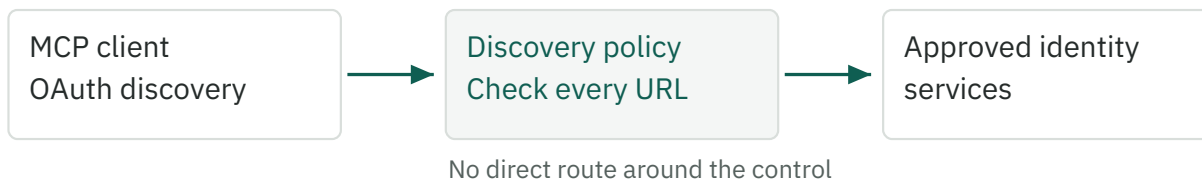
Separate reachability from authority

Illustrative design • Each request path needs its own enforcement

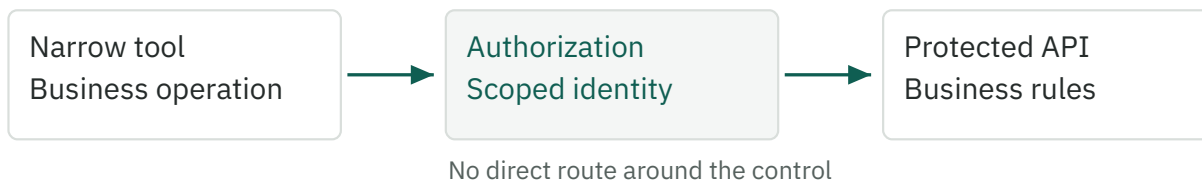
PUBLIC CONTENT



AUTHENTICATION DISCOVERY



BUSINESS OPERATIONS



Local metadata is a separate boundary. An external proxy may not see it.

Illustrative design, not an AWS service topology. Public retrieval and privileged business operations have separate controls. Local metadata access must be assessed within the actual runtime.

What the verified fixes cover

MCP fetch. CVE-2026-104120 exists. Its retrieved record identifies fetch-related SSRF, but still says the fix awaits acceptance. Later primary records establish that PR #5033 merged on 5 October and the package 2026.10.10 was uploaded on 11 October. Treat the older CVE wording as a chronology discrepancy, not proof that no fix shipped. [8, 9, 10]

The published guard checks non-public destinations on the tool and prompt paths, including redirects. Its documentation explicitly leaves a DNS check-to-connection window and excludes hostnames that only a proxy can resolve. The release therefore supports a bounded fix claim, not “SSRF eliminated.” [9, 10]

Google MCP Toolbox. CVE-2026-14540 covers versions 0.3.0 through 1.4.0 and describes missing redirect restrictions and target-IP validation in the generic HTTP source/tool. The maintainer's v1.5.0 release includes the SSRF guard. The release date is 18 June; the CVE was published on 31 July. [11, 12]

rmcp Rust SDK. GHSA-c9xm-49cp-xcr9 identifies OAuth metadata SSRF below 2.0.0 and lists 2.0.0 as patched. The repository advisory date is 29 June; the GitHub Advisory Database publication is 2 October. [7]

Use maintained releases containing the relevant fixes, pin reviewed dependencies and confirm the running artifact. A patch in one library does not establish coverage for every network path in an application.

Enforce the destination when the connection is made

For a public retrieval component, begin with no access to private, loopback, link-local or other reserved destinations. Permit only documented exceptions needed by its function. An internal business connector needs a narrower allowlist for its intended service, together with authorization at that service. OWASP distinguishes these known-destination and general-outbound cases and recommends application and network controls. [13]

Check scheme, host, port and resolved addresses using maintained parsers. Handle IPv4 and IPv6. Disable automatic redirects where practical; otherwise apply the same destination policy at every hop. Bind the address decision to the connection so that a second DNS resolution cannot silently change the destination. Preserve normal TLS hostname verification. This is an engineering requirement for the transport or enforcing proxy, not a prompt instruction. [6, 13]

MCP guidance says server-deployed clients **MUST** consider SSRF and implement appropriate mitigations for OAuth URLs. Its specific private-IP blocking, HTTPS and redirect recommendations use **SHOULD**. Preserve that distinction when describing the protocol. [6]

A proxy is effective only for traffic it mediates. Verify direct sockets, alternative libraries and shell commands cannot bypass the intended route. A local credential endpoint may never traverse that external proxy. Separate an untrusted fetch component from privileged execution where the platform permits it; assess the local boundary explicitly where it does not.

The fetch server's `--allow-private-ips` flag belongs to host configuration rather than the fetch tool's arguments. Protect that configuration. An agent with permission to rewrite its own launch settings or run unrestricted code can undermine an otherwise useful restriction. [9]

Preserve metadata protection and limit identity

EC2 IMDSv2 requires a session token and can be made mandatory. This helps defend against request paths that cannot perform the necessary method/header exchange. It does not decide whether code already inside a workload should be trusted with its role. A tool capable of arbitrary local code or requests needs a stronger boundary than token mechanics alone.

[14; architectural analysis]

For EC2, AWS documents local firewall controls that can restrict metadata access by process identity. Preserve required SDK and operational access. Its hop-limit guidance also needs context: a limit of one can prevent a container receiving a response, and AWS discusses reconfiguration or a limit of two. Do not prescribe one as a universal setting. [15, 16]

For AgentCore, use its own MMDS configuration and role guidance. AWS warns against using CLI-generated development policies in production. Define custom policies for required actions and resource ARNs. Review existing roles directly rather than assuming a newer deployment default changed them. [17]

Prefer a narrow business operation to generic fetch or shell access. Keep the caller's business authorization at the destination, and do not expose credentials in model-visible results. Short-lived credentials still carry their granted permissions while valid. [18; architectural recommendation]

This extends the distinction in [runtime authorization](#): access to a service does not authorize every proposed action. It also connects to [credential exposure in tool output](#). Verify separately whether sensitive output enters model context, is forwarded to a provider, or is retained in logs; one observation does not establish all three.

Evidence to collect in an architecture review

The following are proposed review outputs, not claims about an existing deployment.

1. **Request map.** Name every process that opens a connection, including OAuth discovery, and identify its owner and execution identity.
2. **Policy proof.** Record allowed destinations and denied classes. Demonstrate the effective connection policy with isolated fixtures for redirects, changing DNS answers, IPv6 and proxy-side resolution.
3. **Bypass review.** Check alternate request libraries, shell access, configuration permissions and local metadata reach. Use inert fixtures; do not retrieve live credentials for a demonstration.

4. **Role review.** Inspect effective actions and resources, cross-agent access, secret retrieval and memory-writing permissions. Explain each permission in terms of a required operation.
5. **Telemetry proof.** Correlate agent, session, tool, destination, resolved address, policy decision and request ID. Redact credentials, authorization headers and sensitive query values.
6. **Maintenance evidence.** Record running package versions, artifact digests, approved exceptions, owner and review date. Retest when the runtime or request path changes.

Check the monitoring surface before promising coverage. AWS excludes instance-metadata traffic to and from 169.254.169.254 from VPC Flow Logs. Use appropriate tool, process or enforcing-proxy records, and establish their coverage in the actual deployment. [19]

An architecture review should leave the team able to state which component may make which request, under which identity, and where that decision is enforced.

[TrustCyber architecture advisory](#) can help review agent tool egress, metadata exposure and execution-role scope.

References

Retrieved 13 October 2026. Researcher claims remain attributed. Living documentation may change.

1. Zenity: AgentCorruption overview. 8 October 2026.

<https://labs.zenity.io/post/agentcorruption-how-a-single-prompt-collapsed-the-entire-cloud-security-model>

2. AWS: Credentials management in AgentCore. Living documentation.

<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/security-credentials-management.html>

3. AWS: Troubleshoot AgentCore Runtime. Living documentation.

<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/runtime-troubleshooting.html>

4. OWASP: LLM01:2025 Prompt Injection. 2025 edition.

<https://genai.owasp.org/llmrisk/llm01-prompt-injection/>

5. Zenity: Initial IMDS Access. 8 October 2026.

<https://labs.zenity.io/post/agentcorruption-initial-imds-access>

6. MCP: Security Best Practices, versioned route. Version route 2026-07-28; retrieved 13 October 2026.

https://modelcontextprotocol.io/docs/2026-07-28/tutorials/security/security_best_practices

References, continued

- 7. GitHub advisory: GHSA-c9xm-49cp-xcr9.** Repository publication 29 June 2026; database publication/review/update 2 October 2026.
<https://github.com/advisories/GHSA-c9xm-49cp-xcr9>
- 8. CVE-2026-104120: CVE Program record.** Published and updated 2 October 2026 in retrieved record.
<https://raw.githubusercontent.com/CVEProject/cvelistV5/main/cves/2026/104xxx/CVE-2026-104120.json>
- 9. MCP servers: PR #5033.** Merged 5 October 2026.
<https://github.com/modelcontextprotocol/servers/pull/5033>
- 10. PyPI: mcp-server-fetch 2026.10.10.** Uploaded 11 October 2026.
<https://pypi.org/project/mcp-server-fetch/2026.10.10/>
- 11. CVE-2026-14540: CVE Program record.** Published 31 July 2026; updated 31 July 2026.
<https://raw.githubusercontent.com/CVEProject/cvelistV5/main/cves/2026/14xxx/CVE-2026-14540.json>
- 12. Google MCP Toolbox: v1.5.0 release.** 18 June 2026.
<https://github.com/googleapis/mcp-toolbox/releases/tag/v1.5.0>
- 13. OWASP: SSRF Prevention Cheat Sheet.** Living guidance.
https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html
- 14. AWS: How IMDSv2 works.** Living documentation.
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-instance-metadata-service.html>
- 15. AWS: Limit access to IMDS.** Living documentation.
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-metadata-limiting-access.html>
- 16. AWS: Metadata access considerations.** Living documentation.
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-data-retrieval.html>
- 17. AWS: AgentCore Runtime security best practices.** Living documentation.
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/runtime-security-best-practices.html>
- 18. OWASP: LLM06:2025 Excessive Agency.** 2025 edition.
<https://genai.owasp.org/llmrisk/llm062025-excessive-agency/>
- 19. AWS: VPC Flow Log limitations.** Living documentation.
<https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs-limitations.html>